Development and Integration of the RADIANCE Photon Map Extension — Technical Report —

Roland Schregle (roland.schregle@{hslu.ch, gmail.com})
CC Envelopes and Solar Energy
Lucerne University of Applied Sciences and Arts

Revision 1.14
June 1, 2015

Abstract

This technical report documents the development of the RADIANCE photon mapping extension and its integration into the official RADIANCE 5.0 distribution. A number of new techniques are now supported for a more efficient daylight simulation workflow, notably support for data driven material models via the **bsdf** modifier, and ray coefficients for climate based daylight modelling using the ***rcontrib*** tool. This document focuses primarily on the implementation details of the photon mapping software in support of this new functionality.

Contents

1 Introduction

The accurate simulation of daylit interiors is essential in assessing a building's performance in the context of glare prediction and energy saving potential through daylight autonomy. Raytracing techniques have proven to be expedient in this application as they accurately model the light transport and how it propagates in a typical office environment. In particular, they also account for light redirecting components and glazings for sun shading and glare reduction, which dramatically affect the daylight availability. The lighting levels are typically analysed for an entire year to account for the temporal dependency of the sun position, notably due to seasons, commonly referred to as *climate based daylight modelling*, or CBDM.

Although light transport along a light ray is inherently bidirectional (reversing the direction of light propagation does not invalidate the simulation), there is a distinction between backward and forward raytracers; the former emit rays at the view or measurement point(s), whereas the latter emits rays from the light sources. Forward raytracing is particularly effective at modelling highly specular light redirection to produce concentrated highlights (caustics), which typically constitute glare if they compromise an office occupant's visual comfort.

Photon mapping [5] is a forward raytracing technique which supplements a standard backward raytracer, resulting in bidirectional light transport. The technique mimics light particle transport by recording hitpoints along with their associated energy, and uses density estimation to reconstruct the resulting irradiance on the surfaces.

The forward raytracing solution detailed in this report is based on a photon mapping extension to the RADIANCE rendering system originally developed by the author [7]. It extends the RADIANCE backward raytracing core [11] with a forward raytracer as described above.

Recently introduced functionality in RADIANCE and new workflows have necessitated a revision of the photon mapping extension to support these new developments.

2 Monte Carlo Theory

The RADIANCE photon mapping extension implements a Monte Carlo path tracing algorithm which connects a series of scattering events as individual photon rays interact with the materials which characterise the scene geometry. The combination of these rays results in a photon path which represents a *random walk* (see figure 1).

Photon mapping belongs to the class of forward path tracers, i.e. which emit rays from the light sources instead of the viewpoint, as RADIANCE CLASSIC™ does. Photons are then stored in a space subdividing data structure (traditionally a k-d tree [3], although more efficient alternatives have recently been proposed) at every secondary ray intersection. The direct intersection immediately after emission from the light source is handled by more efficient techniques which specifically sample the light source, as implemented in the RADIANCE library.

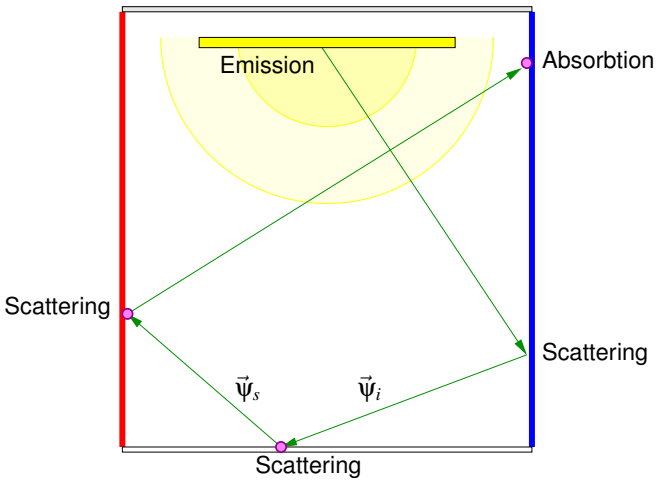

Figure 1: A random walk representing a photon path. A photon is deposited at every secondary ray intersection (magenta points) following emission from the light source, then scattered according to the incident and scattering directions $\vec{\psi}_i$ and $\vec{\psi}_s$, until it is absorbed and the path is terminated.

2.1 Random Walk

The implementation of this random walk relies on a material scattering routine to probabilistically generate a scattering direction $\vec{\psi}_s$ for a photon incident from direction $\vec{\psi}_i$ according to the material's BSDF $f_s(\vec{\psi}_i, \vec{\psi}_s)$. This commonly employed *importance sampling* strategy reduces variance but is inherently biased due to the nonuniform sampling [4, 6]. Consequently, the sample is inversely weighted by the normalised probability of choosing the direction $\vec{\psi}_s$ as defined by the *probability density function*, or PDF, p_s:

$$\Phi(\vec{\psi}_s) = \Phi(\vec{\psi}_i)\frac{f_s(\vec{\psi}_i, \vec{\psi}_s)}{p_s(\vec{\psi}_s)}, \tag{1}$$

where $\Phi(\vec{\psi}_i)$ and $\Phi(\vec{\psi}_s)$ is the incident resp. scattered photon flux.

It can be shown that the expected value an estimator $\hat{\Phi}_s$ for the scattered photon flux defined as above is unbiased by integrating over all scattering directions:

$$\begin{aligned} E\left[\hat{\Phi}_s\right] &= \int_{\vec{\psi}_s \in \Omega} \Phi(\vec{\psi}_i)\frac{f_s(\vec{\psi}_i, \vec{\psi}_s)}{p_s(\vec{\psi}_s)} p_s(\vec{\psi}_s)\, d\omega & (2)\\ &= \int_{\vec{\psi}_s \in \Omega} \Phi(\vec{\psi}_i)\, f_s(\vec{\psi}_i, \vec{\psi}_s)\, d\omega & (3)\\ &= \Phi_s, & (4) \end{aligned}$$

where Ω is the set of all scattering directions. This gives the total scattered flux Φ_s that is reradiated from the material.

2.2 Russian Roulette

Since photon paths cannot be of infinite length in a practical context, the random walk is usually terminated probabilistically by a statistical process aptly named *Russian Roulette*. Photons are then scattered with a normalised acceptance probability p_a dependent on the incident direction, otherwise they are absorbed and the path is terminated [2]. Because this constitutes non-uniform sampling, it again introduces bias for the same reasons as the non-uniform scattering, and requires similar weighting of

the samples by $\frac{1}{p_a}$ for compensation. Equation 1 then becomes:

$$\Phi(\vec{\psi}_s) = \begin{cases} \Phi(\vec{\psi}_i)\frac{f_s(\vec{\psi}_i,\vec{\psi}_s)}{p_a(\vec{\psi}_i)p_s(\vec{\psi}_s)} & : \xi < p_a(\vec{\psi}_i) \\ 0, \text{ photon absorbed} & : \xi \geq p_a(\vec{\psi}_i), \end{cases} \tag{5}$$

where $\xi \in [0,1]$ is a random number. In an exercise similar to equation 2, it can be shown that this estimator is also unbiased:

$$E\left[\hat{\Phi}_s\right] = \int_{\vec{\psi}_s \in \Omega} \Phi(\vec{\psi}_i) \left(\overbrace{\frac{f_s(\vec{\psi}_i,\vec{\psi}_s)}{p_a(\vec{\psi}_i)\,p_s(\vec{\psi}_s)} p_a(\vec{\psi}_i)}^{\text{scattering}} + \underbrace{0\,(1-p_a(\vec{\psi}_i))}_{\text{absorption}} \right) p_s(\vec{\psi}_s)\,d\omega \tag{6}$$

$$= \int_{\vec{\psi}_s \in \Omega} \Phi(\vec{\psi}_i)\, f_s(\vec{\psi}_i,\vec{\psi}_s)\,d\omega \tag{7}$$

$$= \Phi_s. \tag{8}$$

So there.

2.3 Analogue Simulation

Equation 5 introduces variance into the scattered photon flux Φ_s due to the non-uniform sample weighting. The RADIANCE photon map extension adopts the common practice of maintaining a uniform photon flux through *analogue simulation* [8]. With this approach, only the photon density is modulated, thus greatly reducing the variance. As a bonus, it also saves on computation.

In an analogue simulation, all photons emitted from the light sources carry the same flux

$$\Phi_p = \frac{\sum_i \Phi_{e,i}}{N_e}, \tag{9}$$

where $\Phi_{e,i}$ is the integrated flux emitted from the i^{th} light source, and N_e is the total number of emitted photons.

Uniform flux is maintained during photon scattering by relating the PDF p_s to the BSDF f_s, thus concentrating the reradiated photons in those directions of high scattering probability. Similarly, the acceptance probability for Russian Roulette, p_a can be related to the material's scattering albedo α, which is the integral of the BSDF f_s over all scattering directions:

$$p_a(\vec{\psi}_i) = \alpha(\vec{\psi}_i) = \int_{\vec{\psi}_s \in \Omega} f_s(\vec{\psi}_i,\vec{\psi}_s)\,d\omega \tag{10}$$

$$p_s(\vec{\psi}_i,\vec{\psi}_s) = \frac{f_s(\vec{\psi}_i,\vec{\psi}_s)}{\alpha(\vec{\psi}_i)}. \tag{11}$$

Equation 5 now simplifies to:

$$\Phi(\vec{\psi}_s) = \Phi(\vec{\psi}_i)\frac{f_s(\vec{\psi}_i,\vec{\psi}_s)}{\alpha(\vec{\psi}_i)\frac{f_s(\vec{\psi}_i,\vec{\psi}_s)}{\alpha(\vec{\psi}_i)}} \tag{12}$$

$$= \Phi(\vec{\psi}_i). \tag{13}$$

.

To account for spectral effects, equation 12 is modified to attenuate the incident photon flux by the spectral BSDF components (limited to the red, green, and blue colour channels in the current implementation), then normalised:

$$\Phi(\vec{\psi}_s) = \left|\left|\Phi(\vec{\psi}_i)\, f_{s,\lambda}(\vec{\psi}_i,\vec{\psi}_s)\right|\right|_\lambda \tag{14}$$

$$||f||_\lambda = \frac{f}{\int f(\lambda)\,d\lambda}, \tag{15}$$

where $\|f\|_\lambda$ is a spectral normalisation operator which averages f to unity over all wavelength bands. The photon flux is then scaled to Φ_p in a postprocess after the forward pass, once the total number of emitted photons is known.

3 BSDF Support

Data-driven materials have become increasingly important for physically accurate simulations, particularly when modelling daylight redirecting components. CC EASE recently installed a pgII goniophotometer [1] and is adopting a workflow which integrates measured scattering distributions from material samples into its RADIANCE simulations using the **bsdf** material modifier.

This necessitated implementing support for **bsdf** in the existing photon mapping code in order to accurately simulate BSDFs with localised peaks, which predominate with specular materials.

3.1 Implementation

The RADIANCE photon map extension uses the standard Monte Carlo path tracing techniques from section 2 to trace photons as they are scattered by materials. Each supported material type has a corresponding scattering routine in the **pmapmat** source module. For **bsdf**, this is provided by the **bsdfPhotonScatter()** routine. This code is largely based on the **m_bsdf()** shading and **SDsampBSDF()** sampling routines in RADIANCE CLASSIC™.

While **SDsampBSDF()** already provides Monte Carlo sampling functionality, it was customised as part of **bsdfPhotonScatter()** since it does not return the type of scattering (diffuse, non-diffuse), nor the probability for the samples it generates. This is crucial for unbiased results as discussed in section-ref:monteCarlo.

The **bsdf** material is characterised by a non-diffuse component f_s consisting of actual BSDF data, and a constant diffuse term f_d extracted from this data as its minimum value; this is essentially interpreted as a DC offset for the non-diffuse component, and treated separately via RADIANCE's ambient calculation.

bsdfPhotonScatter() uses the sum of the diffuse and non-diffuse reflectance and transmittance $\rho_d + \rho_s + \tau_d + \tau_s$ as acceptance probability for Russian Roulette. If the photon survives, a scattering event (diffuse/non-diffuse reflection/transmission) is probabilistically chosen depending on the relative magnitudes of ρ_d, τ_d, ρ_s, and τ_s.

3.1.1 Sample Generation

Sampling directions for diffuse scattering are trivially generated with a cosine distribution by the **diffPhotonScatter()** routine in the **pmapmat** module, and the photon flux is modified according to a user-defined colour triplet before being renormalised. The non-diffuse case on the other hand requires inverting a cumulative distribution function, or CDF, P_s which is the integral of the PDF p_s over a linearly ordered domain of incident directions.

The RADIANCE BSDF library routines use a Hilbert space filling curve to impose a linear ordering on the set of directions $\vec{\psi}_s$ for the BSDF [10]. Hilbert curves possess the unique property of covering a partitioned, multidimensional space with a linear, meandering pattern. Each segment on the curve defines a node in the multidimensional space, which is uniquely identified by its *Hilbert index* H and represents the linear distance of the curve up to this node.

The CDF P_s with Hilbert ordering is then defined as follows:

$$P_s(\vec{\psi}_s) = \int_{\vec{\psi}:H(\vec{\psi})<H(\vec{\psi}_s)} p_s(\vec{\psi})\, d\omega. \tag{16}$$

Note that this function, like the PDF p_s, is normalised.

A sampling direction is then generated by inverting this function, i.e. by finding the Hilbert index of the node containing $P_s^{-1}(\xi)$ for a random variable $\xi \in [0,1]$:

$$P_s^{-1}(\xi) = H^{-1}\left(\inf\{H(\vec{\psi}_s) : P_s(\vec{\psi}_s) \geq \xi\}\right). \tag{17}$$

The **sampCDist()** routine in the RADIANCE library performs this inversion efficiently by caching the CDF for the current incident direction (on which it depends), minimising the amount of computation spent in redundantly (re)evaluating the underlying PDF. The sampled direction $\vec{\psi}_s$ is then obtained via an inverse lookup on its Hilbert index, which is also highly efficient.

3.1.2 Proxy Geometry Support

In RADIANCE CLASSIC™, the indirect components (both ambient and specular) are scattered by the **bsdf** material. When applied to primary (view) rays, the material often appears objectionably opaque and noisy (depending on the data resolution), particularly when transmitting an exterior view through a fenestration.

To this end, a *proxy geometry* can be presented to the view rays en lieu of the BSDF, assuming the BSDF correlates with the behaviour of the geometry, as is the case if the was former generated from the latter with ***genbsdf***. The view rays will then ignore the BSDF and be scattered by the geometry, thus accurately rendering its appearance. Indirect rays still benefit from the BSDF as no additional rays are spawned within the (potentially complex) proxy geometry, thus accelerating the ambient and indirect specular components.

Like RADIANCE CLASSIC™, the photon map ignores the proxy geometry and uses the BSDF for scattering in forward path tracing, translating the rays by the thickness specified for the proxy geometry. Note that it is a user error to include proxy geometry with a zero **bsdf** thickness, as photons will be doubly scattered by the geometry *and* the BSDF.

3.2 Results

A series of sample renderings using the BSDF of retroreflecting blinds in conjunction with the photon map is shown in figure 2. The BSDF was generated from a geometric model of the lamellae by ***genBSDF*** and represented as a rank 4 tensor tree with a resolution of 32^4 directions.

Each lamella is characterised by a specular, longitudinal prismatic profile on the upper (concave) side to effect retroreflection of sunlight incident at high angles. The underside of the lamella is diffuse and bounces skylight reflected from the lamella below it towards the interior.

In the left rendering in figure 2, the blinds are represented as a polygon modified by the BSDF, which was used to scatter photons in the forward pass. In the centre rendering, the corresponding geometry was placed just behind the polygon as a proxy for the view rays, which does not affect the photon scattering. In the right rendering, the BSDF is absent, and photons are scattered by the geometry, yielding the characteristic caustics on the wall from retroreflection. While the BSDF cannot resolve these details, the overall behaviour is consistent with that of the geometry.

4 Ray Coefficient / Light Source Contribution Support

The ability to quantify the contribution of each light source to the irradiance at some position in an interior space is essential to climate based daylight modelling, or CBDM. In this context, the (usually annual) daylight availability is assessed as a function of the seasonal sun position taken at regular intervals, e.g. daily or hourly.

Rather than conducting a series of individual simulations for each light source configuration, it is far more efficient to combine them into one simulation with multiple light sources, and then tabulate the

Figure 2: Retroreflecting blinds rendered with 4m caustic and 400k global photons using a tensor tree BSDF without (left) and with (centre) proxy geometry, and with geometry only (right). The limited resolution of the BSDF precludes the formation of the retroreflection caustics, but the system's overall behaviour is consistent in all cases.

resulting irradiance contributions for each light source (or ray coefficients, in which case the source's emitted radiance is normalised). RADIANCE CLASSIC™ offers the ***rcontrib*** tool for this purpose, which establishes an elegant and efficient workflow for CBDM.

Support for this application, specifically in conjunction with ***rcontrib***, has been lacking in the photon map extension up to now. As of RADIANCE release 5.0, the photon map has been adapted for this task to accurately account for contributions from caustics in particular.

Rcontrib tabulates contributions from arbitrary objects whose corresponding modifiers are specified by the user. By contrast, the photon map is currently restricted to contributions from light sources only, which is however the most common application for CBDM.

4.1 Photon Contribution Density Estimate

The contributions of each light source to the irradiance at a point $\vec{x}$ are evaluated by a modified photon density estimate, referred to here as a photon contribution density estimate (see figure 3). To obtain the contributions, we locate the n_p nearest photons around $\vec{x}$ in the k-d tree. We then collect the flux ϕ_p of each photon into a "bin" corresponding to its emitting light source. This accumulated flux is then divided by the area intercepted by the search radius r, on which the photons are assumed to lie:

$$E_l(\vec{x}) \approx \sum_i K(\|\vec{x},\vec{x}_i\|)\frac{\phi_p(\vec{x}_i)}{\pi r^2} \qquad \forall i : B(\phi_p(\vec{x}_i)) = l, \tag{18}$$

where

- n_p is the density estimate bandwidth (number of nearest photons to search)
- r is the radius containing the n_p phound photons
- K is a normalised weighting function based on the photon's distance to $\vec{x}$
- ϕ_p is the flux of a photon emitted from light source l as mapped by a binning function B
- E_l is the the resultant irradiance contributed by light source l.

4.2 Implementation

Parts of the photon mapping code have been modified to provide optimised support for ray coefficients, and collected in the **pmapcontrib** module. Additional modifications in the base photon mapping code

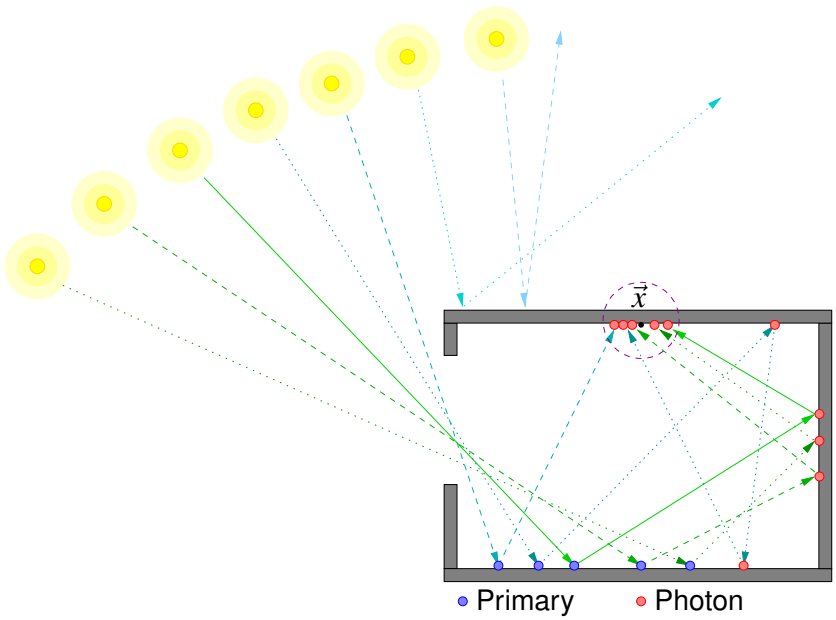

Figure 3: Climate based daylight modelling using photon contributions from multiple sun positions. The contribution from each source to the irradiance at point $\vec{x}$ is evaluated by a modified density estimate for a number of nearby photons around $\vec{x}$. Each photon (red) along a path references its primary (blue), which in turn contains a reference to its emitting light source.

have been made, notably in the **pmapdata** module, such as additional fields in the photon map data structures, and scaling of photon flux per light source while building the photon map. A dedicated contribution photon map type has been introduced to encapsulate the new functionality. This implementation is currently still considered experimental.

4.2.1 Modified Photon Distribution

Ideally, each light source should contribute to the irradiance at any point in the scene (unless obstructed), regardless of its emitted power. While this can be achieved through some local photon density control during the forward ray tracing step [9], it requires that a searchable photon map of some form already exists at this point. As this is not the case with our implementation, we ensure each light source at least contributes an equal number of *stored* photons to the scene.

While this does not guarantee that a contribution density estimate will include photons from all light sources, it significantly raises its likelihood, particularly for weak light sources whose contributions would otherwise be underestimated and noisy, or even zero in extreme cases.

This deviates from the default photon distribution algorithm, in which the number of photons *emitted* per light source is proportional to its emitted power, resulting in a uniform photon flux as in equation 12. To compensate, the photon flux must be modulated per light source, which does increase the variance as tradeoff.

To this end, the default **distribPhotons()** routine from the **pmap** module has been customised into the **distribPhotonContrib()** routine as part of the **pmapcontrib** module.

4.2.2 Modified Photon Lookup and Density Estimate

The standard nearest neighbour lookup in the k-d tree also required adaptation to isolate contributions. The default **nearestNeighbours()** search in **pmapdata** has been adapted to accept only those photons

emitted from light sources whose contributions are sought (i.e. whose modifiers are in the modifier list passed to ***rcontrib***). Furthermore, specularly reflected (caustic) photons within the contribution photon map can be selected to obtain their contributions at primary rays. This on-the-fly filtering thus efficiently obtains contributions from only those photons relevant to the simulation.

The standard **photonDensity()** routine from the **pmap** module that computes the density estimate for the phound photons has also been adapted. The resulting customised **photonContrib()** in the **pmapcontrib** module implements the photon contribution density estimate as summarised in equation 18.

4.2.3 Light Source Contributions

The definition of the original **PhotonMap** data structure in the **pmapdata** module was modified to accommodate light source contribution as follows:

```
typedef struct PhotonMap {
    Photon *heap;                /* Photon k-d tree as linear array */
    ...
    PhotonPrimary *primary;      /* Primary photon rays */
    unsigned long primarySize,
                  primaryEnd;
    LUTAB *srcContrib;           /* lookup table for source contribs */
} PhotonMap;
```

Listing 1: **PhotonMap** data structure modified for light source contributions

This refers to a lookup table **srcContrib** which contains the user specified light source modifiers whose contributions are sought, and their associated "bins" in which the contributions are accumulated for evaluation by ***rcontrib***. This lookup table is implemented in the base RADIANCE library and is also used by ***rcontrib*** in RADIANCE CLASSIC™ mode.

The **PhotonMap** data structure points via **heap** to an array of photons arranged as a linear k-d tree, which comprises the actual photon map. Each photon node is defined as follows:

```
typedef struct {
    float pos [3];               /* Photon position */
    signed char norm [3];        /* Surface normal at pos */
    char flags;                  /* Bit 0-1: kd-tree discriminator axis,
                                    Bit 2:   caustic photon */
#ifdef PMAP_FLOAT_FLUX
    COLOR flux;
#else
    COLR flux;                   /* Photon flux */
#endif
    uint32 primary;              /* Index to primary rays */
} Photon;
```

Listing 2: **Photon** data structure modified for light source contributions

This characterises each photon by its position (hitpoint during forward raytracing), surface normal (for exclusion of backface visibility), flags (k-d tree splitting dimension, caustic tag), and flux. In support of ray coefficients, it also indexes its primary ray to identify the light source whence it was emitted.

The photon primary rays indexed by each photon are contained in an array pointed to by **primary** in **PhotonMap**. This data structure is specific to ray coefficients and is defined as follows:

```
/* Primary photon ray for light source contributions */
typedef struct {
   int32 srcIdx;              /* Index of emitting light source */
   int32 dir[3];              /* Encoded ray direction */
   float pos[3];              /* Hit point */
} PhotonPrimary;
```

Listing 3: **PhotonPrimary** data structure

There is one **PhotonPrimary** node per primary hitpoint during forward raytracing, but only nodes that spawn secondary hits, and therefore photons, are actually stored. Note that a primary may be multiply referenced, namely by each photon it spawns along its path.

Photon primaries are not actual photons, and therefore not stored in the k-d tree, nor are they relevant to lookups therein. They are simply stored in a linear, unsorted list and participate in the binning process which assigns the photon contributions.

Each photon primary refers to the light source that emitted the primary ray (via an index into the **source** array defined in the RADIANCE library). In addition, the ray's incident direction (encoded as 32-bit integer) and hitpoint is stored for finer grained binning using runtime functions as described in the next section.

4.2.4 Binning with Runtime Functions

In addition to binning based on each photon's emitting light source, finer grained binning can be achieved as in ***rcontrib*** via runtime binning functions. A photon path's primary incident direction and hitpoint (corresponding to a shadow ray aimed at the light source in a backward raytracing context) is then passed to the function via the variables D_x, D_y, D_z and P_x, P_y, P_z, respectively. These are obtained directly from the **PhotonPrimary** node indexed by each photon during the contribution density estimate. The corresponding bin number is then obtained by evaluating the function. A typical example is the directional discretisation of a hemispherical sky dome into 145 patches using the Tregenza mapping.

4.2.5 Ray Coefficients

Ray coefficients can be obtained from ***rcontrib*** using the **-V-** option. In this mode, the emitted radiance from every light source is normalised, and can be individually scaled *a posteriori* without having to rebuild the photon map. This is implemented by dividing a photon's flux by the emitted radiance of its corresponding light source (after applying any modifiers which affect its distribution pattern). As in binning with runtime functions, the necessary variables (direction, position) to evaluate the source's modified emission are also obtained from the photon's primary ray.

4.3 Results

Figure 4 is an example of an annual climate based simulation of the irradiance for an interior ceiling sensor in reference room. This required binning the contributions of 365 directional light sources representing the daily sun positions at 1:00pm over an entire year. A total of 10M photons were used, with a bandwidth of 365 per contribution density estimate – the minimum for this example without omitting contributions from at least one light source.

Figure 5 is an example of light source binning for the same room and sensor using the photon primaries' incident directions to identify the contributions from 145 Tregenza patches of an intermediate sky without sun. The window faces south, thus resulting in peaks for bins 16, 46, 73, 97, 118, 133, and 142. A total of 1M photons were used with a bandwidth of 50 per contribution density estimate.

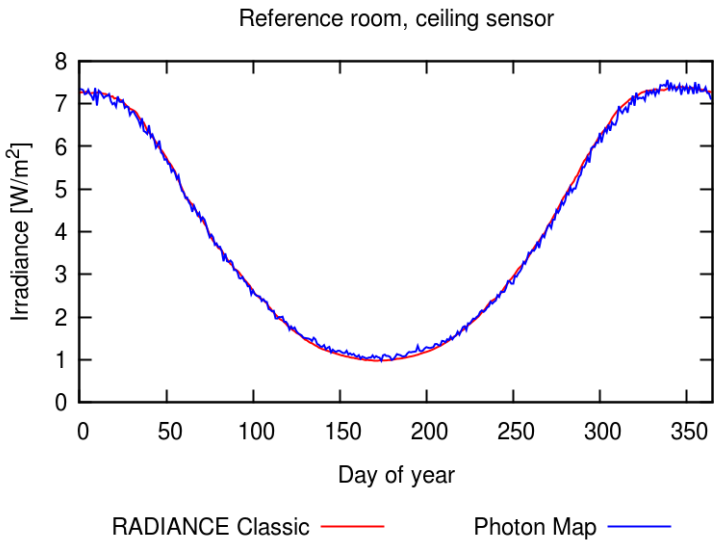

Figure 4: Comparison of annual interior ceiling irradiance over 365 days at 1:00pm simulated with RADIANCE CLASSIC™ (red) and photon map (blue).

In both cases, the irradiance contributions obtained from the photon map agree well with those obtained with RADIANCE CLASSIC™, although the contribution density estimates do introduce additional noise. The noise level is tolerable, however, and more than compensated for by the increased accuracy when analysing contributions due to glare and light redirection from specular surfaces.

A visual analysis for a half year in weekly intervals is shown for a typical 6×6m office space in figure 6. The performance of the redirecting system (retroreflecting blinds in this example) can then be assessed while taking seasonal daylight availability into account. Redirection towards the ceiling is evident for every instance, particularly from January to April. Transmission, on the other hand, occurs at low solar angles until March, before transitioning into retroreflection at high solar angles. The validity of this simulated behaviour has been verified by the manufacturer of the blinds.

5 Maximum Photon Search Radius

The photon lookup routine **nearestNeighbours()** maintains a running distance to the furthest of the n_p photons (as defined by the bandwidth) found thus far. This distance defines the radius covered by the density estimate (see equation 18) and is initialised with a value r_{max} when the routine is called, and progressively reduced during the search until the n_p nearest photons have been identified within this radius. Photons beyond r_{max} are disregarded.

An appropriately sized r_{max} will eliminate many distant photons from the search by pruning branches in the k-d tree, thus greatly accelerating the process. Setting it too small, however, will result in a "short" lookup, with fewer than n_p photons found. While this does not invalidate the resulting density estimate, a very low number of photons will render it inaccurate due to excessive noise.

Previous versions of the photon map extension derived r_{max} from a minimum irradiance threshold E_t based on the average photon flux, under the assumption that density estimates covering a larger radius no longer yield a discernible irradiance [8]. This parameter could be modified by the user for exceptionally dark scenes. Because r_{max} was static, it did not adapt to non-uniform distributions where photons tended to cluster, notably in caustics. An optimal r_{max} should adapt to the local photon density, but this is dynamic and difficult to predict.

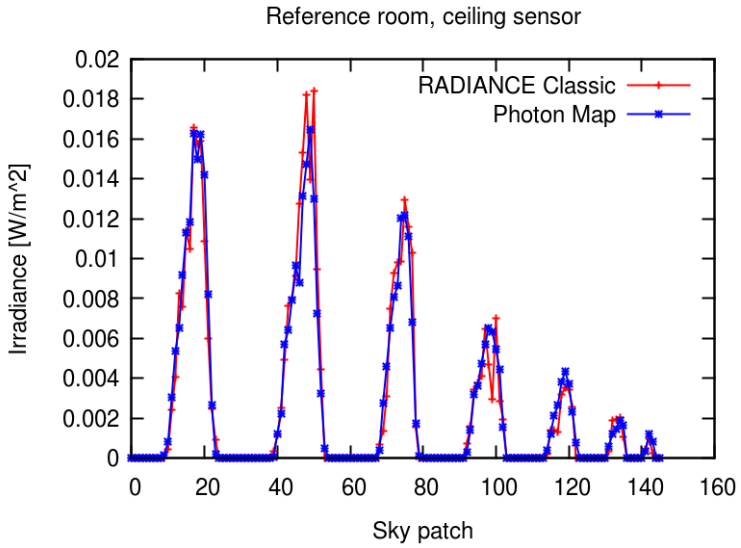

Figure 5: Comparison of contributions from 145 Tregenza sky patches to the indirect irradiance on an interior ceiling sensor via a window facing south.

As of RADIANCE release 5.0, the photon map extension estimates an initial value $r_{max,0}$ for r_{max} from the average photon distance r_c to the centre of gravity $\vec{x}_c$ of the photon distribution. Similarly to the previous version, this is derived from a minimum threshold irradiance E_t as a fraction k_t of the average irradiance E_c around the photons' centre of gravity:

$$E_t(\vec{x}) = k_t E_c, \qquad k_t < 1 \tag{19}$$

$$\sum_i^{n_p} \frac{\phi_p(\vec{x}_i)}{\pi r_{max,0}^2} = k_t \sum_i^{N_p} \frac{\phi_p(\vec{x}_i)}{\pi r_c^2} \tag{20}$$

$$r_{max,0} = \sqrt{\frac{n_p r_c}{k_t N_p}}, \tag{21}$$

where the centre of gravity $\vec{x}_c$ and the average photon distance r_c to it are defined as

$$\vec{x}_c = \sum_i^{N_p} \frac{\vec{x}_i}{N_p}, \qquad r_c = \sum_i^{N_p} \frac{\|\vec{x}_i - \vec{x}_c\|}{N_p}, \tag{22}$$

and n_p and N_p is the density estimate bandwidth and total number of photons, respectively.

It is assumed that density estimates over a larger area than $\pi r_{max,0}^2$ yield negligible irradiance. In the particular case where no photons are found within a radius $r_{max,0}$, this also implies the assumption that the surface is entirely devoid of photons, thereby obviating an exhaustive search over a larger radius.

In contrast to previous releases, the current photon map extension dynamically adapts r_{max} after initialisation with $r_{max,0}$. As a large search radius impacts performance, r_{max} is gradually downscaled by a factor $d_r < 1$ every n_r successful (i.e. non-short) photon lookups. On the other hand, when a short photon lookup does occur, r_{max} is upscaled by a factor $u_r > 1$. If the lookup returns no photons at all within r_{max}, the abovementioned reasoning applies and the surface is considered devoid of photons, implying that enlarging r_{max} will have no effect. By definition we clamp r_{max} to $r_{max,0}$ when upscaling so as to never exceed its initial value.

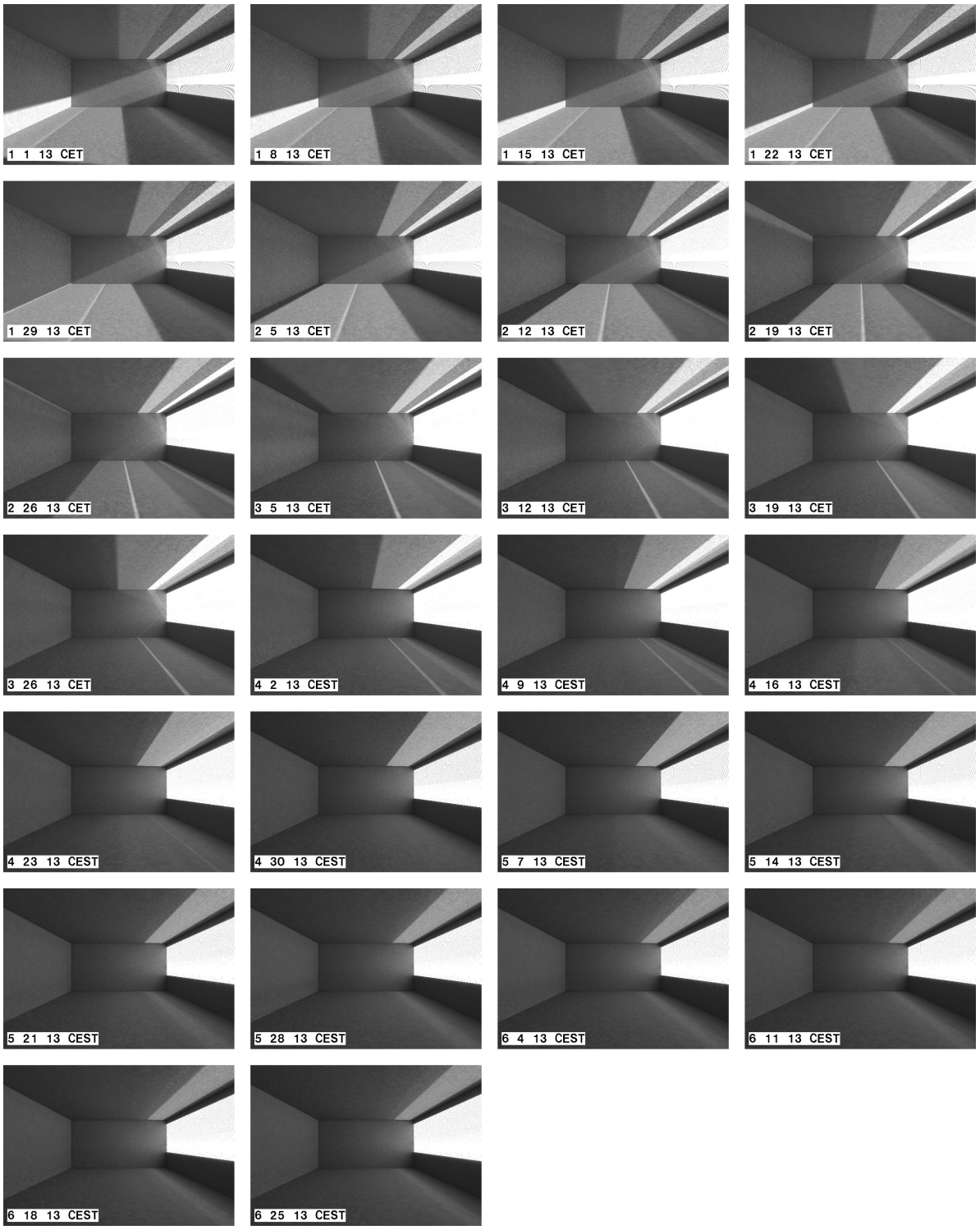

Figure 6: Retroreflecting blinds at 26 solar angles for a half year (January to June with weekly intervals) rendered with contribution photon map and ***rcontrib***. In each instance the time of day is 1:00pm. Redirection and transmission dominate until March, while retroreflection dominates in the following months.

Note that u_r and d_r are chosen to be asymmetric, with values of 4 and 0.9, respectively, in the current implementation. This results in cyclic behaviour, alternating between a large increase in r_{max} and a number of smaller reductions until a new short lookup occurs, which repeats the cycle.

A judicious choice of the maximum search radius reduction interval n_r balances the frequency of short lookups against the penalty incurred by an oversized r_{max}. We find that the current value of n_r = 1000 photon lookups achieves this balance satisfactorily for a variety of test scenes, notably problematic ones containing surfaces devoid of photons.

Verbose warnings about short lookups (indicating the position and object on which it took place) can be output by defining **PMAP_LOOKUP_WARN** during compilation. This is intended for debugging and disabled by default due to the ensuing Console Clutter^{TM}.

A short lookup may be optionally restarted with the enlarged radius by defining **PMAP_LOOKUP_REDO**. In our tests the improvements have been negligible, hence this is disabled by default.

These simple modifications adapt the maximum search radius to the average photon density and in particular account for non-uniform concentrations of photons in caustics.

There are situations where this automation may overestimate $r_{max,0}$ with grossly nonuniform photon distributions, leading to excessive bias in sparsely populated regions. The user may then override and specify $r_{max,0}$ via the **-am** option to ***rpict***, in which case it remains fixed and is no longer adapted. This, however, assumes some knowledge of the average photon density on behalf of the user.

6 Region of Interest (ROI)

The photon map extension supports the specification of a region of interest, or ROI, in which to exclusively store photons. This is done by supplying appropriate 3D coordinates defining a bounding box as $(x_{min}, x_{max}, y_{min}, y_{max}, z_{min}, z_{max})$ to ***mkpmap*** via the **-api** option.

The **addPhoton()** routine then checks whether a photon to store lies within the defined ROI, in which case it will be added to the photon map. The photon path and the resulting contribution to the lighting solution remains unaffected; path segments entering the ROI will deposit photons, while those leaving the ROI will not. Note that the number of stored photons remains approximately the same.

The primary purpose of defining a ROI is to optimise the photon distribution by concentrating photons where they contribute significantly, and disregarding parts of the geometry which are out of view or insignificant. A typical example is an interior space with a fenestration, with many photons being deposited within the latter, thus reducing the number of useful photons in the interior. The ROI would then correspond to the extent of the interior and exlude the fenestration. An example of this is shown in figure 7.

Using a ROI has inherent pitfalls, as excluding photons from the scene can lead to bias. It is assumed that the user can estimate the bias that results from omitting the indirect illumination contributed by the excluded photons outside the ROI. Using a ROI corresponding to the abovementioned interior to estimate glare from the fenestration would clearly lead to underestimation as interreflections within the fenestration are omitted.

Since indiscriminate use of a ROI can lead to erroneous results, we only consider the option appropriate for advanced users. Consequently, it is not accessible by default and must be enabled at compile time by defining **PMAP_ROI**.

7 Integration with RADIANCE Release 5.0

Prior to version RADIANCE release 5.0, the photon map extension was only available as a separate patch which had to be applied to the official RADIANCE distribution. This was not only cumbersome and

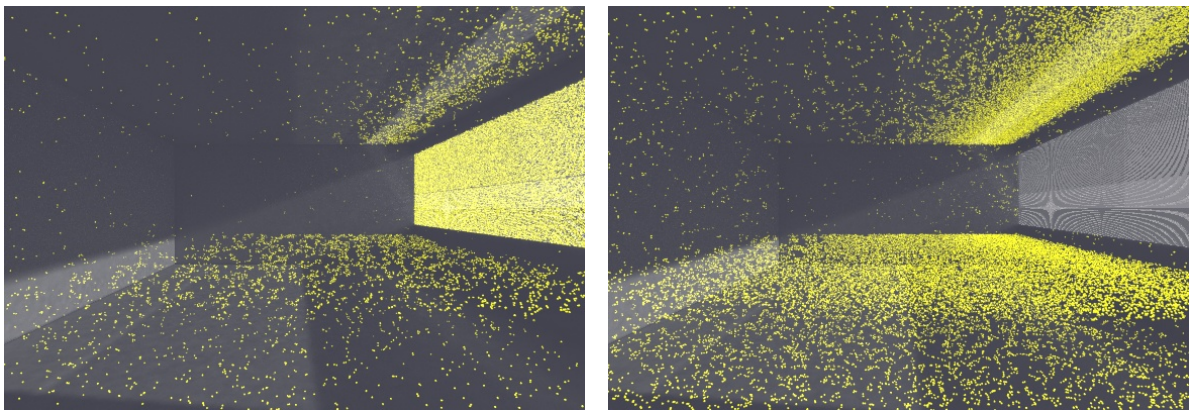

Figure 7: Photon distributions (visualised with ***pmapdump***) within an interior fitted with a redirecting system. Without constraint, the majority of photons are stored within the fenestration due to inter-reflection (left). By defining a region of interest corresponding to the room's extent, photons are only deposited in the interior, resulting in an improved distribution and rendering quality (right). Note the use of an invisible **antimatter** sensor surface to deposit photons at work plane height (0.75m). This is noticeably underpopulated without ROI, thus impacting illuminance measurements on it.

error prone, but a major hurdle for users, most of whom are not familiar with patching and compiling binaries.

The photon map extension is now integrated into the official RADIANCE 5.0 distribution. A number of overall changes have been made to the code in order to isolate photon map specific functionality to facilitate maintenance. These are summarised below:

- All photon map sources where moved from their separate source tree directory into **ray**/**src**/**rt**. **libpmap** has been deprecated, and the photon map binaries are now included in **libradiance**.
- All **#ifdef PHOTON_MAP** instances, formerly used to selectively enable the photon mapping code during compilation, have been removed; a /* **PMAP** */ comment now facilitates location of photon map specific modifications.
- All significant photon map specific code has been isolated by encapsulating it in interface routines in separate files.
- The photon map file format is now conformant with that of RADIANCE; Photon maps from earlier versions are no longer compatible.

In addition, a number of local changes specific to individual modules have been made:

ambient: photon map code has been encapsulated in **ambPmap()** and **ambPmapCaustic()** in the new **pmapamb** module.

source: direct photon irradiance (useful for debugging photon emission) and volume photon inscattering are now handled by **multDirectPmap()** and **inscatterVolumePmap()**, respectively, which are included in the **pmapsrc** module.

rpmain, rtmain, rvmain rcmain: photon map initialisation and cleanup replaced with calls to **ray_init_pmap()** and **ray_done_pmap()** in new **pmapray** module. In addition, **rcmain** initialises photon map contributions via **initPmapContrib()** in the new **pmapcontrib** module.

renderopts: parsing of photon mapping parameters moved to **getPmapRenderOpt()**, output of defaults moved to **printPmapDefaults()** in new **pmapopt** module.

pmapinfo: deprecated, since plaintext statistics are now embedded in photon map files and output by ***getinfo***.

8 Acknowledgements

This research was supported by the Swiss National Science Foundation as part of the project "Simulation-based assessment of daylight redirecting components for energy savings in office buildings" (DRC, #147053).

Thanks to Greg Ward for his support in integrating the code, and Prof. Stephen Wittkopf and Lars Grobe for initiating the latest phase of development within the scope of the DRC project.

Thanks also to David Geisler-Moroder, who took the time to test the tool with REALLIFE™ scenes and report bugs and provide valuable feedback!

A Implementation Schematic

Figure 8 is a schematic overview of the modules comprising the photon mapping extension and how the interface with RADIANCE. Arrows indicate function calls (and therefore dependencie); where this is contained within the same module, it is denoted with nested blocks for the sake of legibility, with the outer block calling the inner.

Of particular note is the interface block, which hides the core photon mapping functionality from ***rpict***, ***rtrace***, and ***rvu*** via wrapper routines to facilitate maintenance.

Note also the recursion indicated by the arrow loop around **tracePhoton()** within the scattering block; this function is called by each material specific scattering routine (indexed through the **photonScatter[]** function lookup table) and terminated when the photon is absorbed as a result of Russian Roulette.

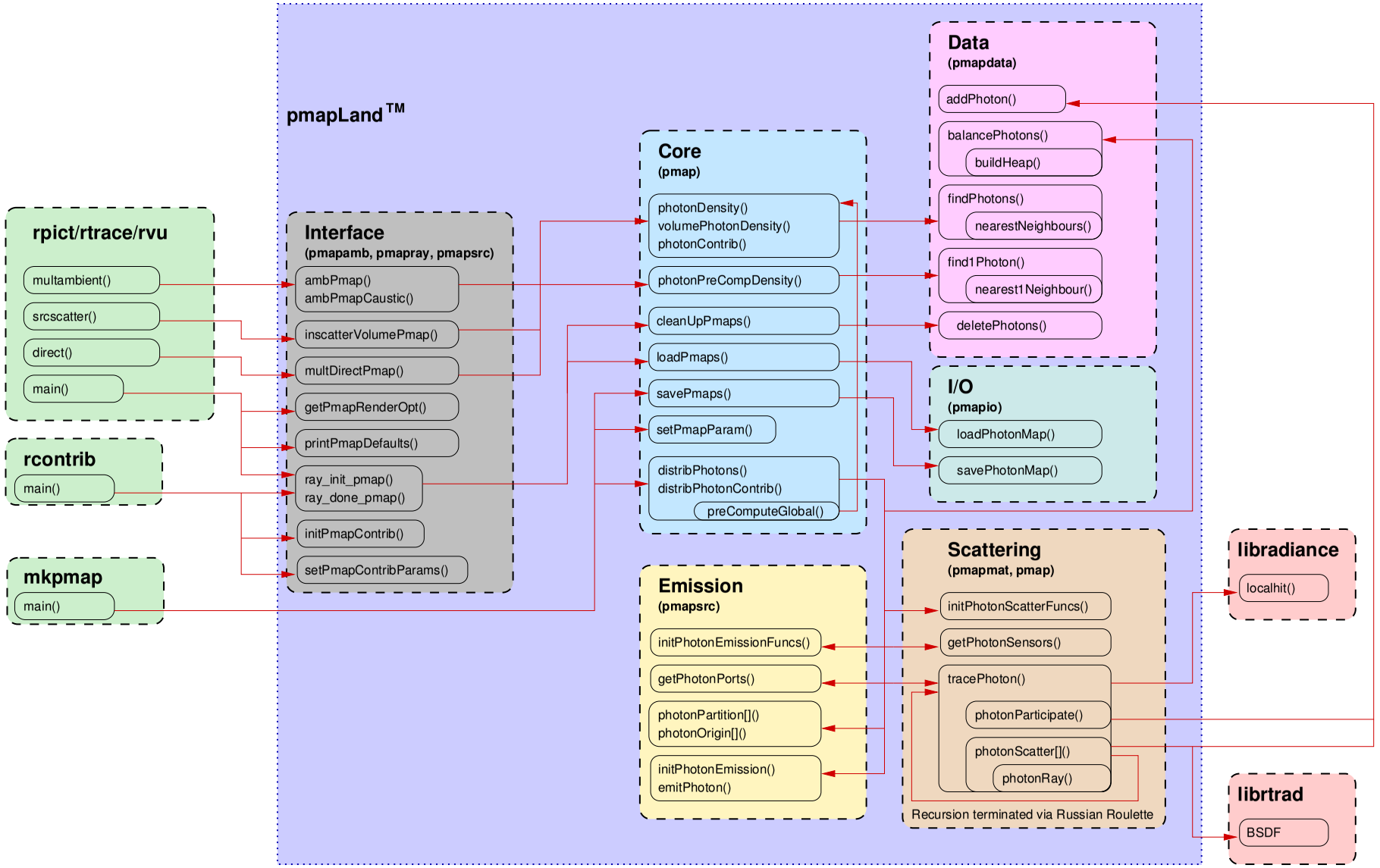

Figure 8: Schematic overview of photon mapping extension and its interface to RADIANCE.

References

[1] P. Apian-Bennewitz. New scanning gonio-photometer for extended brtf measurements. In *Proc. SPIE*, volume 7792, pages 77920O–77920O–20, 2010.

[2] J. Arvo and D. Kirk. Particle transport and image synthesis. In *Proceedings of the 17th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '90, pages 63–66, New York, NY, USA, 1990. ACM.

[3] J. L. Bentley. Multidimensional binary search trees used for associative searching. *Commun. ACM*, 18(9):509–517, Sept. 1975.

[4] J. M. Hammersley and D. C. Handscomb. *Monte Carlo methods*. Monographs on statistics and applied probability. Chapman and Hall, London, New York, 1965.

[5] H. W. Jensen. *Realistic Image Synthesis Using Photon Mapping*. A. K. Peters, Ltd., Natick, MA, USA, 2001.

[6] R. Rubinstein. *Simulation and the Monte Carlo Method*. Wiley Series in Probability and Statistics. Wiley, 1981.

[7] R. Schregle. *Daylight simulation with photon maps*. PhD thesis, Universität des Saarlandes, Saarbrücken, 2004.

[8] F. Suykens. *On robust Monte Carlo algorithms for multi-pass global illumination*. PhD thesis, Department of Computer Science, K.U.Leuven, Leuven, Belgium, Sept. 2002. Willems, Yves (supervisor).

[9] F. Suykens and Y. D. Willems. Density control for photon maps. In *Proceedings of the Eurographics Workshop on Rendering Techniques 2000*, pages 23–34, London, UK, UK, 2000. Springer-Verlag.

[10] G. Ward, M. Kurt, and N. Bonneel. A practical framework for sharing and rendering real-world bidirectional scattering distribution functions. Technical Report LBNL-5954E, Lawrence Berkeley National Laboratory, Oct. 2012.

[11] G. J. Ward. The radiance lighting simulation and rendering system. In *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '94, pages 459–472, New York, NY, USA, 1994. ACM.