On-line Learning of Parametric Mixture Models for Light Transport Simulation: Brief Documentation

July 10, 2014

Contents

1 Introduction

This document contains a brief documentation of the *LibImportance* library that is an implementation of the importance sampling method described in the paper [Vorba *et al.*, 2014]. The method learns about the rendered scene in a preprocessing pass and uses the obtained information about important directions during rendering to guide the path-sampling process. The information about radiance/importance distribution is stored on scene surfaces in form of directional distributions trained during the training phase. The distributions are represented by the Gaussian mixture model. The LibImportance library can be attached to *any physically-based renderer*. This demo package takes advantage of *Mitsuba* renderer in its version 0.5.0.

Although the LibImportance library provides distributions of both radiance and importance (a quantity emitted by the camera), that can be used to guide path-sampling in bidirectional light transport methods, this demo contains only an implementation of guided path tracing and guided progressive photon mapping. The respective integrators are named as `guided_path`, `guided_ppm` and `guided_sppm` and are directly derived from the original implementations of `path`, `ppm` and `sppm` integrators.

Note that even though the implementation of the `guided_path` integrator is capable of handling participating media, the code was never tested on any scene containing media.

An integral part of the LibImportance library is also a *visualization* tool of trained distributions in the scene.

2 Licence

3 Platform

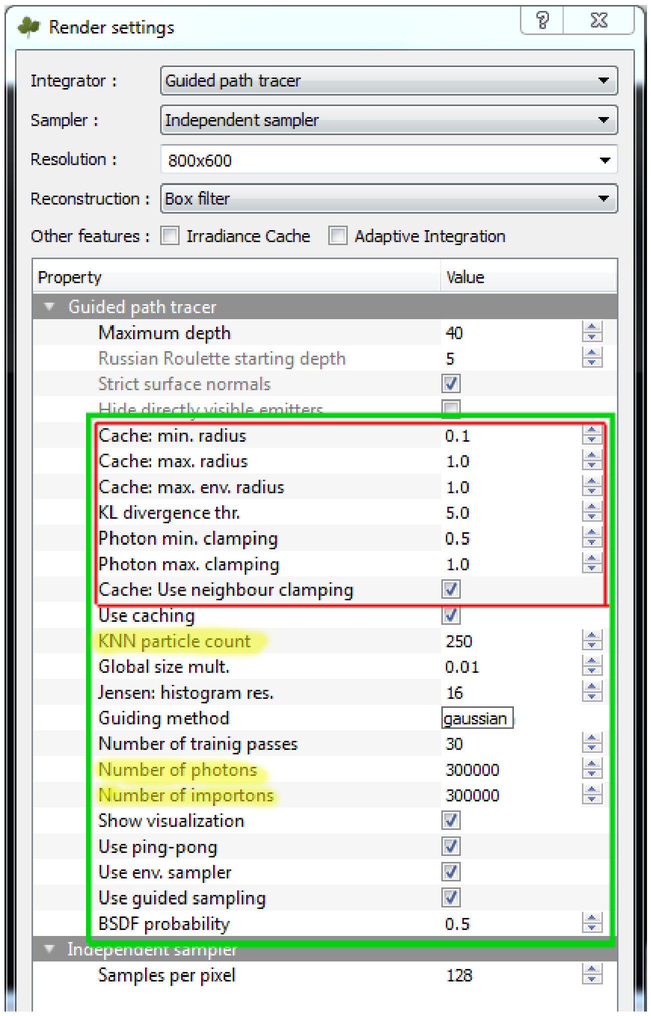
The LibImportance library and also this demo package were developed and tested only under Windows 7 x64. We compiled the code using MSVC 2010.

4 Settings

Figure 1 shows the parameters that are common for all guided integrators (the green frame). There are cache related parameters that have direct impact on spacing of cached distributions in the scene. For most scenes they can be left on its default values. These parameters are framed in red.

The parameters that usually need to be changed are highlighted in yellow. These are particle related parameters that influence the efficiency of the guiding method and they have also impact on cache spacing since it is bounded by multiplies of a particle search radius. The rest of the parameters can usually be left on their default values.

The parameters and their meaning are as follows:


Render settings

Integrator : Guided path tracer

Sampler : Independent sampler

Resolution : 800x600

Reconstruction : Box filter

Other features : ☐ Irradiance Cache ☐ Adaptive Integration

Property	Value
Guided path tracer	
Maximum depth	40
Russian Roulette starting depth	5
Strict surface normals	✓
Hide directly visible emitters	☐
Cache: min. radius	0.1
Cache: max. radius	1.0
Cache: max. env. radius	1.0
KL divergence thr.	5.0
Photon min. clamping	0.5
Photon max. clamping	1.0
Cache: Use neighbour clamping	✓
Use caching	✓
KNN particle count	250
Global size mult.	0.01
Jensen: histogram res.	16
Guiding method	gaussian
Number of trainig passes	30
Number of photons	300000
Number of importons	300000
Show visualization	✓
Use ping-pong	✓
Use env. sampler	✓
Use guided sampling	✓
BSDF probability	0.5
Independent sampler	
Samples per pixel	128

Figure 1: The settings of guided path tracer in Mitsuba GUI. The guiding-related settings are framed in green. The cache related settings that can be left on its default values for most of the scenes is framed in red. The only parameters that should be set carefully per scene are highlighted in yellow.

- **Cache: min. radius**

  Limits the allowed validity radius of a cached distribution. The smaller the value the denser the distribution cache could become.

- **Cache: max. radius**

  Limits the allowed validity radius of a cached distribution. The higher the value the less dense the distribution cache could become.

- **Cache: max. env. radius**

  Limits the maximum allowed validity radius of a cached distribution for sampling the starting position of a photon emitted from the environment.

- **KL divergence thr.**

  The sensitivity of the distribution validity radius to changes in the equilibrium radiance/importance. The higher the value the more sensitive caching scheme (i.e. the cache could become denser in caustics and near the surfaces that are sources of reflections). Please, refer to the paper for details.

- **Photon min. clamping**

  Restricts the size of the validity radius of a distribution. The value is a multiply of a particle-search radius. The smaller the value the smaller the validity radius could become.

- **Photon max. clamping**

  Restricts the size of the validity radius of a distribution. The value is a multiply of a particle-search radius. The higher the value the bigger the validity radius could become.

- **Cache: Use neighbour clamping**

  If `true` then the validity radius of a distribution is clamped by values of its neighbouring distributions.

- **Use caching**

  Switches the caching of distributions on/off. If caching is switched off then a distribution is always trained anew when requested at a scattering position. This is only meant for debugging.

- **KNN particles count**

  Number of nearest neighbour particles from a distribution position that are searched for when the distribution is created or progressively trained later on.

- **Global size mult.**

  This parameter influences the size of maximum validity radius of a distribution in the cache and maximum particle search radius. It is given in multiplies of the scene bounding box diagonal.

- **Jensen: histogram res.**

  This parameter has an effect only when **Guiding method** is set to `Jensen`. It is the resolution of one side of a histogram grid (it is mapped on hemisphere).

- **Guiding method**

  Possible values are `jensen`, `hey`, `pharr`, `gaussian`. The last one is the method thoroughly described in our paper [Vorba *et al.*, 2014]. For brief description of the other methods, please, refer also to the paper. Note that Jensen's, Hey's and Pharr's method were not designed to allow progressive training. While we implemented Jensen's method to simply merge the particles into the existing histograms and thus made it deceivingly progressive, the other two methods will crash the application if the `Number of t.passes` will be greater than one and the `Use ping-pong` is set to true.

  Also note that we designed the caching originally only for our Gaussian mixtures. That means that the parameter **KL divergence thr.** which determines the sensitivity of caching scheme to the changes of cached quantity (radiance/importance) has no effect when the other distribution models are used. The spacing of distributions is then based solely on the particle search radius.

- **Number of training passes**

  Number of training passes. One training pass consists of tracing photons followed by tracing importons and updating of cached distributions of radiance and importance.

- **Number of photons**

  Number of emitted photons per one training pass.

- **Number of importons**

  Number of emitted importons per one training pass.

- **Show visualization**

  If `true` then cached distributions and particles from the last training pass are visualized when the rendering pass is done.

- **Use ping-pong**

  If `false` then no importons are traced and there is only one (unguided) photon tracing during the training phase. Thus, in fact, the actual training of distributions does not start before the rendering phase. Also, there is no progressive training.

- **Use env. sampler**

  Switches on/off the guided sampling from the environment light source (if one is present in the scene).

- **Use guided sampling**

  If `false` then no parameter in the green frame (see Figure 1) has an effect. The integrator falls back to its original unguided implementation.

- **BSDF probability**

  Probability that a direction will be sampled from BSDF and not from a guiding distribution if one is available at a scattering position. These two strategies are combined via MIS.

5 Compilation

1. Download the source code (`http://cgg.mff.cuni.cz/~jirka/papers/2014/olpm/olpm2014_source.zip`)
2. Unzip the `olpm2014_source.zip` archive to an empty folder.
3. Install Python
4. Install Scons (`http://www.scons.org/`) (make sure it is compatible with your Python version)
5. Make sure that Scons is in your system Path variable
6. Open the Visual Studio project `"Mitsuba 0.5.0/build/mitsuba-msvc2010.sln"`
7. Select Release or Debug configuration on platform x64
8. Press F7

If it does not work for you, please send an email to jirka@cgg.mff.cuni.cz.

6 Implementation

This demo package has two layers. One layer is the LibImportance library that can be attached to any physically-based rendering system written in C++ and the other enables the usage of the library in Mitsuba renderer.

The former contains the Expectation-Maximization based learning algorithms, caching, guided sampling API (including environment guiding), and a visualization tool. The latter involves training phase implementation, one-sample estimator combining BSDF and guided sampling via MIS, implementation of LibImportance library environment light adapter and particles adapter.

6.1 LibImportance

The LibImportance library source files are all located in the **LibImportance** directory. The library is attached to Mitsuba through including the LibImportance.h file and linking the LibImportance.lib that is compiled together with Mitsuba (see Section 5).

Here, we list the main LibImportance subdirectories and briefly describe their content:

- **caching**

  The octree structure to accelerate search for suitable distribution, lazy caching of guiding distributions (CachedSampler.h)

- **em**

  The off-line and on-line stepwise EM algorithms (stepwise_emsse.h)

- **gaussian**

  The Gaussian mixture model (GMM) implementation, the EM update formulae for GMM (gaussian_stepwise_emsse.h)

- **hey**

  Hey and Purgathofer's method implementation.

- **jensen**

  Jensen's histogram method implementation.

- **pharr**

  Pharr's method implementation.

- **sampler**

  An interface for sampling from a guiding distribution and its progressive training.

- **shared**

  Basic components of LibImportance library such as vectors, SSE support, point kdtree for searching the nearest particles, mappings etc.

- **viz**

  The visualization tool based on the openGl and the wxWidgets libraries.

6.2 Mitsuba Layer

To use the LibImportance library in Mitsuba we implemented the training phase (guiding.h), one-sample estimator combining BSDF and guided sampling via MIS (guided_brdf.h), Russian roulette according to particle weights, guided particle tracing (based on Mitsuba implementation) (guided_particletracing.h/cpp) and particle and environment light adapters (libImpUtils.h).

We list all the files that were added or modified (in Mitsuba) in order to use the LibImportance library:

- emitter.h (small modification)
- guided_brdf.h
- guided_particletracing.h
- guiding.h
- guiding_config.h
- libImpUtils.h
- libImpVizUtils.h
- nondiscretphoton.h (small modification of original photon)
- envmap.cpp (added guided emission)
- guided_path.cpp
- guided_ppm.cpp
- guided_sppm.cpp
- guided_particletracing.cpp
- nondiscretephoton.cpp (small modification of original photon)
- scene.cpp (small modification)

7 Visualization

It is possible to visualize cached distributions in the scene when the rendering is finished (see Section 4). It is also possible to visualize the particles stored during the last tracing step. The first window that appears shows radiance distributions and photons while the second window shows importrance distributions and importons. If the environment light source is present in the scene and the emission guiding is switched on the second window also shows the distributions that guide the emission of a particle start position. The visualization tool is shown in Figures 2-4.

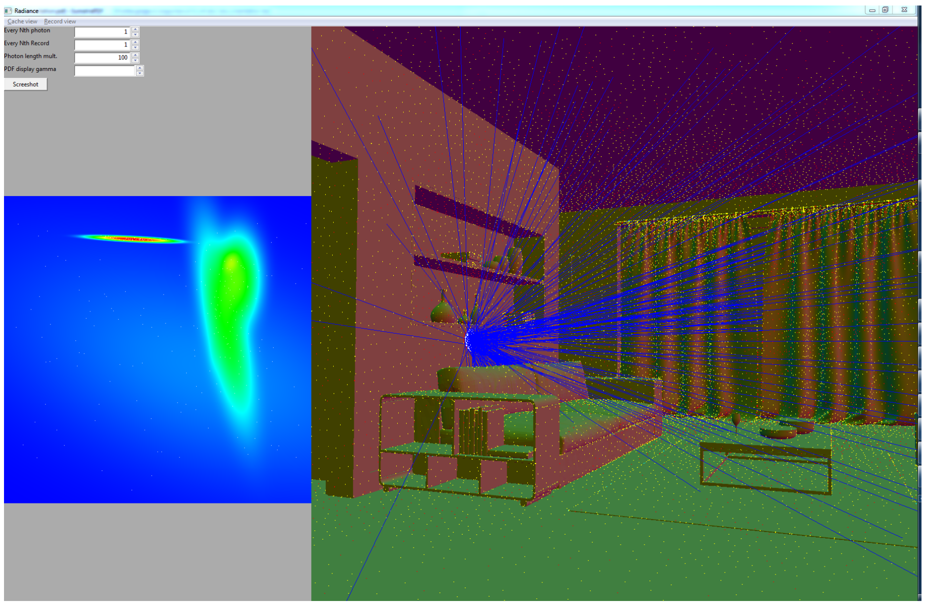
The basic controls are as follows:

1. **Middle button**

   Moves the camera in the camera plane.

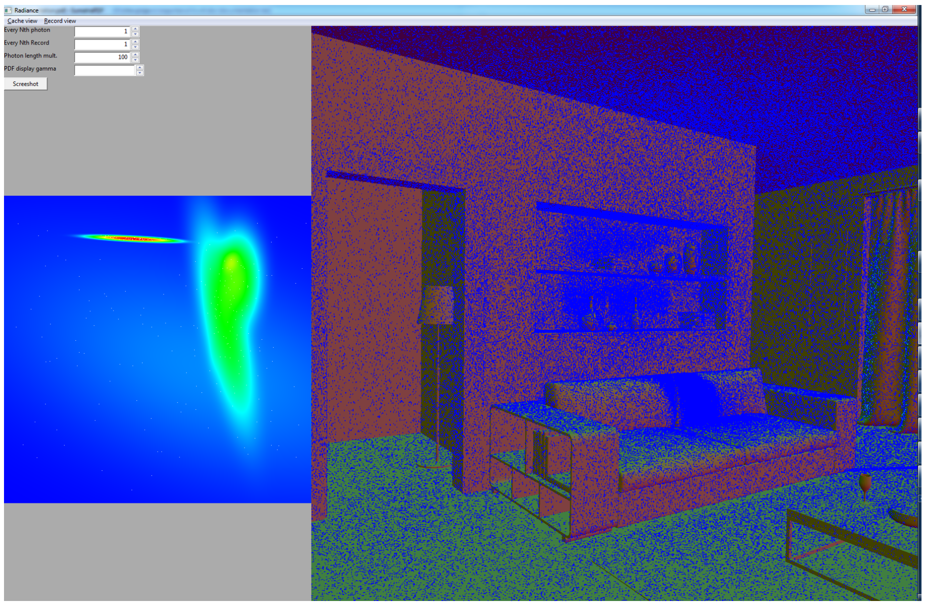
2. **Mouse wheel or Middle button + Ctrl**

   Moves the camera forwards/backwards.



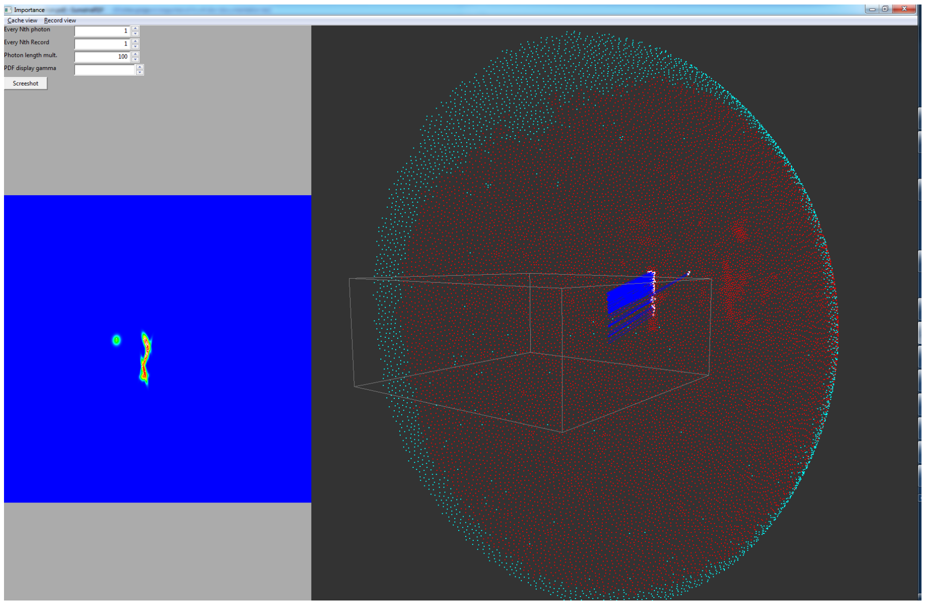
Every Nth photon 1
Every Nth Record 1
Photon length mult. 100
PDF display gamma

Figure 2: Visualization of cached radiance distributions in the scene. The distributions are colorful dots on surfaces. Red and yellow is used for distributions that are considered reliable (i.e. were trained from a sufficient number of particles) while the light blue is used for distributions that were trained using only a few particles. The geometry is colored according to the normals. The nearest photons to the point of interest are visualized by rays in directions of their incidence. The trained distribution is visualized in false color on the left (together with nearest particles coming from the last training phase).



Every Nth photon 1
Every Nth Record 1
Photon length mult. 100
PDF display gamma

Figure 3: Visualization of photons from the last training phase.



Every Nth photon 1
Every Nth Record 1
Photon length mult. 100
PDF display gamma

Figure 4: Visualization of distributions for sampling the photon start position together with importons coming from a selected direction. The grey box is the scene bounding box.

3. **Shift**

   Accelerates the movement in previous cases.

4. **Middle button + Alt or Right button**

   Rotates the camera.

5. **Left button**

   Selects a distribution (only distributions on surfaces with normals towards the view direction can be selected).

6. **f**

   Resets the camera.

7. *****

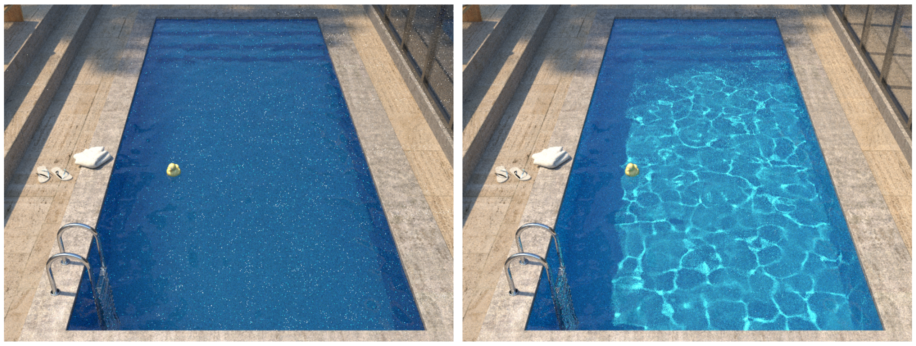
   Switch between visualizing all the nearest neighbour particles and all the nearest neighbour particles that can be used for fitting based on particle and surface normals. The difference can be noticed only if `Record view` > `Interpolate on click` is on and a distribution is selected (ideally one near corners).

8 Scenes

Part of this demo package is the **Door** scene that was presented in our paper [Vorba *et al.*, 2014]. The scene is a recreation of the well-known scene from Veach and Guibas' paper [1997] provided by Lehtinen et al. [2013]. Note, that we further modified the scene that is now even more difficult to render. To make the scene more realistic, we have made the area light source much smaller and used more (40) light bounces. Further, the materials were replaced by Ashikhmin-Shirley BRDF implementation that we were using in Corona Renderer so that the results were completely the same in Corona and Mitsuba - however the overall appearence was preserved.

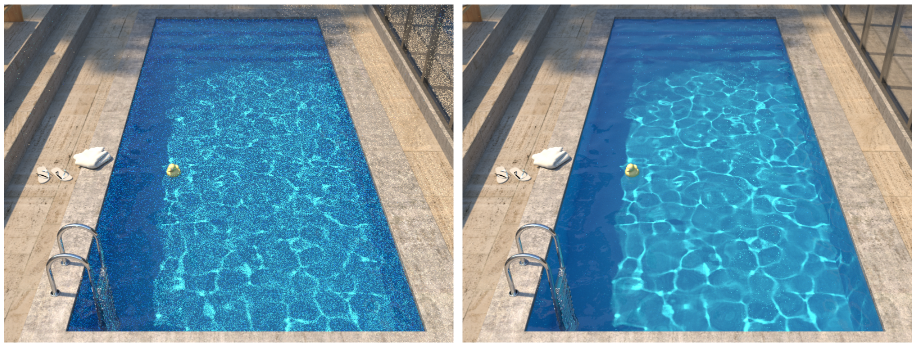
If you would like to try out other scenes from the paper, please, send an email to jirka@cgg.mff.cuni.cz. We were not given a permission by the scene authors to make the scenes freely available without any control over the purpose of their use.

In addition to the paper, this demo package involves the **Swimming pool** scene (see Figures 5 and 6). Note, that even though the scene is very difficult to render for simple path tracing, bidirectional path tracing or progressive photon mapping, it is not the visibility that makes this scene difficult. Thus it could be rendered efficiently by VCM without a need for guiding. However, the guiding nicely allows rendering of this scene using just a simple path tracer.



(a) Path tracing (b) Our guided path tracing

Figure 5: Same-sample (64 per pixel) comparison of the swimming pool scene rendered by (a) path tracing and (b) our guided path tracing.



(a) Path tracing (b) Our guided path tracing

Figure 6: Equal time (1h) comparison of the swimming pool scene rendered by (a) path tracing and (b) our guided path tracing.

References

[Lehtinen *et al.*, 2013] Jaakko Lehtinen, Tero Karras, Samuli Laine, Miika Aittala, Frédo Durand, and Timo Aila. Gradient-domain Metropolis light transport. *ACM Trans. Graph.*, 32(4), 2013.

[Veach and Guibas, 1997] Eric Veach and Leonidas J. Guibas. Metropolis light transport. In *SIGGRAPH '97*, 1997.

[Vorba *et al.*, 2014] Jiří Vorba, Ondřej Karlík, Martin Šik, Tobias Ritschel, and Jaroslav Křivánek. On-line learning of parametric mixture models for light transport simulation. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2014)*, 33(4), aug 2014.