Implementation of a real-time algorithm for estimating spatially resolved optical properties from hyperspectral images

Asgeir Bjorgan and Lise Lyngsnes Randeberg
Department of Electronics and Telecommunications, Norwegian University of Science and Technology, Trondheim

First made available on February 20, 2015 *

Non-contact spectroscopy of human tissue is possible through the use of hyperspectral imaging technology. While this imaging technique is fast, the required data processing can be extensive and time-consuming due to the high spectral and spatial resolution of the raw data. Clinical applicability requires a low latency in the processing pipeline. This technical report presents the design and implementation of a GPU-based algorithm for extracting skin optical properties from hyperspectral images. The algorithm is shown to have real-time performance with respect to the typical image acquisition speed. Validation and modeling details of the base inverse modeling technique are provided in a previous study (doi:10.1117/1.JBO.19.6.066003). Source code is available under the MIT license at `http://www.github.com/ntnu-bioopt/gpudm`.

1 Introduction

Hyperspectral imaging has recently been adopted for imaging of human tissue [1]. High spectral resolution in the technology enables non-contact, spatially resolved skin spectroscopy. The combination of statistical methods and physics-informed models can be used to derive objective, diagnostic information. Possible examples of clinical application include monitoring the progression of wound healing, monitoring tissue perfusion, detecting arthritic finger joints and diagnosing atherosclerosis [2, 3, 4, 5, 6].

The high inherent data dimensionality of hyperspectral images can cause high processing times, which could extend beyond what would be usable in the clinic. Limited patient time requires tools aiding in diagnostic decisions to be fast. Processing times must be constrained to a bare minimum. Results, possibly down to a final diagnosis, should be available already within the end of image acquisition.

One of the prerequisites for a diagnostic system based on hyperspectral imaging is a general inverse modeling technique. This is used for estimating skin optical properties from hyperspectral images. The layered, scattering nature of skin requires optical modeling of light transport for proper treatment of optical properties [7, 8, 9]. This can be achieved through the iteration of light transport models with respect to input optical properties [10, 11, 12]. Such algorithms exist, but have not been developed with firm timing constraints in mind. Required processing times can be arbitrary, depending on the complexity of the models.

The core inverse modeling technique is required to be as fast as possible. Real-time performance is therefore desired. The presented algorithms have been developed for a line scanning camera setup. In our study, the Hyspex VNIR-1600 camera from Norsk Elektro Optikk, Lillestrøm was used for data collection. For this specific system the real-time processing is constrained by the time window between arrival of subsequent lines of data, which is 30 ms per line of data (1600 pixels × 160 wavelengths (bands)). However, the presented algorithms can easily

[*] Available on http://github.com/ntnu-bioopt/gpudm,

be adapted to fit other line scanning camera configurations.

An algorithm being sufficient both in inverse modeling performance and processing time has been developed using general-purpose GPU programming (GPGPU). A previous study [13] presented the physical model and tested it using Monte Carlo simulations [14]. This technical report concerns the technical implementation issues of the algorithm. The source code has been made available.

2 GPU programming

GPU technology is frequently used for speeding up processing algorithms in hyperspectral remote sensing [15, 16, 17, 18, 19, 20, 21]. The GPU follows a SIMD (Single instruction, multiple data) principle. Processing is ideally parallelized by applying the same processor instructions on different datasets [22]. Threads are run concurrently in batches (*warps*) of 32 on a single GPU core. Branches and divergent execution is handled serially. Threads are organized in blocks, handled concurrently by different GPU cores. Each thread runs a *kernel*, a C function with CUDA extensions. Each thread will process the instructions contained in the kernel, on different data accessed using thread- and block indices [22].

CUDA kernels have access to roughly three different kinds of memory, listed in order of access speeds from slow to fast, and amount from abundant to scarce: *global memory*, *shared memory*and a *registry* (DRAM, and two kinds of cache). The slow global memory is accessible outside of the kernels and by all threads. Shared memory is accessible from all threads within a block. Variables within the scope of the CUDA kernels are primarily allocated within the fast registry.

Coalesced global memory access is one way to ensure efficient global memory access. The principle of spatial locality ensures that each access to global memory caches previous and subsequent data. Threads in a warp can take full advantage of the cache if they all access subsequent positions in the global memory, with the access in the first thread being aligned with the first position of the cache reading line.

3 Optical modeling

The optical inverse modeling is described in full in [13], but will be summarized here.

3.1 Modeling diffuse reflectance from skin

A diffusion model with isotropic source functions was used to simulate the light transport in human tissue [9, 13, 8]. It is possible to obtain a closed-form, analytic expression for the diffuse reflectance from this model [9]. The complexity of the expression and its analytic derivative is suitable for a self-contained GPU kernel implementation by limiting the model to a two-layered skin model (epidermis and dermis). This also ensures a simpler optimization scheme for the iterative inverse model by having fewer parameters to fit.

Real human skin is non-homogeneous and approximated more appropriately by multiple skin layers. This is partly corrected by applying the inverse model to parts of the diffuse reflectance spectrum having a uniform penetration depth. The estimated properties are assumed to be a mixture of the properties contained in the separate layers down to the penetration depth. Depth-resolved properties down to superficial and deeper layers are obtained by exploiting the variation of penetration depth between the shorter and longer wavelengths. See Fig. 1.

The first layer of the skin model contains melanin. The second, semi-infinite layer contains linearly mixed deoxyhemoglobin, oxyhemoglobin and other chromophores (e.g. water, methemoglobin, lipids, background absorption). The skin materials are included through their various absorption spectra, and combined

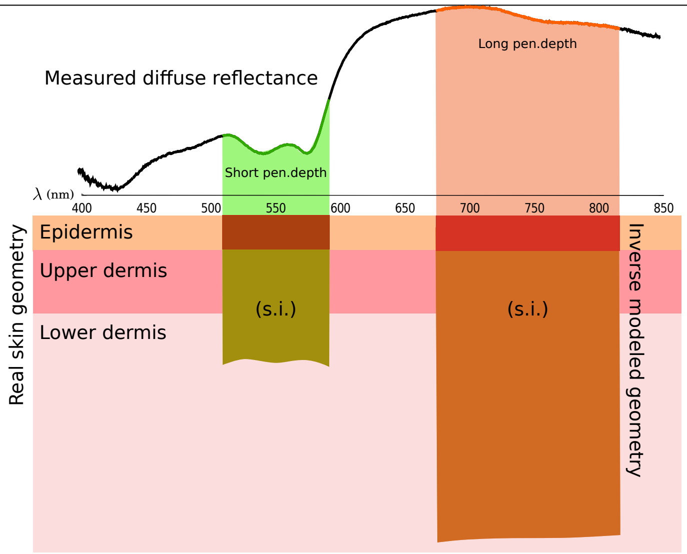

Figure 1: *Application of a two-layered inverse model to a three-layered situation. The two-layered skin model approximates the properties of multiple, inhomogeneous layers to a single, homogeneous and semi-infinite (s.i.) layer where the derived properties are distributed evenly throughout the layer. Different depths are targeted by exploiting the variation in penetration depth. The figure is modified from a figure previously published under the Creative Commons Attribution 3.0 Unported license [23] in Bjorgan et al. [13] (doi:10.1117/1.JBO.19.6.066003).*

to yield the epidermal absorption coefficient $\mu_{a,e}(\lambda)$ (`muae`) and dermal absorption coefficient $\mu_{a,d}(\lambda)$ (`muad`). A fixed, wavelength-dependent scattering function $\mu_s(\lambda)$ (`muse`, `musd`) is assumed in both skin layers [13].

3.2 Inverse model

The inverse model is based around the independent estimation of $\mu_{a,e}(\lambda)$ and $\mu_{a,d}(\lambda)$. Either absorption coefficient can be found in a non-thread-divergent way by fitting the simulated reflectance to the measured reflectance through a few, fixed number of iterations of Newton-Rhapson's method. The epidermal melanin concentration is determined by estimating and unmixing dermis and epidermis in turn and assuming a temporary melanin amount in dermis. Now having an estimate of $\mu_{a,e}(\lambda)$, $\mu_{a,d}(\lambda)$ can be found. The skin constituents can then be estimated through a separate spectral unmixing algorithm. Details on the inverse model and evaluation of the estimation performance is found in a previous study [13]. See also Fig. 2.

The inverse model is independently applied to each pixel in the hyperspectral image. This eases the applicability of SIMD parallelization to the problem.

3.3 Spectral unmixing

Spectral unmixing is used to estimate skin constituents given their fixed absorption spectra and the estimated absorption spectrum. Advanced algorithms, from algorithms taking spatial information into account, to algorithms where the chromophore spectra are estimated along with their concentrations, can be used. In this study, we use a simpler algorithm. We assume the absorption spectra of the materials fixed and estimate the skin constituents using a non-negative least squares (NNLS) algorithm. The problem is expressed as

$$\vec{\mu}_{a,d} = A\vec{x}, \tag{1}$$

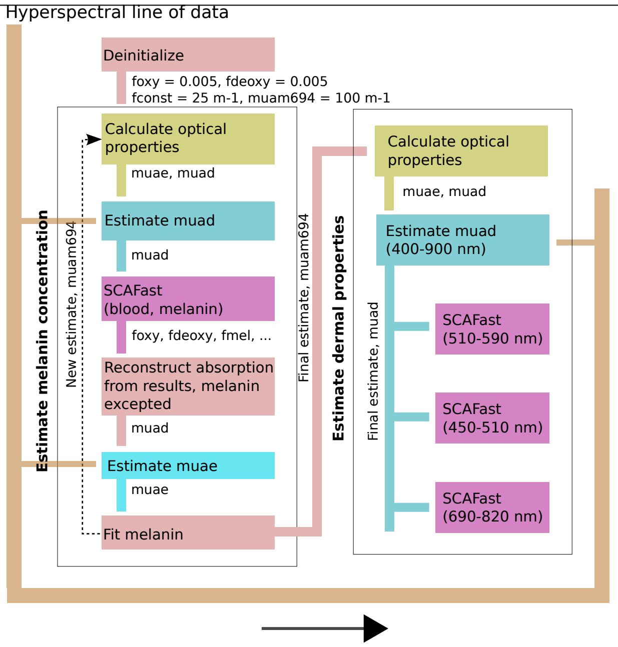
Hyperspectral line of data
Deinitialize
foxy = 0.005, fdeoxy = 0.005
fconst = 25 m-1, muam694 = 100 m-1
Estimate melanin concentration
New estimate, muam694
Calculate optical properties
muae, muad
Estimate muad
muad
SCAFast (blood, melanin)
foxy, fdeoxy, fmel, ...
Reconstruct absorption from results, melanin excepted
muad
Estimate muae
muae
Fit melanin
Final estimate, muam694
Estimate dermal properties
Calculate optical properties
muae, muad
Estimate muad (400-900 nm)
Final estimate, muad
SCAFast (510-590 nm)
SCAFast (450-510 nm)
SCAFast (690-820 nm)

Figure 2: *The sequence of CUDA operations for the inverse modeling of a hyperspectral line. The variables* `muae`*,* `muad` *correspond to the epidermal and dermal absorption coefficients. The variables foxy, fdeoxy correspond to the fraction of oxygenated and deoxygenated blood in dermis. The variable fconst is the constant background absorption in dermis. The variable fmel is a temporary melanin amount set in dermis, while muam694 is the amount of melanin in epidermis. The melanin estimation algorithm is run twice.*

where $\vec{x}$ is to be found under a non-negativity constraint. A is a matrix containing the chromophore absorption spectra as its row vectors.

Our current implementation makes use of the sequential coordinate-wise algorithm for non-negative least squares problems (SCA) [24]. The algorithm is simple and iterative and suitable for SIMD implementation given a fixed amount of iterations. It is also memory effective, needing in each iteration access only to the smaller matrix A^TA (size `numChromophores` × `numChromophores`) and two small arrays (each of size `numChromophores`).

4 Real-time implementation

Clinical applicability of the inverse modeling methods puts constraints on the processing time due to the limited time the medical personnel has for each patient. Each processing step towards the final result is required to be as fast as possible, preferably real-time.

The inverse modeling algorithm has therefore been designed to meet a soft real-time requirement. This is achieved through the use of an NVIDIA GPU (Geforce GTX 670). One line of data consisting of 1600 pixels and 160 wavelengths is inverse modeled at a time within 30 ms. This is the data size and streaming speed of the assumed hyperspectral camera (Hyspex VNIR-1600).

The basic building blocks of the inverse algorithm are the estimation of $\mu_{a,e}$, the estimation of $\mu_{a,d}$ and spectral unmixing using SCA. These operations are implemented as separate GPU kernels, along with helper kernels for the calculation of optical properties. See Fig. 2.

Source code for the implementation is available under the MIT license at `http://www.github.com/ntnu-bioopt/gpudm`.

4.1 Parallelization and memory structure

Hyperspectral data can be structured in memory space using three interleaving schemes, exemplified using an ordinary red, green and blue (RGB) image:

- BSQ - band-sequential. Ex.: `rrrr... gggg... bbbb... END`
- BIP - band-interleaved-by-pixel. Ex.: `rgbrgbrgbrgbrgb... END`
- BIL - band-interleaved-by-line. Ex.: `[rrr... ggg... bbb...]`$_{\text{line 1}}$ `[...]`$_{\text{line 2}}$ `... END`

Only BIP and BIL are suitable for streaming of hyperspectral data line by line.

The inverse modeling chain requires efficient implementation of the following tasks:

- The estimation of the absorption coefficient μ_a for each wavelength and pixel
- Spectral unmixing of the absorption spectra across all pixels

The estimation of the absorption coefficient is independent of pixel and wavelength, and can be fully parallelized using either interleave. It is assumed that spectral unmixing is done independent of pixel. A full parallelization of spectral unmixing, down to band level, would require BIP interleave. The algorithm would have to make use of shared memory across bands, resulting in one block of CUDA threads being assigned to one or several spectra. The threads would have to access subsequent band values associated with a pixel for coalesced memory access (i.e. BIP interleave). However, this parallelization strategy poses difficulties for the free wavelength choice and some spectral unmixing algorithms.

The inverse modeling method requires spectral unmixing of defined wavelength intervals within the full spectral range. The above scheme requires arbitrary wavelength ranges to start at wavelength indices which are not a multiple of 32, and have a number of threads per block not being a multiple of 32. Both cases result in inefficient, non-coalesced memory access. The latter problem can be alleviated by including multiple spectra in the unmixing. The former can be alleviated by pitching the memory arrays between kernel calls. This will introduce additional complexity and overhead, however. Not all spectral unmixing algorithms are suitable for full parallelization at band level. The SCA algorithm has an initial step which is parallelizable at band-level, but the subsequent steps are not. The simplest scheme, suitable for SCA and for the free wavelength choice, is to use BIL interleave and parallelize only at pixel level. This is illustrated in Fig. 3. Omitting wavelengths will not interfere with memory alignment or thread distribution in this scheme. Similarly, thread parallelization using BIL interleave for the estimation of absorption coefficients is shown in Fig. 4.

The number of threads per block was chosen to be 160 for all kernels, as this was found to be suitable for the limits set by the high registry usage.

4.2 Kernel implementation

4.2.1 Estimation of absorption

The CUDA kernels dealing with the estimation of optical properties are

- **__global__ void calcSkinData()** - calculate optical properties given skin properties
- **__global__ void ReflIsoL2EstimateMuae()** - estimation of $\mu_{a,e}$ with respect to the input reflectance. Assumes a fixed $\mu_{a,d}$ and uses the two-layered diffusion model with isotropic source functions.
- **__global__ void ReflIsoL2EstimateMuad()** - estimation of $\mu_{a,d}$ with respect to the input reflectance. Assumes a fixed $\mu_{a,e}$.

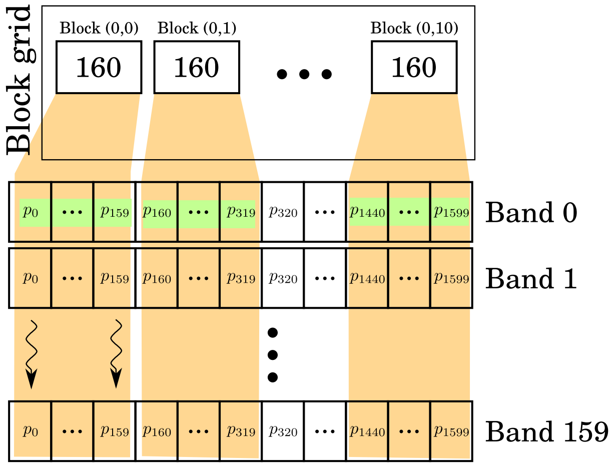

Figure 3: *Spectral unmixing parallellized in CUDA for BIL interleave. Threads are assigned in blocks of 160, each thread dealing with the unmixing of an independent pixel. The memory accesses will be coalesced across the threads at each step where values are read in from the hyperspectral data array. Any matrices dealing with chromophore absorption values are read into shared memory and broadcast across the threads within a specific block.*

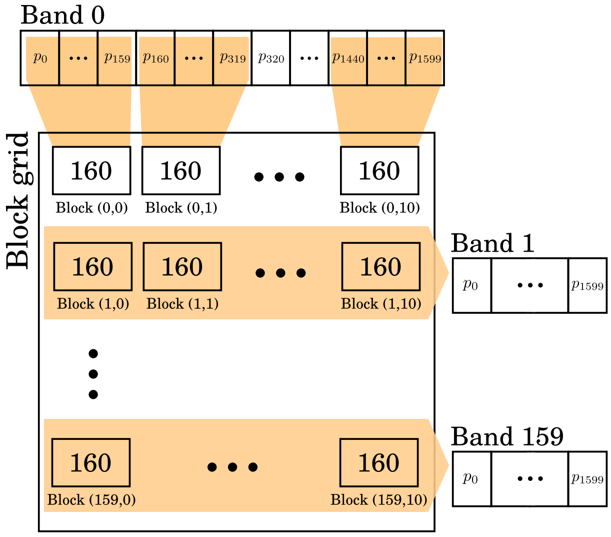

Figure 4: *CUDA block and thread distribution across wavelengths and pixels for hyperspectral absorption extraction. For this particular GPU, threads are assigned in blocks of 160 to subsequent pixels in a particular band to ensure coalesced memory access. Blocks are arranged in the larger grid according to the bands. Each thread estimates its own absorption coefficient independently from the others, one for each pixel and band.*

Estimation of the individual absorption coefficient involves the use of the analytic expression for calculating the diffuse reflectance, $R(\mu_{a,e}, \mu_{a,d}, \mu'_{s,e}, \mu'_{s,d})$, and its derivative, $\frac{\partial R}{\partial \mu_{a,*}}$, into Newton-Rhapson's method (15 iterations). The expressions are complex, and it is assumed that they are not easily optimized by the CUDA compiler while retaining the required numerical accuracy. The parts of the expressions which can be calculated only once are calculated initially in the kernel and saved in the registry, while the parts depending on the variable under estimation is calculated during the for loop iteration. This can limit throughput due to the high registry usage.

Values related to fixed absorption spectra are read into the **calcSkinData()** function from global memory and used directly. Absorption or scattering spectra which can be calculated using analytic functions are calculated through the use of a shared variable containing the current wavelength.

4.2.2 Spectral unmixing

The CUDA kernels dealing with spectral unmixing are

- **__global__ void SCA()**
- **__global__ void SCAFast()**.

The SCA algorithm is used in this study primarily due to its few and efficient computations at each iteration of the method. Lag due to excess global memory access should be reduced in order to take advantage of this. The parallelization strategy requires, for each pixel, repeated access to an array of chromophore fractions, an array of Lagrange multipliers and the matrix $A^T A$.

Fast access using shared memory is implemented in **SCA()**. However, this amount of shared memory occupies 12 kB of the allowable shared memory (varies with compute capability from 14 to 48 kB). Using this much of the allowable shared memory for a single block of threads limits throughput. Shared memory is also slower than the registry, and the scheme does not actually take advantage of the fact that shared memory can be shared between the threads. Shared memory is not suitable as a solution to the problem, and also does not scale well with increased GPU power.

The kernel **SCAFast()** was therefore written to move the memory allocation from shared memory to the per-thread registry. This is no more optimal than the use of shared memory with respect to occupancy, but the registry is faster and more suitable for per-thread usage of memory. The difficulty of allocating arrays in registry memory is circumvented by naming the variables explicitly and using C macros and just-in-time compilation. Code readability is reduced, however.

SCA is run for 300 iterations.

5 Results and discussion

A clinical application of hyperspectral imaging requires fast processing methods. One of the necessary steps for the realization of a diagnostic system is an inverse modeling technique for estimating skin optical properties. As a core technique, it is important that the processing time is minimal. The aim of this technical report is to present the technical details of an inverse modeling method for hyperspectral images of skin and evaluate its real-time computational performance.

Total timing results are shown in Fig. 5. Memory transfer times between host and GPU is not considered as it is assumed that this can be done in parallel with the processing. Skin optical parameters are estimated already after 3.5 ms of the allotted 30 ms of computational time. This leaves GPU and CPU time for other, future processing operations which can be scheduled in the time taken to scan a line of hyperspectral data. Examples include statistical processing and noise removal [25].

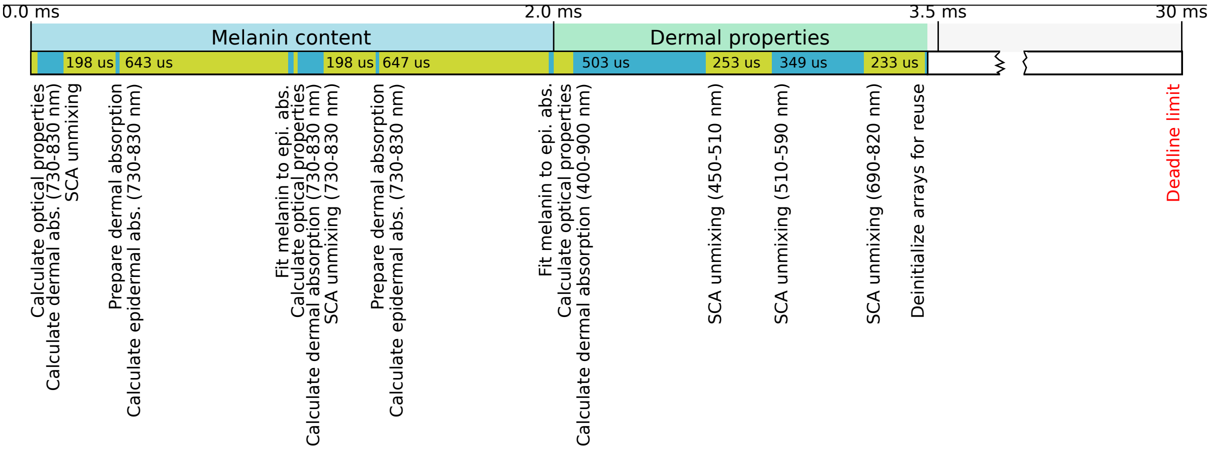

Figure 5: *Total computational times for the inverse modeling of one hyperspectral line of data. Three wavelength intervals were used in the unmixing of the dermal absorption, the line of data had 1600 samples × 160 bands of data. The figure is modified from a figure previously published under the Creative Commons Attribution 3.0 Unported license [23] in Bjorgan et al. [13] (doi:10.1117/1.JBO.19.6.066003).*

CUDA has no real-time guarantees, but it can safely be assumed that no other application will be able to disturb the GPU processing times significantly as long as the computer is left for processing operations only. Visualization and CPU processing can cause delay in the kernel launches. However, this is not significant for the fulfillment of a soft real-time requirement.

The optimality and scalability of the implementations can be evaluated by measuring the performance boost gained from upgrading the computer hardware. Performance of selected kernel calls are compared in Fig. 6 for different GPUs. Performance is increased far more for the kernels estimating absorption coefficients than for SCA (shared memory implementation).

The kernels dealing with the estimation of μ_a are registry bound due to the high amount of intermediary calculations. Speed is increased by a significant amount with newer GPU, but only due to the higher availability of registers and/or faster registers. The speed gain would be higher with better throughput, but the complex computations do not allow for this.

The SCA implementation using shared memory did not improve performance significantly between GPU models, due to no changes in the amount of shared memory. Shared memory usage is maximized, and the GPU is not able to switch blocks of threads in and out of the GPU core in order to hide away excess lag due to memory access. This strategy was therefore compared to an implementation using the registry (see table 1) and an implementation using global memory in every iteration (see table 2).

The variant using global memory has a lower running time per line of data when using a higher number of threads per blocks and a higher number of lines inverse modeled in parallel. This is not surprising as the GPU is able to switch blocks in and out of the GPU cores to hide global memory access. The version using the registry has a lower running time and outperforms the rest. As of now, the registry version has highest performance. Changes to the amount of data or number of chromophores may necessitate the registry version to be abandoned, where it is likely that the version using global memory may be more suitable due to scalability. This can also change with GPU models.

A real-time inverse modeling technique for estimating skin optical properties from hyperspectral images is an important step towards a usable diagnostic system for clinical applica-

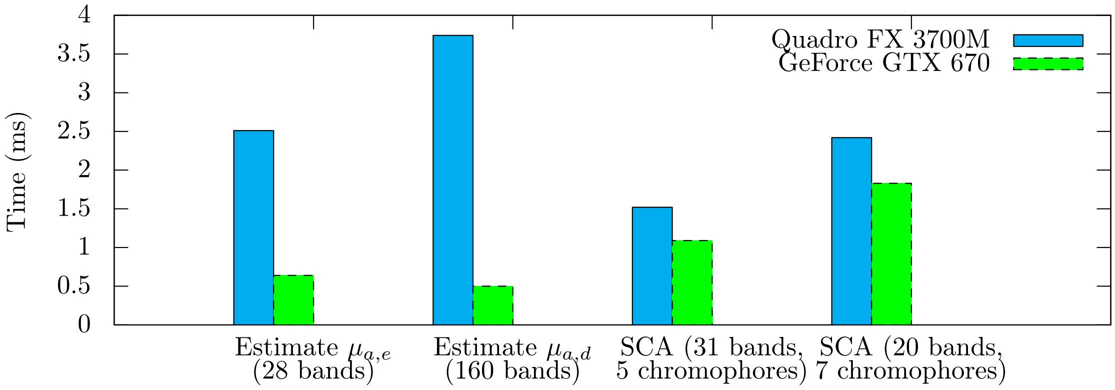

Figure 6: *Comparison of running times for some CUDA kernels across graphics cards.*

Table 1: *Comparison between SCA and SCAFast, 20 bands and 7 chromophores*

Function	Time (ms)	Shmem (kB)	Registers	Theor. occ. (%)
SCA	1.8	12.9	18	23
SCAFast	0.3	0.43	63	47

Table 2: *Comparison of running times for different variants of SCA either allocating all arrays in shared memory or using the global memory. 20 bands and 7 chromophores.*

SCA implementation	Running time per line (ms)	Theor. occupancy (%)	Shared memory/block (kB)
w/ shared memory	1.8	23	12.9
w/ global memory	4.8	94	0.40
800 threads/block, 4 lines	1.5	78	0.20

tions. The presented approach has been shown to have sufficient performance in terms of time, and source code availability can facilitate further development.

6 Conclusion

An inverse modeling tool for hyperspectral images of human tissue has been developed and found to fulfill the defined real-time requirements and still leave possibilities for further improvements or other processing.

Acknowledgments

This work is a part of the Iacobus project, `http://www.iacobus-fp7.eu`. Iacobus is supported by the European Commission's 7th RTD Framework Programme Collaborative Project No. 305760. The study has also been funded by the MedIm Bridging Grant.

References

[1] G. Lu and B. Fei, "Medical hyperspectral imaging: a review," *J. Biomed. Opt.*, vol. 19, no. 1, p. 010901, 2014.

[2] M. Denstedt, B. S. Pukstad, L. A. Paluchowski, J. E. Hernandez-Palacios, and L. L. Randeberg, "Hyperspectral imaging as a diagnostic tool for chronic skin ulcers," in *Proc. SPIE*, vol. 8565, 2013, pp. 85 650N–85 650N–14.

[3] L. L. Randeberg, E. L. P. Larsen, and L. O. Svaasand, "Characterization of vascular structures and skin bruises using hyperspectral imaging, image analysis and diffusion theory," *J Biophotonics*, vol. 3, no. 1-2, pp. 53–65, 2010.

[4] B. S. Sorg, B. J. Moeller, O. Donovan, Y. Cao, and M. W. Dewhirst, "Hyperspectral imaging of hemoglobin saturation in tumor microvasculature and tumor hypoxia development," *J. Biomed. Opt.*, vol. 10, no. 4, 2005.

[5] M. Milanic, L. A. Paluchowski, and L. L. Randeberg, "Simulation of light transport in arthritic-and non-arthritic human fingers," in *Proc. SPIE*, vol. 8936, 2014.

[6] E. L. Larsen, L. L. Randeberg, E. Olstad, O. A. Haugen, A. Aksnes, and L. O. Svaasand, "Hyperspectral imaging of atherosclerotic plaques in vitro," *J. Biomed. Opt.*, vol. 16, no. 2, 2011.

[7] L. Wang, S. L. Jacques, and L. Zheng, "Mcml monte carlo modeling of light transport in multi-layered tissues," *Comput Meth Prog Bio*, vol. 47, no. 2, pp. 131 – 146, 1995.

[8] R. C. Haskell, L. O. Svaasand, T. Tsay, T. Feng, M. S. McAdams, and B. J. Tromberg, "Boundary conditions for the diffusion equation in radiative transfer," *J Opt Soc Am A*, vol. 11, no. 10, pp. 2727–2741, Oct 1994.

[9] L. Svaasand, L. Norvang, E. Fiskerstrand, E. Stopps, M. Berns, and J. Nelson, "Tissue parameters determining the visual appearance of normal skin and port-wine stains," *Laser Med Sci*, vol. 10, pp. 55–65, 1995.

[10] I. Fredriksson, M. Larsson, and T. Stromberg, "Inverse monte carlo method in a multilayered tissue model for diffuse reflectance spectroscopy," *J Biomed Opt*, vol. 17, no. 4, pp. 047 004–1–047 004–12, 2012.

[11] R. Zhang, W. Verkruysse, B. Choi, J. A. Viator, B. Jung, L. O. Svaasand, G. Aguilar, and J. S. Nelson, "Determination of human skin optical properties from spectrophotometric measurements based on optimization by genetic algorithms," *J Biomed Opt*, vol. 10, no. 2, pp. 024 030–024 030–11, 2005.

[12] L. L. Randeberg, O. A. Haugen, R. Haaver-

stad, and L. O. Svaasand, "A novel approach to age determination of traumatic injuries by reflectance spectroscopy," *Laser Surg Med*, vol. 38, no. 4, pp. 277–289, 2006.

[13] A. Bjorgan, M. Milanic, and L. L. Randeberg, "Estimation of skin optical parameters for real-time hyperspectral imaging applications," *J. Biomed. Opt.*, vol. 19, no. 6, 2014.

[14] E. Alerstam, W. C. Y. Lo, T. D. Han, J. Rose, S. Andersson-Engels, and L. Lilge, "Next-generation acceleration and code optimization for light transport in turbid media using gpus," *Biomed Opt Express*, vol. 1, no. 2, pp. 658–675, 2010.

[15] C. Gonzalez, S. Sanchez, A. Paz, J. Resano, D. Mozos, and A. Plaza, "Use of fpga or gpu-based architectures for remotely sensed hyperspectral image processing," *Integration*, vol. 46, pp. 89–103, 2013.

[16] D. González, C. Sánchez, R. Veguilla, N. G. Santiago, S. Rosario-Torres, and M. Vélez-Reyes, "Abundance estimation algorithms using nvidia (r) cuda (tm) technology," in *Proc. of SPIE Vol. 6966*, 2008.

[17] J. Setoain, M. Prieto, C. Tenllado, and F. Tirado, "Gpu for parallel onboard hyperspectral image processing," *Int J High Perform Comput Appl*, vol. 22, no. 4, pp. 424–437, Nov. 2008. [Online]. Available: http://dx.doi.org/10.1177/1094342007088379

[18] J. Setoain, C. Tenllado, M. Prieto, D. Valencia, A. Plaza, and J. Plaza, "Parallel hyperspectral image processing on commodity graphics hardware," in *Parallel Processing Workshops, 2006. ICPP 2006 Workshops. 2006 International Conference on*, 0-0 2006, pp. 8 pp. –472.

[19] Y. Tarabalka, T. V. Haavardsholm, I. Kåsen, and T. Skauli, "Real-time anomaly detection in hyperspectral iimage using multivariate normal mixture models and gpu processing," *J Real-Time Image Proc*, vol. 4, no. 3, pp. 287–300, 2009.

[20] S. Sanchez and A. Plaza, "Real-time implementation of a full hyperspectral unmixing chain on graphics processing units," in *Proc. SPIE, Satellite Data Compression, Communications, and Processing VII*, 2011, pp. 81 570F–81 570F–9. [Online]. Available: +http://dx.doi.org/10.1117/12.892284

[21] S. Sanchez, R. Ramalho, L. Sousa, and A. Plaza, "Real-time implementation of remotely sensed hyperspectral image unmixing on gpus," *J Real-Time Image Proc*, 2012.

[22] *CUDA C Programming Guide*, October 2012, http://docs.nvidia.com/cuda.

[23] C. C. A. . Unported, "http://creativecommons.org/licenses/by/3.0/legalcode," visited 2014-09-03.

[24] V. Franc, V. Hlavac, and M. Navara, "Sequential coordinate-wise algorithm for the non-negative least squares problem," in *Computer Analysis of Images and Patterns, 11th International Conference, CAIP 2005, Versailles, France, September 5-8, 2005. Proceedings*, ser. Lecture Notes in Computer Science, vol. 3691, 2005, pp. 407–414.

[25] A. Bjorgan and L. L. Randeberg, "Real-time noise removal for line-scanning hyperspectral devices using a minimum noise fraction-based approach," *Sensors*, vol. 15, no. 2, 2015.