High-performance image reconstruction in fluorescence tomography on desktop computers and graphics hardware

M. Freiberger H. Egger M. Liebmann

H. Scharfetter

SFB-Report No. 2010–026 July 2010

Supported by the
Austrian Science Fund (FWF)

SFB sponsors:

- **Austrian Science Fund** (FWF)
- **University of Graz**
- **Graz University of Technology**
- **Medical University of Graz**
- **Government of Styria**
- **City of Graz**

High-performance image reconstruction in fluorescence tomography on desktop computers and graphics hardware

Manuel Freiberger[a], Herbert Egger[b], Manfred Liebmann[b] and Hermann Scharfetter[a]

[a]Graz University of Technology, Institute of Medical Engineering, Kronesgasse 5/II, 8010 Graz, Austria

[b]University of Graz, Institute for Mathematics and Scientific Computing, Heinrichstr. 36/III, 8010 Graz, Austria

ABSTRACT

Image reconstruction in fluorescence optical tomography is a three-dimensional nonlinear ill-posed problem governed by a system of partial differential equations. In this paper we demonstrate that a combination of state of the art numerical algorithms and a careful hardware optimized implementation allows to solve this large-scale inverse problem in a few seconds on standard desktop PCs with modern graphics hardware. A comparison of optimized CPU and GPU implementations shows that the reconstruction can be accelerated by factors of about 15 through the use of the graphics hardware without compromising the accuracy in the reconstructed images.

Keywords: fluorescence tomography, inverse problems, finite element methods, image reconstruction, GPU-programming

Further author information: (Send correspondence to Manuel Freiberger)
Manuel Freiberger: E-mail: manuel.freiberger@tugraz.at
Herbert Egger: E-mail: herbert.egger@uni-graz.at
Manfred Liebmann: E-mail: manfred.liebmann@uni-graz.at
Hermann Scharfetter: E-mail: hermann.scharfetter@tugraz.at

1. INTRODUCTION

The rapid hardware evolution of the last 10 years brought small super computers to everyone's desktop. In fact, with a theoretical peak performance of almost 40 Gflops, a standard consumer market CPU, like the Intel Core 2 Q6600 running at 2.4 GHz, would have been listed under the top 30 supercomputers of the year 1995, and would still almost make it into the Top500 list of the year 2000. A standard desktop computer with such a multicore CPU and 4-8 GByte main memory can therefore serve as a workbench for medium to large scale scientific computing problems. Following a recent trend in computer science, a further substantial increase in performance can be obtained by utilizing the tremendous compute power of modern graphics hardware: NVIDIA's GeForce 285GTX for instance peaks at 1063 Gflops, which tops the performance of modern CPUs by a factor of more than 20.

The ability to run highly complex simulation tasks even on cheap commodity hardware has a strong impact on the development and feasibility of new technologies, e.g., it is common practice in engineering today, to design, test, and optimize components or even full devices by simulation rather than manufacturing a variety of test samples. Computer simulation of physical processes also allows to utilize new measurement modalities, i.e., to extract information about inaccessible quantities from easily attainable outputs of physical systems.

In this paper we study such an indirect measurement technology, namely fluorescence optical tomography, which uses the ability of fluorescent dyes to absorb and re-emit light at different wavelengths. The aim of fluorescence tomography is to obtain information about the fluorophore distribution inside an object (not directly accessible quantity) from optical measurements at the boundary (easily obtainable information).

In contrast to other tomographic imaging modalities, like X-ray computed tomography or magnetic resonance imaging, fluorescence tomography relies on non-ionizing radiation and relatively cheap equipment. Another great advantage is the fact that the optical properties of certain fluorophores depend on physiologically interesting parameters like the pH value,[1,2] or tissue oxygenation.[3] This allows to obtain anatomical images and additional information about physiological states and processes at the same time.

The drawback of using low-energy radiation is that the image reconstruction in fluorescence tomography requires the accurate simulation of light propagation in highly scattering media and the solution of nonlinear inverse problems, which are computationally intensive tasks. Therefore, fast and reliable numerical algorithms will crucially determine the success of this new imaging modality in biomedical sciences as well as in engineering applications.

The aim of this paper is to demonstrate that the image reconstruction in fluorescence tomography is feasible in the sense that tomographic images—even of large three dimensional models—can be computed in a few seconds on standard computers without compromising accuracy. Our algorithms rely on state of the art numerical methods as well as on a hardware oriented implementation; in particular, we intend to make excessive use of graphics hardware for performining compute intensive tasks.

Let us shortly outline the contents of this paper: Section 2 provides background material about fluorescence tomography and describes the iterative regularization method that is used for the stable solution of the inverse problem. Section 3 then deals with the discretization of the problem by finite element methods and provides a high-level description of the reconstruction algorithm (a discretization of the regularization method). In Section 4, the numerical methods that are utilized in our simulations as well as their efficient implementation are discussed in detail. In Section 5, we define a test problem, and derive estimates for the complexity and required memory of the reconstruction algorithm. The theoretical results are confirmed by numerical tests, in which we compare the performance of our algorithms on CPU and GPU hardware. Our tests illustrate that the reconstruction times can be reduced significantly (by a factor of about 15) through the use of the graphics hardware.

Since our presentation touches various fields of research, the references to related work will be given in the individual sections rather than generating a heterogeneous overview here. Some concluding discussion of related work and our contributions is contained in the final section.

2. FLUORESCENCE TOMOGRAPHY

In the following, we introduce the mathematical models that are used to describe the light propagation in highly scattering media. Moreover, we present—on an abstract level—the iterative regularization method which are used for the stable solution of the inverse problem and which are thus the basis for the image reconstruction algorithms to be discussed in the following sections.

2.1 Mathematical model

The propagation of light in highly scattering media, e.g., in biological tissue, is typically modeled by the diffusion approximation,[4] which serves as a first order approximation to the radiative transport equation. Since light of two different wavelengths is employed in fluorescence tomography, the mathematical model consists of a coupled system of partial differential equations,[5] namely

$$-\mathrm{div}(\kappa_x \nabla \varphi_x) + \mu_x \varphi_x = q, \qquad \text{in } \Omega, \tag{1}$$

$$-\mathrm{div}(\kappa_m \nabla \varphi_m) + \mu_m \varphi_m = \gamma c \varphi_x, \qquad \text{in } \Omega. \tag{2}$$

Here, φ_i, $i = x, m$ are the photon densities for the excitation (x) and the emission (m) light in the domain Ω covered by tissue, and the source term q models a collimated light source. The parameters κ_i and μ_i are the diffusion and absorption coefficients of the tissue, and γ is a measure incorporating the fluorophore's extinction coefficient and its quantum yield. We assume that these parameters depend in a known form on the concentration c of fluorophore in the tissue, and we write $\kappa_i(c)$, $\mu_i(c)$ or $\gamma(c)$ if this dependence needs to be emphasized.

Homogeneous Robin boundary conditions

$$\varrho\varphi_i + \kappa_i \frac{\partial \varphi_i}{\partial n} = 0, \qquad \text{on } \partial\Omega, \qquad i = x, m.$$

are used to model that no light enters the domain from the outside apart from the light injected through the internal source q.

A boundary measurement of the photon density at emission wavelength is given by

$$m(\varphi_m) = \int_{\partial\Omega} d(s)\varphi_m(s)ds, \tag{3}$$

where the function d allows to model the transfer characteristics of the detector.

In short, the physical process can be described as follows: (i) The object is illuminated by a light source ($q \rightarrow \varphi_x$); (ii) a part of the excitation light is absorbed by the fluorophores and re-emitted at a longer wavelength ($\gamma c \varphi_x \rightarrow \varphi_m$); (iii) the emitted light propagates through the tissue and is recorded by a detector ($\varphi_m \rightarrow m_d(\varphi_m)$).

The tomographic measurement process finally consists of illuminating the object with a sequence of light sources q_j, $j = 1, \dots, ns$, and taking measurements of the emission light with an array of detectors d_i, $i = 1, \dots, nd$ at the boundary. This gives rise to a measurement matrix $\mathcal{M}$ of dimension $nd \times ns$, which obviously depends on the distribution c of the fluorophore, and we write $\mathcal{M}(c)$ to express this dependence.

2.2 The tomographic image reconstruction

The aim of fluorescence tomography is to determine the inaccessible fluorophore distribution c which gave rise to the measurements $\mathcal{M}^\delta$ of the emitted light. (We write $\mathcal{M}^\delta$ to indicate that the measurements of the light intensities are perturbed by measurement and model errors. The noise level δ serves as a measure for the magnitude of these errors). The reconstruction of the fluorophore distribution amounts to solving the nonlinear inverse problem

$$\mathcal{M}(c) = \mathcal{M}^\delta,$$

for whose fast and stable solution we intend to use iterative methods of Newton-type. Starting from an initial guess c_0, the classical Newton method defines a sequence of approximations for the solution by

$$S_k\ (c_{k+1} - c_k) = \mathcal{M}^\delta - \mathcal{M}(c_k), \qquad k = 0, 1, \dots$$

Here, $S_k := \frac{\partial \mathcal{M}}{\partial c}(c_k)$ denotes the sensitivity operator, which measures the effect of a change in the fluorophore distribution c onto the output $\mathcal{M}(c)$. If the number of unknowns (degrees of freedom used for discretizing c) does not match the number of measurements (which will be the case in general), the Newton step is not well-defined. In order to deal with this situation, and to address the ill-posedness of the inverse problem, we utilize the iteratively regularized Gauß-Newton method[6]

$$\begin{aligned} (S_k^* S_k + \alpha_k I)(c_{k+1} - c_k) & \qquad\qquad k = 0, 1, \dots \\ = S_k^*(\mathcal{M}^\delta - \mathcal{M}(c_k)) + \alpha_k(c_0 - c_k) & \end{aligned} \tag{4}$$

for a stable reconstruction. Note that for $\alpha_k \equiv 0$, this algorithms results from formally multiplying the Newton system by the adjoint sensitivity S_k^*. In order to deal with the ill-posedness of the problem, α_k should however be chosen atrictly greater than zero.

2.3 Remarks and relation to other work

Let us make a few remarks concerning the reconstruction process:

(i) After discretization, $\mathcal{M}(c)$ is just a vector valued function, and the sensitivity operator S_k is the Jacobian matrix of this nonlinear function computed at c_k.

(ii) Every evaluation of $\mathcal{M}(c)$ requires to solve (a discretization of) the coupled system of governing differential equations, so applying the forward operator is a computationally demanding step.

(iii) There are various ways for regularizing the Newton iteration, e.g., the truncated Newton-CG algorithm,[7] the Levenberg Marquardt method,[8] or the Newton-Landweber method.[9] Several of these approaches have been demonstrated to work well for tomographic applications.[5,10–13] Although we focus on the iteratively regularized Gauß-Newton in this presentation, other algorithms can be realized in a similar manner.

(iv) The choice of the norm influences the meaning of the * in the definition of the adjoint sensitivity. In the simplest case, the adjoint of the sensitivity is just the transpose of the corresponding matrix obtained by discretization of the sensitivity.

(v) Algorithm (4) is formulated for a linear regularization term, e.g., the L^2-norm. More general (nonlinear) regularization terms $\mathcal{R}(c)$ can be considered by replacing $\alpha_k I$ and $\alpha_k(c_0 - c_k)$ with appropriate terms $\alpha_k \mathcal{R}''(c_k)$ and $\alpha_k \mathcal{R}'(c_k)(c_0 - c_k)$, respectively.[14,15]

(vi) The choice of the regularization parameters α_k critically determines the quality of the reconstructions. E.g., choosing α_k very large would imply $c_{k+1} \sim c_0$; on the other hand, setting $\alpha_k \sim 0$ yields an unstable method. Reasonable strategies for the parameter choice will be discussed in section 3.4 below.

3. DISCRETIZATION

For the discretization of the governing partial differential equations, we consider a standard finite element method with piecewise linear basis functions for the photon fields. In our computations, we use an unstructured mesh with N_V vertices consisting of N_T tetrahedral elements. Such meshes can be generated for general geometries by freely available meshing software. For simplicity, the optical parameters and the fluorophore concentration c are discretized by piecewise constant functions over the same mesh.

3.1 The forward problem

Through discretization, the governing partial differential equations of the system (1)–(2) are turned into linear systems

$$[K_x(c) + M_x(c) + R_x]V_x = Q$$
$$[K_m(c) + M_m(c) + R_m]V_m = G(c)V_x.$$

Each column of the $N_V \times ns$ matrix Q corresponds to a different excitation q_j, $j = 1, \dots, ns$, and the ns columns of the solution matrices V_x and V_m store the vector representations of the corresponding photon density fields φ_x and φ_m. The assembly of the system matrices will be discussed in more detail below.

The measurement data are then given by $\mathcal{M}(c) = D^\top V_m$, where each of the nd columns of the detector matrix D corresponds to one of the detectors d_i, $i = 1, \dots, nd$ in (3).

The mapping $c \mapsto \mathcal{M}(c)$ defines a discretization of the forward operator, and its derivative $S := \frac{\partial \mathcal{M}(c)}{\partial c}$ is a discretization for the sensitivity operator. Note that we use the same symbols for the infinite dimensional and discretized operators.

3.2 Sensitivity and ajoint problems

The discrete sensitivity $S = S(c)$ is an $nd \times ns \times N_T$ tensor (respectively an $(nd \cdot ns) \times N_T$ matrix). Its action onto a parameter perturbation $h \in \mathbb{R}^{N_T}$ is given by

$$\begin{aligned} Sh := &-W_m^\top K_m(h)V_m - W_m^\top M_m(h)V_m \\ &- W_x^\top K_x(h)V_x - W_x^\top M_x(h)V_x + W_m^\top G(h)V_x, \end{aligned}$$

where h is an arbitrary element from the N_T dimensional parameter space, and W_m, W_x denote the solutions of the adjoint system

$$[K_m(c) + M_m(c) + R_m]W_m = D,$$
$$[K_x(c) + M_x(c) + R_x]W_x = G(c)W_m.$$

The interpretation of this system in physical terms is that (i) the domain is illuminated at the emission wavelength by sources placed at the detector locations; (ii) light is absorbed by the fluorophore and re-emitted at the excitation wavelength which then propagates through the domain. Note that due to the reciprocity principle, the order of the equations has changed.

3.3 Discrete reconstruction algorithm

The fully discrete equivalent of the iteratively regularized Gauß-Newton method (4) has the following form:

```
% Gauss-Newton algorithm
initialize;
for k=0:maxit-1
  [Ax,Am,G] = assembleSystemMatrices(ck);
  Vx = pbcgSolve(Ax,Q);
  Vm = pbcgSolve(Am,G*Vx);
  M  = D'*V_m;
  Wm = pbcgSolve(Am,D);
  Wx = pbcgSolve(Ax,G*Wm);
  Sk = assembleSensitivity(ck);
  ap = @(x) Sk'*Sk*x+ak*R*x;
  dc = pcgSolve(ap,Sk'*(Md-M));
```

```
    c  = c + dc;
  end
```

In the next section, we will discuss the individual steps of this reconstruction algorithm in more detail. Let us, however, mention already at this point, that the algorithm is of fully iterative nature, i.e., only matrix vector products are required.

3.4 Remarks and related work

(i) The finite element method provides a very general and flexible framework for the discretization of our problem. E.g. complex geometries and various kinds of boundary conditions can be considered consistently. Additionally, fast (multigrid) solvers for the resulting linear systems are available, and a-posteriori error estimators allow to acquire information about the accuracy of the simulations during the computation. The discrete reconstruction algorithm can be shown to be mesh-independent, i.e., the parameters α_k or the stopping index `maxit` can be chosen independently of the mesh size, and convergence of the iterates of the discrete algorithm to that of the continuous iteration (4) can be shown. To the best of our knowledge, such rigorous results are not valid for other discretization methods, e.g., the finite difference method.[16]

(ii) Our iterative method does not rely on the Born approximation[17, 18] or other model reductions. In fact, we utilize the full derivative of the (discretized) nonlinear forward operator. Further information on the iterative numerical algorithms for fluorescence optical tomography can be found in [5, 10, 11, 15].

(iii) Each entry S_{ijk} of the sensitivity can be computed in $O(1)$ operations, once the solutions V_x, V_m and W_x, W_m of the forward and adjoint problems are available.

(iv) The choice of an appropriate sequence of regularization parameter α_k crucially determines the performance of the iterative reconstruction algorithm. Following the analysis of the iteratively regularized Gauß-Newton method,[6, 9] the α_k's, should decay exponentially, e.g., $\alpha_k = \alpha_0 q^k$

for some $0 < q < 1$. The choice of α_0 and q, however, depends on the problem. While in principle, appropriate values can be found by a careful analysis of the problem under investigation, a more practical way to tune these parameters is to perform a sequence of reconstructions for known solutions and select regularization parameter that work well for all cases in this training set. The same set of parameters can then be used for similar problems.

(v) The choice of the stopping index `maxit` is another crucial ingredient for the numerical algorithm. If α_0 and q have been fixed, and $\alpha_k = \alpha_0 q^k$, then this amounts to choosing a minimal regularization parameter $\alpha_{min} := \alpha_{\texttt{maxit}}$. A rigorous way to choose α_{min} (respectively `maxit`) is provided by Morozov's discrepancy principle,[19,20] i.e., the reconstruction algorithm is stopped, as soon as $\|\mathcal{M}(c_k) - \mathcal{M}^\delta\| \sim \delta$, where δ is a measure for the measurement and model errors. Another rigorous way would be to choose $\alpha \sim \delta$. Both choices however require the knowledge of the level δ of the perturbations, which is usually not known in practice. A heuristic way to choose α_{min} without knowledge of δ is provided by the L-curve method[20,21] or generalized cross-validation.[22] If a series of similar problems is considered, then `maxit` (or α_{min}) can again be obtained by "training" the algorithm on a small data set with known solutions. This strategy is probably the most practical one and also the method we utilize in our experiments.

4. IMPLEMENTATION

In the following, we discuss the the main steps of the reconstruction algorithm in more detail, and provide some information about an efficient implementation. In order to quantify the expected computational workload, we also derive complexity estimates for the individual steps and provide upper bounds for the memory requirements.

Before going into details of the implementation, let us make some general remarks:

(i) The presented algorithms are independent of the architecture used for the actual computations, i.e., apart from some implementation details, the same algorithms are used for CPU and GPU implementations. In our numerical tests, the high-level algorithms are always controlled by the CPU, while the actual computations (e.g., matrix-vector multiplications) are executed on the CPU or GPU, respectively, inside a few “kernels”, e.g., the matrix-vector product. Only these few kernels have to be adapted to the specific hardware. The advantage of such a “minimally invasive” framework is that algorithms can be migrated gradually from one hardware to another while at the same time the high-level abstraction of the algorithms can be preserved. Such an approach seems very natural, and has been employed previously, e.g. for the simulation of fluid dynamics on CPU-GPU clusters.[23]

(ii) All compute intensive steps of the reconstruction algorithm have a high degree of locality: A major part of the operations, e.g. the assembly of the system matrices or the sensitivity, can be performed in an element-by-element fashion. This means that similar computations can executed for individual elements independently from each other. Global operations, e.g. the solution of the linear systems, rely on sparse matrix algebra. Note that each row (or column) of the sparse matrices corresponds to a vertex of the mesh, and the non-zero entries of this row stem from degrees of freedom (vertices) belonging to the patch of elements surrounding this vertex. Therefore, also sparse-matrix operations share a high degree of locality, and consequently, the overall reconstruction algorithm is very well-suited for parallelization.

4.1 Initializatation

The initialization step performs the following tasks:

```
  % initialize
  readMeshHierarchy;
  createLookUpTables;
  buildElementMatrices;
  generateSourcesAndDetectors;
  initializeParameters;
```

```
% readMeshHiearachy;
```

The first task is to set up a multilevel mesh hierarchy: The vertex coordinates of the finest level are loaded, and the transfer operators between different levels are stored as sparse prolongation matrices P_l^{l+1}. This allows to lift a function from level l to the next finer level $l+1$, i.e., $u_{l+1} = P_l^{l+1} u_l$ denotes the linear prolongation of a piecewise linear finite element function u_l to the finer mesh where it is represented by u_{l+1}.

The mesh hierarchy is only required for the initialization step. The memory requirement for storing the prolongation matrices P_l^{l+1} is $O(N_V)$.

```
% createLookUpTables;
```

The element vertex numbers are stored in an $N_T \times 4$ table containing the global vertex numbers for the N_T individual elements; the array g_T then contains the numbers of the vertices spanning the tetrahedron T. A similar $N_T \times 3$ table containing the vertices of the boundary triangles is generated, and g_E denotes the three vertices spanning the boundary element E. We also require the vertex-to-element connectivity, which is simply given by $L_V^E = (L_E^V)^\top$. The memory requirement for the look-up tables is $O(N_V + N_T)$.

```
% buildElementMatrices;
```

Since by assumption, the coefficients in the finite element systems are piecewise constant, the element contributions to the system matrices can be computed by a scaling operation from the element matrices for unit coefficients, i.e.,

$$\begin{aligned} K_{ij}^T &:= \int_T \nabla\psi_{g_T(i)} \nabla\psi_{g_T(j)} dx, & \text{and} \\ M_{ij}^T &:= \int_T \psi_{g_T(i)} \psi_{g_T(j)} dx & 1 \le i, j \le 4. \end{aligned}$$

These element matrices do not change during the reconstruction process, and can therefore be computed in advance. The same applies for the element matrices of the boundary terms

$$R_{ij}^E := \int_E \psi_{g_E(i)} \psi_{g_E(j)} dx \qquad\qquad 1 \le i, j \le 3.$$

The computation and storage of all element matrices requires $O(N_T)$ algebraic operations and $O(N_T)$ memory.

```
% generateSourcesAndDetectors;
```

The matrices Q and D for sources and detectors are defined by

$$Q_{ij} := \int_{\partial\Omega} \psi_i q_i ds \qquad \text{and} \qquad D_{ij} := \int_{\partial\Omega} \psi_i d_j ds,$$

and are assembled by summing over local element contributions. Since sources and detectors can be assumed to be non-overlapping, assembling of the source and detector matrices requires $O(N_V)$ operations. However, the matrices are stored as dense matrices for the sake of fast element access in the matrix-vector multiplication kernels and so the memory requirement is $O(ns \cdot N_V)$ and $O(nd \cdot N_V)$, respectively.

```
% initializeParameters;
```

As a final step in the initialization, we assign initial values to c_k, define the strategy for the choice of the regularization parameters α_k, and set tolerances and other inputs required by the solvers.

4.2 Assembly of the system matrices

The assembly of the system matrices consists of the following two steps:

```
  % assembleSystemMatrices(ck);
  assembleElementContributions;
  convertToCRSformat;
```

`% assembleElementContributions;`

In a first step, the precomputed matrices are used to generate the element and boundary element contributions for every element T respectively boundary element E. The element contributions for the system matrices $A_x := [K_x(c_k) + M_x(c_k) + R_x]$ for the excitation and $A_m := [K_m(c_k) + M_m(c_k) + R_m]$ for the emission wavelength are given by

$$A_x^T = \kappa_x(c_k^T)K^T + \mu_x(c_k^T)M^T, \qquad A_x^E = \rho_x R^E,$$
$$A_m^T = \kappa_m(c_k^T)K^T + \mu_m(c_k^T)M^T, \qquad A_m^E = \rho_m R^E.$$

Note that the construction of the local element matrices does only require local memory access and can be done fully in parallel. Therefore, this step provides almost optimal performance on both, CPU and GPU processors. The total complexity of this step $O(N_T)$.

`% convertToCRSformat;`

In a second step, the global matrices A_x and A_m are assembled from the local element matrices and stored in a sparse matrix in compressed row-storage (CRS) format. This step utilizes the look-up tables to map from an element to the associated nodes and requires non-local random memory access. The storage requirement for the sparse matrices is $O(N_V)$. For details on an efficient implementation of sparse matrix formats, we refer to [24, 25].

4.3 Solution of the linear systems

For the solution of the linear systems $A_xV_x = Q$ and $A_mV_m = G(c)V_x$ for the forward problem, respectively, $A_mW_m = D$ and $A_xW_x = G(c)W_m$ for the adjoint problem, we employ a preconditioned blocked conjugate gradient method (`pbcgSolve`): instead of solving for one right hand side (i.e. column in Q) after another, we solve simultaneously for all right hand sides (blocked). Additionally, we use a multigrid preconditioner to accelerate the convergence. The outline of the `pbcgSolve` routine is as follows:

```
% [x,d] = pbcgSolve(A,b)
x   = 0;
d   = b;
w   = mgPreconditioner(A,d);
s   = w;
wd  = w(:)'*d(:);
wd0 = wd;
for i=1:maxit
  As  = A*s;
  al  = wd / (As(:)'*s(:));
  x   = x + al*s;
  d   = d - al*As;
  w   = mgPreconditioner(A,d);
  wdo = wd;
  wd  = w(:)'*s;
  if wd<eps*wd0, break; end
  be  = wd/wdo;
  s   = w + be*s;
end
```

If we assume that the right hand side `b` has ns columns, then this algorithm is equivalent to solving an $ns \times N_V$ block diagonal system

$$\begin{pmatrix} A & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & A \end{pmatrix} \begin{pmatrix} x_1 \\ \vdots \\ x_{ns} \end{pmatrix} = \begin{pmatrix} b_1 \\ \vdots \\ b_{ns} \end{pmatrix},$$

with diagonal blocks A and right hand side $b(:) = [b_1; \ldots ; b_{ns}]$. Using the blocked version of the algorithm, and row-major storage for the right hand side b significantly improves the cache efficiency on both, CPU and GPU hardware.

As a preconditioner, we employ a single V-cycle of a geometric multigrid preconditioner

```
% x = mgPreconditioner(A,d)
[x,d] = bcgSolve(A,d);
dx = P * mgpre(A.parent,P'*d);
x  = x + dx;
r  = r - A*dx;
[x,d] = bcgSolve(A,d);
```

Here `P` denotes the prolongation matrix of the coarse to the fine mesh, and `A.parent` is the system matrix on the coarse mesh. For smoothing, one iteration of a blocked conjugate gradient method is employed (see `pbcgSolve`, but without further preconditioning). The application of one multigride V-cycle has linear complexity.

Note that all basic operations in the conjugate gradient method and the multigrid preconditioner (e.g., the matrix-vector products or simpler vector operations) can easily be executed on CPUs or GPUs, respectively. Due to the preconditioning, the typical iteration numbers required for convergence of the `pbcgSolve` algorithm are 10–15. Therefore, the total complexity of the solution of the forward and adjoint problems is $O(ns \cdot N_V)$ and $O(nd \cdot N_V)$, respectively.

4.4 Assembly of the sensitivity matrix

For the assembly of the the sensitivity matrix, we utilize the adjoint approach outlined in section 3.2. After discretization, the assembly process has the folloing form:

```
% assembleSensitivity
for k=1:nt
```

```
    Vmk = Vm(g(k),:); Vxk = Vx(g(k),:);
    Wmk = Wm(g(k),:); Vxk = Vx(g(k),:);
    S(:,:,k) = Wmk'*GT*Vxk ...
             - Wmk'*(KmT + MmT)*Vmk ...
             - Wxk'*(KxT + MxT)*Vxk;
  end
```

Here `KmT` denotes the element matrix K_m^T for the kth element T, and g is the element-to-vertex look-up table. Similar abbreviations are used for the other pre-computed element matrices. Since the computations for the individual elements are independent from each other, this loop can be executed in parallel. Assembling the sensitivity matrix requires $O(ns \cdot nd \cdot N_T)$ operations and the same amount of memory. Hence, this is the most time consuming and memory intensive step in the current algorithm. The assembly of the sensitivity matrices S_k can be avoided completely; cf [13] and the remarks in section 5.6.

4.5 Solution of the Gauß-Newton system

Formally, the Gauß-Newton matrix `Sk'*Sk + ak*R` is a dense $N_T \times N_T$ matrix, which would be expensive to compute and almost impossible to store. Since by construction `Sk'*Sk + ak*R` is symmetric and positive definite, the Gauß-Newton system can be solved efficiently by the method of conjugate gradients, which requires only the multiplication with S_k and S_k^*. In this manner, the complexity of multiplication with the Gauß-Newton matrix is given by $O(ns \cdot nd \cdot N_T)$ algebraic operations, whereas a multiplication with the pre-multiplied matrix `Sk'*Sk + ak*R` would be $O({N_T}^2)$.

4.6 Remarks

(i) The 4×4 structure of the element matrices can be utilized to increase GPU performance, since memory access and computations are very efficient for blocks of 16 threads.[26]

(ii) To achieve efficient code on graphics hardware, care has to be taken when accessing global memory. Memory access is most efficient, if a sequence of threads reads or writes to a sequential memory area (*coalesced* memory access).[26] Therefore, the photon field matrices V_x, V_m and W_x, W_m, but also the associated source and detector matrices Q and D are stored in row-major order. As one column of these matrices belongs to one photon field, this means that the elements of a given photon field are interleaved. In combination with the block conjugate gradient method, the use of this storage format provides speed-ups by a factor of $2-3$ on CPUs and up to 10 on GPUs.

(iii) The solution of the linear systems heavily relies on fast sparse matrix-vector products. For comparison of matrix-vector kernel performance of different sparse storage layouts we refer to [27]. Our implementation is based on the blocked interleaved format (a variant of the compressed row-storage format) described in [25].

(iv) For the iterative solution of the Gauß-Newton systems we utilize hardware optimized dense matrix-vector kernels, i.e., `LAPACK-BLAS`[28] for the CPU and a custom implementation for the GPU version.

5. RESULTS

5.1 Model problem

For our numerical tests, we considered a mouse phantom based on the model published in.[29] The measurement setup consisted of $ns = 24$ sources and $nd = 24$ detectors, which were arranged on three rings around the homogeneous torso. Two spherical fluorophore inclusions with a diameter of 5 mm were placed at the height of the middle optode ring inside the body, and the aim of our test was to reconstruct these inclusions. The setup is depicted in Figure 1.

For our simulations we used realistic optical tissue properties which were gathered from literature,[30–33] cf. table 1. These parameters were kept constant during simulation and reconstruction. In practice, the estimation of those background parameters would require additional a-priori knowledge or measurements.

The data used in our numerical tests were generated on a finer mesh with 200 835 elements and additionally perturbed by 1 % Gaussian noise relative to the largest measurement datum.

For the finite element discretization, we used a hierarchy of three nested tetrahedral meshes, which were generated with the open source mesh generator NETGEN.[34] The three meshes consisted of 2720, 21 760, and 174 080 elements and 853, 5103 and 34 677 vertices, respectively.

5.2 Choice of parameters

The regularization parameters α_k and the stopping index `maxit` in the Gauß-Newton algorithm were chosen as follows: We conducted a series of numerical tests for ten similar phantoms, and determined a set of parameters α_k that worked reasonably well for all samples. These parameters were then used for the actual reconstruction. For the results presented here, a geometric sequence $\alpha_k = q\alpha_0$ with $\alpha_0 = 1$ and $q = 0.2$ was used. The Gauß-Newton iteration was stopped when the regularization parameter reached the pre-defined threshold $\alpha_{min} = 1 \times 10^{-5}$, which corresponds to `maxit=8` Newton iterations.

5.3 Numerical results

All computations were executed on an AMD Phenom 9950 processor. The compute intensive operations (e.g. the matrix-vector multiplications and the assembly of the sensitivity matrix) were realized as compute “kernels”, which were implemented for both, CPU and GPU hardware. For our GPU tests, we used an NVIDIA GTX 280 graphics card.

In Figure 2, we also display a typical result of the image reconstruction.

5.4 Performance analysis

In table 2, we provide a detailed performance analysis of the reconstruction algorithm and compare the computation times of the CPU and GPU implementations. For all relevant tasks, the execution on the GPU results in a significant speed-up. An exception is the assembly of the system matrices on the coarsest level, where the number of elements is so small that the acceleration achieved on the graphics card is spoiled by the latency for starting the "kernels" on the GPU. This part of the computation is however negligible. The observed speed-up by factors of about 10 is in good agreement to the accelerations of sparse matrix-vector products.[24] (Note that the factor 10 is mainly due to the approximately 10 times larger memory bandwidth provided by GPUs).

5.5 Accuracy

Only single precision arithmetic executes very fast on current commodity graphics hardware, i.e., use of double-precision operations results in increased memory requirements and a tremendous performance drop. In order to obtain insight into the errors introduced by using single precision operations, we conducted the tests in single precision on CPU and GPU hardware, and also in double precision on the CPU.

First, the outcome of the forward operator (i.e. the simulated measurement data) is compared for the true fluorescence distribution on the mesh with with 200 835 elements. The errors are related to the largest measurement datum such that the numerical error can be compared to the data noise easily. Given the single- and double-precision CPU and GPU measurement data $\mathcal{M}_{CS}$, $\mathcal{M}_{CD}$ and $\mathcal{M}_{GS}$ the discrepancies are

$$\frac{\|\mathcal{M}_{CD} - \mathcal{M}_{CS}\|}{\max\left(\mathcal{M}_{CD}\right)} = 4.71 \times 10^{-5}$$

and

$$\frac{\|\mathcal{M}_{CD} - \mathcal{M}_{GS}\|}{\max\left(\mathcal{M}_{CD}\right)} = 7.63 \times 10^{-7}.$$

As can be seen, the numerical error is far beyond the measurement noise of 1 % of the largest measurement datum. For practical applications, the "noise" will be determined mainly by the measurement setup (e.g. inaccuracies in the optode locations), model errors and the detector noise, whereas the additional numerical noise is negligible. As a consequence, the same regularization parameter (which has to be chosen depending on the noise level) can be used for both the CPU and GPU implementation.

To give evidence for the similarity of the resultant images, the relative errors based on the L^2 norm $\|c\|_{L^2}^2 := \int_\Omega c(x)^2 dx$ of the reconstructed concentrations are compared. Denoting the reconstructed fluorophore concentration from the single- and double-precision CPU and GPU implementation with c_{CS}, c_{CD} and c_{GS}, these relative errors are

$$\|c_{CD} - c_{CS}\| / \|c_{CD}\| = 6.94 \times 10^{-3} \qquad \text{and}$$
$$\|c_{CD} - c_{GS}\| / \|c_{CD}\| = 2.39 \times 10^{-3}.$$

The resultant images differ by less than 1 % which we consider to be sufficient for practical applications.

5.6 Memory requirements

In table 3, we list the memory requirements for the essential data structures on the finest finite element mesh. The additional memory required for the coarse level matrices in the multigrid preconditioner is negligible.

In our implementation, the sensitivity matrix requires most of the GPU memory. To avoid excessive memory consumption, the product of the sensitivity matrix with a vector can also be computed “on the fly”, without the need to store the whole sensitivity. Also the complexity of the multiplication with the sensitivity (now $O(ns \cdot nd \cdot N_T)$) can be reduced, i.e., it has been demonstrated in[13] that the application of the sensitivity can be realized in $O(ns \cdot N_V)$ operations. Such an approach, however, requires the solution of the adjoint problems for every multiplication with the sensitivity matrix. Comparing with the computation times in table 2, this would drastically slow down the computations in practice, at least for the values of ns, nd, and N_T, N_V used in our test study. If the number of sources or detectors becomes very large (e.g. if a camera is used as detector), such an alternative might become favorable.

6. SUMMARY

In this article, we considered the image reconstruction in fluorescence tomography by iterative regularization methods of Newton-type and a discretization by finite element methods. We demonstrated that a combination of state of the art numerical methods, the utilization of modern hardware, and a hardware optimized implementation allows to solve this large-scale nonlinear inverse problem within a few seconds on standard desktop PCs, without compromising accuracy. In particular, the extensive use of the compute power of the GPU provided a significant speed-up of the reconstruction process by factors of approximately 10–15. Such a degree of acceleration is in good agreement to other work.[24,35]

In our opinion, the fast and reliable image reconstruction is a core requirement for the acceptance of new imaging modalities, like fluorescence tomography in pre-clinical but also in technical applications. Reconstruction times of a few seconds will allow the inspection of the results on-the-fly, and to acquire additional data if necessary.

7. DISCUSSION

The use of unstructured meshes for our finite element simulations provides high flexibility with respect to geometry and facilitates the mesh generation process. A further acceleration of the computations could however be achieved by utilizing structured grids, which lead to structured sparsity patterns of the system matrices and allow faster access of global memory.[36]

The lack of fast double-precision floating point operations on current graphics hardware limits the accuracy of the reconstructions to some extent. In our numerical examples, the relative errors in the forward simulation due to the single-precision arithmetic was less then 1% for both the CPU and GPU reconstructions. Thus, the numerical errors can be expected to be dominated by measurement noise and model errors. Single precision arithmetic may well suffice for this kind of (severely ill-posed) inverse problems.

By the advent of the next generation of graphics hardware (e.g. NVIDIA's FERMI architecture) in the next few months, the limitation to single-precision arithmetics on GPUs will be releaved. Moreover, we expect a further speed-up of our algorithms by a factor of about two by migrating to this new hardware.

ACKNOWLEDGMENTS

This work is supported by the project F3207-N18 granted by the Austrian Science Fund (FWF).

REFERENCES

[1] Mordon, S., Maunoury, V., Devoisselle, J. M., Abbas, Y., and Coustaud, D., "Characterization of tumorous and normal tissue using a ph-sensitive fluorescence indicator (5,6-carboxyfluorescein) in vivo.," *J Photochem Photobiol B* **13**, 307–314 (May 1992).

[2] Gannot, I., Ron, I., Hekmat, F., Chernomordik, V., and Gandjbakhche, A., "Functional optical detection based on ph dependent fluorescence lifetime.," *Lasers Surg Med* **35**(5), 342–348 (2004).

[3] Shives, E., Xu, Y., and Jiang, H., "Fluorescence lifetime tomography of turbid media based on an oxygen-sensitive dye," *Opt. Express* **10**, 1557–1562 (2002).

[4] Arridge, S. R., "Optical tomography in medical imaging," *Inverse Problems* **15**, R41–R93 (1999).

[5] Joshi, A., Bangerth, W., and Sevick-Muraca, W. M., "Adaptive finite element based tomography for fluorescence optical imaging in tissue," *Opt. Express* **12**, 5402–5417 (2004).

[6] Bakushinsky, A., "The problem of the convergence of the iteratively regularized Gauss-Newton method," *Comput. Math. Math. Phys.* **32**, 1353 – 1359 (1992).

[7] Hanke, M., "Regularizing properties of a truncated Newton-cg algorithm for nonlinear inverse problems," *Numer. Func. Anal. Optim.* **18**, 971 – 993 (1997).

[8] Marquardt, D., "An algorithm for least-squares estimation of nonlinear parameters," *SIAM J. Appl. Math.* **11**, 431 – 441 (1963).

[9] Kaltenbacher, B., "Some Newton-type methods for the regularization of nonlinear ill-posed problems," *Inverse Problems* **13**, 729 – 753 (1997).

[10] Jiang, H., "Frequency-domain fluorescent diffusion tomography: A finite-element-based algorithm and simulations," *Appl. Opt.* **37**, 5337–5343 (1998).

[11] Roy, R. and Sevick-Muraca, E. M., "Truncated newtons optimization scheme for absorption and fluorescence optical tomography: Part i theory and formulation," *Optics Express* **4**, 353–371 (1999).

[12] Schweiger, M., Arridge, S. R., and Nissilä, I., "Gauss–Newton method of image reconstruction in diffuse optical tomography," *Phys. Med. Biol.* **50**, 2365 – 2386 (2005).

[13] Egger, H. and Schlottbom, M., "Efficient reliable image reconstruction schemes for diffuse optical tomography," Tech. Rep. AICES-2010-10, RWTH Aachen University (2010).

[14] Godavarty, A., Sevick-Muraca, E. M., and Eppstein, M. J., "Three-dimensional fluorescence lifetime tomography," *Med. Phys.* **32**(4), 992–1000 (2005).

[15] Freiberger, M., Egger, H., and Scharfetter, H., "Nonlinear inversion schemes for fluorescence optical tomography," *IEEE Transactions on Biomedical Engineering* (in press).

[16] Milstein, A. B., Oh, S., Webb, K. J., Bouman, C. A., Zhang, Q., Boas, D. A., and Millane, R. P., "Fluorescence optical diffusion tomography," *Applied Optics* **42**, 3081–3094 (2003).

[17] O'Leary, M. A., Boas, D. A., X. D. Li, B. C., and Yodh, Y. G., "Fluorescence lifetime imaging in turbid media," *Opt. Lett.* **21**, 158–160 (1996).

[18] Ntziachristos, V. and Weissleder, R., "Experimental three-dimensional fluorescence reconstruction of diffuse media by use of a normalized born approximation," *Optics Letters* **26**, 893–895 (2001).

[19] Morozov, V. A., "On the solution of functional equations by the method of regularization," *Soviet Math. Dokl.* **7**, 414–417 (1966).

[20] Engl, H. W., Hanke, M., and Neubauer, A., [*Regularization of Inverse Problems*], Kluwer, Dordrecht (1996).

[21] Hansen, P. C., [*Rank-Defficient and Discrete Ill-Posed Problems*], SIAM, Philadeplphia (1998).

[22] Wahba, G., [*Spline Models for Observational Data*], SIAM, Philadelphia (1990).

[23] Göddeke, D., Becker, C., and Turek, S., "Integrating GPUs as fast co–processors into the parallel fe package FEAST," in [*19th Symposium Simulationstechnique*], Becker, M. and Szczerbicka, H., eds., *Frontiers in Simulation*, 277–282, SCS Publishing House e.V. (2006). ISBN 3-936150-49-4.

[24] Bell, N. and Garland, M., "Implementing sparse matrix-vector multiplication on throughput-oriented processors," in [*SC '09: Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis*], 1–11, ACM, New York, NY, USA (2009).

[25] Liebmann, M., *Efficient PDE Solvers on Modern Hardware with Applications in Medical and Technical Sciences*, PhD thesis, University of Graz (2009).

[26] NVIDIA, [*NVIDIA CUDA Programming Guide 2.0*], NVIDIA Cooperation (2008).

[27] Baskaran, M. M. and Bordawekar, R., "Optimizing sparse matrix-vector multiplication on GPUs," IBM Technical Report RC24704, IBM Ltd. (Apr. 2009).

[28] Anderson, E., Bai, Z., Bischof, C., Blackford, S., Demmel, J., Dongarra, J., Du Croz, J., Greenbaum, A., Hammarling, S., McKenney, A., and Sorensen, D., [*LAPACK Users' Guide*], Society for Industrial and Applied Mathematics, Philadelphia, PA, third ed. (1999).

[29] Dogdas, B., Stout, D., Chatziioannou, A. F., and Leahy, R. M., "Digimouse: a 3d whole body mouse atlas from ct and cryosection data," *Physics in Medicine and Biology* **52**, 577–587 (2007).

[30] Alexandrakis, G., Rannou, F. R., and Chatziioannou, A. F., "Tomographic bioluminescence imaging by use of a combined optical-pet (opet) system: a computer simulation feasibility study," *Physics in Medicine and Biology* **50**, 4225–4241 (2005).

[31] Keijzer, M., Star, W. M., and Storchi, P. R. M., "Optical diffusion in layered media," *Applied Optics* **27**, 1820–1824 (1988).

[32] Joshi, A., *Adaptive finite element methods for fluorescence enhanced optical tomography*, PhD thesis, Texas A&M University (2005).

[33] Landsman, M. L., Kwant, G., Mook, G. A., and Zijlstra, W. G., "Light-absorbing properties, stability, and spectral stabilization of indocyanine green," *J Appl Physiol* **40**(4), 575–583 (1976).

[34] Schöberl, J., "Netgen - an advancing front 2d/3d-mesh generator based on abstract rules," *Computing and Visualization in Science* **1**(1), 41–52 (1997).

[35] Göddeke, D., Wobker, H., Strzodka, R., Mohd-Yusof, J., McCormick, P. S., and Turek, S., "Co-processor acceleration of an unmodified parallel solid mechanics code with FEASTGPU," *International Journal of Computational Science and Engineering* **4**, 254–269 (Oct. 2009).

[36] Köster, M., Göddeke, D., Wobker, H., and Turek, S., "How to gain speedups of 1000 on single processors with fast FEM solvers – benchmarking numerical and computational efficiency," tech. rep., Fakultät für Mathematik, TU Dortmund (Oct. 2008). Ergebnisberichte des Instituts für Angewandte Mathematik, Nummer 382.

Table 1. Optical parameters used for the simulations.

	μ_s'	$\mu_{a,i}$	ε	ρ
	mm^{-1}	mm^{-1}	$\mathrm{mm}^{-1}\,\mathrm{M}^{-1}$	
excitation	0.275	0.036	$8.35 \cdot 10^3$	0.2
emission	0.235	0.029	$2.81 \cdot 10^3$	0.2

Table 2. Comparsion of CPU and GPU reconstruction times for the digimouse phantom using 174 080 elements.

Task	CPU		GPU		Speed-up
Assembly of the system matrices					
`assembleElementContributions`					
Mesh level 1	0.63	ms	0.04	ms	
Mesh level 2	6.37	ms	0.10	ms	
Mesh level 3	121.97	ms	0.66	ms	
`convertToCRSformat`					
Mesh level 1	n.a.[1]	ms	0.72	ms	
Mesh level 2	n.a.[1]	ms	2.06	ms	
Mesh level 3	n.a.[1]	ms	13.17	ms	
Total					
Mesh level 1	0.63	ms	0.76	ms	0.83
Mesh level 2	6.37	ms	2.16	ms	2.95
Mesh level 3	121.97	ms	13.83	ms	8.82
Solution of the linear systems					
`pbcgSolve` without multigrid					
Forward solution (V_x, V_m)	10.12	s	736.64	ms	13.73
Adjoint solution (W_x, W_m)	8.84	s	633.92	ms	13.94
`pbcgSolve` with multigrid					
Forward solution (V_x, V_m)	5.45	s	461.60	ms	11.81
Adjoint solution (W_x, W_m)	6.02	s	508.29	ms	11.85
Computation of the matrix of measurements					
$D^\top V_m$	496.40	ms	6.53	ms	76.01
Assembly of the sensitivity matrix					
`assembleSensitivity`	16.68	s	1.03	s	16.23
Solution of the Gauß-Newton system					
`Sk'*Sk + ak*R`	10.87	s	331.92	ms	32.75
Total reconstruction time					
Without multigrid	6.39	min	27.39	s	14.00
With multigrid	5.41	min	24.94	s	13.01

[1] On the CPU the system matrices are assembled in one step

Table 3. Estimation and true memory required (in mega bytes) for selected data structures of the reconstruction algorithm. using the mouse phantom with $N_V = 34\,677$ nodes, $N_T = 174\,080$ tetrahedral elements, $ns = 24$ sources, $nd = 24$ detectors and a floating point size type of 4 bytes (i.e. single-precision).

Data structure	Estimate	MB
Local element matrices K, M, R	$3 \cdot (4 \cdot 4) \times N_T$	31.88
Assembled matrices A_x, A_m, G	depending on connectivity	15.31
Source vectors Q	$ns \times N_V$	3.17
Detector vectors D	$nd \times N_V$	3.17
Forward solutions V_x, V_m	$2 \cdot ns \times N_V$	6.35
Adjoint solutions W_x, W_m	$2 \cdot nd \times N_V$	6.35
Sensitivity S	$(nd \cdot ns) \times N_T$	382.50
Total		448.73

List of Figures

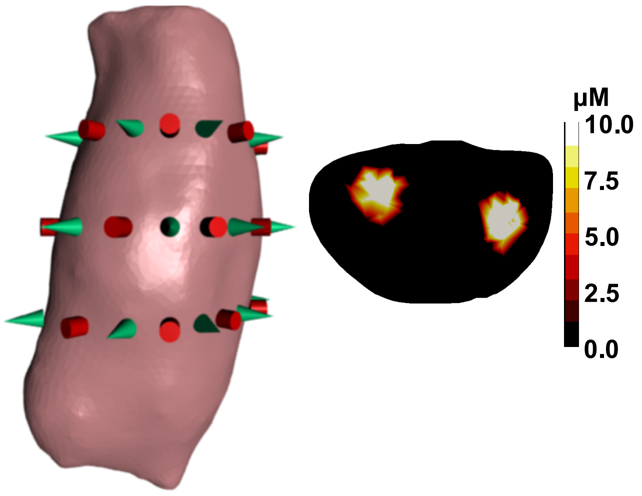

Figure 1. Simulation setup showing the mouse phantom, the excitation sources (cylinders) and the photon-detectors (cones). The cross-section at the height of the central optode ring shows the assumed concentration distribution which consists of two 5 mm spheres with a fluorophore concentration of 10 μM.

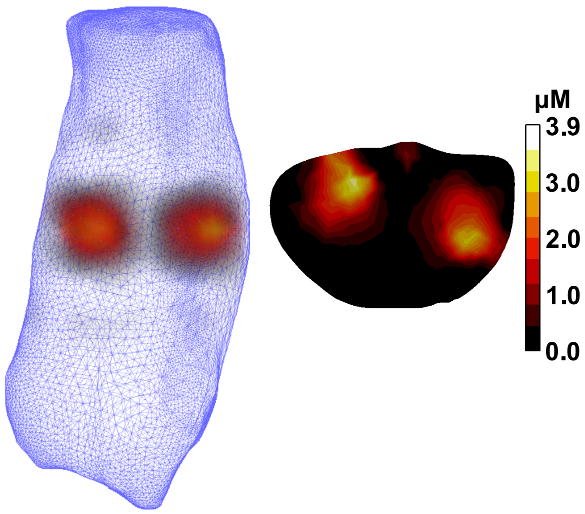

Figure 2. Reconstruction of the two fluorescent inclusions shown in Figure 1 performed with graphics hardware acceleration. The cross-section is again taken at the height of the middle optode ring. 1 % noise relative to the largest measurement datum was added for to the reconstruction.