# Image-Based Modelling Using Nonlinear Function Fitting on a Stream Architecture

Karl E. Hillesland

January 26, 2004

## 1 Introduction

An important goal in computer graphics is to be able to synthesize photo-realistic images of objects and environments. The realism of the images depend on the adequacy of the model used for synthesis. *Image-based modelling* is an automated modelling technique aimed at achieving this realism by construction of models from photographs of real-world objects or environments.

A wide number of image-based modelling techniques have been proposed, each with its own technique for model generation. This is due to the fact that there is variation in the goals of the model and input data availability. However, we can treat all image-based modelling as a form of function fitting, where the function is our representation for appearance, and the data we want to fit are the photographs. This dissertation will address the use of nonlinear optimization to construct image-based models in general, including a new algorithm for efficient computation. This algorithm is designed to take advantage of programmable hardware.

The next section gives background on image-based modelling, and the approach that will be documented in the proposed dissertation. Following is a summary of previous work upon which the proposed dissertation is built. A brief summary of work completed to date is given, along with a schedule for remaining tasks. Most of the work has already been completed, and is documented in a siggraph paper [7], attached as appendix A.

## 2 Background

I will concentrate on image-based models that are spatially-varying and surface-based. *Surface-based* refers to a model that includes some geometric description of the object or environment, typically as a collection of triangles. *Spatially-varying* refers to models that take into account how appearance changes as we move from one point on the surface to another. I will refer to this class of models as SSIMs, for Surface-based, Spatially-varying, Image-based Models. Although SSIMs can be used for both objects and environments, the term“object” will be used to describe both.

A SSIM has an associated geometric description of the object to be modelled and a surface parametrization for one or more textures. The textures contain parameters for the appearance model. The appearance model is a function of the position on the surface of the object (surface parametrization), as well as other quantities such as viewing conditions and lighting conditions. These other quantities may also be stored in texture maps organized either by surface parametrization or some other parametrization, such as view direction.

Construction of the model typically consists of an iterative process in which the SSIM is used to synthesize an image corresponding to a photograph. The result of this comparison is used to update the model parameters.

To build a model, we start with the radiance images. Radiance samples are pixels in the images. In the problems we are interested in, there are typically hundreds to thousands of images, and a total of millions to billions of radiance samples.

The goal is to find the parameter values of the appearance model that best approximate the radiance data. A typical model will have 1M-10M parameters. Rather than trying to use a single model of this many parameters to fit all the data, the problem is often broken down into patches on the surface of the object and a local model is built for each patch independently. The number of patches varies from about ten thousand to a million, and the number of parameters per patch varies from tens to thousands. The size of the patch is a trade-off between the number of patches and the number of parameters for each patch.

As part of the model construction we must compute parameter dependency for each subproblem. This dependency is typically a set of matrices that describe the inter-dependency of the model parameters, as well as relations between the model parameters and the radiance samples. Examples of dependency data are the covariance matrix for PCA (principal component analysis), or first and second derivatives of the objective function in the case of optimization.

Note that the dependency data for each subproblem are a function of the radiance samples originating from all radiance images. The dependency data depend on all radiance data. Both radiance data and dependency data are often too large to fit in the computer memory. Therefore some sort of streaming of information from disk needs to take place. I describe two possible approaches next.

In the conventional approach, the distribution step is performed separately, where the radiance samples are used to compute the dependency data for all subproblems. Note that this process is closely related to rendering, and requires operations such as transformation, rasterization, and visiblity determination. Since the dependency data cannot, in general, fit into physical memory, the image data is processed multiple times, working on a subset of the patches as can fit into physical memory at a time. This part of the process can be quite time consuming in itself. The solution is then computed sequentially for each subproblem. The two-step approach outlined here can result in considerable disk utilization. We next turn to the approach proposed for this dissertation.

I propose an approach where the graphics system receives and processes the radiance data one image at a time, computing the parameter dependencies for that image on all subproblems simultaneously, and updating the solution for all patches simultaneously. This approach has its drawbacks. For example, since the radiance data are no longer distributed between individual subproblems, the data structures for all problems need to be active at all times. Additionally, this approach performs redundant computation at every iteration, since the same reordering and distribution of data samples is redone many times. However, benefits outweigh the limitations. First, the computational redundancy has almost no extra time cost. Second, there is no bottleneck of writing large quantities of data to disk or read-backs from the graphics hardware. Most importantly, all computations are performed using the graphics processing unit (GPU), resulting in a performance advantage.

# 3 Thesis

*Image-based models generated through nonlinear optimization can be computed efficiently using programmable graphics hardware as a streaming process.*

### 3.1 What is Meant by "Image-based Models"?

Although the framework presented might be applied to reconstruction of geometric information, this framework will only be applied to recovering surface appearance information. In fact, the framework is only intended for SIMMs.

### 3.2 What is Meant by "Nonlinear Optimization"?

The nonlinear optimization technique must be one that works well within the streaming framework proposed. For example, a full Newton's method would be impractical in terms of space requirements, which are $O(n^2)$ with respect to the number of parameters. Secondly, the nonlinear optimization technique will require continuous derivatives. However, the model may have simple bounds on model parameters. Third, testing is for functions for which an analytical expression for the gradient and second derivative information (Hessian) is given.

### 3.3 What is Meant by "Efficiently"?

In order to be useful, the algorithm must be scale well in terms of memory requirements, and it must show favorable performance relative to conventional methods on the CPU.

### 3.4 Demonstration of Thesis

The thesis will be demonstrated by a combination of theoretical and empirical study. The following is a list of tasks that will demonstrate the thesis.

- The framework is not necessarily tied to a particular nonlinear optimization technique. I will document an evaluation of various nonlinear optimization techniques given the criteria of image-based modelling, and the constraints of graphics hardware.
- I have implemented two nonlinear optimization techniques: steepest descent, and conjugate gradient. This is documented in Appendix A.
- The framework is intended to work for a number of image-based modelling techniques. I have implemented the system for two diverse types of SSIMs: the Spatial Bidirectional Reflectance Distribution Function (SBRDF) model of McAllister *et al.* [13] and the light field mapping (LFM) model of Chen *et al.* [4]. This is documented in Appendix A.
- A comparison has been made between this framework implemented on a graphics processor, and previous approaches on a CPU. Specifically, timings for the light field mapping approach under this framework have been compared against a CPU implementation of using nonlinear optimization. The graphics card was an ATI Radeon 9700 with 128MB of memory. The CPU was a 2 GHz Pentium 4 with a gigabyte of memory. The streaming framework on graphics hardware performed a factor of five better than the CPU implementation. This is documented in Appendix A. Updates to this comparison may become necessary to remain relevant at the time of the final defence.
- Analysis of error has been done to demonstrate the validity of the approach. The error is evaluated by comparing reconstructed images against the original images. Comparisons were made with respect to the light field mapping approach, where we can compare against the original

PCA-based approach, as well as a conventional CPU-based approach that uses nonlinear optimization. The nonlinear optimization approach actually resulted in slightly lower error than the PCA-based approach, probabaly due to more resampling in the PCA-based approach. Comparisons of the CPU at 32-bit floating point precision against the 16-bit GPU implementation showed higher, but comparable error for the GPU implementation. This analysis has been included in Appendix A.

- Further analysis of error will be conducted with respect to precision requirements of the application, and precision constraints of graphics hardware.

# 4 Achievements

Nonlinear optimization is an important tool for generating image-based models. This work explores how to generate an image-based model using nonlinear optimization. In particular, I cast nonlinear optimization techniques as a data-streaming process to compute image-based models. In completing this work, I plan to, or have already, achieved the following goals:

- The framework is applicable to a number of image-based modelling techniques.
- The framework is scalable.
- The framework enables fast computation of image-based models.
- The final product will include an evaluation of various nonlinear optimization techniques with respect to the criteria of image-based modelling and the use of graphics hardware.

Each of the above items is described in detail in the following subsections.

## 4.1 Diversity of Image-Based Modelling Techniques

As described in the introduction, all image-based modelling can be considered a nonlinear function fitting process. The framework has already been demonstrated for two image-based modelling techniques. The first [13] uses an analytic model that is fit to acquired data, and has an independent model for each texel of a texture map. The second [4] uses a compressed representation of the original data rather than an analytic model. In this case, there is an independent model for the ring of triangles around each vertex of the model. Therefore, the framework has been demonstrated on two classes of image-baesd modelling techniques, and two different “patch” sizes.

In general I would not expect a general nonlinear optimization technique to be more effective than a special-purpose algorithm designed for a specific image-based modelling technique. However, I believe an analysis of how nonlinear optimization may be applied to image-based modelling in general should provide some useful insights with broad application in image-based modelling.

## 4.2 Scalability

Image-based modelling involves processing of large data sets. Although the image data itself can be quite large itself, there are typically intermediate data that expand the size of the data set considerably. To meet this challenge, this work treats the image-based modelling process as a streaming process. The original images constitute the data in the stream. Any re-sampling is done on the fly, and only for a single image at a time.

### 4.3 Fast Computation

The computation to construct an image-based model can often be fairly time consuming. For example, the construction of an image-based model for re-lighting can take on the order of several hours or days for current techniques [13], [5]. This computation becomes an important bottleneck in the image-based modelling process. It is desirable to have some way to quickly evaluate the adequacy of the data captured, as it can be inconvenient, or even impossible, to return to a capture site after finding that there are data missing. With the ability to compute and evaluate a model in a timely manner, any missing or inadequate data can be identified while the object or environment is still readily available.

This dissertation will address speed in computation by showing how nonlinear optimization for image-based modelling can be formulated as a streaming process for efficient computation on programmable graphics hardware. This takes advantage of an important computational resource that is a component of nearly every computer system. I believe the streaming formulation is an important step towards real-time image-based model generation.

### 4.4 Evaluation of Nonlinear Optimization Techniques

In previous work, the nonlinear optimization technique adopted was typically the Levenburg-Marquardt or Powell's methods. There has been little to no documentation of the evaluation of other nonlinear optimization techniques, except those that were specifically designed for the particular image-based model in question. This dissertation will evaluate a number of nonlinear optimization techniques in the context of their use for image-based modelling. Results will be applicable to some aspects of linear and quadratic techniques as well. For example, the image-based model need only match the photographs to within the 8-bit precision of a color buffer channel. Graphics hardware is restricted to 32 bits of floating point precision, and performs much better in a 16 bit mode. Although the error analysis in Appendix A indicates that 16 bit precision has so far been adequate, I will explore the implications of these precision issues further. It's also important to take into consideration that geometric error in the model can make it impossible to achieve low error in the approximation.

Secondly, as the technique proposed in this dissertation is intended for implementation on programmable graphics hardware, these nonlinear optimization techniques will be evaluated in terms of how well they can be mapped to graphics hardware. Memory constraints are also important to consider for the sake of handling large data sets.

## 5 Previous Work

There are a number of image-based modelling techniques that have been developed, as well as a number of papers on how to use graphics hardware for non-graphics applications. This work fits somewhere between these two areas. This section begins by summarizing some previous work in image-based modelling and the use of the GPU for non-graphics applications, and concludes with previous work in which the GPU was used for image-based modelling applications.

### 5.1 Image-Based Modelling

Although there is a wide range of image-based modelling techniques, the SBRDF [13] and LFM [4] models were used to test the framework presented here. The SBRDF technique describes how light reflects at each point on a surface. Angular variation is described in terms of a nonlinear function developed by Lafortune *et al.* [10]. Spatial variation is encoded by storing per-texel parameters of

this model, which is a discrete sampling of the reflectance on the surface. The data is captured from photographs of a sample of material attached to an acquisition device developed by the authors. They present two methods to compute these parameters. The first method is the "Levenberg-Marquardt" method, a generic nonlinear optimization technique for nonlinear least squares fitting. Times reported for this method are on the order of several hours. The Levenberg-Marquardt method was also used in the original paper by Lafortune *et al.* [10] and another paper by Lensch *et al.* [11], which also use the the model of Lafortune *et al.* . The second method adopted by McAllister *et al.* is a special-purpose method designed to give a more approximate set of parameters in exchange for far less computation time. It gives results in time on the order of an hour. Note that these timings are for a single sample of material, and not a complete object or environment, which would require much more time. Both methods were implemented on a general-purpose processor. I am proposing to instead look at optimization techniques that can be run efficiently using programmable graphics hardware without resorting to special-purpose methods.

The light field mapping method [4] approximates a surface light field using matrix factorization. It requires an expensive resampling step that increases the complexity of the data processing pipeline and the amount of data to process. Matrix factorization cannot be easily mapped to streaming architectures. The optimization framework adopted here eliminates resampling of image data and can handle a broad class of function approximations.

The use of nonlinear optimization for image-based modelling is well established. Sato *et al.* [**?**] compute a fixed reflectance model from a set of images of an object captured under controlled lighting conditions by means of nonlinear optimization. The proposed method is closely tied to the specific reflectance model used in the paper. The method works only for objects made of a uniform material that can be approximated with the proposed reflectance model. Inverse global illumination [20] reconstructs the reflectance properties of a scene from a sparse set of photographs using an iterative optimization procedure that additionally estimates the inter-reflections between surfaces present in the scene. As with the earlier approaches, this method is limited to a predefined, low-parameter reflectance model that is not flexible enough to approximate complex surface material properties.

### 5.2 Non-traditional Use of Graphics Hardware

Several papers describing applications from non-graphics domains being ported to graphics hardware have appeared in recent years. Hoff *et al.* [8] use graphics hardware to compute Voronoi diagrams. Strzodka and Rumpf [17] implement multiscale methods for image processing using multipass rendering. Harris *et al.* [6] implement an explicit solver for dynamic systems represented as coupled map lattices and use it to simulate convection and diffusion.

There has also been recent interest in using the GPU for general numerical methods as well. Thompson *et al.* [18] implement a variety of non-graphics problems such as matrix multiplication and 3-satisfiability in current graphics architectures using a vertex unit. When compared against standard CPU implementations, they often demonstrate significant performance improvements. The recent advent of floaing point support in the fragment processing unit of graphics hardware has enabled further exploration of numerical computation on graphics hardware. Bolz *et al.* [1] implement two sparse matrix solvers using a fragment unit. Krüger and Westermann [9] develop linear algebra operators for implementing numerical algorithms on graphics hardware.

My work is related to this interest in the sense that I implement a widely-applicable numerical technique, nonlinear optimization. However, implementing a general numerical technique does not, by itself, solve a practical problem. In image-based modelling, one has to compute intermediate data before the more general technique can be applied. This framework includes this computation as well.

Second, previous work has focused on solving a single problem at once. In practice, it is often necessary to compute solutions for many problems. This framework uses the massive parallelism available in graphics hardware by taking advantage of the independence between these many problems.

The stream processing model for computation is not new. One copy of the IBM Stretch (IBM 7030) supercomputer was equipped with a streaming coprocessor called the IBM Harvest (IBM 7950) coprocessor in 1961 [2]. The principle was to set up a single operation to be performed on a long string of characters. A modern stream processor and applications are described by Rixner *et al.* [16]. Simulation is used to evaluate performance of some of these applications [14]. The applications share characteristics that make them suitable for their architecture. The applications are important computations in media processing: signal processing, video and image processing, and graphics. These applications share the characteristic that the same operations must be performed in an independent manner on a large number of data records with identical formats. Nonlinear optimization for SSIMs is similar in that the same operation must be preformed on identical data records, but the data is streamed through multiple times. However, we should still be able to take advantage of the computational efficiency of a stream processor.

Another example of a stream processor is a graphics processor. Although the graphics processor is designed for a particular application, that of realtime 3D graphics, it has become more programmable, approaching the flexibility of a general purpose stream processor. Purcell *et al.* [15] used this paradigm to describe how a ray tracer can be implemented on a graphics processor. Carr *et al.* [3] implement ray-triangle intersection as a fragment program and use it for a host of applications, such as ray tracing, photon mapping and subsurface scattering. The framework I propose follows this work in that nonlinear optimization is also cast as a streaming process. However, there are two differences. First, there is far more data that must be streamed from disk for image-based modelling than for the raytracing scenes adopted in Purcell's work. Second, the vertex engine, rasterization, and the depth buffer are also important in the context of image-based modelling on the GPU.

### 5.3 Use of the GPU in Image-Based Modelling and Computer Vision

In terms of using graphics hardware for image-based modelling and computer vision, there is the work of Yang *et al.* [19]. They used graphics hardware to quickly estimate geometry from images collected from five cameras using a plane-sweeping algorithm. In summary, they project image pixels of the five cameras onto a sequence of planes, finding the minimum color error to determine depth. Thus, their search process consists of trying all possibilities, and selecting the one that gives the minimum error. This is possible because their problem is essentially one dimensional; they are only concerned with depth. We, however, are concerned with higher dimensionality, where an exhaustive search of this nature is impractical. Yang *et al.* [**?**] also used graphics hardware in a Space Carving framework, where graphics hardware was used to resolve visibility and to create a look-up table for finding closest image pixels after projection.

Lensch *et al.* [12] developed an acquisition planning system to assist in the capture of the SBRDF of an object. Their system is able to make an intelligent choice of where to place the camera and light for each image after the first by minimizing an objective function representing the uncertainty of the model parameters. A portion of the objective function evaluation is computed on graphics hardware. As the system is used for light and view planning only, they have a very coarse texture atlas. Their objective function involves the use of a co-variance matrix, which is $O(n^2)$ in the number of parameters, and is stored only on the host processor. They use Powell's method and the Simplex method, which are non-gradient methods. Their work is related to mine in that they evaluate the gradient of a Lafortune model using graphics hardware, storing the results in a texture atlas. They

use a depth buffer comparison method, which is similar to my use of an item buffer for visibility determination with respect to updating the texture atlas for each choice of light and camera position.

## 6 Progress to Date and Remaining Tasks

Much of the work has already been completed. Results of applying the framework to two particular image-based modelling techniques [4] [13] were presented at Siggraph 2003 [7]. The paper is attached as Appendix A. Comparison between the proposed framework and a CPU implementation showed favorable results.

TIMELINE GOES HERE AND WILL INCLUDE THE FOLLOWING

- Summary of error in image-based modelling arising from visibility and measurement. This will be a summary of the work of others as much as possible.
- Investigation of classes of nonlinear optimization techniques beyond steepest descent and conjugate gradient for implementation under the proposed framework [7].
- Dissertation writing
- Oral Exam
- Defence

## 7 Oral Exam

The following is a list of possible topics for my Oral Exam. I have tried to identify persons responsible for each item.

- Nonlinear Optimization (Tolle)
- Image-Based Modelling and Rendering (McMillan)
- Graphics Hardware at an Architectural Level (Lastra)
- Implementation of Non-Graphics/Numerical Computation on Programmable Graphics Hardware (Manocha)
- Specific issues related to nonlinear optimization for image-based modelling under a streaming paradigm(?) (Grzeszczuk)

## References


[1] J. Bolz, I. Farmer, E. Grinspun, and P. Schröder. Sparse Matrix Solvers on the GPU: Conjugate Gradients and Multigrid. *ACM Transactions on Graphics*, 22(3), July 2003. (Proceedings of ACM SIGGRAPH 2003).

[2] W. Buchholz, editor. *Planning a Computer System.* McGraw-Hill, 1962.

[3] N. A. Carr, J. D. Hall, and J. C. Hart. Ray Engine. *2000 SIGGRAPH / Eurographics Workshop on Graphics Hardware*, pages 1–10, 2002.

[4] W-C. Chen, J-Y. Bouguet, M. H. Chu, and R. Grzeszczuk. Light Field Mapping: Efficient Representation and Hardware Rendering of Surface Light Fields. *ACM Transactions on Graphics*, 21(3):447–456, July 2002.

[5] R. Furukawa, H. Kawasaki, K. Ikeuchi, and M. Sakauchi. Appearance Based Object Modeling Using Texture Database: Acquisition, Compression and Rendering. *Eurographics Rendering Workshop 2002*, 2002.

[6] M. J. Harris, G. Coombe, T. Scheuermann, and A. Lastra. Physically-Based Visual Simulation on Graphics Hardware. *2002 SIGGRAPH / Eurographics Workshop on Graphics Hardware*, pages 1–10, 2002.

[7] K. E. Hillesland, S. Molinov, and R. Grzeszczuk. Nonlinear Optimization Framework for Image-Based Modeling on Programmable Graphics Hardware. *Proceedings of SIGGRAPH 03*, pages 925–934, July 2003.

[8] K. Hoff, T. Culver, J. Keyser, M. Lin, and D. Manocha. Fast Computation of Generalized Voronoi Diagrams Using Graphics Hardware. In *Proceedings of SIGGRAPH 99*, Computer Graphics Proceedings, Annual Conference Series, pages 277–286, 1999.

[9] J. Krüger and R. Westermann. Linear Algebra Operators for GPU Implementation of Numerical Algorithms. *ACM Transactions on Graphics*, 22(3), July 2003. (Proceedings of ACM SIGGRAPH 2003).

[10] E. P. F. Lafortune, S-C. Foo, K. E. Torrance, and D. P. Greenberg. Non-Linear Approximation of Reflectance Functions. *Proceedings of SIGGRAPH 97*, pages 117–126, August 1997.

[11] H. P. A. Lensch, J. Kautz, M. Goesele, W. Heidrich, and H-P. Seidel. Image-Based Reconstruction of Spatially Varying Materials. In *Twelveth Eurographics Rendering Workshop 2001*, pages 104–115. Eurographics, June 2001.

[12] Hendrik P.A. Lensch, Jochen Lang, Asia M. Sà, and Hans-Peter Seidel. Planned Sampling of Spatially Varying BRDFs. *Proceedings of Eurographics 2003*, 22(3), 2003.

[13] D. K. McAllister, A. Lastra, and W. Heidrich. Efficient Rendering of Spatial Bidirectional Reflectance Distribution Functions. *Eurographics Rendering Workshop 2002*, June 2002.

[14] John D. Owens, Ujval J. Kapasi, Peter Mattson, Brian Towles, Ben Serebrin, Scott Rixner, and William J. Dally. Media processing applications on the Imagine stream processor. In *Proceedings of the IEEE International Conference on Computer Design*, pages 295–302, September 2002.

[15] T. J. Purcell, I. Buck, W. R. Mark, and P. Hanrahan. Ray Tracing on Programmable Graphics Hardware. *ACM Transactions on Graphics*, 21(3):703–712, July 2002. ISSN 0730-0301 (Proceedings of ACM SIGGRAPH 2002).

[16] S. Rixner, W. J. Dally, U. J. Kapasi, A. L.-L. Khailany, P. R. Mattson, and J. D. Owens. A Bandwidth-Efficient Architecture for Media Processing. In *Proceedings of the 6th International Symposium on Microarchitecture*, November 1998.

[17] R. Strzodka and M. Rumpf. Nonlinear Diffusion in Graphics Hardware. *Proceedings EG/IEEE TCVG Symposium on Visualization*, pages 75–84, 2001.

[18] C. J. Thompson, S. Hahn, and M. Oskin. Using Modern Graphics Architectures for General-Purpose Computing: A Framework and Analysis. *Proceedings of 35th International Symposium on Microarchitecture (MICRO-35)*, 2002.

[19] R. Yang, G. Welch, and G. Bishop. Real-Time Consensus-Based Scene Reconstruction using Commodity Graphics Hardware. *Proceedings of Pacific Graphics*, 2002.

[20] Y. Yu, P. E. Debevec, J. Malik, and T. Hawkins. Inverse Global Illumination: Recovering Reflectance Models of Real Scenes From Photographs. *Proceedings of SIGGRAPH 99*, pages 215–224, August 1999.