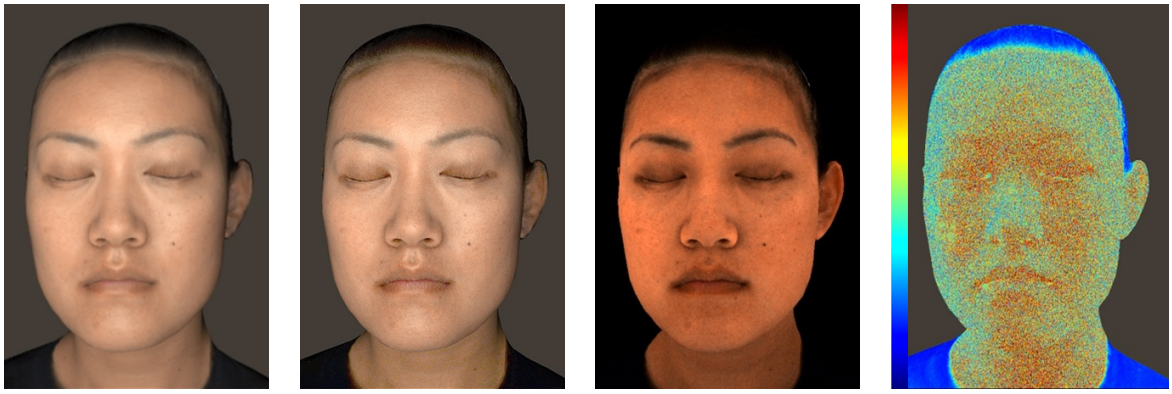
GPU-Based Inverse Rendering With Multi-Objective Particle Swarm Optimization

Koki Nagano* Thomas Collins Chi-An Chen Aiichiro Nakano
University of Southern California



(a) (b) (c) (d)

Figure 1: *(a) ground truth, (b) rendered image with (c) per-pixel estimated parameters, and (d) real-time error visualization in progress*

Abstract

We present a novel, GPU-accelerated per-pixel inverse rendering (IR) optimization algorithm based on Particle Swarm Optimization (PSO), IRPSO. IRPSO estimates the per-pixel scene attributes including reflectance properties of a 3D model, and is fast enough to do in situ visualization of the optimization in real-time. We utilize the GPU framebuffer as a computational domain, where each pixel is treated as an independent computational "swarm". We propose a parallel framework that recovers the reflectance at each pixel location in a massively parallel fashion. The algorithm's high parallel efficiency is demonstrated through our GPU/GLSL shader implementation of the method. IRPSO is validated experimentally on examples including simulated ground truth images.

1 Introduction

Photorealistic rendering of real-world objects has been a long-standing goal in computer graphics. It is a fundamentally difficult problem that requires a detailed representation of 3D scene geometry, a physically-accurate model of the reflectance properties of the materials in the scene, and a sophisticated light transport model. Together, we refer to these models as a *rendering model*. Given a rendering model and an instantiation of its parameters, *forward rendering* (FR) is the process of specifying the pixel color at each pixel coordinate in an image. When these parameters are known *a priori*, a host of deterministic and stochastic techniques exist to generate convincing renderings of 3D scenes. *Inverse rendering* (IR), on the other hand, is the process of recovering or estimating the unknown attributes of a scene (e.g, illumination, reflectance parameters) given a reference image such that the image generated by the rendering model matches the reference image as closely as possible. Once these parameters are recovered, they can be used to realistically render the 3D model with the material under novel lighting conditions, in novel camera views, and so on.

Current state of the art rendering techniques leverage 3D scanning techniques to record accurate 3D geometry as well as reflectance properties such as per-pixel surface orientation, surface and subsurface reflectance ([Ghosh et al. 2011] [Weyrich et al. 2006]) up to sub-milimeter resolution. However significant manual effort is still required to produce photorealistic renderings from the data mostly due to the measured reflectance data consisting only of a subset of the per-pixel reflectance properties necessary to create physically plausible renderings. Hence more research is necessary to investigate a semi-automatic method to estimate complete per-pixel reflectance properties in a way that scales up to such high resolution data. The primary difficulty inherent in this problem is the sheer number of parameters to be estimated, which is easily on the order of millions, necessitating a novel framework that can handle this computational burden in an efficient manner. In this work, we formulate the computational grid on a GPU framebuffer to exploit current graphics hardware to accelerate the computation in a massively parallel way. We propose a GPU-accelerated, inverse rendering algorithm, called IRPSO, which estimates the per-pixel reflectance parameters of a realistic rendering model using an optimization approach based on Particle Swarm Optimization (PSO) [Eberhart and Kennedy 1995].

2 Related Work

2.1 Inverse Rendering

A significant body of work has been done in the category of image-based techniques. Given photographs, and rendering models that define how the light interacts with the scene, unknown scene geometry [Debevec et al. 1996], reflectance parameters, and/or lighting [Marschner 1998] can be solved, even under global illumination [Yu et al. 1999]. The quality and detail of the acquired unknown parameters is largely limited by the quality of the measured data, the fidelity of the presumed rendering model, and known attributes of the scene. Since it is very costly to simulate the entire light

*e-mail:knagano@usc.edu

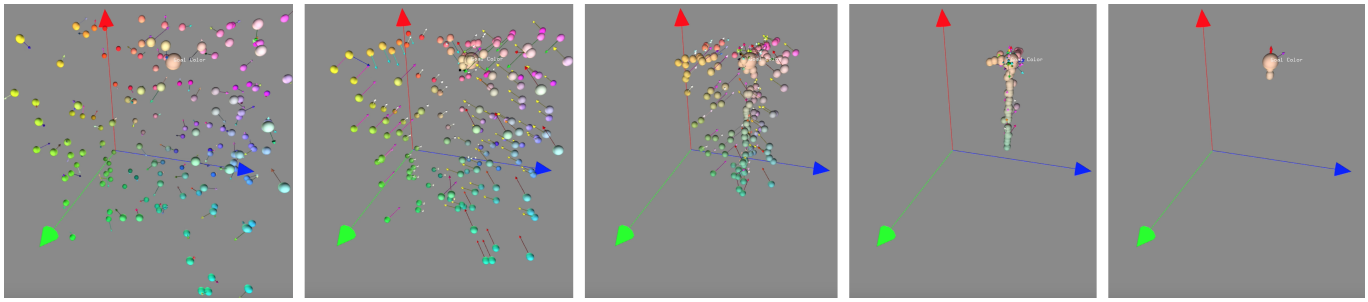


Figure 2: *A visualization of the PSO algorithm optimizing in three-dimensional RGB space converging to the color (labeled "Goal Color") that minimizes the fitness function. The arrows associated with each particle are the velocity vectors.*

transport process (including global illumination), entire global illumination may not be considered and/or people have been using techniques to separate the response of the light on the material to bring down the complexity of the light transport model necessary to solve for the unknown parameters such as with spherical harmonics [Ramamoorthi and Hanrahan 2001]. People also employed a gradient based optimization including [Donner et al. 2008] that estimated per-pixel subsurface reflectance. However they are limited to a subset of surface and subsurface reflectance parameters, unable recover spatially varying parameters, and/or the result is shown on a rather small patch. It is still non-trivial to scale up the existing method to acquire per-pixel reflectance with the entire material – which requires estimating or optimizing on the order of millions of points with potentially high dimensional parameters – present in real-world materials, and some research needs to be done in efficiently optimizing such large numbers of parameters.

2.2 Particle Swarm Optimization

Particle Swarm Optimization (PSO) [Eberhart and Kennedy 1995] is a swarm-based optimization algorithm that has been shown to be effective in solving difficult optimization problems in many diverse domains [Sun et al. 2011; Poli 2008]. The basic idea of PSO is that a swarm of m particles, each n-dimensional, perform an independent search in the space of possible n-dimensional solutions. Each particle i is a point x_i in this search space with a certain velocity v_i and has an associated fitness given by the objective function (F, to be minimized) value at that point $F(x_i)$. Particles move around in this search space randomly with sampling biased toward a random weighted average of the best (lowest $F(x_i)$) position achieved by any particle in the swarm g and the best position achieved by each particle individually, p_i. This focuses random searches on areas of the search space where a global optimum is expected to be. g and p_i are updated at each iteration.

This randomized searching process continues until a certain fitness threshold h is reached by some particle in the swarm (i.e., a low enough value of F is found) or a maximum number of iterations N is performed. At termination, the global best position found (g) is returned. Fig. 2 visualizes the particles converging to the solution with the PSO algorithm.

Many real-world problems are most readily described as multi-objective optimization problems, in which a set of objective functions must be simultaneously minimized in the same search space. Such problems can be solved using a Multi-Objective version of Particle Swarm Optimization (MOPSO).

Many different MOPSO frameworks have been proposed in order to manage the increased computational and mathematical difficulty that comes from trying to minimize many (possibly conflicting) objective functions simultaneously. For our purposes, the most relevant MOPSO approaches are those that devote one swarm to optimizing each objective function (Vector-evaluated MOPSO; [Chow and Tsui 2004; Parsopoulos and Vrahatis 2004]) as they naturally lend themselves to parallelism. An overview of the literature on MOPSO is given in [Parsopoulos and Vrahatis 2008].

To the best of our knowledge, there is no previous work regarding the application of PSO or MOPSO to the inverse rendering problem. However, there is considerable experimental evidence that PSO and MOPSO are powerful optimization frameworks that provide high-quality solutions to very difficult optimization problems. We detail a massively parallel MOPSO solution to the IR problem in the next section.

3 Multi-Objective PSO for Inverse Rendering

3.1 Algorithm Overview

We can view model data T to be estimated stored in a texture as an n dimensional $width \times height$ array of pixels, with n being the number of unknowns per pixel. The reference pixel value (i.e. pixel color) stored in a vector corresponding to pixel (i, j) in T is denoted $c_{[i,j]}$, which we assume can be reasonably approximated by rendering model R. The reference pixel $c_{[i,j]}$ is given as *known* together with known camera intrinsic and extrinsic parameters which map the coordinates between the model and the reference camera if necessary. R has q parameters, which, when instantiated, determine the rendered image. $r \leq q$ of these parameters are known at each pixel, while the remaining $q - r$ parameters are unknown at each pixel. Denote the set of known and unknown rendering model parameters $\hat{\mathbf{p}}_{[i,j]}$. Then we have that $R(\hat{\mathbf{p}}_{[i,j]}) = \hat{c}_{[i,j]}$.

The inverse rendering problem is that of recovering the unknown $q - r$ parameters in $\hat{\mathbf{p}}_{[i,j]}$ at each pixel that minimize the total amount of error between the generated image and the ground truth image. This reduces to solving the following minimization problem at each pixel:

$$argmin_{\hat{\mathbf{p}}_{[i,j]}} \|\hat{c}_{[i,j]} - c_{[i,j]}\| \quad (1)$$

The multi-objective optimization problem at hand is to find a set of $n \cdot height \cdot width$ parameters that simultaneously minimizes the errors at each pixel. We assume that each pixel's parameters can be optimized separately using PSO, but allow for interaction between spatially close swarms using a special term in the PSO fitness function (described below).

The IRPSO algorithm (illustrated in Fig. 3) begins by initializing a swarm of m particles associated with each pixel(i, j). Once the particle swarms have been initialized, the following procedure is

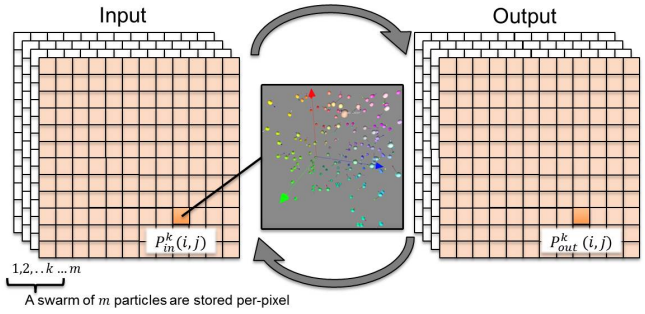


Figure 3: *A visual representation of the IRPSO framework.*

repeated until some stopping criterion is met (number of iterations, fitness threshold, etc.):

1. In parallel, a random particle from each swarm at each pixel $P^k_{in}(i,j)$ with $k(\leqslant m)$ being the particle id is selected and its position and velocity are updated to the associated data structure with $P^k_{out}(i,j)$ according to equations (2) and (3) below.
2. In parallel, the personal best particles and global best particles for each swarm are updated to $P^k_{out}(i,j)$ as in the original single-objective PSO algorithm.
3. Once all threads are synchronized, return to step 1.

Though it may seem that this is simply a parallel evaluation of single-objective PSO procedures, it is truly an MOPSO approach as we use the following velocity and position update equations, which modify those in [Chow and Tsui 2004] to operate over a regular grid of particle swarms:

$$v^l_{[i,j]_k,t+1} = v^l_{[i,j]_k,t} + c_1 R^l_{1[i,j]_k,t}(p^l_{[i,j]_k,t} - x^l_{[i,j]_k,t}) + c_2 R^l_{2[i,j]_k,t}(g^l_{[i,j]_t} - x^l_{[i,j]_k,t}) + c_3 A^l_{[i,j]_k} \quad (2)$$

$$x^l_{[i,j]_k,t+1} = x^l_{[i,j]_k,t} + v^l_{[i,j]_k,t+1} \quad (3)$$

In Equation 2, we update each dimension l of each n-dimensional particle k associated with pixel (i,j) at time t in order to generate a new velocity at time $t+1$. c_1, c_2 and c_3 are constants (algorithm parameters, where $c_1 \approx c_2$ and $c_3 << c_1$), $R^l_{1[i,j]_k,t} \sim U(0,1)$, and $R^l_{2[i,j]_k,t} \sim U(0,1)$. The term $A^l_{[i,j]_k}$ at the end of Equation 2 is defined as follows:

$$A^l_{[i,j]_k} = \left(\frac{1}{(H)^2}\sum_{m=0}^{H-1}\sum_{n=0}^{H-1}(g^l_{[i+(m-H/2),j+(n-H/2)]_t})\right) - x^l_{[i,j]_k,t} \quad (4)$$

This term effectively biases the solution at each pixel (i,j) toward the average of the current global best positions of neighboring particle swarms in an $H \times H$ window centered at swarm (i,j). By increasing or decreasing H, users can determine the range of influence each swarm has on neighboring swarms during optimization.

3.2 Implementation

In this work, we chose the GLSL shader to be the experimental platform to demonstrate the generality of our computational framework. The same algorithm can be implemented in an advanced compute shader, CUDA, OpenCL, and so on depending on applications. Taking advantage of the fact that we wish to associate a particle swarm with each pixel, we can use OpenGL textures to store PSO data vectors and values on a per-pixel (and thus per-swarm) basis. Assuming that each swarm has the same number of particles, we can store the position of one particle of each swarm in a $width \times height$ texture of 3D vectors. By allocating m position textures, each consisting of $width \times height$ 3D vectors, we can create a swarm of m particles per pixel. We can use this same data layout to store the velocities, best positions, and best fitnesses associated with each particle in m textures each. Similarly, the global best fitness of each swarm and the global best particle of each swarm can be stored in one texture each, as they are shared amongst all the particles in a swarm. We can then perform steps 1 and 2 of the algorithm described above in GLSL shaders during each rendering loop. Step 3 is performed by OpenGL automatically as the GPU threads are synchronized before the next rendering loop is completed.

4 Synthetic Target Image Preparation and Results

In order to verify the IRPSO algorithm, we tested it on synthetic ground truth images. In this way, we knew the values of the parameters to be recovered at each pixel and could accurately gauge the quality of the solutions produced by IRPSO.

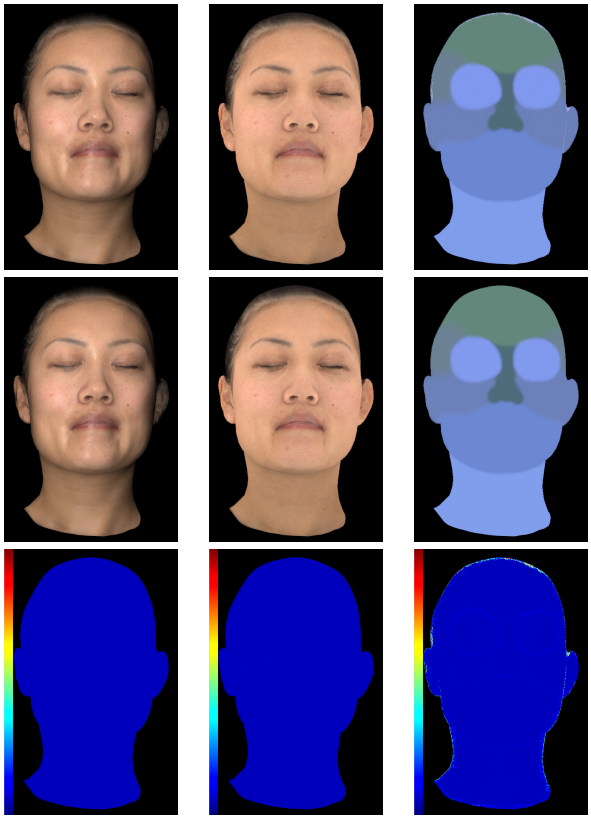
The synthetic ground truth images were generated using a high-quality 3D mesh of a human head as well as per-pixel surface orientation maps and surface and subsurface albedo maps captured by a spherical LED scanning device [Ghosh et al. 2011]. Then the head model was rendered with known camera parameters using GLSL shaders, evaluating diffuse and/or specular reflections under a point light source(s) using captured per-pixel reflectance maps, and per-pixel color is recorded in the ground truth image.

Fig. 1 illustrates the real-time IRPSO visualization in progress that minimizes the error (d) between the ground truth (a) and the rendering (b) with estimated parameters (c).

Fig. 4 shows the comparison between our per-pixel estimation (top row) and ground truth (middle row) with error visualization (bottom row) with full composite render (first column), diffuse color (second column), and surface roughness with a two lobe Cook-Torrance model [Cook and Torrance 1982] (third column). The ground truth diffuse color is acquired from [Ghosh et al. 2011], and the ground truth roughness parameters are manually painted based on measurement with RGB channels storing two roughness values, and a convex weight between the lobes. Though some error is still visible in the error visualization in Fig. 4 (c) around the edges due to the grazing angle, it can be fixed employing multi-view data.

5 Conclusions and Future Work

In this paper, we presented a novel, GPU-accelerated inverse rendering algorithm based on Vector-evaluated Multi-objective Particle Swarm Optimization. The use of a parallelized Vector-evaluated PSO based algorithm was motivated both by PSO and MOPSO's proven track record in the literature of efficiently solving (to a high-quality approximation) difficult real-world optimization problems as well as the results of a suite of tests we performed on the University of Southern California's High Performance Computing Center cluster. These tests illustrated the strong parallel efficiency of PSO and Vector-evaluated PSO even on very large swarms of particles distributed over a large number of computing nodes.



(a) (b) (c)

Figure 4: *A comparison between IRPSO optimized best guess results (top row) and ground truth (middle row) showing a maximum of less than 1 % error (bottom row).*

We validated the IRPSO algorithm using a realistic, high-quality 3D model of a head and a complex skin texture. IRPSO was successfully able to recover both per-pixel diffuse color reflectance parameters and per-pixel surface roughness while interactively visualizing the optimization. The composite optimized best-guess images were visually indistinguishable from the ground truth images after only a few seconds of optimization time, as is clearly visible in the figures above. For the next step, we wish to apply our method to real data sets in order to observe its real-world performance. It would be also of interest to compare the PSO algorithm with other optimization methods such as a gradient based method in terms of the number of parameters, and the smoothness of the target function.

GLSL implementation allowed for a straightforward texture-based implementation of grids of particle swarms. It is also an interesting example of large scale, general purpose computation in GLSL shaders. However, as a next step, we wish to make this method more general by developing a CUDA or OpenCL implementation. This would eliminate some of the restrictions present in the GLSL implementation and would likely lead to better overall performance.

Acknowledgements

This work was developed from a final project for CSCI 596 (scientific computing and visualization) course at the University of Southern California. Integration of research and education in CSCI 596 was supported by the National Science Foundation, Grant # 1508131.

References

CHOW, C.-K., AND TSUI, H.-T. 2004. Autonomous agent response learning by a multi-species particle swarm optimization. In *Evolutionary Computation, 2004. CEC2004. Congress on*, vol. 1, IEEE, 778–785.

COOK, R. L., AND TORRANCE, K. E. 1982. A reflectance model for computer graphics. *ACM TOG 1*, 1, 7–24.

DEBEVEC, P. E., TAYLOR, C. J., AND MALIK, J. 1996. Modeling and rendering architecture from photographs: A hybrid geometry- and image-based approach. In *Proceedings of SIGGRAPH 96*, Computer Graphics Proceedings, Annual Conference Series, 11–20.

DONNER, C., WEYRICH, T., D'EON, E., RAMAMOORTHI, R., AND RUSINKIEWICZ, S. 2008. A layered, heterogeneous reflectance model for acquiring and rendering human skin. In *ACM TOG (Proceedings of SIGGRAPH Asia)*.

EBERHART, R., AND KENNEDY, J. 1995. A new optimizer using particle swarm theory. In *Micro Machine and Human Science, 1995. MHS '95., Proceedings of the Sixth International Symposium on*, 39–43.

GHOSH, A., FYFFE, G., TUNWATTANAPONG, B., BUSCH, J., YU, X., AND DEBEVEC, P. 2011. Multiview face capture using polarized spherical gradient illumination. In *Proceedings of the 2011 SIGGRAPH Asia Conference*, ACM, New York, NY, USA, SA '11, 129:1–129:10.

MARSCHNER, S. 1998. *Inverse Rendering for Computer Graphics*. PhD thesis, Cornell University.

PARSOPOULOS, K. E., AND VRAHATIS, M. N. 2004. On the computation of all global minimizers through particle swarm optimization. *Evolutionary Computation, IEEE Transactions on 8*, 3, 211–224.

PARSOPOULOS, K. E., AND VRAHATIS, M. N. 2008. Multiobjective particles swarm optimization approaches.

POLI, R. 2008. Analysis of the publications on the applications of particle swarm optimisation. *J. Artif. Evol. App. 2008* (Jan.), 4:1–4:10.

RAMAMOORTHI, R., AND HANRAHAN, P. 2001. A signal-processing framework for inverse rendering. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, ACM, New York, NY, USA, SIGGRAPH '01, 117–128.

SUN, J., LAI, C., AND WU, X. 2011. *Particle Swarm Optimisation: Classical and Quantum Perspectives*. Chapman & Hall/CRC Numerical Analysis and Scientific Computing Series. Taylor & Francis.

WEYRICH, T., MATUSIK, W., PFISTER, H., BICKEL, B., DONNER, C., TU, C., MCANDLESS, J., LEE, J., NGAN, A., JENSEN, H. W., AND GROSS, M. 2006. Analysis of human faces using a measurement-based skin reflectance model. *ACM Transactions on Graphics 25*, 3 (July), 1013–1024.

YU, Y., DEBEVEC, P., MALIK, J., AND HAWKINS, T. 1999. Inverse global illumination: recovering reflectance models of real scenes from photographs. In *SIGGRAPH '99*, 215–224.