Importance Sampling of Glittering BSDFs based on Finite Mixture Distributions (Supplemental Material 2/2) Code Documentation

X. Chermain[1], B. Sauvage[1], J.-M. Dischler[1], C. Dachsbacher[2]
[1]Université de Strasbourg, CNRS, ICube UMR 7357, France
[2]Institute for Visualization and Data Analysis, Karlsruhe Institute of Technology, Germany

1 Outline

The document is structured as follows. Section 2 describes the file organisation. Section 3 provides building instructions. Section 4 provides details on using the conductor and dielectric materials in a `pbrt-v3` scene. Finally, Section 5 provides information on the glint tools.

2 File organisation

We integrate our glittering materials in the `pbrt-v3` renderer:

- The implementation of the glittering *conductor* material is located in the files `src/materials/glitteringconductor.*`.
- The implementation of the glittering *dielectric* material is located in the files `src/materials/glitteringdielectric.*`.
- The implementation of the glint tools (convergence comparisons and chi square tests) is located in the file `src/tools/glinttools.cpp`.

3 Building instructions

The following commands build the project

```
cd pathtopbrt/
mkdir build-release
cd build-release
cmake ../
make -j<number of threads>
```

Add `pathtopbrt/build-release` to your `PATH` environment variable to use the `pbrt` and `glinttool` commands anywhere. See the `readme.md` of `pbrt-v3` for more information concerning the building.

4 Render scenes and material usage

Scenes using our glittering materials are available in the `pbrt-v3-scenes` folder. Here an example usage of one of them:

```
pbrt fig1_left.pbrt
```

where the command line is launched from the `pbrt-v3-scenes` folder and when the command `pbrt` is included in the `PATH` environment variable.

The following python script

```
pbrt-v3-scene/launchallscenes.py
```

launches all the renderings of the paper. Python ≥ 3.5 with `subprocess` dependency is required to use the python script.

We use a modified version of the `pbrt-v3` path tracing algorithm for rendering. In the original path tracer, the algorithm uniformly samples *one* light for each ray – scene intersection. In our version, the algorithm uniformly samples *all* lights for each intersection. See the source files `src/integrators/path.*` for more details.

In the following, we use the same parameter presentation as `pbrt-v3`.

The `glitteringconductor` material models reflection from glittering conductors. Its parameters are:

Type	Name	Default Value	Description
spectrum texture	eta	(copper)	Index of refraction to use in computing the material's reflectance.
spectrum texture	k	(copper)	Absorption coefficient to use in computing the material's reflectance.
float texture	alphax	0.5	Beckmann roughness in the x direction.
float texture	alphay	0.5	Beckmann roughness in the y direction.
float texture	rho	0.	Slope correlation factor.
float texture	logmicrofacetdensity	20.	The logarithm of the microfacet density, without the microfacet relative area parameter applied. Set to a high value (e.g. 40) to have a glossy material (without glints).
float texture	microfacetrelativearea	1.	Percentage of the surface without microfacets. Note: *Effective* microfacet density = exp(logmicrofacetdensity) * microfacetrelativearea
float texture	alphaxbasematerial	0.01	Roughness in the x direction for the base material[1].
float texture	alphaybasematerial	0.01	Roughness in the y direction for the base material[1].
float texture	rhobasematerial	0.	Slope correlation factor for the base material[1].
float	densityrandomisation	2.	Randomly changes the density of microfacets per cell. More precisely, this parameter is the standard deviation of a normal distribution sampled to randomise the microfacet density.
bool	fresnelnoop	false	If true, the Fresnel term is always 1 (useful for white furnace test).
bool	samplevisiblearea	true	If true, samples the visible area of the normal distribution function.
bool	sampleapproximation	false	If true, samples the Gaussian approximation of the normal distribution function.
spectrum texture	dictionary	n/a	Dictionary of multi-scale, piecewise linear 1D distributions.
integer	nlevels	n/a	Number of levels of detail of the dictionary.
integer	N	n/a	Number of multi-scale 1D distributions in the dictionary.
float	alpha_dict	n/a	Roughness used to generate the dictionary.

Table 1: Parameters of the `glitteringconductor` material.

The `glitteringdielectric` material models reflection and transmission from glittering dielectrics. This material has the same parameters as the `glitteringconductor` material, without `eta`, `k` and `fresnelnoop`, and with the following specific parameters.

[1]These parameters are only used when the microfacetrelativearea < 1, i.e., when (1 - microfacetrelativearea) of the surface is covered by a base material.

Type	Name	Default Value	Description
spectrum texture	Kr	1	The reflectivity of the surface.
spectrum texture	Kt	1	The transmissivity of the surface.
float texture	index	1.5	The index of refraction of the inside of the object. (`pbrt` implicitly assumes that the exterior of objects is a vacuum, with IOR of 1.)

Table 2: Parameters of the `glitteringdielectric` material different from the `glitteringconductor` material.

5 Tool commands

We provide the following `C++` commands with the `glinttool`:

- `plotglitteringndf`
- `chisquaretestglitteringvndf`
- `chisquaretestglitteringbrdf`
- `chisquaretestglitteringbsdf`
- `convergencecomparisons`

The source code of these commands can be found in `src/tools/glinttool.cpp`. We also provide python scripts which call these `C++` commands. Python ≥ 3.5 with `matplotlib`, `numpy` and `subprocess` dependencies is required to use the python scripts (`glinttool` must also be included in the `PATH` environment variable).

These commands generate python scripts using the `matplotlib` library. Execute the generated script to visualise the result. For example:

```
python generatedfile.py
```

For Chi square tests, the commands also display the result test result (success or failure).

5.1 Plot glittering NDF

Manual of the command `plotglitteringndf`:

```
plotglitteringndf: Plots the glittering ndf of Chermain et al. 2020. Filename 1:
path to the dictionary. Filename 2: output matplotlib filename.
  options:
    --imagesize        Size of the output image. Default: 256
    --alphax           Alpha roughness of the surface in the s direction.
                       Default: 0.5
    --alphay           Alpha roughness of the surface in the t direction.
                       Default: 0.5
    --dsdx             Partial derivative of s (first component of the surface
                       position) with respect to x (first component of the
                       pixel coordinate). Default: 0.0005
    --dtdx             Partial derivative of t (second component of the surface
                       position) with respect to x (first component of the
                       pixel coordinate). Default: 0.0
    --dsdy             Partial derivative of s (first component of the surface
                       position) with respect to y (second component of the
                       pixel coordinate). Default: 0.0
    --dtdy             Partial derivative of t (second component of the surface
                       position) with respect to y (second component of the
                       pixel coordinate). Default: 0.0005
```

Example:

```
glinttool plotglitteringndf -imagesize 256 -alphax 0.6 -alphay 0.6 -dsdx
0.00052 -dtdy 0.00052 dict_N768_nLevels8.exr plotglitteringndf.py
```

where `dict_N768_nLevels8.exr` is located in the current directory. See also the python script using this command:

```
command_plotglitteringndf/command_plotglitteringndf.py
```

We use this command to plot the NDFs in the paper.

5.2 Chi square tests

The correctness of our sampling algorithms is verified with chi square tests. We validate the sampling procedure of the glittering VNDF, BRDF and BSDF with the commands `chisquaretestglitteringvndf`, `chisquaretestglitteringbrdf` and `chisquaretestglitteringbsdf`, respectively.

5.2.1 Glittering VNDF

Manual of the command `chisquaretestglitteringvndf`:

```
chisquaretestglitteringvndf: Validates our sampling of the vndf of Chermain et
al. 2020 with a chi square test. Filename 1: path to the dictionary.
Filename 2: output matplotlib filename. The plot shows differences between the
analytic PDF and the histogram built by sampling the PDF.
  options:
    --nsamplehisto     Number of samples to compute the histogram.
                       Default: 1000000
    --res              Size of integration grid. Default: 512
    --alphax           Alpha roughness of the surface in the s direction.
                       Default: 0.3
    --alphay           Alpha roughness of the surface in the t direction.
                       Default: 0.3
    --rho              Slope correlation factor. Default: 0
    --stx              X component of the pixel footprint center. Default: 0.
    --dstdxx           Partial derivative of s (first component of the surface
                       position) with respect to x (first component of the
                       pixel coordinate). Default: 0.001
    --dstdyy           Partial derivative of t (second component of the surface
                       position) with respect to y (second component of the
                       pixel coordinate). Default: 0.001
    --thetao           Polar angle of the observation direction. Default: 1.5
    --phio             Azimuthal angle of the observation direction.
                       Default: 0.
```

Example:

```
glinttool chisquaretestglitteringvndf -nsamplehisto 1000000 -res 512 -alphax 0.3
-alphay 0.3 -rho 0. -stx 0. -dstdxx 0.001 -dstdyy 0.001 -thetao 1.5 -phio 0.
dict_N768_nLevels8.exr plotchisquaretestglitteringvndf.py
```

where `dict_N768_nLevels8.exr` is located in the current directory. See also the python script using this command:

```
command_chisquaretestglitteringvndf/command_chisquaretestglitteringvndf.py
```

5.2.2 Glittering BRDF

Manual of the command `chisquaretestglitteringbrdf`:

```
chisquaretestglitteringbrdf: Validates our sampling of the glittering BRDF of
Chermain et al. 2020 with a chi square test. Filename 1: path to the dictionary.
Filename 2: output matplotlib filename. The plot shows differences between the
analytic PDF and the histogram built by sampling the PDF.
```

```
  options:
    --nsamplehisto     Number of samples to compute the histogram.
                       Default: 1000000
    --res              Size of integration grid. Default: 512
    --alphax           Alpha roughness of the surface in the s direction.
                       Default: 0.3
    --alphay           Alpha roughness of the surface in the t direction.
                       Default: 0.3
    --rho              Slope correlation factor. Default: 0
    --mra              Microfacet relative area. Default: 1.
    --stx              X component of the pixel footprint center. Default: 0.
    --dstdxx           Partial derivative of s (first component of the surface
                       position) with respect to x (first component of the
                       pixel coordinate). Default: 0.001
    --dstdyy           Partial derivative of t (second component of the surface
                       position) with respect to y (second component of the
                       pixel coordinate). Default: 0.001
    --thetao           Polar angle of the observation direction. Default: 0.2
    --phio             Azimuthal angle of the observation direction.
                       Default: 0.
```

Example:

```
glinttool chisquaretestglitteringbrdf -nsamplehisto 1000000 -res 512 -alphax 0.3
-alphay 0.3 -rho 0. -stx 0. -dstdxx 0.001 -dstdyy 0.001 -thetao 0.2 -phio 0. -mra 1.
dict_N768_nLevels8.exr plotchisquaretestglitteringbrdf.py
```

where `dict_N768_nLevels8.exr` is located in the current directory. See also the python script using this command:

```
command_chisquaretestglitteringbrdf/command_chisquaretestglitteringbrdf.py
```

5.2.3 Glittering BSDF

Manual of the command `chisquaretestglitteringbsdf`:

```
chisquaretestglitteringbsdf: Validates our sampling of the glittering BSDF with
a chi square test. Filename 1: path to the dictionary. Filename 2: output
matplotlib filename. The plot shows differences between the analytic PDF and
the histogram built by sampling the PDF.
  options:
    --nsamplehisto     Number of samples to compute the histogram.
                       Default: 256000000
    --res              Size of integration grid. Default: 4096
    --alphax           Alpha roughness of the surface in the s direction.
                       Default: 0.25
    --alphay           Alpha roughness of the surface in the t direction.
                       Default: 0.25
    --rho              Slope correlation factor. Default: 0
    --mra              Microfacet relative area. Default: 1.
    --stx              X component of the pixel footprint center. Default: 0.
    --dstdxx           Partial derivative of s (first component of the surface
                       position) with respect to x (first component of the
                       pixel coordinate). Default: 0.001
    --dstdyy           Partial derivative of t (second component of the surface
                       position) with respect to y (second component of the
                       pixel coordinate). Default: 0.001
    --thetao           Polar angle of the observation direction. Default: 0.2
    --phio             Azimuthal angle of the observation direction.
                       Default: 0.
```

Example:

```
glinttool chisquaretestglitteringbsdf -nsamplehisto 256000000 -res 4096 -alphax 0.25
-alphay 0.25 -rho 0. -stx 0. -dstdxx 0.001 -dstdyy 0.001 -thetao 0.2 -phio 0. -mra 1.
dict_N768_nLevels8.exr plotchisquaretestglitteringbsdf.py
```

where `dict_N768_nLevels8.exr` is located in the current directory. See also the python script using this command:

```
command_chisquaretestglitteringbsdf/command_chisquaretestglitteringbsdf.py
```

5.3 Convergence comparisons

Manual of the command `convergencecomparisons`:

```
convergencecomparisons: Compares convergences of two importance sampling
schemes. The first (our) uses sampling of the multi-lobe component of the
glittering BSDF, and the second (previous) uses sampling of the gaussian
approximation of the multi-lobe component. Filename 1: path to the dictionary.
Filename 2: output matplotlib filename. Output: Two matplotlib files. The first
named <filename 2>_radiance_n.py shows eight convergence curves for each
sampling strategy. The second, named <filename 2>_pointwise_boxplots.py shows
the pointwise boxplot of the <nruns> for each sampling strategy.
  options:
    --nsamples         Number of samples to compute the integral.
                       Default: 10000
    --nruns            Number of runs. Default: 8
    --pgf              If 1, matplotlib will use pgf exporter. Default: 1
    --alphax           Alpha roughness of the surface in the s direction.
                       Default: 0.3
    --alphay           Alpha roughness of the surface in the t direction.
                       Default: 0.3
    --rho              Slope correlation factor. Default: 0
    --mra              Microfacet relative area. Default: 1.
    --stx              X component of the pixel footprint center. Default: 0.
    --dstdxx           Partial derivative of s (first component of the surface
                       position) with respect to x (first component of the
                       pixel coordinate). Default: 0.001
    --dstdyy           Partial derivative of t (second component of the surface
                       position) with respect to y (second component of the
                       pixel coordinate). Default: 0.001
    --thetao           Polar angle of the observation direction. Default: 0.2
    --phio             Azimuthal angle of the observation direction.
                       Default: 0.
```

Example:

```
glinttool convergencecomparisons -stx 0. -thetao 1.5 -alphax 0.6 -alphay 0.6
-dstdxx 0.0001 -dstdyy 0.0001 -pgf 0 -nruns 8 dict_N768_nLevels8.exr plot.py
```

where `dict_N768_nLevels8.exr` is located in the current directory. See also the python scripts using this command:

```
command_convergencecomparisons/convergencecomparisons.py
command_convergencecomparisons/allconvergencecomparisons.py
```

The last python script generates the pointwise boxplots of the supplemental material 1.