JYVÄSKYLÄ STUDIES IN COMPUTING
71

Pavlo Turchyn

Adaptive Meshes in Computer Graphics and Model-based Simulation

JYVÄSKYLÄN YLIOPISTO

JYVÄSKYLÄ STUDIES IN COMPUTING 71

Pavlo Turchyn

Adaptive Meshes in Computer Graphics and Model-based Simulation

Esitetään Jyväskylän yliopiston informaatioteknologian tiedekunnan suostumuksella
julkisesti tarkastettavaksi yliopiston Agora-rakennuksessa (Ag Aud. 3)
joulukuun 28. päivänä 2006 kello 10.

Academic dissertation to be publicly discussed, by permission of
the Faculty of Information Technology of the University of Jyväskylä,
in the building Agora, Ag Aud. 3, on December 28, 2006 at 10 a.m.

UNIVERSITY OF JYVÄSKYLÄ

JYVÄSKYLÄ 2006

Adaptive Meshes in Computer Graphics and Model-based Simulation

JYVÄSKYLÄ STUDIES IN COMPUTING 71

Pavlo Turchyn

Adaptive Meshes in Computer Graphics and Model-based Simulation

UNIVERSITY OF JYVÄSKYLÄ

JYVÄSKYLÄ 2006

Editors
Tommi Kärkkäinen
Department of Mathematical Information Technology, University of Jyväskylä
Irene Ylönen, Marja-Leena Tynkkynen
Publishing Unit, University Library of Jyväskylä

URN:ISBN:9513927172
ISBN 951-39-2717-2 (PDF)

ISBN 951-39-2722-9 (nid.)
ISSN 1456-5390

Jyväskylä University Printing House, Jyväskylä 2006

ABSTRACT

Turchyn, Pavlo
Adaptive meshes in computer graphics and model-based simulation
Jyväskylä: University of Jyväskylä, 2006, 27 p.(+included articles)
(Jyväskylä Studies in Computing
ISSN 1456-5390; 71)
ISBN 951-39-2717-2 (PDF), 951-39-2722-9
Finnish summary
Diss.

This work presents improvements to mesh generation algorithms employed in computer graphics and numerical solution of boundary value problems of elliptic type.

The first part of the thesis concerns creation of the meshes with various polygonal complexity, which are used in computer graphics to create an image of a given object. This work contributes to the analysis of the sliding window progressive meshes algorithm. Several improvements to the algorithm are suggested to solve its major problems. The first problem is the cache-coherent access to the mesh vertices; it is solved with help of heuristics-based reordering of triangles. The second problem is the excessive size of resulting datasets. It is demonstrated that the datasets can be reduced using optimization of mesh connectivity, via hierarchical data structures, and with help of special mesh operators that remove several vertices at a time.

The second part of the thesis is focused on the a posteriori error estimation for finite element approximations in terms of linear functionals. Here the role of mesh generation is two-fold. First, the error value estimated with help of estimator naturally suggests mesh refinement strategy. Second, the estimator requires solving an additional *adjoint* problem on a mesh that does not coincide with the mesh used to solve the main problem. In order to relief the requirements for the adjoint mesh, a new method to estimate the error is developed and tested; the method not only accepts anisotropic adjoint meshes, but also does not require extra regularity of the adjoint problem solution.

Keywords: continuous level-of-detail, sliding window progressive meshes, finite element method, a posteriori error estimation, quantities of interest

Author

Pavlo Turchyn
Department of Mathematical Information Technology,
University of Jyväskylä,
Finland

Supervisors

Professor Pekka Neittaanmäki
Department of Mathematical Information Technology,
University of Jyväskylä,
Finland

Professor Sergey Repin
V.A. Steklov Institute of Mathematics,
St.-Petersburg, Russia

Reviewers

Professor Nikolay Banichuk
The Institute for Problems in Mechanics of RAS,
Moscow, Russia

Dr. Satyendra Tomar
J. Radon Institute of Computational and Applied Math.,
Linz, Austria

Opponent

Assistant Prof. Maxim Frolov
St.-Petersburg Polytechnical University,
Russia

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to Prof. Pekka Neittaanmäki and Prof. Sergey Repin for their continuous guidance and support. I would like to thank my colleagues Dr. Sergey Korotov, as well as to Dr. Tuomo Rossi and Dr. Viktor Kalvin for encouraging discussions. I am thankful to Prof. Nikolay Banichuk and Dr. Satyendra Tomar for reviewing the thesis and making valuable suggestions, and to Markku Häkkinen for the assistance with English language.

This work was carried out in the course of several projects. The major part of polygonal mesh simplification code was developed within the framework of Agora Center SIMPLY project supported by Elomatic Papertech Engineering Oy. The estimator for linear elasticity problems was developed as a part of SCOMA project supported by TEKES. Author was also supported by COMAS Graduate School of the University of Jyväskylä, and the project 104409 of Academy of Finland.

Jyväskylä, 14th of December 2006
Pavlo Turchyn

LIST OF FIGURES

CONTENTS

LIST OF INCLUDED ARTICLES

I Sergey Korotov and Pavlo Turchyn, Topology-Driven Progressive Mesh Construction for Hardware-Accelerated Rendering, *Computer Graphics and Geometry, 6(3), pp. 100–119, 2004*

II Pavlo Turchyn, Memory-Efficient Sliding Window Progressive Meshes, *To appear in Journal of WSCG*, 2007

III Sergey Korotov and Pavlo Turchyn, A posteriori error estimation of "Quantities of Interest" on Tetrahedral Meshes, *In CD-Rom Proceedings of ECCOMAS'2004, the 4th European Congress on Computational Methods in Applied Sciences and Engineering, Eds.: Neittaanmäki, P., Rossi, T., Korotov, S., Onate, E., Periaux, J., Knörzer, D., Volume II, Jyväskylä, Finland*, 2004

IV Pekka Neittaanmäki, Sergey Repin, Pavlo Turchyn, A Posteriori Error Estimation in Terms of Linear Functionals for the Linear Elasticity Problems, *Revised version to appear in Russ. J. Numer. Anal. Math. Modelling.*, 2007

1 INTRODUCTION

1.1 Polygonal mesh simplification

Virtual worlds with rich details require large amounts of information to be processed. On the other hand, most interactive applications are limited to the computational capacity of the computer hardware upon which they reside. Thus, one faces a fundamental problem of computer science: the search for a balance between data size and processing time. In particular, this problem arises when processing geometrical information during image synthesis. Normally a large amount of information is discarded at the stage of hidden surface removal using a variety of algorithms, such as *view frustum culling*, *occlusion culling*, *potentially visible sets*, etc. However, even after removal of all invisible geometrical primitives, the complexity of scene may still be excessive for the rendering and display hardware. For instance, primitives with a projection size near one pixel may cause noticeable *noise* due to *aliasing*; primitives with a sub-pixel projection size may be simply invisible, and thus are a waste of processing time; the cost of geometry processing may be too high for the underlying graphical hardware, so that interactive framerates are impossible. The main approach for reducing geometrical complexity of the scene consists of replacing the initially provided meshes with their reasonable approximations.

Let us give a formal description of the process of constructing an optimal approximation of a given mesh. Let $M_0 = (V_0, T_0)$ be a given 2-manifold triangle mesh in $\mathbb{R}^3$, where V_0 is a set of vertices, and T_0 is a set of triangles. We look for an optimal approximation $M_r = (V_r, T_r)$ of the mesh M_0 as the solution of the following combinatorial optimization problem

$$J_\alpha(M_r) = \min_{M_i \in \mathbb{M}} J_\alpha(M_i), \tag{1}$$

where $\mathbb{M} = \{M_0, M_1, \ldots, M_N\}$ is a set of all possible meshes constructed over a given set of points P. One may set $P = V_0$, or choose P as a set of points sampled within the bounding volume of M_0, etc. Typically, the following goal functional J is used

$$J_\alpha(M_i) := \alpha E(M_0, M_i) + (1-\alpha) C(M_i),$$

where $E(M_0, M_i)$ is a metric that evaluates a visual difference between the meshes M_0 and M_i (the forms of this functional are described in the subsequent section), the functional

$C(M_i)$ evaluates expenditures related to the image synthesis using the mesh M_i, and the parameter α adjusts the relative importance.

In general, problem (1) is very complicated because the solution space $\mathbb{M}$ may have an extremely large dimension. Many authors believe that the problem might be *NP-hard* (see e.g. [LRC*02, p.20]), which means that the polynomial-time algorithms for solving this problem may not exist. Although some commercial systems claim to use sophisticated optimization approaches, such as *genetic algorithms*, a generally accepted approach is the use of *greedy* iterative algorithm, which is explained in the subsequent sections. Although it is assumed that the mesh M_n, which is obtained with help of the latter algorithm, is close to M_r, the exact difference between M_n and M_r is unknown.

1.1.1 Visual quality criterion

Assigning a numerical value to the visible difference between two polygonal approximations of a real-world object is not a trivial task. Aside from being subjective and context-dependent, such difference is heavily influenced by human visual perception. Appearance of an object depends on the region of retina at which its image is projected, on its motion speed, on the distance at which the eyes are converged, etc. In practice, the large body of these factors is disregarded, in particular due to the lack of widely available eye tracking systems.

One way to measure visual difference between the meshes M_0 and M_i is to create two images I_0 and I_i, respectively. Then, the metric E is a distance between two images, e.g. RMS pixel-by-pixel difference [LH00].

Unfortunately, the image-based method is relatively slow. A faster approach is to compute geometrical distance between two meshes, such as *point-to-surface distance* (see e.g. [Hop96, CRS98])

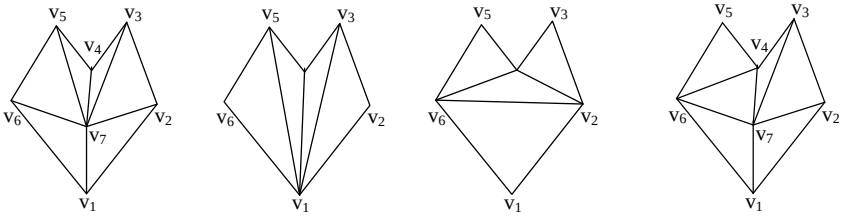
$$E(M_0, M_i) = \sum_{v_j \in Q} |v_j - \mathscr{M} v_j|,$$

where Q is a set of points sampled on the surface of the mesh M_i, and the operator $\mathscr{M}$ maps a point v_j to the closest point on the surface of mesh M_0. Since implementation of $\mathscr{M}$ involves spatial proximity queries, this method is also relatively slow. A cheaper distance measure is *point-to-plane distance*, such as *quadratic error metric* (QEM, see [GH98]). QEM is defined as

$$E(M_0, M_i) = \sum_{v_j \in V_i} \sum_{\triangle_k \in L(v_j)} |v_j - \mathscr{P}(\triangle_k) v_j|^2,$$

where the operator $\mathscr{P}(\triangle_k)$ projects a point onto the plane of triangle $\triangle_k$. Let $S(v_j)$ be a subset of triangles of the mesh M_i that are incident to the vertex v_j. Informally, the set $L(v_j)$ includes the triangles of mesh M_0 that form a part of the surface being approximated by the surface formed by the triangles of $S(v_j)$. Correspondence between the sets $L(v_j)$ and $S(v_j)$ is naturally established in the simplification process, so finding $L(v_j)$ is simple and does not involve proximity queries.

Typically, mesh simplification is an iterative process with n steps that produces a sequence of approximations $M_0 \equiv M^{(0)}, M^{(1)}, \ldots, M^{(n-1)}, M^{(n)}$. One way of accelerating the computation of the metric $E(M_0, M_i)$ during the iterative process is to replace the



a) M_0 b) $\mathscr{H}(v_7, v_1)M_0$ c) $\mathscr{R}(v_7)M_0$ d) $\mathscr{E}(v_5, v_7, v_4, v_6)M_0$

FIGURE 1 Examples of mesh operators

initial mesh M_0 (which is relatively complex) with a simpler mesh produced at some preceding iteration. In other words, we use $E(M^{(j)}, M_i)$ instead of $E(M_0, M_i)$. Surprisingly, in some cases such a strategy not only results in a very small error increase, but even leads to the improvement of the resulting mesh [LT99, Hop99a]. The latter is explained by the fact that in practice $E(M_0, M_i)$ may demonstrate a complicated behavior; some sequences of meshes lead to its local minima. On the other hand, $E(M^{(j)}, M_i)$ may be less problematic for the optimization algorithm in use.

1.1.2 Iterative algorithm

As we have already noted, the problem (1) is often solved with help of an iterative algorithm that generates a sequence of meshes $M_0 \equiv M^{(0)}, M^{(1)}, \ldots, M^{(n)}$ where

$$M^{(i)} := \mathscr{O}(P_i)M^{(i-1)}, \tag{2}$$

where $\mathscr{O}$ denotes a mesh operator (the structure of such operator is explained below), and P_i is a set of operator parameters. The process stops at the iteration n when the cost $C(M^{(n)})$ is below an application-defined threshold.

For $\mathscr{O}$ we take one of the following operators

- *half-edge collapse* operator $\mathscr{H}(v_i, v_j)$ replaces vertex v_i with vertex v_j in all triangles of the mesh, and then removes resulting degenerate edge and corresponding degenerate triangles from the mesh; Fig. 1b displays the result of applying $\mathscr{H}(v_7, v_1)$ to the mesh M_0 shown in Fig. 1a, so the edge (v_1, v_7) and the triangles (v_6, v_7, v_1) and (v_1, v_7, v_3) are removed.

- *vertex removal* operator $\mathscr{R}(v_i)$ deletes vertex v_i and all the incident triangles from the mesh and then re-triangulates resulting hole with a triangulation algorithm, such as [BDE96, BS96]; the example is shown in Fig. 1c;

- *edge swap* operator $\mathscr{E}(v_k, v_\ell, v_m, v_n)$ replaces two triangles $\triangle_i = (v_k, v_\ell, v_m)$ and $\triangle_j = (v_\ell, v_k, v_n)$, which share the common edge (v_k, v_ℓ), with the triangles $\triangle'_i = (v_k, v_n, v_m)$ and $\triangle'_j = (v_\ell, v_m, v_n)$ (see the example in Fig. 1d).

In our method we construct a sequence of meshes that generally satisfies the rule (2) by choosing during each step the operators $\mathscr{O}$ and its parameters P_i such that they solve the following combinatorial optimization problem

$$E(M^{(i)}) = \min_{\mathscr{O}(P_i) \in \mathbb{O}} E\left(M^{0)}, \mathscr{O}(P_i)M^{(i-1)}\right). \tag{3}$$

Here the solution space $\mathbb{O}$ consists of the operators $\mathscr{H}$, $\mathscr{R}$, and $\mathscr{E}$ with all possible combination of parameters. As we have discusssed, the parameters of $\mathscr{H}$ are the pairs of vertices connected with an edge to be removed; the parameters of $\mathscr{E}$ are the vertices of two triangles that share a common edge. Thus, the dimension of $\mathbb{O}$ is bounded

$$|\mathbb{O}| < C\,N,$$

where N is the number of mesh vertices, and C is a constant that depends on mesh *genus*. Thus, in principle, we can obtain the solution of (3) with help of the *complete enumeration* strategy.

The iterative algorithm we describe is *greedy*. The latter term is used for only locally-optimal algorithms that choose the best operation for a single iteration. The further iterations are not taken into account.

1.1.3 Progressive meshes

It is typical that a graphical application creates not a single image, but a series of images. The process (2) constructs a certain set of meshes $\mathbf{M} = \{M^{(0)}, M^{(1)}, \ldots, M^{(n)}\}$. In practice, it is typical that all elements of $\mathbf{M}$ may be required, so that the application could choose the most appropriate approximation for a each image.

However the set $\mathbf{M}$ is too big to be effectively stored as an array of separate meshes. Hoppe [Hop96] proposed to store only the initial mesh $M^{(0)}$ and the respective sequence of operators with parameters (which is rather inexpensive to store). Assume that a variable μ holds a mesh approximation $M^{(i)} \in \mathbf{M}$. If an application want to use a coarser approximation, it applies the transformation $\mathscr{O}(P_{i+1})\mu$, where the operator $\mathscr{O}$ and its parameters P_{i+1} are taken from the precalculated sequence. Otherwise, if an application wants to use a finer mesh approximation, it applies the transformation $\mathscr{O}^{-1}(P_i)\mu$, where $\mathscr{O}^{-1}$ is the inverse of the respective operator $\mathscr{O}$, e.g. the inverse of *edge collapse* operator is *vertex split* operator. The required mesh approximation is obtained by progressively applying the precalculated sequence of transformations, so this method is named *progressive meshes*. However, this method may demonstrate several drawbacks explained below.

Some applications create images that contain several objects of the same type, e.g. an image of a snowfall contains a big number of identical snowflakes. Such objects, which are called *instances*, represent the same type of an object, so it is logical to use the same mesh for each instance. Each instance has a given orientation and position in space, so the mesh vertices need to be transformed by a corresponding linear operator. Normally, each instance should use its own optimal mesh approximation, so each instance needs a separate memory buffer to hold its own approximation (which is actually being transformed). As the number of instances grows, the total required amount of memory may become prohibitive.

Another drawback of the progressive meshes is as follows. The process of image synthesis involves a certain processing performed per mesh vertex. Let V be a set of mesh vertices. The basic processing is

$$\tilde{v}_i = Av_i, \quad i = 1,2,\ldots,|V|,$$

where A is the transformation matrix. Storing the vertices $\tilde{v}_1, \tilde{v}_2, \ldots, \tilde{v}_{|V|}$ would require a memory buffer of the size $|V|$. One can avoid the additional memory expenditures

simply by not storing the processed vertices at all. Let T be a list (ordered set) of mesh triangles. A graphical processor sequentially takes triangles from T, one-by-one. For each triangle, the processor first transforms the triangles vertices, and then creates an image of this triangle. After that the transformed vertices are simply discarded, and the processor continues to the next triangle from the list T. The obvious disadvantage of this approach is increased computational expenditures since each vertex, which is incident to several triangles, will be processed multiple times.

In order to reduce the amount of computations the graphical processor keeps a small number of recently processed vertices (typically, 16 vertices) in a memory buffer called *vertex cache* (see [Hop99b]). If two subsequent triangles from the list T share a vertex, the processor uses an already transformed vertex from the vertex cache when processing the second triangle. High efficiency of such a caching strategy demands a specific order of triangles in T. On the other hand, maintaining this order in progressive meshes is a complicated task because triangles are removed or added as mesh level-of-detail changes. Many authors discuss integration of vertex cache into progressive meshes (see e.g. [ESAV99, Ste01, BG01, For01, SP03]). In order to obtain a feasible triangles order, the approaches require an additional amount of processing.

1.1.4 Sliding window progressive meshes

Now we proceed to the description of our work. The scheme named *sliding window progressive meshes* (SWPM) was developed by Forsyth [For01] to solve the instancing problem of progressive meshes. However, the original algorithm has introduced several problems itself. The main drawback was relatively poor utilization of the vertex cache. Another problem was relatively high memory consumption; although the scheme offered instancing without additional memory expenditures, the memory requirements for storing a single (non-instanced) progressive mesh were high. In our work we propose several improvements to SWPM that address both disadvantages.

The general idea of SPWM is as follows. The mesh operators $\mathscr{H}$, $\mathscr{R}$, $\mathscr{E}$ affect only a small number of topologically connected mesh triangles. A patch is a set of triangles that contains the triangles modified by an operator. Formally, we define a patch for the operators $\mathscr{H}$ and $\mathscr{R}$ as follows. Let $M = (V,T)$ be a given mesh. Let $S(v_i)$ be the set of the triangles incident to the vertex v_i, i.e. $S(v_i) = \{\triangle_\ell | \triangle_\ell \ni v_i, \triangle_\ell \in T\}$. We call a subset $P(v_i)$ a patch if $S(v_i) \subseteq P(v_i) \subseteq T$; here v_i is the *inner vertex*.

Let $\mathbf{P} = \{P_1, P_2, \ldots, P_n\}$ be a list (ordered set) of patches, such that all patches are pairwise disjoint. Additionally,

$$P_1 \cup P_2 \cup \cdots \cup P_n = T, \tag{4}$$

so all triangles of the original triangulation T are included into the patches. For simplicity we assume that each patch is a list of triangles, i.e. the order of the triangles in a patch is defined in some manner. We define a list of *simplified patches* $\mathbf{Q} = \{Q_1, Q_2, \ldots, Q_n\}$. Each *simplified patch* $Q_i \in \mathbf{Q}$ is obtained by applying a simplification operator to the corresponding patch $P_i \in \mathbf{P}$, i.e. $\mathscr{R}(v_j) : P_i(v_j) \longmapsto Q_i$, or $\mathscr{H}(v_j, v_k) : P_i(v_j) \longmapsto Q_i$, where $v_k \in P_i(v_j)$. We stress that a simplification operator removes the inner vertex only. Thus, we can replace each P_i with Q_i while maintaining conformity (i.e. no cracks or

T-joints) of the mesh.

We construct a memory buffer, where we put the elements of **P** followed by the elements of **Q**. Now can obtain the required mesh approximation simply by choosing the appropriate *window* in this buffer. This is illustrated in Fig. 2. The original mesh $M_0 \equiv M^{(0)}$ is obtained using the window $P_1 \cdots P_n$, the approximation $M^{(1)}$ is the window $P_2 \cdots Q_1$, the approximation $M^{(2)}$ is the window $P_3 \cdots Q_2$, etc.

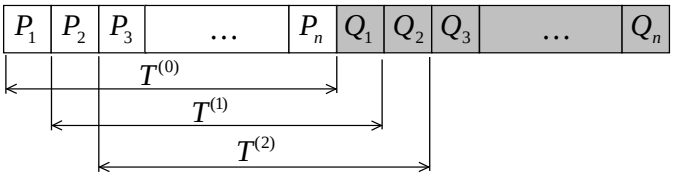


FIGURE 2 Sliding window progressive mesh

1.1.5 Memory-efficient SWPM

First we derive the bounds for the size of SWPM memory buffer, which is shown in Fig. 2. We denote the buffer as B_0. One can interpret B_0 as

$$B_0 = P_1 \cup P_2 \cup \cdots \cup P_n \cup Q_1 \cup Q_2 \cup \cdots \cup Q_n.$$

The first part of the buffer, which contains the non-simplified patches from the list **P**, forms triangles $T^{(0)}$ of the initial mesh $M^{(0)} = (V^{(0)}, T^{(0)})$

$$T^{(0)} = P_1 \cup P_2 \cup \cdots \cup P_n.$$

The last part of the buffer, which contains the simplified patches from the list **P**, forms the triangles of the approximation $M^{(n)} = (V^{(n)}, T^{(n)})$ obtained at the last iteration of the process (2)

$$T^{(n)} = Q_1 \cup Q_2 \cup \cdots \cup Q_n.$$

Then, the number of triangles in the buffer B_0 is a sum of the number of triangles in $T^{(0)}$ and the number of triangles in $T^{(n)}$

$$|B_0| = |T^{(0)}| + |T^{(n)}|.$$

In what follows we assume that the meshes are 2-manifolds without holes. The operator $\mathscr{H}$ (or $\mathscr{R}$) remove exactly two triangles from a patch, thus

$$|T^{(n)}| = \sum_{i=1}^{n} |Q_i| = \sum_{i=1}^{n} (|P_i| - 2) = |T^{(0)}| - 2n, \tag{5}$$

where n is the number of patches. Thus, one has to maximize n in order to minimize $|B_0|$. It follows from the definition of a patch that maximization of n is the *maximum independent set* problem solved on a mesh connectivity graph; the set of inner vertices is a *maximum independent set*. This is a classical *NP-hard* problem. There exist polynomial-time algorithms for finding its approximate solution with a good accuracy. One greedy

algorithm for planar graphs is described by Shoeyink and van Kreveld [SvK97]. The algorithm's lower bound for the size of independent set is $n > |V_0|/6$. It follows from the Euler's relation that

$$n > \frac{|V^{(0)}|}{6} > \frac{|T^{(0)}|}{12} \tag{6}$$

On the other hand, we note that each patch contains at least three triangles, so

$$n \leqslant \frac{|T_0|}{3}. \tag{7}$$

Combining (5)–(7), we obtain the bounds

$$\frac{1}{3}|T^{(0)}| \leqslant |T^{(n)}| < \frac{5}{6}|T^{(0)}|. \tag{8}$$

Thus, the number of triangles in the mesh $M^{(n)}$ is bounded. However, some applications may need the approximations, which contains less triangles than the value of lower bound. Therefore, one has to simplify $M^{(n)}$ further. Using $M^{(n)}$ as an initial mesh, we construct another SWPM buffer. We denote this buffer B_1. Again, the simplest triangulation within the buffer B_1 may not be sufficient for the application. Thus, one has to continue constructing SWPM buffers $B_2, B_3, \ldots, B_z$ until the required mesh complexity is achieved. Our main interest is estimating the parameter

$$\beta := \sum_{i=0}^{z} |B_i|,$$

which determines the amount of memory required for storing these buffers. The approximation $M^{(n)}$ serves as an initial mesh when we construct the buffer B_1. Let $M^{(m)} = (V^{(m)}, T^{(m)})$ be the simplest approximation of the buffer B_1. Following the same reasoning, which is used to obtain (8), we have

$$\frac{1}{3}|T^{(n)}| \leqslant |T^{(m)}| < \frac{5}{6}|T^{(n)}|.$$

In order to obtain the upper bound for $|T^{(m)}|$, for $|T^{(n)}|$ we take the upper bound provided by (8). Then

$$|T^{(m)}| < \frac{5}{6}|T^{(n)}| < \left(\frac{5}{6}\right)^2 |T^{(0)}|,$$

so

$$|B_1| = |T^{(n)}| + |T^{(m)}| < \frac{5}{6}|T^{(0)}| + \left(\frac{5}{6}\right)^2 |T^{(0)}|.$$

Generally,

$$|B_i| < \left(\frac{5}{6}\right)^i |T^{(0)}| + \left(\frac{5}{6}\right)^{i+1} |T^{(0)}|.$$

By summing up the inequalities for every $|B_i|$, we obtain the upper bound for β

$$\beta < |T^{(0)}| + 2\left(\frac{5}{6}\right)|T^{(0)}| + \cdots + 2\left(\frac{5}{6}\right)^{z-1} |T^{(0)}| + \left(\frac{5}{6}\right)^{z} |T^{(0)}|.$$

We can interpret the right hand side as a converging geometric series. This is justified since in practice z is relatively big. Thus, we conclude that

$$\beta < 11|T^{(0)}|.$$

It is possible to derive the lower bound for β using a similar reasoning. Finally, we arrive at the following inequalities

$$2|T^{(0)}| \leqslant \beta < 11|T^{(0)}|. \tag{9}$$

It is natural to express β as a size of initial mesh triangulation multiplied by a coefficient

$$\beta = |T^{(0)}|\lambda,$$

where $\lambda \in [2,11)$ according to (9). We derive the bounds for the coefficient λ under relatively general assumptions about the input mesh. However, memory-limited applications require more precise estimation of λ for specific meshes. Thus, we have to derive an estimate for λ that exploits properties of a particular mesh.

Degree of a mesh vertex is the number of triangles incident to this vertex. Let $\overline{d}$ be a mean degree of the mesh vertices of *maximum independent set*. The value of $\overline{d}$ depends mainly on the structure of triangulation. In what follows we assume that

1. the choice of simplification operators and their parameters preserves the structure of initial triangulation, so $\overline{d}$ is approximately the same for any approximation $M^{(i)}$ in (2);

2. the size of a patch is approximately equal to the degree of its inner vertex, thus $\overline{d}$ is a mean patch size.

Under these assumptions, the number of patches is a ratio of total number of triangles to $\overline{d}$. Applying this relation to (5), it is easy to show that

$$|T^{(n)}| = |T^{(0)}|(1-2/\overline{d}),$$

$$|B_0| = |T^{(0)}| + |T^{(n)}| = |T^{(0)}|(2-2/\overline{d}),$$

$$|B_i| = |T^{(0)}|(2-2/\overline{d})(1-2/\overline{d})^i.$$

Again, β is expressed as a converging geometric series

$$\beta = |T^{(0)}|(\overline{d}-1).$$

Thus, the size of resulting dataset is proportional to $\overline{d}$, which is a mesh-dependent parameter. In some cases one may alter the structure of initial mesh triangulation in order to minimize $\overline{d}$. We investigate one possible re-meshing strategy in [Tur07]. Additionally, in [Tur07] we discuss optimization at the level of data structures.

1.1.6 SWPM and vertex caching

The main drawback of the original algorithm described in [For01] was a poor usage of the vertex cache. Forsyth represents each patch as a star of its inner vertex, i.e. $P \equiv S$. Let R be a set of the triangles, which are not included into such set of patches

$$R := T \setminus \bigcup_{v_i \in U} S(v_i),$$

where T is the set of all triangles, and U is a set of inner vertices. The set R is usually not empty. In order to satisfy the condition (4), Forsyth constructs the set of patches as follows: $P_1 := R \cup S(v_1)$, $P_i := S(v_i)$, where $i = 2 \ldots |U|$. As a result, the patch P_1 usually includes a large number of topologically disconnected triangles. This makes vertex caching inefficient.

We use a different strategy to build the patches. First, we initialize the list of patches; initially, each patch is a star of its inner vertex. Then, for each triangle of $\triangle_i \in R$, we select the patch $P_j \in \mathbf{P}$ that share the maximum number of vertices with $\triangle_i$; then we add $\triangle_i$ to P_j. Resulting patches mainly consist of topologically connected triangles.

Forsyth defines the order of the patches in the list $\mathbf{P}$ according to the error caused by the simplification of a patch (the patches simplified with the smallest error are at placed at the beginning of the list). However, such strategy is hardly practical. In our work [KT04b] we define the order of patches in $\mathbf{P}$ in such a manner that maximizes efficiency of the vertex caching. Formally, the problem of defining such an order may be cast out as a *minimum linear arrangement* problem (MLA) on a hypergraph (V,E). A vertex $v_i \in V$ identifies a patch $P_i \in \mathbf{P}$, and a hyperedge $e_j = (v_a, v_b, \ldots, v_c) \in E$ connects the patches that share a vertex. Let φ be a mapping $\varphi : V \to \{1,2,\ldots,|V|\}$. The length of a hyperedge is defined as

$$|e_j|_\varphi = \max(\varphi(v_a), \varphi(v_b), \ldots, \varphi(v_c)) - \min(\varphi(v_a), \varphi(v_b), \ldots, \varphi(v_c)).$$

We look for a mapping φ_p that solves the following problem

$$\sum_{e_i \in E} |e_i|_{\varphi_p} = \min_{\varphi_j} \sum_{e_i \in E} |e_i|_{\varphi_j}.$$

The mapping φ_p determines the position of each patch in the list $\mathbf{P}$.

Typically, MLA is used to obtain *cache oblivious* order of patches, which means the order of patches is determined under quite general assumptions about the underlying hardware. It is a matter of common knowledge (see e.g. [BG01, YL06]) that the algorithms, which take into account specific cache parameters (e.g. cache size), are often more efficient. We discuss one such heuristics-based algorithm in [KT04b]. The use of the latter algorithm improves vertex cache utilization nearly by a factor of two (comparing to the original algorithm [For01]).

1.2 Mesh generation in adaptive numerical methods

Efficient mesh generation procedures are of high demand in modern applications that use computer simulation of various applied problems. At present, it is a matter of common

knowledge that successful approximations of stationary and evolutionary problems should be based not on a single computation performed on a certain a priori given mesh, but on a sequence of consequently refined meshes adapted to specific features of a particular solution.

It is clear that the efficient creation of a finer mesh M_h (h is a *discretization step*) with help of an approximate solution computed on a coarser mesh M_H (such that $H > h$) requires an *error indicator* able to show where the errors are high and where they are relatively small.

The main purpose of *non-qualified estimates* is to prove that the difference between an exact solution u and an approximation u_h found in a finite-dimensional subspace V_h tends to zero as $h \to 0$ (see e.g. [Cia78, SF73]). Derivation of these estimates was an important step in the numerical analysis of a problem since it showed that the used approximation method is theoretically correct.

The estimates of another type are derived in the framework of the a priori error estimation approach show the convergence rate. They have the general form

$$\|u - u_h\| \leqslant Ch^k, \tag{10}$$

where C is a positive constant independent of h. Certainly, the constant C depends on the solution u and structure of the approximations used. Estimates of the type (10) are often called *qualified asymptotic estimates*. They show rates of convergence for the whole set of approximate solutions computed with help of approximations of a particular type. Unfortunately, such estimates are unable to reliably evaluate the error bound for a concrete approximate solution. Moreover, a priori error estimates are applicable only to *Galerkin approximations* and require *extra regularity* of the exact solutions. In many practically important cases, such as for the problems in geometrically non-trivial domains with non-smooth boundaries, it is impossible to guarantee such an extra regularity.

These reasons have led to the development of *a posteriori error estimates* that could explicitly characterize the accuracy of approximate solutions. Nowadays, a posteriori error estimates are an important part the modern numerical analysis.

1.2.1 Error estimates in terms of linear functionals

Typically, global error estimates provide a general idea about the quality of an approximate solution, and a stopping criteria for adaptive methods. However, from the viewpoint of engineering purposes, this is often insufficient. In many cases, analysts are highly interested in the errors over certain subdomains, lines, or at special points. One way to estimate such errors is to introduce a linear functional l, which is associated with a quantity of interest. The error is estimated as the linear functional $\langle l, u - \widetilde{u} \rangle$, where u is the exact solution and $\widetilde{u}$ is the approximate one. A linear functional associated with a subdomain ω is defined as

$$\langle l, u - \widetilde{u} \rangle = \int_\Omega \ell_\omega (u - \widetilde{u}) dx, \tag{11}$$

where a function ℓ is often chosen as

$$\ell_\omega(x) = \begin{cases} 1, & \text{if } x \in \omega \\ 0, & \text{otherwise.} \end{cases}$$

Many methods of a posteriori error control in terms of goal-oriented quantities (see e.g. [AO00, BR96, OP01]) find estimates of $\langle l, u-u_h\rangle$, where u_h is a Galerkin approximation of the problem considered, by employing an additional *adjoint* problem, whose right-hand side is formed by the functional l. Having the Galerkin approximation of the adjoint problem computed on the same mesh as the used for u_h, one can express the functional $\langle l, u-u_h\rangle$ via a certain integral functional that can be estimated by using, for example, *equilibrated residual method* (see e.g. [AO00, OP01]).

1.2.2 Error estimation by gradient averaging

In this section we describe the main idea of the method proposed in the paper [KNR03]. Let Ω be a bounded and connected domain $\Omega \subset \mathbb{R}^2$ with a Lipschitz continuous boundary $\partial\Omega$. Consider the following problem

$$\nabla\cdot A\nabla u_f + f = 0 \quad \text{in} \quad \Omega,,$$

$$u_f = 0 \quad \text{on} \quad \partial\Omega,$$

where A is a positive definite real matrix, and $f \in L^2(\Omega)$ is a given function. The respective generalized solution u_f satisfies the integral identity

$$\int_\Omega (A\nabla u_f \cdot \nabla w - fw)dx = 0 \quad \forall w \in V_0 \equiv H_0^1(\Omega),$$

Let $u_{fh} \in V_h$ be an approximate solution in the finite element space $V_h \subset V_0$ defined on the simplicial mesh $M_h = (V_h, T_h)$. We need to estimate the quantity $\langle l, u_f - u_{fh}\rangle$, where the functional l is in the space $V_0^\star$ topologically dual to V_0. Define u_ℓ by the relation

$$\int_\Omega A^\star\nabla u_\ell \cdot \nabla w dx = \langle l, w\rangle \quad \forall w \in V_0,$$

where $A^\star$ is the matrix adjoint to A. Then,

$$\langle l, u_f - u_{fh}\rangle = \int_\Omega A^\star\nabla u_\ell \cdot \nabla(u_f - u_{fh})dx = \\ = \int_\Omega (f\cdot u_\ell - A\nabla u_{fh}\cdot\nabla u_\ell)dx = E(u_\ell, u_{fh}). \quad (12)$$

Thus, $\langle l, u_f - u_{fh}\rangle$ can be easily estimated provided that u_ℓ is defined. Certainly, in practice u_ℓ is replaced with an approximation $u_{\ell\tau} \in V_\tau$ in the finite element space V_τ defined on *adjoint mesh* $M_\tau = (V_\tau, T_\tau)$, which is may not coincide with M_h. If $u_{\ell\tau}$ is a sharp approximation of u_ℓ, then the quantity $E(u_{\ell\tau}, u_{fh})$ could be a good indicator of the error. To obtain a sharper estimator, we rewrite (12) in the form

$$\langle l, u_f - u_{fh}\rangle = E(u_\ell, u_{fh}) = E_0(u_{fh}, u_{\ell\tau}) + E_1(u_f, u_{fh}, u_\ell, u_{\ell\tau}), \quad (13)$$

where

$$E_0(u_{fh}, u_{\ell\tau}) := \int_\Omega (fu_{\ell\tau} - A\nabla u_{fh}\cdot\nabla u_{\ell\tau})dx$$

is a directly computable functional, and

$$E_1(u_f, u_{fh}, u_\ell, u_{\ell\tau}) := \int_\Omega A(\nabla u_f - \nabla u_{fh}) \cdot (\nabla u_\ell - \nabla u_{\ell\tau}) dx.$$

In [KNR03, NR04], it was suggested to replace unknown functions ∇u_f and ∇u_ℓ with the post-processed gradients $\mathscr{G}_h \nabla u_{fh}$ and $\mathscr{G}_\tau \nabla u_{\ell\tau}$, where

$$\mathscr{G}_h : \left(L^2(\Omega)\right)^2 \to (V_h)^2 \quad \text{and} \quad \mathscr{G}_\tau : \left(L^2(\Omega)\right)^2 \to (V_\tau)^2$$

are the averaging operators that map gradients to the respective finite element spaces. It is proved that under the standard assumptions that guarantee superconvergence of the primal and adjoint approximations such a replacement leads to a higher order error. Thus, we use the computable quantity

$$\widetilde{E}_1(u_{fh}, u_{\ell\tau}) := \int_\Omega A(\mathscr{G}_h \nabla u_{fh} - \nabla u_{fh}) \cdot (\mathscr{G}_\tau \nabla u_{\ell\tau} - \nabla u_{\ell\tau}) dx$$

instead of E_1. It can be proved (see the above cited papers) that under the standard assumptions on the regularity of u_f and u_ℓ and the respective meshes M_h and T_ℓ that guarantee superconvergence in *both* primal and adjoint problems the replacement of the exact estimator (13) with the approximate estimator

$$\widetilde{E}(u_{fh}, u_{\ell\tau}) = E_0(u_{fh}, u_{\ell\tau}) + \widetilde{E}_1(u_{fh}, u_{\ell\tau}) \tag{14}$$

leads the error that is of higher order with respect to the error we measure.

It is important to outline that the efficiency of the discussed estimator depends on the mesh used to solve the adjoint problem. The use of the same mesh for both primal and adjoint problems is not efficient. The number of degrees of freedom in the adjoint problem should be minimal in order to minimize the cost of solving the adjoint problem and, thus, the overall cost of the error estimation. If the functional l is a locally supported integral functional, such as (11), then the adjoint solution may have complicated behavior only in the support area. Thus, it is worth putting more degrees of freedom in the support area, whereas other mesh regions may be coarser.

Moreover, the use of identical M_h and M_τ meshes implies worse accuracy of the error estimation. The term E_0 in (14) vanishes on coinciding meshes (due to Galerkin orthogonality condition), so the error is expressed solely via the approximated term $\widetilde{E}_1$.

Thus, a success in the error estimation method discussed essentially depends on the structure of the primal and adjoint meshes. In principle, the requirements we state for an optimal (or quasi-optimal) mesh are the same as in computer graphics: a mesh must be valid for representation (of a solution in scientific computing, or of a shape in graphics) with sufficient accuracy and must contain minimal number of nodes. The major difference is that the goal functional, which is used to evaluate mesh feasibility, is unknown for computational meshes since, in principle, this functional includes a priori unknown solution of the corresponding adjoint problem. Therefore, one has to construct a quasi-optimal mesh in the process of step-by-step clarification of this solution.

It was verified (see [KNR03, KT04a, NRT07]) that the estimator (14) provides a good indication of the goal-oriented error provided that the above formulated conditions

for the adjoint mesh are indeed satisfied. However, these conditions also raise concerns about the regularity (quasi-uniformity) of the mesh used in the adjoint problem. Typically, the solutions of the primal problem are obtained using regular (or almost regular) meshes where gradient averaging methods work stably and accurately. However, attempts to construct an adapted mesh for the adjoint problem may be problematic since relatively coarse meshes are usually strongly irregular. In the latter case it is impossible to guarantee superconvergence in the adjoint problem and, therefore, justify the error estimation approach in use.

1.2.3 New modus operandi

In our work we have developed a new *modus operandi* that relieves the constrains on the structure of adjoint mesh. We present the exact estimator (13) in a different form and, as a result, suggest another way of its practical computation. This method is explained in details in ([NRT07]). We explain the main idea of it below. We denote

$$\sigma_\ell := A^\star \nabla u_\ell \quad \text{and} \quad \sigma_{\ell\tau} := A^\star \nabla u_{\ell\tau}.$$

Then

$$E_1(u_f,u_{fh},u_\ell,u_{\ell\tau}) := \sum_{\triangle_i\in T_\tau}\int_{\triangle_i}(\nabla u_f-\nabla u_{fh})\cdot(\sigma_\ell-\sigma_{\ell\tau})dx =$$

$$= \sum_{\triangle_i\in T_\tau}\left(\int_{\triangle_i}(u_{fh}-u_f)(\ell+\nabla\cdot\sigma_{\ell\tau})dx+\int_{\partial\triangle_i}(u_f-u_{fh})(\sigma_\ell-\sigma_{\ell\tau})\cdot\nu ds\right),$$

where ν is a unit outward normal. It is easy to show that

$$\sum_{\triangle_i\in T_\tau}\int_{\partial\triangle_i}(u_f-u_{fh})(\sigma_\ell-\sigma_{\ell\tau})\cdot\nu ds = \sum_{e_j\in\mathbf{E}_\tau}\int_{e_j}(u_{fh}-u_f)[\sigma_{\ell\tau}\cdot\nu_{e_j}]_{e_j}ds, \qquad (15)$$

where $\mathbf{E}_\tau$ is the set of all edges in the adjoint mesh, ν_e is a unit outward normal to the edge e, and $[\cdot]_e$ is the *jump* of a quantity across the edge e

$$[g]_e := \sum_{\triangle_i\ni e} g|_{\triangle_i}.$$

Hence, we observe that

$$E_1(u_f,u_{fh},u_\ell,u_{\ell\tau}) = E_2(u_f,u_{fh},u_{\ell\tau}) + E_3(u_f,u_{fh},u_{\ell\tau}),$$

where

$$E_2(u_f,u_{fh},u_{\ell\tau}) := \sum_{\triangle_i\in T_\tau}\int_{\triangle_i}(u_f-u_{fh})r(\sigma_{\ell\tau})dx,$$

$$E_3(u_f,u_{fh},u_{\ell\tau}) := \sum_{e_j\in\mathbf{E}_\tau}\int_{e_j}(u_{fh}-u_f)[\sigma_{\ell\tau}\cdot\nu_{e_j}]_{e_j}ds,$$

and $r(\sigma_{\ell\tau}) := \ell+\nabla\cdot\sigma_{\ell\tau}$ is the residual of the equation related to the adjoint problem.

We stress that in this form the exact solution of the adjoint problem is *completely excluded* from the goal-oriented error estimator. Thus, in order to justify the estimator, we do not need to attract gradient superconvergence of the adjoint problem.

We still use superconvergence in order to replace the unknown solution of the primal problem with explicitly computable quantity. Here we apply a post-processing method to the solution u_{fh} itself (not to the solution gradient $A\nabla u_{fh}$). At this point we refer to the work of Wang [Wan00] who investigated such type of post-processing methods. The main result of [Wan00] is described below. Let V_μ be a finite element space consisting of piecewise polynomials of degree $r \geqslant 0$ over the mesh T_μ defined on $\Omega_0 \subseteq \Omega$ with mesh-size μ. Denote the L^2 projection operator $\mathscr{S}_\mu : L^2(\Omega_0) \to V_\mu$. It can be proved that

$$\|u_f - \mathscr{S}_\mu u_{fh}\|_{0,\Omega_0} \leqslant C h^{\beta(h,\mu,r,k)} \left(\|u_f\|_{r+1,\Omega_0} + \|u_f\|_{k+1,\Omega} \right), \tag{16}$$

where the exact solution $u_f \in H^{k+1}(\Omega) \cap H^{r+1}(\Omega_0) \cap V_0$ is H^s-regular with $1 \leqslant s \leqslant k+1$. The rate is $\beta > 2$ when u_f is regular enough, and V_μ is selected appropriately.

By the trace theorem, we can prove that the post-processed trace constructed by averaging of the traces of $\mathscr{S}_\mu u_{fh}$ on both sides of the inter-element boundary is also superconvergent provided that the meshes for the primal problem do not degenerate (in the sense of elements aspect ratio) in the process of mesh refinement.

Now, we replace E_2 and E_3 with approximations $\widetilde{E}_2$ and $\widetilde{E}_3$, respectively, where

$$\widetilde{E}_2(u_{fh}, u_{\ell\tau}) := \sum_{\triangle_i \in T_\tau} \int_{\triangle_i} (u_{fh} - \mathscr{S}_\tau u_{fh}) r(\sigma_{\ell\tau}) dx,$$

$$\widetilde{E}_3(u_{fh}, u_{\ell\tau}) := \sum_{e_j \in \mathbf{E}_\tau} \int_{e_j} (\mathscr{S}_\tau u_{fh} - u_{fh}) [\sigma_{\ell\tau} \cdot \nu_{e_j}]_{e_j} ds.$$

Thus, we obtain a computable estimator

$$\widehat{E}(u_{fh}, u_{\ell\tau}) = E_0(u_{fh}, u_{\ell\tau}) + \widetilde{E}_2(u_{fh}, u_{\ell\tau}) + \widetilde{E}_3(u_{fh}, u_{\ell\tau}). \tag{17}$$

Unlike the estimator (14), the estimator (17) neither exploits superconvergence in the adjoint problem, nor needs superconvergence of the fluxes in the primal problem. The only required post-processing procedure is related to the smoothing of the approximate solution u_{fh}, which is computed on a sufficiently regular mesh M_h. In the present research, we performed numerical experiments that have confirmed the efficiency of the estimator (17) and in many cases demonstrated its advantage with respect to the estimator (14). These results are shown in the paper [NRT07].

Moreover, it was discovered that the quality of the error estimation can be significantly improved if we apply a certain post-processing procedure to the function $u_{\ell\tau}$. The operator $\mathscr{C}_\tau$ constructs a C^1-interpolant of the finite element solution $u_{\ell\tau}$, then we use $\sigma^C_{\ell\tau} := A^\star \nabla \mathscr{C}_\tau u_{\ell\tau}$ instead of $\sigma_{\ell\tau}$. We construct the interpolant using Hsieh-Clough-Tocher triangles, which require (per triangle) the nodal values, the nodal derivatives, and the normal derivatives at the midpoints. For nodal values we take the nodal values of the finite element solution. It is known that nodal values possess certain superconvergent properties (see e.g. [BS01]). We choose remaining derivative-related degrees of freedom in such a

manner that the norm of residual $\|r(\sigma^C_{\ell\tau})\|_\Omega$ is minimized. The aim of the procedure described is two-fold. First, we construct a smooth function $\mathscr{C}_\tau u_{\ell\tau}$, so $[\sigma^C_{\ell\tau}\cdot\nu_e]_e = 0$ on all the edges, and the term $\widetilde{E}_3$ of the estimator vanishes. Second, minimization of the residual should improve the quality of the estimator since the error in estimator, which arises due to the replacement of u_f by $\mathscr{S}_\mu u_{fh}$ on each element, is equal to

$$\int_{\triangle_i}(u_f - \mathscr{S}_\mu u_{fh})r(\sigma_{\ell\tau})dx.$$

Thus, the third form of the goal oriented error estimator is

$$\widehat{E}(u_{fh},u_{\ell\tau}) = E_0(u_{fh},\mathscr{C}_\tau u_{\ell\tau}) + \widetilde{E}_2(u_{fh},\mathscr{C}_\tau u_{\ell\tau}). \tag{18}$$

The results of numerical experiments in [NRT07] confirm high efficiency of the estimator (18).

We have described the idea of a new goal-oriented error estimation method for a case of the diffusion equation. However, it is equally applicable to other linear elliptic problems. For instance, in [NRT07] we apply the estimators $\widetilde{E}$ and $\widehat{E}$ for the linear elasticity problem in the forms. In the latter case σ is the stress field, A is the tensor of elastic coefficients and u is the displacement vector.

1.3 Conclusions

Our work makes several contributions in computer graphics and adaptive numerical methods.

We have addressed two major problems of the sliding window progressive meshes described in Sec. 1.1.4–1.1.6. Namely, we have significantly improved vertex cache usage in [KT04b], and reduced the size of the resulting datasets [Tur07].

We have tested the estimator (14) for the cases of three-dimensional Poisson equation [KT04a] and Lamé equation [NRT07]. New versions (17) and (18) of the estimator were proposed and tested in [NRT07]. The methods developed in the framework of the thesis can be used in various applications that provide reliable modelling of real life objects with help of a posteriori error estimates and mesh adaptive procedures, such as [INKT03, AHS*04].

REFERENCES

[AHS*04] ARKHIPOV B., HUTTULA T., SOLBALKOV V., LINDFORS A., SEPPÄNEN J., KANGAS A., KOROTOV S., SALONEN K.: Experimental and numerical investigation of circulation and thermal structure in lake jyväsjärvi: Short-term variability. In *Proceedings of ECCOMAS'04* (2004), Neittaanmäki P., Rossi T., Korotov S., Onate E., Periaux J.,, Knörzer D., (Eds.).

[AO00] AINSWORTH M., ODEN J. T.: *A posteriori error estimation in finite element analysis*. Wiley and Sons, 2000.

[BDE96] BAREQUET G., DICKERSON M., EPPSTEIN D.: On triangulating three-dimensional polygons. In *SCG '96: Proceedings of the twelfth annual symposium on Computational geometry* (1996), ACM Press, pp. 38–47.

[BG01] BOGOMJAKOV A., GOTSMAN C.: Universal rendering sequences for transparent vertex caching of progressive meshes. In *Proceedings of Graphics Interface 2001* (2001), Watson B., Buchanan J. W., (Eds.), pp. 81–90.

[BR96] BECKER R., RANNACHER R.: A feed–back approach to error control in finite element methods: Basic approach and examples. *East–West J. Numer. Math. 4* (1996), 237–264.

[BS96] BAJAJ C., SCHIKORE D.: Error-bounded reduction of triangle meshes with multivariate data. In *Proc. SPIE* (1996), Grinstein G., Erbacher R., (Eds.), vol. 2656, pp. 34–45.

[BS01] BABUŠKA I., STROUBOULIS T.: *The finite element method and its reliability*. The Clarendon Press, Oxford University Press, 2001.

[Cia78] CIARLET P. G.: *The finite element method for elliptic problems*. North Holland, New York, Oxford, 1978.

[CRS98] CIGNONI P., ROCCHINI C., SCOPIGNO R.: Metro: Measuring error on simplified surfaces. *Computer Graphics Forum 17*, 2 (1998), 167–174.

[ESAV99] EL-SANA J., AZANLI E., VARSHNEY A.: Skip strips: maintaining triangle strips for view-dependent rendering. In *Proceedings of the conference on Visualization '99* (1999), IEEE Computer Society Press, pp. 131–138.

[For01] FORSYTH T.: Comparison of vipm methods. In *Game Programming Gems 2* (2001), DeLoura M., (Ed.), Charles River Media, pp. 363–376.

[GH98] GARLAND M., HECKBERT P. S.: Simplifying surfaces with color and texture using quadric error metrics. In *IEEE Visualization '98* (1998), Ebert D., Hagen H.,, Rushmeier H., (Eds.), pp. 263–270.

[Hop96] HOPPE H.: Progressive meshes. *Computer Graphics 30*, Annual Conference Series (1996), 99–108.

[Hop99a] HOPPE H.: New quadric metric for simplifiying meshes with appearance attributes. In *Proceedings of the conference on Visualization '99* (San Francisco, California, 1999), pp. 59–66.

[Hop99b] HOPPE H.: Optimization of mesh locality for transparent vertex caching. In *Siggraph 1999, Computer Graphics Proceedings* (Los Angeles, 1999), Rockwood A., (Ed.), Addison Wesley Longman, pp. 269–276.

[INKT03] IVANENKO S. A., NEITTAANMÄKI P., KOROTOV S., TURCHYN P.: Shallow water model of the lake jyväsjärvi. *University of Jyväskylä, Preprint B3/2003* (2003).

[KNR03] KOROTOV S., NEITTAANMAKI P., REPIN S.: A posteriori error estimation of goal-oriented quantities by the superconvergent patch recovery. *J. Numer. Math 11* (2003), 33–59.

[KT04a] KOROTOV S., TURCHYN P.: A posteriori error estimation of quantities of interest on tetrahedral meshes. In *Proceedings of ECCOMAS'04* (2004), Neittaanmäki P., Rossi T., Korotov S., Onate E., Periaux J.,, Knörzer D., (Eds.).

[KT04b] KOROTOV S., TURCHYN P.: Topology-driven progressive mesh construction for hardware-accelerated rendering. *Computer Graphics and Geometry (Internet) 6*, 3 (2004), 100–119.

[LH00] LINDSTROM P., HECKBERT G.: Image-driven simplification. *ACM Transations on Graphics 19*, 3 (2000), 204–241.

[LRC*02] LUEBKE D., REDDY M., COHEN J., VARSHNEY A., WATSON B., HUEBNER R.: *Level of Detail for 3D Graphics*. Computer Graphics and Geometric Modeling. Morgan Kaufmann, 2002.

[LT99] LINDSTROM P., TURK G.: Evaluation of memoryless simplification. *IEEE Transactions on Visualization and Computer Graphics 5*, 2 (1999), 98–115.

[NR04] NEITTAANMÄKI P., REPIN S.: *Reliable methods for computer simulation. Error control and a posteriori estimates*. Elsevier, New York, London, 2004.

[NRT07] NEITTAANMÄKI P., REPIN S., TURCHYN P.: A posteriori error estimation in terms of linear functionals for the linear elasticity problems. *Russian J. Numer. Anal. Math. Modelling (submitted)* (2007).

[OP01] ODEN J. T., PRUDHOMME S.: Goal-oriented error estimation and adaptivity for the finite element method. *Comput. Math. Appl. 41* (2001), 735–756.

[SF73] STRANG G., FIX G.: *An analysis of the finite element method*. Prentice Hall, Englewood Cliffs, 1973.

[SP03] SHAFAE M., PAJAROLA R.: Dstrips: Dynamic triangle strips for real-time mesh simplification and rendering. In *Proceedings Pacific Graphics Conference* (2003).

[Ste01] STEWART A. J.: Tunneling for triangle strips in continuous level-of-detail meshes. In *Proceedings of Graphics Interface* (2001), Watson B., Buchanan J. W., (Eds.), pp. 91–100.

[SvK97] SNOEYINK J., VAN KREVELD M. J.: Linear-time reconstruction of delaunay triangulations with applications. In *ESA '97: Proceedings of the 5th Annual European Symposium on Algorithms* (London, UK, 1997), Springer-Verlag, pp. 459–471.

[Tur07] TURCHYN P.: Memory-efficient sliding window progressive meshes. In *Proceeding WSCG'07 (submitted)* (2007).

[Wan00] WANG J.: A superconvergence analysis for finite element solutions by the least-squares surface fitting on irregular meshes for smooth problems. *J. Math. Study 33*, 3 (2000), 229–243.

[YL06] YOON S.-E., LINDSTROM P.: Mesh layouts for block-based caches. *IEEE Transactions on Visualization and Computer Graphics 12* (2006), 1213–1220.

YHTEENVETO (FINNISH SUMMARY)

Tässä väitöskirjassa esitetään parannuksia verkongenerointialgoritmeihin, joita hyödynnetään tietokonegrafiikassa ja elliptisten reuna-arvotehtävien ratkaisussa.

Väitöskirjan ensimmäinen osa käsittelee esineiden mallintamiseen käytettävien verkkojen luomista tietokonegrafiikassa. Työn erityinen kontribuutio liittyy ns. liukuvan ikkunan progressiiviseen verkkoalgoritmiin (*engl. sliding window progressive meshes*), jonka suurimpien ongelmakohtien ratkaisemiseksi esitetään useita parannelmia. Ensimmäinen ongelma on välimuistin tehokas käyttö (*engl. cache-coherency*) verkon kärkipisteiden käsittelyssä. Tämä ratkaistann heuristiikkaan perustuvalla verkon kolmioiden uudelleenjärjestämisellä. Toinen ongelma on tarvittavien datajoukkojen valtava koko. Väitöskirjassa osoitetaan, että datajoukkoja voidaan pienentää optimoimalla verkon konnektiivisuutta käyttämällä hierarkkisia tietorakenteita ja erityisiä operaattoreita, jotka poistavat kerralla useita kärkipisteitä.

Väitöskirjan toinen osa keskittyy elementtimenetelmäratkaisuiden *a posteriori* virhearviointiin. Verkon luonti liittyy tähän kahdella tapaa: Ensinnäkin estimaattorin antama virhearvio antaa vihjeitä strategiasta verkon parantamiseen. Toisekseen estimaattori tarvitsee ns. liitto-ongelman ratkaisemista eri verkolle kuin päätehtävässä käytetty. Liittoverkon vaatimusten lieventämiseksi väitöskirjassa kehitetään ja testataan uusi menetelmä, joka hyväksyy anisotrooppiset liittoverkot eikä myöskään vaadi liitto-ongelman ratkaisulta ylimääräistä säännöllisyyttä.