Implementing Vertex Connection and Merging

Iliyan Georgiev*
Saarland University
Intel VCI, Saarbrücken

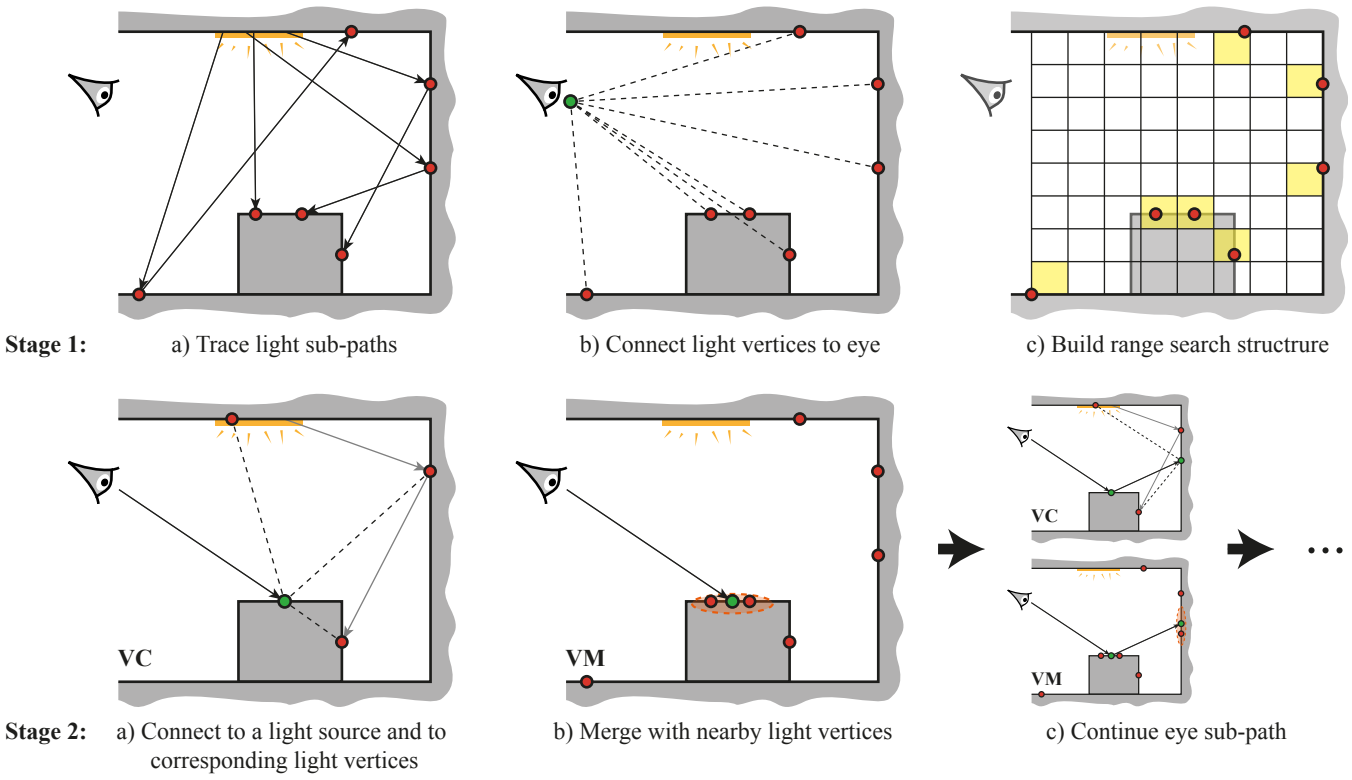


Figure 1: *An overview of our combined vertex connection and merging (VCM) algorithm. Rendering an image is performed in two stages. In the first stage, we trace sub-paths from the light sources, store their vertices, connect them to the eye, and additionally build a range search acceleration structure over them. In the second stage, we trace an eye sub-path for each pixel, and with each eye vertex we construct full paths via vertex connection (VC) and vertex merging (VM), evaluate their associated multiple importance sampling (MIS) weights, and accumulate the weighted contributions. The path weight evaluation is an important implementation aspect and is the main focus of this report. We present a way to efficiently evaluate the weight for a path using only data that is available at the two vertices being connected or merged, avoiding access to any other path vertices in memory. This also allows us to avoid storing the eye sub-path vertices, effectively making the second stage of the algorithm an extension of traditional path tracing with next event estimation, adding light vertex connection and merging techniques.*

Abstract

Bidirectional path tracing (BPT) and photon mapping (PM) are probably the two most versatile physically based rendering algorithms available today. It has been acknowledged that BPT and PM are complementary in terms of the types of light transport effects they can efficiently capture. Our recently proposed vertex connection and merging (VCM) algorithm aims to leverage the advantages of both methods by combining *vertex connection* techniques from BPT and *vertex merging* techniques from PM via multiple importance sampling [Georgiev et al. 2012]. We showed that this combined algorithm can efficiently capture a wide range of effects, and can be substantially more robust than either BPT or PM alone, while preserving the higher asymptotic performance of BPT.

The focus of our original paper is on the formal derivation, asymptotic analysis, and evaluation of the VCM algorithm. In this technical report, we address the most technically challenging part of its practical implementation – the multiple importance sampling (MIS) weighting. Indeed, correctly implementing MIS is already taxing in BPT, and VCM increases the complexity by adding even more path sampling techniques. More importantly, the efficient light sub-path reuse with vertex merging allows for cheaply constructing large amounts of full paths for each pixel, which in turn significantly increases the impact of path weight evaluation on the overall performance. The traditional BPT-style MIS weight computation that iterates over all path vertices can therefore become inefficient. We derive a new scheme to accumulate and store partial weight sums in the vertices of light and eye sub-paths. This allows us to efficiently compute the weight for a full path only using data cached at the two vertices that are connected or merged. The scheme is similar to the one independently developed by van Antwerpen [2011a; 2011b] for BPT, but in addition accounts for vertex merging techniques.

We also discuss how to handle infinite and singular light sources, cameras and materials with MIS, how to use per-pixel merging radii, and how to implement VCM in a memory efficient way.

*e-mail: georgiev@cs.uni-saarland.de

Symbol	Description	Appears in
$\overline{\mathbf{x}} = \mathbf{x}_0 \dots \mathbf{x}_k$	Full length-k path: vertex $\mathbf{x}_0$ is on a light source, $\mathbf{x}_k$ is on the eye lens	
$\overline{\mathbf{y}} = \mathbf{y}_0 \dots \mathbf{y}_{s-1}$	Light sub-path, with first vertex $\mathbf{y}_0 \equiv \mathbf{x}_0$ on a light	
$\overline{\mathbf{z}} = \mathbf{z}_0 \dots \mathbf{z}_{t-1}$	Eye sub-path, with first vertex $\mathbf{z}_0 \equiv \mathbf{x}_k$ on the eye lens	
$\overrightarrow{p}_i$, $\overleftarrow{p}_i$	Forward (i.e. actual) and reverse area pdfs for sub-path vertex i ($\overrightarrow{p}_i = \overrightarrow{p}_{\sigma,i}\,\overrightarrow{g}_i$ and $\overleftarrow{p}_i = \overleftarrow{p}_{\sigma,i}\,\overleftarrow{g}_i$)	(2), (6)
$\overrightarrow{p}_{\sigma,i}$, $\overleftarrow{p}_{\sigma,i}$	Forward and reverse solid angle pdfs for sub-path vertex i	(3), (7)
$\overrightarrow{g}_i$, $\overleftarrow{g}_i$	Forward and reverse pdf conversion factors from solid angle measure to area measure	(4), (8)
$p_{\text{VC},s,t}$, $p_{\text{VC},s}$	Vertex connection (VC) pdf for a length-k path with s light vertices and $t = k+1-s$ eye vertices	(10), (13)
$p_{\text{VM},s,t}$, $p_{\text{VM},s}$	Vertex merging (VM) pdf for a length-k path with s light vertices and $t = k+2-s$ eye vertices	(10), (13)
n_{VC}, n_{VM}	Number of samples (i.e. paths) used for vertex connection and vertex merging, respectively	(11)
$\eta_{\text{VCM}} = \frac{n_{\text{VM}}}{n_{\text{VC}}}\pi r^2$	Shorthand for all constants that appear in the multiple importance sampling path weight (12)	(20)

Table 1: *A list of some commonly used symbols in this document. Figure 2 illustrates the redundant (sub-)path notation and the pdf notation.*

1 Notation

The technical nature of this document requires the extensive use of mathematical notation. In this section, we introduce the notation for paths and their sampling probability densities. Table 1 summarizes some commonly used symbols in the document. For more definitions we refer to our original paper [Georgiev et al. 2012].

Paths. Veach's [1997] path integral formulation of light transport defines the value of a pixel as an integral over the space of all *paths*:

$$I = \int_\Omega f(\overline{\mathbf{x}})\,\mathrm{d}\mu(\overline{\mathbf{x}}) = \mathrm{E}\left[\frac{f(\overline{\mathbf{x}})}{p(\overline{\mathbf{x}})}\right] = \mathrm{E}\left[\langle I\rangle\right]. \tag{1}$$

A path of length k (edges) $\overline{\mathbf{x}} = \mathbf{x}_0 \dots \mathbf{x}_k$ is a tuple of $k+1$ vertices, where the vertex $\mathbf{x}_0$ is on a light source, and $\mathbf{x}_k$ is on the eye. A Monte Carlo estimator $\langle I\rangle$ for this integral can be constructed by sampling a random path $\overline{\mathbf{x}}$ and dividing its measurement contribution $f(\overline{\mathbf{x}})$ by its probability density $p(\overline{\mathbf{x}})$. Path tracing based methods do not sample the individual vertices on a path independently, but sequentially by performing random walks in scene.

Sub-paths. Bidirectional algorithms construct a full path $\overline{\mathbf{x}}$ by joining the endpoints of one *sub-path* traced from a light source and another one traced from the eye. We will denote a light sub-path with s vertices by $\overline{\mathbf{y}} = \mathbf{y}_0 \dots \mathbf{y}_{s-1}$ and an eye sub-path with t vertices by $\overline{\mathbf{z}} = \mathbf{z}_0 \dots \mathbf{z}_{t-1}$. Here, the vertex $\mathbf{y}_0$ is a point on a light source, and $\mathbf{z}_0$ is on the eye lens. These notations, illustrated in Figure 2, are redundant with the $\mathbf{x}$-notation, but conveniently index the vertices in the order of their generation. This symmetry will allow us to perform the same derivations for light and eye sub-paths. In particular, the forward and reverse probability density notation defined for $\overline{\mathbf{y}}$ below also applies to $\overline{\mathbf{z}}$.

Forward vertex pdfs. The probability density function (pdf) of a (sub-)path describes the joint distribution of its vertices via a chain of conditional vertex pdfs. We denote these vertex pdfs by:

$$\overrightarrow{p}_i(\overline{\mathbf{y}}) = \begin{cases} p(\mathbf{y}_0) & \text{if } i = 0, \\ \overrightarrow{p}_{\sigma,i}(\overline{\mathbf{y}})\,\overrightarrow{g}_i(\overline{\mathbf{y}}) & \text{otherwise,} \end{cases} \tag{2}$$

with

$$\overrightarrow{p}_{\sigma,i}(\overline{\mathbf{y}}) = \begin{cases} p_\sigma(\mathbf{y}_0 \to \mathbf{y}_1) & \text{if } i = 1, \\ p_\sigma(\mathbf{y}_{i-2} \to \mathbf{y}_{i-1} \to \mathbf{y}_i) & \text{if } i > 1 \end{cases} \tag{3}$$

$$\overrightarrow{g}_i(\overline{\mathbf{y}}) = \frac{\cos\theta_{i\to i-1}}{\|\mathbf{y}_i - \mathbf{y}_{i-1}\|^2}. \tag{4}$$

Above, $p(.)$ denotes an unconditional pdf expressed w.r.t. the area measure, e.g. $p(\mathbf{y}_0)$ for sampling $\mathbf{y}_0$ on a light source. The subscript σ denotes a solid angle pdf. The factor $\overrightarrow{g}_i$ converts the pdf

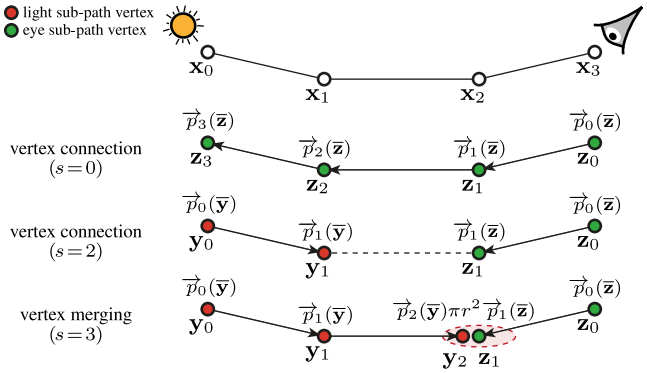


Figure 2: *An illustration of vertex connection and merging path sampling techniques, along with their associated vertex pdfs. The different VC and VM techniques for sampling length-k paths are identified by the number of light sub-path vertices s.*

measure from solid angle to area, with $\theta_{i\to i-1}$ being the angle between the surface normal at $\mathbf{y}_i$ and the unit vector $\overrightarrow{\mathbf{y}_i\mathbf{y}_{i-1}}$. We call $\overrightarrow{p}_i$ *forward* vertex probabilities (w.r.t. the random walk direction). With this notation, the pdf of a sub-path $\overline{\mathbf{y}}$ with s vertices is:

$$p_s(\overline{\mathbf{y}}) = \prod_{i=0}^{s-1} \overrightarrow{p}_i(\overline{\mathbf{y}}). \tag{5}$$

Reverse vertex pdfs. Analogously to the forward vertex pdf notation, we define a *reverse* notation:

$$\overleftarrow{p}_i(\overline{\mathbf{y}}) = \begin{cases} p(\mathbf{y}_k) & \text{if } i = k, \\ \overleftarrow{p}_{\sigma,i}(\overline{\mathbf{y}})\,\overleftarrow{g}_i(\overline{\mathbf{y}}) & \text{otherwise,} \end{cases} \tag{6}$$

with

$$\overleftarrow{p}_{\sigma,i}(\overline{\mathbf{y}}) = \begin{cases} p_\sigma(\mathbf{y}_{k-1} \leftarrow \mathbf{y}_k) & \text{if } i = k-1, \\ p_\sigma(\mathbf{y}_i \leftarrow \mathbf{y}_{i+1} \leftarrow \mathbf{y}_{i+2}) & \text{if } i < k-1 \end{cases} \tag{7}$$

$$\overleftarrow{g}_i(\overline{\mathbf{y}}) = \frac{\cos\theta_{i\to i+1}}{\|\mathbf{y}_i - \mathbf{y}_{i+1}\|^2}. \tag{8}$$

The arrow notation makes it easy to distinguish between the actual sampling pdf of a vertex, $\overrightarrow{p}_i(\overline{\mathbf{y}})$, and its "reverse" pdf, $\overleftarrow{p}_i(\overline{\mathbf{y}})$. That is, the latter denotes the probability density for sampling $\mathbf{y}_i$ in a direction opposite to that of the random walk. For example, in (6) $\mathbf{y}_k$ is a light sub-path vertex that has landed on the eye lens, and $p(\mathbf{y}_k)$ denotes the probability for sampling that point directly on the lens surface, i.e. as a part of an eye sub-path. These reverse pdfs are needed for the multiple importance sampling path weights. Recall that all notation above applies to eye sub-paths $\overline{\mathbf{z}}$ as well.

2 Vertex Connection and Merging

As we mentioned earlier, bidirectional path tracing and photon mapping complement each other in terms of the light transport effects they can efficiently capture. Aiming to combine their advantages, vertex connection and merging (VCM) [Georgiev et al. 2012] utilizes path sampling techniques from both algorithms:

- *Vertex connection* (VC) creates an edge between the endpoints of a light and an eye sub-paths. This is the sampling technique used in bidirectional path tracing.
- *Vertex merging* (VM) can intuitively be thought to concatenate the two sub-paths by virtually welding their endpoints, if they lie within a given distance r to each other. VM is our reformulation of photon mapping as a path sampling technique.

Figure 2 graphically illustrates the VC and VM techniques. In the rest of this section, we summarize the mathematical formulation of VCM and outline its algorithmic implementation.

2.1 Primary Estimators and Path Densities

Having a path $\overline{\mathbf{x}}$ sampled via vertex connection or merging, we can obtain a primary estimate for the pixel value (1) by dividing the measurement contribution of the path by its probability density:

$$\langle I \rangle_v(\overline{\mathbf{x}}) = \frac{f_{v,s,t}(\overline{\mathbf{x}})}{p_{v,s,t}(\overline{\mathbf{x}})}, \tag{9}$$

where v is either VC or VM, and s and t denote the number of vertices on the light and eye sub-paths, respectively. Taking into account that the sub-paths $\overline{\mathbf{y}}$ and $\overline{\mathbf{z}}$ are sampled independently, the full path pdfs for vertex connection and merging are, respectively:

$$\begin{aligned} p_{\text{VC},s,t}(\overline{\mathbf{x}}) &= p_s(\overline{\mathbf{y}}) p_t(\overline{\mathbf{z}}) \\ p_{\text{VM},s,t}(\overline{\mathbf{x}}) &= p_s(\overline{\mathbf{y}}) p_t(\overline{\mathbf{z}}) \pi r^2, \end{aligned} \tag{10}$$

where r is the vertex merging radius. The measurement contribution function $f_{v,s,t}$ has slightly different shapes for VC and VM. For its definition, as well as for the derivation of the VM path pdf, we refer to our paper [Georgiev et al. 2012]. It is important to note at this point that, unlike the paper, in this report we adhere to the intuitive definition of "vertex merging". That is, we consider the two merged vertices to be the endpoints of the eye and light sub-paths, as this view conveniently aligns with the implementation. With this interpretation, a full VM path (called *extended path* in the paper) constructed by a (VM, s, t) technique is one segment (edge) shorter than a VC path constructed by a (VC, s, t) technique (Figure 2).

2.2 Combined Estimator

The heart of our VCM algorithm is the secondary multiple importance sampling (MIS) pixel estimator that combines vertex connection estimators $\langle I \rangle_{\text{VC}}$ and vertex merging estimators $\langle I \rangle_{\text{VM}}$:

$$\begin{aligned} \langle I \rangle_{\text{VCM}} = \frac{1}{n_{\text{VC}}} \sum_{l=1}^{n_{\text{VC}}} \sum_{s \geq 0, t \geq 0} w_{\text{VC},s,t}(\overline{\mathbf{x}}_l) \, \langle I \rangle_{\text{VC}}(\overline{\mathbf{x}}_l) + \\ \frac{1}{n_{\text{VM}}} \sum_{l=1}^{n_{\text{VM}}} \sum_{s \geq 2, t \geq 2} w_{\text{VM},s,t}(\overline{\mathbf{x}}_l) \, \langle I \rangle_{\text{VM}}(\overline{\mathbf{x}}_l). \end{aligned} \tag{11}$$

It considers one eye sub-path through the corresponding pixel, whose vertices are connected to the vertices of n_{VC} light sub-paths and potentially merged with the vertices of n_{VM} light sub-paths. Our implementation uses $n_{\text{VC}} = 1$, and we set n_{VM} to the total number of light sub-paths, which for symmetry reasons we choose to be equal to the total number of eye sub-paths, i.e. the image resolution.

The *power heuristic* [Veach 1997] weight for technique (v, s, t) is

$$\begin{aligned} w_{v,s,t}(\overline{\mathbf{x}}) &= \frac{n_v^\beta \, p_{v,s,t}^\beta(\overline{\mathbf{x}})}{n_{\text{VC}}^\beta \sum_{s' \geq 0, t' \geq 0} p_{\text{VC},s',t'}^\beta(\overline{\mathbf{x}}) + n_{\text{VM}}^\beta \sum_{s' \geq 2, t' \geq 2} p_{\text{VM},s',t'}^\beta(\overline{\mathbf{x}})} \\ &= \frac{1}{\frac{n_{\text{VC}}^\beta}{n_v^\beta} \sum_{s' \geq 0, t' \geq 0} \frac{p_{\text{VC},s',t'}^\beta(\overline{\mathbf{x}})}{p_{v,s,t}^\beta(\overline{\mathbf{x}})} + \frac{n_{\text{VM}}^\beta}{n_v^\beta} \sum_{s' \geq 2, t' \geq 2} \frac{p_{\text{VM},s',t'}^\beta(\overline{\mathbf{x}})}{p_{v,s,t}^\beta(\overline{\mathbf{x}})}}, \end{aligned} \tag{12}$$

which takes into account all possible ways of sampling $\overline{\mathbf{x}}$ with vertex connection and merging. Note that the weight for a technique is amplified by the number of samples n_v, i.e. light paths, it uses.

2.3 Algorithm

Since equation (11) is a straightforward extension of Veach's [1997, p. 300] bidirectional path tracing (BPT) estimator, we could follow the classical BPT implementation for VCM as well. That is, for each pixel we would first trace one light and one eye sub-path, and then connect and merge each pair of opposing vertices, thereby reusing the sub-paths to compute multiple VC and VM estimates. However, the VM sub-path concatenation lends itself to a significantly more efficient reuse scheme: we can potentially merge each eye sub-path vertex with the vertices of *all* light sub-paths with a single range search, without tracing any rays. To be able to do this, we split rendering into two stages, as illustrated in Figure 1:

1. We first trace all light sub-paths and connect their vertices to the eye. Just like in photon mapping, we then build a range search structure over the vertices, e.g. a kd-tree or a hashed regular grid. Our implementation uses a hashed grid.
2. In the second stage, we trace an eye sub-path for every pixel. Each sampled eye vertex is first connected to a light source. We then connect it to the vertices of the light sub-path associated with the corresponding pixel. Finally, we merge the eye vertex with the vertices of all light sub-paths that fall within a user-specified search radius r. The primary estimate of each constructed full path is multiplied by a MIS weight before it is ultimately accumulated to the combined pixel estimate (11).

In the paper we provide pseudocode for the above algorithm. Note that we do not store the first vertex of any sub-path (i.e. $\mathbf{y}_0$ and $\mathbf{z}_0$), as we usually can efficiently reduce correlation by sampling a new vertex on a light source or on the camera lens for each connection.

The last step in the second stage – the path weight evaluation – is an important implementation aspect of the algorithm. Every path requires a MIS weight, and Veach's [1997] evaluation scheme can be inefficient for vertex merging which constructs a significantly larger number of paths than BPT. In the following section we will derive a new weight evaluation scheme that is more efficient than Veach's scheme for both vertex connection and vertex merging.

3 Efficient Path Weight Evaluation

The naïve way to evaluate the weight (12) is to compute all path pdfs in the denominator independently. For bidirectional path tracing, which uses the same formula, minus the VM sum, Veach [1997, p. 306] developed a more efficient scheme by exploiting the fact that many of the terms in the fractions cancel out when the path pdfs are expanded. He computes the VC sum by looping once over the light and eye sub-path vertices, accumulating the pdf fractions.

While Veach's scheme can be easily extended to vertex merging, it has sub-optimal efficiency. First, it makes many redundant computations, as every time a sub-path is reused (for connecting or

merging) the same terms associated with its vertices are recomputed. Moreover, it requires accessing every path vertex in memory. Weight computation can therefore become a significant overhead for vertex merging, which relies heavily on sub-path reuse and often constructs a large number of full paths with a single range query.

To see how to solve these problems, notice that the unweighted contribution of a path is actually computed efficiently in BPT, and also in VCM. Each sub-path vertex stores its throughput, i.e. the accumulated contribution terms from all preceding sub-path vertices [Veach 1997, p. 304]. Upon connecting or merging two vertices, the unweighted contribution, i.e. the primary estimate (9), is quickly evaluated using only the data stored at those two vertices. We set out to apply the same efficient scheme to path weight evaluation as well. To this end, we reformulate the two sums in (12) as recursive quantities that can be incrementally computed and cached at the sub-path vertices as we perform the random walks.

At this point, readers not interested in the formal derivation of our new scheme can skip to Section 4. There we discuss its practical implementation, describing the quantities stored at each sub-path vertex and how to compute the full path weight from this data.

3.1 Partial Sub-Path Weights

In order to keep the notation simple, in this section we will consider paths $\overline{\mathbf{x}}$ of length k with s light sub-path vertices, often omitting the redundant subscript t. Also, without loss of generality, we will assume $\beta = 1$, i.e. that the balance heuristic is used. We now rewrite the path weight (12) more compactly:

$$w_{v,s,t} = \frac{1}{\frac{n_{\text{VC}}}{n_v}\sum_{j=0}^{k+1}\frac{p_{\text{VC},j}}{p_{v,s}} + \frac{n_{\text{VM}}}{n_v}\sum_{j=2}^{k}\frac{p_{\text{VM},j}}{p_{v,s}}}, \tag{13}$$

where $p_{v,j}$ is the pdf for sampling a length-k full path using a light sub-path with j vertices, and $v \in \{\text{VC}, \text{VM}\}$.

We now write the path weight in the form

$$w_{v,s,t} = \frac{1}{w_{v,s}^{\text{light}} + 1 + w_{v,s}^{\text{eye}}}, \tag{14}$$

where we have rearranged the sums in (13) to iterate over the light and eye sub-path vertices, respectively, and have extracted the term $p_{v,s}/p_{v,s} = 1$ from the appropriate sum. The partial light and eye sub-path weights $w_{v,s}^{\text{light}}$ and $w_{v,s}^{\text{eye}}$ have slightly different shapes depending on the value of v.

Vertex connection. For paths sampled with $v = \text{VC}$ we have:

$$w_{\text{VC},s}^{\text{light}} = \sum_{j=0}^{s-1}\frac{p_{\text{VC},j}}{p_{\text{VC},s}} + \frac{n_{\text{VM}}}{n_{\text{VC}}}\sum_{j=2}^{s}\frac{p_{\text{VM},j}}{p_{\text{VC},s}} \tag{15}$$

$$w_{\text{VC},s}^{\text{eye}} = \sum_{j=s+1}^{k+1}\frac{p_{\text{VC},j}}{p_{\text{VC},s}} + \frac{n_{\text{VM}}}{n_{\text{VC}}}\sum_{j=s+1}^{k}\frac{p_{\text{VM},j}}{p_{\text{VC},s}}. \tag{16}$$

Vertex merging. For paths sampled with $v = \text{VM}$ we have:

$$w_{\text{VM},s}^{\text{light}} = \frac{n_{\text{VC}}}{n_{\text{VM}}}\sum_{j=0}^{s-1}\frac{p_{\text{VC},j}}{p_{\text{VM},s}} + \sum_{j=2}^{s-1}\frac{p_{\text{VM},j}}{p_{\text{VM},s}} \tag{17}$$

$$w_{\text{VM},s}^{\text{eye}} = \frac{n_{\text{VC}}}{n_{\text{VM}}}\sum_{j=s}^{k+1}\frac{p_{\text{VC},j}}{p_{\text{VM},s}} + \sum_{j=s+1}^{k}\frac{p_{\text{VM},j}}{p_{\text{VM},s}}. \tag{18}$$

To summarize, the weight for a full path in VCM is given by equation (14). Its terms are given by equations (15)-(16) or (17)-(18), depending on the technique used to construct the path.

3.2 Recursive Formulation

We now reformulate the path weight (14) in a form that is suitable for evaluation in a *forward* manner, i.e. in the order of the generation of the sub-paths, instead of iterating backwards from their endpoints as Veach [1997, p. 306] suggests. We write the weight as

$$w_{v,s,t} = \frac{1}{\overline{w}_{v,s-1}(\overline{\mathbf{y}}) + 1 + \overline{w}_{v,t-1}(\overline{\mathbf{z}})}, \tag{19}$$

where $\overline{w}_{v,s-1}(\overline{\mathbf{y}})$ and $\overline{w}_{v,t-1}(\overline{\mathbf{z}})$ are recursive formulations of the partial light and eye sub-path weights $w_{v,s}^{\text{light}}$ and $w_{v,s}^{\text{eye}}$. These two quantities can be tracked and updated as we trace the sub-paths $\overline{\mathbf{y}}$ and $\overline{\mathbf{z}}$, and be readily available for summing up when we connect or merge the sub-path endpoints. We now derive these recursive quantities separately for VC and VM.

In the following, we will use a shorthand notation to group all constants that appear in the path weight into a single term:

$$\boxed{\eta_{\text{VCM}} = \frac{n_{\text{VM}}}{n_{\text{VC}}}\pi r^2.} \tag{20}$$

Vertex connection. Using the path pdf definitions (10), we follow Veach [1997, p. 306] to expand the pdf fractions in (15):

$$w_{\text{VC},s}^{\text{light}} = \sum_{j=0}^{s-1}\prod_{i=j}^{s-1}\frac{\overleftarrow{p}_i(\overline{\mathbf{y}})}{\overrightarrow{p}_i(\overline{\mathbf{y}})} + \eta_{\text{VCM}}\sum_{j=2}^{s}\overleftarrow{p}_{j-1}(\overline{\mathbf{y}})\prod_{i=j}^{s-1}\frac{\overleftarrow{p}_i(\overline{\mathbf{y}})}{\overrightarrow{p}_i(\overline{\mathbf{y}})}, \tag{21}$$

where we use the forward and reverse vertex pdf notations from (2) and (6). Next, we reformulate the VC and VM sums as two recursive quantities (omitting the $\overline{\mathbf{y}}$ arguments for readability):

$$\begin{aligned} \overline{w}_{\text{VC},0}^{\text{VC}} &= \frac{\overleftarrow{p}_0}{\overrightarrow{p}_0} & \overline{w}_{\text{VC},0}^{\text{VM}} &= 0 \\ \overline{w}_{\text{VC},i}^{\text{VC}} &= \frac{\overleftarrow{p}_i}{\overrightarrow{p}_i}\left(1 + \overline{w}_{\text{VC},i-1}^{\text{VC}}\right) & \overline{w}_{\text{VC},i}^{\text{VM}} &= \overleftarrow{p}_i\left(1 + \frac{1}{\overrightarrow{p}_i}\,\overline{w}_{\text{VC},i-1}^{\text{VM}}\right), \end{aligned} \tag{22}$$

and write the partial light sub-path weight (15) as

$$w_{\text{VC},s}^{\text{light}} = \overline{w}_{\text{VC},s-1}^{\text{VC}}(\overline{\mathbf{y}}) + \eta_{\text{VCM}}\,\overline{w}_{\text{VC},s-1}^{\text{VM}}(\overline{\mathbf{y}}). \tag{23}$$

We can further combine $\overline{w}_{\text{VC},i}^{\text{VC}}$ and $\overline{w}_{\text{VC},i}^{\text{VM}}$ to formulate (23) as a single recursive quantity (again omitting the $\overline{\mathbf{y}}$ arguments):

$$\boxed{\overline{w}_{\text{VC},0} = \frac{\overleftarrow{p}_0}{\overrightarrow{p}_0}} \tag{24}$$

$$\boxed{\overline{w}_{\text{VC},i} = \overleftarrow{p}_i\left(\eta_{\text{VCM}} + \frac{1}{\overrightarrow{p}_i} + \frac{1}{\overrightarrow{p}_i}\overline{w}_{\text{VC},i-1}\right)} \tag{25}$$

Finally, with the eye sub-path notation $\overline{\mathbf{z}}$ (Figure 2) the above recursive formula applies without any modification to (16) as well, allowing us to plug $\overline{w}_{\text{VC},s-1}(\overline{\mathbf{y}})$ and $\overline{w}_{\text{VC},t-1}(\overline{\mathbf{z}})$ into (19).

Vertex merging. We now expand equation (17) as above:

$$w_{\text{VM},s}^{\text{light}} = \frac{1}{\eta_{\text{VCM}}}\frac{1}{\overrightarrow{p}_{s-1}(\overline{\mathbf{y}})}\sum_{j=0}^{s-1}\prod_{i=j}^{s-2}\frac{\overleftarrow{p}_i(\overline{\mathbf{y}})}{\overrightarrow{p}_i(\overline{\mathbf{y}})} + \sum_{j=2}^{s-1}\prod_{i=j}^{s-1}\frac{\overleftarrow{p}_{i-1}(\overline{\mathbf{y}})}{\overrightarrow{p}_i(\overline{\mathbf{y}})}. \tag{26}$$

Using the same methodology as for vertex connection, we obtain:

$$\boxed{\overline{w}_{\text{VM},1} = \frac{1}{\overrightarrow{p}_1}\left(\frac{1}{\eta_{\text{VCM}}} + \overleftarrow{p}_0\frac{1}{\eta_{\text{VCM}}\,\overrightarrow{p}_0}\right)} \tag{27}$$

$$\boxed{\overline{w}_{\text{VM},i} = \frac{1}{\overrightarrow{p}_i}\left(\frac{1}{\eta_{\text{VCM}}} + \overleftarrow{p}_{i-1} + \overleftarrow{p}_{i-1}\overline{w}_{\text{VM},i-1}\right)} \tag{28}$$

Again, using the $\overline{\mathbf{z}}$ notation for eye sub-paths, we can apply the above formula to (18) as well, which allows us to plug $\overline{w}_{\text{VM},s-1}(\overline{\mathbf{y}})$ and $\overline{w}_{\text{VM},t-1}(\overline{\mathbf{z}})$ into (19) for the weight for a vertex-merged path.

4 Practical Implementation

Equations (24)-(25) and (27)-(28) define the partial weight quantities $\overline{w}_{\text{VC},i}$ and $\overline{w}_{\text{VM},i}$ associated with the sub-path vertices. Ideally, we want to cache these at every light and eye vertex as we trace the sub-paths. Then, upon connecting or merging any two vertices, the full weight would be obtained by simply summing up their respective $\overline{w}_{\text{VC},i}$ or $\overline{w}_{\text{VM},i}$ quantities, as postulated by equation (19).

Unfortunately, the above scheme cannot be directly implemented, since $\overline{w}_{\text{VC},i}$ and $\overline{w}_{\text{VM},i}$ require reverse probabilities that are not yet known at the point of sampling sub-path vertex i. More specifically, $\overleftarrow{p}_i(\overline{\mathbf{y}})$ depends on the next two vertices via $p_\sigma(\mathbf{y}_i \leftarrow \mathbf{y}_{i+1} \leftarrow \mathbf{y}_{i+2})$. Similarly, $\overleftarrow{p}_{i-1}(\overline{\mathbf{y}})$ depends on the next sub-path vertex, $\mathbf{y}_{i+1}$.

Luckily, an efficient implementation of the scheme is made possible by splitting up its computations. To this end, at $\mathbf{y}_i$ we cache only (and all) the terms that depend on the sub-path vertices sampled up to and including $\mathbf{y}_i$. We extract three vertex quantities from $\overline{w}_{\text{VC},i}$ and $\overline{w}_{\text{VM},i}$, postponing the evaluation of the remaining terms until we have information about the required vertices, e.g. when we sample the next sub-path vertex or couple $\mathbf{y}_i$ with another vertex:

$$\overline{w}_{\text{VC},i} = \overleftarrow{p}_i \left(\eta_{\text{VCM}} + \underbrace{\frac{1}{\overrightarrow{p}_i}}_{\boxed{d_i^{\text{VCM}}}} + \underbrace{\frac{1}{\overrightarrow{p}_i}\,\overline{w}_{\text{VC},i-1}}_{\overleftarrow{p}_{\sigma,i-1}\,\boxed{d_i^{\text{VC}}}} \right) \tag{29}$$

$$\overline{w}_{\text{VM},i} = \underbrace{\frac{1}{\overrightarrow{p}_i}}_{\boxed{d_i^{\text{VCM}}}} \frac{1}{\eta_{\text{VCM}}} + \overleftarrow{p}_{\sigma,i-1} \underbrace{\frac{\overleftarrow{g}_{i-1}}{\overrightarrow{p}_i}\left(1 + \overline{w}_{\text{VM},i-1}\right)}_{\boxed{d_i^{\text{VM}}}}, \tag{30}$$

where we have used the expansion $\overleftarrow{p}_{i-1} = \overleftarrow{p}_{\sigma,i-1}\,\overleftarrow{g}_{i-1}$, with $\overleftarrow{p}_{\sigma,i-1}$ being the solid angle reverse pdf for sub-path vertex $i-1$. Note that d_i^{VCM} appears in the path weight for both VC and VM, whereas d_i^{VC} and d_i^{VM} are specific to VC and VM, respectively. Also, recall from Section 3.2 that the recursive formulas above apply to both the light sub-paths $\overline{\mathbf{y}}$ and the eye sub-paths $\overline{\mathbf{z}}$.

4.1 Sub-Path Vertex Data

As we trace a sub-path, we update and store the quantities d_i^{VCM}, d_i^{VC} and d_i^{VM} at each vertex. Their formulas are the same for all light light and eye vertices, with the only exception for $\mathbf{y}_1$ and $\mathbf{z}_1$. The reason is that we do not consider path sampling techniques with zero eye sub-path vertices, as the probability of hitting the camera lens is usually very low (or even zero in the case of a pinhole camera model). Also, recall from Section 2.3 that we do not store $\mathbf{y}_0$ and $\mathbf{z}_0$. The precise data we store at every sub-path vertex are:

$$\mathbf{y}_1 : d_1^{\text{VCM}} = \frac{p_0^{\text{connect}}}{p_0^{\text{trace}}} \frac{1}{\overrightarrow{p}_1} \qquad \mathbf{z}_1 : d_1^{\text{VCM}} = \frac{p_0^{\text{connect}}}{p_0^{\text{trace}}} \frac{n_{\text{light}}}{\overrightarrow{p}_1} \tag{31}$$

$$d_1^{\text{VC}} = \frac{\overleftarrow{g}_0}{p_0^{\text{trace}}\overrightarrow{p}_1} \qquad d_1^{\text{VC}} = 0 \tag{32}$$

$$d_1^{\text{VM}} = \frac{\overleftarrow{g}_0}{p_0^{\text{trace}}\overrightarrow{p}_1 \eta_{\text{VCM}}} \qquad d_1^{\text{VM}} = 0 \tag{33}$$

$$\mathbf{y}_i,\, \mathbf{z}_i : d_i^{\text{VCM}} = \frac{1}{\overrightarrow{p}_i} \tag{34}$$

$$d_i^{\text{VC}} = \frac{\overleftarrow{g}_{i-1}}{\overrightarrow{p}_i}\left(\eta_{\text{VCM}} + d_{i-1}^{\text{VCM}} + \overleftarrow{p}_{\sigma,i-2}\, d_{i-1}^{\text{VC}}\right) \tag{35}$$

$$d_i^{\text{VM}} = \frac{\overleftarrow{g}_{i-1}}{\overrightarrow{p}_i}\left(\ 1 \ + \frac{d_{i-1}^{\text{VCM}}}{\eta_{\text{VCM}}} + \overleftarrow{p}_{\sigma,i-2}\, d_{i-1}^{\text{VM}}\right) \tag{36}$$

In the equations above, p_0^{connect} and p_0^{trace} account for the fact that different techniques may be used for sampling a vertex on a light source or on the eye lens, depending on whether it will be connected to a sub-path or used to start a new sub-path. The total number of light sub-paths, n_{light}, is the number of samples the eye connection technique $(\text{VC}, s, 1)$ takes. We have obtained d_i^{VC} and d_i^{VM} by recursively expanding $\overline{w}_{\text{VC},i-1}$ and $\overline{w}_{\text{VM},i-1}$ in equations (29) and (30). Note that the three floating point quantities d_i are the only path weight related data that we need to store with the sub-path vertices.

4.2 Full Path Weight

The weight for a full path constructed from a light sub-path $\overline{\mathbf{y}}$ with s vertices and an eye sub-path $\overline{\mathbf{z}}$ with t vertices is given by

$$w_{v,s,t} = \frac{1}{\overline{w}_{v,s-1}(\overline{\mathbf{y}}) + 1 + \overline{w}_{v,t-1}(\overline{\mathbf{z}})}. \tag{37}$$

This equation is the same as (19), but we also include it here, for easy reference. We will next show how to compute $\overline{w}_{v,s-1}(\overline{\mathbf{y}})$ and $\overline{w}_{v,t-1}(\overline{\mathbf{z}})$ from the vertex quantities d_i, depending on the technique $v \in \{\text{VC}, \text{VM}\}$. The general-case formulas for VC and VM (i.e. for $s > 1$ and $t > 1$) follow directly from (29) and (30). Note that due to the symmetry in the notation for light and eye sub-paths, these general-case formulas are the same for $\overline{\mathbf{y}}$ and $\overline{\mathbf{z}}$.

Vertex merging $(s > 1, t > 1)$**.** A path is constructed by merging light sub-path vertex $\mathbf{y}_{s-1}$ and eye sub-path vertex $\mathbf{z}_{t-1}$:

$$\overline{w}_{\text{VM},s-1}(\overline{\mathbf{y}}) = \frac{d_{s-1}^{\text{VCM}}}{\eta_{\text{VCM}}} + \overleftarrow{p}_{\sigma,s-2}\, d_{s-1}^{\text{VM}} \tag{38}$$

$$\overline{w}_{\text{VM},t-1}(\overline{\mathbf{z}}) = \frac{d_{t-1}^{\text{VCM}}}{\eta_{\text{VCM}}} + \overleftarrow{p}_{\sigma,t-2}\, d_{t-1}^{\text{VM}}. \tag{39}$$

Vertex connection $(s > 1, t > 1)$**.** A path is constructed by connecting the light vertex $\mathbf{y}_{s-1}$ to the eye vertex $\mathbf{z}_{t-1}$:

$$\overline{w}_{\text{VC},s-1}(\overline{\mathbf{y}}) = \overleftarrow{p}_{s-1}\left(\eta_{\text{VCM}} + d_{s-1}^{\text{VCM}} + \overleftarrow{p}_{\sigma,s-2}\, d_{s-1}^{\text{VC}}\right) \tag{40}$$

$$\overline{w}_{\text{VC},t-1}(\overline{\mathbf{z}}) = \overleftarrow{p}_{t-1}\left(\eta_{\text{VCM}} + d_{t-1}^{\text{VCM}} + \overleftarrow{p}_{\sigma,t-2}\, d_{t-1}^{\text{VC}}\right). \tag{41}$$

Vertex connection $(s = 0)$**.** The eye sub-path vertex $\mathbf{z}_{t-1}$ is sampled on a light source, i.e. the light sub-path has zero vertices:

$$\overline{w}_{\text{VC},s-1}(\overline{\mathbf{y}}) = 0 \tag{42}$$

$$\overline{w}_{\text{VC},t-1}(\overline{\mathbf{z}}) = p_{t-1}^{\text{connect}}\, d_{t-1}^{\text{VCM}} + p_{t-1}^{\text{trace}}\, \overleftarrow{p}_{\sigma,t-2}\, d_{t-1}^{\text{VC}}. \tag{43}$$

Vertex connection $(s = 1)$**.** The eye sub-path vertex $\mathbf{z}_{t-1}$ is connected to vertex $\mathbf{y}_0$ on a light source (a.k.a. next event estimation):

$$\overline{w}_{\text{VC},0}(\overline{\mathbf{y}}) = \frac{\overleftarrow{p}_0}{p_0^{\text{connect}}} \tag{44}$$

$$\overline{w}_{\text{VC},t-1}(\overline{\mathbf{z}}) = \frac{p_0^{\text{trace}}(\overline{\mathbf{y}})}{p_0^{\text{connect}}(\overline{\mathbf{y}})}\, \overleftarrow{p}_{t-1}\left(\eta_{\text{VCM}} + d_{t-1}^{\text{VCM}} + \overleftarrow{p}_{\sigma,t-2}\, d_{t-1}^{\text{VC}}\right). \tag{45}$$

Vertex connection $(t = 1)$**.** The light sub-path vertex $\mathbf{y}_{s-1}$ is connected to vertex $\mathbf{z}_0$ on the eye lens (a.k.a. eye/camera projection):

$$\overline{w}_{\text{VC},s-1}(\overline{\mathbf{y}}) = \frac{p_0^{\text{trace}}(\overline{\mathbf{z}})}{p_0^{\text{connect}}(\overline{\mathbf{z}})} \frac{\overleftarrow{p}_{s-1}}{n_{\text{light}}}\left(\eta_{\text{VCM}} + d_{s-1}^{\text{VCM}} + \overleftarrow{p}_{\sigma,s-2}\, d_{s-1}^{\text{VC}}\right) \tag{46}$$

$$\overline{w}_{\text{VC},0}(\overline{\mathbf{z}}) = 0. \tag{47}$$

Recall that n_{light} is the total number of light sub-paths, which is the number of samples this eye connection technique uses.

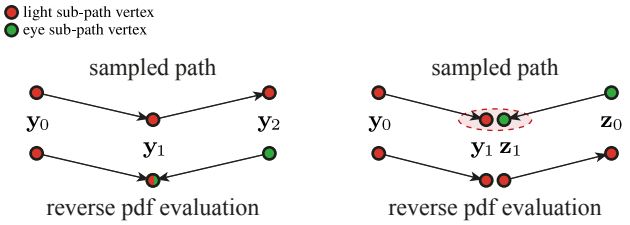


Figure 3: *Reverse pdf evaluation. Left: The VM pdf for the unidirectionally sampled segment is evaluated by interpreting $\mathbf{y}_1$ as the point where two "merged" vertices coincide. Right: The directional sampling pdf for a vertex-merged segment is evaluated at the eye sub-path vertex $\mathbf{z}_1$ with directions $\overrightarrow{\mathbf{y}_0\mathbf{y}_1}$ and $\overrightarrow{\mathbf{z}_1\mathbf{z}_0}$.*

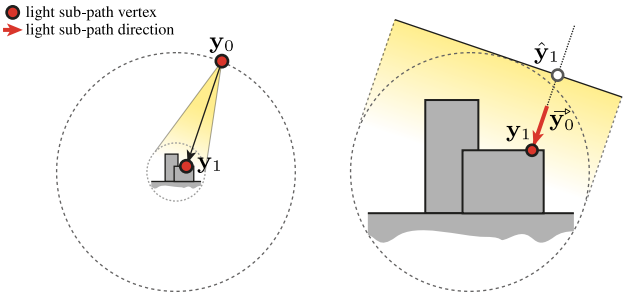


Figure 4: *Left: Turning an infinite light source to a finite spherical light allows for expressing path pdfs w.r.t. a product area measure, but is only approximate and can lead to numerical inaccuracies. Right: We avoid the issues by handling infinite lights via solid angle integration, and derive the corresponding path pdfs for use in MIS.*

4.3 Reverse Pdf Evaluation

Having constructed a path segment unidirectionally or via an explicit vertex connection, evaluating the pdf for vertex merging is straightforward, as it can also sample the segment. This is illustrated in Figure 3 left, where the "merged" vertices coincide at $\mathbf{y}_1$.

The opposite case, shown in Figure 3 right, is slightly more complicated, as vertex merging evaluates light transport along approximate paths that cannot be sampled any other way. Nevertheless, we can still evaluate the pdf for directionally sampling a similar segment along the same path edges, as shown in the figure. The pdf for explicit vertex connection is evaluated by considering the actual connection edge between the corresponding light and eye vertices. That is, in the example in Figure 3 right, for a connection between $\mathbf{y}_0$ and $\mathbf{z}_1$ we have $p_{\text{VC},1} = \overrightarrow{p}_0(\overline{\mathbf{y}})\,\overrightarrow{p}_0(\overline{\mathbf{z}})\,\overrightarrow{p}_1(\overline{\mathbf{z}})$, and for a connection between $\mathbf{y}_1$ and $\mathbf{z}_0$ we have $p_{\text{VC},2} = \overrightarrow{p}_0(\overline{\mathbf{y}})\,\overrightarrow{p}_1(\overline{\mathbf{y}})\,\overrightarrow{p}_0(\overline{\mathbf{z}})$.

5 Special Cases

In this section, we discuss how to handle infinite light sources and orthographic cameras with multiple importance sampling, and how to deal with materials and light sources whose definitions involve delta distributions. We also show how to apply our recursive weight evaluation scheme to bidirectional path tracing and to bidirectional photon mapping.

5.1 Infinite Light Sources

Light sources that are located very (or even infinitely) far away from the rest of the scene geometry are usually not defined via emitting surfaces but via directional *incident radiance* distributions at the receiving points. This prevents the straightforward use of infinite lights in VCM, as they cannot be handled by the path integral (1) which only considers area integration [Veach 1997, p. 222].

A common approach is to turn infinite lights into finite area lights by mapping their directional emission onto a large sphere that surrounds the scene and emits inwards (with properly scaled power). These area lights are naturally handled by the path integral, but they only approximate the incident radiance distribution of the original infinite light. Enlarging the sphere improves the approximation accuracy, but can cause numerical issues due to the large distance between the light and the scene and also due to the small solid angles involved in the emission sampling for light sub-paths (see Figure 4).

We can avoid the problems by handling infinite lights in their original formulation, i.e. using solid angle integration. Paths involving an infinite light source now have the form $\overline{\mathbf{x}} = \vec{\mathbf{x}}_0^{\triangleright}\mathbf{x}_1 \dots \mathbf{x}_k$, where we have replaced the point $\mathbf{x}_0$ with a direction $\vec{\mathbf{x}}_0^{\triangleright}$. We start a light sub-path $\overline{\mathbf{y}}$ by first sampling $\vec{\mathbf{y}}_0^{\triangleright}$ (deterministically for directional lights). We then sample a point $\hat{\mathbf{y}}_1$ on a plane perpendicular to $\vec{\mathbf{y}}_0^{\triangleright}$ and project it onto the scene along $\vec{\mathbf{y}}_0^{\triangleright}$ to obtain $\mathbf{y}_1$ (see Figure 4 right and also Pharr and Humphreys [2010, p. 714]). We preserve the path pdf notation used so far, but accommodate for the changes in the path geometry and sampling procedure via the following modifications:

- $p_0^{\text{connect}}(\overline{\mathbf{y}}) = p_\sigma^{\text{connect}}(\vec{\mathbf{y}}_0^{\triangleright})$ and $p_0^{\text{trace}}(\overline{\mathbf{y}}) = p_\sigma^{\text{trace}}(\vec{\mathbf{y}}_0^{\triangleright})$ are now expressed w.r.t. the solid angle measure,
- $\overleftarrow{p}_0(\overline{\mathbf{y}}) = \overleftarrow{p}_{\sigma,0}(\overline{\mathbf{y}}) = p_\sigma(\mathbf{y}_0 \leftarrow \mathbf{y}_1 \leftarrow \mathbf{y}_2)$ is now a solid angle probability density as well,
- $\overleftarrow{g}_0(\overline{\mathbf{y}}) = 1$, as no pdf measure conversion is needed anymore (a consequence of $\overleftarrow{p}_0(\overline{\mathbf{y}}) = \overleftarrow{p}_{\sigma,0}(\overline{\mathbf{y}})$ above),
- $\overrightarrow{p}_1(\overline{\mathbf{y}}) = p(\hat{\mathbf{y}}_1)\cos\theta_{1\to0}$, where $\theta_{1\to0}$ is now the angle between the normal at $\mathbf{y}_1$ and $-\vec{\mathbf{y}}_0^{\triangleright}$.

The pdf modifications for $\vec{\mathbf{z}}_k^{\triangleright} \equiv \vec{\mathbf{y}}_0^{\triangleright}$ and $\mathbf{z}_{k-1} \equiv \mathbf{y}_1$ are symmetric.

5.2 Orthographic Cameras

When an orthographic camera model is used, the same modifications we made to the path geometry and the sub-path pdfs for infinite lights apply on the eye side as well. That is, vertex $\mathbf{x}_k$ in $\overline{\mathbf{x}}$ is first replaced with a direction $\vec{\mathbf{x}}_k^{\triangleright}$, and then the pdf modifications above also apply to $\vec{\mathbf{z}}_0^{\triangleright} \equiv \vec{\mathbf{y}}_k^{\triangleright}$ and $\mathbf{z}_1 \equiv \vec{\mathbf{y}}_{k-1}^{\triangleright}$.

5.3 Point and Directional Light Sources

Omnidirectional and spot lights have infinitely small areas, and directional lights emit in a single direction. Such light sources cannot be hit by random rays, as their emission is defined via a delta distribution, i.e. we have $p_{\text{VC},0} = 0$. The light sub-path quantities d_1 then become identical to the eye sub-path quantities d_1, minus the n_{light} factor in d_1^{VCM}:

$$\mathbf{y}_1 : d_1^{\text{VCM}} = \frac{p_0^{\text{connect}}}{p_0^{\text{trace}}}\frac{1}{\overrightarrow{p}_1} = \frac{1}{\overrightarrow{p}_1} \tag{48}$$

$$d_1^{\text{VC}} = 0 \tag{49}$$

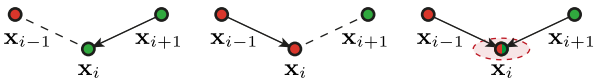
$$d_1^{\text{VM}} = 0. \tag{50}$$

Recall that the eye sub-path quantities were originally defined differently because we do not allow randomly hitting the eye lens. Also, equation (44) simplifies to $\overline{w}_{\text{VC},0}(\overline{\mathbf{y}}) = 0$. Finally, since emission is defined only at a single point or in a single direction, we set the "connect" and "trace" probabilities in equations (43) and (45) to 1, since the light source sampling is deterministic.

5.4 Specular Materials

Directional scattering from materials like mirror and glass is defined using delta distributions that cannot be handled by Lebesgue integration. In practice, this means that path vertices with such delta BSDFs cannot be connected to or merged with other vertices. Hence, certain path sampling techniques become impossible, and their zero probabilities need to be accounted for in the MIS weights. Specifically, if specular scattering is sampled at vertex $\mathbf{x}_i$ in a random walk, then:

$$p_{\text{VC},i} = 0 \qquad p_{\text{VC},i+1} = 0 \qquad p_{\text{VM},i+1} = 0 \tag{51}$$



Also, since the scattering direction at $\mathbf{x}_i$ is deterministic, we set:

$$\begin{aligned} \overrightarrow{p}_{\sigma,i+1} &= p_\sigma(\mathbf{x}_{i-1} \to \mathbf{x}_i \to \mathbf{x}_{i+1}) = 1 \\ \overleftarrow{p}_{\sigma,i-1} &= p_\sigma(\mathbf{x}_{i-1} \leftarrow \mathbf{x}_i \leftarrow \mathbf{x}_{i+1}) = 1. \end{aligned} \tag{52}$$

When the BSDF is a mixture of distributions (e.g. reflection and refraction), the pdfs in (52) are equal to the probability of choosing the particular specular scattering component at $\mathbf{x}_i$ in the random walk. Note also that since we do not allow randomly hitting the camera, purely specular paths, i.e. LS^*E paths (using Heckbert's [1990] regular expression notation), can only be sampled unidirectionally, and thus have the trivial MIS weight of 1.

We can account for the zero-probability techniques in the path weight as we trace the sub-paths. If the scattering event at vertex $i-1$ is specular, then the d_i quantities for vertex i simplify to:

$$\mathbf{y}_i,\, \mathbf{z}_i : d_i^{\text{VCM}} = 0 \tag{53}$$

$$d_i^{\text{VC}} = \frac{\overleftarrow{g}_{i-1}}{\overrightarrow{p}_i} \overleftarrow{p}_{\sigma,i-2}\, d_{i-1}^{\text{VC}} \tag{54}$$

$$d_i^{\text{VM}} = \frac{\overleftarrow{g}_{i-1}}{\overrightarrow{p}_i} \overleftarrow{p}_{\sigma,i-2}\, d_{i-1}^{\text{VM}}. \tag{55}$$

Note that with these modifications, our scheme correctly handles the cases where the BSDF is a mixture of specular and non-specular components. Even if we sample specular scattering at vertex $i-1$ in the random walk, we do not modify its weight quantities d_{i-1}. The vertex is still stored and used for connection and merging to construct other paths, for which the weight formulas from Section 4.2 apply as usual.

5.5 Bidirectional Path Tracing

The path weight evaluation scheme from Section 4 can be easily applied to traditional BPT by restring the formulas to only account for vertex connections. This is achieved by simply setting $\eta_{\text{VCM}} = 0$ in equations (35), (40)-(47), and also eliminating the vertex quantity d_i^{VM}. With these modifications, our scheme becomes nearly identical to the one proposed by van Antwerpen [2011a; 2011b].

5.6 Bidirectional Photon Mapping

Bidirectional photon mapping [Vorba 2011] is a special case of VCM that uses multiple importance sampling for vertex merging only. Analogously to the bidirectional path tracing case above, restricting the weighting to vertex merging techniques is as simple as setting the terms involving η_{VCM} in equations (33), (36), (38) and (39) to zero, and also eliminating the vertex quantity d_i^{VC}.

6 Extensions

In this section, we first discuss how to accommodate per-pixel vertex merging radii in our MIS weight accumulation scheme, and then go on to describe a progressive VCM variant that is much more memory-efficient than the one presented in Section 2.3.

6.1 Per-Pixel Merging Radius

The merging radius r is an important VCM parameter that controls the performance of the vertex merging techniques and their relative weights in the combined estimator (11). The optimal radius size may vary across the scene, however our MIS weight accumulation scheme from Section 4 implicitly assumes that all merging queries use the same global radius r. This is because the cumulative sub-path vertex quantities d_i^{VC} and d_i^{VM} depend on r via η_{VCM}. If we want to use different merging radii at different locations in the scene and still obtain correct MIS weights, we need to make the vertex quantities independent of r. To do this, we avoid computing and storing d_i^{VC} and d_i^{VM} at every vertex during sub-path construction, and instead keep track of two new quantities, c_i^{VC} and c_i^{VM}:

$$\mathbf{y}_1 : d_1^{\text{VCM}} = \frac{p_0^{\text{connect}}}{p_0^{\text{trace}}} \frac{1}{\overrightarrow{p}_1} \qquad \mathbf{z}_1 : d_1^{\text{VCM}} = \frac{p_0^{\text{connect}}}{p_0^{\text{trace}}} \frac{n_{\text{light}}}{\overrightarrow{p}_1} \tag{56}$$

$$c_1^{\text{VC}} = \frac{\overleftarrow{g}_0}{p_0^{\text{trace}} \overrightarrow{p}_1} \qquad c_1^{\text{VC}} = 0 \tag{57}$$

$$c_1^{\text{VM}} = 0 \qquad c_1^{\text{VM}} = 0 \tag{58}$$

$$\mathbf{y}_i,\, \mathbf{z}_i : d_i^{\text{VCM}} = \frac{1}{\overrightarrow{p}_i} \tag{59}$$

$$c_i^{\text{VC}} = \frac{\overleftarrow{g}_{i-1}}{\overrightarrow{p}_i} \left(d_{i-1}^{\text{VCM}} + \overleftarrow{p}_{\sigma,i-2}\, c_{i-1}^{\text{VC}} \right) \tag{60}$$

$$c_i^{\text{VM}} = \frac{\overleftarrow{g}_{i-1}}{\overrightarrow{p}_i} \left(\quad 1 \quad + \overleftarrow{p}_{\sigma,i-2}\, c_{i-1}^{\text{VM}} \right) \tag{61}$$

For any vertex, d_i^{VC} and d_i^{VM} can be computed as follows:

$$d_i^{\text{VC}} = c_i^{\text{VC}} + \eta_{\text{VCM}}\, c_i^{\text{VM}} \tag{62}$$

$$d_i^{\text{VM}} = \frac{c_i^{\text{VC}}}{\eta_{\text{VCM}}} + c_i^{\text{VM}} \tag{63}$$

The evaluation of η_{VCM}, and thus of d_i^{VC} and d_i^{VM}, can now be postponed to the point when an actual pixel estimate is constructed using a particular eye vertex. This allows us to choose a different radius r for every eye sub-path, e.g. derived from the pixel footprint. Note that for any given full path we must use the same r in the MIS weights of all possible techniques, so they sum up to one.

6.2 Memory-Efficient Implementation

In Section 2.3 we described a two-stage implementation of the VCM algorithm, where we first trace all light sub-paths and store their vertices, similarly to photon mapping. Since we also use these vertices for connections in the second stage, we need to cache their associated BSDF structures as well. In some renderers, the shading structure at a surface point can be as large as 1KB[1], requiring gigabytes of storage for millions of light vertices. This memory issue does not appear in bidirectional path tracing (BPT), where pixels are rendered independently and every pair of light and eye

[1] Shading structures often store the results from multiple texture queries, local geometric and shading coordinate frames and derivatives, as well as reflectance distribution parameters.

sub-paths is immediately discarded upon connection. Photon mapping can also maintain a low memory footprint, as it always evaluates the BSDF at *eye* vertices, thereby avoiding the need for storing shading data at light vertices.

To reduce the memory footprint of light vertices, we modify progressive VCM to operate in a single stage as follows. At every iteration, for each pixel we first trace one light sub-path. We then store its vertices in a separate list without their BSDF structures, just like in photon mapping, but with the addition of the three cumulative weight quantities. After that, we trace an eye sub-path through the pixel. Every eye vertex is connected to the light sub-path and also merged with all nearby light vertices from the *previous* iteration. Finally, we discard both sub-paths. After processing all pixels, we build a range search acceleration structure over the light vertex list and dispose the structure from the previous iteration. Rendering begins by tracing an initial set of light sub-paths that are only used for merging at the first iteration. Alternatively, we can skip merging at the first iteration, and scale the VM contributions at the second iteration by a factor of two.

With the above modifications, the VCM algorithm becomes very similar to traditional BPT, with the extension of caching all light vertices at every iteration and then using them for merging at the subsequent iteration. By delaying vertex merging by one iteration, its memory footprint becomes almost as low as that of photon mapping, with only three additional floats per light vertex.

7 Conclusions

In this technical report, we addressed important aspects of the practical implementation of the vertex connection and merging (VCM) algorithm [Georgiev et al. 2012]. We first derived a computationally efficient scheme for evaluating the multiple importance sampling (MIS) path weights. This scheme avoids redundant computations, and more importantly, significantly reduces memory access, making the algorithm well suited for GPU implementation. We then described how to handle special cases such as infinite and singular light sources, cameras and materials. We finally discussed how to accommodate per-pixel merging radii in the MIS weights and how the progressive VCM algorithm can be restructured to significantly improve its memory-efficiency.

Limitations. One constraint that our recursive path weight evaluation scheme poses is that no terms in the weight can depend on the (sub-)path length. For example, we cannot base the Russian roulette path termination probability or the number of vertex connections on the eye sub-path length. The reason is that we need to be able to compute these quantities when we accumulate reverse probabilities during the *light* sub-path tracing, without any knowledge of the full path length. Van Antwerpen [2011a] also points out this limitation. One way to mitigate it is to consider such terms constant or completely exclude them from the MIS weight. Alternatively, per-vertex storage can be augmented with additional partial weights that are specialized for connections to sub-paths of different lengths.

Reference implementation. We provide an implementation of the VCM algorithm that uses our path weight evaluation scheme in the open source renderer SmallVCM [Davidovič and Georgiev 2012]. This implementation covers all special cases discussed in Section 5. The renderer also includes traditional path tracing and light tracing.

Acknowledgements. The author would like to thank Tomáš Davidovič for implementing SmallVCM, and Miloš Hašan and Jaroslav Křivánek for the insights into handling infinite lights. Also thanks to Anton Kaplanyan, Tomáš Davidovič and Jaroslav Křivánek for proofreading the text.

References

DAVIDOVIČ, T., AND GEORGIEV, I., 2012. SmallVCM renderer. http://www.smallvcm.com and https://github.com/SmallVCM.

GEORGIEV, I., KŘIVÁNEK, J., DAVIDOVIČ, T., AND SLUSALLEK, P. 2012. Light transport simulation with vertex connection and merging. *ACM Trans. Graph. 31* (December). (Proc. SIGGRAPH Asia 2012).

HECKBERT, P. S. 1990. Adaptive radiosity textures for bidirectional ray tracing. In *SIGGRAPH '90*, ACM, New York, USA.

PHARR, M., AND HUMPHREYS, G. 2010. *Physically Based Rendering: From Theory To Implementation*, 2nd ed. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

VAN ANTWERPEN, D. 2011. Recursive MIS computation for streaming BDPT on the GPU. Tech. rep., Delft University of Technology.

VAN ANTWERPEN, D. 2011. A survey of importance sampling applications in unbiased physically based rendering. Tech. rep., Delft University of Technology.

VEACH, E. 1997. *Robust Monte Carlo Methods for Light Transport Simulation*. PhD thesis, Standford Univeristy.

VORBA, J. 2011. Bidirectional photon mapping. In *Proc. Central European Seminar on Computer Graphics (CESCG '11)*.