Andreas Schilling (Hrsg.)

Festschrift

zum 60. Geburtstag von

Wolfgang Straßer

ISSN 0946-3852 WSI-2001-20

ISSN 0946-3852

WSI-2001-20

Wilhelm-Schickard-Institut für Informatik
Graphisch-Interaktive Systeme
Auf der Morgenstelle 10/C9
D-72076 Tübingen
Tel.: +49 7071 29-75462
Fax: +49 7071 29-5466

URL: http://www.gris.uni-tuebingen.de/

Contents

Vorwort 8

1 Reconstructing from Partial Information **9**
V. Blanz, T. Vetter
1.1 Introduction 10
1.2 Representation of Class-Specific Knowledge 12
1.3 Incomplete Measurements 13
1.4 Prior Probability versus Matching quality 15
1.4.1 Bayesian Approach to Reconstruction 15
1.4.2 Combined Cost Function 16
1.5 Special case $\mathbf{L} = id_n$ 17
1.6 Application to Face Data 18
1.6.1 Reconstruction of Novel Faces 20
1.6.2 Correct Reconstruction of Training Faces 22
1.6.3 Noisy Feature Point Coordinates 23
1.6.4 Robustness with respect to $\mathbf{L}$ 23
1.6.5 Results on Real Images 24
1.7 Conclusion 25
1.7.1 Acknowledgements 26

2 Object Extraction from Volume Data **27**
H.-H. Ehricke
2.1 Introduction 28
2.2 System architecture 30
2.3 Low-level image processing 32
2.4 Control system 32
2.5 Results 33

2.6 Summary and conclusion 34

3 History between the Realities **39**
L. M. Encarnação, O. Bimber, M. Billinghurst
3.1 Introduction 40
3.2 Motivation 40
3.3 Technological Approaches 41
3.3.1 Display technologies for Cultural Heritage Exhibits – The Virtual Showcase 43
3.3.2 Interaction Technologies for Cultural Heritage Exhibits 45
3.4 Discussion 51
3.5 Case Study: Treasures from a Lost Civilization 52
3.5.1 The Experience 53
3.5.2 The User Response 53
3.6 Conclusions 54

4 Scientific Visualization of Large Datasets **57**
T. Ertl
4.1 Introduction 58
4.2 Data compression 60
4.3 Adaptive and progressive mapping algorithms 61
4.4 Exploiting advanced rasterization hardware 63
4.5 Conclusions 64

5 Connectivity Coding **67**
S. Gumhold
5.1 Introduction 68
5.2 Edge Breaker Coding 69
5.3 Constraints 73
5.4 Conditional Unities 74
5.5 The State Machine 77
5.6 Numerical Solution 83
5.7 Results 84
5.8 Conclusion 87

6 Weber et al., Real-Time Fluid Animation **91**
R. Klein, T. May, S. Schneider, A. Weber
6.1 Introduction 92

6.2 The Sequential Algorithm 93
6.2.1 Simulation program 93
6.2.2 Volume renderer 96
6.3 Parallel Architecture 96
6.3.1 Communication and synchronization between renderer and solver components 97
6.3.2 Parallelization of simulation program 97
6.3.3 Parallelization of the projection step 99
6.3.4 Using large step sizes and interpolation 101
6.4 Results 102
6.4.1 Performance measurements 102
6.4.2 Animations 103
6.5 Conclusion 103

7 Efficient Multiresolution Models **109**
R. Klein, A. Schilling
7.1 Introduction and previous work 110
7.1.1 Handling huge textures 111
7.1.2 Dealing with geometry 112
7.2 The multiresolution model 114
7.2.1 The simplification algorithm 115
7.2.2 Storing the model 119
7.2.3 Storing and transmitting the geometry data 122
7.3 Rendering the model 124
7.3.1 The geometric and the screen space errors 124
7.3.2 The extraction algorithm 124
7.3.3 Frame to frame coherency 126
7.A Coding the transitions 126

8 High Quality 3D Models **131**
H. P. A. Lensch, M. Goesele, H.-P. Seidel
8.1 Introduction 132
8.2 3D Object Acquisition Pipeline 133
8.3 Image-Based Acquisition Techniques 134
8.3.1 Photographic Equipment 134
8.3.2 Lighting Equipment 135
8.3.3 Camera Calibration 136
8.4 Appearance Acquisition 139

8.4.1 Reflection Properties . . . 140
8.4.2 Measuring Reflection Properties . . . 141
8.4.3 Measuring Spatially Varying BRDFs . . . 142
8.4.4 Normal Maps . . . 145
8.5 Acquisition of 3D Geometry . . . 146
8.6 Registration of Geometry and Texture Data . . . 148
8.6.1 Manual Registration . . . 149
8.6.2 Automatic Registration . . . 149
8.6.3 Texture Preparation . . . 151
8.7 Interactive Display . . . 153
8.7.1 Rendering with Arbitrary BRDFs . . . 153
8.7.2 Rendering with Normal Maps . . . 154
8.7.3 Spatially Varying BRDFs . . . 155
8.8 Examples . . . 155
8.9 Conclusion . . . 158
8.10 Acknowledgements . . . 160

9 Fast Accurate Integration 167
F. Reck, G. Greiner
9.1 Introduction . . . 168
9.2 Particle Tracing in Tetrahedral Grids . . . 169
9.2.1 Numerical Integration Methods . . . 169
9.2.2 Local Exact Integration . . . 171
9.2.3 Comparison . . . 173
9.3 Modification of Nielson's Approach . . . 174
9.3.1 Overview of the local exact method . . . 174
9.3.2 Data structure . . . 174
9.3.3 Preprocessing . . . 176
9.3.4 Calculation of path lines in world coordinates . . . 177
9.4 Results . . . 178
9.4.1 Regular vs. non-regular cells . . . 178
9.4.2 Precision . . . 179
9.4.3 Memory Requirement . . . 182
9.4.4 Time behavior . . . 182
9.4.5 Advantages . . . 183
9.5 Acknowledgements . . . 183

10 Notes on Sampling **185**
A. Schilling
10.1 Introduction 186
10.2 Discretizing Continuous Signals 186
10.2.1 Sampling and Reconstruction 187
10.3 Application Areas 188
10.3.1 Image Synthesis 188
10.3.2 Natural Images 190
10.3.3 Image Sequences 191
10.3.4 Voxel Data 191
10.3.5 Geometry 191
10.4 Errors 192
10.4.1 Aliasing Errors — Sampling Theorems 192
10.4.2 Errors Resulting from Filtering 194
10.4.3 Errors due to Reconstruction 195
10.4.4 Errors Resulting from Non-Linearities 197
10.4.5 Errors Caused by Perception 198
10.5 Antialiasing — Optimal Approximation 198
10.6 A Note on Mip-Mapping 200
10.7 A Note on Creating Geometrical Multiresolution Models . . . 203

11 Physically-based techniques **209**
F. Wagner, D. Jackèl
11.1 Introduction 209
11.2 Motion control 210
11.2.1 Controller 211
11.2.2 Optimization techniques 212
11.2.3 Constraints 213
11.3 Dynamics formulations 214
11.3.1 Lagrange multiplier formulation (LMF) 214
11.3.2 Other approaches 216
11.3.3 Comparison and valuation of the LMF 218
11.4 An animation system based on the LMF 219
11.5 Conclusion 222

Vorwort

Die vorliegende Festschrift ist Prof. Dr.-Ing. Dr.-Ing. E.h. Wolfgang Straßer zu seinem 60. Geburtstag gewidmet. Eine Reihe von Wissenschaftlern auf dem Gebiet der Computergraphik, die alle aus der „Tübinger Schule“ stammen, haben — zum Teil zusammen mit ihren Schülern — Aufsätze zu dieser Schrift beigetragen.

Die Beiträge reichen von der Objektrekonstruktion aus Bildmerkmalen über die physikalische Simulation bis hin zum Rendering und der Visualisierung, vom theoretisch ausgerichteten Aufsatz bis zur praktischen gegenwärtigen und zukünftigen Anwendung. Diese thematische Buntheit verdeutlicht auf anschauliche Weise die Breite und Vielfalt der Wissenschaft von der Computergraphik, wie sie am Lehrstuhl Straßer in Tübingen betrieben wird.

Schon allein an der Tatsache, daß im Bereich der Computergraphik zehn Professoren an Universitäten und Fachhochschulen aus Tübingen kommen, zeigt sich der prägende Einfluß Professor Straßers auf die Computergraphiklandschaft in Deutschland. Daß sich darunter mehrere Physiker und Mathematiker befinden, die in Tübingen für dieses Fach gewonnen werden konnten, ist vor allem seinem Engagement und seiner Ausstrahlung zu verdanken.

Neben der Hochachtung vor den wissenschaftlichen Leistungen von Professor Straßer hat sicherlich seine Persönlichkeit einen entscheidenden Anteil an der spontanten Bereischaft der Autoren, zu dieser Festschrift beizutragen. Mit außergewöhnlich großem persönlichen Einsatz fördert er Studenten, Doktoranden und Habilitanden, vermittelt aus seinen reichen internationalen Beziehungen Forschungskontakte und schafft so außerordentlich gute Voraussetzungen für selbständige wissenschafliche Arbeit.

Die Autoren wollen mit ihrem Beitrag Wolfgang Straßer eine Freude bereiten und verbinden mit ihrem Dank den Wunsch, auch weiterhin an seinem fachlich wie menschlich reichen und bereichernden Wirken teilhaben zu dürfen.

Eine Auswahl der Beiträge wird durch Vermittlung von Prof. Hans-Peter Seidel in der Zeitschrift IT+TI erscheinen, zusammen mit weiteren Aufsätzen, die aufgrund des engen zeitlichen Rahmens hier nicht mehr aufgenommen werden konnten.

Andreas Schilling November 2001

Chapter 1

Reconstructing the Complete 3D Shape of Faces from Partial Information

Volker Blanz, Thomas Vetter[*]

Based on the assumption that a class of objects or data can be represented as a vector space spanned by a set of examples, we present a general method to estimate vector components of a novel vector, given only a subset of its dimensions.

We apply this method to recover 3D shape of human faces from 2D image positions of a small number of feature points. The application demonstrates two aspects of the estimation of novel vector components: (1) From 2D image positions, we estimate 3D coordinates, and (2) from a small set of points, we obtain vertex positions of a high-resolution surface mesh. We provide an evaluation of the technique on laser scans of faces, and present an example of 3D shape reconstruction from a photograph.

Our technique involves a tradeoff between reconstruction of the given

[*] Graphische Datenverarbeitung, Universität Freiburg, Germany. E-Mail: {volker|vetter}@informatik.uni-freiburg.de

measurements, and plausibility of the result. This is achieved in a Bayesian approach, and with a statistical analysis of the examples.

1.1 Introduction

Arguments by analogy are a useful mode of reasoning if we lack sufficient information about a problem for a rigirous conclusion, but are provided with many instances of solutions for similar settings. In this paper, we address the problem of estimating the components of a vector, given only some of the components' values. More generally, the input may be the result of any linear mapping to a lower dimensional space. The prior knowledge that helps to solve this ill-posed problem is represented by a set of examples of vectors, and the assumption that any novel solution is in the span of these examples. Moreover, we exploit the statistical properties of the examples to obtain an estimate of prior probability. The correlation of vector components within the set of examples is the core property that makes an estimate of unknown vector components possible.

As an example of a vector space of objects, we apply our method to the geometry of faces. The morphable face model approach [BV99] provides a representation of facial shapes in terms of shape vectors, such that any linear combination of vectors describes a realistic face. Shape vectors are defined by concatenating the x, y, and z coordinates of a large set of surface points to a single, high-dimensional vector. The technique for selecting these surface points on individual faces ensures that each component of the shape vector refers to corresponding points on all faces, such as the tip of the nose.

In this paper, we estimate full 3D structure of a face from 2D image positions of a subset of the morphable model's vertices. Image positions are taken from a front view of the face, and with orthographic projection. However, the system can also be applied to any other viewing direction, or a combination of views. Restricted to linear mappings of the original data, the system cannot handle perspective projection from close viewpoints. For larger distances, the difference between perspective and orthographic projection decreases, and our technique provides realistic results.

The morphable face model has previously been used to estimate 3D shape from a single image [BV99]. Comparing color values of the image with those obtained from the model, this system iteratively matches the morphable model to the image. Similar to the approach presented here, the system relies on the

vector space structure of faces for estimating 3D structure. However, it also exploits shading information from the image. Matching the entire facial surface to the image, the result recovers many facial details. In contrast, the method presented here relies only on a relatively small set of feature points provided by the user. However, the matching problem solved here is computationally much simpler, and can be solved in a single step in a robust way. Therefore, the algorithm is considerably faster and may be applied in interactive tools for face reconstruction.

We extend and generalize a method that has been applied to estimate dense optic flow fields in image data [HBVL00], using a data set of flow vectors obtained by a 2D projection of a 3D morphable face model. The modification presented in this paper makes the system more robust, which proves to be crucial to achieve high overall quality of the estimate.

The problem addressed in this study is related to the statistical problem of regression. In regression, a set of measurements (x_i, y_i) of a random variable y for different values of the known parameter x is used to estimate the expectation value $y(x)$ at any x. Regression techniques select a function y from a family of functions, which can be linear mappings, polynomials, or any other function space. If the capacity of the function space is too large, some methods produce overfitting effects (see [DHS01]): the function fits the measurements precisely, but varies drastically in between, rather than being smooth. The desired generalization of $y(x)$ to novel values of x tends to be poor.

As we demonstrate in Section 1.6, a similar effect may occur here, if the low-dimensional input vector is subject to noise or other sources of error, or if the desired solution cannot be entirely captured by the model.

To overcome the problem of overfitting, most regression techniques impose a smoothness constraint on the solution, or restrict the family of functions [Vap95, DHS01]. In our approach, we restrict solutions to the span of a set of examples, and impose an additional penalty on solutions far from the observed average. The result will be a tradeoff that is both plausible a priori, and still fits the given measurements well.

In the following section, we give a definition of object classes in terms of a probabilistic criterion for class membership. Section 1.3 presents a direct approach to estimating vector components from sparse data. Section 1.4 derives a framework to avoid overfitting and accomodate noisy measurements. Section 1.5 discusses a special case that relates the theory to a straightforward projection into the span of examples. In Section 1.6, we present results obtained with 3D models of faces.

1.2 Representation of Class-Specific Knowledge

We assume that the examples of class elements

$$\mathbf{v}_i \in I\!R^n, \qquad i = 1, \ldots, m \tag{1}$$

are given in a vector space representation such that linear combinations

$$\mathbf{v} = \sum_{i=1}^{m} a_i \mathbf{v}_i \tag{2}$$

describe new elements of the class. However, the coefficients of the linear combinations must be restricted by additional conditions to ensure realistic results.[2]

An estimate of the prior probability of vectors within the span of examples can be obtained by a Principal Component Analysis (PCA, see [Hay98]). The original data are centered around the origin by subtracting the arithmetic mean

$$\mathbf{x}_i = \mathbf{v}_i - \overline{\mathbf{v}}, \qquad \overline{\mathbf{v}} = \frac{1}{m} \sum_{i=1}^{m} \mathbf{v}_i, \tag{3}$$

and concatenated to a data matrix

$$\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m) \in I\!R^{n \times m}. \tag{4}$$

The covariance matrix of the data set is given by

$$\mathbf{C} = \frac{1}{m} \mathbf{X}\mathbf{X}^T = \frac{1}{m} \sum_{j=1}^{m} \mathbf{x}_j \mathbf{x}_j^T \ \in I\!R^{n \times n}, \tag{5}$$

PCA is based on a diagonalization of the covariance matrix,

$$\mathbf{C} = \mathbf{S} \cdot diag(\sigma_i^2) \cdot \mathbf{S}^T. \tag{6}$$

Since $\mathbf{C}$ is symmetrical, the columns $\mathbf{s}_i$ of $\mathbf{S} = (\mathbf{s}_1, \mathbf{s}_2, \ldots)$ form an orthogonal set of eigenvectors. $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_m$ are the standard deviations within

[2] Coefficients might be constrained to the convex hull by $a_i \in [0, 1]$ and $\sum_{i=1}^{m} a_i = 1$. The first constraint is replaced here by a probabilistic measure. The second is enforced implicitly by forming linear combinations relative to $\overline{\mathbf{v}}$: Any linear combination $\mathbf{v} = \sum_{i=1}^{m} b_i \mathbf{x}_i + \overline{\mathbf{v}}$ can be shown to satisfy $\sum_{i=1}^{m} a_i = 1$ in terms of (2) and (3).

the data along each eigenvector $\mathbf{s}_i$. The diagonalization of $\mathbf{C}$ can be calculated by a Singular Value Decomposition (SVD, [PTVF92]) of $\mathbf{X}$.

Having subtracted the arithmetic mean, the m vectors $\mathbf{x}_i$ are linearly dependent, so their span is at most $m' = (m-1)$ dimensional, and the rank of $\mathbf{X}$ and $\mathbf{C}$ is at most m'. Therefore, $\sigma_m = 0$, and $\mathbf{s}_m$ is irrelevant.

In the following, we use the eigenvectors as a basis,

$$\mathbf{x} = \sum_{i=1}^{m'} c_i \sigma_i \mathbf{s}_i = \mathbf{S} \cdot diag(\sigma_i)\mathbf{c}. \tag{7}$$

An important property of PCA is that variations along the eigenvectors are uncorrelated within the set of examples. Assuming a normal distribution in each of the directions, the probability density at $\mathbf{x}$ is

$$p(\mathbf{x}) = \prod_{i=1}^{m'} \frac{1}{\sqrt{2\pi}\sigma_i} e^{-\frac{1}{2\sigma_i^2}\langle \mathbf{s}_i, \mathbf{x}\rangle^2} = \prod_{i=1}^{m'} \frac{1}{\sqrt{2\pi}\sigma_i} e^{-\frac{1}{2}c_i^2} \tag{8}$$

$$= \frac{1}{(2\pi)^{m'/2}\prod_i \sigma_i} e^{-\frac{1}{2}\|\mathbf{c}\|^2}. \tag{9}$$

The probability density for $\mathbf{c}$ can be rescaled to

$$p(\mathbf{c}) = \nu_c \cdot e^{-\frac{1}{2}\|\mathbf{c}\|^2}, \qquad \nu_c = (2\pi)^{-m/2}. \tag{10}$$

The exponent $\|\mathbf{c}\|^2$ is often referred to as Mahalanobis Distance.

1.3 Incomplete Measurements

Given a measurement $\mathbf{r} \in I\!R^l$, $l < n$, we would like to find the full vector $\mathbf{x} \in I\!R^n$ such that

$$\mathbf{r} = \mathbf{L}\mathbf{x} \tag{11}$$

with a mapping $\mathbf{L} : I\!R^n \mapsto I\!R^l$. $\mathbf{L}$ can be any linear transformation, and does not need to be a projection.

If $\mathbf{L}$ is not a one-to-one mapping, the solution (11) is not uniquely defined. Therefore, we restrict the admissible solutions to the span of $\mathbf{x}_i$. As we cannot expect to find a linear combination of the examples that solves (11) exactly, we compute a vector $\mathbf{x}$ that minimizes

$$E(\mathbf{x}) = \|\mathbf{L}\mathbf{x} - \mathbf{r}\|^2. \tag{12}$$

Let $\mathbf{q}_i = \sigma_i \mathbf{L}\mathbf{s}_i \in I\!R^l$ be the reduced versions of the scaled eigenvectors, and

$$\mathbf{Q} = (\mathbf{q}_1, \mathbf{q}_2, ...) = \mathbf{L}\mathbf{S} \cdot diag(\sigma_i) \in I\!R^{l \times m'}. \tag{13}$$

In terms of model parameters c_i, Equation (12) is

$$E(\mathbf{c}) = \|\mathbf{L}\sum_i c_i \sigma_i \mathbf{s}_i - \mathbf{r}\|^2 = \|\sum_i c_i \mathbf{q}_i - \mathbf{r}\|^2 \tag{14}$$

$$= \|\mathbf{Q}\mathbf{c} - \mathbf{r}\|^2. \tag{15}$$

The optimum can be found by a Singular Value Decomposition [PTVF92]

$$\mathbf{Q} = \mathbf{U}\mathbf{W}\mathbf{V}^T \tag{16}$$

with a diagonal matrix $\mathbf{W} = diag(w_i)$, and $\mathbf{V}^T\mathbf{V} = \mathbf{V}\mathbf{V}^T = id_{m'}$.

The pseudoinverse (see [Sto99]) of $\mathbf{Q}$ is

$$\mathbf{Q}^+ = \mathbf{V}\mathbf{W}^+\mathbf{U}^T, \quad \mathbf{W}^+ = diag\begin{pmatrix} w_i^{-1} & \text{if } w_i \neq 0 \\ 0 & \text{otherwise} \end{pmatrix}. \tag{17}$$

To avoid numerical problems, the condition $w_i = 0$ may be replaced by a threshold. The minimum of (15) is

$$\mathbf{c} = \mathbf{Q}^+\mathbf{r}, \tag{18}$$

which is optimal in two respects [Sto99]:

1. $\mathbf{c}$ minimizes E, so for all $\mathbf{c}'$, $\quad E(\mathbf{c}') \geq E(\mathbf{c})$.

2. Among the set of solutions $\{\mathbf{c}' | E(\mathbf{c}') = E(\mathbf{c})\}$, $\mathbf{c}$ has minimum norm $\|\mathbf{c}\|$ and thus maximum prior probability (Equation 10).

By Equation (3) and (7), $\mathbf{c}$ is mapped to $I\!R^n$:

$$\mathbf{v} = \mathbf{S} \cdot diag(\sigma_i)\mathbf{c} + \overline{\mathbf{v}}. \tag{19}$$

For solving Equation (11), it might seem more straightforward to compute the pseudoinverse of $\mathbf{L}$ and set $\mathbf{x} = \mathbf{L}^+\mathbf{r}$. However, among vectors with equal error $\|\mathbf{L}\mathbf{x} - \mathbf{r}\|$, this method would return the solution with minimum $\|\mathbf{x}\|$ rather than minimum $\|\mathbf{c}\|$. Vector components x_i that do not affect $\mathbf{L}\mathbf{x}$ would be zero, and the result would not be in the span of the examples.

1.4 Prior Probability versus Matching quality

The previous solution will always ensure that E is minimized, and in particular that $E = 0$ whenever this is possible. Prior probability is only considered within solutions of equal $E(\mathbf{c})$.

However, it may well be that the measurement $\mathbf{r}$ cannot be fully accounted for by an element $\mathbf{v}$ of the object class. First, $\mathbf{r}$ may be subject to noise or other sources of error, such as wrong assumptions on $\mathbf{L}$. Moreover, we cannot expect to cover the full range of the object class with the set of examples.

Therefore, minimizing $E(\mathbf{x}) = \|\mathbf{L}\mathbf{x} - \mathbf{r}\|^2$ may lead to model coefficients far from the average, and a heavily distorted vector $\mathbf{v}$. To avoid this overfitting, we propose a tradeoff between matching quality and prior probability of the solution. This tradeoff will be derived from a Bayesian approach in the following section.

1.4.1 Bayesian Approach to Reconstruction

For an element of the model that is defined by model parameters $\mathbf{c}$, a noiseless measurement would be

$$\mathbf{r}_{model} = \mathbf{L}\sum_i c_i \sigma_i \mathbf{s}_i = \sum_i c_i \mathbf{q}_i = \mathbf{Q}\mathbf{c} \tag{20}$$

We assume that each dimension j of the measured vector $\mathbf{r}$ is subject to uncorrelated Gaussian noise with a variance σ_N^2. Then, the likelihood of measuring $\mathbf{r} \in I\!R^l$ is given by

$$\begin{aligned} P(\mathbf{r}|\mathbf{r}_{model}) &= \prod_{j=1}^{l} P(r_j|r_{model,j}) && (21)\\ &= \prod_{j=1}^{l} \nu_N \cdot e^{-\frac{1}{2\sigma_N^2}(r_{model,j}-r_j)^2} && (22)\\ &= \nu_N^l \cdot e^{-\frac{1}{2\sigma_N^2}\sum_j (r_{model,j}-r_j)^2} && (23)\\ &= \nu_N^l \cdot e^{-\frac{1}{2\sigma_N^2}\|\mathbf{r}_{model}-\mathbf{r}\|^2} && (24) \end{aligned}$$

with a normalization factor ν_N. In terms of the model parameters $\mathbf{c}$, the likelihood is

$$P(\mathbf{r}|\mathbf{c}) = \nu_N^l \cdot e^{-\frac{1}{2\sigma_N^2}\|\mathbf{Q}\mathbf{c}-\mathbf{r}\|^2}. \tag{25}$$

Given an observed vector $\mathbf{r}$, we are looking for the estimate $\mathbf{c}$ with maximum probability. According to Bayes Rule [DHS01], this posterior probability is given by

$$P(\mathbf{c}|\mathbf{r}) = \nu \cdot P(\mathbf{r}|\mathbf{c}) \cdot p(\mathbf{c}). \tag{26}$$

with a constant factor $\nu = (\int P(\mathbf{r}|\mathbf{c}') \cdot p(\mathbf{c}')d\mathbf{c}')^{-1}$.

Substituting (10) and (25) yields

$$P(\mathbf{c}|\mathbf{r}) = \nu \cdot \nu_N^l \cdot \nu_c \cdot e^{-\frac{1}{2\sigma_N^2}\|\mathbf{Q}\mathbf{c}-\mathbf{r}\|^2} \cdot e^{-\frac{1}{2}\|\mathbf{c}\|^2}, \tag{27}$$

which is maximized if the cost function

$$E = -2 \cdot logP(\mathbf{c}|\mathbf{r}) = \frac{1}{\sigma_N^2}\|\mathbf{Q}\mathbf{c}-\mathbf{r}\|^2 + \|\mathbf{c}\|^2 + const. \tag{28}$$

is minimized.

1.4.2 Combined Cost Function

In this section, we show that the cost function (28) can be minimized in a single step. To simplify the calculation, we introduce a weight factor $\eta = \sigma_N^2 \geq 0$ and minimize

$$E = \|\mathbf{Q}\mathbf{c}-\mathbf{r}\|^2 + \eta \cdot \|\mathbf{c}\|^2. \tag{29}$$

This can be expanded to

$$E = \langle \mathbf{Q}\mathbf{c}, \mathbf{Q}\mathbf{c}\rangle - 2\langle \mathbf{Q}\mathbf{c}, \mathbf{r}\rangle + \|\mathbf{r}\|^2 + \eta \cdot \|\mathbf{c}\|^2 \tag{30}$$

$$E = \langle \mathbf{c}, \mathbf{Q}^T\mathbf{Q}\mathbf{c}\rangle - 2\langle \mathbf{c}, \mathbf{Q}^T\mathbf{r}\rangle + \|\mathbf{r}\|^2 + \eta \cdot \|\mathbf{c}\|^2 \tag{31}$$

In the optimum,

$$0 = \nabla E = 2\mathbf{Q}^T\mathbf{Q}\mathbf{c} - 2\mathbf{Q}^T\mathbf{r} + 2\eta\mathbf{c}, \tag{32}$$

so

$$\mathbf{Q}^T\mathbf{Q}\mathbf{c} + \eta\mathbf{c} = \mathbf{Q}^T\mathbf{r}. \tag{33}$$

Singular Value Decomposition $\mathbf{Q} = \mathbf{U}\mathbf{W}\mathbf{V}^T$ yields[3]

$$\mathbf{Q}^T\mathbf{Q} = \mathbf{V}\mathbf{W}\mathbf{U}^T\mathbf{U}\mathbf{W}\mathbf{V}^T = \mathbf{V}\mathbf{W}^2\mathbf{V}^T. \tag{34}$$

[3]The matrix $\mathbf{U} \in I\!R^{l \times m'}$ computed by SVD has the following property [PTVF92]: If $m' \leq l$, $\mathbf{U}^T\mathbf{U} = id_{m'}$. If $m' > l$, only the first m' columns of $\mathbf{U}$ are orthogonal, while the others are 0, so $\mathbf{U}^T\mathbf{U}$ is not the full identity matrix. However, $w_i = 0$ for $i > m'$, so $\mathbf{W}\mathbf{U}^T\mathbf{U}\mathbf{W} = \mathbf{W}^2$ still holds.

From (33), we obtain

$$\mathbf{V}\mathbf{W}^2\mathbf{V}^T\mathbf{c} + \eta\mathbf{c} = \mathbf{V}\mathbf{W}\mathbf{U}^T\mathbf{r}. \tag{35}$$

Multiplying by $\mathbf{V}^T$, this can be solved for $\mathbf{c}$:[4]

$$\mathbf{W}^2\mathbf{V}^T\mathbf{c} + \eta\mathbf{V}^T\mathbf{c} = \mathbf{W}\mathbf{U}^T\mathbf{r} \tag{36}$$

$$diag(w_i^2 + \eta) \cdot \mathbf{V}^T\mathbf{c} = \mathbf{W}\mathbf{U}^T\mathbf{r} \tag{37}$$

$$\mathbf{V}^T\mathbf{c} = diag(\frac{w_i}{w_i^2 + \eta})\mathbf{U}^T\mathbf{r} \tag{38}$$

$$\mathbf{c} = \mathbf{V}diag(\frac{w_i}{w_i^2 + \eta})\mathbf{U}^T\mathbf{r} \tag{39}$$

Note that in the special case $\eta = 0$, this equivalent to Equation (18).
The overall result is

$$\begin{aligned} \mathbf{x} &= \sum_i c_i\sigma_i\mathbf{s}_i = \mathbf{S}diag(\sigma_i)\mathbf{c} && (40)\\ &= \mathbf{S}diag(\sigma_i)\mathbf{V}diag(\frac{w_i}{w_i^2 + \eta})\mathbf{U}^T\mathbf{r}. && (41) \end{aligned}$$

and

$$\mathbf{v} = \mathbf{x} + \overline{\mathbf{v}}. \tag{42}$$

1.5 Special case $\mathbf{L} = id_n$

In some applications it may be desirable to find the closest element of the span of examples from a vector $\mathbf{x}$ that is entirely known, or to approximate a given element of the span by a more plausible solution. Both cases are covered by the previous results if we set $\mathbf{L} = id_n$.

If $\mathbf{L} = id_n$, the Singular Value Decomposition of $\mathbf{Q}$ is trivial

$$\mathbf{Q} = \mathbf{S} \cdot diag(\sigma_i) = \mathbf{U}\mathbf{W}\mathbf{V}^T \tag{43}$$

[4] If $(w_i^2 + \eta) = 0$, which only occurs if $w_i = \eta = 0$, we replace $\frac{w_i}{w_i^2+\eta}$ by 0, as we did for the pseudoinverse.

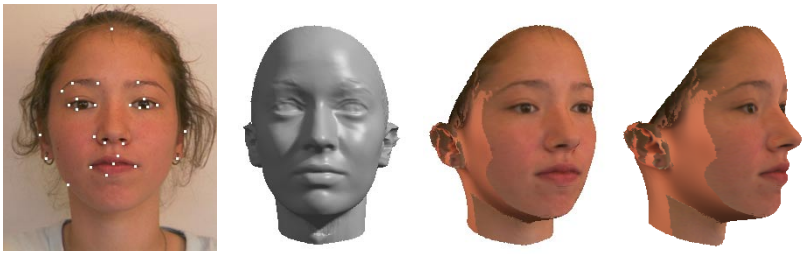


Figure 1.1: From 26 feature coordinates manually defined on the original image (top left), the system recovered the overall shape of the face (top right). With an additional texture extraction, color information can be transferred to the 3D model (bottom line) to generate new views. Vectors for translation and scaling (Equation 49) were added to the 99 principal components.

with the orthogonal matrix $\mathbf{U} = \mathbf{S}$, the diagonal matrix $\mathbf{W} = diag(\sigma_i)$, and $\mathbf{V} = id_{m'}$. Then, Equation (41) reduces to

$$\mathbf{x} = \mathbf{S} \cdot diag(\sigma_i) id_{m'} \cdot diag(\frac{\sigma_i}{\sigma_i^2 + \eta}) \mathbf{S}^T \mathbf{r} \tag{44}$$

$$= \mathbf{S} \cdot diag(\frac{1}{1 + \frac{\eta}{\sigma_i^2}}) \mathbf{S}^T \mathbf{r} \tag{45}$$

$$= \sum_i \frac{1}{1 + \frac{\eta}{\sigma_i^2}} \langle \mathbf{s}_i, \mathbf{r} \rangle \, \mathbf{s}_i. \tag{46}$$

The most relevant dimensions $\mathbf{s}_i$ with large standard deviation σ_i are affected less by η than those with small σ_i. In the special case $\eta = 0$, $\mathbf{x}$ is given by a simple projection

$$\mathbf{x} = \sum_i \langle \mathbf{s}_i, \mathbf{r} \rangle \, \mathbf{s}_i. \tag{47}$$

1.6 Application to Face Data

In the morphable face model [BV99], facial surface data that were recorded with a laser scanner are represented in shape vectors that combine x, y, and z

coordinates of all vertices:

$$\mathbf{v} = (x_1, y_1, z_1, \ldots x_p, y_p, z_p)^T \in I\!R^n,\ n = 3 \cdot p \tag{48}$$

Sampled at a spacing of less than 1mm, surface is represented by $p = 75972$ vertices. Linear combinations of shape vectors will only produce realistic novel faces if corresponding points, such as the tip of the nose, are represented by the same vector components across all individual shape vectors. This is achieved by establishing dense correspondence between different scans, and forming vectors $\mathbf{v}_i$ in a consistent way.

Along with shape, the morphable face model also represents texture. In this study, texture is not considered, and all images are rendered with the average texture. The method described in this paper could also be applied to texture vectors, filling in occluded regions of the face.

The database of 200 individual faces used in this study has been randomly split into a training set and a test set of $m = 100$ faces each. The training set provides the examples $\mathbf{v}_i$ that are available to the system. From these, we computed $m' = 99$ principal components which are used throughout the following evaluation. The test set provides data for performance assessment on novel faces.

From the vertices of the full model, we selected sets of $f = 17, 50$, or 1000 vertices (Figure 1.2). The smaller sets are formed by salient points such as the corners of the mouth, that can be identified in an image. The set of 1000 vertices was selected at random.

Computed by orthographic projection in a frontal orientation, the image plane coordinates of these feature points form the vectors $\mathbf{r} \in I\!R^l$, $l = 2 \cdot f$, that are used for evaluation.

Projection and orientation also define the mapping $\mathbf{L}$, which is assumed to be known. For real images, it is important that the system can automatically adapt at least to translation and scaling. This is achieved if vectors

$$\mathbf{s}_{tx} = (1, 0, 0, 1, 0, 0, \ldots)^T, \quad \mathbf{s}_{ty}, \quad \mathbf{s}_{tz}, \quad \text{and} \tag{49}$$

$$\mathbf{s}_s = \overline{\mathbf{v}} \tag{50}$$

are added to the principal components in $\mathbf{S}$.

The evaluation of the algorithm is based on the following quantities, which are averaged across all 100 training or test faces:

- $E_r = \|\mathbf{Q}\mathbf{c} - \mathbf{r}\|$, the image plane matching error for all feature points, measured in units of pixels in a 300x300 image.

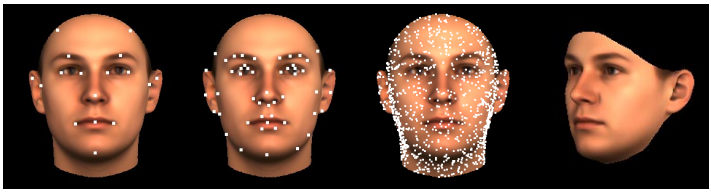


Figure 1.2: The first three images show the sets of 17, 50, and 1000 feature points used for evaluation. The image on the right illustrates where the error of 3D shape reconstruction was evaluated.

- $\|\mathbf{c}\|$, the Mahalanobis distance of the resulting face from the average.
- The per-vertex average of distances in 3D space between reconstruction and original, computed over the entire set of vertices:

$$E_{full} = \frac{1}{p} \| \begin{pmatrix} x_{p,reconst.} \\ y_{p,reconst.} \\ z_{p,reconst.} \end{pmatrix} - \begin{pmatrix} x_{p,orig.} \\ y_{p,orig.} \\ z_{p,orig.} \end{pmatrix} \| \tag{51}$$

  The neck and the top of the forehead are ignored in this measure, as shown in Figure 1.2.

For 99 principal components and 50 feature points, the computation takes 1.6 seconds on an SGI O_2 with R12000 processor. This includes forming $\mathbf{Q}$ from the much larger matrix $\mathbf{S}$, SVD of $\mathbf{Q}$, and computation of the full face model with 75972 vertices. Computation time depends mainly on the dimensions of $\mathbf{S}$.

1.6.1 Reconstruction of Novel Faces

In this Section, we examine how the technique performs on the test faces that are not included in the set of examples. The image coordinates of feature points provided to the algorithm are computed from the 3D vertex positions of the 3D faces. For $f = 50$, $m' = 99$, and different values of η, errors are plotted in Figure 1.3, and results are shown in Figure 1.4. Since the feature point coordinates of the novel faces may be difficult to recover exactly by the model, low values of η lead to overfitting: For $\eta = 0$ and $\eta = 0.0001$, the facial surface

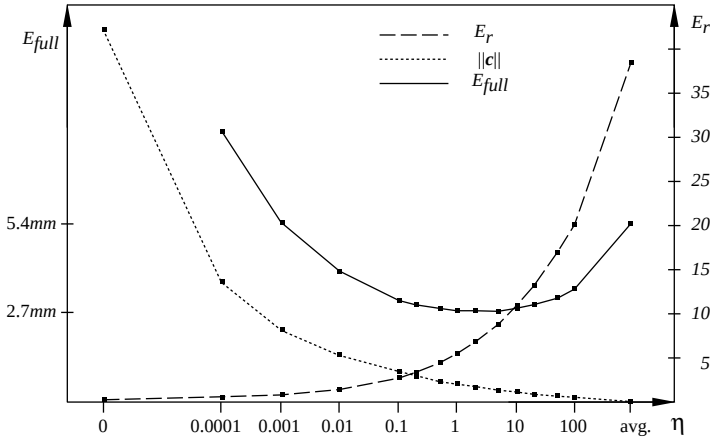


Figure 1.3: The effect of η on average reconstruction results for 100 novel faces, given 50 feature points, and using 99 principal components. As η increases, the feature points are matched less precisely, so E_r grows. In contrast, $\|\mathbf{c}\|$ decreases, as the results become more plausible. The overall 3D shape error E_{full} is lowest for a tradeoff between both criteria.

is heavily distorted, and the overall error E_{full} is large. Still, the feature point coordinates are precisely recovered, as indicated by the low error E_r.

As η increases, E_r grows, while $\|\mathbf{c}\|$ decreases, indicating that the prior probability of the solution gains more weight in the optimization. As the shape becomes more smooth and more plausible, the overall reconstruction error E_{full} decreases, and reaches its minimum at $\eta = 2$.

If η is too large, the output is too close to the average to fit the data, so both E_r and E_{full} are high. The values on the right in Figure 1.3 are the baseline obtained with the average head $\overline{\mathbf{v}}$.

Table 1.1 demonstrates how the number of feature points and principal components affects matching quality. The reduced set of 40 principal components is formed by those dimensions $\mathbf{s}_i$ with maximum variance. As expected, the error E_{full} is lowest with the largest set of feature points and the full set of 99 principal components.

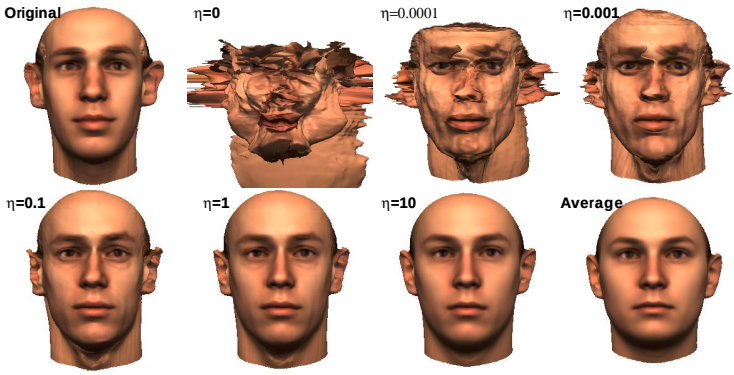


Figure 1.4: Given the image coordinates of 50 feature points of a novel face (top left), 3D shape was reconstructed with 99 principal components. The result depends on a tradeoff between the precision of feature point matching, and prior probability. This tradeoff is controlled by the parameter η.

1.6.2 Correct Reconstruction of Training Faces

In this section, we verify that the faces of the training set are exactly recovered by the system, using all $m' = 99$ principal components, if f is large enough, and if the feature point coordinates are precise.

For $f = 50$ and $f = 1000$, the dimension of $\mathbf{r}$ is $l = 2 \cdot f \geq m'$, so the problem $\mathbf{Qc} = \mathbf{r}$ has a unique solution. This solution is recovered by the system, as indicated by the low values of E_r and E_{full} in Table 1.2, for $\eta = 0$.

In contrast, the solution for $f = 17$ is not unique. Within the set of solutions, the method returns a vector that is closer to the average (smaller $||\mathbf{c}||$), yet

E_{full}	$f = 17$	$f = 50$	$f = 1000$
40 principal components	3.21	2.81	2.38
99 principal components	3.16	2.72	2.24

Table 1.1: The average 3D shape error E_{full} for reconstruction of 100 novel faces at optimal η depends on the number of principal components, and the number of feature point positions available.

Training data	$f = 17$	$f = 50$	$f = 1000$
E_r	8.9e-5	1.4e-4	3.7e-3
$\|\mathbf{c}\|$	5.8	9.9	9.9
E_{full}	2.1	0.0017	5.6 e-05

Table 1.2: Average reconstruction errors for all training faces, given different numbers of feature points f. With all 99 principal components and $\eta = 0$, the problem $\mathbf{Qc} = \mathbf{r}$ is solved exactly, so E_r is low . However, for $f = 17$, the solution is not uniquely defined.

produces an error $E_{full} > 0$. Still, this is the best guess, given the ambiguous information.

1.6.3 Noisy Feature Point Coordinates

As discussed in the previous section, the shape of training faces can be recovered prefectly from 50 feature points if their coordinates are precise. In this case, $\eta > 0$ would impair the quality of the result, as shown by the solid line in Figure 1.5.

However, if Gaussian noise is added to the 2D point coordinates, $\mathbf{r}$ becomes more and more difficult to recover, and overfitting occurs. This is demonstrated by the large errors E_{full} observed for small η if noise with a standard deviation of $\sigma_N = 0.1$ and $\sigma_N = 1$ pixels is added to the horizontal and vertical image coordinates of each feature point.

As we observed previously with novel faces, the values of E_{full} in Figure 1.5 have a clear minimum for intermediate values of η. In fact, these minima occur at $\eta = \sigma_N^2$, so matching quality is best for the vectors with maximum posterior probability (Section 1.4.1).

1.6.4 Robustness with respect to L

A similar effect to noise occurs if the matrix $\mathbf{L}$ used for reconstruction is different from the mapping that produced the feature coordinates in $\mathbf{r}$. This mismatch is relevant for real images, since the geometry of the imaging setup will in general be unknown. In particular, perspective projection produces results that are slightly different from what is simulated by the orthographic projection in $\mathbf{L}$.

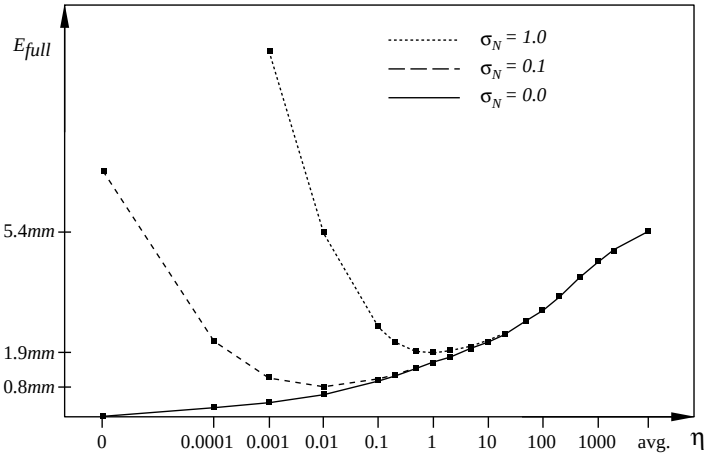


Figure 1.5: The average shape reconstruction errors for 100 training faces depend on the level of noise σ_N added to each feature point coordinate. While noise-free data are best analyzed with $\eta = 0$ (solid line), reconstruction quality is best at $\eta = \sigma_N^2$ for noisy data.

Figure 1.6 shows overall shape errors E_{full} obtained with 50 feature coordinates that were computed for a frontal view. The matrices $\mathbf{L}$ used for reconstruction include rotations of $\phi = 0°$, $1°$, $2°$, and $4°$ around the vertical axis. While the training faces are perfectly recovered with the correct mapping $\phi = 0°$ for $\eta = 0$, performance at angles $\phi > 0°$ is improved significantly with appropriate values of η.

1.6.5 Results on Real Images

Figure 1.1 shows an example of 3D shape reconstruction from a set of 26 feature points that were selected by the user. With the limited information about the face, the method cannot capture details of face shape as precisely as an optimization based on color values in the image [BV99]. However, the overall shape is recovered well, and if texture is extracted from the image [BV99], the technique provides realistic 3D head models.

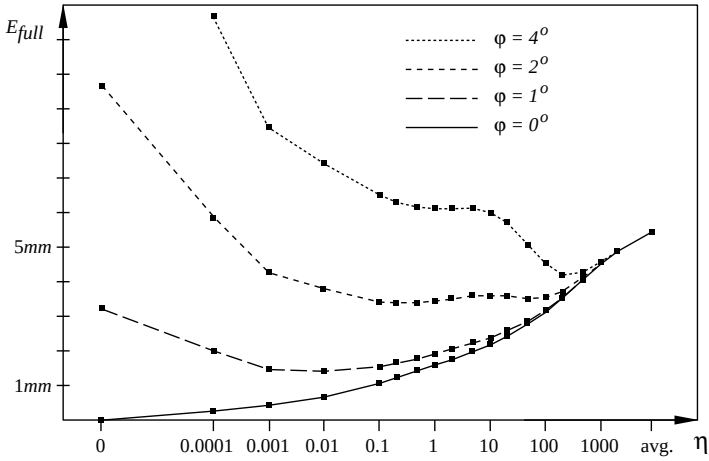


Figure 1.6: Reconstruction from 50 feature coordinates of the training faces at frontal orientation with an incorrect mapping $\mathbf{L}$ *that includes rotations around the vertical axis. 3D shape error* E_{full} *is reduced by choosing appropriate weights* η*.*

1.7 Conclusion

We have presented a method that infers vector dimensions of data vectors from incomplete measurements. The method is based on a vector space spanned by a set of examples, and on statistical properties of the data. Derived from a Bayesian framework, the technique finds the vector with maximum posterior probability, given the measurement and the examples.

With the vector space of faces provided by a morphable face model, we estimated 3D shape of a high resolution face model from the positions of a small set of feature points in an image. We evaluated reconstruction quality in terms of 3D displacements from the veridical shape of faces, and investigated sensitivity to noise and misalignments.

Clearly, a small set of feature positions is insufficient to recover all details of a face, such as the shape of the nose. However, the technique reliably estimates the overall shape and aligns the 3D face with the image, which can be useful for many application. Since the reconstruction is calculated in a single

step, the computation is performed fast enough for interactive tools.

In the future, we are planning to develop methods for choosing the optimal weight factor η by techniques such as cross validation within the training set.

1.7.1 Acknowledgements

The database of 200 laser scans was collected by Niko Troje and Tordis Philipps at Max-Planck-Institut für Biologische Kybernetik, Tübingen.

References

[BV99] V. Blanz and T. Vetter. A morphable model for the synthesis of 3d faces. In *Computer Graphics Proceedings SIGGRAPH'99*, pages 187–194, Los Angeles, 1999.

[DHS01] R.O. Duda, P.E. Hart, and D.G. Stork. *Pattern Classification*. John Wiley & Sons, New York, 2nd edition, 2001.

[Hay98] S.S. Haykin. *Neural Networks: A Comprehensive Foundation*. Prentice Hall, 1998.

[HBVL00] B. W. Hwang, V. Blanz, T. Vetter, and S. W. Lee. Face reconstruction from a small number of feature points. In *International Conference on Pattern Recognition, ICPR2000*, Barcelona, Spain, 2000.

[PTVF92] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical recipes in C : the art of scientific computing*. Cambridge University Press, Cambridge, 1992.

[Sto99] J. Stoer. *Numerische Mathematik I*. Springer, Berlin, 8 edition, 1999.

[Vap95] V. N. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag, New York, 1995.

■

Chapter 2

Anatomic Object Extraction from Medical Volume Data with a Rule-Based Image Analysis System

Hans-Heino Ehricke[*]

Due to low tissue contrasts, the exploration of medical volume data often requires the extraction of anatomic objects of interest prior to 3D visualization. This paper describes a PROLOG-based approach for this highly complex computer vision task. An image analysis system is proposed, which introduces knowledge from the fields of human anatomy, medical physics and pattern recognition into the interpretation process. It utilizes a rule-based control component to manage selection of appropriate low-level operators, adaptation of their parameters, application to the image data and verification of operator output. Results from the automatic

[*] Computer Graphics Lab, University of Applied Sciences Stralsund, Germany. E-Mail: Hans.Ehricke@fh-stralsund.de

interpretation of MRI head datasets illustrate, that by the introduction of even little pieces of knowledge great advances towards the automation of image analysis procedures are possible.

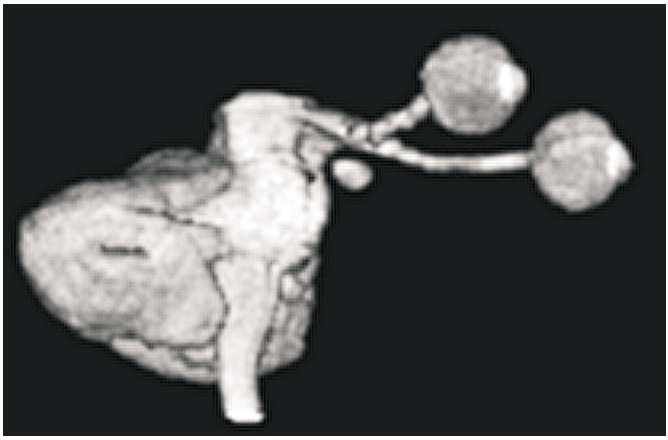


Figure 2.1: 3D visualization of brainstem, cerebellum and optic nerve after segmentation from a MRI dataset.

2.1 Introduction

The last decade has seen tremendous advances in 3D image acquisition techniques, especially in medical imaging. Ultrafast Magnetic Resonance Imaging (MRI), Spiral and Multidetector Computed Tomography (CT) and 3D Ultrasound have proven their clinical value. The demand for intelligent image processing tools to cope with the resulting data flood has become an urgent problem. Computer Graphics researchers have proposed volume visualization techniques, many of which contain great potential for medical visualization applications. Particularly by the availability of hardware-accelerators for volume and surface rendering the practical value of these methods has been substantially increased. Clinical applications of 3D image analysis include neurosurgical treatment planning, surgical training simulators, morphometric evaluation (e.g. brain or cardiac volume) and multi-modality image fusion.

Hardware-accelerated 3D visualization systems allow interactive investigation of volume datasets by playing with rendering parameters, e.g. voxel opacities [CCF95]. However, only a small number of high-contrast objects and tissues, eg. bone in CT, may be visualized with sufficient quality by this approach. This is due to the high complexity of medical volume data and the usually low tissue contrast. Therefore, in most situations volume data have to be preprocessed in order to enhance the contrast between different tissues or to completely isolate objects of interest (Fig. 2.1).

Generations of researchers have dedicated their work to the extraction of anatomic objects from medical images. The goal has always been to reduce the amount of tedious manual interactions. In the early ninetees a variety of authors reported good results with multispectral tissue classification strategies [HZS+90, GMK+91]. They rely on the availability of multispectral image data, i.e. multiple channels for each pixel/voxel, and apply classification techniques, such as clustering or bayesian classifiers. It could be demonstrated, that they work well e.g. with MRI data, acquired with multiple-echo pulse sequences.. Due to the fact, that multispectral image data can hardly be acquired as 3D datasets, multispectral tissue classification approaches have not become practical in volume visualization. Recently, a number of image analysis systems for medical volume data have been proposed, which require a minimum of user interaction. These systems have in common, that they not only rely on the application of a few low-level operators, but explicitly make use of a priori knowledge. Pizer et al. [PFY+99], Lötjönen et al. [LMR+98], Kelemen et al. [KSG99] and Buck et al. [BEST95] use shape models, which are elastically deformed and matched to object surfaces within volume data. In this manner anatomic objects are delineated and extracted. Jia et al. [JTL+98] and Dawant et al. [DHT+99] use detailed digital anatomic atlasses as the knowledge base. Atlas data are elastically deformed and matched to the image data, similar as in the above approaches. Van Leemput et al. propose to use an atlas, consisting of a priori probability maps for white matter, gray matter and cerebrospinal fluid, in order to enhance the outcome of a maximum likelihood classification operator. Haris et al. [HEM+99] describe a directed tree graph as an anatomy model for the cardiac arteries. The graph is used for the segmentation and labeling of coronary angiograms. Atkins and Mackiewich [AM98] demonstrate, that with little anatomic knowledge the automatic segmentation of brain structures from MRI data is possible. They use a spatial information map to eliminate presegmented regions, which do not correspond to brain tissue.

In this paper we present an approach, integrating knowledge from differ-

ent disciplines, i.e. image processing, anatomy, medical physics, in order to control image analysis processes.

2.2 System architecture

A hierarchical system for the stepwise segmentation of 3D datasets has been developed. In biological visual systems a top-down strategy in object recognition can be observed. Starting with the subdivision of a scene into larger objects and background, structures with growing level of detail are recognized step by step. This strategy of gradual refinement has been used as a model for the design of the analysis system. If we take the example of interpreting a tomographic head dataset, the system typically would start with delineating head-contours in the slice-images thus providing a subdivision into head and background. The head outline is used as a spatial map to discriminate between brain tissue and extracerebral regions, which would be achieved in step two. Using brain contours, the ventricular system may be delineated. The anatomic model, which can be used in this case, is demonstrated by Fig. 2.2.

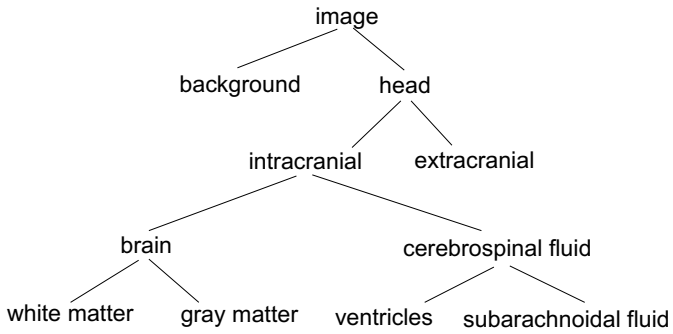


Figure 2.2: Anatomic model of the human head describing a hierarchical object modularization.

Fig. 2.3 illustrates the structure of the analysis system. Basically we can distinguish between three components:

1. Control system,
2. Methods pool,

3. Knowledge base.

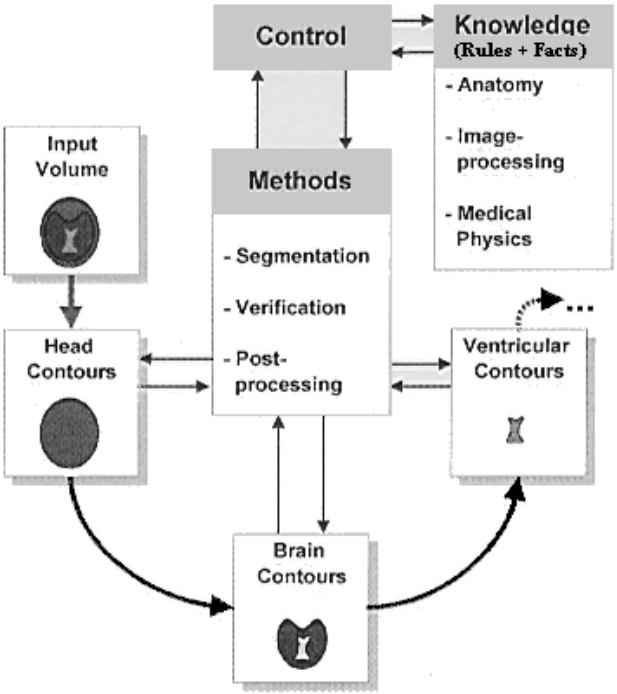


Figure 2.3: Components of the rule-based analysis system and stepwise process of subdividing a 3D head dataset into smaller and smaller anatomic regions.

The analysis process is supervised by the control system. It is responsible for the selection of operators from the methods pool, which seem to be suitable in a certain situation during the analysis process. It adjusts the parameters, initiates the application of the operator to the image data and checks the result. If the result is not satisfactory, parameters are changed or another method is selected. For this purpose the control system makes use of non-procedural knowledge, which is stored as a list of rules and facts in a knowledge base.

2.3 Low-level image processing

A pool of various low-level segmentation operators has been developed. The available methods fall into the categories segmentation, verification and post-processing. They have been implemented as C-routines. The segmentation operators may be subdivided into region- and edge-based approaches including classifiers, first- and second-order gradient operators and line-tracking techniques. The objective of the verification methods is to check the results generated by the application of a segmentation operator. This allows the control system to acquire knowledge about the status of the analysis process in order to select actions which have to be initiated next. Postprocessing methods are used to refine segmentation results, e.g. refine the location of object edges. Examples are standard morphologic operators (erosion, dilation, opening, closing) and special purpose operators which have been adapted to certain objects and situations.

2.4 Control system

Supervision of the analysis process is carried out by a control system, which has been realized using a PROLOG interpreter. Based on the concepts of predicative calculus and backward reasoning the system starts with a goal, e.g. `find_brain`, and tries to meet all the conditions necessary to achieve it. Within the chain of backward reasoning low-level segmentation operators are selected, their parameters adapted, application to the image space initiated and the results checked by verification operators. Examples of rules are:

```
find_brain:-
  apply_marr_hildreth_operator,
  divide_image_into_anatomically_meaningful_regions,
  find_brain_starting_region,
  aggregate_all_brain_regions,
  verify_brain_result,
  refine_brain_result.

verify_brain_result:-
  no_connection_to_skincontour,
  no_overlap_with_eye_region,
  reasonable_contour_length,
```

```
comparable_to_adjacent_slices.
```

In order to accept the goal part of the clause as true, all the conditions in the list must be found true. These conditions may be production rules themselves or direct calls of low-level procedures. If a rule cannot be set true, the interpreter searches the database for an alternative rule with the same name and examines its conditions.

Although the image dataset is analyzed in a slice-wise manner, many of the segmentation operators work in 3D space. The 3D information is also used to compare the segmentation results of adjacent slices and thus check them for severe errors.

2.5 Results

In order to evaluate our image analysis strategy, we have developed a rule database for the interpretation of MRI 3D head datasets. The system was tested with 10 different MRI datasets acquired with standard pulse sequences. The spatial resolution was about 1.0 mm isotropically. The primary goal of the test was to perform an automatic segmentation of brain tissue. Very reliable results were achieved in all datasets for the upper part of the head down to the level of the orbita (see Fig. 2.4). Advancing further downwards, the complexity of anatomic structures increases and therefore segmentation errors are more likely to occur. Nevertheless, by the introduction of verification operators good results were achieved. An example of an automatically detected and resolved segmentation error is illustrated by Fig. 2.5. In this case a Marr-Hildreth operator was applied for edge extraction and thus the image subdivided into anatomically meaningful regions. The experienced viewer may observe a connection between areas belonging to the brain and extracerebral parts. This error may be easily detected, since a region which is supposed to be brain must have no connection to the head outline and the image background. Application of the Marr-Hildreth operator with a modified set of parameters or using a different segmentation operator may resolve this problem. As an overview over the segmentation results, Fig. 2.6 presents a 3D surface image of the brain. This was generated on basis of the segmented dataset with surface rendering. It demonstrates that the segmentation worked very well for the whole brain down to the temporal lobe. Beyond this region errors - marked by an ellipse - which are caused by insufficient anatomic knowledge in the knowledge base occured.

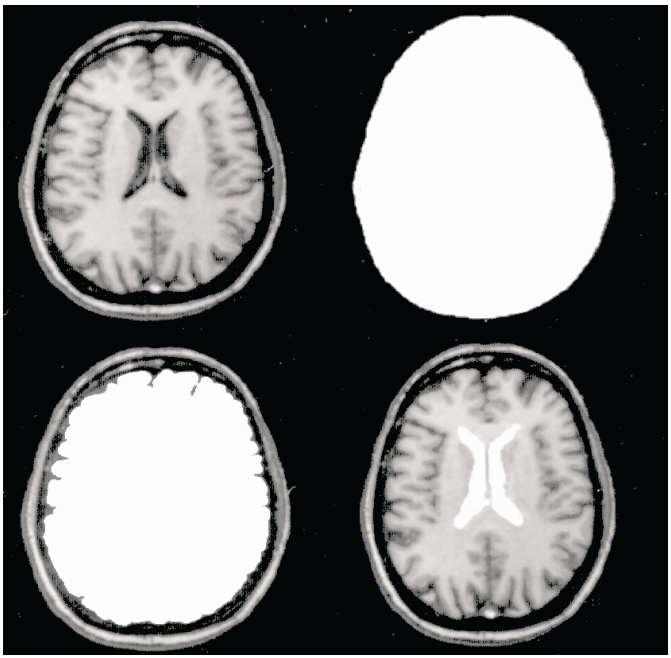


Figure 2.4: Result of fully automatic interpretation of a MRI dataset. From u.l. to l.r.: Original slice image, head contour mask, brain contour mask and ventricular contour mask.

2.6 Summary and conclusion

An artificial intelligence approach for the analysis of medical 3D datasets was presented. A rule-based system for supervision of low-level image processing operators was developed according to the objective of subdividing head datasets into smaller and smaller anatomic entities. Knowledge from the fields of human anatomy, medical physics and image processing was incorporated as a set of rules and facts.

The system allowed the reliability of the vision process to be considerably enhanced. The results indicate, that by the introduction of even small pieces of knowledge the amount of user interaction for the correction of errors may substantially be reduced. On the other hand, the role of powerful low-level

operators must not be underestimated. Only by the combined application of low- and higher-level image processing techniques may the problem of medical 3D data interpretation be satisfactorily solved. The presented system is regarded as a step in this direction. We expect substantial improvements by the combination of approaches for eleastic model matching with the presented method. This could be achieved by the extension of the knowledge base with an anatomic atlas and the introduction of matching operators to the operators pool.

References

[AM98] M.S. Atkins and B.T. Mackiewich. Fully automatic segmentation of the brain in mri. *IEEE Transactions on Medical Imaging*, 17(1):98–107, 1998.

[BEST95] T. Buck, H.-H. Ehricke, W. Strasser, and L. Thurfjell. 3-d segmentation of medical structures by integration of raycasting with anatomic knowledge. *Computers & Graphics*, 19(3):441–449, 1995.

[CCF95] B. Cabral, N. Cam, and J. Foran. Accelerated volume rendering and tomographic reconstruction using texture mapping hardware. In S. Spencer, A. Kaufman, and W. Krueger, editors, *Proceedings 1994 Symposium on Volume Visualization*, pages 91–98. IEEE Computer Society Press, Los Alamitos, 1995.

[DHT+99] B.M. Dawant, S.L. Hartmann, J.-P. Thirion, F. Maes, D. Vandermeulen, and P. Demaerel. Automatic 3-d segmentation of internal structures of the head in mr images using a combination of similarity and free-form transformations: Part i, methodology and validation on normal subjects. *IEEE Transactions on Medical Imaging*, 18(10):909–916, 1999.

[GMK+91] G. Gerig, J. Martin, R. Kikinis, , and O. Kübler. Segmentation of dual-echo mr head data. In H.U. Lemke, M.L. Rhodes, C.C. Jaffe, and R. Felix, editors, *Computer Assisted Radiology*, pages 606–608. Springer, Berlin, Heidelberg, New York, 1991.

[HEM+99] K. Haris, S.N. Efstratiadis, N. Maglaveras, C. Pappas, J. Gourassas, and G. Louridas. Model-based morphological segmentation and labeling of coronary angiograms. *IEEE Transactions on Medical Imaging*, 18(10):1003–1015, 1999.

[HZS+90] W. Härle, I. Zuna, L.R. Schad, R. Brix, W. Semmler, G. van Kaick, and W.J. Lorenz. Mri tissue characterization and segmentation of human brain tissues using a prolog-based expert system. In H.P. Higer and G. Bielke, editors, *Tissue Characterization in MR Imaging*, pages 313–318. Springer, Berlin, Heidelberg, 1990.

[JTL+98] C. Jia, O. Tan, X. Lu, H. Duan, and W. Lu. Automatic registration and segmentation of human brain with a computerized brain atlas. In H.U. Lemke, M.W. Vannier, K. Inamura, and A. Forman, editors, *Computer Assisted Radiology*, pages 153–158. Elsevier, Amsterdam, 1998.

[KSG99] A. Kelemen, G. Szekely, and G. Gerig. Elastic model-based segmentation of 3-d neuroradiological data sets. *IEEE Transactions on Medical Imaging*, 18(10):828–839, 1999.

[LMR+98] J. Lötjönen, I.E. Magnin, P.-J. Reissman, J. Nenonen, and T. Katila. Segmentation of magnetic resonance images using 3d deformable models. In W.M. Wells, A. Colchester, and S. Delp, editors, *Medical Image Computing and Computer-Assisted Intervention*, pages 1213–1221. Springer, Berlin, Heidelberg, New York, 1998.

[PFY+99] S.M. Pizer, D.S. Fritsch, P.A. Yushkevich, V.E. Johnson, and E.L. Chaney. Segmentation, registration, and mesurement of shape variation via image object shape. *IEEE Transactions on Medical Imaging*, 18(10):851–865, 1999.

■

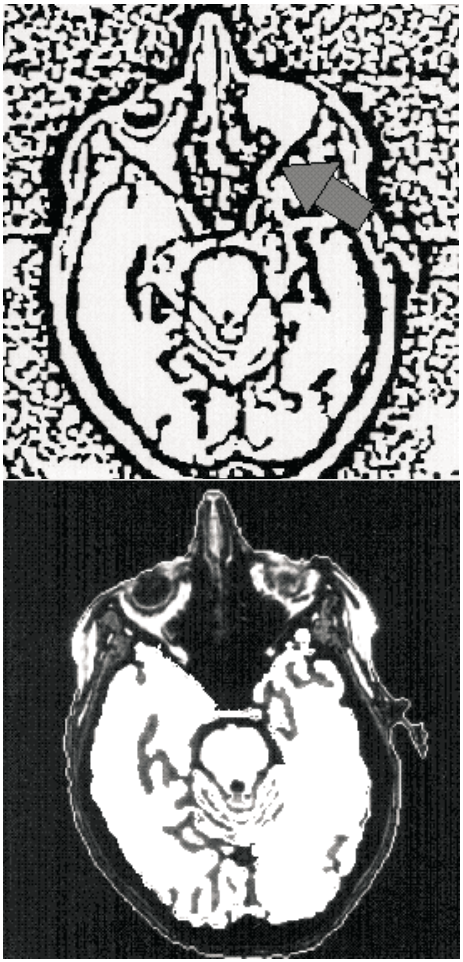


Figure 2.5: Automatic detection and orrection of segmentation errors, generated by the application of a Marr-Hildreth operator. Top: The brain region is directly connected with the region of the right eye (arrow). Bottom: Correct result (brain mask) after application of the same operator with adjusted parameters.

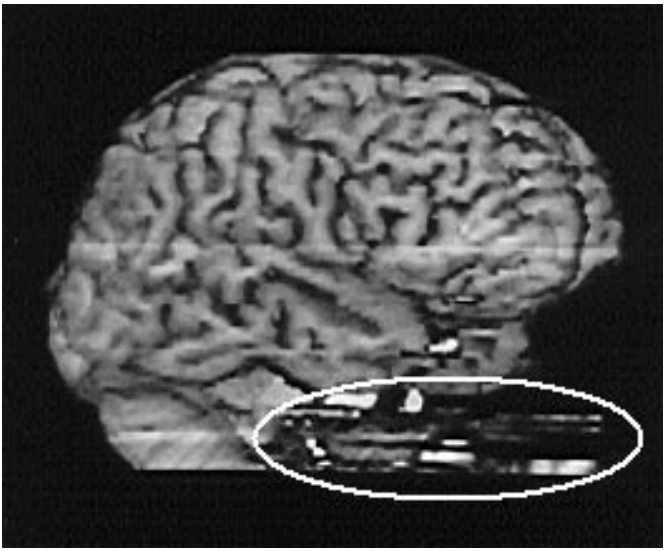


Figure 2.6: 3D brain surface generated by surface rendering on the basis of an automatically segmented MRI dataset. The eeliptical region marks segmentation errors, due to the lack of detailed anatomic knowledge.

Chapter 3

History between the Realities — Hi-Tech User Interfaces for Exploring the Past

L. Miguel Encarnação, Oliver Bimber[*]*, Mark Billinghurst*[†]

The learning possibilities using artifacts on display in today's museums are limited since they are removed from their original context, are too fragile to handle and are often damaged. In recent years museums have tried to overcome these limitations through the use of a variety of multimedia technologies including audio and video guides – with moderate success. This article outlines display and interaction technologies that could transform the museum experience from passive viewing to interactive learning.

[*] Fraunhofer Center for Research in Computer Graphics, Providence RI, USA. E-Mail: {mencarna|obimber}@crcg.edu

[†] Human Interface Technology Laboratory, University of Washington, Seattle, WA, USA. E-Mail: grof@hitl.washington.edu

3.1 Introduction

In Lewis Carroll's classic book "Alice in Wonderland" Alice is prevented from exploring the magical Wonderland until she passes through the looking glass and leaves the real world behind. In many ways, when people go to a museum today they hope for a similar experience. Visitors can see amazing artifacts of cultural heritage in front of them; the glass cases enclosing them are like windows to the past.

However, unlike Alice's looking glass it is impossible for a person to step through these windows into a historic wonderland. The artifacts on display are removed from their original context, are too fragile to handle and are often damaged. The pieces are so rare that it is difficult to show them out of the museum setting, and related pieces from the same historical site may be scattered in museums across the world. All of these factors mean that museums may not be achieving their potential as centers for historical learning.

3.2 Motivation

In recent years museums have tried to overcome the described limitations through the use of a variety of technologies. Audio guides are commonplace and visitors can often use multimedia kiosks to explore topics in more depth. Many early approaches of modeling and animating historical artifact showed promising successes in discovering invisible aspects of ancient art and ingenuity. The reconstruction and subsequent analysis of Raphael's 'School of Athens' painting, for instance, discovered the artist's intentions with respect to perspectives within the image [HKM87]; the Castel del Monte architecture [EHS96] exposes to the bird's eye perspective interesting fractal base geometries; the Stoeffler Globe's [OG95] star maps give an insight into the function of the delicate ancient mechanism. In continuation of this history of applying computer graphics approaches to the exploration of Cultural heritage at the University of Tübingen[3], more recently the reconstruction of the Schickard calculator [HEN01] employs Java 3D to communicate the artifact's functionality to a broader audience.

However, in many ways such technologies just replace one glass case for another; rotating an artifact on a computer screen is little more satisfying than

[3] An overview of international work in this area can be found in [enc98].

seeing it in a display case. In an ideal museum setting the interfaces to the technology should be transparent to visitors, allowing them to interact with the digital domain as easily as they can with the real world around them.
This article outlines three Mixed Reality (MR) technologies developed at the CRCG and HIT Lab that could transform the museum experience from passive viewing to interactive learning. The Virtual Showcase, MagicBook, and HI-Space developments are explained in this paper. The potential for these technologies is illustrated by the recent Virtual Dig installation at the Seattle Art Museum in Seattle, Washington. In this case visitors were actively involved in the discovery process and gained a great understanding of the context of the original artifacts and the wonder of archaeology. Yet this experience only hints at the future possibilities that could occur by combining these three technologies.

3.3 Technological Approaches

In an ideal museum setting the technology should vanish so that visitors can interact with the digital domain as easily as they can with the real world around them. There are several ways to make computers invisible. In the area of Tangible Interfaces [IU97] real objects are used as interface widgets, while in immersive Virtual Reality (VR) the real world is replaced entirely with a computer-generated environment. As Milgram points out [MK94], these types of computer interfaces can be placed along a continuum according to how much of the users environment is computer generated (Fig. 3.1). On this Reality-Virtuality line, Tangible Interfaces lie far to the left, while immersive Virtual Environments are placed at the rightmost extreme. However it is the middle ground that is most interesting. It is here that Augmented Reality (AR) techniques can be used to overlay virtual images onto real world objects, or Augmented Virtuality (AV) methods used to add real world content to completely virtual scenes.

Following this analogy with respects to objects of cultural heritage, looking from a different perspective one could argue that a museum exhibit presents a historically virtual environment, since the context, function and purpose of the artifact might not be easily accessible or understandable anymore. The physical artifact could be considered the ‘real’ component in this setting. Consequently, the surroundings of the exhibit, the documentation on the walls, audio guides

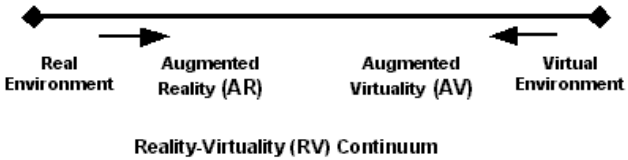


Figure 3.1: Milgram's Reality-Virtuality Continuum [MK94]

and video mock-ups would represent the augmentation.

The unique characteristics of AR and VR interfaces provide many opportunities for them to be used in a museum setting. For example, in an AR experience, missing portions of a damaged artifact could be shown as a virtual overlay, completing the artifact, or a real artifact could be shown in a virtual surround enabling visitors to see the piece in context. Currently, on-the-side illustrations aim at communicating such missing information to the observer, yet burdening him the additional task of correlating the exhibited artifact with the illustration. Fully immersive VR experiences, on the other hand, are much more powerful ways of demonstrating how an object worked in its original setting, allowing people to pick up and manipulate virtual artifacts, or even immerse the visitors into the original historical setting, creating a kind of time-travel interface.

Many researchers around the world have begun applying VR and specifically AR technologies to the interactive exploration of Cultural Heritage (e.g., [enc01b, enc01a, enc00]). However there are problems with VR and AR experiences that may prevent them from being widely used. In immersive VR interfaces users wear bulky head-mounted displays that isolate them from the real world and reduce the sense of shared discover that is found in museums. The equipment is expensive and hard to integrate into a museum setting, often requiring specially trained staff to operate it. Cost considerations and the time it takes to use the interface may also limit the number of people that can try the experience. Finally, it is often difficult to switch between modes (the real world, AR scene, VR scene) for different purposes.

One solution to these problems is to explore other interface types that lie along the Reality-Virtuality Continuum, particularly those that support seamless tran-

sitions between real and virtual worlds. In our work we have developed a variety of Augmented Reality technologies that are ideally suited to a museum setting. Generally, the involved interface technologies can be grouped into display technologies and interaction tools. An innovative technology which supports the smooth transition between VR and AR displays is the *Virtual Showcase*, a projection table that allows multiple people to see three-dimensional virtual content as if it was really on the tabletop [BFSEa01].

On the interaction side, the *HI-Space* represents projection table that allows for free-hand and object-based interaction with digital content, whereas *the MagicBook* depicts an application of AR technology that allows a user to smoothly move between reality and virtuality in a collaborative setting [BKP01, KB99].

3.3.1 Display technologies for Cultural Heritage Exhibits – The Virtual Showcase

The Virtual Showcase [BFSEa01] has the same form factor as a real showcase making it compatible with traditional museum displays. Physical scientific and cultural artifacts can be placed inside the Virtual Showcase, thus additionally allowing for their three-dimensional graphical augmentation. Inside the Virtual Showcase, virtual representations and physical artifacts share the same space, thus providing for new ways of merging and exploring real and virtual content. The virtual part of the showcase can respond in various ways to a visitor's input enabling intuitive interaction with the displayed content. A Virtual Showcase consists of two main parts(Fig. 3.2): a convex assembly of half-silvered mirrors (1) and a graphics display (2). So far, we have built Virtual Showcases with two different mirror configurations. Our first prototype consists of four half-silvered mirrors assembled as a truncated pyramid. Our second prototype uses a single mirror sheet to form a truncated cone.

These mirror assemblies are placed on top of a projection screen in both setups. Multiple users can see real objects inside the mirror assembly through the optics merged with the graphics displayed on the projection screen. The showcases' contents are illuminated using a controllable light source while view-dependent stereoscopic graphics are presented to the observer(s). For our current prototypes, stereo separation and graphics synchronization are achieved using active shutter glasses in combination with infrared emitters. Head tracking is achieved using an electro-magnetic tracking device. Particularly intriguing is our second, cone-shaped prototype, which additionally provides a seam-

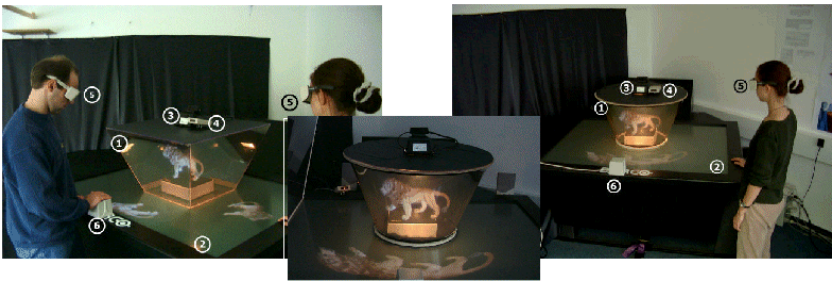


Figure 3.2: Prototype Virtual Showcases

less surround view onto displayed artifacts.

The virtual part of the showcase can react in various ways to single or multiple users enabling intuitive interaction with the displayed content. These interactive showcases are an important step in the direction of ambient intelligent landscapes, where the computer acts as an intelligent server in the background, and users can focus on their tasks rather than on operating computers.
The described features in support of Virtual or Augmented Reality environments make the Virtual Showcase a testbed platform for a variety of applications, currently developed in different projects, ranging from training and rehearsal as well as scientific visualization (cf. Fig. 3.3) to entertainment and education (i.e., edutainment) (cf. Fig. 3.4).

Fig. 3.5 gives a glimpse onto the future design of the Virtual Showcase, based on a refinement of the system and integrating our "lessons-learned" and application experiences. The depicted inverted setup ensures even less obstruction by the setup and supports closer multi-user collaboration.

The Virtual Showcase is currently further developed in collaboration with several European museums sponsored by the European Union under IST-2001-28610.

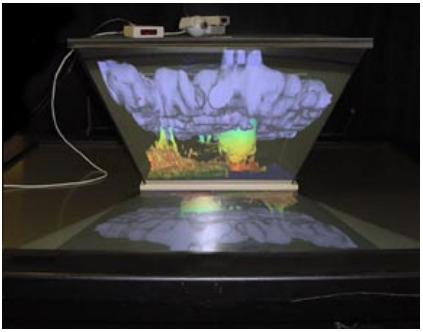


Figure 3.3: Weather Visualization in the Virtual Showcase (conceptual image).

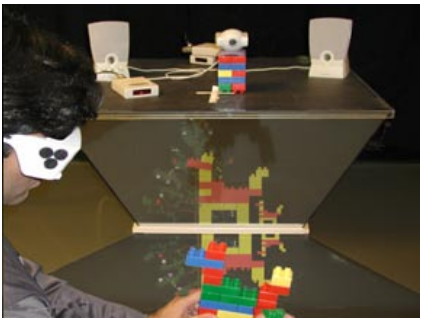


Figure 3.4: Edutainment application focusing on problem-solving skills (conceptual image).

3.3.2 Interaction Technologies for Cultural Heritage Exhibits

The Magic Book Interface

Young children often fantasize about being able to fly into the pages of a fairy tale and becoming part of the story. The MagicBook makes this fantasy a re-

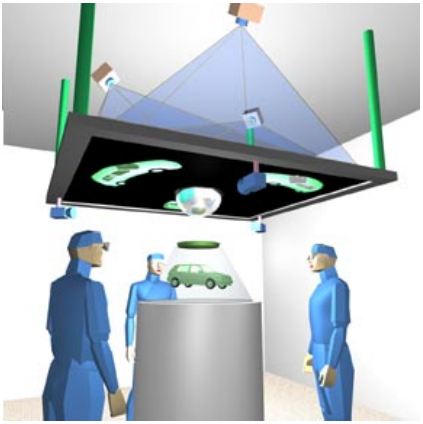
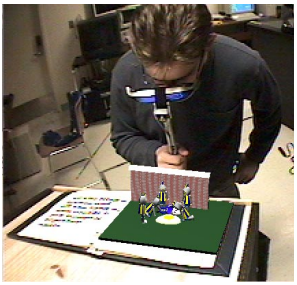


Figure 3.5: Future concept for the Virtual Showcase.

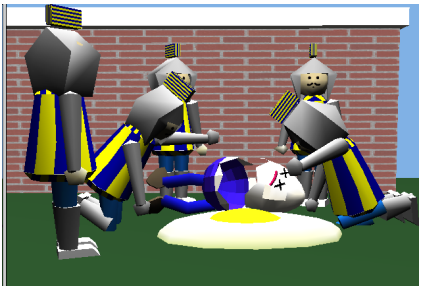
ality using a normal book as the main interface object. People can turn the pages of the book, look at the pictures, and read the text without any additional technology (Fig. 3.6a)). However, if a person looks at the pages through an Augmented Reality display they see three-dimensional virtual models appearing out of the pages (Fig. 3.6b)). The models appear attached to the real page so users can see the AR scene from any perspective simply by moving themselves or the book. The models can be any size and are also animated, so the AR view is an enhanced version of a three-dimensional "pop-up" book. Users can change the virtual models simply by turning the book pages and when they see a scene they particularly like, they can fly into the page and experience the story as an immersive virtual environment (Fig. 3.6c)). In the VR view they are free to move about the scene at will, so in the MagicBook people can experience the full Reality-Virtuality continuum.

The current MagicBook interface has two components; a handheld display (HHD) and the physical book. Each user has their own graphics computer with a HHD attached and these computers are networked. The HHD is a handle with a Sony Glasstron display mounted at the top, an InterSense InterTrax inertial tracker at the bottom, a small camera on the front of the Glasstron display and a switch and pressure pad (Fig. 3.7). The books used in the MagicBook

a) Reality



b) Augmented Reality



c) Immersive Virtual Reality

Figure 3.6: Using the MagicBook interface to move between Reality and Virtual Reality.

interface are normal books with text and pictures on each page. Certain pictures have thick black borders surrounding them. When the reader looks at these pictures through the HHD, computer vision techniques are used to precisely calculate the camera position and orientation relative to the picture [KB99]. Once this is known the graphics computer generates virtual images that appear precisely registered with the real pages. Simply turning the pages will reveal new pictures and virtual images.

Users each have their own displays, and if two or more users are looking at the same page, they will see the same virtual image attached to the picture from their individual viewpoints. Since they can see each other at the same time they can use natural non-verbal and verbal communication cues to enhance the collaboration. The HHD can also easily be shared, or taken away from the face if the user wants to experience unmediated reality.

When the user sees an AR scene they wish to explore, flicking the switch on the handle will fly them smoothly into the scene, and transition them into an immersive VR environment. Head tracking is changed from the computer vision module to the InterTrax inertial orientation tracker so readers can look around the scene in any direction. By pushing the pressure pad on the handle they can fly in the direction they're looking. The harder they push the faster they fly. To return to the real world users simply need to flick the switch again.

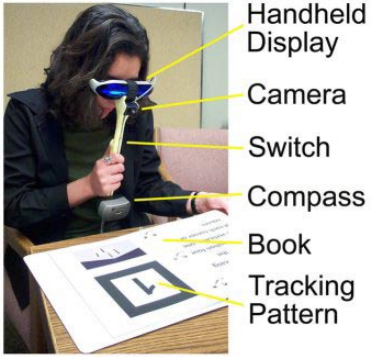


Figure 3.7: MagicBook Technology

When users are immersed in the virtual environment or are viewing the AR scenes their position and orientation is broadcast to the other users. This is used

to place virtual avatars of users that are viewing the same scenes, so users can collaboratively explore the virtual content. Thus the MagicBook supports collaboration on three levels: as a physical object (book), an AR object (overlays on pages) and as an immersive virtual space (by 'flying' into the book.

The HI-Space Interface

The HI-SPACE system[4] is a projective table that supports nature gesture-based interaction with virtual data (Fig. 3.8). It utilizes knowledge from many areas of research, including Psychology, Human-Computer Interaction, Virtual Reality, Kinesiology, and Computer Science, to create a workspace that blurs the boundaries between physical and electronic information. The most desirable aspects of both the physical and electronic information spaces are used to enhance the ability to interact with information, promote group dialog, and to facilitate group interaction with information to solve complex tasks.

The HI-SPACE is being developed to support leading edge HCI features, such as:

- taking advantage of the redundancy of multi-modal (gesture, speech) input,
- direct interaction, which allows a more natural interface,
- supporting groups of people interacting with the same data at the same time and in the same space,
- enabling users in different physical locations to interact with each other and with the same data set,
- supporting the fluid transfer of information and interaction between the physical and electronic spaces,
- maintaining an unencumbered work environment.

In the current HI-SPACE system, sensors (camera, radio frequency tagging antenna, and microphone array) are placed over the table to capture user interactions with a table display. The display itself is a rear-view projection screen being fed by a standard LCD projector. The HI-SPACE places the

[4]Developed by the HITLab in collaboration with Battelle Pacific Northwest Laboratories, Richland, Washington.

Figure 3.8: The HI-Space Interface

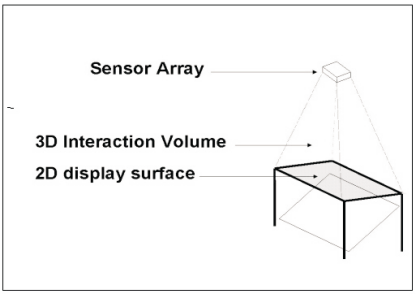


Figure 3.9: The HI-Space Interaction Space

user interaction space between the sensor array and the 2D display surface, as depicted in Fig. 3.9. This creates a 3D interaction volume that allows the user a much greater degree of freedom. The system has the potential to interpret gestures or actions anywhere in the interaction volume and respond accordingly, giving the HI-SPACE much greater potential for advanced interactions than technologies that only mimic touch screen type functionality.

Artifact	Example type of artifact	**Condition of artifact**		
		Complete	**Incomplete**	**Documented only**
Aesthetical	Impressionist painting	AR annotations	AR completion	VR
Conceptual/ Structural	Baroque architecture	AV	AR completion	AV
Functional	Ancient calculator	AR annotations + VR context	AR completion + VR context	AV/VR
Organic	Evolution	AR annotations + VR behavior	AR completion + VR behavior	AV/VR

AR = Augmented Reality, AV = Augmented Virtuality, VR = Virtual Reality

Table 3.1: Potential applicability of mixed reality technology by type and condition of artifact.

3.4 Discussion

The Augmented Reality technologies described able can be applied in a number of ways to create powerful museum experiences. Depending on:

- the nature of the physical artifact
- the condition of the physical artifact
- the exhibition and learning objective associated with the artifact
- the expected audience

and a variety of other factors, different sets of technologies along the mixed reality continuum might be applicable and should be supported in a seamlessly exchangeable way. Table 3.1 shows how different types and conditions of artifacts might favor different types of technologies for display and interaction.

Learning by	**Sample art**	**Potentially valuable technology**		
		AR	**AV**	**VR**[1]
Description	Raphael's "School of Athens", Castel del Monte	MB, VS	MB,HS,VS	HS,VS,MB
Experimentation	Stoeffler Globe, Schickard calculator, Virtual Archeological Dig	VS	VS, HS	HS, VS
Discussion/ Comparison	Virtual Archeological Dig	VS, MB	VS, HS	HS, VS

MB = MagicBook, HS = HI-Space, VS = Virtual Showcase
[1] (incl. non-stereo)

Table 3.2: Potential applicability of presented technologies by learning objectives

Table 3.2 depicts some examples of how the currently developed technologies described above might be applicable to different learning objectives.

These assumptions are derived from experimental applications of these or similar technologies to the interactive visualization of the mentioned classes of artifacts.

3.5 Case Study: Treasures from a Lost Civilization

From May 10^{th} to August 12^{th}, 2001, the Seattle Art Museum showed an exhibit titled Treasures from a Lost Civilization: Ancient Chinese Art from Sichuan. On display were more than 150 priceless bronze, clay and gold pieces of art from the Sichuan region in China. Some of the pieces were more than 3,000 years old and they included the oldest life-sized bronze figure in the world.

3.5.1 The Experience

The Virtual Reality Dig combined the MagicBook and HI-Space technology into an intuitive and engaging museum experience. When users entered the room they saw three tables with projection screens in then. In front of each table was an additional screen. Moving up to a table the visitors were invited to pick up a real brush or shovel. Looking down the visitors could see a projected image of a field of flowers on the table surface. When they brushed the surface of the table, the flowers and dirt were cleared away to reveal buried artifacts underneath.

Users could get a closer look at the artifacts by putting down their brushes, picking up their shovels and scooping them over the table surface. As they did this, on the front projection screen a life-sized three-dimensional virtual model of an artifact appeared on their virtual shovel. They could turn and view it from any viewpoint or compare it to the different artifacts being held by their friends.

The final section of the experience involved discovery and assembly of a life-sized bronze statue. The statue was in several pieces so users first had to find the pieces by pointing at different locations on the table with their brushes. When the pieces were been found they could be picked up using the shovels and placed together. Once assembled a large virtual bronze figure was shown attached to one of the user's shovels. This figure had empty hands, and when assembled a number of objects appeared on the table surface. Users could pick these objects off the surface with their shovels and place them into the statues hands to see which was the perfect fit.

3.5.2 The User Response

While the exhibit was open almost two thousand people a week tried the Virtual Dig experience, for a total of more than 25,000 people. The experience was designed to be able to handle up to 36 people simultaneously, each of whom could walk up to a table and start interacting right away, ensuring a very high throughput rate.

The user response was overwhelmingly positive. Combining table interface elements (the brush and shovel) with augmented reality graphics meant that people had no trouble operating the interface. The unencumbered nature of the interface also meant that people could easily collaborate with each other and experience the wonder of shared discovery. It was not uncommon to see

parents and children working hand in hand to manipulate the virtual models, or grandchildren showing their grandparents how to brush away the virtual dirt. Perhaps the greatest success was the questions that the experience planted in the visitors' mind and the incentive it gave them to find out more. Most tried the virtual dig experience before heading to view the real artifacts on show. So by the time they are saw the bronze masks in the display cases they already had an appreciation for how they were discovered and the features they should be looking for to solve the archaeological mystery. Thus the Virtual Dig transformed the museum experience from one of passive observation to active discovery and enquiry.

3.6 Conclusions

In this paper we have briefly outlined several Augmented Reality display and interaction technologies that could transform the museum experience from passive viewing to interactive learning. The potential for these technologies is illustrated by the Virtual Dig installation at the Seattle Art Museum. In this case visitors were actively involved in the discovery process and gained a great understanding of the context of the original artifacts and the wonder of archaeology. Yet this experience only hinted at the future possibilities that could occur by combining technologies such as the Virtual Showcase, the MagicBook, and the HI-Space. The challenge remains to conduct cross-disciplinary research and application development for the Cultural Heritage domain, in order to invent, develop and deploy display and interaction technologies that are robust enough to subsist the excessive daily use in exhibit environments yet flexible enough to provide transparent interfaces towards museum applications with ever-changing interface requirements.

References

[BFSEa01] O. Bimber, B. Fröhlich, D. Schmalstieg, and L.M. Encarnação. Virtual showcase.s. *IEEE Computer Graphics & Applications*, 21(6):48–55, November/December 2001. Presented in Conference Abstracts and Applications of SIGGRAPH 2001, Los Angeles, CA, USA, ACM Press, 2001.

[BKP01] M. Billinghurst, H. Kato, and I. Poupyrev. The magicbook: A transitional ar interface. *Computers and Graphics*, November 2001.

[EHS96] B. Eberhardt, J. Hahn, and R. Sonntag. Castel del monte - eine virtuelle welt friedrichs ii. *unix/mail*, pages 205–212, 1996.

[enc98] On the net resources - museums and cultural heritage, September 1998. HITLab, University of Washington, URL http://www.hitl.washington.edu/kb/museum.html.

[enc00] Virtual heritage network, 2000. URL: http://www.virtualheritage.neti/. International Society on Virtual Systems and MultiMedia.

[enc01a] Ucla cultural vr lab, 2001. URL http://www.cvrlab.org/. University of California, Los Angeles, CA.

[enc01b] Vis.i.t., 2001. URL: http://www.cineca.it/HPSystems/Vis.I.T/Researches/tecbec.html. Italy.

[HEN01] Frank Hanisch, Bernhard Eberhardt, and Benjamin Nill. Reconstruction and virtual model of the schickard calculator. *Journal of Cultural Heritage*, 1:335–340, January 2001.

[HKM87] G. R. Hofmann, D. Krömker, and G. Mazzola. Rasterbild - bildraster: Anwendungen der graphischen datenverarbeitung zur geometrischen analyse eines meisterwerks der renaissance: Raffaels "schule von athen". In *Beiträge zur Graphischen Datenverarbeitung*. Springer, Berlin/Heidelberg, 1987.

[IU97] H. Ishii and B. Ullmer. Tangible bits: Towards seamless interfaces between people, bits and atoms. In *Proceedings of CHI 97, Atlanta, Georgia, USA*, pages 234–241. ACM Press, 1997.

[KB99] H. Kato and M. Billinghurst. Marker tracking and hmd calibration for a video-based ar conferencing system. In *Proceedings of IWAR 99, San-Francisco, USA*, October 1999.

[MK94] P. Milgram and F. Kishino. A taxonomy of mixed reality visual displays. *IECE Trans. on Information and Systems*, E77-D(12):1321–1329, 1994. (Special Issue on Networked Reality).

[OG95] G. Oestmann and T. Grunert. Johannes stoeffler's celestial globe. *Der Globusfreund*, (43/44):59–76, December 1995. URL: http://www.coronelli.org/.

■

Chapter 4

Scientific Visualization of Large Datasets

Thomas Ertl[*]

One of the main goals of scientific visualization is the development of algorithms and appropriate data models which allow interactive visual analysis and direct manipulation of the increasingly large data sets which result from time-dependent 3D simulations running on massive parallel computer systems or from measurements employing fast high-resolution sensors. This task can only be achieved with the optimization of all steps of the visualization pipeline: semantic compression and feature extraction based on the raw data sets, adaptive visualization mappings which allow the user to choose between speed and accuracy, and exploiting new graphics hardware features for fast and high-quality rendering. The paper presents some of the recent advances in those areas of scientific visualization showing examples from computer aided engineering in the automotive industry like Lattice-Boltzmann based flow simulation and pre- and postprocessing in crash-worthiness analysis, as well as volume visualiza-

[*] Abteilung Visualisierung und Interaktive Systeme (VIS), Universität Stuttgart, Germany. E-Mail: Thomas.Ertl@informatik.uni-stuttgart.de

tion of chemical and medical datasets. It will be demonstrated that the proliferation of 3D graphics adapters for the PC and increasing network bandwidth will bring web-based visualization techniques to new application domains while at the same time high-end graphics solutions continue to be required for productive work in virtual reality installations.

4.1 Introduction

The ever growing size of data sets resulting from industrial and scientific simulations and measurements have created an enormous need for analysis tools allowing interactive visualization. Although processing speed and graphics hardware performance are still improving dramatically, additional very specialized algorithmic innovations and methodical developments are necessary in order to yield the overall performance improvements end users are demanding. Therefore, scientific visualization which started as an interdisplinary activity of engineers, physicists, mathematicians and other computational scientists grew into a research field of its own right during the last decade. Today, scientific visualization is well established within the field of computer science bridging the gap between simulation and computer graphics by means of dedicated conferences, journals and reputable research groups.

Formally, we describe scientific visualization as the process of generating a visual representation of the information contained in abstract data fields resulting from computer simulations or sensoric measurements. The standard model of this process comprises a pipeline of three stages (see Figure 4.1). The *filter* stage is a preprocessing step converting the raw input data into visualization data which is usually reduced by operations like sampling, slicing, cropping, etc. The *mapper* stage performs a mapping of the abstract data fields into a visual representation consisting of geometric primitives like points, lines, surfaces or voxels and associated graphical attributes like color, transparency, texture, etc. The *renderer*, finally, uses this scene description to generate images by means of 3D graphics APIs such as OpenGL or OpenInventor, possibly exploiting 3D graphics hardware to achieve interactive frame rates. Many different mapping algorithms have been developed for various scenarios. A crude classification of these methods distinguishes between the dimensionality of the data set, the underlying data type, such as scalar, vector, multivariate, and the supported grid structure, such as regular, curvi-linear or unstructured. While the generation of graphs and images from 1D and 2D data is usually well sup-

ported by traditional plotting software, advanced visualization techniques focus on data defined on 3D geometry (e.g. displacement values on the finite element mesh of a car body) or volumetric data (e.g. scalar 3D data given on Cartesian grids like from CT or MRI scanners or vectorial 3D data given on unstructured grids like velocities from CFD simulations). Additional complexity arises from time dependent data potentially given on grids with changing geometry (like in crash simulations) or even changing topology (like in combustion simulation).

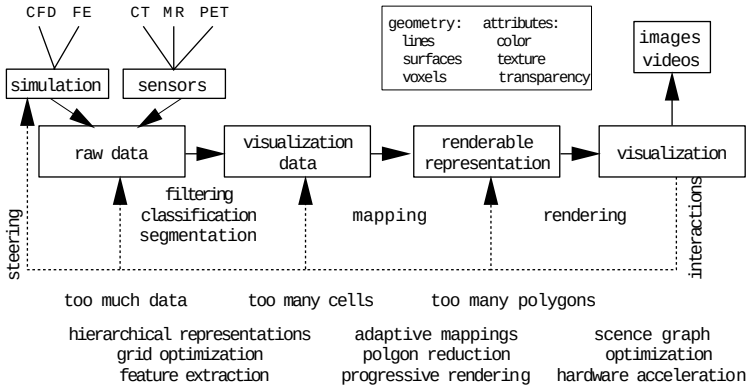


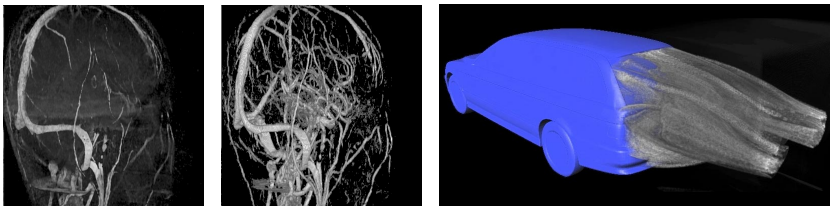
Figure 4.1: The basics stages of the visualization pipeline and the problems and approaches associated with visualizing large data sets.

Since those data sets are intrinsically huge, a lot of efforts have been undertaken during the last years to come up with optimized visualization algorithms. The goal is to develop algorithms which react to changes of mapping parameters (e.g. varying the iso-value of an iso-surface) by regenerating within seconds the corresponding geometrical representation which can then be rendered with several frames per second. Only with this type of real-time interaction and navigation is it possible to analyze an unknown data set and to compensate for the information lost during the projection of the 3D scene onto the screen. However, despite all the sophistication incorporated into these methods, does it seem that the data sets are growing faster than algorithmic progress is made. Therefore, all stages of the visualization pipeline have to be subject to optimization in order to be able to match our goal. In the following, some ideas and examples are described which demonstrate recent progess made in accelerating filtering, mapping and rendering of the visualization pipeline.

4.2 Data compression and feature extraction by multiresolution analysis

It is obvious that visualization methods which essentially have to access each cell of a volumetric data set containing millions of cells in order to derive a visual mapping might not catch up to the goal of interactive processing. Thus, we have to reduce the number of cells which have to be visually mapped, which means that we have to compress the data set in a preprocessing filtering step. Since we strive to reduce the number of cells by at least an order of magnitude, only lossy compression schemes will be employed. However, this does not always have to lead to a significant loss of information. On the contrary, the compression scheme will be chosen in such a way, that only redundant or irrelevant information (e.g. CT voxels containing air) is discarded, while important features like high gradients, edges, etc. are retained or even emphasized. An example for this is shown in Figure 4.2(b) where wavelet analysis was used to automatically segment brain vessels from a CT angiography [WE97]. Tracing wavelet maxima across various scales extracts the vessel walls and removes noise at the same time. A similar idea was used to visualize relevant flow structures from an aerodynamics simulation of a car [WJE00]. Time surfaces are integrated from the velocity field and their curvature is measured to separate principal streams. Successive smoothing removes fine-scale fluctuations making the prominent features visible in the semi-transparent rendering of a LIC volume as shown in Figure 4.2(c).

In any compression scheme an error is introduced and the user has to be given control over the threshold letting him choose between a fast visualization of a very crude approximation of the data and an almost perfect representation of the data which took perhaps minutes to compute. This requirement can only be met, if not only one compressed version of the data, but a complete hierarchy of representations of the data set at different levels of resolution is available or can be generated on the fly. There are various ways to derive such a hierarchy and they have all been applied in visualization ranging from simple octrees to wavelets and multilevel finite elements [GE98]. Even more, hierachical data structures are used in a variety of modern numerical methods. In order to visualize these results they have to be interpolated to standard grids before general purpose visualization packages can be applied. Examples where this is hardly possible and completely new visualization algorithms have to be developed are sparse grids [THGE99] or locally refined cartesian grids as they



(a) Standard volume rendering of CT angiography data

(b) Edge enhancement and noise reduction through wavelet analysis

(c) LIC volume rendering of principle aerodynamic streams behind a car

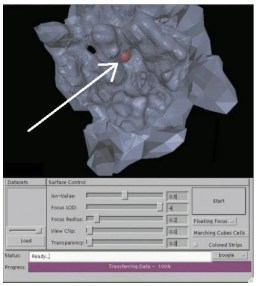
Figure 4.2: Feature extraction from scalar and vector data sets based on multiresolution analysis

are used in industrial Lattice-Boltzmann CFD codes [SRBE99].

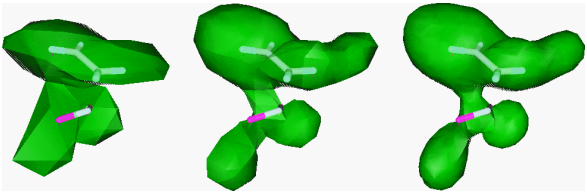
4.3 Adaptive and progressive mapping algorithms

Developing efficient algorithms for generating a multiresolution hierarchy of a volumetric data set by compressing redundant information with respect to an error measure is only one side of the visualization pipeline. On the mapping side we need just as efficient algorithms which can take advantage of the hierarchical data structures. It turns out that this is by no means a trivial task as in general the traversal of a hierarchy is slow compared to full grid algorithms which have been optimized over years. However, ideally, a mapping algorithm exploiting the hierarchical representation of the data fields will automatically generate geometrical representations at various levels of detail, thus allowing incremental and progressive rendering.

Examples of this are shown in Figure 4.3. On the left hand side, an octree hierarchy was built from a scalar cartesian volume resulting from cryo-electron microscopy of a ribosom of Escheria Coli. An iso-surface was reconstructed from various levels of the hierarchy depending on the distance of the octree leaves from a momentarily defined focus point. Special attention has to be paid to avoid holes in the iso-surface near level transitions. Real-time performance



(a) Adaptive refinement of a ribosom iso-surface around a focus point



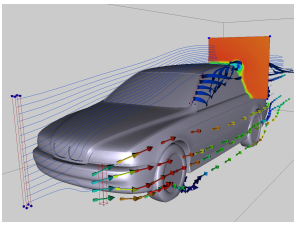
(b) Progressive refinement of the iso-surface of a molecular electron density

Figure 4.3: Adaptive and progressive visualization from hierarchical data representations

allows to interactively investigate details of the iso-surface which would have not been possible for the full resolution surface [WKE99]. On the right hand side, several levels of an adaptively refined tetrahedral mesh were generated from a high resolution regular volume of the electron density of a molecule. Due to the enormous reduction in the order of cells iso-surfaces can interactively be extracted from the unstructured mesh and progressively refined while descending the hierarchy [EGE98].

Besides the exploitation of hierarchical data structures for adaptive and progressive visualizations research still concentrates on generating new visual mappings for complex and derived quantities. Figure 4.4(b) shows an example for the fruitful combination of various particle probes for the interactive investigation of the air flow around a car body. While the rake of stream lines gives the overall picture, the ribbons are helpful in understanding the A-frame vortex and the glyphs carry additional information about the pressure mapped to the color [REB01]. One important result of front crash simulations is the information how much force is transmitted by the longitudinal strucures and how much energy is absorbed by those car body parts. A new way to visualize the temporal variation of the sectional forces at various positions is by using an additional geometry called force tube and by mapping those values to its radius

(a) The force flux during a front impact is visualized by means of an animated force tube with the radius and the color representing the cross sectional forces



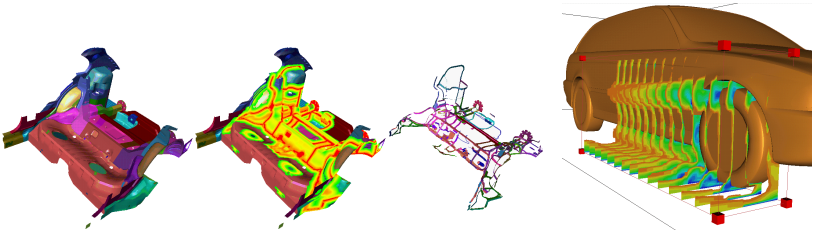
(b) The simulated air flow around a car is investigated by the interactive manipulation of a combination of stream lines, ribbons, and glyphs

Figure 4.4: Additional geometry allows for efficient mapping of complex relationsships

and its color [KHSE98] (see Figure 4.4(a)).

4.4 Exploiting advanced rasterization hardware

Despite the advances in innovative algorithms for filtering data sets and mapping the relevant entities to geometry and to visual variables visualization still requires hardware accelerated graphics to achieve interactive rates. Driven by the mass market of computer games the performance of graphics adapters has made fascinating progress during the last five years. Besides new peak rates for geometry processing in the order of several millions of lit and shaded triangles per second a drastic increase in rasterization functionality has to be noted. Especially texturing and blending of semi-transparent pixels is a commodity now available in almost any workstation and PC. Exploiting rasterization hardware for scientific visualization has brought interactivity to several well-known but expensive methods like direct volume rendering and has laid foundation for new approaches [WE98]. One example is shown in Figure 4.5(b) where the pressure values of a CFD simulation are mapped to colored iso-contours on several slice planes by means of a 1D texture map. Occlusion of the slice planes



(a) Visualizing potential flanges of the back compartment of a car by rendering only nearby geometry opaque

(b) Stacked slice planes showing contours of the pressure, transparency removes occlusion of the vortex

Figure 4.5: Exploiting rasterization functionality like texturing and transparency

is reduced by mapping uninteresting pressure values to transparent pixels. In Figure 4.5(a) transparency is used in assisting the user to place spot welds on flanges of a crash simulation finite element model. Distances between each of the elements are efficiently computed by a bounding box hierarchy. Elements which are closer than a certain distance are colored as potential flanges and the rest of the geometry is finally made transparent [SE00].

4.5 Conclusions

As the size of data sets resulting from simulation or measurement will be continuously increasing so is the need for interactive visualization techniques and tools. We have shown that multiresolution analysis is a promising approach for compressing data sets in order to arrive at a concise visualization of the relevant features. The hierachical data structures also form a basis for adaptive and progressive visualization algorithms to be used in web applications. Exploiting the wealth of graphics hardware functionality for new visualization methods is only at its beginning and will allow for the analysis of huge data sets in virtual environments. Finally, optimal visualization performance will only be achieved by a very tight adaption to the specific requirements of the simula-

tion. This demands for an even closer collaboration between the computational scientist and the visualization expert in the future.

References

[EGE98] K. Engel, R. Grosso, and T. Ertl. Progressive Iso-Surfaces on the Web. In *Late Breaking Hot Topics*. IEEE Visualization, 1998.

[GE98] R. Grosso and T. Ertl. Progessive Isosurface Extraction from Hierarchical 3D Meshes. *Computers Graphics Forum (EUROGRAPHICS '98)*, 17(3), September 1998.

[KHSE98] S. Kuschfeldt, M. Holzner, O. Sommer, and T. Ertl. Efficient Visualization of Crash-Worthiness Simulations. *IEEE Computer Graphics and Applications*, 18:60–55, 1998.

[REB01] S. Röttger, T. Ertl, and W. Bartelheimer. Automotive Soiling Simulation Based on Massive Particle Tracing. In *Procceedings of EG/IEEE Symposium on Visualization VisSym '01*, 2001.

[SE00] O. Sommer and T. Ertl. Geometry and Rendering Optimization for the interactive Visualization of Crash-Worthiness Simulations. In *Proceedings of the Visual Data Exploration and Analysis Conference in IT&T/SPIE Electronic Imaging*, pages 124–134, 2000.

[SRBE99] M. Schulz, F. Reck, W. Bartelheimer, and T. Ertl. Interactive Visualization of Fluid Dynamics Simulations in Locally Refined Cartesian Grids. In *Proc. Visualization '99*, pages 413–416. IEEE, 1999.

[THGE99] C. Teitzel, M. Hopf, R. Grosso, and T. Ertl. Volume Visualization on Sparse Grids. *Computing and Visualization in Science*, 2:47–59, 1999.

[WE97] R. Westermann and T. Ertl. A Multiscale Approach to Integrated Volume Segmentation and R endering. *Computers Graphics Forum (EUROGRAPHICS '97)*, 16(3):96–107, 1997.

[WE98] R. Westermann and T. Ertl. Efficiently Using Graphics Hardware in Volume Rendering Applications. *Computer Graphics (SIGGRAPH '98)*, 32(4):169–179, 1998.

[WJE00] R. Westermann, C. Johnson, and T. Ertl. A Level-Set Method for Flow Visualization. In *Proc. Visualization '00*, pages 147–152. IEEE, 2000.

[WKE99] R. Westermann, L. Kobbelt, and T. Ertl. Real-time exploration of regular volume data by adaptive reconstruction of isosurfaces. *The Visual Computer*, 15:100–111, 1999.

■

Chapter 5

Connectivity Coding: New Perspectives for Mesh Compression

Stefan Gumhold[*]

Compact encodings of the connectivity of planar triangulations is a very important subject not only in graph theory but also in computer graphics. For triangle meshes used in computer graphics the planar regions dominate by far. New results by Isenburg, Gumhold and Gotsman [IGG01] even show that the connectivity is sufficient to describe shape by itself. Most coding methods for planar triangulations can be extended to connectivities of manifold meshes with arbitrary topology. Upper and lower bounds on the bit rate are preserved if the maximum genus of the mesh is limited.

In 1962 Tutte determined the number of different planar triangulations. From his results follows that the encoding of the connectivity graph of planar triangulations with three border edges and v vertices consumes in the asymptotic limit for $v \to \infty$ at least $3.245v + o(\log(v))$ bits. The

[*] WSI/GRIS, Universität Tübingen, Germany. E-Mail: sgumhold@gris.uni-tuebingen.de

so far best compression method [Gum00] with guaranteed upper bounds for the bit rate is based on the encoding of $CRLSE$-Edge Breaker strings and improves the bit rate of 3.67 bits per vertex established by King and Rossignac [KR99]. In this article we elaborate on the coding technologies indicated in [Gum00] and improve the bit rate to less than 3.525 bits per vertex, while ensuring a linear run time for encoding and decoding.

5.1 Introduction

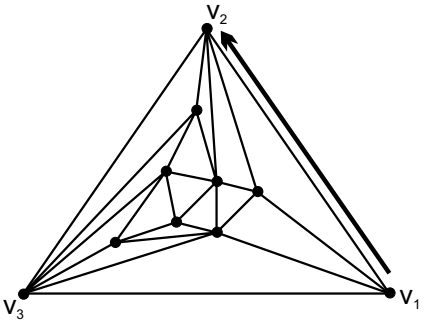


Figure 5.1: Sample triangulation with the three external vertices v_1, v_2 and v_3.

This article improves the lowest upper bound for the encoding of planar triangulations with three border edges as defined by Tutte in [Tut62]. Fig. 5.1 shows an example of a planar triangulation with the three border vertices v_1, v_2 and v_3. Two planar triangulations are defined to be equal, if there exists a bijection between their connectivity graphs that maps all border vertices of the first triangulation to the border vertices with the same indices of the second triangulation. Tutte enumerated all different planar triangulations and showed in this way that an optimal encoding uses at least 3.245 bits per vertex for a sufficiently large number of vertices. So far the best encoding schemes [CGHK98] and [Ros99] consumed 4 bits. The latter – the Edge Breaker scheme – could be improved to 3.67 bits per vertex [KR99].

Planar triangulations are a special case of closed manifold triangle meshes where the genus of the triangle mesh is zero. As most encoding schemes for planar triangulations can be extended to manifold triangle meshes with border,

the schemes are also important in the representation of surface models and have been studied extensively [Dee95, Hop96, Cho97, DS97, GS98, LK98a, LK98b, MC98, TR98, TGHL98, TG98, BPZ99b, BPZ99a, COLR99, IS99a, Ise00, PR00, AD01b, AD01a]. Latest work by Isenburg et al. [IGG01] shows that the connectivity contains enough information to describe shape.

The algorithmic scheme of the Edge Breaker [Ros99] is very simple and similar to the work of Itah [IR82] and to the Cut-Border Machine [GS98]. It visits the triangles of an edge-connected component of a triangle mesh in an order defined by the triangle connectivity itself. The same traversal is used for encoding and decoding[2] the connectivity – triangle by triangle – into one of the five operation symbols CLRSE for each triangle. By the use of code books, the Edge Breaker allows to encode the connectivity of typical triangle meshes to an average of 2.2 bits per triangle, whereas the Cut-Border Machine achieves with arithmetic coding and conditional probabilities an average of 1.9 bits per vertex [Gum99]. These results are only valid for regular meshes. For an arbitrary mesh the improved techniques cannot guarantee a good upper bound.

In the following we extend the techniques developed in [Gum00] and give a linear runtime encoding and decoding scheme with an upper bound of 3.525 bits per vertex. Section 5.2 reviews the Edge Breaker encoding scheme with a small modification on the split operation. If the reader is familiar with the Edge Breaker coding it is sufficient to just review the descriptions of the split operations. In section 5.3 we describe constraints on the symbol stream produced by the Edge Breaker encoding, that can be exploited to reduced the worst case bit rate. The next two sections 5.4 and 5.5 describe techniques to exploit the constraints on the symbol stream. Numerical issues are discussed in section 5.6. After the results in section 5.7 we give concluding remarks in section 5.8.

5.2 Edge Breaker Coding

The Edge Breaker translates the connectivity of a planar triangulation into a sequence of five different symbols. The encoding algorithm is a region growing algorithm, which stores at any time all vertices and edges of the planar triangulation, which divide the so far encoded triangles from the not yet encoded triangles. These vertices and edges form a set of closed loops, which is called the *cut-border*. Before encoding starts, the cut-border is initialised to one loop containing the external edges and vertices.

[2] where decoding is done in reverse direction

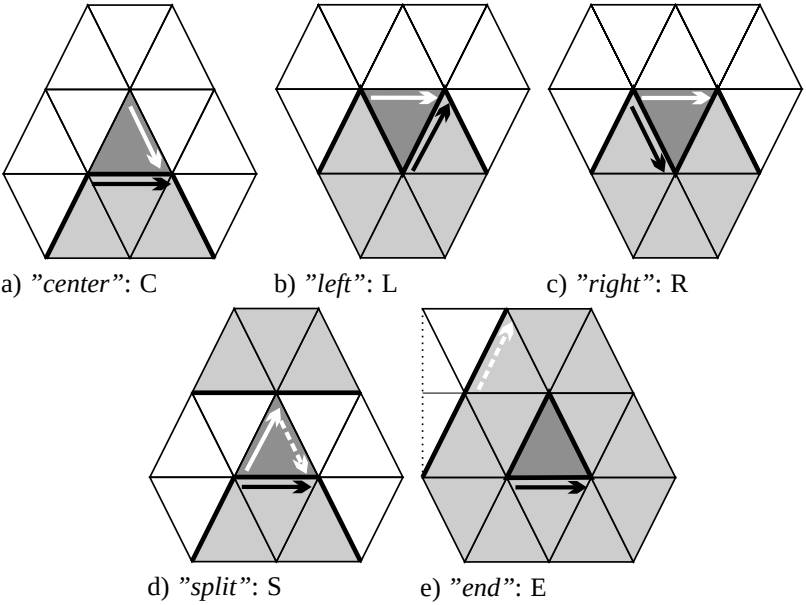


Figure 5.2: The five different Edge Breaker operations needed for the encoding of planar triangulations together with the corresponding symbols. The so far encoded triangles are shaded in a light grey, the cut-border edges are bold black as is the gate before the operation, the currently encoded triangle is dark grey and the new gate(s) after the operation is/are white.

Triangles are encoded at a specific cut-border edge which is called the *gate*. Each time a triangle has been encoded at the gate location, the gate is set to another cut-border edge in a predetermined way such that all cut-border edges will be visited in the end. In the beginning the gate is set to the external edge v_1v_2. Fig. 5.2 shows the different operations and their symbols that we collect in the Edge Breaker alphabet

$$\mathcal{A} \stackrel{\text{def}}{=} \{C, L, R, S, E\}$$

- *"centre"* (C): The newly encoded triangle is formed upon the gate with a new vertex. After the operation the gate is set to the right newly introduced cut-border edge, such that the gate cycles around the vertex it points to.

- *"left"* (L): The currently encoded triangle connects the gate to the previous edge on the cut-border. After the current triangle is encoded the gate is set to the only newly added cut-border edge.
- *"right"* (R): The mirror image of the left operation. The currently encoded triangle connects the gate to the next edge on the cut-border.
- *"split"* (S): The currently encoded triangle splits the current loop of the cut-border into two loops. For each loop the gate is chosen as the newly introduced cut-border edge incident to the loop. The loop corresponding to the left new edge is processed next, the other loop is pushed together with its gate onto a loop stack. In the original scheme the loop incident to the right edge is processed next as in the centre operation. Our modification only exchanges the two parts of the symbol stream that encode the two loops after the split operation. The modification will help us later on to account for some of the constraints more easily.
- *"end"* (E): The end operation eliminates the current cut-border loop, pops a loop from the loop stack and activates its gate. If no more loop is on the loop stack the current connected component is completely encoding.

The encoding permutes the vertices. The sample triangulation in Fig. 5.1 is encoded to the string

$$CCCRCCCRRRCSREE,$$

when the edge v_1v_2 is used as initial gate. The only operation which cannot be decoded by simple repetition of the encoding operation is the split operation. Due to Isenburg [IS99b] is the observation that the inverse operations of the Edge Breaker operations can be done without any further knowledge. The decompression is therefore done in reverse order. The symbol string is scanned from back to front. The different operations are performed in reverse order. In this way the different cut-border loops, which have been generated by split operations, are re-built in reverse order and accumulated on a loop stack. Each split operation merges two loops from the loop stack. In more detail the inverse operations work as follows. For this have another look at Fig. 5.2, where the white triangles are the so far decoded part, the dark grey triangle is decoded next and the grey triangles have not been encoded yet. The white arrow(s) is/are the gate(s) before the inverse operation and the black arrow is the gate after the inverse operation.

- *"inverse centre"* (C^{-1}): The newly decoded triangle connects the gate to the previous edge on the cut-border and is the mirror image of the forward L operation.
- *"inverse left"* (L^{-1}): The gate is replaced by the newly decoded triangle and the gate is set to the right newly introduced cut-border edge as in the forward C operation.
- *"inverse right"* (R^{-1}): The gate is replaced by the newly decoded triangle and the gate is set to the other newly introduced cut-border edge as in the L^{-1} operation.
- *"inverse split"* (S^{-1}): The current loop is merged with the top loop on the loop stack. The gate is set to the only newly introduced cut-border edge. The inverse split operation unifies two vertices from the two different loops into one vertex, where the two gates meet. This implies that the final indices of the vertices are not known at their creation time during decoding. When a vertex is created during decoding it receives a dummy index that can be unified with another dummy index after an inverse split operation. The index can be finalized whenever all triangles incident to a vertex have been decoded. This happens after each inverse centre operation. Thus the final vertex index 0 is given to the first vertex whose neighbourhood is closed by an inverse centre operation. The index 1 to the second vertex and so on. In this way the vertices are enumerated during decoding in exactly the reverse order as they are introduced by centre operations in the encoding process.
- *"inverse end"* (E^{-1}): The inverse end operation creates a new cut-border loop consisting of three edges and three vertices with dummy indices. For this the current loop is pushed onto a loop stack.

As all forward and inverse operations can be performed in constant time, we can state the following theorem:

Theorem 5.2.1 *The connectivity of a planar triangulation with v vertices and 3 external edges can be encoded with a unique string of length $2v$ over five different symbols in linear time in v. The original connectivity can be decoded also in linear time in v.*

By encoding the C-symbol with one bit and all other symbols with three bits, the Edge Breaker scheme allows to encode any planar triangulation with

no more than four bits per vertex[3]. In [KR99] the upper bound for the storage space is improved to 3.67 bits per vertex. Gumhold [Gum00] improved upon this result to an upper bound of 3.552. In the next chapters we show how to achieve an even better bound of 3.525 bits.

5.3 Constraints

The upper bound of four bits per vertex can be improved as not all possible symbol strings are allowed. In this section we gather the different constraints on the symbol strings. During encoding we first produce the symbol string. Then we reverse the order of the symbols and the vertex data as a preparation for decoding. The reversed symbol string is finally transformed into a bit stream via arithmetic coding as described in the next section. The decoding algorithm uses arithmetic decoding to get back the symbol string and builds the connectivity in the given previously reversed order. To improve the arithmetic coding stage we investigate the reversed symbol string and examine for each inverse operation the necessary preconditions that will ensure the reconstruction of a planar triangulation and nothing else:

1. *"inverse centre"* (C^{-1}): The inverse centre operation removes the gate and the previous edge from the cut-border and adds a new edge connecting their far apart end points. This operation can cause two invalid configurations.

   (a) The removal of one cut-border edge can reduce the length of the current cut-border loop to less than the minimal number of three edges.

   (b) The newly added edge could have been part of the decoded connectivity before the inverse centre operation. This is for example always the case if the inverse centre operation is performed after an inverse left operation. Then the inverse centre operation would add a third triangle to the interior edge introduced by the left operation. But in a planar triangulation an edge with more than two edges is not allowed.

[3] Each symbol introduces one triangle. There are twice as many triangles as vertices. Each vertex corresponds to exactly one C symbol. This sums up to $v + 3v = 4v$ bits.

2. *"split"* (S^{-1}): An inverse split operation is only allowed if there is a cut-border loop on the loop stack.

3. *"inverse left / right / end"* (L^{-1} / R^{-1} / E^{-1}): The inverse left, right and end operations all introduce new cut-border edges. Therefore, they are not allowed at the end of the decoding process when the remaining inverse split and centre operations cannot reduce the number of cut-border edges to the number of external edges.

The next two sections describe how to account for the constraints 1a and 1b on the inverse centre operations. We do not take into account the other two constraints. The constraint 2 implies that during decoding the number of E operations must be at anytime larger than the number of S operations. To understand this constraint better, we virtually remove all C,L and R symbols from the sequence of Edge Breaker symbols. Then we replace each S symbol by an open parenthesis and each E symbol by a closed parenthesis. In the setting with the parentheses constraint 2 is fulfilled, iff the parentheses are balanced. As the number of balanced strings of parentheses is only half the number of arbitrary strings over two symbols we can only save one bit for the encoding of the planar triangulation. As this bit does not help anything in terms of bits per vertex, we can safely neglect constraint 2.

The last constraint 3 can be neglected with a similar argument. Each inverse end operation is paired with an inverse split operation. The inverse end operation introduces three cut-border edges and the inverse split operation removes one. A pair of inverse end and split symbols introduces two cut-border edges, i.e. one new cut-border edge per symbol. Thus each of the inverse operations for the symbols LRSE introduces one cut-border edge and only the inverse centre operation removes one cut-border edge. If we replace each of the symbols LRSE with an open parenthesis and each C symbol with a closed parenthesis and claim that the resulting string of parentheses must be balanced, we restrict the symbol string even more than constraint 3 demands. But again this restriction can only save one bit for the encoding of the planar triangulation and we can also neglect constraint 3.

5.4 Conditional Unities

In order to achieve near optimal compression rates and to have at the same time the flexibility to avoid the constraints on the inverse centre operations,

we use arithmetic coding to transform the symbol stream into a bit stream (see [WNC87] for an introduction to arithmetic coding). To encode a symbol, the arithmetic coder subdivides the unit interval into as many sub-intervals as there are different symbols, in our case into five or four [4] sub-intervals. The sizes of the sub-intervals are chosen equal to the probabilities of each of the allowed symbols. A specific symbol from the set of possible symbols is encoded by selecting the corresponding interval. To encode the next symbol the arithmetic coder uses the selected interval as basis and splits this again according to the probabilities into sub-intervals and so on. For the encoding of each symbol one can specify different probabilities for the symbols depending on the history of the so far encoded or decoded symbol string. The symbol string is transformed by the arithmetic coder through an interval subdivision into a very tiny interval uniquely specifying the symbol string. This tiny interval is then encoded with the shortest binary fraction that represents a number within the sub-interval. The overhead for specifying the binary fraction is very small and one can assume that the arithmetic coder does an optimal job.

Suppose we are given the Edge Breaker symbol string $S = s_1 s_2 \dots s_t$ in reverse order, where t is the number of operation symbols which is equal to the number of triangles. Then we can specify for each symbol s_i five probabilities $p_{i,\alpha}$ for $\alpha \in \{C, L, R, S, E\}$, that may depend on the previous symbols: $p_{i,\alpha}(s_1 \dots s_{i-1})$. Each symbol s_i shrinks the current interval by a factor of p_{i,s_i}. Thus the size of the final interval is

$$I_t = \prod_{i=1}^{t} p_{i,s_i}.$$

To specify this interval with a binary fraction we need no more than

$$\mathcal{B}(S) \stackrel{\text{def}}{=} \log_2 \frac{1}{I_t} + 1$$

bits. If we split the logarithmic term into a sum, we see that each symbol contributes exactly

$$\mathcal{B}(s_i) \stackrel{\text{def}}{=} \log_2 \frac{1}{p_{i,s_i}}$$

fractional bits. Thus it makes sense to say that a specific symbol consumes for example 2.525 bits. In the simplest encoding of the Edge Breaker symbols the

[4] if no C symbol is allowed

probabilities of the symbols CLRSE are set to $\frac{1}{2}$ for C and $\frac{1}{8}$ for the other four symbols, corresponding to one and three bits. From Euler's equation we know that the number of triangles t is equal to $2v - m - 1$, where v is the number of vertices and m the number of external edges. As each C operation introduces one new vertex, the number of C symbols is equal to $v - m$. The number of the remaining symbols is $v - 1$, such that the consumed storage space for the simple coding sums up to $v - m$ plus $3 \cdot (v - 1)$ equals $4v - m - 3$ bits, what is less than four bits per vertex.

There is no way around the fact that the C symbols constitute about half of the symbols. Thus we can fix the probability of the C symbols to $\frac{1}{2}$. For all the remaining symbols we assume a fixed probability of τ, which should be larger than $\frac{1}{8}$ in order to achieve a better bit rate per vertex. In order to keep τ as the constant that tells us the total bit rate via the formula

$$b(S) \stackrel{\text{def}}{=} \frac{\mathcal{B}(S)}{v} = 1 + \log_2 \frac{1}{\tau}, \tag{1}$$

we introduce the concept of the *conditional unity*. Without any knowledge of the decoding process, we know that the probabilities of the different symbols must sum up to one

$$1 = p_C + p_L + p_R + p_S + p_E \stackrel{?}{=} \frac{1}{2} + 4\tau.$$

If the second equality would be true, we could compute τ to $\frac{1}{8}$ and would end up with a bit rate of four bits per vertex. But here we neglected the fact, that after we saw an E symbol in the reversed symbol string, no C symbol may follow because it would reduce the number of cut-border edges to two, what is not allowed. Thus the probabilities of the symbols that are allowed under the precondition that the previous symbol has been an E do not sum up to one. Instead they will sum up to a number smaller than one that we denote as the conditional unity ${}^{E}\mathbf{1}$ under precondition that an E symbol preceded the current symbol. The C symbol may neither follow a L symbol because it would destroy the planarity of the triangulation. We can now revise our first equation on the probabilities to three new ones:

$$\begin{aligned} \mathbf{1} &= \frac{1}{2} + \tau\left({}^{L}\mathbf{1} + \mathbf{1} + \mathbf{1} + {}^{E}\mathbf{1}\right) \\ {}^{L}\mathbf{1} &= \tau\left({}^{L}\mathbf{1} + \mathbf{1} + \mathbf{1} + {}^{E}\mathbf{1}\right) \\ {}^{E}\mathbf{1} &= \tau\left({}^{L}\mathbf{1} + \mathbf{1} + \mathbf{1} + {}^{E}\mathbf{1}\right). \end{aligned}$$

We see at once that the conditional unities ${}^{L}\mathbf{1}$ and ${}^{E}\mathbf{1}$ are the same resulting in only two equations:

$$\begin{aligned} \mathbf{1} &= \frac{1}{2} + 2\tau\left({}^{L}\mathbf{1} + \mathbf{1}\right) \\ {}^{L}\mathbf{1} &= 2\tau\left({}^{L}\mathbf{1} + \mathbf{1}\right) \end{aligned}$$

We can easily solve for ${}^{L}\mathbf{1} = \frac{1}{2}$ and $\tau = \frac{1}{6}$. Thus the bit rate would be $1 + \log_2 6 < 3.585$ bits per vertex. The corresponding arithmetic coder has two different lists of probabilities $\left(p_{1/2,\alpha}\right)$ – if no precondition holds $\left(\frac{1}{2}, \frac{1}{12}, \frac{1}{6}, \frac{1}{6}, \frac{1}{12}\right)$ and under precondition of an E or L $\left(0, \frac{1}{12}, \frac{1}{6}, \frac{1}{6}, \frac{1}{12}\right)$.

5.5 The State Machine

With the tool of conditional unities we can exploit the constraints 1a and 1b on page 73.

Constraint 1a does not allow a C symbol if the length of the current cut-border loop is three. Every time when an inverse end operation generates a new loop, we know that the length of the current loop is three afterwards. To remember this knowledge we introduce the conditional probabilities $\mathbf{1}_{\,l}$, where l specifies the known length of the current loop. Each inverse left and right operations increment the current cut-border loop by one, whereas the inverse C operation decrements it by one. After the inverse split operation we don't know anything about the length of the current loop as we didn't remember the length of the loop on the stack. Considering only the length of the current loop, the equations for the conditional unities are

$$\begin{aligned} \mathbf{1} &= \frac{1}{2} + \tau\left(3 \cdot \mathbf{1} + \mathbf{1}_3\right) \\ \mathbf{1}_3 &= \tau\left(2 \cdot \mathbf{1}_4 + \mathbf{1} + \mathbf{1}_3\right) \\ \mathbf{1}_l &= \frac{1}{2}\mathbf{1}_{l-1} + \tau\left(2 \cdot \mathbf{1}_{l+1} + \mathbf{1} + \mathbf{1}_3\right) \quad \forall l > 3. \end{aligned}$$

We can extend this approach by also remembering the length p of the loop on top of the loop stack by introducing the conditional unities $\mathbf{1}_{\,l}^{\,p}$. This will allow to determine the loop length after an inverse split operation. Then the current loop will have the length $l + p - 1$. The second new rule is that after an inverse

end operation the current loop length is stored in the length of the loop on top of the stack resulting in the equations

$$\begin{aligned}
\mathbf{1} &= \frac{1}{2} + \tau\left(3 \cdot \mathbf{1} + \mathbf{1}_3\right) \\
\mathbf{1}_3 &= \tau\left(2 \cdot \mathbf{1}_4 + \mathbf{1} + \mathbf{1}_3^3\right) \\
\mathbf{1}_l &= \frac{1}{2}\mathbf{1}_{l-1} + \tau\left(2 \cdot \mathbf{1}_{l+1} + \mathbf{1} + \mathbf{1}_3^l\right) \quad \forall l > 3 \\
\mathbf{1}_3^p &= \tau\left(2 \cdot \mathbf{1}_4^p + \mathbf{1}_{2+p} + \mathbf{1}_3^3\right) \quad \forall p \geq 3 \\
\mathbf{1}_l^p &= \frac{1}{2}\mathbf{1}_{l-1}^p + \tau\left(2 \cdot \mathbf{1}_{l+1}^p + \mathbf{1}_{l+p-1} + \mathbf{1}_3^l\right) \quad \forall l > 3 \forall p \geq 3.
\end{aligned}$$

Finally, we introduce the conditional unities ${}_s\mathbf{1}_l^p$ that can also remember the length s of the loop on the second highest position on the loop stack. The equations are extended in the same way.

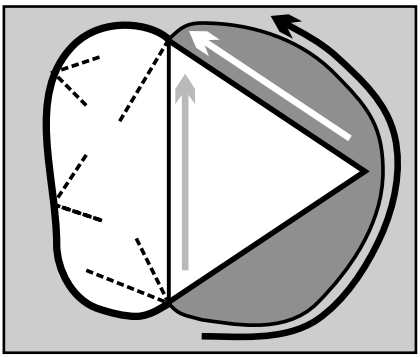


Figure 5.3: Illustration of constraint 1b on page 73.

The second constraint 1b on page 73 says that the edge introduced by the inverse centre operation is not allowed to be present in the so far decoded part. Fig. 5.3 illustrates the second constraint on the C symbol. The so far decoded part is shown in white, the not yet decoded part in grey. The gate before the inverse centre operation is the white arrow, the gate after the operation the black arrow. The dark grey shaded triangle with the bent edge is the currently decoded triangle. The bent edge introduced by the inverse centre operation is the same as the edge cutting the so far decoded part into a triangle on the right

and an arbitrary part on the left. Thus the newly added edge would coincide with the interior edge, which would be incident to the white triangle, the dark grey triangle and another triangle of the unspecified so far decoded part. This situation can arise after an inverse left operation. Suppose in Fig. 5.3 that the gate has been the grey arrow before the inverse centre operation and then the white triangle has been encoded by an inverse left operation. Thus every time an inverse left operation has been performed an inverse centre operation is not allowed to follow.

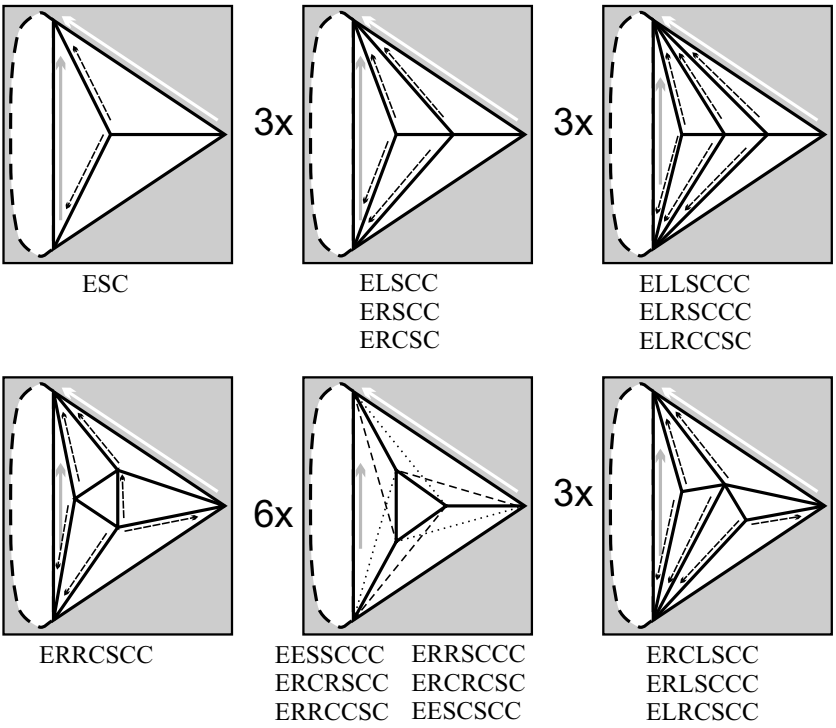


Figure 5.4: Different triangulations inside a triangle with one, two or three interior vertices.

But that is just the simplest scenario for constraint 1b. Inside the white triangle in Fig. 5.3 can be further vertices forming a more complicated triangulation. Fig. 5.4 illustrates all different triangulations with up to three interior

vertices, together with the reversed sequence of operation symbols that will produce this situation.

Let us check the simplest one in the top row on the left side of Fig. 5.4 by first encoding in forward direction. The white arrow gives the gate before any of the white triangles are encoded. The cut-border consists of the outline of the white area. A centre operation introduces the vertex in the middle of the big white triangle and moves the gate to the dashed arrow in the newly added triangle. Then a split operation follows that splits the cut-border into two loops, the dashed one on the left and the single triangle on the bottom right. Here comes into play why we modified the split operation such that the left loop is encoded next. Because now we go on with the loop inside the big white triangle first and encode the remaining triangle with an end operation. The reversed symbol sub-string that encodes the connectivity inside the big white triangle is ESC as written under the top left drawing in Fig. 5.4. The modification of the split operation will work for any triangulation inside the big white triangle, because the forward symbol sequence will always start with centre operations followed by a split operation that splits of the loop outside the big white triangle. One can easily check that no other split, no left, no right and no end operations can arise in forward encoding direction before this split operation.

In Fig. 5.4 we did not draw all 17 cases but wrote the number of similar case to the left of one representative. For representatives with three cases, the other two cases result from the representatives through rotation of the connectivity inside the big triangle by an angle of 120° and 240°. In the representative we drew the gate locations during forward encoding for the first symbol string under the drawing. The representative with six different cases is drawn with three quadrilaters each of which can be triangulated either with the dotted or the dashed line. All different choices of triangulations would result in eight cases but the cases with only dotted lines or only dashed lines are the same as the case on the left side in the second row of Fig. 5.4. If we add the two one symbol constraints E and L, we end up with 19 constraints. Let $\mathcal{C}$ be the set of all constraints

$$\begin{aligned}\mathcal{C} \quad \stackrel{\text{def}}{=} \{ \quad & E, L, ESC, ELSCC, ERSCC, ERCSC, ELLSCCC, \\ & ELRSCCC, ELRCCSC, ERRCSCC, EESSCCC, \\ & ERCRSCC, ERRCCSC, ERRSCCC, ERCRCSC, \\ & EESCSCC, ERCLSCC, ERLCSSS, ELRCSCC \quad \}.\end{aligned}$$

Next we define two predicates on Edge Breaker symbol strings

$$\begin{aligned} \forall S \in \mathcal{A}^* : \quad & \text{isConstraint}(S) \quad \Leftrightarrow S \in \mathcal{C} \\ \forall S \in \mathcal{A}^* : \quad & \text{isPrefix}(S) \quad \Leftrightarrow \exists H \in \mathcal{C} | S = H_{1..|S|}, \end{aligned}$$

where $H_{1..l}$ denotes the string composed of the first l symbols of the string H.

To ensure constraint 1b we define conditional unities of the form ${}^{H}_{s}\mathbf{1}^{p}_{l}$, where H is a string that specifies the history of symbols decoded before. For example the conditional unities ${}^{ESC}_{4}\mathbf{1}^{3}_{5}$ remembers the knowledge that the last encoded symbol was a C, the one before a S and the one before an E. Further more it remembers that the current loop length is 5, the length of the loop on top of the loop stack is 3 and the length of the next lower loop on the stack is 4. The knowledge that the previous symbols have been ESC allows us to discard the C symbol from the allowed symbols for this conditional unity.

Only strings H, for which "isPrefix(H)" holds, are useful to be remembered in the conditional unities. But for the loop lengths there is no natural limit such that we have to introduce artificial limits $l_{\max}$, $p_{\max}$ and $s_{\max}$. With these limits we can produce all the conditional unities with the following algorithm

1. push $\mathbf{1}$ onto a stack of unities and add it to the set of unities
2. while the stack of unities is not empty
3. pop a unity ${}^{H}_{s}\mathbf{1}^{p}_{l}$ from the stack
4. the C symbol can follow, iff $l \neq 3 \wedge \neg$isConstraint(H)
5. for all symbols α out of LRSE and C, if it can follow
6. find the unity ${}^{H'}_{s'}\mathbf{1}^{p'}_{l'}$ that incorporates α into the preconditions
7. if ${}^{H'}_{s'}\mathbf{1}^{p'}_{l'}$ is not in the set of unities, add it and push it onto the stack

Only the line 6 needs further clarification. The conditional unity ${}^{H'}_{s'}\mathbf{1}^{p'}_{l'}$ that follows the conditional unity ${}^{H}_{s}\mathbf{1}^{p}_{l}$, if the symbol α is encoded, can be found in

two stages. First we incorporate α into the fields via the "proceed" function

$$\begin{aligned}
\text{proceed}({}_s^H\mathbf{1}_l^p, C) &\stackrel{\text{def}}{=} \text{correct}({}_s^{H.C}\mathbf{1}_{l-1}^p)\\
\text{proceed}({}_s^H\mathbf{1}_l^p, L) &\stackrel{\text{def}}{=} \text{correct}({}_s^{H.L}\mathbf{1}_{l+1}^p)\\
\text{proceed}({}_s^H\mathbf{1}_l^p, R) &\stackrel{\text{def}}{=} \text{correct}({}_s^{H.R}\mathbf{1}_{l+1}^p)\\
\text{proceed}({}_s^H\mathbf{1}_l^p, S) &\stackrel{\text{def}}{=} \text{correct}({}_s^{H.S}\mathbf{1}_{l+p-1}^s)\\
\text{proceed}({}_s^H\mathbf{1}_l^p, E) &\stackrel{\text{def}}{=} \text{correct}({}_p^{H.E}\mathbf{1}_3^l),
\end{aligned}$$

where the "." operator appends the symbol α to the history H. The second stage is the function "correct" which makes sure that the new conditional unity is useful. If any of the lengths l, p or s exceeds the corresponding maximum $l_{\max}$, $p_{\max}$ or $s_{\max}$, "correct" removes the length from the preconditions. While isPrefix(H) does not hold for the history, "correct" removes the first symbol of the history.

With the described algorithm we cannot only find all reachable conditional unities but also their defining equations

$$\begin{aligned}
{}_s^H\mathbf{1}_l^p &= \frac{1}{2}\text{proceed}({}_s^H\mathbf{1}_l^p, C) + \tau\left(\text{proceed}({}_s^H\mathbf{1}_l^p, L) + \right.\\
&\quad \text{proceed}({}_s^H\mathbf{1}_l^p, R) + \text{proceed}({}_s^H\mathbf{1}_l^p, S) +\\
&\quad \left.\text{proceed}({}_s^H\mathbf{1}_l^p, E)\right), \qquad \text{if } l \neq 3 \wedge \neg\text{isConstraint}(H)\\
{}_s^H\mathbf{1}_l^p &= \tau\left(\text{proceed}({}_s^H\mathbf{1}_l^p, L) + \right.\\
&\quad \text{proceed}({}_s^H\mathbf{1}_l^p, R) + \text{proceed}({}_s^H\mathbf{1}_l^p, S) +\\
&\quad \left.\text{proceed}({}_s^H\mathbf{1}_l^p, E)\right), \qquad \text{if } l = 3 \vee \text{isConstraint}(H), \qquad (2)
\end{aligned}$$

which on the one hand define a set of equations that yield τ and from τ via equation 1 the bit rate in bits per vertex. On the other hand it defines the arithmetic coder in form of a state machine. Each conditional unity ${}_s^H\mathbf{1}_l^p$ is a different state of the arithmetic coder. The corresponding defining equation 2 tells the coder how to split the current interval into pieces. The symbol probability of the symbol α under the preconditions remembered by the conditional unity ${}_s^H\mathbf{1}_l^p$ is

$$\begin{aligned}
p_{\alpha,{}_s^H\mathbf{1}_l^p} &\stackrel{\text{def}}{=} \frac{\frac{1}{2}\text{proceed}({}_s^H\mathbf{1}_l^p, \alpha)}{{}_s^H\mathbf{1}_l^p} \quad \text{if } \alpha = C\\
p_{\alpha,{}_s^H\mathbf{1}_l^p} &\stackrel{\text{def}}{=} \frac{\tau\text{proceed}({}_s^H\mathbf{1}_l^p, \alpha)}{{}_s^H\mathbf{1}_l^p} \quad \text{if } \alpha \neq C.
\end{aligned}$$

After the symbol α has been encoded, the arithmetic coder changes into the new state "proceed$({}_s^H \mathbf{1}_l^p, \alpha)$".

5.6 Numerical Solution

Let us collect for a given number of constraints and given maximum lengths $l_{\max}$, $p_{\max}$ and $s_{\max}$ all the different conditional unities in the vector $\vec{\mathbf{1}}$, such that the first component $\vec{\mathbf{1}}_1$ equals to $\mathbf{1}$. Let n be the dimension of $\vec{\mathbf{1}}$. Then the defining equations 2 are of the form

$$\vec{\mathbf{1}} = M(\tau)\vec{\mathbf{1}} \qquad M(\tau) \in I\!R^{n \times n},$$

with a sparse square matrix M, that depends on τ. This equation is of the form of an eigenvalue problem. The difference is that the eigenvalue is known to be one, but the matrix depends on the parameter τ. We are looking for a τ within $[\tau_{\min}, \tau_{\max}]$ corresponding to the bit rates 4 and 3.245. Once we have found the correct τ_1, the matrix $M(\tau_1)$ has an eigenvalue of 1. Furthermore all entries auf $M(\tau_1)$ are less or equal to 1, which implies that 1 is the largest eigenvalue of $M(\tau_1)$. If we choose a τ_+ larger than τ_1, we underestimate the bit rate and all entries of the matrix $M(\tau_+)$ either increase or stay the same, such that also the largest eigenvalue has to increase or stay the same. Similarly, for a τ_- smaller than τ_1 the eigenvalue is less or equal to 1. Thus the largest eigenvalue of $M(\tau)$ is a monotonous function of τ. This enables us to apply a simple interval subdivision to find τ_1 starting with the interval $[\tau_{\min}, \tau_{\max}]$.

To compute the largest eigenvalue we exploit the fact that a multiplication of $M(\tau)$ with a n-dimensional vector can be done in $O(n)$ flops. We use the standard method to compute the eigenvector $\vec{v}_{\max}(M)$ corresponding to the largest eigenvector of M. We initialise $\vec{v}$ to ones. Then we repeat to set $\vec{v} \stackrel{\text{def}}{=} M\vec{v}/(M\vec{v})_1$ until only the component in direction of the eigenvector corresponding to the largest eigenvalue remains in $\vec{v}$ and the iteration does not change $\vec{v}$ anymore. The searched eigenvalue is $(M\vec{v})_1$. The method converges after a number of iterations that depends linear in the precision of the used floating point numbers and therefore is constant. Also the interval subdivision finds τ_1 in machine precision in a constant number of iterations, such that the overall runtime is linear in n.

After τ_1 is found, the conditional unities are given in the vector $\vec{v}(M(\tau_1))$. Before we use them to define the state machine of the arithmetic coder, we want to throw away conditional unities that correspond to the same state. A

conditional unity can be is not necessary if there is another conditional unity of the same value with the same defining equation. Thus we first sort the conditional unities by their value and their entries in the defining equation. Then we select for all successive conditional unities with the same value and defining equation one representative and replace all others in the defining equations by the representative. In this way we can define a state machine with a minimal number of states.

5.7 Results

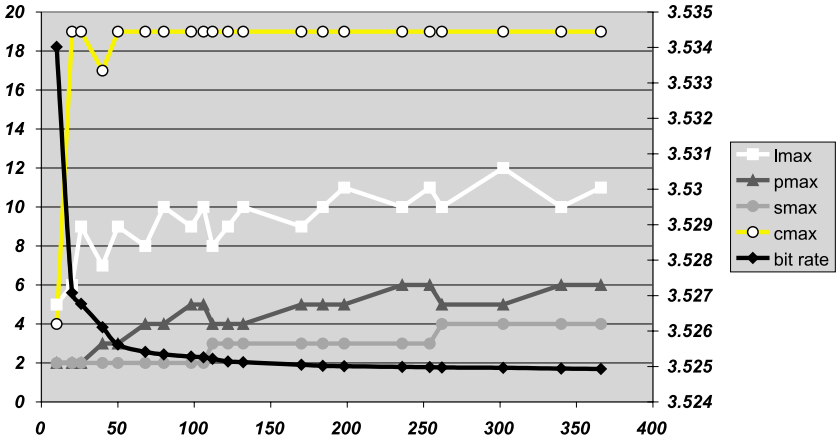


Figure 5.5: For given maximum number of variables the smallest bit rates plotted together with the set of limit parameters

In order to analyse the effect of the constraints versus the different cut-border loop lengths, we also restrict the number of exploited constraints to $c_{\max} \in \{0, \ldots, 19\}$. We call the numbers $l_{\max}$, $p_{\max}$, $s_{\max}$ and $c_{\max}$ the *limit parameters*. To find the best values for the limit parameters we implemented an optimisation method, that finds for a given maximum number of states the limit parameters that produced the smallest bit rate. Figure 5.5 plots the best bit rate and the limit parameters over the number of necessary states. The scale of the best bit rate is shown on the right. The rate drops off in an exponential manner converging to something below 3.525 bits per vertex.

unity	value	C	L	R	S	E
$\mathbf{1}$	1	$\mathbf{1}$	$^{ERRSCCC}\mathbf{1}$	$\mathbf{1}$	$\mathbf{1}$	$^{E}\mathbf{1}_{3}$
$\mathbf{1}_{9}$	0.984983	$\mathbf{1}_{8}$	$^{ERRSCCC}\mathbf{1}$	$\mathbf{1}$	$\mathbf{1}$	$^{E}\mathbf{1}_{3}$
$\mathbf{1}_{8}$	0.969957	$\mathbf{1}_{7}$	$^{ERRSCCC}\mathbf{1}$	$\mathbf{1}_{9}$	$\mathbf{1}$	$^{E}\mathbf{1}_{3}$
$\mathbf{1}_{7}$	0.945118	$\mathbf{1}_{6}$	$^{L}\mathbf{1}_{8}$	$\mathbf{1}_{8}$	$\mathbf{1}$	$^{E}\mathbf{1}_{3}$
$^{ERLS}\mathbf{1}$	0.9375	$^{ERS}\mathbf{1}$	$^{ERRSCCC}\mathbf{1}$	$\mathbf{1}$	$\mathbf{1}$	$^{E}\mathbf{1}_{3}$
$\mathbf{1}_{6}$	0.901554	$\mathbf{1}_{5}$	$^{L}\mathbf{1}_{7}$	$\mathbf{1}_{7}$	$\mathbf{1}$	$^{E}\mathbf{1}_{3}$
$^{ERS}\mathbf{1}$	0.875	$^{ERLSCC}\mathbf{1}$	$^{ERRSCCC}\mathbf{1}$	$\mathbf{1}$	$\mathbf{1}$	$^{E}\mathbf{1}_{3}$
$\mathbf{1}_{5}$	0.824112	$\mathbf{1}_{4}$	$^{L}\mathbf{1}_{6}$	$\mathbf{1}_{6}$	$\mathbf{1}$	$^{E}\mathbf{1}_{3}$
$^{ERR}\mathbf{1}_{5}$	0.791576	$^{ERCR}\mathbf{1}_{4}$	$^{L}\mathbf{1}_{6}$	$\mathbf{1}_{6}$	$^{ERLS}\mathbf{1}$	$^{E}\mathbf{1}_{3}$
$^{ERLSCC}\mathbf{1}$	0.75	$^{ERRSCCC}\mathbf{1}$	$^{ERRSCCC}\mathbf{1}$	$\mathbf{1}$	$\mathbf{1}$	$^{E}\mathbf{1}_{3}$
$\mathbf{1}_{4}$	0.686027	$\mathbf{1}_{3}$	$^{L}\mathbf{1}_{5}$	$\mathbf{1}_{5}$	$\mathbf{1}$	$^{E}\mathbf{1}_{3}$
$^{ERCR}\mathbf{1}_{4}$	0.642645	$^{ELRCC}\mathbf{1}_{3}$	$^{L}\mathbf{1}_{5}$	$\mathbf{1}_{5}$	$^{ERS}\mathbf{1}$	$^{E}\mathbf{1}_{3}$
$^{ER}\mathbf{1}_{4}$	0.629472	$^{ERC}\mathbf{1}_{3}$	$^{ERL}\mathbf{1}_{5}$	$^{ERR}\mathbf{1}_{5}$	$^{ERS}\mathbf{1}$	$^{E}\mathbf{1}_{3}$
$^{ERRSCCC}\mathbf{1}$	0.5		$^{ERRSCCC}\mathbf{1}$	$\mathbf{1}$	$\mathbf{1}$	$^{E}\mathbf{1}_{3}$
$^{L}\mathbf{1}_{8}$	0.497395		$^{ERRSCCC}\mathbf{1}$	$\mathbf{1}_{9}$	$\mathbf{1}$	$^{E}\mathbf{1}_{3}$
$^{L}\mathbf{1}_{7}$	0.494337		$^{L}\mathbf{1}_{8}$	$\mathbf{1}_{8}$	$\mathbf{1}$	$^{E}\mathbf{1}_{3}$
$^{L}\mathbf{1}_{6}$	0.489495		$^{L}\mathbf{1}_{7}$	$\mathbf{1}_{7}$	$\mathbf{1}$	$^{E}\mathbf{1}_{3}$
$^{L}\mathbf{1}_{5}$	0.481096		$^{L}\mathbf{1}_{6}$	$\mathbf{1}_{6}$	$\mathbf{1}$	$^{E}\mathbf{1}_{3}$
$^{ERL}\mathbf{1}_{5}$	0.470251		$^{L}\mathbf{1}_{6}$	$\mathbf{1}_{6}$	$^{ERLS}\mathbf{1}$	$^{E}\mathbf{1}_{3}$
$^{L}\mathbf{1}_{4}$	0.466199		$^{L}\mathbf{1}_{5}$	$\mathbf{1}_{5}$	$\mathbf{1}$	$^{E}\mathbf{1}_{3}$
$^{ERCL}\mathbf{1}_{4}$	0.444509		$^{L}\mathbf{1}_{5}$	$\mathbf{1}_{5}$	$^{ERS}\mathbf{1}$	$^{E}\mathbf{1}_{3}$
$\mathbf{1}_{3}$	0.439653		$^{L}\mathbf{1}_{4}$	$\mathbf{1}_{4}$	$\mathbf{1}$	$^{E}\mathbf{1}_{3}$
$^{EL}\mathbf{1}_{4}$	0.436981		$^{ERL}\mathbf{1}_{5}$	$^{ERR}\mathbf{1}_{5}$	$^{ERS}\mathbf{1}$	$^{E}\mathbf{1}_{3}$
$^{ELRCC}\mathbf{1}_{3}$	0.396271		$^{L}\mathbf{1}_{4}$	$\mathbf{1}_{4}$	$^{ERLSCC}\mathbf{1}$	$^{E}\mathbf{1}_{3}$
$^{ERC}\mathbf{1}_{3}$	0.38498		$^{ERCL}\mathbf{1}_{4}$	$^{ERCR}\mathbf{1}_{4}$	$^{ERLSCC}\mathbf{1}$	$^{E}\mathbf{1}_{3}$
$^{E}\mathbf{1}_{3}$	0.381387		$^{EL}\mathbf{1}_{4}$	$^{ER}\mathbf{1}_{4}$	$^{ERLSCC}\mathbf{1}$	$^{E}\mathbf{1}_{3}$

Table 5.1: These are the 26 conditional unities produced with the limit parameters $l_{\max} = 9$, $p_{\max} = .$, $s_{\max} = .$ and $c_{\max} = 19$, which achieve a bit rate of 3.527 bits per vertex.

More interesting is that the limit parameters for the loop lengths grow linearly, whereas the number of used constraints nearly instantly hits the maximum of 19. This result suggest that the highest potential for reducing the bit rate further lays in the number of considered constraints. Table 5.1 and table 5.2 show two sets of conditional unities and their transitions with ten and thirty variables achieving bit rates of 3.525 and 3.555 bits per vertex. The first

unity	value	unity	value	unity	value
$\mathbf{1}$	1	${}^{ERCS}\mathbf{1}_6$	0.730321	${}^{ERRSCCC}\mathbf{1}_4$	0.46495
$\mathbf{1}_8$	0.972951	${}^{ERRCCS}\mathbf{1}_5$	0.713541	${}^{ERL}\mathbf{1}_5^3$	0.460852
$\mathbf{1}_8^4$	0.969924	$\mathbf{1}_4$	0.683458	${}^{L}\mathbf{1}_4^4$	0.45092
$\mathbf{1}_8^3$	0.965448	$\mathbf{1}_4^4$	0.657366	${}^{ERCL}\mathbf{1}_4$	0.443238
$\mathbf{1}_7$	0.945898	${}^{ELRC}\mathbf{1}_4$	0.640036	${}^{L}\mathbf{1}_4^3$	0.438188
$\mathbf{1}_7^4$	0.939842	$\mathbf{1}_4^3$	0.634757	$\mathbf{1}_3$	0.437016
${}^{ERLS}\mathbf{1}$	0.9375	${}^{ELRC}\mathbf{1}_4^4$	0.627687	${}^{ERCL}\mathbf{1}_4^4$	0.436081
$\mathbf{1}_7^3$	0.930883	${}^{ER}\mathbf{1}_4$	0.626838	${}^{EL}\mathbf{1}_4$	0.435696
${}^{ELLS}\mathbf{1}_8$	0.930233	${}^{ER}\mathbf{1}_4^4$	0.618666	${}^{EL}\mathbf{1}_4^4$	0.430926
${}^{ELLS}\mathbf{1}_7$	0.918584	${}^{ERCR}\mathbf{1}_4^3$	0.61578	${}^{ERCL}\mathbf{1}_4^3$	0.428699
$\mathbf{1}_6$	0.901187	${}^{ER}\mathbf{1}_4^3$	0.610012	${}^{EL}\mathbf{1}_4^3$	0.425403
$\mathbf{1}_6^4$	0.890117	${}^{L}\mathbf{1}^3$	0.5	$\mathbf{1}_3^4$	0.412887
${}^{ELS}\mathbf{1}$	0.875	${}^{L}\mathbf{1}_7$	0.495302	${}^{ELRCC}\mathbf{1}_3$	0.393593
$\mathbf{1}_6^3$	0.873755	${}^{L}\mathbf{1}_7^4$	0.494777	$\mathbf{1}_3^3$	0.393132
${}^{ERLSC}\mathbf{1}_7$	0.860464	${}^{L}\mathbf{1}_7^3$	0.494	${}^{ERCRC}\mathbf{1}_3^4$	0.383208
${}^{ERLSC}\mathbf{1}_6$	0.846559	${}^{L}\mathbf{1}_6$	0.489788	${}^{ERC}\mathbf{1}_3$	0.38228
$\mathbf{1}_5$	0.822796	${}^{L}\mathbf{1}_6^4$	0.488645	${}^{E}\mathbf{1}_3$	0.378678
$\mathbf{1}_5^4$	0.802935	${}^{L}\mathbf{1}_6^3$	0.482258	${}^{ERC}\mathbf{1}_3^4$	0.375476
${}^{ELR}\mathbf{1}_5$	0.790229	${}^{ELLSCCC}\mathbf{1}_5$	0.481064	${}^{ERRCC}\mathbf{1}_3^3$	0.374155
$\mathbf{1}_5^3$	0.78298	${}^{L}\mathbf{1}_5^4$	0.474247	${}^{E}\mathbf{1}_3^4$	0.373013
${}^{ERR}\mathbf{1}_5^4$	0.780676	${}^{ELL}\mathbf{1}_5$	0.470209	${}^{ERC}\mathbf{1}_3^3$	0.369211
${}^{ERR}\mathbf{1}_5^3$	0.768747	${}^{ELL}\mathbf{1}_5^4$	0.466827	${}^{E}\mathbf{1}_3^3$	0.367636
${}^{ERCLSC}\mathbf{1}$	0.75	${}^{L}\mathbf{1}_5^3$	0.465597		

Table 5.2: These are the 68 conditional unities produced with the limit parameters $l_{\max} = 8$, $p_{\max} = 4$, $s_{\max} = .$ and $c_{\max} = 19$, which achieve a bit rate of 3.5254 bits per vertex.

set of limit parameters that achieves a bit rate less than 3.525 bits per vertex is $l_{\max} = 10$, $p_{\max} = 6$, $s_{\max} = 3$ and $c_{\max} = 19$, which result in 236 conditional unities. We can refine our coding theorem to:

Theorem 5.7.1 *The connectivity of a planar triangulation with v vertices and 3 external edges can be encoded with less than 3.525 bits per vertex in linear time in v. The original connectivity can be decoded also in linear time in v.*

5.8 Conclusion

In this article we showed how to encode a planar triangulation with no more than 3.525 bits. This is currently the best known result. First we analyzed the constraints on the Edge Breaker code strings. Then we introduced a new coding technique – the conditional unities in combination with an arithmetic coder based on a state machine – which allows to exploit the constraints. We developed fast numerical methods to solve the resulting modified eigenvalue problem in order to build these state machine based arithmetic coders for given parameters that limit the extend in which the constraints are exploited. This allows us to trade off between the number of states in the arithmetic coder and the achieved bit rate. To exploit this flexibility in full depth we designed a search function that finds the parameters, which achieve the smallest bit rates, for a given maximum number of states. A plot of these parameters showed that there is so far unexploited potential in the constraint that ensures that no more than two triangles are incident to one edge. In future work we will design an automatic method to fully exploit this constraint.

References

[AD01a] P. Alliez and M. Desbrun. Progressive compression for lossless transmission of triangle meshes. In Eugene Fiume, editor, *SIGGRAPH 2001, Computer Graphics Proceedings*, Annual Conference Series, pages 195–202. ACM Press / ACM SIGGRAPH, 2001.

[AD01b] P. Alliez and M. Desbrun. Valence-Driven connectivity encoding for 3D meshes. In *Proceedings of the Eurographics '01 Conference*, pages 480–489, 2001.

[BPZ99a] C. Bajaj, V. Pascucci, and G. Zhuang. Progressive compression and transmission of arbitrary triangular meshes. In B.Hamann D.Ebert, M.Gross, editor, *Proceedings of the Visualization '99 Conference*, pages 307–316, San Francisco, CA, October 1999. IEEE Computer Society Press.

[BPZ99b] C. Bajaj, V. Pascucci, and G. Zhuang. Single resolution compression of arbitrary triangular meshes with properties. In *DCC: Data Compression Conference*. IEEE Computer Society TCC, 1999.

[CGHK98] R. C. Chuang, A. Garg, X. He, and M. Kao. Compact encodings of planar graphs via canonical orderings and multiple parentheses. In *Proceedings of the 25th International Colloquium on Automata, Languages and Programming*, pages 118–129, 1998.

[Cho97] M. Chow. Optimized geometry compression for real-time rendering. In Roni Yagel and Hans Hagen, editors, *IEEE Visualization 97*, pages 346–354. IEEE, November 1997.

[COLR99] D. Cohen-Or, D. Levin, and O. Remez. Progressive compression of arbitrary triangular meshes. In David Ebert, Markus Gross, and Bernd Hamann, editors, *Proceedings of the 1999 IEEE Conference on Visualization (VIS-99)*, pages 67–72, N.Y., October 25–29 1999. ACM Press.

[Dee95] M. Deering. Geometry compression. In Robert Cook, editor, *SIGGRAPH 95 Conference Proceedings*, Annual Conference Series, pages 13–20. ACM SIGGRAPH, Addison Wesley, August 1995. held in Los Angeles, California, 06-11 August 1995.

[DS97] M. Denny and C. Sohler. Encoding a triangulation as a permutation of its point set. In *Proceedings of the 9th Canadian Conference on Computational Geometry*, pages 39–43, August 1997. held in Ontario, August 11-14.

[GS98] S. Gumhold and W. Straßer. Real time compression of triangle mesh connectivity. In Michael Cohen, editor, *SIGGRAPH 98 Conference Proceedings*, Annual Conference Series, pages 133–140. ACM SIGGRAPH, Addison Wesley, July 1998.

[Gum99] S. Gumhold. Improved cut-border machine for triangle mesh compression. In *Erlangen Workshop '99 on Vision, Modeling and Visualization*, Erlangen, Germany, November 1999. IEEE Signal Processing Society.

[Gum00] S. Gumhold. New bounds on the encoding of planar triangulations. Technical Report WSI–2000–1, Wilhelm-Schickard-Institut für Informatik, University of Tübingen, Germany, January 2000.

[Hop96] H. Hoppe. Progressive meshes. In Holly Rushmeier, editor, *SIGGRAPH 96 Conference Proceedings*, Annual Conference Series, pages 99–108. ACM SIGGRAPH, Addison Wesley, August 1996. held in New Orleans, Louisiana, 04-09 August 1996.

[IGG01] M. Isenburg, S. Gumhold, and C. Gotsman. Connectivity shapes. In *Proceedings of the IEEE Visualization Conference*, page to appear. IEEE Computer Society, October 2001.

[IR82] A. Itai and M. Rodeh. Representation of graphs. *Acta Informatica*, 17(2):215–219, June 1982.

[IS99a] M. Isenburg and J. Snoeyink. Mesh collapse compression. In *Proceedings of the Conference on Computational Geometry (SCG '99)*, pages 419–420, New York, N.Y., June 13–16 1999. ACM Press.

[IS99b] M. Isenburg and J. Snoeyink. Spirale reversi: Reverse decoding of the edgebreaker encoding. Technical Report TR-99-08, Department of Computer Science, University of British Columbia, October 4 1999. Mon, 04 Oct 1999 17:52:00 GMT.

[Ise00] M. Isenburg. Triangle strip compression. In *Proceedings Graphics Interface 2000*, pages 197–204. Morgan Kaufmann Publishers, May15–17 2000.

[KR99] D. King and J. Rossignac. Guaranteed 3.67V bit encoding of planar triangle graphs. *Proceedings of 11th Canadian Conference on Computational Geometry*, pages 146–149, 1999.

[LK98a] J. Li and C.-C. Kuo. A dual graph approach to 3d triangle mesh compression. In *IEEE International Conference on Image Processing*, Chicago, 1998.

[LK98b] J. Li and C.-C. Kuo. Progressive coding of 3d graphics models. In *Proceedings of the IEEE, Special Issue on Multimedia Signal Processing*, volume 86(6), pages 1052–1063, June 1998.

[MC98] T. Mitra and T. Chiueh. A breadth-first approach to efficient mesh traversal. In Stephen N. Spencer, editor, *Proceedings of the Eurographics / Siggraph Workshop on Graphics Hardware (EUROGRAPHICS-98)*, pages 31–38, New York, August 31–September 1 1998. ACM Press.

[PR00] R. Pajarola and J. Rossignac. Compressed progressive meshes. In Hans Hagen, editor, *IEEE Transactions on Visualization and Computer Graphics*, volume 6 (1), pages 79–93. IEEE Computer Society, 2000.

[Ros99] J. Rossignac. Edgebreaker: Connectivity compression for triangle meshes. *IEEE Transactions on Visualization and Computer Graphics*, 5(1), 1999.

[TG98] C. Touma and C. Gotsman. Triangle mesh compression. In Wayne Davis, Kellogg Booth, and Alain Fourier, editors, *Proceedings of the 24th Conference on Graphics Interface (GI-98)*, pages 26–34, San Francisco, June18–20 1998. Morgan Kaufmann Publishers.

[TGHL98] G. Taubin, A. Gueziec, W. Horn, and F. Lazarus. Progressive forest split compression. In Michael Cohen, editor, *SIGGRAPH 98 Conference Proceedings*, Annual Conference Series, pages 123–132. ACM SIGGRAPH, Addison Wesley, July 1998. ISBN 0-89791-999-8.

[TR98] G. Taubin and J. Rossignac. Geometric compression through topological surgery. *ACM Transactions on Graphics*, 17(2):84–115, April 1998.

[Tut62] W. Tutte. A census of planar triangulations. *Canadian Journal of Mathemetics*, 14:21–38, 1962.

[WNC87] I. Witten, R. Neal, and J. Cleary. Arithmetic coding for data compression. *Communications of the ACM*, 30(6):520–540, June 1987.

■

Chapter 6

Real-Time Fluid Animation by Parallel and Stable Solution Techniques

Reinhard Klein[*], *Thorsten May*[†], *Sascha Schneider*[‡], *Andreas Weber*[§]

We describe new techniques for the parallelization of an unconditional stable solution scheme of the Navier-Stokes equation that has recently been introduced for animation purposes. The parallel improvement over existing schemes eleminates the sequential bottleneck. Additionally our solution method makes use of time steps which are bigger than the frame-rate for interpolation between them. We achieve real-time animation rates for several examples on current multi-processor workstations and PC's.

[*] Institut für Informatik II, Universität Bonn, Germany. E-mail rk@cs.uni-bonn.de

[†] Fraunhofer-Institut für, Graphische Datenverarbeitung, Darmstadt, Germany. E-mail tmay@igd.fhg.de

[‡] Fraunhofer-Institut für, Graphische Datenverarbeitung, Darmstadt, Germany. E-mail sschneid@igd.fhg.de

[§] Institut für Informatik II, Universität Bonn, Germany. E-mail weber@cs.uni-bonn.de

6.1 Introduction

Some of the most fascinating observations one can make in nature can be explained as effects of turbulent fluids or gases. So simulating turbulent fluids is not only a major topic in engineering but *computational fluid dynamics* (CFD) has also become a topic in computer graphics and animation [FSJ01, SF93, SF95, FM97, CFW99, WW99, Sta99, DKY+00, YOH00]. However, in an animation context the topic of real time simulations is much more important than in a general engineering setting, whereas the accuracy of the simulations can be much less in general: very often the result of the simulation has to be just "visually right". So the introduction of an unconditionally stable solution scheme of the Navier-Stokes equations into the area of computer graphics [Sta99] was seen to be an important step by the graphics community. The "almost sufficiency" of staying in the stability region of a solution scheme for animation purposes has also been used in entirely different contexts [BW98, DMB00], which nevertheless are quite similar in various aspects: The implicit schemes used in that work on cloth animation as well as the implicit solution scheme proposed by Stam for fluid animation allow large step-sizes on the prize of an over-damping of the solution. We refer to these papers for a discussion why such a over-damping is not much of a problem in an animation context (in contrast to an engineering context). Moreover, for the case of the Euler equations, i.e. for systems without viscosity, Stam has shown in recent work how this over-damping can be corrected [FSJ01].

For same relatively small but non-trivial 3D-examples the animation system of Stam, which uses a sequential implementation, was a major step towards the goal of real-time animation. So it is natural to ask to what extend the realm of real-time animations can be extended to larger examples by parallelization on multi-processor-systems. In previous work [BKM+00] we showed that this stable solution scheme is in principle well suited for parallelization and we described a parallel implementation that is portable on workstations and PC's running under the UNIX or the WIN32 operating systems. However, one sequential bottleneck remained (in a parallel sub-task) which prohibited the scaling of our parallelization on more than about 3 or 4 processors. Moreover, we used the unconditional stable scheme only to the extend of using time-steps of the size of the frame rate. We will show in this paper that by using time steps higher than the frame-rate and interpolation our parallel environment is able to achieve real-time fluid animations on examples that are by about a factor 2 to 5 larger than previous ones.

We will briefly recapitulate the previous work of Stam [Sta99] and of ourselves [BKM+00] as the basis of the description of our current work.

6.2 The Sequential Algorithm

There is a vast literature on computational fluid dynamics. For general information we refer to recent textbooks [YMO97, Whi94, FP99, Abb89, CM90]. For the specifics of the sequential algorithm that is the basis of our work we refer to the paper of Stam [Sta99].

6.2.1 Simulation program

The simulation program solves the Navier-Stokes equations

$$\nabla \cdot \mathbf{u} = 0 \tag{1}$$

$$\frac{\partial u}{dt} = -(\mathbf{u} \cdot \nabla)\mathbf{u} - \frac{1}{\rho}\nabla p + \nu \nabla^2 \mathbf{u} + \mathbf{f} \tag{2}$$

for an incompressible fluid, where u is the velocity vector field, v is the viscosity, d the density, p the scalar pressure field and f the external forces acting on the fluid, e. g. buoyancy.

In an incompressible fluid the role of the pressure is to allow continuity to be satisfied. It is therefore possible to write eqn. 1 and eqn. 2 as just one equation using an operator P, which orthogonaly projects a vector field on to the set of divergence free fields. Using this operator the equations 1 and 2 can be written as

$$\frac{\partial u}{dt} = \mathbf{P}(-(\mathbf{u} \cdot \nabla)\mathbf{u} - \nu \nabla^2 \mathbf{u} + \mathbf{f}) \quad . \tag{3}$$

It is this equation that is solved by the simulation program.

A fractional step method consisting of the following parts is used to solve the equation. The method we use for the solution is the one described by Stam [Sta99], to which we refer for more details and a theoretical base for this method. We just sketch the main steps as a reference for the parallelization.

$$\mathbf{w}_0(\mathbf{x}) \overset{\text{addforce}}{\rightarrow} \mathbf{w}_1(\mathbf{x}) \overset{\text{advect}}{\rightarrow} \mathbf{w}_2(\mathbf{x}) \overset{\text{diffuse}}{\rightarrow} \mathbf{w}_3(\mathbf{x}) \overset{\text{project}}{\rightarrow} \mathbf{w}_4(\mathbf{x}) \tag{4}$$

Add force

The effect of the external forces is solved using an ordinary explicit Euler scheme.

$$\mathbf{w}_1(\mathbf{x}) = \mathbf{w}_0(\mathbf{x}) + \Delta t \mathbf{f}(\mathbf{x}, t) \tag{5}$$

Transport

Using a technique based on the method of characteristics the nonlinear transport (advection) part of the equation is solved. This method has several advantages. Two of them are ease of implementation and—as will be shown later—ease of adaption to parallel computing. A point $\mathbf{x}$ is back-traced along a streamline in the old vector field to time $t = -\Delta t$. The velocity at $\mathbf{x}$ in the new vector field is then set to the value at the back-traced point. Velocity vectors not directly on a grid-point are defined using trilinear interpolation from the adjacent grid-points. Using a function $\mathbf{p}(\mathbf{x}, t)$ defined as the streamline passing through point $\mathbf{x}$ at $t = 0$ the step can be written as

$$\mathbf{w}_2(\mathbf{x}) = \mathbf{w}_1(\mathbf{p}(\mathbf{x}, -\Delta t)) \quad . \tag{6}$$

A particle tracer can easily be implemented using any standard ODE solver. The choice for the program described in this paper is a Runge-Kutta solver of the fourth-order.

Diffuse

The third step deals with the effect of viscosity. This step calculates the solution of a standard diffusion equation for each Cartesian component of the vector field. These equations are solved using an implicit Euler scheme, so the resulting system is

$$(\mathbf{I} - \nu \Delta t \nabla^2)\mathbf{w}_3(\mathbf{x}) = \mathbf{w}_2(\mathbf{x}) \quad , \tag{7}$$

where the ∇^2 operator is approximated using finite differences. Several efficient methods for solving such equations exist. Specifically in the implementation described in this paper the `pois3d`-solver from the FISHPAK library is used. This library is available from `http://www.netlib.org`.

Although this routine has a complexity of about $O(n \log n)$ and this equation can be solved theoretically in $O(n)$ using a multi-grid method [Hac85], the practical performance of `pois3d` is much better on the currently used

grids: For grids consisting of about 4000 cells the `pois3d` routine is about 10 times faster than the best of the multi-grid methods that we have tested, for grids consisting of about 125000 cells it is still about a factor of 9 faster. The given factors are the ones for "good" grid sizes; for "bad" grid sizes the factor is about 5: The computation time of `poisd3` strongly depends on the largest prime factor of the grid dimension (plus one), e. g. the computation time on a cube of $(32-1)^3$-cells is much smaller than the one on a cube of $(31-1)^3$ cells!

If the viscosity is 0, an approximation that might be made for air, the diffusion step can be omitted.

Project

The vector field produced by the above steps are not ensured to satisfy the continuity equation. The final step is therefore a projection step which makes the field divergence free. The step can be written as

$$\nabla^2 q = \nabla \cdot \mathbf{w}_3 \tag{8}$$

$$\mathbf{w}_4 = \mathbf{w}_3 - \nabla q \quad . \tag{9}$$

This Poisson equation is solved using the same subroutine that is used in the diffuse step.

Computing scalar quantities

The evolution of a scalar quantity, e. g. temperature or smoke density in the fluid is computed using a method very similar to the one for the vector field described above. The equation that describes the behaviour of the scalar field is

$$\frac{ds}{dt} = -\mathbf{u} \cdot \nabla a + \kappa \nabla^2 s - \alpha s + S \quad , \tag{10}$$

where κ is the diffusion constant, α the dissipation rate and S a source term. All terms are solved using the same steps that are used for the vector field except of *project*, which is not needed. The dissipation term not present in the Navier-Stokes equations is solved using the following equation

$$(1 + \Delta\alpha) s(\mathbf{x}, t + \Delta t) = s(\mathbf{x}, t) \quad .$$

6.2.2 Volume renderer

The principal components of the used volume renderer have not changed substantially. We thus refer to our previous work [BKM+00] for the details of the used splatting method and the method of using textures for optical refinement of a flow. As this method has also been used by others [Sta99, MCW92], we refer to this sources for a description of the underlying ideas of this very important part of our fluid animation system.

6.3 Parallel Architecture

The sequential components described above can be split into various parallel components. In the following we will give a "top down" description of the parallelized components: First, we give the top-level parallel architecture between the renderer and solver components giving then the details on the parallelization of the solver components itself, starting again from the higher levels.

Since for real time animations a parallelization over a network has a too high latency, we focus on shared-memory architectures, which are nowadays available in the form of relatively low cost multi-processor PCs or workstations. The general programming paradigm that is available for these platforms is the one of multi-threaded programming. Although quite similar from a programming point of view the APIs that are offered by the WIN32 operating systems and the various UNIX systems differ. However, it is possible to provide abstraction classes with little performance overhead that allow a platform independent access to the thread systems. The *Adaptive Communication Environment* (ACE) [Sch93] and the OMNI thread package [Tri97] are two such packages. Currently, we use the OMNI thread package as an platform independent abstraction for the threads of the WIN32 operating systems and for POSIX threads, but switching to another one like ACE would be possible with a moderate programming effort.

The windowing classes that we use are completely written in Qt [Tro99] in order to allow a platform independent implementation. The graphical context of the window is filled with rendered OpenGL graphics [WNDS99]. These two together allow our program to run on all Operating Systems which support Qt and OpenGL. Recently platform independent thread abstraction classes have been added to Qt. Thus we might substitute the OMNI thread package by these thread classes of Qt in the future.

Thus our system in particular runs under the WIN32 operating system and most of the UNIX systems.

6.3.1 Communication and synchronization between renderer and solver components

In our program the renderer (OpenGL/QT window) interacts with the CFD solverthread using a cyclic n-fold buffer. This is an extension of our previous architecture, where we used a double-buffer. When a simple linear interpolation is used to refine the simulation steps in the animation performed by the renderer, we have $n = 3$, i. e. a cyclic triple-buffer. Each buffer contains the velocity field and the entities used for rendering, e. g. the current texture coordinates. For interpolation, $n - 1$ parts of the buffer are used by the renderer. After the solver thread has completed one calculation-step and has written its results in the currently unused part of the buffer, the marker in the cyclic buffer, which indicates the last entry to be used by the renderer, is advanced in one position. Thus only a "switch of pointers" has to be locked. A schematic view is given in Fig. 6.1.

6.3.2 Parallelization of simulation program

We compute the vector field of velocities and the scalar quantities at discrete time-steps. The simulation of a the scalar quantities at time t_n requires the knowledge of the vector field at this time step. However, in a simulation loop the simulation of the vector field for time step t_{n+1} (using the values of the vector field at time step t_n) can be done in parallel to the simulation of the time step of the scalar field at time step t_n (using the value of the scalar field t_{n-1} and the vector field at time step t_n).

Any simulation step of the vector field requires the computation of the sequence *add force* $\rightarrow$ *transport* $\rightarrow$ *diffuse* $\rightarrow$ *project* (and similarly for the scalar quantities). These steps have to be taken sequentially.

Add force and transport

The *add force* step takes little time but the addition of any force to a cell can be done in parallel. In the sequential program the *transport* step requires a major part of the computation time of the vector field simulation (about 40 %

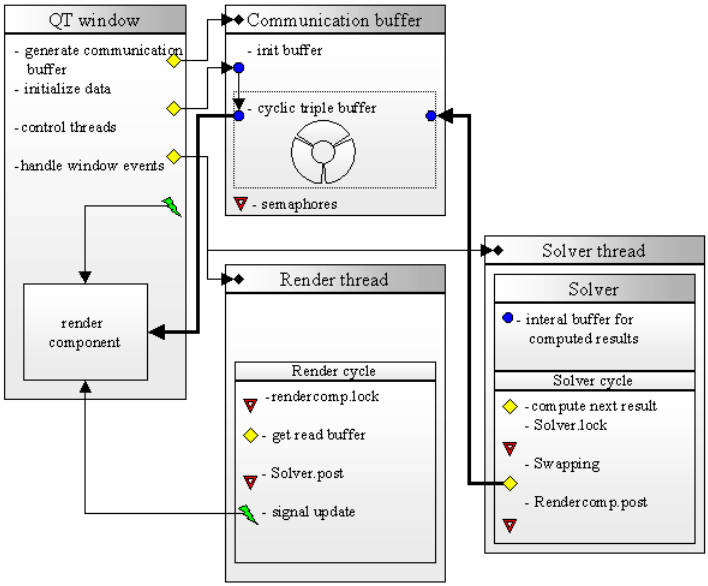


Figure 6.1: Communication and synchronization between renderer and solver components

for the examples given below). This step can be well parallelized without synchronization requirements: the computation for any grid cell is independent of each other. Using n threads, where n is about four times the number of available processors to reach saturation, we divide the cells in n parts and do the transport computation in the n threads, each responsible for one part. As the computation of transport step roughly takes the same time for any grid cell these threads can be expected to have the same run-time.

Diffusion

The *diffusion* step requires a solution of the corresponding equations in 3 dimensions, which are independent of each other. So we can use 3 threads to do these computations in parallel, which require no synchronization and which have nearly the same run-time. Parallelizing the diffusion computation for each dimension themselves is possible in principle with the technique described below for the project step.

Projection

The *project* step requires the solution of one equation using `pois3d` and was unparallelized previously. It will become a sequential bottleneck for a higher number of processors; for a smaller number of processors (2 or 3) there is still good parallelism due to the parallel execution of vector field computation, the scalar field computations, interpolation, and renderer. The new parallelization scheme that we have recently developed, which allows the splitting into 8 parallel sub-tasks, is described in Sec. 6.3.3.

Parallelization schemes for vector field and scalar field solvers

The corresponding parallelization scheme for the vector field solver is given in Fig. 6.2, the one for the scalar field solver and the other quantities used for visualization (particles, texture coordinates) are given in Fig. 6.3.

6.3.3 Parallelization of the projection step

The full details of the derivation of the parallelizable scheme are given in the thesis of one of the authors [May00], which we refer to for further details. We will only briefly sketch the main idea below.

We have to consider the discretization by finite differences of equations 8 and 9. For inner grid-points the differences to be considered are not the ones of a grid-point with its neighbors, but with the neighbors of the neighbors, as we have roughly speaking second-order differences. As there is no dependency on the direct neighbors, we can partition the grid in 8 independent sub-grids, as is shown in Fig. 6.4.

The projection operator of each of these 8 sub-grids can then be solved in parallel by an invocation of `pois3d`. For the grid points at the boundary we have to add a correction term.

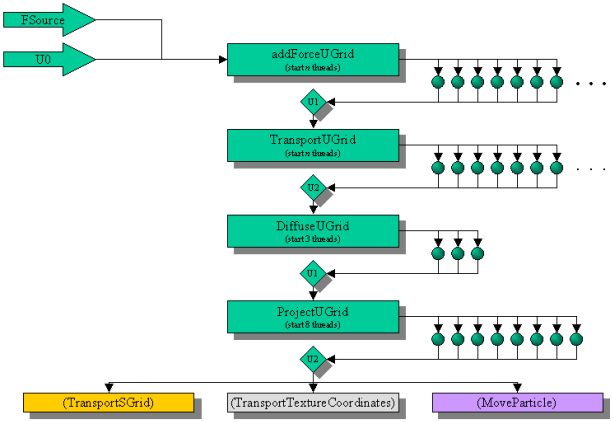


Figure 6.2: Parallelization of vector field solver

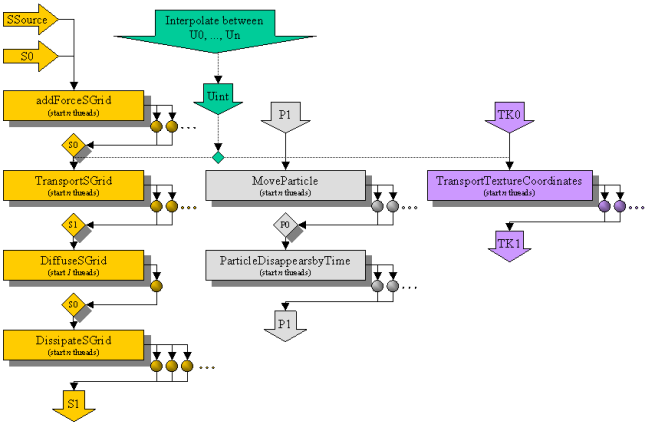
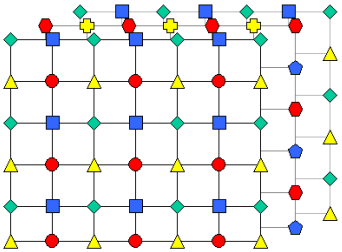


Figure 6.3: Parallelization of scalar field solver



Dependency of grid points for finite difference discretization of projection operator. There are 8 independent sub-grids (visualized by different symbols). Only 3 levels of the grid are shown.

Figure 6.4: Splitting of grid in 8 sub-grids for projection operator

For smaller grids a potential problem is that the sub-grids might have a less favorable dimension for the `pois3d` algorithm, cf. Sec. 6.2.1. However, as the `pois3d` algorithm only has to be applied to the sub-grids only (and no longer to the entire grid), it is possible to adjust the grid-sizes according to the rules sketched in Sec. 6.2.1: Only the sub-grids should have a favorable dimension, it is no longer necessary that the dimension of the entire grid itself has one. Specifically, if a $n_1 \times n_2 \times n_3$ grid is used, the integers n_i should have the properties that $\lfloor n_i/2 \rfloor + 1$ and $\lceil n_i/2 \rceil + 1$ ($i = 1, 2, 3$) have only small prime divisors.

6.3.4 Using large step sizes and interpolation

The unconditionally stable solution scheme described by Stam was used by him and in our previous work to obtain step-sizes in the simulation that are as big as the frame time. This already gave a major gain in performance in comparison to explicit solution schemes, as these schemes typically can only take much smaller step-sizes without loosing stability: using an explicit scheme many simulation steps have to be taken to simulate the frame time. For engineering purposes, explicit schemes are usually preferred as the accuracy of the simulation is of importance, which is already limiting the possible step sizes.

Usually step-sizes are less than 0.001 sec with explicit solution schemes.

From the point of view of simulations, which do not require much accuracy as the ones used for flow animations, the time steps taken by an unconditionally stable scheme can be much larger than the frame time. However, for the purposes of real-time animations "smooth" transitions between two consecutive frames has to be ensured. But these smooth transitions can be obtained in principle by interpolation, which are computationally much cheaper than full simulation steps.

In our parallel environment, in which the renderer and solver are running in parallel, it was relatively easy to add an interpolator and to decouple the simulation from the animation accordingly. The simulation steps are multiples of the frame time and the intermediate frames are obtained by interpolation.

We found on several examples that we could interpolate about 5 frames without much loss of quality in the animation, but this number is certainly dependent on the application. Given the rather short computation time needed for interpolation between two vector fields in comparison to the time needed for a simulation step—in our implementation this ratio is less than 5 %—examples that are almost 5 times bigger can be animated in real-time than could be done without using interpolation.

6.4 Results

6.4.1 Performance measurements

We measured the speed-up of our parallelization by determining the wall clock times for the sequential and parallel versions as the averages of 10 runs. When determining the speed-ups of the different steps in the vector field computations we disabled the render components in order to avoid disturbances.

On a 4 processor SGI Onyx 2 workstation we obtained speed-ups of 2.8–3.1 on grids of sizes 31^3 and 64^3 for our new parallelization scheme of the project step. The parallelization of the transport and diffuse step has not changed in a essential way in comparison to our previous work [BKM+00]: On grids of size 31^3 cells the speed-up for the transport step has been 3.1 and the one for the diffuse step has been 1.7. On larger grids (with 49^3 cells or more) the speed-up of the transport step reached the theoretically possible value of 4 up to 5 %.

As the speed-ups scale to faster processors, animations on grids of these

sizes are possible in real-time using our programming techniques on new multi-processor systems that consist of GHz-processors. As such a system has not been available to us right now we have provided somewhat smaller examples as accompanying material.

6.4.2 Animations

Some snapshots from real-time animations of our system are given in the appendix, cf. Fig. 6.5. Movies showing the animations are provided as accompanying material and can also by found in the Web: `http://www.igd.fhg.de/igd-a3/projects/physically-based-simulation-and-animation/cfd/`

The animations were generated on a dual processor 500 MHz Pentium III PC with GeForce DDR graphics card. The example of a smoking chimney was computed on a grid of size $31 \times 15 \times 15$ without interpolation. The second animation, steam from three nozzles, was computed on a grid of size 24^3 interpolating 4 frames between two simulation steps.

6.5 Conclusion

We have shown that the unconditionally stable solution scheme of the Navier-Stokes equation that has recently been used for animation purposes can be parallelized with good speed-ups on current multi-processor workstations. We could close a previously existing sequential bottleneck, the project step, by an analysis of the underlying scheme, allowing to split the task into 8 parallel subtasks. We also take the idea of using an unconditionally stable scheme allowing large step-sizes further than previously done by using step-sizes larger than the frame rate and using interpolation for smooth animation. The speed-ups that we obtain by combining these techniques close the gap that existed towards real time animation for several 3D-examples.

Our parallelization is portable on workstations and PC's running under the UNIX or the WIN32 operating systems. Because of the improving performance of relatively low cost multiprocessor PC's our system will also allow real time fluid animations on this rapidly growing hardware segment.

Acknowledgments

We are grateful to P. Borodin and F. Birra for their help in implementing parts of the system, and to M. Bryborn for his achievements at the beginning of this project.

References

[Abb89] M. B. Abbott. *Computational Fluid Dynamics : An Introduction for Engineers*. Wiley, 1989.

[ACM99] ACM SIGGRAPH. *SIGGRAPH 99 Conference Proceedings*, Annual Conference Series, Los Angeles, CA, USA, August 1999.

[ACM00] ACM SIGGRAPH. *SIGGRAPH 2000 Conference Proceedings*, Annual Conference Series, New Orleans, LA, USA, July 2000.

[BKM+00] Mattias Bryborn, Reinhard Klein, Thorsten May, Sascha Schneider, and Andreas Weber. A portable, parallel, real-time animation-system for turbulent fluids. In M. Guizani and X. Shen, editors, *Parallel and Distributed Computing and Systems (PDCS 2000)*, pages 394–400, Las Vegas, USA, November 2000. International Association of Science and Technology for Development.

[BW98] David Baraff and Andrew Witkin. Large steps in cloth simulation. In Michael Cohen, editor, *SIGGRAPH 98 Conference Proceedings*, Annual Conference Series, pages 43–54, Orlando, FL, USA, July 1998. ACM SIGGRAPH.

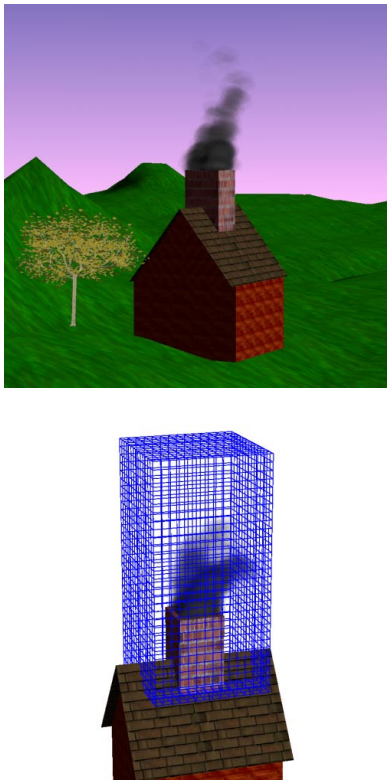
[Cam99] AT&T Laboratories Cambridge. omniORB, 1999. `http://www.uk.research.att.com/omniORB/`.

[CFW99] Jim X. Chen, Xiaodong Fu, and Edward J. Wegman. Real-time simulation of dust bahavior generated by a fast traveling vehicle. *ACM Transactions on Modeling and Computer Simulation*, 9(2):81–104, 1999.

[CM90] A. J. Chorin and J. E. Marsden. *A Mathematical Introduction to Fluid Mechanics*, volume 4 of *Texts in Applied Mathematics*. Springer-Verlag, 2nd edition, 1990.

[DKY+00] Yoshinori Dobashi, Kazufumi Kaneda, Hideo Yamashita, Tsuyoshi Okita, and Tomoyuki Nishita. A simple, efficient method for realistic animation of clouds. In *SIGGRAPH 2000 Conference Proceedings* [ACM00], pages 19–28.

[DMB00] Mathieu Desbrun, Mark Meyer, and Alan H. Barr. Interactive animation of cloth-like objects for virtual reality. In Donald H. House and David E.



Some snapshots from a simulation sequence of a smoking chimney. The first snapshot shows the smoke simulation together with a background scene. In the second snapshot the vector field of wind velocities is visualized with arrows. In the third snapshot the used grid of size $31 \times 15 \times 15$ *is shown. The forth snapshot shows the vector field at another stage of the simulation.*

Figure 6.5: Simulation sequence of smoking chimney

Breen, editors, *Cloth Modeling and Animation*, chapter 9, pages 219–239. A. K. Peters, 2000.

[FM97] Nick Foster and Dimitris Metaxas. Modeling the motion of a hot, turbulent gas. In *SIGGRAPH 97 Conference Proceedings*, Annual Conference

Series, pages 181–188, Los Angeles, CA, USA, August 1997. ACM SIGGRAPH.

[FP99] Joel H. Ferziger and Milovan Peric. *Computational Methods for Fluid Dynamics*. Springer Verlag, 2nd edition, 1999.

[FSJ01] Ronald Fedkiw, Jos Stam, and Henrik Wann Jensen. Visual simulation of smoke. In *SIGGRAPH 2001 Conference Proceedings*, Annual Conference Series, Los Angeles, CA, USA, August 2001. ACM SIGGRAPH.

[Hac85] W. Hackbusch. *Multi-grid Methods and Applications*. Springer-Verlag, 1985.

[May00] Thorsten May. Schnelle Visualisierung von Strömungseffekten. Diplomarbeit, Technische Universität Darmstadt, Darmstadt, Germany, December 2000.

[MCW92] N. Max, R. Crawfis, and D. Williams. Visualizing wind velocities by advecting cloud textures. In *Proceedings of Visualization 92*, pages 179–183, Los Alamitos, CA, USA, 1992. IEEE.

[Sch93] Douglas C. Schmidt. The ADAPTIVE Communication Environment: An object-oriented network programming toolkit for developing communication software., 1993. `http://www.cs.wustl.edu/~schmidt/ACE-papers.html`.

[SF93] J. Stam and E. Fiume. Turbulent wind fields for gaseous phenomena. In *SIGGRAPH 93 Conference Proceedings*, Annual Conference Series, pages 369–376, Anaheim, CA, USA, August 1993. ACM SIGGRAPH.

[SF95] J. Stam and E. Fiume. Depicting fire and other gaseous phenomena using diffusion processes. In *SIGGRAPH 95 Conference Proceedings*, Annual Conference Series, pages 129–136, Los Angeles, CA, USA, August 1995. ACM SIGGRAPH.

[Sta99] Jos Stam. Stable fluids. In *SIGGRAPH 99 Conference Proceedings* [ACM99], pages 121–128.

[Tri97] Tristan Richardson. The OMNI Thread abstraction. `http://www.lfpt.rwth-aachen.de/Links/GNU/gnu/omniorb/omnithread/omnithr%ead.html`, 1997. Available as part of omniORB [Cam99].

[Tro99] Trolltech. Qt. `http://www.trolltech.com/`, 1999.

[Whi94] Frank M. White. *Fluid Dynamics*. McGraw-Hill, 3rd edition, 1994.

[WNDS99] Mason Woo, Jackie Neider, Tom Davis, and Dave Shreiner. *OpenGL Programming Guide*. Addison-Wesley, third edition, 1999.

[WW99] Henrik Weimer and Joe Warren. Subdivision schemes for fluid flow. In *SIGGRAPH 99 Conference Proceedings* [ACM99].

[YMO97] Donald F. Young, Bruce R. Munson, and Theodore H. Okiishi. *A Brief Introduction to Fluid Dynamics*. John Wiley & Sons, 1997.

[YOH00] Gary D. Yngve, James F. O'Brien, and Jessica K. Hodgins. Animating explosions. In *SIGGRAPH 2000 Conference Proceedings* [ACM00], pages 29–36.

■

Chapter 7

Efficient Multiresolution Models for progressive Terrain Rendering

Reinhard Klein[*]*, Andreas Schilling*[†]

This paper deals with the problem of simplifying, transmitting and rendering large textured terrain models. The terrain is rendered from a hierarchy of quadratic tiles of geometry. Each of these tiles has a corresponding texture with a constant number of texels, e.g. 64x64, in each resolution level. Therefore, the ratio between the needed geometric accuracy within a tile and the texel size is a constant, independent of the level of detail, e.g. one texel (1/64 of the width of the tile). This observation allows us to create an efficient new multiresolution model for terrain data where not only the number of vertices is adapted to the level of detail but also the relative accuracy of the coordinates. Although we do not use a pure quadtree data structure due to its known problems with the representation of not axis aligned geometric features, our data structure is very

[*] Institut für Informatik II, Universität Bonn, Germany. E-mail rk@cs.uni-bonn.de
[†] WSI/GRIS, Universität Tübingen, Germany. E-Mail: schilling@uni-tuebingen.de

compact and avoids redundant transmission of coordinates.

7.1 Introduction and previous work

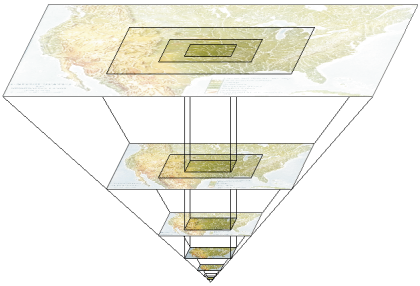


Figure 7.1: Clipmaps: Instead of holding the entire texture pyramid in texture memory the finer levels (where the images are huge) are clipped to a fixed size texels around a center point called clip center. In this way the finest loaded texture covers the smallest part of the geometric model. Coarser levels cover correspondingly larger parts.

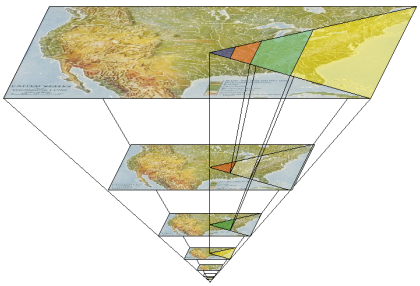


Figure 7.2: The really needed parts of the texture are shown. This is only a fraction of the textures loaded with Clip-Mapping.

Rendering of textured terrain models has become a wide used technique in the field of GIS applications. Possible textures are aerial or satellite images, data derived from such images like pollution maps or arbitrary kinds of other

maps. All these texture images share the common property that they can be very large e.g. $4,000,000 \times 4,000,000$ texels for an aerial image of a region of $1000 \times 1000 km^2$ (25cm resolution).

7.1.1 Handling huge textures

To deal with the large textures there are two main approaches. An elegant way is the use of Clipmaps [TMJ98]. Instead of holding the entire texture pyramid in texture memory the finer levels (where the images are huge) are clipped to a fixed size, e.g. 2048×2048 texels around a center point called clip center. In this way the finest loaded texture covers the smallest part of the geometric model. Coarser levels cover correspondingly larger parts. In principle this is the behaviour needed for rendering, see Figure 7.1. One problem of the Clipmap is the selection of the center-point. In flat terrain the simplest way would be to project the observer position onto the terrain and use the texture coordinates as clip center. The drawback of this approach is that in this situation the clipmap contains texture behind the observer and outside of the view frustum, compare Figure 7.1 with 7.2. It is difficult to improve the situation by choosing a clip center in some distance in front of the observer, as it would be necessary to choose different clip centers for the different resolution levels. Even if the clip center can be optimally chosen in the different levels the quadratic clip map would still not optimally fit the triangular view frustum. Choosing the clip center is even more complicated in the presence of mountains. In addition it has to mentioned that in this case occlusion culling techniques can be employed but the nature of the clipmap does not allow to take advantage of the occlusion culling as it is not possible to reduce the amount of loaded texture. A further problem of clip maps is that they are still only available on high performance graphics hardware. Another technique is the MP-Grid [Hüt98]. The idea of the MP-Grid is to split the texture into a regular grid of standard Mipmaps. This approach gives the possibility to handle textures of arbitrary size. Secondly only textures for visible grid cells must be loaded. The loading can be restricted to the necessary levels of detail which are precomputed and stored. Furthermore, this approach fits directly to the available OpenGL API. To guarantee that the texture for a single triangle belongs to only a single grid cell the borders of the texture grid are inserted into the triangulation before rendering. The problem is that the regular grid cannot be adapted to the camera. This has the consequence that the grid cell can become arbitrarily small on the screen. Besides the problem that it is imprac-

tical to load a very large number of Mipmaps in this case, the filtering of the textures when the footprint of a pixel covers several grid cells can not be performed as the lowest available level is reached when one texel corresponds to one grid cell. Furthermore, insertion of the cell boundaries into the triangulation means that simplification of the original model with triangles larger than the cell size is cancelled out. Assume for example a viewer looking down from space on the above mentioned $1000 \times 1000\text{km}^2$ textured area which is divided in 4000×4000 cells containing each a 1024^2. If the viewer is far away it would be enough to represent the whole area with a simplified model consisting of just two triangles. Using the MP-Grid 16,000,000 cells (32,000,000) triangles must be rendered. In this case the grid cells are smaller than a screen pixel. Therefore, the corresponding Mipmap level is not available. To overcome the problem of the cell boundaries in the paper a new hardware architecture that supports the MP-Grid is proposed. An important feature of this architecture is that the insertion of cell boundaries can be avoided as the hardware generates on the fly the cell numbers. Furthermore, the hardware allows to merge small grid cells into a larger one.

7.1.2 Dealing with geometry

As for textures the problem of the large size also exists for the geometric data, normally given as a Digital Elevation Model (DEM). A DEM is defined as a set of points $(x_i, y_i, h(x_i, y_i)), 1 \leq i \leq N$, and interpolation rules to derive height values in between. An interpolation rule usually incorporates information how (e.g. polynomial degree) and from where (e.g. a local mesh topology) data is interpolated. The height values $h(x_i, y_i)$ are always fixed given data and have to be stored for every DEM.

To deal with the huge amount of data a number of different elaborated multiresolution DEMs were developed [LKR+96a, KH96, Pup96, DWS+97, KCOH98, Hop98, Paj98, LP01]. All these approaches concentrate on the problem of geometry but not on texturing. Doellner et al. [DBH00] address the issue of external memory handling of large textures for terrain visualization.

There is a great variety of methods to generate multiresolution DEMs, usually classified as top-down (refinement) and bottom-up (decimation) methods, see e.g. [HG97] for a survey. The top-down method starts with a coarse approximate DEM and inserts single elements such as points, edges or triangles until the original DEM is reached. The bottom-up approach begins with the input DEM and removes elements successively.

In order to allow accurate view-dependent visualization the approximation error between the different Level of Detail (LOD) approximations and the original DEM must be known [KS96, KCOH98]. Unfortunately, many of the simplification methods need the entire original data within main memory to calculate the approximation error. Therefore, this methods are limited to small or medium size DEMs or they must run on super computers. For real DEMs new algorithmic techniques are needed to deal with this problem.

The resulting multiresolution DEMs can roughly be classified according to how much additional storage is required for the interpolation rules and the coordinates of the points (x_i, y_i). If the additional storage requirement is of order $O(N)$, the respective set of data is called explicit. If the order is lower, it is called implicit since the additional memory has no practical influence on the total size. Current hierarchical DEMs can be divided into the following classes:

- coordinates implicit, interpolation explicit: quadtrees, adaptive hierarchical triangulations and adaptive tensor product grids,
- coordinates explicit, interpolation implicit: Delaunay triangulations, relocated regular grids,
- coordinates and interpolation explicit: data-dependent triangulations, general tessellations.

Clearly, more explicit schemes allow greater flexibility than less explicit ones. But typically also the complexity of corresponding algorithms and supporting data structures grow.

In addressing the problems of texturing and efficient DEM management, this paper makes two major contributions. First, it introduces the texture tile representation. The main idea of this representation is to subdivide the texture into equal sized blocks called texture tiles. The block size is the same in all levels of the texture pyramid. In the geometric model this structure corresponds to a quadtree subdivision: Texture tiles from coarse levels correspond to large areas, those from finer levels to small areas, see Figure 7.2. Note the similarity to the Clipmap approach. The difference is that in such a way only the needed textures are loaded. In contrast to the MP-grid the size of a single cell on the screen is of constant order.

The other contribution of this paper is a new simple geometric multiresolution model adapted to the texture tiles. The geometric accuracy of the (x_i, y_i)

coordinates as well as the height-values $h(x_i, y_i)$ on each tile are limited to the needed texture resolution. Therefore, in our model in addition to the texture also the position and elevation data are hierarchically organized and incrementally transmitted. In the light of the above classification it is a hybrid scheme that combines the advantages of the implicit representation of quadtrees with the flexibility of data-dependent triangulations. During its constructions the measurements of the approximation error are kept local and therefore even huge DEMs can be processed on a standard PC.

7.2 The multiresolution model

As we suppose that only standard Mipmapping hardware is available, the geometric multiresoltuion model must contain the borders of the texture tiles. Therefore, the different levels of detail must contain the borders of the quadtree structure defined by the texture tiles.

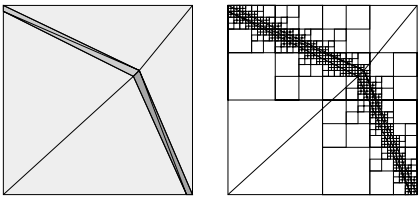


Figure 7.3: Structures not aligned to the rectangular basic structure of the quadtree are difficult to represent and in general they lead to unwanted refinement. The cliff in the example (indicated by dark slopes) could be represented by only eight triangles. With a quadtree hundreds or even thousands of triangles will be produced.

Quadtree tiles are not a new idea in the context of view dependent rendering [HB87, LKR+96b] to cite only two early ones. In all of these approaches we have the problem that geometric structures not aligned to the quadtree lead to highly refined quadtree cells. This could be avoided if inside a tile an arbitrary triangulation would be used, see Figure 7.3. Hoppe [Hop98] addresses the problem of constructing a progressive mesh of a large terrain using a bottom up scheme, by decomposing the terrain into square tiles that are merged after independent decimation, and which are then further simplified. The new repre-

sentation we describe here has the same advantage. Furthermore, for each tile in our model a guaranteed error bound is available. A new very compact representation for connecting tiles at different levels without gaps allows to store the whole multiresolution model in an efficient way. In contrast to previous approaches no constraints are put on the triangulation to achieve this goal.

View-dependent rendering of the model requires a coarse approximation in the distance and finer approximation close to the observer. In contrast to common MRMs that start with the coarsest triangulation and decide for each triangle if further refinement is needed we restrict ourselves to one decision per quadtree cell. This means for each quadtree cell we have in principle only one triangulation that is totally precomputed and stored in form of efficient triangle strips. Note that triangle strips automatically deliver a very good compression for the connectivity of the triangle meshes. Of course transitions between different levels of detail cannot be handled without gaps using these fixed triangulations. Fortunately, as the limit for the acceptable approximation error grows with the reciprocal of the distance to the viewer, it is sufficient that adjacent quadtree cells differ in only one level. Now we use a simple approach. We have to deal with the situation that at each of the four edges the LOD may change. In the case of a change in the LOD another precomputed triangulation for the quadtree cell is used that avoids the gaps. Instead of storing triangulations for all possible combination of transitions it is sufficient to consider only transitions to a higher level of detail. The opposite case is handled by changing the triangulation in the cell on the other side of the border. We store the different triangulations in an efficient new way by combining them in a single representation.

7.2.1 The simplification algorithm

An important question is which approximation error can be allowed for a certain cell. The property of the cells to have similar size on the screen after rendering leads directly to the conclusion that the allowed approximation error is a certain fraction of the size of the cell itself. If we assume for example that the maximum length of a grid cell on the screen is about 64 pixels, an approximation error on cell of the cell size divided by 64 corresponds to a screen space error of 1 pixel at most. Therefore, during the simplification algorithm the triangulation in each cell is reduced until the geometric approximation error is larger than such a user defined fraction of the cell size.

For simplicity we assume that the finest triangulation contains the edges of

a regular grid. If this is not the case additional edges can be inserted. Furthermore, the grid dimensions are assumed to be powers of two. In the following we describe the simplification of the mesh by one LOD. This simplification is applied recursively to get coarser and coarser levels.

Coarse cells with coarse neighbors

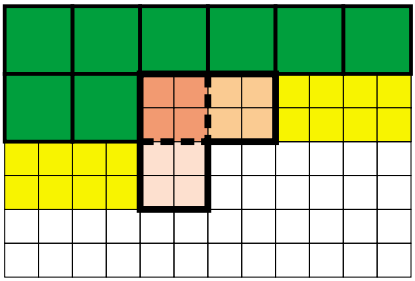


Figure 7.4: The dark red cell is the current cell. This cell is adjacent to two already simplified cells shown in green (the upper and the left one). Furthermore, the cell to the right like the current cell itself have already been partially simplified. The cell to bottom is not yet simplified. During the simplification of the current cell only the three reddish cells are considered. Edges on the border to the left and to the upper are not allowed to be collapsed. Half-edge collapses that would move a vertex away from the dashed cell boundary (and therefore would destroy the cell boundary) are forbidden.

In the first step the whole mesh is reduced to the next LOD. This means, that after the simplification, neighboring cells have the same LOD. To avoid loading the whole original mesh we scan the cells of the next level (consisting of four cells of the current level) line by line. The triangulations of the current cell as well as the triangulations of their neighbors together with the corresponding original triangulation are loaded, see Figure 7.4. Now simple half-edge collapses (a vertex is unified with an existing one) are performed, see Figure 7.5. Inside the current cell and inside its neighbors to the right and to the bottom all half-edge collapses are allowed. On the borders of the current cell (the black dashed ones in Figure 7.4) only that half-edge collapses are allowed that move vertices along the edge itself. This means the edge is preserved. On

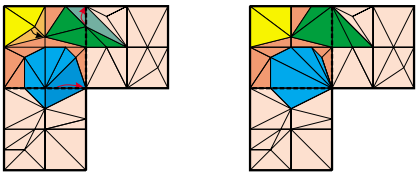


Figure 7.5: The figure shows an enlargement of the center part of figure 7.4. The simplification is performed by half edge collapses where one vertex is moved into the other vertex of an edge which destroys the edge and the two adjacent triangles. The vertices inside the current cell may move along each adjacent edge, the vertices on the dashed line only along the borders. Nevertheless, together with the current cell the neighbor cells on the other side of the dashed line are also influenced and partially simplified.

the other edges (the solid black ones in Figure 7.4) all half-edge collapses are forbidden. To select the next half-edge collapse a priority queue of possible half-edge collapses is build. This queue is sorted by the approximation error that would occur if the operation was performed [KS96].

Measuring the error - the reference mesh

In principle the approximation error of the current triangulation should be measured against the original triangulation. But then for coarser levels larger and larger parts of the original triangulation must be loaded. For huge databases the needed main memory is a serious problem. But since the approximation error of consecutive levels shrinks by a factor of 2 in each level accumulation of the errors does not lead to an unacceptable overestimation of the error. If always measure against the previous LOD the accumulated error can be at most two times the real error $(1 + \frac{1}{2} + \frac{1}{4} + \cdots \leq 2)$. If we measure against the level before the previous one the overestimation is at most $1\frac{1}{3}$ of the real error. Therefore, we measure the approximation error of the current triangulation always against the level before the previous one. We called the triangulation on the level before the previous one reference triangulation. As for fractal surfaces the number of triangles per cell remains approximately constant (zooming does not change the characteristics of the surface) the needed memory to load this triangulations is approximately independent of the level. Therefore, this algorithm can be performed even for huge data sets on an average PC.

Coarse cells with fine neighbors

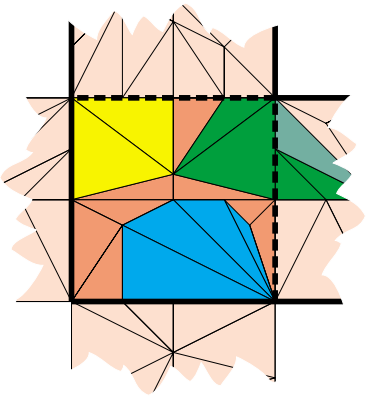


Figure 7.6: To generate the transition triangulations between cells at a coarse level and neighboring cells at finer levels the cell itself, the part of the reference mesh corresponding to the cell and the borders of the neighboring cells at a finer level are loaded into memory. The dark black lines indicate that in the current step the cell is refined along the borders to the top and to the right.

After the simplification of all cells to the next LOD the coarse to fine connections between the cells and their neighbors of the next finer level are computed. As in the case of the simplification with coarse neighbors we scan the cells line by line. The triangulation of the cell itself is loaded at the coarse level, the triangulations of the neighboring cells at the fine level, see Figure 7.6. As the cell itself is triangulated at the coarse level and the neighboring cells at the fine level, gaps may occur at the borders of the cell. Therefore, along the edges of the cell vertex splits are performed to close the gaps. Note that although the vertex splits normally decrease the approximation error in the coarse cell in rare cases the approximation error of the coarse triangulation may grow, see Figure 7.8. In such cases the vertex of the reference triangulation causing the maximum approximation error is inserted into the current coarse triangulation. As the maximum approximation error always occurs inside the cell this operation would not influence the triangulation at the border. In some cases this procedure must be repeated until the approximation error is

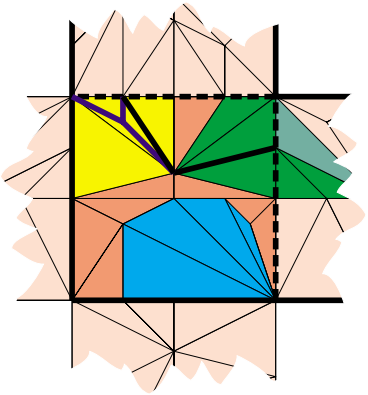


Figure 7.7: To close the gaps along the border vertex splits are performed until all vertices that are contained in the border of the adjacent cell are inserted into the triangulation. Sometimes it may be necessary to insert a vertex of the reference mesh inside the cell to guarantee the desired approximation error. In this example an inner vertex is inserted after closing the gap at the top.

below the corresponding threshold.

7.2.2 Storing the model

As described above together with the coarse level triangulation different triangulations are needed for each tile in order to provide a transition to neighboring tiles with finer levels of detail. Of course, parts of these triangulations may remain identical, when one or more borders are refined. We do not try do decompose the triangulation of the tiles into permanent parts in the center of the tile and changing parts along the borders. Instead of this we begin with generating generalized triangle strips for each geometry tile. A discussion of generalized triangle strips and an efficient algorithms to generate and render them using OpenGL can be found in [ESV96, Dyn01]. The triangulation defined by a triangle strip depends on the sequential ordering of the vertices. Therefore, for each tile all the vertices are transmitted in the order defined by the triangle strips.

All triangulations needed for a transition to neighboring cells with finer

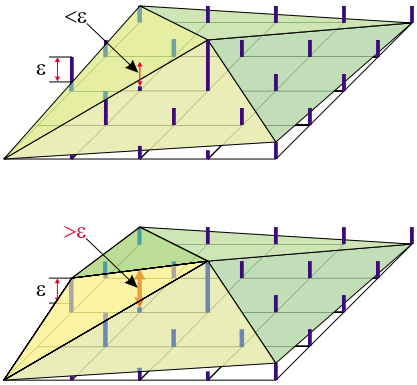


Figure 7.8: This example shows that the approximation error may increase if a vertex split along the border is performed. The blue bars indicate the height values at the vertices of the original mesh. The approximation errors are indicated by red arrowed lines. In this case the vertex of the reference mesh causing that error is simply inserted into the current triangulation. This procedure is repeated until the desired approximation error can be guaranteed.

level of detail can be derived from generalized triangle strips using a small modification. This is based on the following observations:

- To get the triangulation needed for a transition to a finer level of detail edges are only inserted, no edges are removed.
- If edges are inserted, one triangle is replaced by several triangles that can always be represented as a strip.
- This strip of newly created triangles can always be inserted into the existing strip, replacing the old triangle. For this it is sufficient to insert the new vertices and operations into the existing sequence. The reason is that only special retriangulations of triangles are needed, see Figure 7.9. No vertices have to be removed.

The modification of the generalized triangle strips is done in a preprocessing step: starting with the strips of the coarse triangulation the algorithm considers all new triangles that are created by inserting vertices in all

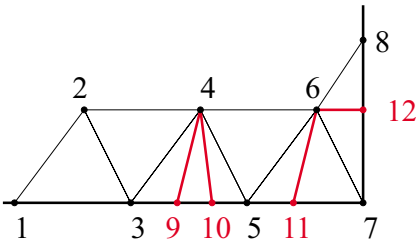


Figure 7.9: This example shows how a strip of the coarse triangulation can be modified to include the triangles needed to close the gaps along an edge to a adjacent cell at a finer level. The original strip is drawn in black. The sequence describing the inserted triangles are drawn in red.

configurations of neighbors with different levels of detail. The triangles are inserted into the existing strips as shown in Figure 7.9.

To generate the triangulation during rendering from the triangle strips, we must know which of the inserted operations are needed and which ones have to be skipped, depending on the level of detail of the neighbor tiles. For this purpose for each operation a transition-flag is stored which tells if the operation belongs to the original coarse triangulation (and is thus needed in any case) or if the operation is a refinement operation for the border. In this case two additional border-bits indicate on which border the newly inserted vertex is located. With this information it is straightforward to decide for each configuration, if the current operation is needed. In addition to the standard case described so far there are two special cases discussed in 7.A where also examples for the bit-coding are shown.

The great advantage of this approach is that during rendering after loading the tile (coded as efficient triangle strips) all possible transition triangulations are available without further loading.

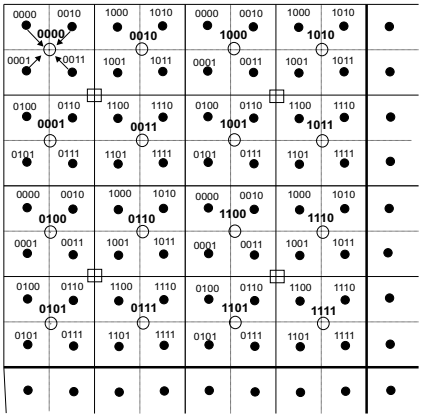


Figure 7.10: This example shows how the relative x, y/coordinates of points inside the quadtree cells are rounded. In this example the $x-$ and $y-$ directions are coded by 2 bits each. The dark point indicate the finest level, the empty circles and the rectangles the next coarser levels, respectively. During the rounding step the (x, y) position of four points are rounded into the same one. This new relative position within the coarser cell is easily computed by dropping the last two bits (one in each direction) and adding the 2 bits for the relative position of the cell in the finer level at the beginning.

7.2.3 Storing and transmitting the geometry data

The advantages of free triangulations generally are paid with the disadvantage that in addition to the height values, the (x, y)-coordinates of the points must be explicitly stored and cannot be calculated from the location in the hierarchy. However, a nice property in our approach is that a part of the (x, y)-coordinates is already implicitly given by the quadtree cell that contains the point. Furthermore, the accuracy needed for the coordinates in the cell is defined by the texture resolution of the cell. For example, if the texture consists of 64×64 texels and we assume that the size of a texel is not larger than one pixel in the image plane the accuracy must be $1/64$ of the side-length of a cell in order to guarantee an error of about one pixel. Therefore, the (x_i, y_i) coordinates of points within a given cell are always stored and transmitted with a fixed number of bits only, e.g. 6 bits (64 positions) for each direction. During simplification the rounding that is necessary from level to level can easily be

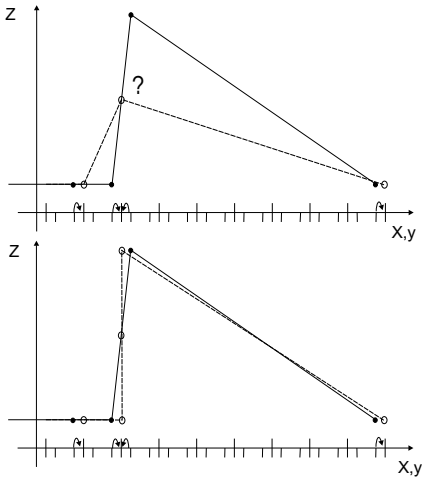


Figure 7.11: The upper figure shows that if the (x, y)-positions of neighboring vertices with different height values are rounded, the vertices can not be merged without introducing approximation errors. In the lower figure our solution is shown, where the same (x, y)-positions are shared by the two vertices.

done by dropping two bits and adding bits, see Figure 7.10. Although, this restricted accuracy leads to a high compression of the (x_i, y_i)-coordinates there is one problem to be solved. Consider two three or four neighboring points with (x_i, y_i)-positions that are rounded to the same (x, y)-coordinate, see e.g. Figure 7.10. If the height values of these points differ more than the distance between the points itself, the vertices and their height values are kept. In such a way it is ensured, that the accuracy of the approximation of the height values remains valid after the rounding step, see Figure 7.11.

The accuracy needed for the height values in a cell is also limited by the resolution of the texture. In contrast to the (x, y)-coordinates the relative position of the quadtree cell does not deliver additional information about the relative height of a vertex. But for the height values the situation is even better as for the (x, y)-positions: the triangulation of a cell approximates all height values of the (x_i, y_i)-vertices up to a certain error ϵ which is for example the distance between two texels. In the next finer level an approximation error of half a texel, i.e. $\epsilon/2$ is needed. Therefore, within triangles parallel to the x-y-

plane, one additional bit is sufficient to correct the interpolated height value. If the triangle is not horizontal, it depends on the slope, how many additional bits are actually needed. Triangles perpendicular to the x-y-plane cannot be used for interpolation (see Fig. 7.11). In these rare cases, all bits of the height value up to the current precision level are needed for the new vertex.

7.3 Rendering the model

For fast rendering it is important to process the data needed to generate an accurate image of the object and, to process only that data. Processing the data consists of two parts, loading the data into main memory and real time rendering of that data. To minimize the processed data, view-frustum culling and choosing the correct level of detail are the two main tasks.

7.3.1 The geometric and the screen space errors

To select the correct level of detail the precomputed geometric error between the original and the simplified model must be converted into a screen space error. Several elaborate techniques that not only consider the distance between the observer and the objects but also the viewing direction and the directions of precomputed errors are proposed in the literature [KCOH98, LKR+96a], but in our case the situation is much simpler. The only parameter to be considered is the distance to the observer. The reason is that we not only consider the geometric deviation to choose the level of detail but also the needed textures. For choosing the needed texture level the only criterion we use is the distance to the observer. This means that even if we look straight down on the terrain (and therefore could neglect height deviations) we want to choose the level of detail that contains the borders of the texture tiles.

7.3.2 The extraction algorithm

After loading the geometry data of the coarsest tile of the terrain its bounding box is calculated and tested against intersections with the view-frustum. This can easily be done by projecting the bounding box onto the screen and clipping the result with the view window. If the bounding box is visible it is checked if the current level of detail is sufficient. Else the tile must be subdivided. The algorithm so far is described by the following pseudo code.

```
process(tile)
  {
    load_geometry_data(tile);
    calc_bounding_box(bbox, tile);
    if (bbox intersects viewfrustum)
    {
      if (level_of_detail sufficient)
        {
        load_texture_data(tile);
        Render(tile);
        }
      else
        {
        subdivide(tile);
    process(subtile1); process(subtile2);
    process(subtile3); process(subtile4);
        }
    }
  }

main()
{
   process(coarsest_tile);
}
```

Of course, the real algorithm is more complicated. Before rendering the tile, we must ensure that the level of detail of adjacent cells differs by no more than one from the lod of the current cell and that the correct transition triangulations for the cell are used. Therefore, instead of rendering the tiles immediately, they are inserted into a quadtree data-structure. At the end of the recursive process we have a standard quadtree containing the absolutely necessary levels of detail. Some of its cells must be further subdivided to ensure that the level of details of adjacent cells do not differ by more than one level. We call this *closing* the quadtree. Note, that no additional textures are loaded for such cells as the closing is only required to ensure correct transitions of the geometry of adjacent cells. As texture the already loaded coarser texture tiles are sufficient[3]. In addition for each of the resulting cells the information which

[3]Of course, in this case the texture coordinates are adjusted accordingly.

of the neighbor cells is at a finer level is generated and then used to extract the correct triangulation for the cell including the transitions to the neighbors.

7.3.3 Frame to frame coherency

An important method to achieve the performance necessary for real-time fly-throughs is to exploit frame-to-frame coherency. For this purpose the quadtree of rendered tiles described in the previous section is used. In addition to the pointers to the data each cell holds the bounding box (maximum height value). When rendering a new frame, the quadtree is processed in the same way as described above. The only difference is that data already present can now be taken from main memory and the bounding boxes are already available.

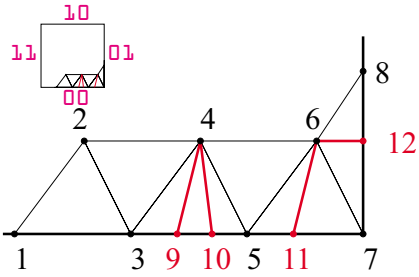
Caching and preloading is supported by supplementing the quadtree with additional cells that are marked as present but inactive. Such cells are not considered for closing the quadtree.

The principle that is used to decide, which cells are loaded and which ones can be discarded is, that in every level around the visible cells a surrounding one cell wide circle of invisible cells is kept in the buffer as inactive cells.

In this way smaller changes of the camera position often do not result in any new data to be loaded and the frame to frame coherency is exploited to a very wide extend.

7.A Coding the transitions

In our data format the strips consist of a sequence of vertices. Each vertex can be preceded or followed by a swap operation. Each vertex defines a new triangle together with the two previous vertices stored in a FIFO. The swap operation allows to change the order of the two vertices in the FIFO. While in the generalized triangle strips it is sufficient to allow swap operations either before or after a new vertex is given, in our case both must be possible. In this way our simple coding technique also works in the case where a triangle of the coarse triangulation belongs to two edges of the tile, see Figure 7.14. Note, that this technique is not necessary in a case where due to the error an additional inner vertex is inserted. In this case the inner vertex can be inserted first and connected to the corner.



□1,□2,□3,□4,□5,□6,□7,SWP,□8

□1,□2,□3,□4,1□□9,SWP1□□10,SWP,
□5,□6,1□□11,SWP,
□7,SWP,1□1□12,SWP,□8

Figure 7.12: The bit-coding for the example of figure 7.9. The transition-flags drawn in green indicate if the following operation belongs to the coarse triangulation (0) or to a transition triangulation(1). If the flag is 1 two additional border-bits drawn in magenta show to which border the operation belongs.

References

[DBH00] Jürgen Döllner, Konstantin Baumann, and Klaus Hinrichs. Texturing techniques for terrain visualization. In T. Ertl, B. Hamann, and A. Varshney, editors, *Proceedings Visualization 2000*, pages 227–234. IEEE Computer Society Technical Committee on Computer Graphics, 2000.

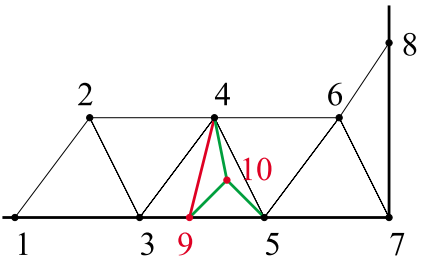
[DWS+97] Mark A. Duchaineau, Murray Wolinsky, David E. Sigeti, Mark C. Miller, Charles Aldrich, and Mark B. Mineev-Weinstein. ROAMing terrain: Real-time optimally adapting meshes. In *IEEE Visualization '97*, 1997.

[Dyn01] James Stewart Dynamic. Tunneling for triangle strips in continuous level–of–detail meshes, 2001.

[ESV96] Francine Evans, Steven S. Skiena, and Amitabh Varshney. Optimizing triangle strips for fast rendering. In Roni Yagel and Gregory M. Nielson, editors, *IEEE Visualization '96*, pages 319–326, 1996.

[HB87] Brian Von Herzen and Alan H. Barr. Accurate triangulations of deformed, intersecting surfaces. volume 21, pages 103–110, July 1987.

[HG97] Paul S. Heckbert and Michael Garland. Survey of polygonal surface simplification algorithms. Technical report, 1997. draft, to appear.



1,2,3,4,5,6,7,SWP,8

1,2,3,4,9,10,5,4,SWP,6,7,SWP,8

Figure 7.13: In addition to a vertex of the border also a vertex inside a triangle is inserted. The example shows how the newly created sequence is inserted into the original strip.

[Hop98] Hugues Hoppe. Smooth view-dependent level-of-detail control and its application to terrain rendering. In *Proceedings IEEE Visualization'98*, pages 35–42. IEEE, 1998.

[Hüt98] T. Hüttner. High resolution textures. *Visualization'98 - Late Breaking Hot Topics Papers*, pages 13–17, November 1998.

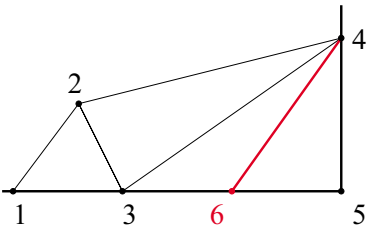
[KCOH98] R. Klein, D. Cohen-Or, and T. Hüttner. Incremental view-dependent multiresolution triangulation of terrain. *The Journal of Visualization and Computer Animation*, 9:129–143, August 1998.

[KH96] R. Klein and T. Hüttner. Simple camera-dependent approximation of terrain surfaces for fast visualization and animation. In R. Yagel, editor, *Visualization 96*. ACM, November 1996.

[KS96] R. Klein and W. Straßer. Generation of multiresolution models from cad-data for real time rendering. In W. Straßer, R. Klein, and R. Rau, editors, *Theory and Practice of Geometric Modeling*. Springer-Verlag, 1996.

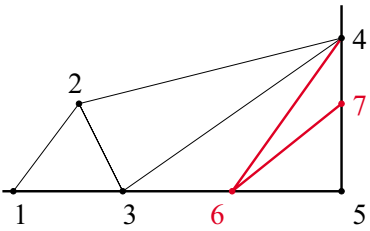
[LKR+96a] P. Lindstrom, D. Koller, W. Ribarsky, L. Hodges, N. Faust, and G. Turner. Real-time continuous level of detail rendering of height fields. *Proceedings of SIGGRAPH'96*, pages 109–118, 1996.

[LKR+96b] Peter Lindstrom, David Koller, William Ribarsky, Larry F. Hodges, Nick Faust, and Gregory A. Turner. Real-time, continuous level of detail rendering of height fields. In *SIGGRAPH '96*, pages 109–118, Aug. 1996.



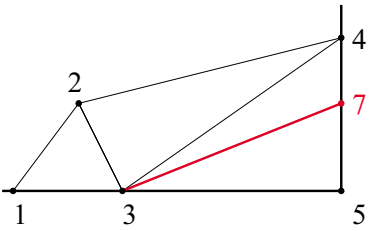
□1,□2,□3,□4,□5

□1,□2,□3,□4,1□□6,SWP,□5



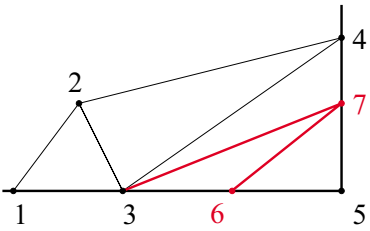
□1,□2,□3,□4,□5

□1,□2,□3,□4,1□□6,SWP,1□□1SWP, 7,□5



□1,□2,□3,□4,□5

□1,□2,□3,□4,1□□1SWP, 7,□5



□1,□2,□3,□4,□5

□1,□2,□3,□4,1□□1SWP, 7,1□□6,SWP,□5

Figure 7.14: On the left hand the original triangulation containing a corner triangle is modified by inserting a vertex on the right edge or on the bottom edge, respectively. To ensure that vertex 5 together with the edge (3,7) form a triangle a swap operation is necessary before *vertex seven. To merge two triangulations it is necessary to preserve the orientation of the last edge. Therefore, a swap is inserted after vertex 6. In this way it is possible to insert both operations in arbitrary order to get one of the two possible merged triangulations on the right hand figure. Of course, this works also in the more complicated case, where more than one vertex on an edge is inserted.*

[LP01] P. Lindstrom and V. Pascucci. Visualization of large terrains made easy. In *In IEEE Visualization 2001 Proceedings*, 2001.

[Paj98] Renato B. Pajarola. Large scale terrain visualization using the restricted quadtree triangulation. In David Ebert, Hans Hagen, and Holly Rushmeier, editors, *IEEE Visualization '98*, pages 19–26, 1998.

[Pup96] E. Puppo. Variable reolution of terrain surfaces. In *Proceedings Eight Canadian Conference on Computational Geometry*, August 1996.

[TMJ98] Christopher C. Tanner, Christopher J. Migdal, and Michael T. Jones. The clipmap: A virtual mipmap. In Michael Cohen, editor, *SIGGRAPH 98 Conference Proceedings*, Annual Conference Series, pages 151–158. ACM SIGGRAPH, Addison Wesley, July 1998. ISBN 0-89791-999-8.

■

Chapter 8

A Framework for the Acquisition, Processing and Interactive Display of High Quality 3D Models

Hendrik P. A. Lensch, Michael Goesele, Hans-Peter Seidel[*]

This article highlights some recent results on the acquisition and interactive display of high quality 3D models. For further use in photorealistic rendering or object recognition, a high quality representation must capture two different things: the shape of the model represented as a geometric description of its surface and on the other hand the appearance of the material or materials it is made of, e.g. the object's color, texture, or reflection properties.

The article shows how computer vision and computer graphics tech-

[*] Max-Planck-Institut für Informatik, Arbeitsgruppe Computergraphik, Saarbrücken, Germany. E-Mail: {lensch|goesele|hpseidel}@mpi-sb.mpg.de

niques can be seamlessly integrated into a single framework for the acquisition, processing, and interactive display of high quality 3D models.

8.1 Introduction

The rapid advances of consumer level graphics hardware make it possible to render increasingly complex and accurate models in real time. Computer-generated movies are getting more and more realistic and users will soon demand a similar level of realism in a wide range of every day applications such as computer games, digital libraries and encyclopedias, or e-commerce applications. Being able to efficiently generate, process and display the necessary models will become a more and more important part of computer vision and computer graphics.

To fulfill these requirements a high quality representation must capture two different things: the shape of the model represented as a geometric description of its surface and the appearance of the material or materials it is made of, e.g. the object's color, texture, or reflection properties. Subsequently, geometry and surface appearance data must be integrated into a single digital model which must then be stored, processed, and displayed, trying to meet several conflicting requirements (such as realism versus interactive speed).

As more and more visual complexity is demanded, it is often infeasible to generate these models manually. Automatic and semi-automatic methods for model acquisition are therefore becoming increasingly important.

A system built to acquire and to process the necessary data relies on computer vision techniques as well as on computer graphics techniques. To obtain the geometry of an object, a 3D scanner is used. The output is transformed into a mesh representation and further processed to reduce noise and complexity. The surface properties of the object are acquired by taking a number of images with constrained lighting. These images have to be registered to the 3D geometry by use of camera calibration techniques. By inspecting the images, the object's texture, the spatially varying reflection properties and microstructure (normal maps) can be extracted.

Combining all the data, a compact representation of the object can be obtained that allows for accurately shaded, photorealistic rendering from new viewpoints under arbitrary lighting conditions. In addition, the high quality 3D model may be used for object recognition and material investigation.

This article highlights some recent results on the acquisition and interactive

display of high quality 3D models. It shows how computer vision and computer graphics techniques can be seamlessly integrated into a single framework for the acquisition, processing, and interactive display of high quality 3D models. Some examples will illustrate the approach. Finally, we point out some remaining questions and important areas for future research concerning both computer graphics and computer vision.

8.2 3D Object Acquisition Pipeline

In this paper we focus on the generation of high quality 3D models containing the object's geometry and the surface appearance Such a model contains information needed for many computer graphics or computer vision applications. However, there are also other types of high quality models such as volumetric or image-based models (e.g., computer tomography data sets, light fields [LH96]) that are suitable for different applications.

In our case, the generation of a high quality 3D model for a real world object includes several, partially independent steps. Figure 8.1 shows an overview of these steps.

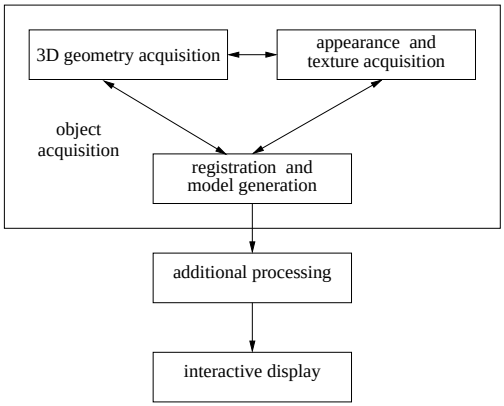


Figure 8.1: The 3D object pipeline. Depending on the applied techniques geometry acquisition, texture and appearance acquisition, and registration depend on each other in different configurations.

First, the geometry and the texture of the object are acquired. Typically, different techniques and acquisition devices for the geometry and the texture are applied which makes it necessary to align both data sets in a separate registration step. However, it is also possible to derive geometry information from texture data and vice versa. Various subsequent processing steps are necessary to extract information such as reflection properties or normal maps from the input data.

Once a complete model is created it can be resampled, converted to a different data representation, or compressed to make it suitable for a particular application scenario. Finally, the target application should be able to display the model interactively without omitting any important information.

In the following sections we give a detailed description of all the steps of the 3D object pipeline. In Section 8.3 we start with image-based acquisition techniques followed by acquisition techniques for appearance properties in Section 8.4. We give an overview over the acquisition of 3D geometry in Section 8.5 and describe a technique to register texture and image data in Section 8.6. Section 8.7 introduces several methods to display the acquired models interactively. We present some examples of acquired models in Section 8.8 before we conclude with Section 8.9.

8.3 Image-Based Acquisition Techniques

Today, image-based techniques become more and more popular to acquire models of real world objects (see Section 8.4). A key element of these methods is a digital camera to capture images of the object from which various properties of the object can be derived. The large number of measurements that can be made in parallel (i.e. one per pixel for a digital camera) lead to efficient methods to sample complex functions such as four-dimensional BRDFs. However, these measurements can only be meaningful if the equipment used is appropriate for the measurements, if the properties of the devices are known, and if the relevant parts are calibrated.

8.3.1 Photographic Equipment

Both analog and digital cameras can be used for measurement purposes. The advantages of analog photography include the high resolution of analog film (especially in combination with commercial high quality digitization services

as the Kodak Photo CD), its comparably large dynamic range, and the huge selection of available cameras, lenses and film types. However, the development and scanning of film can take quite long and the resulting images are not naturally registered against the camera lens system.

In contrast to that, the position of the imaging sensor in a digital camera remains fixed with respect to the lens system which makes it easy to capture several aligned images from the same position under different lighting conditions. If the digital camera is capable of returning the raw image sensor data it is possible to calibrate the individual sensor elements to account for variations on the sensor [Abb95, GHS01b].

Most consumer quality digital cameras use the lossy JPEG compression format to store their images although more recent cameras are often also capable of producing images in a lossless compressed format. The lossy JPEG compression introduces compression artifacts which makes them rather unsuitable for measurement purposes. Additional artifacts can occur due to various steps in the image processing chain of digital cameras such as sharpening operations or the color reconstruction in single chip cameras. The imaging community developed a large number of methods to characterize various aspects of a digital camera such as the modulation transfer function (MTF) [WB01]. These methods are not only helpful to choose an appropriate camera but can also be used to debug a measurement setup when an error occurs.

8.3.2 Lighting Equipment

For most algorithms that reconstruct the appearance properties of an object from images, it is important to control the lighting conditions exactly. Although this is also true for images taken by a regular photographer, the requirements differ strongly. A point light source, i.e. a light source where all light is emitted from a single point is ideal for many of the techniques mentioned above but is rarely used in photography as it casts very hard shadows. A perfectly constant and diffuse lighting is ideal to capture the color of an object but leads from a photographers point of view to very flat looking images due to the absence of shadows.

The surrounding of an object has also a huge influence on the lighting situation, especially if the object has a specular reflecting surface. In order to minimize this influence the measurement region should be surrounded with dark material that absorbs as much light as possible. Furthermore, the light that is not absorbed should be reflected in a very diffuse way. Figure 8.2 shows

Figure 8.2: A view of our photo studio with black, diffuse reflecting material on the floor, walls, and ceiling. This image was generated from a High Dynamic Range image to which a tone-mapper has been applied.

a view of our photo studio whose floor, walls, and ceiling are covered with black, diffuse reflecting material to reduce the influence of the environment on the measurements as much as possible.

A more technical and in-depth discussion of camera and lighting issues can be found in [GHLS00].

8.3.3 Camera Calibration

When using a camera as a measurement device various aspects should be calibrated in order to guarantee high-quality results and the repeatability of the measurements.

Geometric Calibration

The properties of the camera transformation which describes how an object is projected onto the camera's image plane should be recovered e.g. using [Tsa87, Zha99, HS97]. These methods generally use an image or a set of images of a calibration target (e.g. a checkerboard pattern) to determine camera parameters such as the focal length of the lens, the location of the optical axis relative to the imaging sensor (principal point), and various distortion

coefficients. Once this information is known, a ray in space can be assigned to each pixel in an image.

High Dynamic Range Imaging

The dynamic range of a camera, i.e. the ratio between the brightest and the darkest luminance sample that can be captured in a single image, is for most cameras quite small (on the order of $10^2 - 10^3$). As the dynamic range of a scene can be much higher (e.g., about 10^6 between highlight and shadow regions), some techniques have to be used to capture the full dynamic range of a scene.

Several manufacturers have developed CMOS cameras that are capable of capturing a sufficiently large dynamic range by either combining multiple exposures or by the use of special imaging sensors. These cameras are typically video cameras and provide only a limited resolution. Furthermore, the measured values are quantized to 8–12 bits per pixel and color channel leading to a rather low precision.

In the computer graphics community, several authors proposed methods to extend the dynamic range of digital images by combining multiple images of the same scene that differ only in exposure time. Madden [Mad93] assumes linear response of the imaging sensor and selects for each pixel an intensity value from the brightest non-saturated image. Debevec and Malik [DM97] and Robertson et al. [RBS99] recover the response curve of the imaging system and linearize the input data before combining them into a single high dynamic range image. In [GHS01a], Goesele et al. proposed a technique to combine high dynamic range imaging with color management techniques (see Section 8.3.3).

Color Issues

Accurately recording the continuous spectrum of the visible light is difficult – especially if the spectrum is not smooth but contains sharp peeks such as the spectrum of a discharge lamp or even a laser. Likewise, the spectral response curve that describes the way light is reflected by an object is not always smooth. Measurement devices such as a spectrophotometer perform therefore a very dense sampling of the spectrum and output large data sets.

In contrast to that, most analog and digital cameras record only three color values per pixel (tristimulus values). Each sensor in a digital camera integrates

the amount of incoming light weighted by its response curve over the whole visible spectrum. This is inspired by the human visual system that also contains three types of sensors behaving in a similar way [Hun95]. A camera can record the colors of objects as perceived by a human observer most accurately if the corresponding response curves are identical [Lut27], but the true spectrum of the light hitting the sensor can never be reconstructed and different spectra can result in the same tristimulus values (metamerism). Color measurements done with a tristimulus device are therefore always an incomplete representation of the actual spectrum.

White Balance

The human visual system can adapt to a wide range of illumination conditions. Within this range, colored objects look roughly the same even if the spectrum of the light source changes and therefore the spectrum of the reflected light hitting the retina is different. A digital camera can mimic this behavior with a white balancing step: the tristimulus values are multiplied with constant factors so that the color of the light source is recorded as white. The influence of the light source on the recorded color of an object is hereby minimized.

Color Management Systems

For a digital camera, the recorded color of an object depends not only on the light source but also on several other factors including the properties of the optical system, the sensor, and the image processing steps applied by the camera itself or other software.

In order to relate the recorded color to well defined standards, color management systems have become a standard tool. An image of a well known test target such as the IT8.7/2 target (see Figure 8.3) is taken and processed in the same way all later images are processed. The relation between the color values of the test target patches and the color values reported by the camera is analyzed and used as calibration data. The International Color Consortium (ICC) introduced the so called ICC profiles [ICC98, Wal] as a standard way to store this information.

The basic mechanism behind ICC based color management systems is to use a well defined color space as profile connection space (PCS). All input data is converted into the PCS using an ICC input profile associated with the input

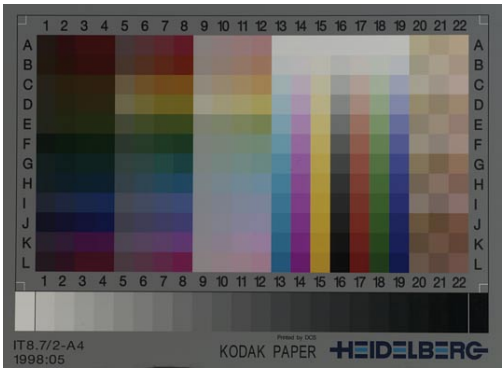


Figure 8.3: IT8.7/2 target used to capture the color properties of an imaging system in order to generate an ICC profile.

device. Other profiles are used to convert data from the PCS into the color space of display or output devices such as monitors and printers.

One of the color spaces used as PCS is the linear CIEXYZ space [CIE86]. In [GHS01a], Goesele et al. have shown that this color space can be used to generate color calibrated high dynamic range images which are a tool to improve the color fidelity of appearance acquisition methods.

8.4 Appearance Acquisition

The appearance of an object consists of several surface properties including color, texture, reflection properties, and normal directions or the local tangent frame in the case of anisotropic materials. Due to their large number they are difficult to acquire but nevertheless necessary to generate a convincing looking representation of an object. It is therefore justifiable to put a lot of effort into this acquisition step.

Traditionally the appearance of an object is captured using a variety of special devices [HH87]. But many surface properties can be acquired by the use of a photographic camera – preferably a digital camera – in a controlled lighting setup. Captured images can for example be used to color the 3D geometry model during rendering. The digital pictures are simply projected onto the

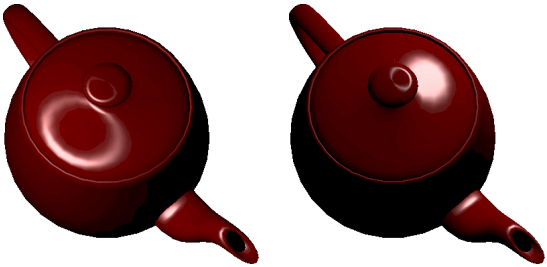


Figure 8.4: A teapot with complex reflection properties illuminated from two different directions.

model as image textures using texture mapping [HS93]. To ensure that each part of the object is colored, a sufficient number of images must be taken from different view points [MK99, Stu99]. During the projection a perspective correction must be performed to gain a seamless transition between textures of different images (see also Section 8.6). To obtain more precise surface properties than just a single color value, further processing is needed.

8.4.1 Reflection Properties

Constant, diffuse lighting during the acquisition phase would reproduce only the object's color. More realistic models can be obtained by considering further aspects of a material's appearance, for example the reflection properties. The intensity and color of any material typically varies if viewed from different directions or under different illumination (see Figure 8.4).

When light interacts with a perfectly reflective surface, i.e. a mirror, the reflected light leaves the surface at the same angle it hits the surface. However, perfect mirrors do not exist in reality. In contrast, most surface have a very complex micro-structure. This micro-structure makes different materials appear differently.

When light hits such a surface, it is not reflected toward a single direction, but rather to a cone of directions. If the surface is perfectly diffuse (e.g. for a piece of chalk), light even scatters equally in all directions.

In computer graphics the *bidirectional reflectance distribution function* (BRDF or also reflectance model) is used to describe the way a surface re-

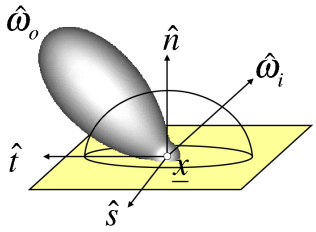


Figure 8.5: Here you can see the values of a BRDF (depicted as a lobe) for one incident light direction $\hat{\omega}_i$ and every possible outgoing direction $\hat{\omega}_o$.

flects light. The BRDF yields the fraction of light arriving at a point from one direction to the light that is reflected off the surface at the same point into an exitant direction.

Hence a BRDF is a four-dimensional function $f_r(\hat{\omega}_o, \hat{\omega}_i)$ that depends on the incident light direction $\hat{\omega}_i$ and the viewing direction $\hat{\omega}_o$ (see Figure 8.5). It should be noted, that it also depends on the wavelength, which is usually represented by three samples (RGB) only. In the following, the wavelength dependency is not stated explicitly.

A number of analytical BRDF models have been developed to approximate the reflection properties of real materials (e.g. [TS67, War92, LFTG97, Ban94]).

8.4.2 Measuring Reflection Properties

In addition to these analytical models, it is possible to measure real-world BRDFs directly. There are special devices available to accomplish this task: The most general approach is to use a gonioreflectometer which measures the light that is emitted in every direction when the object is illuminated from a given direction. However, this measurement procedure can be very time consuming and captures only the properties of a single point on the surface of an object. If the surface is not uniform, this is not very helpful.

One way to overcome the ”single point” constraint for appearance measurements is the use of a digital camera. When an image is taken with such a camera it corresponds to millions of parallel measurements of radiance samples hitting the sensor. The main challenge is to recover the appearance information from

images taken from different positions under controlled lighting conditions.

Marschner [Mar98] used this approach to determine a single BRDF for an object by combining all the pixel data. Compared to a gonioreflectometer this technique is considerably faster, but it still assumes that the entire object consists of a single material, represented by a large number of tabulated BRDF samples. A specific BRDF model can be fitted to these BRDF samples by optimizing for the parameters of the BRDF model as it is for example done in [SHR+99]. The set of BRDF samples is then replaced by a few parameters resulting in a more compact representation.

To allow for variations of the reflectance properties over the object's surface Marschner et al. [MGR00] extracted the purely diffuse part (albedo map) of the object's texture for each visible point using a similar technique. The resulting texture includes only view-independent color information and no specular reflection. Albedo maps plus one reflection model per surface patch have been acquired for indoor scenes by Yu et al. [YDMH99] which assumed that material properties only change from patch to patch.

An approach to acquire distinct reflection properties for every surface point has been published by Debevec et al. [DHT+00]. A set of images of an object, e.g. a person's face, is taken from one viewpoint while the position of a point light source is changed. Hereby, the set of incident light directions is densely sampled. The collected data allows for realistic relighting of the object illuminated by arbitrary virtual environments. Unfortunately, a very large amount of data is needed both during the acquisition and for display.

8.4.3 Measuring Spatially Varying BRDFs

Based on Marschner's approach, Lensch et al. [LKG+01] developed a technique that is able to reconstruct spatially varying reflection properties by just a very few images (around 25). The key idea here is that most objects typically consist of a small number of materials only, i.e. many points on the object's surface have approximately the same reflection properties. By clustering points with different normals but consisting of the same materials, a large number of BRDF samples of that material can be collected by just a few images. After measuring the BRDF for clusters of points, separate reflection properties for each single point are determined to account for subtle details and small changes. The BRDF for each point is determined as a weighted sum of the clusters' BRDFs.

Thus, a high quality and very compact representation of the original object can be obtained with moderate acquisition effort.

Data Acquisition

The entire procedure is as follows: The geometry of the object is obtained by use of a 3D scanner, e.g. a structured light or computer tomography scanner, yielding a triangle mesh. In order to capture the reflection properties a small number of high dynamic range (HDR) images of the object are taken showing the object lit by a single point light source. In a next step the camera position (see Section 8.6) as well as the light source position relative to the geometric model are recovered for all images.

For every point on the object's surface all available data (geometric and photometric) is collected from the different views in a data structure called *lumitexel*. It contains the position of the surface point and its normal derived from the triangular mesh. Additionally, a lumitexel stores a list of radiance samples together with the corresponding viewing and lighting directions, one radiance sample for every HDR image where the point is visible and lit. The radiance sample is obtained by resampling the color value at the position of the surface point projected into the image.

Clustering of Materials

Because only a limited number of different views and lighting directions is acquired a single lumitexel does not carry enough information to reliably fit a BRDF model to the radiance samples. To provide more data from which the parameters can be derived, the lumitexels are grouped into clusters of similar materials. Starting with a single cluster containing all lumitexels, the parameters of an average BRDF are fitted using the Levenberg-Marquardt algorithm to perform a non-linear least square optimization.

In order to separate the distinct materials the initial cluster has to be split. Given the average BRDF, two new sets of parameters are generated by varying the fitted parameters along the direction of maximum variance, yielding two slightly distinct BRDFs.

The lumitexels of the original cluster are then assigned to the nearest of these BRDFs, forming two new clusters. A stable separation of the materials in the clusters is obtained by repeatedly fitting BRDFs to the two clusters and redistributing the original lumitexels. Further splitting isolates the different

materials until the number of clusters matches the number of materials of the object as illustrated in Figure 8.6.

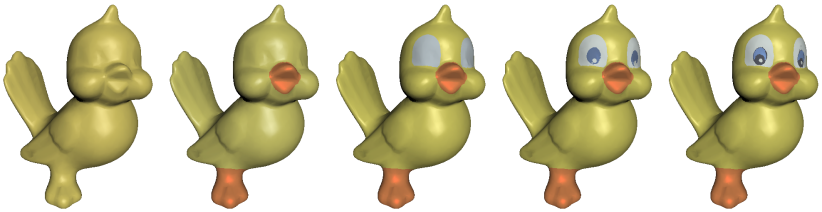


Figure 8.6: The clustering process at work. In every image a new cluster was created.The object was reshaded using only the single BRDFs fitted to each cluster before the projection into a basis of multiple BRDFs.

Spatially Varying Behavior

After the clustering the same reflection behavior is assigned to all lumitexels/points in one cluster. However, small features on the surface and smooth transition between adjacent materials can only be represented if every lumitexel is assigned its own BRDF.

In the algorithm, this BRDF is a weighted sum of the BRDFs recovered by the clustering procedure. The spatially varying reflection properties can be represented by a set of basis BRDFs for the entire model plus a set of weighting coefficients for each lumitexel.

The weighting coefficients are found by projecting the lumitexel's data into the basis of per cluster BRDFs. An optimal set of weighting coefficients minimizes the error between the measured radiance and the weighted sum of radiance values obtained by evaluating the basis BRDFs for the viewing and lighting direction of the measured sample. To recover the coefficients the least square solution of the corresponding system of equations is computed using singular value decomposition (see [LKG+01] for more details).

In Figure 8.7 the result of projecting the collected data for every point into a basis of BRDF is shown. The method allows for accurately shaded, photorealistic rendering of complex solid objects from new viewpoints under arbitrary lighting conditions with relatively small acquisition effort. The reconstructed BRDFs can further be used to classify the objects based on their materials.

Figure 8.7: Left: Last result of the clustering step. Right: Bird with the spatially varying BRDF determined by projecting each lumitexel into a basis of BRDFs. Note the subtle changes of the materials making the object look realistic.

8.4.4 Normal Maps

The resolution of the acquired geometry of an object is typically limited by the used 3D scanning device (see Section 8.5). Additional processing of the 3D data like combining multiple scans, smoothing the surface to remove noise, and mesh simplification to reduce the complexity of the model further erases fine scale geometric detail.

When reconstructing the object using a coarse geometric model, smaller features in the surface's structure like bumps, cracks or wrinkles can be simulated by the use of normal maps or bump maps [Bli78] (see Figure 8.14). These textures store a perturbation of the surface normal for each surface point. After applying the perturbation, the modified normals are used for the lighting calculations. This results in a change of the angle between the viewing direction and the surface at that point as well as between the light direction and the surface. This step approximates the correct lighting of a fine scale geometry model.

Normal maps recording small imperfections of the surface can be acquired

for real world objects: Rushmeier et al. calculated normal directions from a set of images showing the same view of the object illuminated by a point light source placed at different but known positions for each image [RTG97]. The surface is assumed to be perfectly diffuse (Lambertian), reflecting incident light equally in all directions, and thus its color can again be represented by an albedo map [RBMT98].

The restriction of a purely diffuse surfaces can be removed if techniques like [LKG+01] (see Section 8.4.3) are used to first measure the approximate reflection properties at each surface point and then use this data to measure the normal directions.

Since the BRDF at one point is defined for viewing and lighting directions with respect to the local tangent frame at that point, all directions have to be transformed based on the point's surface normal. To measure the exact normal at a point, an initial normal is obtained from the triangular mesh. Given the viewing and lighting directions for the radiance samples in world coordinates, the current estimate of the normal is used to transform them into the local coordinate frame. Then, the error between the measured radiance values and the reconstructed radiance values is computed where the reconstructed radiance values are obtained by evaluating the measured BRDF using the transformed directions. If enough radiance samples are provided for each point the actual normal direction at the point can be found by minimizing this error using a non-linear least square optimization technique. Figure 8.8 shows the quality of the reconstructed normals compared to the normals of the original mesh.

8.5 Acquisition of 3D Geometry

In most cases there exists no high quality 3D geometry model of real world objects like pieces of art. But even if it would exist (e.g. because the object was manufactured using computer based manufacturing methods) it is often only available to a very limited number of persons. Therefore, it is most often necessary to acquire the geometry of objects using a 3D scanner.

Several research groups including [LPC+00, BMR99] have built their own 3D scanner – some of them tailored to specific requirements. Furthermore, there is a broad range of commercial products made by companies like Cyberware, Minolta, or Steinbichler.

There are several different approaches to acquire the 3D geometry of an object (for an overview see [CS00]) but most of the systems for small or medium

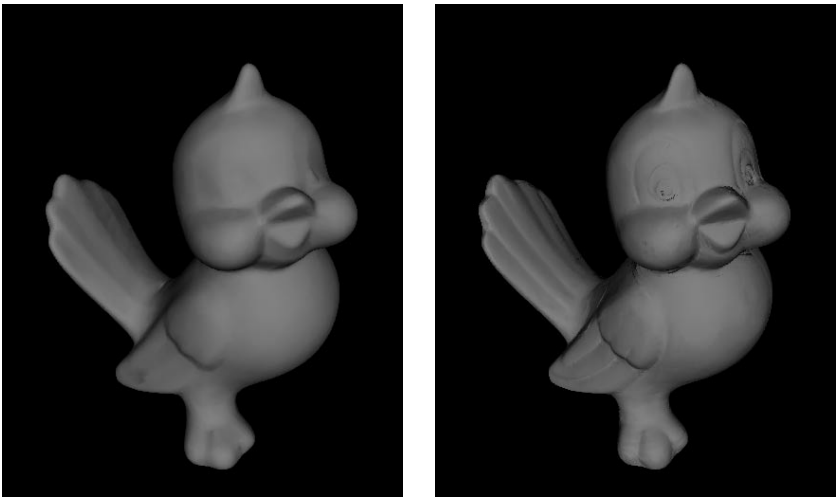


Figure 8.8: Left: Normals of the original mesh. Right: Normals optimized using spatially varying BRDFs

sized objects are based on an active stereo structured light approach. One or several patterns are projected onto the object with a computer controlled projection system (e.g. a video projector, a color coded flash stripe projector, or a laser beam). The projected light patterns on the object are observed by a digital camera which is rigidly connected to the projection system. The 3D location of a point on the surface of an object is then defined by the intersection of a ray from the projected pattern with the viewing ray that corresponds to the pixel in the digital image that observed this ray (see Figure 8.9).

The position of these rays in space is determined in a separate calibration step: The patterns are projected onto a calibration target – typically a flat board or a three-dimensional structure with a regular pattern whose geometric properties are exactly known. The acquired images are analyzed to recover the intrinsic parameters (e.g. focal length, lens distortion) and extrinsic parameters (the relative position and orientation) of the projection system and the camera using standard camera calibration techniques (e.g. [Tsa87, Zha99, HS97]).

Using the active stereo approach most objects cannot be acquired with a single scan either because front and back part of the object cannot be scanned with a single scan or because for a given configuration not all parts of the object

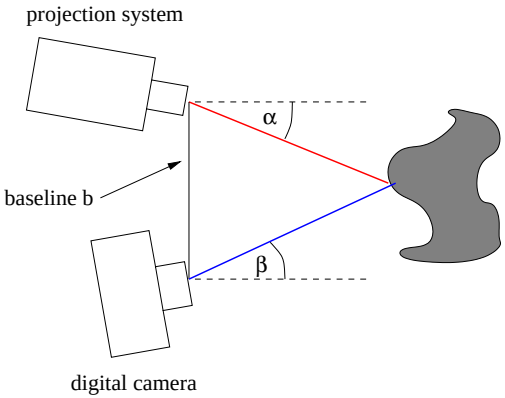


Figure 8.9: Schematic drawing of an active stereo 3D scanner. Given the intrinsic parameters of the projection system and the camera, the baseline b and the angles α and β, the position of a surface point can be recovered using triangulation.

are visible from both the position of the projection system and the digital camera. Therefore several scans have to be registered against each other in order to combine them into a single set of surface points. This is commonly done using a variant of the iterative closest point method (ICP) [BM92, PDH+97]. The resulting point cloud is triangulated leading to a single triangular mesh using one of a large variety of methods (for an overview see [CS00]). Further processing steps include smoothing to reduce noise (e.g. using [Tau95, Kob96]) and editing of the resulting mesh for which a huge selection of tools is available including [KCVS98].

Kobbelt et al. [KBB+00] give a detailed description of the techniques used for the acquisition and processing of 3D geometry data.

8.6 Registration of Geometry and Texture Data

Since texture and geometry are typically acquired by two different processes the collected data has to be merged afterwards. This requires the alignment of the geometry data and the captured images. Only for scanning devices that

capture geometry and texture data with the same sensor, the alignment or registration is already given. But in such a case the user is limited to the texture data provided by the scanner and the lighting setup cannot be changed to perform appearance measurements. Because of this, we further consider the case of two different sensors, a 3D scanner and a digital camera.

8.6.1 Manual Registration

In order to align or register the 3D model to the texture data one has to recover the parameters of the camera transformation that maps points in 3-space (the 3D geometry) onto the 2D image. These parameters describe the camera position, its orientation and the focal length (see Section 8.3.3). Further parameters are the aspect ratio, the principle point and the lens distortion, which are in the following assumed to be already known.

A simple approach to recover the camera position and orientation is to manually select corresponding points on the geometric model and in the picture [RCM99]. If enough correspondences are established the transformation can be directly determined using one of various kinds of camera calibration methods (e.g [Tsa87, Zha99, HS97]). But selecting corresponding points for a set of images is a time-consuming and tedious task. Additionally, the precision is limited by the user, although accuracy could be improved by selecting more points.

8.6.2 Automatic Registration

In order simplify the registration process some semi-automatic approaches have been published [MK99, NK99]. The user is asked to roughly align the 3D model to the image. The algorithm then tries to optimize for the camera parameters by minimizing the distance between the outline of the 3D model rendered with the current set of camera parameters and the outline of the object found in the image. For each tested set of camera parameters the distance between the outlines has to be computed. This is a time-consuming step since the 3D model has to be rendered, its outline must be traced and for some points on it the minimum distance to the other outline must be computed.

In [LHS00], Lensch et al. proposed a method to compute the distance between a view of the 3D model and the 2D image in a different way. Here, silhouettes are compared directly instead of using their outlines. At first the silhouette of the object in the images is extracted by classification of the image

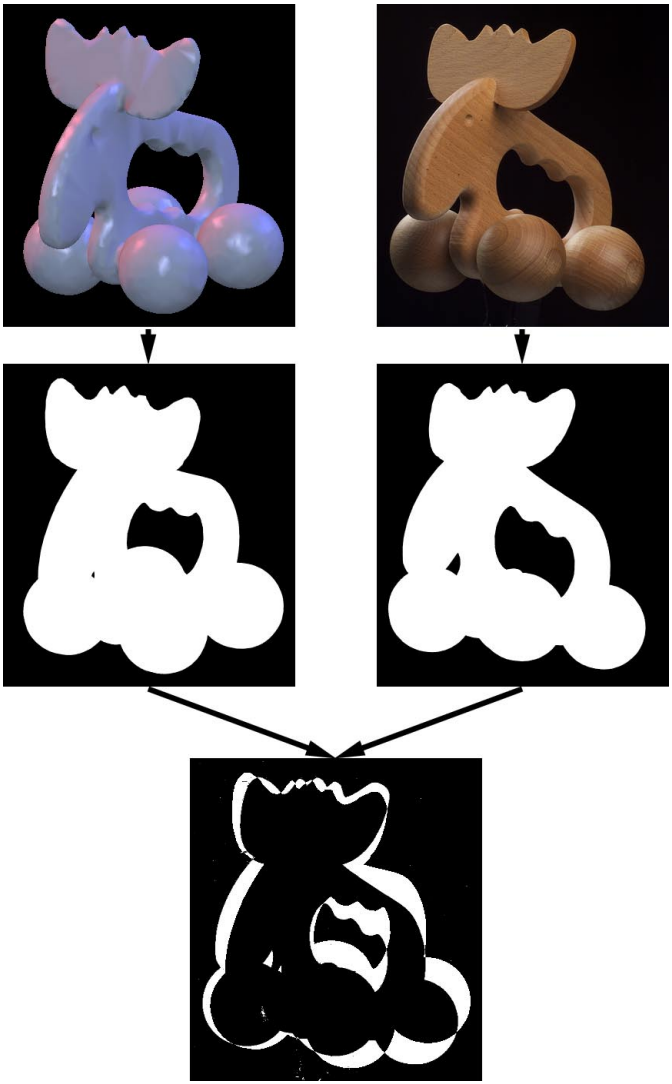


Figure 8.10: Measuring the difference between photo (right) and one view of the model (left) by the area occupied by the XOR-ed foreground pixels.

in foreground and background pixels, which can be done by any segmentation algorithm. Then, the geometry is rendered in front of a black background using a monochrome color. It is combined with the segmented image using the XOR-operation as is visualized in Figure 8.10. The resulting image will be black except for those pixels which are covered by just one silhouette but not by the other, that is to say exactly those pixels where the silhouettes differ. The number of remaining pixels is a measure for the distance between the silhouettes. These pixels can be counted by evaluating the histogram. The optimal set of camera parameters can be found by minimizing the number of remaining pixels.

Note that all three steps, rendering, combining, and histogram evaluation can be performed using graphics hardware and thus can be computed very fast, speeding up the optimization.

Additionally, it is also possible to automatically find a rough initial guess for the camera parameters. The effective focal length is first approximated by the focal length of the applied lens system. Depending on the focal length and the size of the object, the distance to the object can be approximated. It is assumed that the object is centered in the image. What remains to be estimated is the orientation of the camera. The optimization is simply started for a number of equally distributed sample orientation allowing just a few optimization steps per sample. The best result is then taken as a starting point for further optimization.

8.6.3 Texture Preparation

Knowing all camera parameters or the entire camera transformation for one image, it can be stitched onto the surface of the 3D model. The image is projected onto the the 3D model using projective texture mapping. Given a triangular mesh the stitching is done by computing texture coordinates for each vertex of the model that is visible in the image. Texture coordinates are calculated by projecting the 3D coordinates of the vertices into the image plane using the recovered camera transformation. All visible triangles can then be textured by the image as shown in Figure 8.11.

Further, the exact transformation for projecting surface points into the images is known. This information is required when collecting all radiance samples for one point on the objects surface into a lumitexel (compare Section 8.4.3).

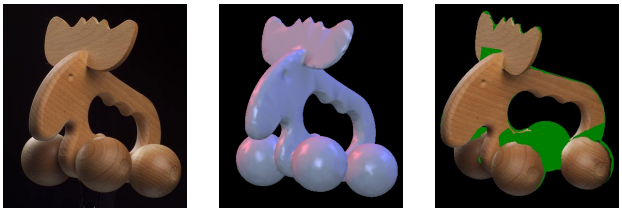


Figure 8.11: The 3D model is aligned to a captured picture which then can be mapped as a texture onto the geometry.

A task that is still left is to determine the set of surface points for which a lumitexel should be generated. In order to obtain the highest quality with respect to the input images, the sampling density of the surface points must match that of the images. To achieve this, every triangle of the 3D model is projected into each image using the previously determined camera parameters. The area of the projected triangle is measured in pixels and the triangle is assigned to the image in which its projected area is largest. For every pixel within the projected triangle a lumitexel is generated. The position of the surface point for the lumitexel is given by the intersection of the ray from the camera through the pixel with the mesh (see Figure 8.12).

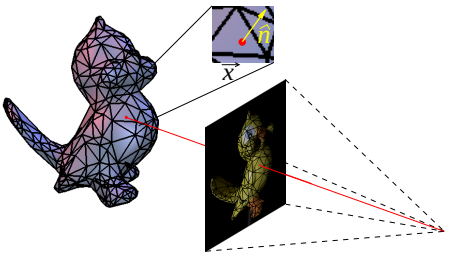


Figure 8.12: The correspondence between pixel position and point position on the object is computed by tracing a ray through the image onto the object.

Since every lumitexel is assigned to a triangular region within one of the HDR images it is possible to construct a 2D texture of lumitexels. This texture will unfortunately consist of a large number of separate triangles. Larger

patches can be obtained by grouping adjacent triangles of the same input image. However, a significant number of isolated regions will remain. Instead of treating these regions as independent textures, it is more convenient to pack the regions into a single image, e.g. using the technique proposed by Rocchini et al. [RCM99]. A result of this packing is shown in Figure 8.13 where the original color values of the input images are used to show the regions for which lumitexels are constructed.

During texture generation all parts of the original images where only the background is visible are discarded. Combined with dense packing of the remaining parts into one image, this reduces the size of the texture compared to the overall volume of the original images. A single image has the further advantage that it can be compressed and transformed into a streamable representation with less effort.

Figure 8.13: Packing of the constructed texture regions for the elk model. Only three pictures were considered in this case to better visualize the layout .

8.7 Interactive Display

After measuring the reflection properties of the object and transforming the images into a single texture, we explain in this section how the combined data can be displayed interactively.

8.7.1 Rendering with Arbitrary BRDFs

At first we will investigate the case of one homogeneous material, i.e. one BRDF per object. Standard OpenGL only supports the empirical and physically implausible Phong model, which makes surfaces always look "plastic"-like.

In order to render surfaces with other BRDFs two similar approaches [HS99, KM99] can be used. Both approaches decompose the four-dimensional BRDF $f_r(\hat{\omega}_o, \hat{\omega}_i)$ into a product of two two-dimensional functions $g(\hat{\omega}_o)$ and $h(\hat{\omega}_i)$. These two functions are stored in two texture maps and re-multiplied using blending. The approach by Heidrich and Seidel [HS99] decomposes the analytical Cook-Torrance model [CT81]. The approach by Kautz and McCool [KM99] numerically decomposes (almost) any BRDF by choosing a better parameterization for the BRDF.

Rendering is very simple. For every vertex of every polygon you have to compute $\hat{\omega}_o$ and $\hat{\omega}_i$ and use it as texture coordinates. Then the polygon has to be texture mapped with the textures containing $g(\hat{\omega}_o)$ and $h(\hat{\omega}_i)$ and the computed texture coordinates. Blending has to be set to modulate, so that $g(\hat{\omega}_o)$ and $h(\hat{\omega}_i)$ are multiplied together.

For an example of this technique, see Figure 8.4.

8.7.2 Rendering with Normal Maps

Blinn [Bli78] has shown how wrinkled surfaces can be simulated by only perturbing the normal vector, without changing the underlying surface itself. The perturbed normal is then used for the lighting calculations instead of the original surface normal. This technique is generally called bump mapping.

A new algorithm has been proposed to render bump maps [Kil00] (as shown in Figure 8.14) at interactive rates using texture maps containing per-pixel normals, which are used to perform the lighting calculations instead of per-vertex normals.

This algorithm relies on features now supported by many graphics cards. These features include per-pixel dot-products, multiplication, addition, subtraction, so lighting models/BRDFs using only these operations can be used to do bump mapping.

Usually the Blinn-Phong model [Bli77] is used to perform bump mapping, because this model mainly uses dot-products. For more details, please see [Kil00].

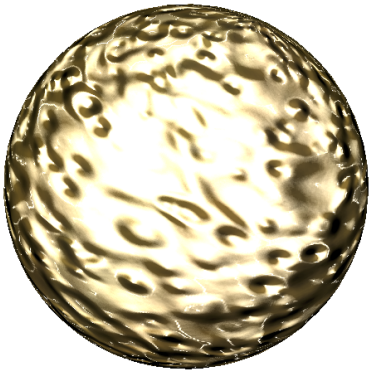


Figure 8.14: A normal map applied to a sphere

Heidrich et al. [HDKS00] also computed consistent illumination on bump maps in fractions of a second exploiting regular graphics hardware.

8.7.3 Spatially Varying BRDFs

One reflection model per surface can be evaluated very fast using the approach presented in [HS99, KM99]. If the reflection properties vary across the surface spatially varying BRDFs must be considered which have been interactively rendered by Kautz et al. [JS00], see Figure 8.15.

Some of these algorithms take advantage of new features of current graphics hardware, e.g. multi texturing and texture combiners [NVI99]. Although they are currently not available on all client machines they will become more and more widespread. In the future there should be a standardized way of transmitting and rendering more complex appearance models including color, BRDFs and bump maps.

8.8 Examples

In this section we describe some examples for high quality 3D object acquisition. Geometry and reflection data have been acquired for a bronze bust of

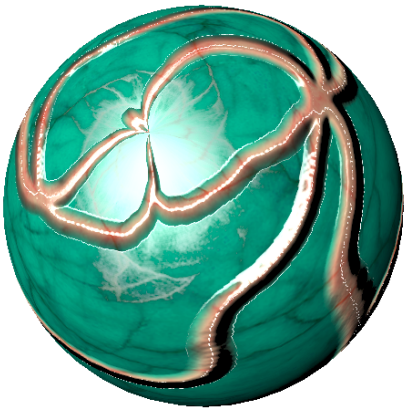


Figure 8.15: A spatially varying BRDF applied to a sphere

Max Planck, a clay bird, and a painted models of two angles. Some statistics about the meshes and the number of acquired views are listed in Table 8.1.

The model of the angels was generated by extracting an isosurface of a computer tomography scan. The 3D geometry model of the bust and the bird were acquired using a Steinbichler Tricolite structured light 3D scanner. More than 20 scans per object were necessary to cover most of the surface. After a manual approximate alignment the scans were pairwise registered against each other. Finally, an optimization procedure reduced the global error. The resulting point clouds were triangulated to form triangle meshes.

Because a structured light scanner can only acquire surface points that are visible from the camera and projector position at the same time the bust mesh contained several holes – mainly around the ears. They were filled manually. Afterwards, a filtering step was applied to improve the smoothness of the meshes. In order to accelerate further processing the triangle count of the initial models was reduced by simplifying the meshes.

The images for the textures and reflection properties were taken with a Kodak DCS 560 professional digital camera, which outputs images consisting of 6 million pixels. To acquire data for the entire surface several views with varying light source positions were captured per model (see Table 8.1). For each view around 15 photographs were necessary: two for recovering the light source po-

model	triangles	views	lumitexels	rad. samples	clusters	basis BRDFs
angels	47000	27	1606223	7.6	9	6
bird	14000	25	1917043	6.3	5	4
bust	50000	16	3627404	4.2	3	4

Table 8.1: This table lists the number of triangles of each model, the number of views we used to reconstruct the spatially varying BRDFs, the number of acquired lumitexels and the average number of radiance samples per lumitexel, the number of partitioned material clusters, and the number of basis BRDFs per cluster.

sition, one to extract the silhouette of the object for the 2D–3D registration, and the rest to provide the necessary high dynamic range.

The acquisition takes about 2.5h. The high dynamic range conversion, registration with the 3D model, and the resampling into lumitexels takes about 5h but is a completely automated task. The clustering and the final projection to recover the BRDFs takes about 1.5h[2].

Figure 8.6 shows how five successive split operations partition the lumitexels (the surface points) for the bird into its five basic materials. Only the per-cluster BRDFs determined by the clustering process are used for shading. Because of this the object looks rather flat. After performing the projection step every lumitexel is represented as a linear combination in a basis of four BRDFs, now resulting in a much more detailed and realistic appearance, see Figure 8.7.

The bust in Figure 8.16 shows another reconstructed object with very different reflection properties. The bronze look is very well captured.

A comparison between an object rendered with an acquired BRDF (using the presented method) and a photograph of the object is shown in Figure 8.17. They are very similar, but differences can be seen in highlights and in places where not enough radiance samples were captured. Capturing more samples will increase the quality. The difference in the hair region is due to missing detail in the triangle mesh. Those would be resolved by recovering the normal map for the object as described in Section 8.4.4.

Generally it can be said that for all the models only a few clusters were needed to accurately represent all the materials since the projection takes care

[2] All timings were measured on a single processor SGI Octane 300 MHz.

Figure 8.16: A bronze bust rendered with a spatially varying BRDF, which was acquired with the presented reconstruction method.

of material changes. In our experiments even Lafortune BRDFs [LFTG97] consisting of a single lobe were sufficient to form good basis for the clustering and projection.

8.9 Conclusion

We presented a framework for acquiring high quality 3D models of real world objects. The resulting models include both geometry and appearance information such as textures, normal maps or spatially varying BRDFs. Each of these is captured with a different setup. Afterwards all data is merged into a single

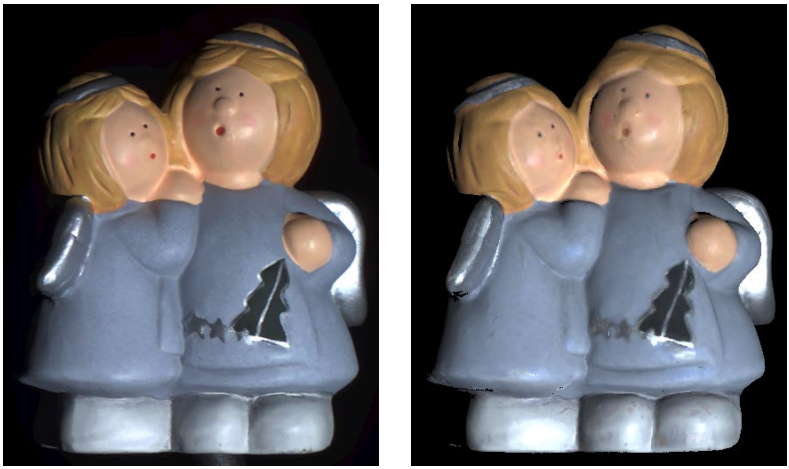


Figure 8.17: Left side: Photograph of model. Right side: Model with acquired BRDF rendered from the same view with similar lighting direction. The difference in the hair region is due to missing detail in the triangle mesh.

model which is a fairly complete representation of the geometry and surface properties of a large class of real world objects. In order to achieve the highest possible quality, state-of-the-art computer vision and computer graphics techniques need to be combined in the acquisition and model generation stage of the framework.

Given such a detailed model, many computer vision algorithms such as the reconstruction of surface normals [RTG97] or the detection of different materials can be improved or extended to other types of objects. Common assumptions about the characteristics of the object (e.g., pure diffuse reflection) are no longer necessary.

The demand for high quality 3D models will further increase in applications such as computer games, digital libraries and encyclopedias, or e-commerce applications. In order to satisfy these demands the presented methods need to be further improved with respect to acquisition speed, automation and quality. Currently, the class of materials that can be acquired and displayed are limited to isotropic materials. Future algorithms should also take effects like anisotropy and subsurface scattering into account.

8.10 Acknowledgements

We would like to thank Thomas Neumann, Kolja Kähler, Christian Rössl, Mario Botsch and the IMP Erlangen for their help in acquiring some of the presented data sets. Thanks also to Jan Kautz for providing Section 8.7, and to Philippe Bekaert for proofreading this paper.

References

[Abb95] T. M. C. Abbott. In situ CCD testing. Available at http://www.cfht.hawaii.edu/˜tmca/cookbook/top.html, 1995.

[Ban94] D. Banks. Illumination in Diverse Codimensions. In *Proceedings of SIGGRAPH 1994*, pages 327–334, July 1994.

[BG01] Samuel Boivin and André Gagalowicz. Image-based rendering of diffuse, specular and glossy surfaces from a single image. In Eugene Fiume, editor, *Proceedings of SIGGRAPH 2001*, Computer Graphics Proceedings, Annual Conference Series, pages 107–116. ACM Press / ACM SIGGRAPH, August 2001. ISBN 1-58113-292-1.

[Bli77] J. Blinn. Models of Light Reflection For Computer Synthesized Pictures. In *Proceedings SIGGRAPH*, pages 192–198, July 1977.

[Bli78] J. Blinn. Simulation of Wrinkled Surfaces. In *Proceedings of SIGGRAPH 1978*, pages 286–292, August 1978.

[BM92] P.J. Besl and N.D. McKay. A method for the registration of 3-d shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–258, 1992.

[BMR99] F. Bernardini, J. Mittleman, and H. Rushmeier. Case study: Scanning michelangelo's florentine pietà. In *Course Notes for SIGGRAPH 1999*, August 1999.

[CIE86] Colorimetry. Publication CIE No. 15.2, 1986.

[CS00] Brian Curless and Steven Seitz. 3D Photography. In *Course Notes for SIGGRAPH 2000*, July 2000.

[CT81] Robert L. Cook and Kenneth E. Torrance. A reflectance model for computer graphics. In *Computer Graphics (Proceedings of SIGGRAPH 81)*, pages 307–316, August 1981.

[DHT+00] Paul Debevec, Tim Hawkins, Chris Tchou, Haarm-Pieter Duiker, Westley Sarokin, and Mark Sagar. Acquiring the reflectance field of a human face. *Proceedings of SIGGRAPH 2000*, pages 145–156, July 2000.

[DM97] P. Debevec and J. Malik. Recovering High Dynamic Range Radiance Maps from Photographs. In *Proceedings of SIGGRAPH 97*, pages 369–378, August 1997.

[GHH01] Simon Gibson, Toby Howard, and Roger Hubbold. Flexible image-based photometric reconstruction using virtual light sources. *Computer Graphics Forum*, 20(3), 2001. ISSN 1067-7055.

[GHLS00] Michael Goesele, Wolfgang Heidrich, Hendrik P.A. Lensch, and Hans-Peter Seidel. Building a Photo Studio for Measurement Purposes. In *Proceedings of the 5th Conference on Vision, Modeling, and Visualization (VMV-00)*, November 2000.

[GHS01a] Michael Goesele, Wolfgang Heidrich, and Hans-Peter Seidel. Color calibrated high dynamic range imaging with ICC profiles. In *Proceedings of the 9th Color Imaging Conference*, Scottsdale, USA, 2001.

[GHS01b] Michael Goesele, Wolfgang Heidrich, and Hans-Peter Seidel. Entropy-based dark frame subtraction. In *Proceedings of PICS 2001: Image Processing, Image Quality, Image Capture, Systems Conference*, Montreal, Canada, April 2001. The Society for Imaging Science and Technology (IS&T).

[HDKS00] W. Heidrich, K. Daubert, J. Kautz, and H.-P. Seidel. Illuminating micro geometry based on precomputed visibility. In *Proc. of SIGGRAPH 2000*, pages 455–464, July 2000.

[HH87] Richard S. Hunter and Richard W. Harold. *The measurement of appearance*. Wiley, 2. ed., 5. print. edition, 1987.

[HS93] P. Haeberli and M. Segal. Texture Mapping As A Fundamental Drawing Primitive. In *Fourth Eurographics Workshop on Rendering*, pages 259–266, June 1993.

[HS97] J. Heikkila and O. Silven. A Four-Step Camera Calibration Procedure With Implicit Image Correction. In *CVPR97*, 1997.

[HS99] Wolfgang Heidrich and Hans-Peter Seidel. Realistic, hardware-accelerated shading and lighting. In *Proceedings of SIGGRAPH 99*, Computer Graphics Proceedings, Annual Conference Series, pages 171–178, August 1999.

[Hun95] Robert W. G. Hunt. *The reproduction of colour*. Fountain Press, 5. ed. edition, 1995.

[ICC98] Specification ICC.1:1998-09, File Format for Color Profiles. available from http://www.color.org, 1998.

[JMLH01] Henrik Wann Jensen, Stephen R. Marschner, Marc Levoy, and Pat Hanrahan. A practical model for subsurface light transport. In Eugene Fiume, editor, *Proceedings of SIGGRAPH 2001*, Computer Graphics Proceedings, Annual Conference Series, pages 511–518. ACM Press / ACM SIGGRAPH, August 2001. ISBN 1-58113-292-1.

[JS00] J.Kautz and H.-P. Seidel. Towards interactive bump mapping with anisotropic shift-variant brdfs. In *Proceedings of the Eurographics/SIGGRAPH Workshop on Graphics Hardware 2000*, pages 51–58, August 2000.

[KBB+00] Leif P. Kobbelt, Stephan Bischoff, Mario Botsch, Kolja Kähler, Christian Rössl, Robert Schneider, and Jens Vorsatz. Geometric modeling based on polygonal meshes. Technical Report MPI-I-2000-4-002, Max-Planck-Institut für Informatik, July 2000.

[KCVS98] Leif Kobbelt, Swen Campagna, Jens Vorsatz, and Hans-Peter Seidel. Interactive multi-resolution modeling on arbitrary meshes. *Proceedings of SIGGRAPH 98*, pages 105–114, July 1998.

[Kil00] M. Kilgard. *A Practical and Robust Bump-mapping Technique for Today's GPUs*. NVIDIA Corporation, April 2000. Available from http://www.nvidia.com.

[KM99] J. Kautz and M. McCool. Interactive Rendering with Arbitrary BRDFs using Separable Approximations. In *10th Eurographics Rendering Workshop 1999*, pages 281–292, June 1999.

[Kob96] Leif Kobbelt. Discrete fairing. In *Proceedings of the Seventh IMA Conference on the Mathematics of Surfaces*, pages 101–131, 1996.

[LFTG97] E. Lafortune, S.-C. Foo, K. Torrance, and D. Greenberg. Non-Linear Approximation of Reflectance Functions. In *Proceedings of SIGGRAPH 1997*, pages 117–126, August 1997.

[LH96] Marc Levoy and Pat Hanrahan. Light field rendering. In *Computer Graphics (SIGGRAPH '96 Proceedings)*, pages 31–42, August 1996.

[LHS00] Hendrik P. A. Lensch, Wolfgang Heidrich, and Hans-Peter Seidel. Automated texture registration and stitching for real world models. In *Pacific Graphics '00*, pages 317–326, October 2000.

[LKG+01] Hendrik Lensch, Jan Kautz, Michael Goesele, Wolfgang Heidrich, and Hans-Peter Seidel. Image-based reconstruction of spatially varying materials. In Steven Gortler and Karol Myszkowski, editors, *Proceedings of the 12th Eurographics Workshop on Rendering*, pages 104–115, London, Great Britain, 2001. Springer.

[LPC+00] Marc Levoy, Kari Pulli, Brian Curless, Szymon Rusinkiewicz, David Koller, Lucas Pereira, Matt Ginzton, Sean Anderson, James Davis, Jeremy, Ginsberg, Jonathan Shade, and Duane Fulk. The digital michelangelo project: 3D scanning of large statues. In *Proceedings of SIGGRAPH 2000*, pages 131–144, July 2000.

[Lut27] R. Luther. Aus dem Gebiet der Farbreizmetrik. *Zeitschrift für technische Physik*, 8(12):540–558, 1927.

[Mad93] Brian C. Madden. Extended Intensity Range Imaging. Technical report, University of Pennsylvania, GRASP Laboratory, 1993.

[Mar98] S. R. Marschner. *Inverse rendering for computer graphics*. PhD thesis, Cornell University, 1998.

[MGR00] S. R. Marschner, B. Guenter, and S. Raghupathy. Modeling and rendering for realistic facial animation. In *Eurographics Rendering Workshop 2000*, pages 231–242, June 2000.

[MK99] Kenji Matsushita and Toyohisa Kaneko. Efficient and handy texture mapping on 3d surfaces. *Computer Graphics Forum*, 18(3):349–358, September 1999.

[MWL+99] S. Marschner, S. Westin, E. Lafortune, K. Torrance, and D. Greenberg. Image-based BRDF Measurement Including Human Skin. In *10th Eurographics Workshop on Rendering*, pages 131–144, June 1999.

[NK99] Peter J. Neugebauer and Konrad Klein. Texturing 3d models of real world objects from multiple unregistered photographic views. *Computer Graphics Forum*, 18(3):245–256, September 1999.

[NVI99] NVIDIA Corporation. *NVIDIA OpenGL Extension Specifications*, October 1999. Available from http://www.nvidia.com.

[PDH+97] K. Pulli, T. Duchamp, H. Hoppe, J. McDonald, L. Shapiro, and W. Stuetzle. Robust meshes from multiple range maps. In *Proceedings of IEEE International Conference on Recent Advances in 3-D Digital Imaging and Modeling*, 1997.

[PvdDJ+01] Dinesh K. Pai, Kees van den Doel, Doug L. James, Jochen Lang, John E. Lloyd, Joshua L. Richmond, and Som H. Yau. Scanning physical interaction behavior of 3d objects. In Eugene Fiume, editor, *Proceedings of SIGGRAPH 2001*, Computer Graphics Proceedings, Annual Conference Series, pages 87–96. ACM Press / ACM SIGGRAPH, August 2001. ISBN 1-58113-292-1.

[RBMT98] Holly Rushmeier, Fausto Bernardini, Joshua Mittleman, and Gabriel Taubin. Acquiring input for rendering at appropriate levels of detail: Digitizing a pietà. *Eurographics Rendering Workshop 1998*, pages 81–92, June 1998.

[RBS99] Mark A. Robertson, Sean Borman, and Robert L. Stevenson. Dynamic Range Improvement Through Multiple Exposures. In *Proceedings of the 1999 International Conference on Image Processing (ICIP-99)*, pages 159–163. IEEE, oct. 1999.

[RCM99] C. Rocchini, P. Cignoni, and C. Montani. Multiple textures stitching and blending on 3D objects. In *Eurographics Rendering Workshop 1999*. Eurographics, June 1999.

[RH01] Ravi Ramamoorthi and Pat Hanrahan. A signal-processing framework for inverse rendering. In Eugene Fiume, editor, *Proceedings of SIGGRAPH 2001*, Computer Graphics Proceedings, Annual Conference Series, pages 117–128. ACM Press / ACM SIGGRAPH, August 2001. ISBN 1-58113-292-1.

[RTG97] Holly Rushmeier, Gabriel Taubin, and André Guéziec. Applying shape from lighting variation to bump map capture. *Eurographics Rendering Workshop 1997*, pages 35–44, June 1997.

[SHR+99] Hartmut Schirmacher, Wolfgang Heidrich, Martin Rubick, Detlef Schiron, and Hans-Peter Seidel. Image-based BRDF reconstruction. In Bernd Girod, Heinrich Niemann, and Hans-Peter Seidel, editors, *Proceedings of the 4th Conference on Vision, Modeling, and Visualization (VMV-99)*, pages 285–292, nov 1999.

[Stu99] Wolfgang Stuerzlinger. Imaging all visible surfaces. In *Graphics Interface '99*, pages 115–122, June 1999.

[Tau95] G. Taubin. A signal processing approach to fair surface design. *Proceedings of SIGGRAPH 1995*, pages 351–358, 1995.

[TS67] K. E. Torrance and E. M. Sparrow. Theory for off-specular reflection from roughened surfaces. *Journal of the Optical Society of America*, 57(9):1105–1114, September 1967.

[Tsa87] R. Tsai. A versatile camera calibration technique for high accuracy 3d machine vision metrology using off-the-shelf tv cameras and lenses. *IEEE Journal of Robotics and Automation*, 3(4), August 1987.

[Wal] Dawn Wallner. Building ICC profiles – the mechanics and engineering. available at http://www.color.org/iccprofiles.html.

[War92] G. Ward. Measuring and modeling anisotropic reflection. In *Proceedings of SIGGRAPH 1992*, pages 265–272, July 1992.

[WB01] Don Williams and Peter D. Burns. Diagnostics for digital capture using MTF. In *Proceedings of PICS 2001: Image Processing, Image Quality, Image Capture, Systems Conference*, pages 227–232, Montreal, Canada, April 2001. The Society for Imaging Science and Technology (IS&T).

[YDMH99] Yizhou Yu, Paul Debevec, Jitendra Malik, and Tim Hawkins. Inverse global illumination: Recovering reflectance models of real scenes from photographs. *Proceedings of SIGGRAPH 99*, pages 215–224, August 1999.

[Zha99] Zhengyou Zhang. A Flexible New Technique for Camera Calibration. Technical Report MSR-TR-98-71, Microsoft Research, 1999. Updated version of March 25, 1999.

■

Chapter 9

Fast and Accurate Integration of Vector Fields in Unstructured Grids

Frank Reck, Günther Greiner[*]

Particle tracing is a widely used method to analyze and interpret results of a flow simulation. In addition, it is a preliminary step for more advanced techniques of flow visualization, e.g. line integral convolution.

For interactive exploration of large data sets, a very efficient and reliable particle tracing method is needed. For data on unstructured grids and data sizes, as they appear in the simulation of wind channel experiments (e.g. automative industry) flight simulation (e.g. aircraft industry), the traditional approach, based on numerical integration methods of ordinary differential equations does not allow sufficiently accurate path calculation at the speed required for interactive use.

Traditional integration techniques require small stepsizes in order to achieve sufficient accuracy. This results in many cell search operations,

[*] Universität Erlangen-Nürnberg, Lehrstuhl für Graphische Datenverarbeitung, Erlangen, Germany. E-Mail: fkreck@immd9.informatik.uni-erlangen.de,greiner@informatik.uni-erlangen.de

which means especially for unstructured grids a bottleneck of the whole procedure.

In [NJ99] Nielson and Jung have proposed a new method, called further on *locally exact method,* which gives sufficient accuracy and traverses each cell in a single step. In this note we extend the approach of Nielson and Jung in such a way that it can be performed in real time. This will be achieved by a sophisticated preprocessing, which allows a fast execution of the locally exact integration method, interactive particle tracing in large data sets can be done. We describe the procedure, compare it with Nielson's original approach, as well as with the traditional method based on numerical integration and report on the performance of the different methods.

9.1 Introduction

In many scientific areas as well as in technical applications, Computational Fluid Dynamics (CFD) is of central importance. The result of a fluid dynamics simulation are data sets which usually describe the flow behavior in a 3D-environment. To understand the characteristics of the simulated process, a meaningful visualization of the data is necessary. The raw data are defined on a discrete structure, a grid, which s build up out of many cells. In the nodes or vertices of the grid the velocity (a vector field) and other simulation data, e.g. pressure, density, energy (scalar values) are stored.

One of the most common methods of acquiring knowledge of flows is the use of particle tracing [SvWHP94, USM96, NJS+97, NJ99, KL95, Frü94]. It is also the basis of many other visualization techniques such as Streaklines, Streakribbons, Time-Surfaces or Line Integral Convolution (LIC) (see [CL93]). This visualization method shows the trajectory of one mass-less particle in the flow. The trajectory is obtained by the integration of the ordinary differential equation corresponding to the vector field.

The integration is usually done numerically. In this paper we describe another approach for vector fields defined on a tetrahedral mesh, i.e. an unstructured grid in which all cells are tetrahedrons. It is a modification of the *exact integration method* introduced in [NJ99]. This method traverses every cell in a single step. Our modification comprises a sophisticated pre-processing of the data, which results in a classification of the cells and provides for each cell sufficient information to perform the exact integration very fast. We determine the

exact exit points of the particle for every cell that is traversed by the particle. The particle trace is the polyline of the particle or any interpolating curve. In contrast to all the numerical integration schemes, this approach guarantees that the local errors will not accumulate. Hence it is globally rather accurate.

The paper is organized as follows: In the next section we review the existing methods for particle tracing in unstructured grids. At first we briefly describe the method based on numerical integration of ordinary differential equations and then in a more detailed way the exact method due to Nielson et al [NJ99]. In Section 3 we outline our modification of the exact method, in particular, we explain the data structures used, describe the preprocessing step in detail and give pseudocode for the algorithm. Section 4 is the result section. We compare the accuracy of the different methods as well as the time performance.

9.2 Particle Tracing in Tetrahedral Grids

Particle tracing in a vector field $\vec{\mathbf{v}}(x)$ (considered as the velocity field) amounts solving the initial value problem for an ordinary differential equation (ODE)

$$\frac{dx(t)}{dt} = \vec{\mathbf{v}}(x(t)) \qquad x(t_0) = x_0. \tag{1}$$

Whereas $x(t)$ represents the position of the particle at the time t, starting at time t_0 at position x_0.

The vector field $\vec{\mathbf{v}}$ is usually not given as an analytic function, only at certain samples, usually in the vertices of a grid. In this paper we will only discuss unstructured grids, more precisely, in Euclidean 3D-space we consider tetrahedral meshes or, when we consider the 2D-analogue, we have got triangular meshes in the plane. In most cases these data result from a numerical simulation and the mesh is the grid from the numerical simulation (e.g. finite volumes or finite elements).

9.2.1 Numerical Integration Methods

The standard way to solve the ODE is numerical integration. This topic is well investigated and there are many different methods [TGE97] [DH96]: single-step or multi-step methods, explicit or implicit methods, constant or

variable step-size. We only outline the steps needed to apply such an integration scheme. The basic steps are:

```
// point location:
find cell containing initial position

while particle in grid
  // interpolation:
  determine velocity at current position
  // integration:
  calculate new position:
  // point location
  find cell containing new position
endwhile
```

By this procedure we get a sequence of sample points $x(t_n)$ of the path line, approximating the position of the particle at time t_n, i.e. $x_n\ x(t_n)$. The polyline connecting these points will then be an approximation to the path line of the particle.

The step "calculation of the new position" is the integration step, the simplest method is the Euler's method

$$x_{n+1} = x_n + (t_{n+1} - t_n) \cdot \vec{\mathbf{v}}(x_n)$$

more accurate are Runge-Kutta methods ($\mathbf{RK}_p$) of order p, e.g. $\mathbf{RK}_2$ also known as Heun's method:

$$\begin{aligned} k_1 &= x_n + (t_{n+1} - t_n) \cdot \vec{\mathbf{v}}(x_n) \\ x_{n+1} &= x_n + (t_{n+1} - t_n) \cdot 1/2(\vec{\mathbf{v}}(x_n) + \vec{\mathbf{v}}(k_1)) \end{aligned}$$

$\mathbf{RK}_4$ which is the classical Runge-Kutta-Method.

$$\begin{aligned} k_1 &= x_n + (t_{n+1} - t_n)1/2 \cdot (t_{n+1} - t_n) \cdot \vec{\mathbf{v}}(x_n) \\ k_2 &= k_1 + (t_{n+1} - t_n)1/2 \cdot (t_{n+1} - t_n) \cdot \vec{\mathbf{v}}(k_1) \\ k_3 &= x_n + (t_{n+1} - t_n)(t_{n+1} - t_n) \cdot \vec{\mathbf{v}}(k_2) \\ x_{n+1} &= x_n + (t_{n+1} - t_n) \cdot 1/6(\vec{\mathbf{v}}(x_n) + 2 \cdot \vec{\mathbf{v}}(k_1) \\ &\quad +2 \cdot \vec{\mathbf{v}}(k_2) + \vec{\mathbf{v}}(k_3)) \end{aligned}$$

The order p of a Runge Kutta method denotes the order of the local approximation error. For $p = 1$ it is identical with Eulers's method, if it is 2, with

Heun's method. Higher order means greater exactness, but also more time for the calculation. Another factor which influences the correctness and time behavior is the step size. The shorter the step size in a particle trajectory calculation, the higher is the number of steps to be executed and the longer is the duration of the process. At the same time error will be smaller. An approach to overcome this is the use of adaptive methods, which influence their step size according to a user given error. These methods aren't deprived of problems: Who can the error be measured? What is a large or a small error for a specified data set? What is the maximum still meaningful step size?

Another problem and time consuming factor in this approach is the point location, which is necessary after each integration step. This can be done easily in rectilinear grids, but in irregular grids a time consuming cell search is needed for every evaluation.

The method described in the next section does not suffer from these problems.

9.2.2 Local Exact Integration

This method has been introduced by Nielson and Jung [NJ99]. The basic observation is the following. We know the vector field only in the vertices of the tetrahedral mesh. For the values in between all we have to do is to interpolate. The simplest and most efficient way is linear interpolation. That is we consider the vector field as piecewise linear mapping. Since for linear mappings $\vec{\mathbf{v}}$ the ODE (1) can be solved analytically, in each tetrahedral cell we can determine the exact path-line, for the linearly interpolated vector field.

In fact, for the "linear vector field"

$$\vec{\mathbf{v}}(x) = Ax + b$$

with a square matrix A and a translation vector b the analytic solution of 1 is

$$x(t) = e^{(t-t_0)A}x_0 + (e^{(t-t_0)A} - Id)A^{-1}b \tag{2}$$

where the exponentials of the matrices are given by the usual series expansion, e.g.,

$$e^{(t-t_0)A} = \sum_{k=0}^{\infty} \frac{(t-t_0)^k}{k!}A^k \tag{3}$$

This can be calculated much faster can this be if the "normal form" of the matrices are known, e.g. when A has three (different) real eigenvalues λ_1, λ_2, λ_3,

then, with the transformation matrix formed by the eigenvectors $\vec{b_1},\ \vec{b_2},\ \vec{b_3}$, we have

$$A = \begin{pmatrix} | & | & | \\ \vec{b_1} & \vec{b_2} & \vec{b_3} \\ | & | & | \end{pmatrix} \begin{pmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3 \end{pmatrix} \begin{pmatrix} | & | & | \\ \vec{b_1} & \vec{b_2} & \vec{b_3} \\ | & | & | \end{pmatrix}^{-1}. \tag{4}$$

In case of complex eigenvalues, this representation is also valid. However, since calculations with real numbers are simpler we rather consider a "real normal form" in this case. Assuming that the real matrix A has complex eigenvalues, then these appear as a conjugate pair: $\lambda_1 = \sigma + i\tau,\ \lambda_2 = \sigma - i\tau,\ (\tau > 0)$ and a real eigenvalue λ_3. The transformation to the "real normal form" of A is given by

$$A = \begin{pmatrix} | & | & | \\ \vec{b_1} & \vec{b_2} & \vec{b_3} \\ | & | & | \end{pmatrix} \begin{pmatrix} \sigma & \tau & 0 \\ -\tau & \sigma & 0 \\ 0 & 0 & \lambda_3 \end{pmatrix} \begin{pmatrix} | & | & | \\ \vec{b_1} & \vec{b_2} & \vec{b_3} \\ | & | & | \end{pmatrix}^{-1} \tag{5}$$

In this case $\vec{b}_1$ and $\vec{b}_2$ are the real and the imaginary part of the eigenvector corresponding to $\lambda_1 = \sigma + i\tau$, and $\vec{b_3}$ is the eigenvector corresponding to λ_3.

Using the normal forms (4) and (5) respectively, the expontionals of A can be determined as follows,

$$e^{\bar{t}A} = S \begin{pmatrix} e^{\lambda_1(\bar{t})} & 0 & 0 \\ 0 & e^{\lambda_2(\bar{t})} & 0 \\ 0 & 0 & e^{\lambda_3(\bar{t})} \end{pmatrix} S^{-1}$$

and

$$e^{\bar{t}A} = S \begin{pmatrix} e^{\sigma\bar{t}}\cos(\tau\tilde{t}) & e^{\sigma\bar{t}}\sin(\tau\tilde{t}) & 0 \\ -e^{\sigma\bar{t}}\sin(\tau\tilde{t}) & e^{\sigma\bar{t}}\cos(\tau\tilde{t}) & 0 \\ 0 & 0 & e^{\lambda_3(\bar{t})} \end{pmatrix} S^{-1}$$

whereas $\tilde{t}$ stands for $t - t_0$. Then the analytic solution from 2 is in the real case

$$x(t) = S\Lambda(t)S^{-1}x_0 + S\Delta(t)S^{-1}b \tag{6}$$

with the diagonal matrices $\Lambda(t)$ and $\Delta(t)$ given in the real case by

$$\Lambda(t) = \begin{pmatrix} e^{\lambda_1 \bar{t}} & 0 & 0 \\ 0 & e^{\lambda_2 \bar{t}} & 0 \\ 0 & 0 & e^{\lambda_3 \bar{t}} \end{pmatrix} \tag{7}$$

$$\Delta(t) = \begin{pmatrix} \frac{e^{\lambda_1 \bar{t}}-1}{\lambda_1} & 0 & 0 \\ 0 & \frac{e^{\lambda_2 \bar{t}}-1}{\lambda_2} & 0 \\ 0 & 0 & \frac{e^{\lambda_3 \bar{t}}-1}{\lambda_3} \end{pmatrix} \tag{8}$$

and in the complex case by a slightly more complicated form.

So far we have tacitly assumed that all eigenvalues are different from each other and different from zero (otherwise A^{-1} does not exist). Concerning the first restriction, we point out that this is the generic case, i.e. choosing randomly any tetrahedron and assigning randomly vectors to its vertices, the probability of obtaining identically eigenvalues will be zero. And our examples coming from a simulation also show, that this situation rarely occurs. Actually, the second restriction (non-zero eigenvalues) was only needed to derive the result more easily. The final formulas make sense and are also correct for zero eigenvalues. In this case in $\Delta(t)$ the term $\frac{e^{\lambda(t-t_0)}-1}{\lambda}$ has to be replaced by $t - t_0$.

Due to these remarks, in contrast to [NJ99], we do not pay much attention to all the other possible cases: zero eigenvalues, identical eigenvalues, multiple eigenvectors, and identical eigenvalues single eigenvector. If one of these cases occurs, we cross the cell by standard numerical integration methods (details see below).

9.2.3 Comparison

The differences of the two methods can be briefly summarized:

- The locally exact method gives the exact solution, provided that (1) the numerical errors are neglected, and (2) one assumes that the underlying vector field is obtained by linear interpolation.
- for the LEM no point location is necessary. In fact, knowing the neighbors of a tetrahedron (one for each face), the next tetrahedron to be processed is given by the location of the exit point.

- Using the LEM one does not have to specify the stepsize, nor does one have to take care of (adaptively) modifying it. In fact, somehow the adaptation is build in. For simulation, typically in (numerically) critical areas, or areas of special importance one has a fine mesh of tetrahedrons. In these areas, the path segments produced by the LEM are then small as well.

9.3 Modification of Nielson's Approach

At first we give an overview of our modification of the method due to Nielson and Jung. We continue to denote this method LEM.

9.3.1 Overview of the local exact method

We classify the tetrahedron as normal cells, parallel cells and extraordinary cells. Parallel cells are cells whose vector field does not change its direction (all four velocity vectors point in the same direction), for the linearly interpolated.

A cell is normal when the linearl when the linear part of the (the linearly interpolated) vector field has three different eigenvalues, and in addition the critical point of the vector field is located outside the tetrahedron.

All the remaining cells are declared to be extraordinary cells.

When the path line enters a parallel cell, the intersection of the ray starting at the entering point in direction of the vector field with the tetrahedron is computed. This is the exit point and the next entrance point for the neighboring cell.

When the path line enters a normal cell we determine the intersection point of the exact solution (given by formula (2)) and the exit face of the tetrahedron. This is the exit point and the next entrance point for the neighboring cell.

When entering an extraordinary cell we switch to a Runge-Kutta integration scheme, till the cell search routine detects a new cell. Finally we display the polyline connecting all entrance/exit points.

9.3.2 Data structure

The basic data structure of simulation data over an unstructured grid consists of a list of all vertices of the space partition and a list of all cells (tetrahedron in our case). One entry in the vertex list contains:

V1) $x-$, $y-$ and $z-$ coordinate of the vertex
V2) the simulation data at this point, most important for our consideration the vector field $\vec{\mathbf{v}}$ ($x-$, $y-$ and $z-$ component) describing the flow and possibly additional simulation data, e.g. pressure, internal energy, temperature.

An entry in the cell list contains
C1) Four references to the vertex list, representing the vertices of the tetrahedron. The sequence determines the internal enumeration V_0, V_1, V_2, V_3 and it is always chosen such that the four vertices are positively orientated.
C2) Four references to the neighboring cells (tetrahedra): The first one, T_0 is opposite to the vertex V_0, T_1 to V_1 and so on.
C3) flag indicating the type of the cell.
C4) information concerning the internal enumeration of the neighboring cells.
C5) Information of the linearly interpolated vector field $\vec{\mathbf{v}}(x) = Ax + b$:
e.g. the translation vector b (or $S^{-1}b$) and eigenvalues and eigenvectors of A.

The flag for the type of the cell indicates whether we have a parallel cell, a normal cell with real eigenvalues, a normal cell with complex eigenvalues or an extraordinary cell. This information is needed because any cell type will be treated differently when the path line is calculated.

The data described in C4) are needed when exiting a cell and entering the neighboring cell. Then it is advantageous to know which is the entrance face, and which is the correspondence of the common vertices: Since all tetrahedra are required to be positively orientated, this can be coded in 4 bits (for one neighboring tetrahedron). In fact, the internal index of the opposite vertex has to be known (2 bits) and the internal index of one additional vertex (2 bits), e.g. the one whose internal index is obtained by bit flip must be known.

T_0	vertex opposite to V_0	(2 bits)
	vertex coincide with V_3	(2 bits)
T_1	vertex opposite to V_1	(2 bits)
	vertex coincide with V_2	(2 bits)
T_2	vertex opposite to V_2	(2 bits)
	vertex coincide with V_1	(2 bits)
T_3	vertex opposite to V_3	(2 bits)
	vertex coincide with V_0	(2 bits)

9.3.3 Preprocessing

In the preprocessing step, we first determine the type of the cell (parallel, normal or extraordinary); for the normal cells an eigenvalue/-vector analysis must be performed in order to supply the information listed in C5 of Sec. 9.3.2

To determine the cell type we first check the velocity vectors $\vec{\mathbf{v_0}}$, $\vec{\mathbf{v_1}}$, $\vec{\mathbf{v_2}}$ and $\vec{\mathbf{v_3}}$. If they all point in the same direction, the cell gets the label *parallel* and we store this direction. In the implementation we specify a threshold ϵ_P which somehow measures the deviation of the directions given by the vector field in the four vertices. A detailed description will be given below.

For the non-parallel cells we determine the linear interpolation $\vec{\mathbf{v}}(x) = Ax + b$ of the vector field. The 3×3 matrix A and the translation vector b are uniquely determined by the four vector equations $AV_i + b = \vec{\mathbf{v}}_i$, $(i = 0, 1, 2, 3)$.

Now we perform an eigenvalue/-vector analysis of A. First the eigenvalues have to be found. This is done with the formula of Cardano, which allows the analytical calculation of the zeros of the third order polynomial. First we check whether the three eigenvalues are different (i.e. differ at least by some threshold), all real or whether two of them are complex conjugates. Then we label the cells to be either *normal & real* or *normal & complex* and otherwise *extraordinary*. As already pointed out the latter case rarely occurs. Finally we check whether the critical point x_{crit} of the linearly interpolated vector field lies in the tetrahedral cell: ($Ax_{crit} + \vec{b} = 0$ or $x_{crit} = -A^{-1}\vec{b}$). If so, we also label this cell to be extraordinary.

If the cell is *normal & real* we determine the eigenvectors $\vec{b}_1$, $\vec{b}_2$ and $\vec{b}_3$ corresponding to the three eigenvalues λ_1, λ_2 and λ_3. By changing the length of the eigenvectors we can achieve that $\det(\vec{b}_1, \vec{b}_2, \vec{b}_3) = 1$. With the eigenvalues the transformation matrix $S = (\vec{b}_1, \vec{b}_2, \vec{b}_3)$ and its inverse $S^{-1} = (\vec{b}_2 \times \vec{b}_3, \vec{b}_3 \times \vec{b}_1, \vec{b}_1 \times \vec{b}_2)^t$ can be determined. Finally we store $S^{-1}\vec{b}$.

If the cell is normal & complex we have eigenvectors $\sigma + i\tau$, $\sigma - i\tau$ and λ_3, with σ, τ, λ_3 being real numbers and $\tau > 0$. b_1 and b_2 will be the real and the complex part of the complex eigenvector corresponding to $\sigma + i\tau$. b_3 is as before the eigenvector to λ_3. As before, by appropriate scaling we can achieve that $\det(\vec{b}_1, \vec{b}_2, \vec{b}_3) = 1$.

$$A = \begin{pmatrix} | & | & | \\ \vec{b}_1 & \vec{b}_2 & \vec{b}_3 \\ | & | & | \end{pmatrix} \begin{pmatrix} \sigma & \tau & 0 \\ -\tau & \sigma & 0 \\ 0 & 0 & \lambda_3 \end{pmatrix} \begin{pmatrix} - \vec{b}_2 \times \vec{b}_3 - \\ - \vec{b}_3 \times \vec{b}_1 - \\ - \vec{b}_1 \times \vec{b}_2 - \end{pmatrix} \tag{9}$$

9.3.4 Calculation of path lines in world coordinates

Given an initial point in a tetrahedron, we determine successively the exit points of the exact path line. The exit point is the entrance point to neighboring tetrahedron.

To start the process, we first have to identify the cell containing the initial point by a standard cell searching procedure. Further on no cell search operation are necessary.

For the cell we know the eigenvalues and the transformation matrix $S = (\vec{b}_1, \vec{b}_2, \vec{b}_3)$, $S^{-1}b$ and can thus compute the inverse S^{-1}. Then the pathline $x(t)$ is given by Equ. (6). We determine the intersection of $x(t)$ with the four planes spanned by the faces of the tetrahedron and use as exit point the one which has the smallest positive t-value.

The intersection point of $x(t)$ with the plane corresponding to face $F_0 = \Delta(V_1, V_2, V_3)$ is obtained by solving $\det(x(t) - V_3, V_1 - V_3, V_2 - V_3)$ or

$$\langle x(t)|(V_1 - V_3) \times (V_2 - V_3)\rangle = \langle V_3|(V_1 - V_3) \times (V_2 - V_3)\rangle$$

and using (6)

$$\begin{aligned} & \langle \Lambda(t)S^{-1}x_0|S^{transp}((V_1 - V_3) \times (V_2 - V_3))\rangle \\ & + \langle \Delta(t)S^{-1}b|S^{transp}((V_1 - V_3) \times (V_2 - V_3))\rangle \\ = \quad & \langle V_3|(V_1 - V_3) \times (V_2 - V_3)\rangle \end{aligned}$$

We use the Newton method to solve this equation, starting with the initial value $t_0 = 0$

`Inizalization:`
$A = S^{-1}x_0$
$B = S^{transp}((V_1 - V_3) \times (V_2 - V_3))$
($C = S^{-1}b$ is precalculated)
$a = \langle V_3|(V_1 - V_3) \times (V_2 - V_3)\rangle$
$t = 0$
$f = A_xB_x + A_yB_y + A_zB_z$
$f' = \lambda_1A_xB_x + \lambda_2A_yB_y + \lambda_3A_zB_z + C_xB_x + C_yB_y + C_zB_z$

`do until convergence`

$$t = t + \frac{a - f}{f'} \tag{10}$$

$f = e^{\lambda_1 t} A_x B_x + e^{\lambda_2 t} A_y B_y + e^{\lambda_1 t} A_z B_z + \frac{e^{\lambda_1 t}-1}{\lambda_1} C_x B_x + \frac{e^{\lambda_2 t}-1}{\lambda_2} C_y B_y + \frac{e^{\lambda_3 t}-1}{\lambda_3} C_z B_z$
$f' = \lambda_1 e^{\lambda_1 t} A_x B_x + \lambda_2 e^{\lambda_2 t} A_y B_y + \lambda_3 e^{\lambda_1 t} A_z B_z + e^{\lambda_1 t} C_x B_x + e^{\lambda_2 t} C_y B_y + e^{\lambda_3 t} C_z B_z$
enddo

These are the formulas for the case of real eigenvalues.

9.4 Results

The program was tested on an Octane from SGI, with a single MIPS R12000 300 MHz processor.

We tested our modified method on three different data sets; all of them are results of air-flow simulations. They were obtained by simulating the flow around a sphere (8193 tetrahedrons and 1550 points), a car model (457874 tetrahedrons and 89881 points) (see Figure 9.2) and the model of a space shuttle (1058775 tetrahedrons and 190584 points) (see Figure 9.1) respectively.

9.4.1 Regular vs. non-regular cells

As explained above, in the pre-processing we label the cells as regular, parallel and extraordinary. Tracing a particle through a parallel cell is simple and fast, even faster than for regular cells. Extraordinary cells are traversed by traditional numerical integration (Runge Kutta method) until the necessary cell searching routine indicates that a new cell has been reached. This is rather time consuming. However, this case hardly ever occurs. In fact, Table 9.1 shows, that less than 0.5 % of the cells are of the extraordinary type.

Table 9.1: Number of extraordinary cells.

# of cells	sphere data	car data	shuttle data
total	8,193	457,874	1,058,785
extraord.	15	1,247	3,232

As to the number of parallel cells (which can be traversed very fast) the actual condition under which the cells are singled out are decisive. One way

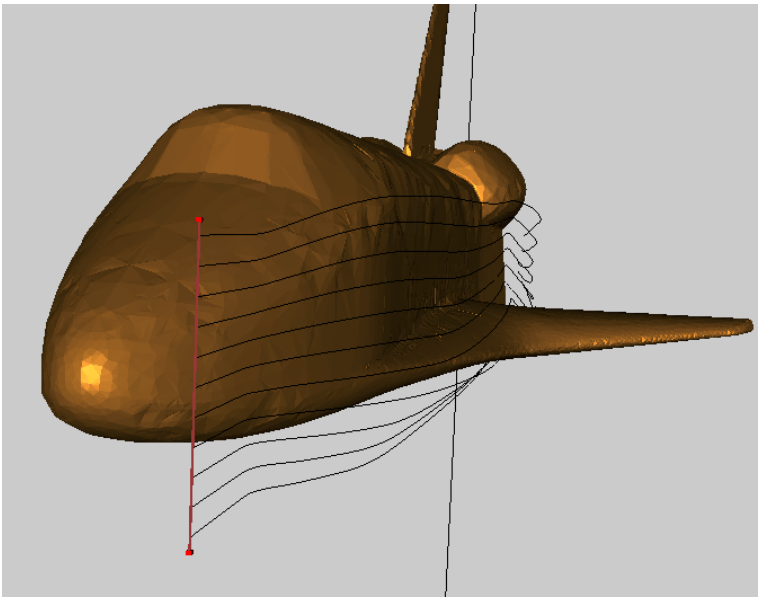


Figure 9.1: Space Shuttle model.

would be to restrict the maximal angular deviation of the vectors at the four vertices. We have chosen a different approach. A cell is parallel provided that two of the three eigenvalues are zero. In the implementation we specify a threshold ϵ_P and label a cell as parallel if the absolute value of two eigenvalues is smaller than ϵ_P. Of course, the number of parallel cells then depends on the threshold ϵ_P. In Table 9.2 we list the numbers of parallel cells for different threshold values.

The threshold value we used in our method is 0.0001, the one marked in Table 9.2 by the asterisk. Thus approximately 10 % of the cells were parallel. By using $\epsilon_P = 0.001$ one can speed up the method without loosing much accuracy.

9.4.2 Precision

The precision of this method was already investigated by Nielson in [NJ99]. Screenshot 9.3 and 9.4 show a sphere within a circular flow. The correct parti-

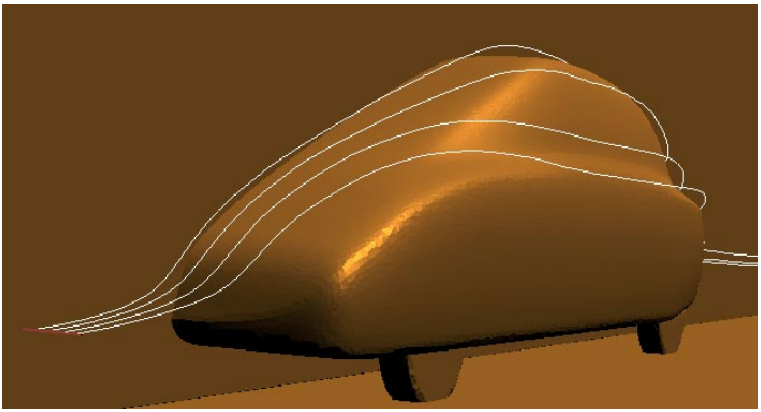


Figure 9.2: Car model in the wind tunnel.

Table 9.2: Influence of the threshold ϵ_P to the number of cells who are parallel.

Number of parallel cells for different ϵ_P			
threshold ϵ_P	car data	shuttle data	
0.0000001	43,886	2,071	
0.000001	44,074	18,910	
0.00001	45,309	33,773	
0.0001	52,670	86,232	*
0.001	144,288	366,028	
0.01	446,722	851,158	
0.1	457,869	1,052,720	

cle path would stay on a circle. The Runge-Kutta method of order two fails to hold the distance, whereas the local exact method finds the correct path.

Our modification allows a scaling of the precision, the number of Newton iterations has got an influence on the accuracy.

More iterations will achieve a more accurate solution. Table 9.3 shows how many iterations are required to reach a given accuracy. If the increment of the parameter t in (10) is below a given threshold ϵ_N then the iteration stops. Since

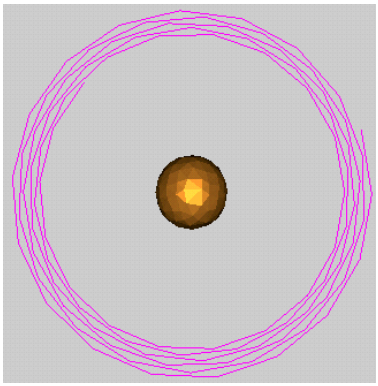


Figure 9.3: The Runge-Kutta method of order 2 does not manage to stay on the circle.

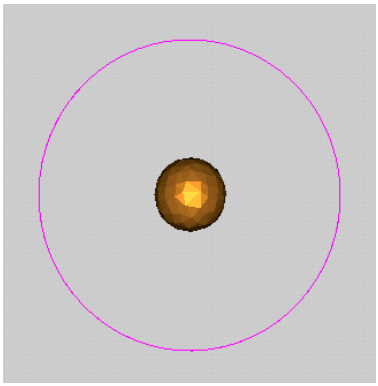


Figure 9.4: This trajectory was computed with the locally exact method.

the converge order of the Newton method is quadratic, the costs for a much better result are comparatively small. The visual effect of such an improvement, e.g. the change of the threshold from 0.1 to 0.01 is not observable. Experiments have shown that it is normally sufficient to execute only four Newton iterations.

Table 9.3: Influence of the threshold ϵ_N on the average number of the Newton iterations.

Sphere data set		
threshold ϵ_N	Iterations	time
1.0e-1	3.76	1.17
1.0e-2	4.94	1.21
1.0e-4	6.29	1.26
1.0e-8	8.00	1.28
1.0e-16	8.29	1.40

9.4.3 Memory Requirement

In a preprocessing step, the cells undergo a transformation. The information is stored in order to speed up the calculation. A data set consists of two large structures, one for the vertices and one for the cells. A vertex holds at least the information about one position-vector and one flow-vector. One cell, in our case a tetrahedron, knows its four edge-vertices and for visualization purposes its four neighbors. This will cause a memory need of 8 pointers or 32 Byte. The storage requirement for our variation of the method is higher, it also stores the three eigenvalues (3*double), three eigenvectors (3*3*doubles, the critical point (3*double) and the flag for the type information (1*int). We stored these things as doubles, so our cell needs 156 bytes of memory. If we look at a single cell, we need 4.85 times more memory. In practice this factor is approximately 3.85, if all cells can be used for the method. It decreases because the memory size for the vertex structure remains the same for both ways.

9.4.4 Time behavior

Table 9.4 shows the overall time, for all methods. The calculated paths consist always of the same number of steps and the same length, so the average step size is the same. The path itself can differ, because the accuracy is not the same in every case. We have measured our method with and without preprocessing, in the latter case the transformation is made when a curve enters a tetrahedron (on the fly) and not in a preprocessing step.

Table 9.4: Timetable for all methods.

Sphere data set	
method	time
Euler	4.92
Heun	6.63
Runge-Kutta 3	10.13
Runge-Kutta 4	10.34
Exact on the fly	15.5
Exact with memory	3.93

9.4.5 Advantages

The obvious disadvantage of our method is that it needs more memory. On the other hand, however, the particle tracing method we have presented is faster than the ordinary Euler method and even more accurate than the Runge-Kutta integration schemes of order 4, so you trade memory usage for accuracy and speed. Furthermore, with our tracing method we overcome the most important disadvantage of normal and even adaptive methods: The correct choice of the step size. For the user it is hard to estimate the error which results from a particular, chosen step size in particular, when the pathline traverses cell boundaries. An adaptive method tries to calculate this error. In order to do this reliable, assumptions about the smoothness of the flow have to be made. If this assumption with regard to the smoothness is not valid, a small vortex can be missed altogether using traditional methods. The new method presented here cannot miss such a vortex because it visits all the cells which are crossed by the trace, taking advantage of the grid structure.

9.5 Acknowledgements

The investigations were supported by the German Science Foundation DFG, which funds the Sonderforschungsbereiches 603 „Modellbasierte Analyse und Visualisierung komplexer Szenen und Sensordaten“, Teilprojekt C7 „Adaptive Verfahren zur Berechnung und Visualisierung von mechatronischen Sensoren und Aktoren“

References

[CL93] B. Cabral and L. Leedom. Imaging Vector Fields Using Line Integral Convolution. In J. T. Kajiya, editor, *Computer Graphics Proceedings*, volume 27 of *Annual Conference Series*, pages 263–270, Los Angeles, California, August 1993. ACM SIGGRAPH, Addison-Wesley Publishing Company, Inc.

[DH96] D. Darmofal and R. Haimes. An analysis of 3d particle path integration algorithms. *Journal of Computational Physics*, 123:182–195, 1996.

[Frü94] Thomas Frühauf. Interactive visualization of vector data in unstructured volumes. *Computers and Graphics*, 18:73–80, 1994.

[KL95] David Kenwright and David Lane. Optimization of time-dependent particle tracing using tetrahedral decomposition. In G. M. Nielson and D. Sil-

ver, editors, *IEEE Visualization '95*, Los Alamitos, CA, 1995. IEEE Computer Society Press.

[NJ99] Gregory M. Nielson and II-Hong Jung. Tools for computing tangent curves for lineary varying vector fields over tetrahedral domains. *IEEE Transactions on Visualization and Computer Graphics*, pages 360–372, 1999.

[NJS+97] G. M. Nielson, I.-H. Jung, N. Srinivasan, J. Sung, and J.-B. Yoon. Tools for Computing Tangent Curves and Topological Graphs for Visualizing Piecewise Linearly Varying Vector Fields over Triangulated Domains. In G. M. Nielson, H. Hagen, and H. Müller, editors, *Scientific Visualization: Overviews, Methodologies, and Techniques*, chapter 21, pages 527–562. IEEE Computer Society Press, Los Alamitos, California, 1997.

[SvWHP94] Ari Sadarjoen, Theo van Walsum, Andrea J. S. Hin, and Frits H. Post. Particle tracing algorithms for 3D curvilinear grids. Technical Report DUT-TWI-94-80, Delft University of Technology, 1994.

[TGE97] C. Teitzel, R. Grosso, and T. Ertl. Efficient and Reliable Integration Methods for Particle Tracing in Unsteady Flows on Discrete Meshes. In W. Lefer and M. Grave, editors, *Eighth Eurographics Workshop on Visualization in Scientific Computing*, pages 49–56, Boulogne sur Mer, France, April 1997. EG, The EuroGraphics Association.

[USM96] S. Ueng, K. Sikorski, and K. Ma. Efficient streamline, streamribbon, and streamtube constructions on unstructured grids. *IEEE Transactions on Visualization and Computer Graphics*, 2:100–110, 1996.

■

Chapter 10

Some Notes on Sampling in Computer Graphics

Andreas Schilling[*]

”Nobody will ever figure out how to do antialiasing” said Jim Blinn in his keynote address at Siggraph ’98. Already 21 years earlier, in 1977, he had included antialiasing in a list of unsolved problems in computer graphics which he had compiled together with Martin Newell. And although the quote from 1998 was not meant to be taken totally serious, he included the same problem again in his 1998 list of unsolved problems.

Antialiasing deals with artifacts that are a consequence of sampling, the process of forcing the quasi continuous world into discrete representations for digital processing. Since the 1970ies, when raster displays started to replace the vector displays, uniform sampling and reconstruction have always been performed in computer graphics, sometimes without knowing it or analyzing the problems in depth. The removal of the aliasing artifacts has been undertaken in many different ways, and some very interesting solutions have been developed, probably sometimes also without deeper analysis of the sampling process.

[*] WSI/GRIS, Universität Tübingen, Germany. E-Mail: schilling@uni-tuebingen.de

In this paper we will give a short introduction into sampling problems in image related research. In contrast to other work on this subject we put special emphasis on the analysis of the whole chain of filtering - sampling - reconstruction, which leads to a better understanding of what "ideal antialiasing" should mean. We will conclude with some notes on practical antialiasing techniques.

10.1 Introduction

Sampling problems are problems that arise when a continuous function has to be represented by a limited number of sample values. Well known examples are moiré effects in images or staircase artifacts in computer generated lines. But sampling problems do not only arise, when pixelated images are displayed: The representation of a surface by the vertices of a triangle-mesh or by control points in the case of a freeform surface represents a process of sampling and reconstruction, which leads to similar problems. In the last years many efforts have gone into solving problems related directly or indirectly to discrete sampling. In this context not only antialiasing techniques have to be mentioned, but also mesh simplification, multiresolution techniques and even methods for lighting scenes.

First we will give a short introduction into the problem of discretization. Then we will address some application areas in the image related sciences, where discretization problems play an important role.

The third section discusses solutions for the problems in the context of sampling and reconstruction.

Finally we will conclude with a note on mip-mapping and formulate a principle that must be considered when creating multiresolution meshes.

10.2 Discretizing Continuous Signals

The way digital computers work requires, that information represented by the computer is of discrete nature. Therefore, continuous signals can normally only be represented in the form of an approximation. The conversion of continuous functions (e.g. images or sounds) from the continuous form to a digital representation has two aspects: first, the values (scalar or vector) need to be represented by digital numbers. This discretization of the range is called quan-

tization and is not covered here in more detail, although it is in no way trivial, especially for vectors.

The second aspect, the topic of these notes, is the representation of a function with continuous domain by a finite number of such numbers. Often, the term discretization is used in this more narrow sense of discretization in the domain of a continuous function. One possibility for discretization (that even preserves a continuous domain) is to represent the function by a finite set of suitable basis functions.

10.2.1 Sampling and Reconstruction

In practice, functions are often discretized by sampling them at selected points within their domain. If the function fulfils certain conditions, it can be shown that this sampling corresponds to a representation of the original function by basis functions. These conditions are specified in the so called sampling theoremes (see below). If they are not met, special problems emerge, that have been extensively studied in the context of signal processing.

In computer graphics, the problem of sampling is most obvious in the context of image sampling. Images are stored and displayed in the form of raster images; therefore synthetic images as well as natural images have to be decomposed into lines and pixels.

The opposite process, the generation of a continuous function from discrete values, is called reconstruction. For images, this reconstruction is accomplished e.g. by outputting the sample values to a screen. Basically, reconstruction can be deemed to be a filtering step.

Fig. 10.1 gives an overview over the standard process of sampling and reconstruction of continuous functions. It suggests itself to perform the sampling on a regular grid, as not only the sampling but also the reconstruction is especially simple in this case. However, regular sampling is in no way imperative.

It can be seen that sampling is preceded by a filtering step which enforces the aforementioned conditions, under which reconstruction can be performed by multiplying the sample values with the corresponding basis functions.

Usually the filter that precedes sampling as well as the reconstruction filter are linear, shift invariant filters. However, this too is not imperative; some kinds of functions can only be reconstructed from sample values, if nonlinear filters are used. An example is shown in Fig. 10.2.

Fig. 10.3 shows different types of reconstruction filter kernels (basis functions) and for each of them an example of a function, that can be represented

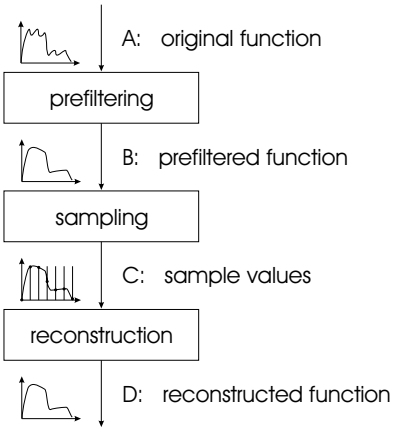


Figure 10.1: Standard process of sampling and reconstruction of continuous image signals

with these basis functions without error.

If the original function is not such a superposition of the basis functions, deviations or errors can not be avoided. In the context of sampling and reconstruction, we observe however not only such unavoidable approximation errors, but often the approximation is not performed in an optimal way, so that additional avoidable errors occur. A special type of error arises as a consequence of sampling the function without optimal prefiltering. High frequency components of the original function lead to bogus low frequency structures in the reconstructed function. This is called aliasing, see Fig. 10.4. Reducing aliasing errors is an active area of research in computer graphics and image processing.

10.3 Application Areas

10.3.1 Image Synthesis

Sampling problems in computer graphics arise in different contexts. Image synthesis, which has already been mentioned, is among the most important

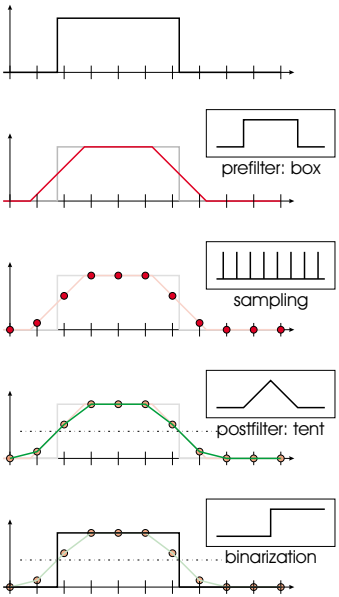


Figure 10.2: Imaginary output device with nonlinear reconstruction filter for exact recovery of an arbitrarily located unit step. The prefilter is a boxfilter of width 2, the reconstruction filter is a tent filter of width 2, followed by a nonlinear binarization step with threshold 0.5*. It can easily be shown, that the reconstruction only works correctly, if two edges don't get too close. If their distance is less than 2 pixels, the reconstructed edges may be displaced.*

ones. Geometrical descriptions of objects are used to calculate pixels for display on a screen. By nature the image functions cannot be represented without error by the basis functions of the output device. The image functions contain e.g. edges (discontinuities) or very small structures that manifest themselves as high frequency components. In contrast to digital sound processing, where as a consequence of physics no arbitrarily high frequencies can be present, here frequency is not bound on principle. With a suitable modeling language it is e.g. possible to describe easily and compactly a chequerboard pattern of any size. This is the reason that aliasing problems are particularly severe and apparent in rendering, and solutions are especially important in this area.

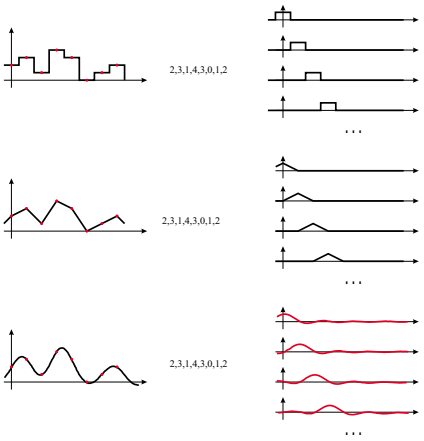


Figure 10.3: Reconstruction using different basis functions: a) box, b) tent, c) sinc

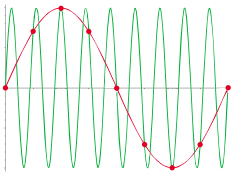


Figure 10.4: Formation of low frequency aliasing signals because of undersampling

10.3.2 Natural Images

However, aliasing can also be a problem, when recorded pictures are reproduced. Although the recording is performed in such a way that the image data contains little or no aliasing errors, this only ensures a correct reproduction if the image is not downsized, i.e. each recorded pixel corresponds to one reproduced pixel. Downsizing increases the frequencies contained in the image function and it is not possible to correctly display the image, if the display resolution is not increased correspondingly. In practice, this is often the case, if scenes with textures are rendered.

10.3.3 Image Sequences

Recording or generation of image sequences yields three dimensional data sets, that are discrete in each of the three dimensions. Changes in time, especially periodic events, can — like changes in space — only be reconstructed correctly, if the necessary conditions are met. If high temporal frequencies are undersampled, temporal aliasing is the consequence. An example is the seemingly reverse rotation of spoke wheels in movies.

10.3.4 Voxel Data

Recording and presentation of voxel data is another area where sampling problems play an important role. In many respects the results gained with two dimensional images and image sequences can be carried forward to the three dimensional case. Often, voxel data is the result of numerical simulations, but one of the most important applications is the processing of voxels that are the result of medical imaging techniques like CT and NMR. In the context of sampling there is a special problem with NMR voxel data sets: because of the involved physics, it is not possible to acquire the data very fast. Therefore, the scanned objects are normally sampled anisotropically: The voxel data set consists of individual image slices, and e.g. 9 out of 10 slices are left out. This corresponds to heavy undersampling in the direction perpendicular to the slices. Various algorithms have been developed to perform 3d-reconstruction from slice data. All of them depend on additional assumptions that are used to create hypotheses about the missing data.

10.3.5 Geometry

Representing geometrical models in computer graphics is another area where sampling problems play an important role. Surfaces in three dimensional space can be represented in very different ways:

1. Isosurface of a scalar function in 3d — implicit surface representation.
2. 3d-vector function with two dimensional domain.
3. Heightfield — distance function on a 2d plane.
4. Unparameterized set of vertices and neighborhood relations — polygon meshes.

The first three representations are based on functions that may be given analytically, or are represented in a discrete way. The fourth representation is already discrete in nature. The application of insights from signal processing is not obvious, yet the least complicated case is the first representation. Parameterized surfaces have the problem, that by the mapping from the parameter domain into the three dimensional space regular sampling grids are distorted. Considerations that assume regular sampling grids, can therefore not always be applied directly. In addition, many surfaces can not be mapped to a rectangular parameter domain. For this reason, the domain has to be split into disjoint parts (patches). The borders between these patches require special treatment. Unparameterized polygon meshes are very common in computer graphics. In most cases they are triangle meshes. Such meshes are generated manually by modeling but also as a result of different simplification methods from parameterized meshes. 3d-scanners output parameterized meshes with very high resolution that are often too large to handle. In order to deal with such meshes, simplification algorithms have been developed, that are based on omitting vertices and retriangulating the resulting holes. These algorithms work independent of the parameterization, a fact, that makes them suitable for models that are composed of several parameterized meshes. The resulting simplified meshes have lost their parameterization and the neighborhood relations are not regular anymore. An important task is the calculation of good parameterizations for such surfaces.

10.4 Errors

This section discusses errors, that arise as a consequence of discretization and reconstruction.

10.4.1 Aliasing Errors — Sampling Theorems

What we normally think of as the sampling theorem is the theorem named after E. T. Whittaker, J. M. Whittaker, V. A. Kotelnikov or C. E. Shannon [Whi15, Whi29, Whi35], [Kot33], [Sha49], cited here after [Jer77].

It states, that a band limited function can be reconstructed exactly from regular samples of the function, if these samples are close enough (at least two samples per smallest involved wavelength). It is less known however, that generalizations of this WKS sampling theorem exist for more general, finite limit

(truncated) integral transforms besides the usual Fourier transform [Jer77]. In addition to those sampling theorems there exist theorems that make statements about reconstruction from irregular samples [FG92].

The sampling theorems contain statements about the conditions under which functions can be reconstructed from samples, some also about the kind of errors that occur, if these conditions are not met. This means that at the point marked D in Fig. 10.1, we can get a reconstructed function that is identical to the function that was present at point B, if the conditions of the sampling theoreme is fulfilled. There are good reasons for the popularity of the Fourier related WKS sampling theorem. The Fourier transform is unique in the sense that the basis functions ($e^{j\omega t}$) are eigenfunctions of arbitrary convolution operators. As a shift operation can be regarded as a convolution operation, it is possible to shift band limited functions by an arbitrary distance without changing their Fourier spectrum. The consequence is that, if a function can be reconstructed from its samples, arbitrarily shifted versions of this function can also be reconstructed from their samples. This is very important, as the sampling grid with its actual positioning is a structure that is artificially imposed on reality. It is therefore desirable to be able to reconstruct independent of the position relative to the sampling grid. This is especially important, to ensure that moving objects are always reconstructed in the same way.

Which are the errors that occur if the sampling theorem is violated? Regular sampling corresponds to multiplying the function with a series of Dirac pulses. The Fourier transform of this comb function is again a comb function. Multiplication in the spatial domain corresponds to convolution in the frequency domain: this means that the spectrum of the sampled function in addition to the spectrum of the original function contains infinitely many copies of this original spectrum, each of them shifted by multiples of the sampling frequency, see Fig. 10.5 b).

If the original signal is band limited, it can be reconstructed from the sampled signal by low pass filtering (Fig. 10.5 c)).

If, however, the original spectrum overlaps its copies, this means that by the sampling process some frequency components of the original spectrum were mapped to other frequencies, frequencies that are too in the interesting part of the spectrum. In this case, it is no longer possible to separate the copy and the original. By subtracting a multiple of the sampling frequency, arbitrarily high frequencies of the original signal can lead to low-frequency aliasing errors. Aliasing errors can be totally avoided, if the original spectrum does not contain any components with frequencies higher than the so called Nyquist frequency,

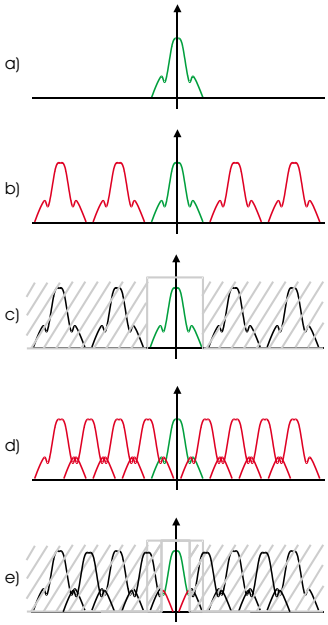


Figure 10.5: Reconstruction of sufficiently sampled signal and of undersampled signal.

which corresponds to half the sampling frequency. However, this is not a necessary condition [BD98], as it is possible to admit a frequency f_h above the Nyquist frequency instead of a frequency f_b below the Nyquist frequency, if the difference between f_b or $-f_b$ and f_h is an arbitrary multiple of the sampling frequency. Fig. 10.6 shows an example, where exact reconstruction of the green original spectrum is theoretically possible.

10.4.2 Errors Resulting from Filtering

The deviations from the original function that result from prefiltering are made intentionally and are necessary to avoid aliasing errors in the sampling process. An example for such errors is the so called ringing, that occurs as a consequence of low pass filtering (see Fig. 10.7). Low pass filtering is con-

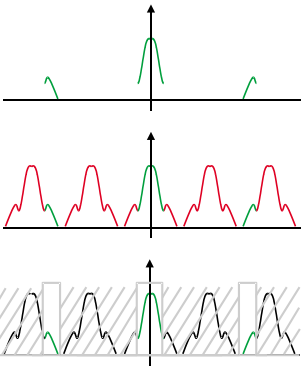


Figure 10.6: Sampling and reconstruction with a combination of low pass and band pass filters.

sidered to be the optimum prefiltering by many authors. We will see in more detail in section 10.5 below that this is not true in general.

Fig. 10.7 shows a low pass filtered image of a single point (Dirac pulse). The squared error is infinite in this case. As the Dirac pulse contains all frequencies with the same amplitude, the portion of the original signal contained in the filtered one remains the same if the low pass filter used as prefilter and reconstruction filter is replaced with a suitable band pass filter of the same width. When choosing a prefilter, not only the used reconstruction filter has to be considered, but also which frequencies are contained in the original image. If e.g. the original image does not contain low frequencies, a band pass filter — like the one shown in Fig. 10.7, right — gives better results than a low pass filter[2].

10.4.3 Errors due to Reconstruction

Reconstruction of the sampled signals has often been neglected. Mostly, it is performed by the output device, the properties of which are determined by the manufacturer. The most widely-used output device for images is still the CRT, followed by liquid crystal displays. In Fig. 10.8 it is shown, how the filtering

[2]Note that images always contain a DC component, as negative brightness values are not permitted. This component with frequency 0 should therefore not be suppressed.

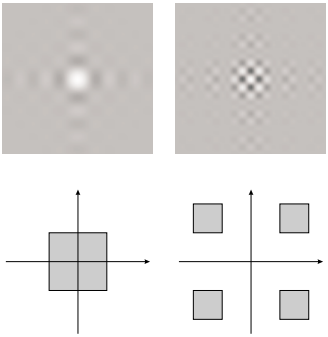


Figure 10.7: Left: Low pass filtered, magnified image of a single point on gray background. This is how, according to many authors, the image of a star on a display had to look like, if prefiltering as well as reconstruction are performed in an ideal way. Right: band pass filtered image of the same point. The portion of the original signal contained in this image is exactly the same as in the low pass filtered image.

properties of a CRT are formed: the samples are output line by line. A latch in front of the A/D converter holds the pixel values until the next one arrives. This corresponds to filtering with a box filter and is often called sample-and-hold circuit (although it should rather only be called hold circuit). The analog signal cannot contain arbitrary high frequencies for physical reasons, and in the process of conveying the signal to the CRT high frequencies are even more damped. The image signal is used to modulate an electron beam in the CRT, while the beam is deflected, to scan the screen line by line. This beam generates a bright spot on the screen, of which the brightness distribution can be approximated by a Gauss function. Because of this mechanism, the resulting point answer is not rotationally symmetric but spreads more in horizontal than in vertical direction (s. Fig. 10.9).

LCD displays are more simple to treat: every pixel shows up as a uniformly bright rectangle or square on the screen, so the reconstruction filter is a simple box filter. Although images displayed on such a screen still contain arbitrarily high frequencies, an observer is not bothered by these frequencies, as long as the pixels are sufficiently small. Spatial frequencies that cannot be resolved by the human visual system need not be suppressed by the output device; this part

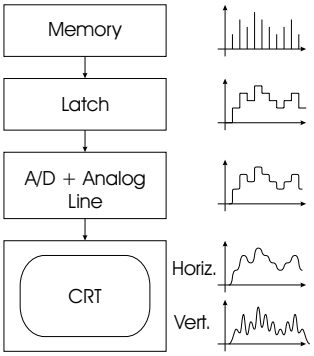


Figure 10.8: Output of an image on a CRT

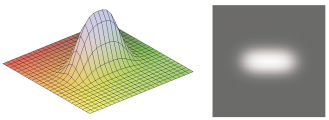


Figure 10.9: Point answer of a CRT-display

of the filtering is dealt with by the eye.

10.4.4 Errors Resulting from Non-Linearities

Nonlinearities in the process of image capture and output are an additional source of errors, that can lead to aliasing effects. As the range of the employed number representations is limited, dynamics are limited by principle, i.e. the recording system has a characteristic with not only linear, but also quadratic and higher order components. The application of such a characteristic to signals with different frequency components results in the generation of product terms and powers of the original signals. These products and powers contain the sums and differences of the original frequencies. As the difference of two desired signal frequencies i.e. frequencies below the Nyquist frequency is all the more below the Nyquist frequency, this difference signal is always in the utilizable range and appears as disturb signal. The sums of two frequencies

on the other hand can, if they are high enough, produce aliasing artifacts when interfering with the sampling frequency, even if the original signal was sufficiently band limited. Similar problems occur on the output side. Here, the most important cause is missing or insufficient gamma correction. In particular, the difference frequency of two close frequency components can appear as a clearly visible structure, even if the two involved original frequencies are suppressed completely by the eye of the observer, e.g. if the viewing distance is large enough.

10.4.5 Errors Caused by Perception

Human perception is the fundamental measure for rating sampling errors in computer graphics. Significant parameters are here the resolution limit of the eye (temporal and spatial) but also special detection mechanisms, that cause a special sensitivity of the eye for particular spatial or temporal patterns. The sensitivity of the eye for different spatial frequencies can be described by a band pass. Processing of time dependent signals in the human perception is especially important for antialiasing. On one hand the human eye is able to detect movement. This makes seeming movements caused by aliasing especially bothersome. On the other hand we have the ability to compensate uniform motion which allows us to analyze objects in detail even if they are moving. In the context of the aliasing problem the consequence of this ability is that position dependent changes of the appearance of moving objects which cannot be avoided when using certain antialiasing filters are particularly recognized. The human perception from the early processing in the sensors to the later processing in the brain is very complex and not totally understood. Therefore a quantitative consideration of cognition e.g. when developing antialiasing algorithms is not yet feasible.

10.5 Antialiasing — Optimal Approximation

For a long time researchers have tried to minimize or even get rid of errors caused by discretization. Coming from electrical signal processing, the focus was laid mostly on avoiding aliasing in the sense of the low frequency result of mixing high frequency signals and sampling signal. Many authors concluded from the WKS sampling theorem that the ideal way to avoid sampling errors is to filter the original signal with an ideal low pass filter. This conclusion can

even be found in the otherwise very nice and didactically well written presentation of Blinn [Bli89]. However, this approach only makes sense if a good approximation of the original signal is not necessary because high frequency components in the reconstructed signal are irrelevant e.g. if they are not received or analyzed by the receptor. Sound is an example, where this is the case. Sound signals are decomposed into their frequency components in the inner ear and then the amplitude is evaluated. Frequency components above a critical frequency (conditioned by age) cannot be perceived due to the physiology of the ear. While in the past discretization errors were generally called aliasing errors, and their analysis was restricted to the application of the WKS sampling theorem, recent treatises often discern between actual aliasing artifacts and reconstruction errors (e.g. [Gla95, Lev00]). This is, however, not sufficient to minimize errors caused by discretization. It is rather necessary to look at the whole signal chain from the original function up to the reconstructed function. In Fig. 10.1 this corresponds to the processing between points A and D. As deviations cannot be avoided, for optimizing the chain we need an error measure to rate the deviations. Unfortunately it is difficult to establish such a measure that takes into account the sensitivity of the human observer for particular errors in a quantitative way. Although knowledge about the (nonlinear) response of the eye would help, it would not be sufficient to establish an error measure as further processing of visual sensations in the brain is not predictable; this processing is affected among other factors by attention and memory. A straightforward choice for an error measure is the often employed mean squared error. Of course, frequencies above the resolution limit of the eye have to be omitted. However, for the time being, this limit is well above the Nyquist frequency for standard displays with normal viewing distance.

With the error measure given, it has to be evaluated, which parameters can be changed, i.e. with regard to which parameters we can optimize. In many cases the reconstruction filter is given and cannot be changed. When using e.g. a liquid crystal display, the reconstruction filter is a boxfilter. For this case it can easily be shown, that the prefilter has to be a boxfilter as well, if the output function shall approximate the original function optimally in a least squares sense. If we look at this situation in the frequency range we see that with this prefilter we get aliasing errors, but that avoiding these aliasing errors by blocking all frequencies above the Nyquist frequency would lead to an even larger contribution to the approximation error. As an example, we choose a signal with a frequency of $f_{\text{signal}} = 2.75 f_{\text{N}} = 1.375 f_{\text{S}}$, where f_{N} and f_{S} denote the Nyquist frequency and the sampling frequency resp., see

Fig. 10.10. If we would prefilter with an ideal low pass filter, we would totally eliminate the signal. In contrast, prefiltering with the box filter leaves a signal with amplitude $a_{\text{signal,prefiltered}} = \operatorname{sinc}(1.375\pi) \approx -0.21$ for sampling. In the sampling process, alias signals are produced having the frequencies: $f_k = (\pm 1.375 + k)\, f_S\, ; k = \pm 1, \pm 2, \ldots$. The amplitude of the original signal as well as the amplitudes of the newly created alias signals have still a value of ≈ -0.21, see Fig. 10.10 b). Reconstruction with a one pixel wide box filter results in a multiplication of these signals with a factor of $r_k = \operatorname{sinc}(\pi f_k / f_S)$. The low frequency alias signals thus have an amplitude of about -0.17 and -0.10 respectively, the signal with original frequency has an amplitude of $\approx (-0.21)^2 \approx 0.046$, and the higher frequency alias signals have accordingly smaller amplitudes. If we calculate an average aliasing amplitude as the square root of the sum of the squared amplitudes of all individual aliasing signals, we get: $a_{\text{aliasing}} = |a_{\text{signal, prefiltered}}| \sqrt{\sum_{k=1}^{\infty} r_k^2} = |\operatorname{sinc}(1.375\pi)| \sqrt{\sum_{k=1}^{\infty} \operatorname{sinc}(\pi(\pm 1.375 + k))^2} \approx -0.209$

It may seem not obvious that it makes sense that 4.6% of the original signal are paid for with more than 20% of aliasing — instead of no original signal but also no aliasing. But if we look at the squared error we see that we are able to decrease the error at the frequency of the original signal from $100\%^2$ (original signal totally suppressed) to $(100 - 4.6)\%^2 \approx 91.1\%$, a reduction of more than 8%, whereas the error caused by the alias signal only increases from 0% to $20.9\%^2 \approx 4.4\%$. In a similar way we can determine an optimal prefilter for the case of reconstruction by Gauss filtering. For the calculation of such dual filters that normally would require inversion of infinite dimensional matrices see [Str00].

In conclusion we can say that ideal low pass filtering cannot be considered to be always the optimal way to perform antialiasing since optimal approximation has to take into account the whole signal processing chain starting from the original function down to the reconstructed function. In addition, it should be noted that the measure for the approximation error also affects the optimization of the prefilter.

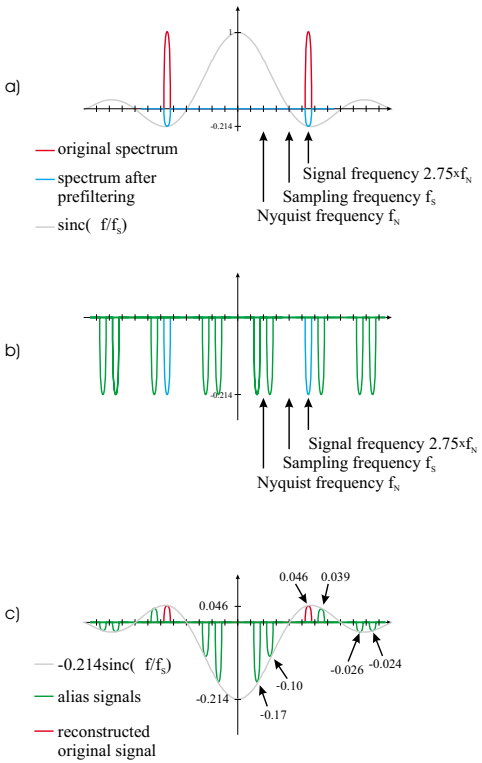


Figure 10.10: Prefiltering and reconstruction with a one pixel wide box filter: a) Spectra before and after prefiltering. b) Spectrum after sampling. Note that the y-scale has changed. c) Spectrum after reconstruction. Scale same as in b). For detailed explanation see text.

10.6 A Note on Mip-Mapping

Mip-mapping is the most commonly used technique for the antialiasing of textures. In spite of this fact, misconceptions about mip-mapping are common. In [MPFJ99] e.g., mip-mapping is mistaken as space invariant linear filtering. We will therefore briefly analyze mip-mapping.

A mip-map is created by space invariant linear filtering with subsequent

sampling. A box filter is frequently used for this purpose. Before using the mip-map to render an image, a further filtering step is needed which can be regarded to be a reconstruction filter to reconstruct the texture from the discrete samples in the mip-map combined with a prefilter before sampling the texture at the pixel locations. This filtering is normally performed by bilinear interpolation which corresponds to space invariant linear filtering with a tent function

$$l(x,y) = \begin{cases} (1-|x|)(1-|y|) & \text{for} \quad \max(|x|,|y|) \leq 1 \\ 0 & \text{otherwise} \end{cases}$$

as the filter kernel.

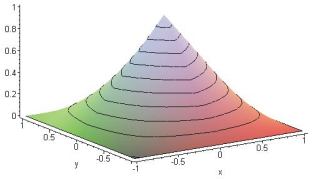


Figure 10.11: Bilinear interpolation corresponds to filtering with a tent function.

Fig. 10.11 shows a tent function, the kernel of a filter that performs bilinear interpolation. This filter is space invariant, in contrast to the filter, that results from the whole mip-mapping process of prefiltering, downsampling and bilinear interpolation. Fig. 10.12 shows this resulting space variant filter. Two different functions for two sample locations are shown. If the resulting filter would be space invariant, it should be possible to transform the red function into the green one only by shifting it. This is obviously not the case. Although the shown example was produced using a box filter the principle does not change, if other filters are employed for creating the mip-map. Fig. 10.13 shows what changes, if the mip-map is produced with a Gauss filter. The filter function is again a weighted sum of two functions that are scaled with scaling factors dependent of the sampling location. The only characteristic that changes is the shape of the two functions.

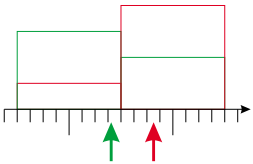


Figure 10.12: Filter functions for two different pixel locations. The function determines how the original texels contribute to the final pixel when mip-mapping is employed. The level in this example is level 3 (8 original texels correspond to one sample in level 3)

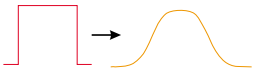


Figure 10.13: If a Gauss filter instead of a box filter is used to create the mipmap, the two stationary box functions that contribute to the final filter function (see Fig. 10.12) are replaced by a sum of Gauss functions.

10.7 A Note on Creating Geometrical Multiresolution Models

In this note, we present a fundamental principle that has to be applied when creating multi resolution models:

> When reducing the resolution of a geometrical model, detail information cannot just be discarded. Instead, its implications on the appearance must be analyzed and preserved in the model.

A multi resolution model contains representations of a model in different resolution levels. It can be generated from the high resolution representation. For this purpose, resolution is reduced step by step. By their nature, these simplification steps imply discarding information. Many simplification algorithms have been developed with the goal of discarding irrelevant information while preserving relevant information. The simplification of geometrical models is often performed by leaving out vertices (sampling). The choice of the vertices that are still needed is controlled by the error that occurs, if points are left out. This approach cannot avoid aliasing errors by undersampling, but these errors

are limited as well as all other kinds of errors by keeping the overall error below a given threshold.

By performing a kind of frequency analysis, wavelet based methods allow to omit selectively the high frequency components. This avoids aliasing errors, however sharp edges in the model are being unnecessarily smoothed.

It is common to all simplification methods that details which are not needed in the lower resolution level are left out. A representative value replaces many individual values. This practice is regularly used in other areas as well, e.g. the results of an examination of a whole class of students can be characterized by an average grade instead of listing the results for every single student. For some purposes this representation is even better suited than the list of individual grades. But other questions cannot be answered knowing only the average grade, e.g. the question how many have passed the exam.

If images are scaled down (e.g. when moving away from it), it is normally sufficient to reduce resolution in the standard way, i.e. by replacing a number of brightness values by one representative brightness value. In principle the same is true for transparency information.

However, the situation is different, if geometrical models are to be simplified for subsequent rendering. Up to now, it has been ignored in literature, that details that individually can well be neglected in a coarser model, collectively can be important for the macroscopic properties of the model, even on a coarse level. A coarse model of a water surface can serve as an example. Although, if we choose an appropriate level, the surface can be modeled by a (smooth) plane, the reflection properties depend strongly on small waves present on the water. A grid (fence, comb, etc.) can be represented as a plane face — if seen from enough distance. However, somehow we must make sure, that the object represented in this way does not occlude totally other objects behind it, but lets pass a certain fraction of the light.

From these observations it becomes clear, that detail information cannot just be omitted, if a geometrical model is being simplified. Instead we have to determine, which information is relevant for the rendering of the coarse model. We can then summarize this information and transform it to a suitable compact representation that can later be used to render the coarse model.

This important principle is illustrated in Fig. 10.14. In the bottom right pyramid, we store the relevant information from the high resolution model, that otherwise would have been lost. This information can be represented in different ways. It is, for example, possible to use images for this representation, images that have been rendered using the detailed geometrical model and

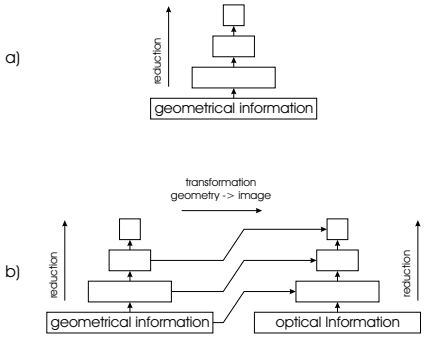


Figure 10.14: Generating a multi resolution model. a) Standard scheme: step-by-step reduction of resolution. b) Preserving important information from higher resolution model by transforming it and storing it in a separate resolution pyramid.

then have been reduced to an appropriate resolution. In the general case, however, we need informations about the direction dependent reflection properties (BRDF) or, in addition, about the transmission or transparency properties. The function that describes these combined informations could be called BRDTF (bidirectional reflectance and transmittance distribution function). The BRDTF should, like a texture, be defined for each point of a surface. Also like a texture, this BRDTF lends itself for generating multi-resolution models in a pyramid like scheme. From image based rendering suitable representations for image informations are known, location dependent BRDFs are also a topic of recent research [DvGNK99]. An alternative is to extract and store those geometrical informations that are relevant for rendering. In [Sch97] we have proposed to represent the geometrical information that is relevant for reflections by a matrix that characterizes the distribution of normal vectors in a region of the surface, or in other words the roughness of the surface in said region.

References

[BD98] M. G. Beaty and M. M. Dodson. An application of a general sampling theorem. *Result. Math.*, 34(3/4):241–254, 1998.

[Bli89] James F. Blinn. Return of the jaggy. *IEEE Computer Graphics & Applications*, pages 82–89, March 1989.

[DvGNK99] Kristin J. Dana, Bram van Ginneken, Shree K. Nayar, and Jan J. Koenderink. Reflectance and texture of real-world surfaces. *ACM Transactions on Graphics*, 18(1):1–34, January 1999. ISSN 0730-0301.

[FG92] H. G. Feichtinger and K. Gröchenig. Irregular sampling theorems and series expansions of band-limited functions. *J. Math. Anal. Appl.*, 167:530–556, 1992.

[Gla95] Andrew S. Glassner. *Principles of Digital Image Synthesis*. The Morgan Kaufmann Series in Computer Graphics and Geometric Modeling. Morgan Kaufmann, 1995.

[Jer77] Abdul J. Jerri. The shannon sampling theorem - its various extensions and applications: a tutorial review. *Proc. IEEE*, 65(11):1565–1596, November 1977.

[Kot33] Vladimir Alexandrowitsch Kotel'nikov. On the transmission capacity of "ether" and wire in electrocommunications. In *Izd. Red. Upr. Svyazi RKKA (Moscow)*, 1933. material for the first all-union conference on questions of communications.

[Lev00] Marc Levoy. Introduction to computer graphics. Lecture Notes, http://graphics. stanford. edu/courses/cs248-00/, 2000. Handout #6, Stanford University, Computer Graphics Lab.

[MPFJ99] Joel McCormack, Ronald Perry, Keith I. Farkas, and Norman P. Jouppi. Feline: Fast elliptical lines for anisotropic texture mapping. In Alyn Rockwood, editor, *Computer Graphics (SIGGRAPH '99 Proceedings)*, volume 33, pages 243–250, August 1999.

[Sch97] Andreas G. Schilling. Towards real-time photorealistic rendering: Challenges and solutions. In *Proceedings of the 1997 SIGGRAPH/EUROGRAPHICS Workshop on Graphics Hardware, Los Angeles, California, August 3–4, 1997*, pages 7–15, August 1997. Invited Paper.

[Sha49] C. E. Shannon. Communications in the presence of noise. *Proc. IRE*, 37:10–21, January 1949.

[Str00] Thomas Strohmer. Rates of convergence for the approximation of dual shift-invariant systems in $l^2(z)$. *J. Four. Anal. Appl.*, 5(6):599–615, 2000.

[Whi15] E. T. Whittaker. On functions which are represented by the expansion of the interpolation theory. *Proceedings of the Royal Society of Edinburgh*, 35:181–194, 1915.

[Whi29] John M. Whittaker. The fourier theory of the cardinal functions. *Proc. Math. Soc. Edinburgh*, 1:169–176, 1929.

[Whi35] John M. Whittaker. *Interpolatory Function Theory*. Cambridge tracts in mathematics and mathematical physics ; 33. Cambridge University Press, Cambridge, UK, 1935.

■

Chapter 11

Physically-based techniques for creating animations

Friedrich Wagner, Dietmar Jackèl[*]

In this paper, we discuss various control techniques for creating physically-based animations. For the incorporation of constraints, which is a key problem for both modeling and motion control, we describe different dynamics formulations and perform an evaluation for their use in the context of computer animation. We suggest new arguments for the advantages of the Lagrange multiplier formulation (LMF) and present an easy-to-use, modular, interactive animation system, based on the LMF.

11.1 Introduction

The development of tools for the generation of physically-based animations is an important application in the field of computer graphics. A decisive advantage of the dynamic approach, compared to kinematic methods, is the simple generation of very realistic motions of complicated objects. Also, because the

[*] Lehrstuhl Interaktive Graphische Systeme, Universität Rostock, Germany. E-Mail: {dj|wagner}@informatik.uni-rostock.de

dynamic behavior of such a system can be changed easily and the motions are generated automatically, this approach provides a high degree of reusability.

An important but difficult task is the flexible and interactive motion control of complex mechanical systems according to the imagination of an animation operator. The imbedding of dynamics into an interactive and "easy-to-use" animation system, which also has to meet requirements such as modularity and generality, is therefore an important aim of current research activities. However, the problems to be solved in order to reach this aim are very complex. The physically based approach requires an animation operator to carry out very complicated tasks such as the specification and solution of the equations of motion as well as the use of dedicated control techniques, which often require extensive mathematical and physical background knowledge.

The following section gives an overview of the three different approaches for physically based motion control techniques that are used in computer animation systems: controller techniques, optimization-techniques and constraint-based methods. Because of the importance of constraints we discuss in section 11.3 dynamics formulations for constraint-based mechanical systems and in this context the Lagrange multiplier formulation (LMF) in more detail. It can be shown that the LMF is especially suited for realizing a physically based animation system that fulfils the demands mentioned above. Finally, an implementation of an animation system based on the LMF will be presented in brief. This easy-to-use system enables the flexible and fully interactive creation of physically based animations.

11.2 Motion control

The different approaches to motion control, shown in figure 11.1, are based on a classification by Zeltzer [Zel91] which was used for a taxonomy of control techniques of bio-mechanical animation systems.

The bottommost layer represents the dynamic scene description. Above this structural layer we find the procedural layer, which contains dynamics and kinematics methods that can be concretely realized by means of controller, optimization and constraint techniques. On the uppermost layer we find the autonomously acting agents, which are rule-based controlled by means of other techniques such as natural-language commands. These high-level control techniques, discussed in an overview in [C+99], are used for the generation of complex motion sequences. They are based on controllers, constraints and op-

timization methods, which will be considered in the following subsections in more detail.

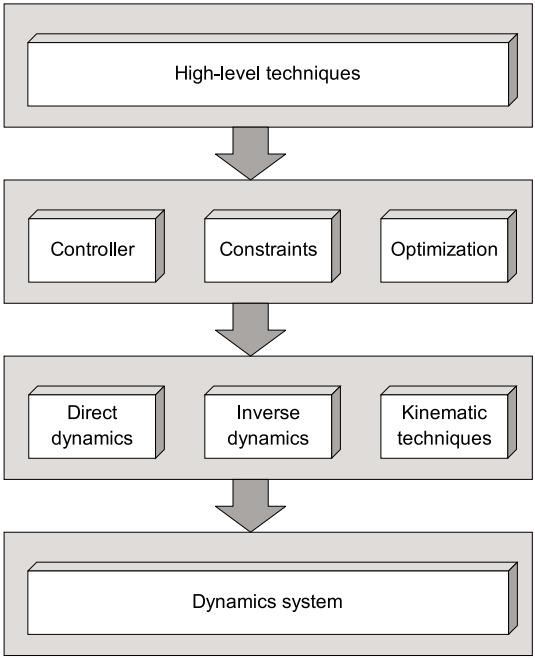


Figure 11.1: Hierarchy of control techniques.

11.2.1 Controller

Controllers provoke explicit time-dependent or state-dependent forces. In robotics, controllers represent the standard technique for the control of articulated figures. By means of an error-function, e.g. $\mathbf{x}^E(t) := \mathbf{x}^{goal} - \mathbf{x}(t)$, where $\mathbf{x}$ denotes the current state of a dynamic system and $\mathbf{x}^{goal}$ the goal state vector, a force $\mathbf{F}$ is defined, which acts contrary to the deviation of the wanted motion. For a more robust simulation the often used "PID-controller" incorporates also

an integral term (I) and a differential term (D):

$$\mathbf{F} = -K_X\,\mathbf{x}^E + K_V\,\dot{\mathbf{x}}^E - K_I \int \dot{\mathbf{x}}^E\,dt\,. \tag{1}$$

The parameters K_V, K_X and K_I represent hereby diagonal matrices with positive elements, which must be assigned concrete values before the simulation is executed. This assignment can be very difficult and application-dependent. A serious drawback to control motion with the technique of controllers is their locally bounded impact. It is therefore often very difficult or even impossible to achieve certain motion goals with this technique.

The controller technique has often been used for the modeling of joint limits and ground contacts in the first attempts at physically based animations of articulated figures (e.g. [AG85], [WB85], [A$^+$87], [Wil87]). It has also been applied for jump motions of a one-legged robot [RH91], walking behavior [BC89], [MZ90], [RH91], [P$^+$92a], [SC92], diving [WH96] or other natural-looking motion forms [IC87], [P$^+$92b], [H$^+$92], [P$^+$93]. In [L$^+$95] and [LG96] a particularly adapted controller enables the specification of kinematic trajectories as motion goals for rigid bodies, without neglecting their dynamic behavior.

11.2.2 Optimization techniques

Optimization techniques are particularly suitable for the generation of a number of special motions such as the creeping of a worm or the swimming motions of a fish. The goal of the optimization techniques is the minimization of physical parameters, such as the sum of energy or force, which are formulated as a cost function. The idea behind this approach is that a minimized cost function precisely corresponds with the motion behavior of many natural systems. For a system depending of the state and velocity variables $\mathbf{x}$ and $\mathbf{v}$, a frequently stated optimization problem consists e.g. of minimizing the following term with regard to Newton's equation of motion:

$$J = f(\mathbf{x}(t_1), \mathbf{v}(t_1)) + \int_{t_0}^{t_1} g(\mathbf{x}(t), \mathbf{v}(t), t)\,dt$$

where t_0 and t_1 denotes the start and termination times of the simulation process, g stands for the cost function (e.g. the total sum of the applied forces) and f denotes a functional, which is evaluated at the end of the simulation period

(e.g. the difference to a given goal state $\mathbf{x}(t_1) - \mathbf{x}^{goal}$). Generally, this problem can be solved by using iterative algorithms.

These algorithms can be very time-consuming because the cost function has to be minimized over the entire simulation period. Moreover, the optimization approach permits no interactive intervention into the process of motion generation. This method therefore belongs to the *offline techniques*, in contrast to *online techniques* such as controller- or constraint-based control. Also, the selection of the parameters of the cost function often turns out to be very difficult and application-dependent.

Optimization techniques were used in [Gir91], [P+92b], [P+92a], [P+93] for the reproduction of human or animal motions. A special use of optimization techniques, which was introduced in [WK88] and further developed in [Coh92] and [NM93], is the so-called "space time constraint" concept. The equations of motion are treated in this approach as secondary conditions, which have to be fulfilled at each place and each time. Taking these conditions into account, the amount of computing time can be extremely large. In the past few years however, this approach was successfully applied for modification and combination of motion fragments ([Gle97], [PW99], [Pop00]).

11.2.3 Constraints

In the framework of physically based animation, constraints play the role of kinematic conditions, which must be fulfilled during the simulation. They are an essential tool for the modeling of complex systems, but also for the flexible control of the generated motion. An example of a constraint is the presence of an impenetrable obstacle that limits the motion space of an articulated figure. Also, kinematic relations between individual rigid bodies can be formulated with constraints to which in particular all joints such as ball, hinge or sliding joints belong. Additional examples of constraints are predefined start and termination points for certain motions, impassable motion boundaries or kinematic motion paths. Constraints are also the basis for many control techniques with a higher degree of abstraction. The use of constraints as a modeling tool, in particular for the description of articulated figures, can be found in nearly in all work on physically based animations of mechanical systems. In [IC87] and [IC88] constraints were introduced as a general motion control technique; in [BC89] they were applied for the animation of the human gait. The purely kinematic control of articulated figures by constraints was described e.g. in [B+87], [P+90], [PB91].

Due to the importance of constraints for physically-based animation the difficult problem of their solution will be discussed in the next section. We show that the choice of the dynamics formulation plays an important role for the application of constraint-based techniques in the framework of computer animation.

11.3 Dynamics formulations for constraint-based mechanical systems

A free mechanical system can be described by a *k*-dimensional time-dependent state vector **x** and the equation of motion $\mathbf{M}(\mathbf{x},t)\ \ddot{\mathbf{x}}(t) = \mathbf{F}(\mathbf{x},\dot{\mathbf{x}},t)$, where the *k*-dimensional square matrix **M** represents the mass distribution of the bodies and the k-dimensional vector **F** denotes the influence of the external and internal forces. To control such a system, constraints are needed, which represent mathematical relations between the state variables. An important class are *holonomic* constraints, which are functions of the state variables and time: $\mathbf{C}(\mathbf{x},t) = 0$. Another class, named *velocity-constraints*, is characterized by the equation $\mathbf{C}(\mathbf{x},\dot{\mathbf{x}},t) = 0$. Both classes are acceleration-linear constraints, defined by the equation

$$\mathbf{J}(\mathbf{x},\dot{\mathbf{x}},t)\cdot\ddot{\mathbf{x}} + \mathbf{c}(\mathbf{x},\dot{\mathbf{x}},t) = 0\ , \tag{2}$$

where **J** denotes an $(m \times n)$-matrix and **c** an *n*-dimensional vector in the case of an *n*-dimensional system with *m* scalar constraints. In the following, some common formulations (described in more detail for example in [Sha98]) to incorporate these constraints are discussed.

11.3.1 Lagrange multiplier formulation (LMF)

From a physical viewpoint, constraints are satisfied by force effects. These forces are explicitly determined by the LMF. Because of the principle of virtual work, they obey the equation $\mathbf{F}^{\,C} = \mathbf{J}^T\,\lambda$, where the components of the vector λ are denoted *Lagrange multipliers*. The constraint equation (2) and the equation of motion, to which we add the constraint forces, form the following system of differential-algebraic equations (DAE):

$$\begin{aligned} \mathbf{M}\,\ddot{\mathbf{x}} - \mathbf{F} - \mathbf{J}^T\,\lambda &= 0 \\ \mathbf{J}\,\ddot{\mathbf{x}} + \mathbf{c} &= 0 \end{aligned} \tag{3}$$

In order to solve this DAE, the Lagrange multipliers can be explicitly computed. We rewrite these equations in the form:

$$(\mathbf{J}\,\mathbf{M}^{-1}\mathbf{J}^T)\,\lambda = -(\mathbf{J}\,\mathbf{M}^{-1}\mathbf{F} + \mathbf{c})\ . \tag{4}$$

This linear equation system can be solved by numerical methods to determine the Lagrange multipliers. After placing these values in the equation

$$\ddot{\mathbf{x}} = \mathbf{M}^{-1}(\mathbf{F} + \mathbf{J}^T\,\lambda) \tag{5}$$

we are able to solve this system of ordinary differential equations (ODE) by means of standard numerical methods.

All approaches based on acceleration-linear constraints in terms of equation (2) have a fundamental difficulty, which is called the *drifting problem*. For holonomic constraints, this equation corresponds to the vanishing of the second derivative $\ddot{\mathbf{C}} = 0$, which is not only compatible with $\mathbf{C} = 0$, but also with a linear time-dependent constraint function $\mathbf{C} = \mathbf{a}\,t + \mathbf{b}$, which denotes an increasing violation of the constraint condition.

This problem can be solved by constraint stabilization techniques. One appropriate method, used in [BB88] and [Pla92], is given by the so-called *Baumgart-stabilization*. By this approach the term $\mathbf{c}$ in equation (2) is substituted:

$$\mathbf{J}\,\ddot{\mathbf{x}} + \mathbf{c}_h\ = 0\ , \quad \mathbf{c}_h := \mathbf{c} + 2\,\alpha\,\dot{\mathbf{C}} + \beta^2\,\mathbf{C}\ , \tag{6}$$

where α and β are constant parameters. In an alternative approach [W$^+$90] the multipliers are at first computed as usual. After this step, the term $\mathbf{J}^T(2\,\alpha\,\dot{\mathbf{C}} + \beta^2\,\mathbf{C})$ is introduced into the system as an additional force.

The LMF was already used in [IC88], [A$^+$89] for the generation of physically-based animations. In [BB88] the properties of the LMF were exploited to support the interactive modeling process. An extension of this approach for general constraints was presented in [W$^+$90] and [Pla92]. The excellent usability of the LMF for animating deformable bodies and heterogeneous systems was proved by [PB88], [WW90], and [H$^+$95]. The interactive specification of constraints was described in [P$^+$00].

For a long time, the efficient use of the LMF was regarded as very complicated. LMF-algorithms very often had square-law or cubical-time-complexity with regard to the number of d.o.f.'s. Significant progress for using this method was presented by David Baraff in [Bar96]. This approach, which serves as the central part of the simulation environment of our animation system described

in section 11.4, allows the simple and efficient implementation of the LMF and is characterized by a linear time-complexity for constraints that affect only two bodies.

11.3.2 Other approaches

Penalty approach. The penalty approach is the most simple method to integrate constraints into the equations of motion and was for example applied in [MW88] for the handling of rest contacts. It is based on the reformulation of constraints as spring forces. In the case of holonomic constraints, the spring forces are defined by $\mathbf{F}^{Penalty} := -\alpha\,\mathbf{J}^T\,(\Omega^2\,\mathbf{C} + 2\,\Omega\,\mu\,\dot{\mathbf{C}})$, where $\mathbf{J}$ is the Jacobian of $\mathbf{C}$, while α, Ω and μ denote constant parameters. By adding the spring forces as additional external forces to the dynamic system, an approximate satisfaction of the constraints can be obtained. Only if the penalty-term α tends to infinite, are the constraints exactly satisfied. As a major disadvantage, we obtain so-called *stiff* ODE's, which can only be solved by time-consuming computation.

Reduced coordinate approach. This method exploits the property of holonomic constraints to reduce the number of d.o.f.'s of the system. A set of $n-m$ independent coordinates clearly describes an n-dimensional system, controlled by m (scalar) holonomic constraints. The selection of the independent, generalized coordinates and the transformation of the state variables into the generalized coordinate system is the first step of the reduced coordinate approach. Based on this specification an $(n-m)$-dimensional system of ODE's can be formulated (e.g. with the Lagrange formulation), which is solvable by standard numerical methods. Nonholonomic constraints are excluded by using this method. Another disadvantage is that for the explicit parameterization in terms of the independent coordinates a nonlinear equation-system must be solved. Moreover, singularities, which can appear during the simulation process, often force a change in the set of generalized coordinates.

The reduced coordinate approach has been used in a number of previous works for the animation of articulated figures (e.g. [AG85], [A+87], [IC87], [BC89]). A recursive variant for hierarchical models, that is time-linear with respect to the number of d.o.f.'s, was introduced in [Fea83]. An extension for nonhierarchical systems was presented in [Lat86] and applied in [SZ90] and [VC91].

	LMF	Penalty	Reduction	Partition.	Direct
Efficiency and robustness	high	poor	high	reasonably	poor
Velocity constraints	yes	yes	no	add.	yes
Explicit constraint forces	yes	(yes)[1]	add.	add.	add.
Gradual constraint fulfillment	yes	yes	(no)[2]	(no)[2]	(no)[2]
Unreduced equations of motion	yes	yes	no	yes	yes
Unseparated state variables	yes	yes	no	no	yes
Closed kinematic loops	yes	yes	add.	yes	yes

add.: only with additional effort
[1] Spring forces instead of exact constraint forces.
[2] Only realizable with substantial extension of the formulation.

Figure 11.2: Fulfillment of the demands with regard to the dynamics formulations.

Coordinate partitioning approach. The coordinate partitioning approach, which is based like the LMF on the algebraic equation system (3), is mainly found in the area of technical simulation. In contrast to the LMF, this method separates the coordinates into $n-m$ independent coordinates $\mathbf{q}_u$ and m dependent coordinates $\mathbf{q}_v$ during the simulation. For this task numerical methods such as LU- or QR-factorization can be used. After the coordinate partitioning, $\mathbf{q}_u$ is evaluated by integrating the equation of motion only in respect to the independent coordinates. Finally, the computation of the dependent variables follows, by means of the constraint equations.

This method can also be extended for velocity constraints and allows the formulation of the equations of motion in arbitrary (nonreduced) coordinates. On the other hand, this approach is confronted with the same problem as in the case of the reduced coordinate approach, the automatic determination of independent coordinates. Moreover, the efficiency of the LMF or the reduced coordinate method cannot be reached by the coordinate partitioning approach.

Direct numerical approach. The differential-algebraic system of equations (3) can be solved directly with dedicated numerical methods, without beeing transformed into a set of ordinary differential equations. But this method is less efficient and less robust than the other approaches that were discussed above.

11.3.3 Comparison and valuation of the LMF

In this section a comparison and valuation of the LMF in respect to the other formulations follows. In this context some application aspects will be especially emphasized, which are very important for the interactive creation of animations. The fulfilment of these demands with regard to the dynamics formulations is summarized in table 11.2. This investigation, discussed in more detail in [Wag01], is a first attempt at a systematic method analysis in the context of the interactive generation of physically based animations.

The analysis shows that the problems of achieving a time-efficient simulation, the flexible and easy-to-use motion control and the design of an interactive animation system are connected in a very complicated way.

In the past, these problems have been insufficiently investigated, because they were mainly considered as problems of technical mechanics instead of computer graphics. This estimation does not seem appropriate. In particular we want to show, that the Lagrange multiplier formulation (LMF) is more suitable for the development of physically-based animation systems than other formulations. The practical impact of these aspects is discussed in section 11.4 by means of a modular animation system based on the LMF.

Efficiency and robustness. Efficiency and robustness are two basic requirements for the interactive generation of physically-based animations. The LMF and the reduced coordinate approach fulfill these requirements slightly better than the coordinate partitioning and much better than the penalty- and the direct numerical approach. Detailed investigations of this subject can be found in [GB94], [Sha98].

Velocity constraints. If the reduced coordinate approach is used, one property of the LMF and other methods, the integration of velocity constraints, is missing. This type of constraint plays an important role for flexible motion control.

Explicit determination of constraint forces. The explicit determination of constraint forces is an integrated part of the LMF and requires in contrast to the other formulations no additional time expenditure. These forces can be considered as additional dynamical features, which are useful for the design of special control techniques. Another possible application is their feedback to an input device, in order to give the animation operator an intuitive system response to his actions.

Gradual satisfaction of constraints. A special feature of the LMF and the penalty approach is the gradual satisfaction of the specified constraints. Constraints are not satisfied instantly but by the influence of forces. In the context of the flexible and intuitive creation of animations, this feature is very useful, because:

- the system modeling process by means of the constraints can be easily observed and intuitively controlled in this way ([BB88], [Gle94]).
- the integration of flexible bodies can be simplified as well as the imbedding of numerical integration routines [Pla92].
- the creation of animations can be simplified, since the constraints used for system modeling and for system control are the same.
- the constraints can be activated or deactivated during a running simulation, without the appearance of motion discontinuities.

Modularity and generality. The LMF is especially suited for designing an animation system with a modular and general architecture. This peculiarity is based firstly in the system definition by means of non-reduced coordinates. Because of this, bodies and constraints can be formulated independently of each other. Another advantage is the schematic conversion from the geometrical and physical system description into its mathematical formalism. The LMF requires no determination of independent coordinates and no transformation to appropriated generalized coordinates. Moreover, the LMF can be applied independently of the initial state of the system and of the topology of the body connections, which is very important with regard to a general animation system. In particular, a special treatment of kinematic loops is not necessary.

11.4 An animation system based on the LMF

In this section the modular animation system EMPHAS (**E**asy-to-use, **M**odular, **PH**ysically-based **A**nimation **S**ystem) developed by us is discussed, its conception beeing based on the LMF. EMPHAS is designed to produce animations interactively by means of a set of pre-defined constraints and pre-defined rigid bodies in an easy-to-use way. Modularity is a key feature of EMPHAS. The system allows the interactive modeling and animation by a “tool box“ of modular control techniques, consisting of a set of holonomic constraints, velocity

constraints, controller, force fields and event-based procedures. Moreover, an automatic treatment of collisions and rest contacts of colliding bodies has been integrated, as well as the support of the animation operator by means of a number of interaction methods. The modular design of the system is shown in picture 11.3.

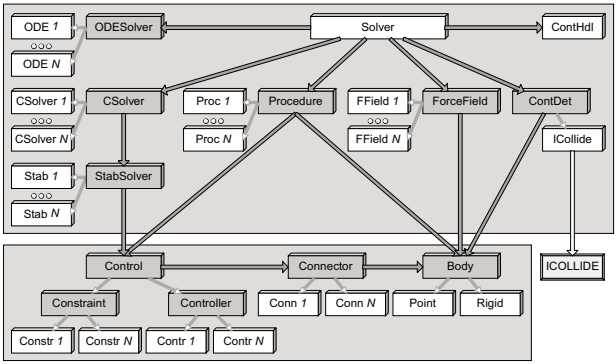


Figure 11.3: Architecture of the simulation environment of EMPHAS.

For the determination of the Lagrange multipliers the algorithm described in [Bar96] is used, which has been extended by us for a robust handling of over-determined systems. For this task the special property of Baraff's method of treating the so-called *primary* and *auxiliary* constraints separately has been exploited. Primary constraints have an effect on only two bodies and are not allowed to form kinematic loops, so they cannot form an over-determined system. Therefore we apply a singular value decomposition method (see for example [Mac90]) only for the solution of the auxiliary constraints, the number of which is normally much less than the number of primary constraints. By means of this new approach the LMF can be applied to simulate ill-conditioned and over-determined systems robustly and efficiently.

For the stabilization of the LMF (see section 11.3.1) the techniques in [BB88] and [W$^+$90] can both be applied. The resulting ODE's are solved by a number of standard numerical methods, that can be selected during run time.

In EMPHAS mechanical systems consist of point particles, predefined rigid bodies and arbitrary polyhedrons, which can be imported via a VRLM-

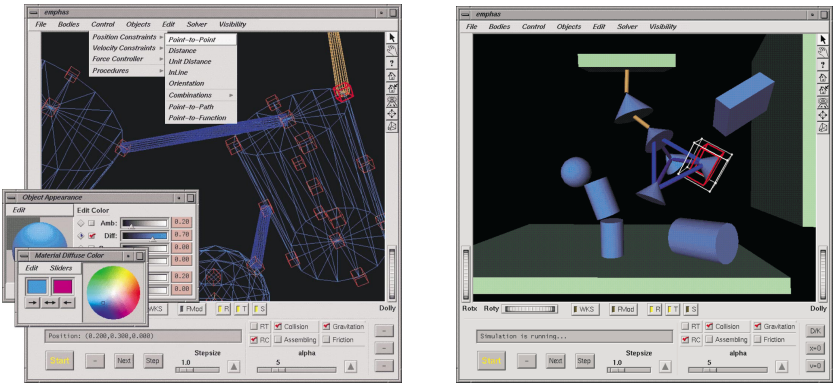


Figure 11.4: Examples of EMPHAS-scenes.

interface. For rendering in further process stages an export of the animation data to other animation systems (e.g. *3D Studio Max*) has been realized. Two examples of scenes in EMPHAS are shown in figure 11.4.

In the following the most important aspects for the interactive animation generation with EMPHAS are given, which exploit the special properties of the LMF. Details of these aspects are given in [Wag01].

"Tool box" concept. Based on the connector concept introduced in [W$^+$90], which has been extended by us for the treatment of velocity-constraints and explicitly time-dependent connectors, constraints and bodies could be enclosed as independent "black box" modules. It was also possible to realize a very modular and easy-to-use means of applying the control components. Rigid bodies and mass points, curves and three-dimensional fixed points, as well as time-dependent 3D motion trajectories, serve as connection points for constraints in a consistent manner.

Combinability. An important feature of EMPHAS is the arbitrary combinability of all control elements including the processes for the collision simulation and the treatment of rest contacts. As is shown in [Wag01], the special properties of the LMF to satisfy constraints gradually by a physically-based process were very helpful in the handling of this problem.

Interaction techniques. In EMPHAS, selections and editing of parameters and state values as well as generation or deletion of single bodies and control elements can be executed during the simulation in run time. These interactions can cause violations of the constraints, but because of their gradual satisfaction by means of the LMF no motion discontinuities appear.

Integration of event-based procedures. By exploiting the special properties of the LMF a generalized concept for the application of event-based procedures has been realized. This concept allows a procedural modification of the system state and all parameters of bodies, controller, and constraints.

11.5 Conclusion

In creating physically-based animation different techniques for motion control can be used, of which constraints play the most important role. We have shown that the selection of the dynamics formulation for simulation of constraint-based mechanical systems has a significant but not sufficiently investigated impact for the interactive motion creation. It was proved that the LMF is more suitable for this purpose than other common approaches. The practical impact of this investigations is demonstrated with the modular animation system EMPHAS, which enables the easy-to-use and effective creation of physically-based animations.

References

[A+87] W. Armstrong et al. Near-real-time control of human figure models. *IEEE Computer Graphics and Applications*, 7(6):52–61, 1987.

[A+89] B. Arnaldi et al. Dynamics and unification of animation control. *Visual Computer*, 5(1/2):22–31, 1989.

[AG85] W. Armstrong and M. Green. The dynamics of articulated rigid bodies for purposes of animation. *Visual Computer*, 1(4):231–240, 1985.

[B+87] N.I. Badler et al. Articulated figure positioning by multiple constraints. *IEEE Computer Graphics and Applications*, 7(6):28–38, June 1987.

[Bar96] D. Baraff. Linear-time dynamics using lagrange multipliers. *Computer Graphics (Proc. SIGGRAPH94)*, 30:137–146, 1996.

[BB88] R. Barzel and A. Barr. A modeling system based on dynamic constraints. *Computer Graphics (Proc. SIGGRAPH88)*, 22(4):179–188, 1988.

[BC89] A. Bruderlin and T. Calvert. Goal-directed, dynamic animation of human walking. *Computer Graphics (Proc. SIGGRAPH89)*, 23(3):233–242, 1989.

[C+99] E. Cerezo et al. Motion and behavior modelling: State of art and new trends. *Visual Computer*, 15:124–146, 1999.

[Coh92] M. Cohen. Interactive spacetime control for animation. *Computer Graphics (Proc. SIGGRAPH92)*, 26(2):293–302, 1992.

[Fea83] R. Featherstone. The calculation of robot dynamics using articulated-body inertias. *International Journal of Robotics Research*, 2(1):13, 1983.

[GB94] J. Garcia de Jalon and E. Bayo. *Kinematic and dynamic simulation of multibody systems*. Springer-Verlag, New York, 1994.

[Gir91] M. Girard. Constrained optimization of articulated animal movement in computer animation. In Norman I. Badler et al., editors, *Making them Move*, pages 209–232. Morgan Kaufmann, 1991.

[Gle94] M. Gleicher. *A Differential Approach to Graphical Manipulation*. Phd thesis, Carnegie Mellon University, 1994.

[Gle97] M. Gleicher. Motion editing with spacetime constraints. In Michael Cohen and David Zeltzer, editors, *1997 Symposium on Interactive 3D Graphics*, pages 139–148. ACM SIGGRAPH, April 1997.

[H+92] J. Hodgins et al. Generating natural-looking motion for computer animation. In *Proceedings of Graphics Interface '92*, pages 265–272, May 1992.

[H+95] M. Harada et al. Interactive physically-based manipulation of discrete/ continuous models. *Computer Graphics (Proc. SIGGRAPH95)*, 29:199–208, 1995.

[IC87] P. Isaacs and M. Cohen. Controlling dynamic simulation with kinematic constraints, behavior functions and inverse dynamics. *Computer Graphics (Proc. SIGGRAPH87)*, 21(4):215–224, 1987.

[IC88] P. Isaacs and M. Cohen. Mixed methods for complex kinematic constraints in dynamic figure animation. *Visual Computer*, 4(6):296–305, 1988.

[L+95] A. Lamouret et al. Combining physically-based simulation of colliding objects with trajectory control. *Journal of Visualization and Computer Animation*, 6(2):71–90, 1995.

[Lat86] R.H. Lathrop. Constrained (closed-loop) robot simulation by local constraint propagation. In *Robotics and Automation*, pages 689–694. IEEE Council on Robotics and Automation, 1986.

[LG96] A. Lamouret and M. Gascuel. Scripting interactive physically-based motions with relative paths and synchronization. *Computer Graphics Forum*, 15(1):25–34, 1996.

[Mac90] A.A. Maciejewski. Dealing with the ill-conditioned equations of motion for articulated figures. *IEEE Computer Graphics and Applications*, 10(3):63–71, 1990.

[MW88] M. Moore and J. Wilhelms. Collision detection and response for computer animation. *Computer Graphics (Proc. SIGGRAPH88)*, 22(4):289–298, 1988.

[MZ90] M. McKenna and D. Zeltzer. Dynamic simulation of autonomous legged locomotion. *Computer Graphics (Proc. SIGGRAPH90)*, 24(4):29–38, 1990.

[NM93] J. Ngo and J. Marks. Spacetime constraints revisited. *Computer Graphics (Proc. SIGGRAPH '93)*, 27:343–350, August 1993.

[P+90] C.B. Phillips et al. Interactive real-time articulated figure manipulation using multiple kinematic constraints. *Computer Graphics (Proc. SIGGRAPH90)*, 24(2):245–250, March 1990.

[P+92a] M. van de Panne et al. Control techniques for physically-based animation. In G. Hegron and D. Thalmann, editors, *Computer Animation and Simulation 1992*, Eurographics, pages 1–15, Cambridge, 1992.

[P+92b] J. Park et al. Realistic animation using musculotendon skeletal dynamics and suboptimal control. In G. Hegron and D. Thalmann, editors, *Computer Animation and Simulation '92*, Eurographics, 1992.

[P+93] M. van de Panne et al. Physically-based modeling and control of turning. *CVGIP - Graphical Models and Image Processing*, 55(6):507–521, 1993.

[P+00] J. Popovic et al. Interactive manipulation of rigid body simulations. *Computer Graphics (Proc. SIGGRAPH 2000)*, pages 209–218, 2000.

[PB88] J.C. Platt and A.H. Barr. Constraint methods for flexible models. *Computer Graphics (Proc. SIGGRAPH88)*, 22(4):279–287, 1988.

[PB91] C.B. Phillips and N.I. Badler. Interactive behaviors for bipedal articulated figures. *Computer Graphics (Proc. SIGGRAPH91)*, 25(4):359–362, July 1991.

[Pla92] J. Platt. A generalization of dynamic constraints. *CVGIP - Graphical Models and Image Processing*, 54(6):516–525, 1992.

[Pop00] Z. Popović. Controlling physics in realistic character animation. *Communications of the ACM*, 43(7):50–58, July 2000.

[PW99] Z. Popović and A. Witkin. Physically based motion transformation. *Computer Graphics (Proc. SIGGRAPH99)*, 33:11–20, 1999.

[RH91] M. Raibert and J. Hodgins. Animation of dynamic legged locomotion. *Computer Graphics (Proc. SIGGRAPH91)*, 25(4):349–358, 1991.

[SC92] J. Stewart and F. Cremer. Beyond keyframing: An algorithmic approach to animation. *Proc. Graphics Interface 1992*, pages 273–281, 1992.

[Sha98] A. Shabana. *Dynamics of Multibody Systems, 2nd Edition*. Wiley, New York, 1998.

[SZ90] P. Schröder and D. Zeltzer. The virtual erector set: Dynamic simulation with linear recursive constraint propagation. *Computer Graphics (Proc. 1990 Symposium on Interactive 3D Graphics)*, 24(2):23–31, 1990.

[VC91] N. Vasilonikolidakis and G.J. Clapworthy. Inverse Lagrangian dynamics for animating articulated models. *Journal of Visualization and Computer Animation*, 2(3):106–113, 1991.

[W+90] A. Witkin et al. Interactive dynamics. *Computer Graphics (Proc. SIGGRAPH90)*, 24(2):11–21, 1990.

[Wag01] F. Wagner. *Konzepte und Methoden zu allgemeinen, physikalisch basierten Animationssystemen auf der Grundlage der Lagrange-Faktoren-Methode*. Phd, University of Rostock, 2001.

[WB85] J. Wilhelms and B. Barsky. Using dynamic analysis to animate articulated bodies such as humans and robots. In *Proc. Graphics Interface 1985*, pages 97–104, 1985.

[WH96] W. Wooten and J. Hodgins. Animation of human diving. *Computer Graphics Forum*, 15(1):3–13, 1996.

[Wil87] J. Wilhelms. Using dynamic analysis for realistic animation of articulated bodies. *IEEE Computer Graphics and Applications*, 7(6):12–27, 1987.

[WK88] A. Witkin and M. Kass. Spacetime constraints. *Computer Graphics (Proc. SIGGRAPH88)*, 22(4):159–168, 1988.

[WW90] A. Witkin and W. Welsh. Fast animation and control of nonrigid structures. *Computer Graphics (Proc. SIGGRAPH90)*, 24(4):243–252, 1990.

[Zel91] D. Zeltzer. Task-level graphical simulation: Abstraction, representation, and control. In Norman I. Badler et al., editors, *Making them Move*, pages 3–33. Morgan Kaufmann, 1991.

■