

# Path differentials and Applications

*Frank Suykens*

*Yves Willems*

*Report CW 307, May 2001*



Katholieke Universiteit Leuven

Department of Computer Science

Celestijnenlaan 200A – B-3001 Heverlee (Belgium)

# Path differentials and Applications

*Frank Suykens*

*Yves Willems*

*Report CW 307, May 2001*

Department of Computer Science, K.U.Leuven

## **Abstract**

Photo-realistic rendering algorithms such as Monte Carlo ray tracing sample individual paths to compute images. Noise and aliasing artefacts are usually reduced by supersampling. Knowledge about the neighborhood of the path, such as an estimated footprint, can be used to reduce these artefacts without having to trace additional paths. The recently introduced ray differentials estimate such a footprint for classical ray tracing, by computing ray derivatives with respect to the image plane. The footprint proves to be useful for filtering textures locally on surfaces. In this paper, we generalize the use of these derivatives to arbitrary path sampling, including general reflection and refraction functions. Sampling new directions introduces additional partial derivatives, which are all combined into a footprint estimate. Additionally the path gradient is introduced; it gives the rate of change of the path contribution. When this change is too steep the size of the footprint is reduced. The resulting footprint can be used in any global illumination algorithm that is based on path sampling. Two applications show its potential: texture filtering in distributed ray tracing and a novel hierarchical approach to particle tracing radiosity.

**Keywords :** BRDF sampling, Global Illumination, Monte Carlo.

**CR Subject Classification :** I.3.7

# Path differentials and Applications

Frank Suykens, Yves D. Willems

May 2001

## Abstract

Photo-realistic rendering algorithms such as Monte Carlo ray tracing sample individual paths to compute images. Noise and aliasing artefacts are usually reduced by supersampling. Knowledge about the neighborhood of the path, such as an estimated footprint, can be used to reduce these artefacts without having to trace additional paths. The recently introduced ray differentials estimate such a footprint for classical ray tracing, by computing ray derivatives with respect to the image plane. The footprint proves to be useful for filtering textures locally on surfaces. In this paper, we generalize the use of these derivatives to arbitrary path sampling, including general reflection and refraction functions. Sampling new directions introduces additional partial derivatives, which are all combined into a footprint estimate. Additionally the path gradient is introduced; it gives the rate of change of the path contribution. When this change is too steep the size of the footprint is reduced. The resulting footprint can be used in any global illumination algorithm that is based on path sampling. Two applications show its potential: texture filtering in distributed ray tracing and a novel hierarchical approach to particle tracing radiosity.

This is a companion technical report to the paper published in *Rendering Techniques 2000* [15]. It contains some more technical details, mostly in the form of appendices.

## 1 Introduction

The traditional image pipeline processes the scene primitives one by one and renders them on screen. Ray tracing on the other hand takes point samples on the image plane, and traces infinitely thin rays through the scene. This allows an easy simulation of reflection and refraction effects. Extensions such as Monte Carlo ray tracing handle arbitrary bidirectional reflection functions (BRDF's) for even more photo-realistic images.

These methods still use ray tracing as the engine to compute light transport. Because a ray (and a path) is a point sample, with no information about its neighborhood, these algorithms are prone to aliasing or noise. The common solution is supersampling, averaging the evaluation of many paths. This is expensive, and researchers have tried to exploit coherence in the neighborhood of the rays to reduce aliasing or noise.

One particularly interesting approach presented by Igehy [8] computes ray differentials, partial derivatives of a ray with respect to the position on the image plane. The footprint of a ray is approximated by the differential vectors: the partial derivatives multiplied by a finite distance on the image plane. The differentials give an idea about the distance to neighboring rays. This proves to be very effective for filtering textures locally over the footprint. Only perfectly specular reflections and refractions are supported, limiting the technique to classical ray tracing.

In this paper we extend the concept of ray differentials to arbitrary sampled paths, including arbitrary reflection and refraction functions and area light source sampling. A key observation is that the sampling of BRDF's or light sources introduces new degrees of freedom in the generation of

the path. This gives rise to extra partial derivatives and differential vectors that must all be combined into a useful estimate of the paths footprint. We will show how to compute the new partial derivatives and present a simple heuristic to derive a useful footprint from these derivatives.

However the (image) contribution of the path does not need to be constant over the estimated footprint. For example real BRDF's can be highly nonlinear and first order differential approximations may not be accurate enough. We also propose to track *path evaluation* derivatives, that tell us how fast the (image) contribution changes over the differential vectors. This *path gradient* is used to refine the footprint estimate; a smaller footprint is used for large gradients.

The combined result forms a convenient footprint estimate for arbitrary sampled eye-paths and light-paths. Therefore our techniques can be applied to any global illumination algorithm that is based on path sampling.

Two applications demonstrate the potential of the method:

- **Texture Filtering:** A classical ray tracer is extended with glossy reflections and refractions. Textures are filtered locally over the estimated footprint to reduce noise.
- **Particle Tracing:** A hierarchical refinement criterion for particle tracing radiosity is presented. The estimated footprint of single paths is sufficient to determine an appropriate level of subdivision.

Many other applications are possible. Some possibilities are discussed in the conclusions (Section 6).

Section 3 explains the framework for computing partial derivatives and differentials for arbitrary sampled paths. Section 4 shows how to compute the path gradient and Section 5 demonstrates applications.

## 2 Related work

Several researchers have tried to exploit coherence in the neighborhood of a path by tracking some kind of footprint.

In beam tracing [7] a ray is extended to a beam with a polygonal footprint. Intersections with and reflection from objects (polygonal only) fragment the beam in smaller pieces. Lighting calculations can be done coherently over the beam.

Cone tracing [1] uses a conical approximation of the footprint. Both approaches allow anti-aliased textures by filtering over the footprint. Cone tracing allows ad hoc dull or glossy reflections by explicitly increasing the cone angle.

Another approach, pencil tracing [13] propagates a bundle of paraxial rays through the scene. Interactions of the rays with the scene are approximated by a linear system matrix.

All these approaches extend a ray to finite width, which makes intersection, reflection and refraction calculations much more difficult. The combination of physically based BRDF's with these methods is a difficult problem.

Collins [5] uses standard ray tracing but the connectivity between neighboring starting rays is explicitly maintained when propagating them through the scene. The distance to neighboring rays was used to determine a filter kernel size for deposition in a caustic lightmap. Connectivity is lost, though, when neighboring rays hit different objects resulting in different ray trees. Stochastic sampling is possible but may diverge the ray trees even more.

As mentioned an interesting alternative are ray differentials as proposed by Igehy [8]. Partial derivatives with respect to the position on the image plane are computed along with the tracing of the rays. Since derivatives are used, a ray stays infinitesimally thin, but the sensitivity of the ray's direction and origin is given by these partial derivatives. The footprint of a ray is approximated by the

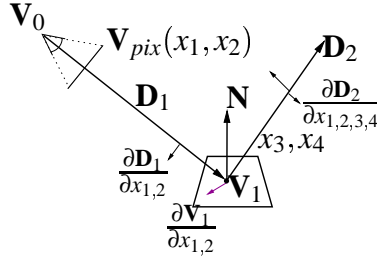


Figure 1: Tracing an eye ray and a scattered ray introduces new variables  $x_j$  in the path, for which partial derivatives need to be computed.

differential vectors: the partial derivatives multiplied by a finite distance on the image plane. When a ray is propagated, reflected or refracted, the partial derivatives of the new ray is easily computed using differential calculus.

The use of derivatives alleviates many of the problems that previously mentioned techniques have, because differential calculus is used a ray or path stays infinitely thin. Our techniques are also based on differential calculus, preserving these advantages while it is not limited to classical ray tracing.

Another interesting use of derivatives of paths was presented by Chen and Arvo [4]. They compute first and second order derivatives of specular reflection paths. Using these derivatives, small perturbations of the paths can be computed very efficiently. The perturbations allow an accurate approximation of neighboring paths. An application was presented that perturbs a sparse set of known paths to efficiently compute reflections on implicit surfaces.

### 3 Path differentials

#### 3.1 Path sampling and path footprint

Stochastic ray tracing constructs paths by sampling. Newly sampled directions (or vertices) depend on the previous directions and vertices and possibly on some new variables. Figure 1 shows a short eye path where  $\mathbf{D}_1$  and  $\mathbf{V}_1$  depend on variables  $x_1, x_2$ , a position on the image plane. The reflected direction  $\mathbf{D}_2$  depends on  $x_1, x_2$  as well, but also on new variables  $x_3, x_4$  determined by the BRDF sampling. In stochastic ray tracing random numbers are used to instantiate the variables and corresponding paths.

For a certain vertex  $\mathbf{V}$  (or a direction) in the path one can say that:

$$\mathbf{V} = g(x_1, x_2, \dots, x_k) = g(X_k)$$

with  $g$  the *path generation function*, and  $k$  the number of variables that  $\mathbf{V}$  depends on.

A small perturbation  $\epsilon_j$  applied to a variable  $x_j$  slightly moves vertex  $\mathbf{V}$ :

$$\mathbf{V} + \delta\mathbf{V}_j = g(x_1, \dots, x_j + \epsilon_j, \dots, x_k)$$

This change can be approximated by a first order Taylor expansion:

$$\delta\mathbf{V}_j \approx \frac{\partial g(x_1, \dots, x_j, \dots, x_k)}{\partial x_j} \epsilon_j$$

The magnitude of the partial derivative determines the sensitivity of  $\mathbf{V}$  in terms of  $x_j$ . Simultaneous perturbation of several variables corresponds to a  $\delta\mathbf{V} = \sum_j \delta\mathbf{V}_j$ .

If we consider all perturbations  $\epsilon_j \in [-\Delta x_j/2, \Delta x_j/2]$  then the set of perturbed vertices  $\delta \mathbf{V}_j$  forms a line segment defined by a vector centered around  $\mathbf{V}$ :

$$\Delta \mathbf{V}_j = \frac{\partial g(x_1, \dots, x_k)}{\partial x_j} \Delta x_j$$

We will call these the *differential vectors*.

Given a perturbation interval  $\Delta x_j$  for each variable and allowing simultaneous perturbation of the variables, the set of all possible perturbed vertices  $\mathbf{V} + \delta \mathbf{V}$  forms an area<sup>1</sup>. We call this area *the footprint* of the path for vertex  $\mathbf{V}$ .

The shape of the footprint is a polygon with the differential vectors as edges (see 3.5). In Igehy's approach only two variables exist, the image plane coordinates. The footprint is a parallelogram formed by the two differential vectors.

Given the perturbation intervals  $\Delta x_j$ , the footprint estimates the region of influence or the sensitivity of the path in a vertex  $\mathbf{V}$ . A suitable choice for  $\Delta x_j$  should ensure coherence over the footprint while being large enough to reduce noise and aliasing (see 3.4 and 4). The computation the partial derivatives themselves is detailed in 3.3.

Other definitions of a path footprint are also possible. For instance a filter kernel could be defined for each differential vector. The resulting 'footprint' filter would be the convolution of these filter kernels. Using Gaussian kernels turns out to be interesting as the convolution of elliptic Gaussians is again an elliptic Gaussian [3].

## 3.2 Sampling Domain

The set of variables  $x_j$  determines the domain of all possible paths. We choose a unit interval  $[0, 1]$  as the domain of each  $x_j$ , corresponding to the random numbers that are used in stochastic sampling. Importance sampling can be used to transform variables with a different, desired distribution.

The domain of all possible paths with  $M$  degrees of freedom ( $g(X_M)$ ) is the  $M$ -dimensional unit hypercube. A point in the domain determines a path. Such a uniform domain will simplify the choice of  $\Delta x_j$ 's when constructing differentials from the partial derivatives (see 3.4).

## 3.3 Partial derivatives

Tracing paths involves sampling of new directions and vertices. For stochastic sampling a certain probability density function (pdf) determines the distribution of the new direction or vertex. Importance sampling [10] is a well known procedure to sample according to a given pdf. In general sampling a new direction (or vertex) with an importance sampling procedure  $h$  derived from the pdf gives:

$$\mathbf{D}' = h(\mathbf{D}, \mathbf{V}, x, y)$$

where  $\mathbf{D}'$  is a new direction dependent on the previous direction and vertex (if these exist) and some new random variables  $x, y$  (a 2D sampling in most cases). Partial derivatives of  $\mathbf{D}'$  can be computed by simply deriving  $h$  for all random variables  $x_j, x, y$ . Note that  $\mathbf{D}$  and  $\mathbf{V}$  depend on previous  $x_j$ . We will briefly describe all relevant sampling events. More details on the derivative computation can be found in Appendix A.

---

<sup>1</sup>Partial derivatives of a vertex do lie in a plane perpendicular to the surface normal due to ray transfer computation (see 3.3)

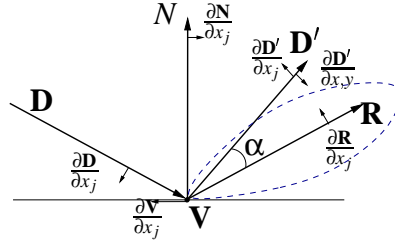


Figure 2: Partial derivatives for phong lobe sampling.  $\mathbf{D}'$  has derivatives for previous sampling variables  $x_j$  but also for new variables  $x, y$  from BRDF sampling

**Pixel sampling** The initial vertex  $\mathbf{V}_0$  of a path is the eye. (This could already be a sampling event, but is not considered as such here.) Pixel sampling generates a ray direction  $\mathbf{D}_1$  based on a randomly chosen point in a certain pixel. Computation of the two derivatives of  $\mathbf{D}_1$  is given in A.1.

**Transfer** Transfer computes a new point in the path by tracing a ray:  $\mathbf{V}' = \mathbf{V} + t\mathbf{D}$ , with  $t$  the traveled distance. No new sampling occurs, so existing derivatives of  $\mathbf{V}$  and  $\mathbf{D}$  are used to compute  $\frac{\partial \mathbf{V}'}{\partial x_j}$ . All partial derivatives of  $\mathbf{V}'$  lie in the plane perpendicular to the geometric normal. See A.2 for more details.

**Scattering** Scattering in a vertex  $\mathbf{V}$  given an incoming direction  $\mathbf{D}$  determines a new direction  $\mathbf{D}'$ . The pdf for direction sampling is usually chosen proportional to the BRDF or 'BRDF  $\times$  cosine'. Partial derivatives of the resulting sampling procedure must be computed.

An example for a glossy phong BRDF is shown in figure 2. The new direction  $\mathbf{D}'$  is distributed according to  $\cos^s \alpha$  around the perfectly reflected direction  $\mathbf{R}$ . Since  $\mathbf{D}'$  depends on  $\mathbf{R}$  and  $\mathbf{R}$  on  $\mathbf{D}$ ,  $\frac{\partial \mathbf{D}'}{\partial x_j}$  can be different from zero.

The new derivatives  $\frac{\partial \mathbf{D}'}{\partial x_j}$  depend on the specific sampling procedure. For example for uniform sampling of a hemisphere:

$$\begin{aligned} \phi &= 2\pi x, \quad \cos(\theta) = 1 - y \\ \mathbf{D}' &= h(x, y) = (\cos(\phi)\sin(\theta), \sin(\phi)\sin(\theta), \cos(\theta)) \end{aligned}$$

Partial derivatives of  $h(x, y)$  are easily computed.

A more detailed example using a phong BRDF and a coordinate transform is given in A.3. For perfectly specular reflection or refraction (deterministic!) no new random variables are introduced. This case was handled in [8].

**Light sampling** When light paths are constructed, a starting point must be chosen on a light and a light direction must be sampled. Again partial derivatives of the specific sampling procedure are easily computed (see A.4).

Any other sampling event not covered here, can usually be easily derived from the sampling procedure.

### 3.4 Choice of $\Delta x_j$

Given a vertex  $\mathbf{V}$  we have computed a number of partial derivatives  $\frac{\partial \mathbf{V}}{\partial x_j}$ . In this section we will derive a heuristic for choosing intervals  $\Delta x_j$  that correspond to the expected distance to a neighboring sample (the closest sample  $(x_1 \dots x_k)$  differing only in  $x_j$ ).

Multiplying the deltas with the corresponding derivative results in differential vectors that approximate the expected distance to a neighboring similar path. For example differential vectors from sampling a Lambertian reflection will usually be larger compared to a glossy Phong reflection, because the partial derivatives are larger while the deltas are the same. The rays get 'spread out' more.

The footprint is inversely proportional to the expected density of similar paths around a vertex.

For classical ray tracing [8] only two differential vectors have to be considered: those with respect to the position on the image plane. The deltas are chosen to be the size of a pixel, the distance to the next sample on the image plane.

In our case we have to make a choice for each  $x_j$  possibly coming from very different sampling events. As said, each variable has a unit interval domain. We consider two approaches for choosing deltas.

**Local deltas** Each  $\Delta x_j$  depends on the number of samples that was used in the sampling event that introduced  $x_j$ . For example if  $N$  samples per pixel are traced, these samples are distributed over the unit square.  $\Delta x_0$  should be chosen as an approximate distance to a neighboring sample. For regular sampling this distance is  $1/\sqrt{N}$  for both  $\Delta x_0$  and  $\Delta x_1$  and we have found this distance to be useful for stochastic sampling also.

If extra information about the sampling process is known (e.g. nonuniform stratification), different values for  $\Delta x_0$  and  $\Delta x_1$  may be better.

For scattering, a 2D sampling event, a splitting factor  $N'$  determines how many scattered samples are spawn. Again we choose deltas to be  $1/\sqrt{N'}$ . If many samples are spawn, the corresponding differentials will be smaller.

**Global deltas** The previous approach does not work well with path tracing, where a large number of samples per pixel is used, but the splitting factor is 1.

In this case we consider the complete  $M$ -dimensional domain of a path, and considerer the samples evenly distributed over the domain. An estimate of the distance to a neighboring sample in one dimension is now given by  $1/\sqrt[M]{N}$ . All  $\Delta x_j$  are chosen equally large.

Longer paths will have a larger delta, as  $N$  samples have to be distributed over a higher dimensional domain.

*Russian roulette*, an unbiased way to limit the length of paths, can be incorporated in this approach. Absorption probabilities  $P_{rr}$  are accumulated along the path, and deltas are computed as  $1.0/\sqrt[M]{N \times \prod_i P_{rr}(\mathbf{V}_i)}$ . Thus the number of samples for 'this kind of path' is decreased by the absorption probabilities, and deltas grow larger.

This approach works well for path tracing and also particle tracing as demonstrated in the second application.

### 3.5 Differential vectors to footprint

Consider a vertex  $\mathbf{V}$  and its (planar) differential vectors  $\Delta \mathbf{V}_j$  (Fig. 3). As said the footprint of the path in  $\mathbf{V}$  is the area reachable by perturbation of the path within the chosen  $\Delta x_j$ 's.

This area can be constructed from the line segments defined by the differential vectors. Any perturbed vertex  $\mathbf{V}'$  is a combination of points on the line segments. The area defined by a set of line segments is the Minkowski sum ( $\oplus$ ) of these segments (centered around  $\mathbf{V}$ ):

$$\bigoplus_j \Delta \mathbf{V}_j = \{ \mathbf{V}' = \sum_j \gamma_j \Delta \mathbf{V}_j \mid -0.5 < \gamma_i < 0.5 \}$$

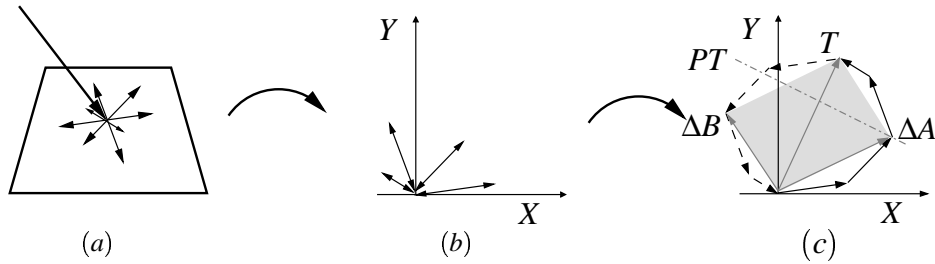


Figure 3: Each path vertex has several differential vectors lying in the same plane (a). These vectors describe the footprint of the path. By transforming to the  $X - Y$  plane (b), this polygonal footprint can be constructed by combining the vectors in a particular order (c). A good approximation is given by vectors  $\Delta A$  and  $\Delta B$ .

For two differentials this sum is a parallelogram formed by the two vectors. In general the Minkowski sum forms a polygon where each vector appears twice as an edge. This polygon can be constructed as follows (see fig. 3):

- Transform the differential vectors to the 2D  $X - Y$  plane. (a to b)
- Sort the segments according to the angle made with  $X$  (b)
- Add the sorted vectors one by one as edges of the polygon (reaching the top of the polygon) (c solid)
- Add edges by subtracting the vectors in the same order, starting at the top of the polygon (c dashed)

To perform operations such as texture filtering over the footprint, a convenient representation is needed. For more than two differential vectors the area expression becomes impractical. Therefore we compute two representative vectors  $\Delta A$  and  $\Delta B$  that give a good approximation of the covered area (see 3 (c)):

- $T =$  the sum of all differential vectors
- $PT =$  a vector perpendicular to  $T$
- $\Delta A =$  sum of all vectors  $\Delta \mathbf{V}_j$  that have  $\Delta \mathbf{V}_j \cdot PT > 0$
- $\Delta B = T - \Delta A$

Constructing the representative vectors processes all differential vectors twice (For  $T$  and  $\Delta A$ ; no sort is needed). This is a linear operation in terms of the number of differentials.

$\Delta A$  and  $\Delta B$  now provide a convenient estimate of the footprint of the path in  $\mathbf{V}$ .

## 4 Path gradient

In the previous section,  $\Delta x_j$  were chosen to approximate the distance to neighboring samples. In this section we propose an alternative choice for  $\Delta x_j$  based on the rate of change of the path contribution when perturbations are applied.

Any function defined on a path is a function of the generating variables  $x_j$ . In this section we compute partial derivatives of the path evaluation, the function that determines the contribution to the quantity (e.g. pixel flux) we want to compute.

We present the technique for eye paths, but it is equally applicable to light paths.

## 4.1 Rendering equation

The light flux reaching the eye  $\mathbf{V}_0$  through a certain pixel is given as an integral of a pixel weighting function  $W_e$  and the incoming radiance (see fig. 1 for notations):

$$\Phi_{pix} = \int_{A_{pix}} dA_{pix} W_e(\mathbf{V}_0 \rightarrow \mathbf{D}_1) L(\mathbf{V}_0 \leftarrow \mathbf{D}_1)$$

$L$  is unknown and can be expanded using the rendering equation:

$$\Phi_{pix} = \int_{A_{pix}} dA_{pix} \int_{\Omega} d\omega W_e(\mathbf{V}_0 \rightarrow \mathbf{D}_1) f_r(\mathbf{D}_1, \mathbf{V}_1, \mathbf{D}_2) \cos(\theta_1) L(\mathbf{V}_1 \leftarrow \mathbf{D}_2)$$

with  $f_r$  the BRDF and  $\cos(\theta_1) = \mathbf{N}_1 \cdot \mathbf{D}_2$ .

Each new reflection or refraction introduces an  $f_r$  and cosine factor. To simplify notation we define the potential function  $W_i$  of a path of length  $i$  as the product of  $W_e$  and all subsequent  $f_r$  and cosine factors. Now the *path evaluation*  $F$  (the integrand) of a path is simply:

$$F = W_i(\mathbf{V}_i \rightarrow \mathbf{D}_{i+1}) \cdot L(\mathbf{V}_i \leftarrow \mathbf{D}_{i+1})$$

And  $W_{i+1}$  is:

$$W_{i+1} = W_i(\mathbf{V}_i \rightarrow \mathbf{D}_{i+1}) f_r(\mathbf{D}_{i+1}, \mathbf{V}_{i+1}, \mathbf{D}_{i+2}) \cos(\theta_{i+1})$$

Actual contributions to the image are made when a light source is hit or by direct sampling of the light sources.

## 4.2 Relative partial derivatives

The partial derivatives of a path evaluation are:

$$\frac{\partial F}{\partial x_j} = \frac{\partial W_i}{\partial x_j} L + W_i \frac{\partial L}{\partial x_j}$$

We compute *relative partial derivatives* by dividing this expression by  $F$ :

$$\frac{\partial F}{\partial x_j} / F = \frac{\partial W_i}{\partial x_j} / W_i + \frac{\partial L}{\partial x_j} / L$$

This expression gives the relative rate of change of  $F$  in terms of  $x_j$ .

Because common factors cancel out it also provides a convenient way for tracking partial derivatives of  $W_i$  when extending a path:

$$\frac{\partial W_{i+1}}{\partial x_j} / W_{i+1} = \frac{\partial W_i}{\partial x_j} / W_i + \frac{\partial f_r}{\partial x_j} / f_r + \frac{\partial \cos(\theta)}{\partial x_j} / \cos(\theta)$$

Just the relative partial derivatives of the individual factors must be added to the known  $\frac{\partial W_i}{\partial x_j} / W_i$ . To start a path  $\frac{\partial W_e}{\partial x_j}$  must be computed, which is 0 for constant  $W_e$ .

The computation of these derivatives is straightforward, as the BRDF evaluation and the cosine factor can be expressed in terms of  $\mathbf{V}_i$ ,  $\mathbf{D}_i$  and  $\mathbf{N}_i$  for which derivatives have already been computed for the differential vectors. Consider for example the cosine factor:

$$\frac{\partial \cos\theta_i}{\partial x_j} = \frac{\partial(\mathbf{N}_i \mathbf{D}_{i+1})}{\partial x_j} = \frac{\partial \mathbf{N}_i}{\partial x_j} \mathbf{D}_{i+1} + \mathbf{N}_i \frac{\partial \mathbf{D}_{i+1}}{\partial x_j}$$

Both derivatives of  $\mathbf{N}_i$  and  $\mathbf{D}_{i+1}$  were computed before.

When an actual radiance contribution from a light source is computed,  $\frac{\partial L}{\partial x_j}/L$  must be added to  $\frac{\partial W_i}{\partial x_j}/W_i$  to get  $\frac{\partial F}{\partial x_j}/F$ . Computation depends on the light source sampling. Currently we consider the change of the light source contribution to be constant, so that its derivative is zero. For moderately distant light sources, this approximation is well acceptable.

### 4.3 Choosing $\Delta x_j$

When a perturbation  $\Delta x_j$  is applied to a path  $X$ , the relative change of the contribution  $F(X)$  can be approximated as:

$$\Delta FR_j \approx \left[ \frac{\partial F}{\partial x_j}(X) / F(X) \right] \cdot \Delta x_j \quad (1)$$

For example a gradient  $\Delta FR_j$  of 300% means the contribution of the perturbed path is approximately three times higher (or lower) than  $F(X)$ .

Now consider a vertex  $\mathbf{V}$  in a path. The differential vectors  $\Delta \mathbf{V}_j = \frac{\partial \mathbf{V}}{\partial x_j} \Delta x_j$  define a maximal allowed perturbation of  $\mathbf{V}$  when changing  $x_j$ . Since the same  $\Delta x_j$  is used as in equation 1, this maximal perturbation corresponds to the relative change  $\Delta FR_j$  of the path contribution.

In our applications we consider the contribution  $F$  to be constant over the differential vectors. Of course  $F$  does change and  $\Delta FR_j$  indicates how much. By constraining the relative change  $\Delta FR_j$  to be smaller than a maximum threshold  $\Delta FR_{max}$ , deltas can be computed from equation 1:

$$\Delta x_j = \frac{\Delta FR_{max}}{\frac{\partial F}{\partial x_j} / F}$$

Using these deltas for the vertex differential vectors results in smaller vectors (and resulting footprint) when the gradient is large. The threshold controls the allowable error.

A very small gradient, however, can lead to arbitrary large footprints. We use the local or global delta heuristic (Section 3.4) as an upper limit for the deltas computed using the gradient. This gives good results, but the choice of  $\Delta FR_{max}$  (that controls over the error) is not obvious.

### 4.4 Discussion

Other functions of a path can also be candidates for derivative computation. The *score function* of a path  $\mathcal{P}$  in Monte Carlo integration is  $F(\mathcal{P})/p(\mathcal{P})$ , where  $p$  is the pdf used for generating the path. It is in fact the score function that is evaluated and averaged when computing pixel fluxes with Monte Carlo ray tracing. This could be an interesting choice for tracking derivatives.

Usually pdf's for direction sampling are chosen proportionally to the BRDF or the cosine. These factors cancel out and the score function only contains factors not used in path sampling.

We have chosen to compute derivatives of  $F$  itself so that these path sampling factors are explicitly included. It can be shown that (for separable pdf's) derivatives of these factors are related to the second order derivative of footprint. However, a full analysis is beyond the scope of this paper.

Both choices do include factors not present in the path sampling, so that bad path sampling is countered with large  $\Delta F_j$ .

For classical ray tracing  $W_{i+1} = W_i \cdot f_s$  with  $f_s$  the *constant* specular reflection or refraction coefficient. Derivatives of  $f_s$  are zero so the path gradient does not yield any extra information for the method presented by Igehy.

## 5 Applications

The differentials, gradients and footprint estimation were implemented in RenderPark. Derivatives are supported for diffuse reflection, phong reflection and refraction, area light sources and pixel sampling.

### 5.1 Texture Filtering

In a first application we extend a ray tracer with glossy reflections and refractions. The computed footprint is used for filtering textures locally on surfaces to reduce noise.

Igehy presented the same application for classical ray tracing. Comparison with other filtering approaches can be found in [8]. We use trilinear interpolated mipmapped textures and anisotropic filtering. The smallest representative vector in the footprint determines the mipmap level, and several samples are averaged along the larger axis. We used a box filter, but weighted filtering might give even better results.

The scene contains several textured surfaces; the two 'playing pieces' consist of a squashed sphere on a diffuse base. One is reflective, the other refractive. Both use a glossy phong lobe for scattering. For all but the reference image we used one sample per pixel and 4 stratified samples for each scattering.

**Figure 5** shows standard distributed ray tracing. The reflections and refractions show a lot of noise, as 4 is a low sampling rate.

**Figure 6** shows filtering with the *local delta* heuristic (3.4). Using only 4 scattering samples, the estimated distance to a neighboring ray is quite large. The filtering reduces the noise of the glossy scatterings, but it sometimes over-blurs, especially for the multiple refractions in the glass object.

For **figure 7** the path gradient was used for estimating deltas. Over-blurring is reduced because the glossy phong lobes give rise to high gradients. The noise still remains a lot lower as in image 5. The threshold value  $FR_{max}$ , that restricts relative changes over the differential vectors, was chosen to be 70%. This choice was not very critical, anything from 50% to 100% worked well.

A reference image using 81 samples per pixel is given in figure 8.

Note that only textures are filtered; 'edge' noise due to scattered rays hitting different objects is not reduced (e.g. the table edge seen in the metal piece). An adaptive sampling scheme could direct more samples towards edges, but not towards varying textures.

The overhead introduced by differential computation was relatively small ( $< 10\%$  for this example). Computing BRDF differentials is about as expensive as sampling and evaluating the BRDF itself. As a path grows longer, however, more differentials have to be computed, and the overhead is not negligible. Reducing the differential vectors to a few representatives before prolonging the path, keeping the amount of tracked differentials constant, could be an interesting line of research.

### 5.2 Particle tracing

Particle tracing constructs paths starting from the light sources. This Monte Carlo simulation of light transport is used in many global illumination algorithms, e.g. radiosity [6, 12], density estima-

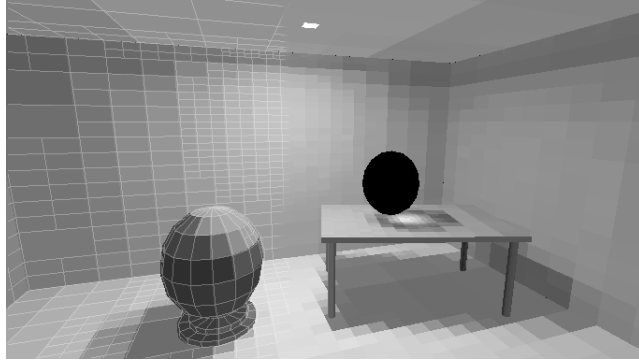


Figure 4: Hierarchical particle tracing radiosity (400k paths) using the area defined by the path differentials as a refinement oracle.

tion [14] and photon map construction [9].

We present a hierarchical radiosity application to demonstrate the usefulness of path differentials for particle tracing, but it can be used as well for the other algorithms.

A hierarchical version of particle radiosity was presented in [6] and [16]. Both methods accumulate the hits (the radiance) on elements and subdivide if the variation over the element is too large. In [16] hits are stored simultaneously on the two lower levels of the hierarchy. Some remarks that are valid for both algorithms:

- While the radiosity solution is hierarchical, the light transport itself is not. All particles contribute to the most detailed or the two most detailed levels in the hierarchy.
- When subdivision occurs, the radiance information of the discarded level is thrown away, because it is equal (and inadequate) for its children.

We use path differentials to estimate a footprint for each individual particle. The area defined by the two representative vectors ( $\Delta A \times \Delta B$ ) gives an idea about the density of similar paths in the neighborhood.

Our refinement oracle looks for the largest element that is just smaller than the footprint, subdividing as necessary. The contribution is made to this element in the hierarchy. The level in the hierarchy can be different for each particle so the light transport is hierarchical. No previous paths are thrown away.

While this refinement oracle based on a single particle has interesting advantages, we do not claim it to be the best oracle in existence. The main purpose is to show the behavior and usefulness of the path differentials for particle tracing.

The implementation of our radiosity algorithm uses clustering and constant basis functions. We used the gradient to determine deltas and the global deltas as an upper bound.

Figure 4 shows the radiosity solution for a simple room with a table, a diffuse and a glossy refractive ball (phong exponent 40). Note that the refractive ball shows up black as it has no diffuse component.

Several interesting things to note in the image:

- A correct caustic is visible on the table. The fine subdivision is due to a large gradient (and small delta). Merely using the global delta heuristic barely shows the caustic. The overall subdivision due to direct light and diffuse interreflection is less influenced by the gradient. The gradient for these paths is small and the global delta upper limit is used (except in corners).

- Only 400,000 paths were traced, resulting in a relatively noiseless solution. This is because global deltas depend on the number of samples. Only where the gradient is used (e.g. the caustic) more noise can be seen.
- The ceiling is lit by diffuse interreflection. A diffuse reflection (using  $\cos\theta$  sampling) results in relatively large derivatives for the sampled direction. Transfer to the ceiling results in a large footprint and thus a coarser subdivision. Near the bright spot on the back wall though a finer subdivision can be seen and some color bleeding.

## 6 Conclusions

In this work ray differentials are generalized to arbitrarily sampled paths, including general BRDF's. Each sampling introduces new partial derivatives that are all combined in a footprint estimate.

We also introduced the computation of a path gradient, the change of the path contribution over the differential vectors. It is used to restrict the footprint in case of high gradients.

We successfully used the footprint estimation for texture filtering in ray tracing with glossy materials and in a novel refinement oracle in hierarchical particle tracing radiosity.

Since our framework allows arbitrary sampling many other methods can benefit from path differentials, in particular Monte Carlo global illumination methods:

- In photon maps the footprint of eye paths can determine the area over which photons must be considered for illumination reconstruction. If too few photons are found the path can be extended.
- Importance calculations can also be performed. The footprint of an eye ray indicates the density of similar paths. A small footprint indicates a high importance, for example by magnification through glass.

Another interesting line of research would be to introduce visibility into the framework. As the footprint is still based on a point sample, nearby visibility changes are ignored. It would be interesting to adjust the footprint by selective visibility tests.

We explored only one possible definition of a path footprint: the area made up by a set of perturbed vertices. Other interesting approaches are possible such as using a convolution of kernel filters defined over the differential vectors.

## 7 Acknowledgments

Many thanks to Vincent Masselus for modeling the scenes and to all the people that made useful suggestions. This research was partly supported by FWO Grant #G.0263.97.

## References

- [1] John Amanatides. Ray tracing with cones. *Computer Graphics*, 18(3):129–135, July 1984.
- [2] Philippe Bekaert. *Hierarchical and Stochastic Algorithms for Radiosity*. PhD thesis, K.U.Leuven, Belgium, December 1999.
- [3] Philippe Bekaert. *Personal Communication*. 2001.

- [4] M. Chen and J. Arvo. Perturbation methods for interactive specular reflections. In Hans Hagen, editor, *IEEE Transactions on Visualization and Computer Graphics*, volume 6 (3), pages 253–264. IEEE Computer Society, 2000.
- [5] Steven Collins. Adaptive Splatting for Specular to Diffuse Light Transport. In *Fifth Eurographics Workshop on Rendering*, pages 119–135, Darmstadt, Germany, June 1994.
- [6] Paul S. Heckbert. Adaptive radiosity textures for bidirectional ray tracing. *Computer Graphics*, 24(4):145–154, August 1990.
- [7] Paul S. Heckbert and Pat Hanrahan. Beam tracing polygonal objects. *Computer Graphics*, 18(3):119–127, July 1984.
- [8] Homan Igehy. Tracing ray differentials. *Computer Graphics*, 33(Annual Conference Series):179–186, 1999.
- [9] Henrik Wann Jensen. Global illumination using photon maps. In Xavier Pueyo and Peter Schröder, editors, *Eurographics Rendering Workshop 1996*, pages 21–30, New York City, NY, June 1996. Eurographics, Springer Wien. ISBN 3-211-82883-4.
- [10] M. Kalos and P. Whitlock. *Monte Carlo Methods, Volume 1: Basics*. J. Wiley, New York, 1986.
- [11] Eric P. Lafortune and Yves D. Willems. Using the Modified Phong BRDF for Physically Based Rendering. Technical Report CW197, Department of Computer Science, Katholieke Universiteit Leuven, Leuven, Belgium, November 1994.
- [12] S. N. Pattanaik and S. P. Mudur. Computation of global illumination by monte carlo simulation of the particle model of light. *Third Eurographics Workshop on Rendering*, pages 71–83, May 1992.
- [13] Mikio Shinya, Tokiichiro Takahashi, and Seiichiro Naito. Principles and applications of pencil tracing. *Computer Graphics*, 21(4):45–54, July 1987.
- [14] Peter Shirley, Bretton Wade, Philip M. Hubbard, David Zareski, Bruce Walter, and Donald P. Greenberg. Global Illumination via Density Estimation. In P. M. Hanrahan and W. Purgathofer, editors, *Rendering Techniques '95 (Proceedings of the Sixth Eurographics Workshop on Rendering)*, pages 219–230, New York, NY, 1995. Springer-Verlag.
- [15] Frank Suykens and Yves Willems. Path differentials and applications. In K. Myskowski and S. Gortler, editors, *Rendering Techniques 2001 (Proceedings of the 12th Eurographics Workshop on Rendering)*, pages 254–265, New York, NY, 2001. Springer-Verlag.
- [16] Robert F. Tobler, Alexander Wilkie, Martin Fedà, and Werner Purgathofer. A hierarchical subdivision algorithm for stochastic radiosity methods. In Julie Dorsey and Philipp Slusallek, editors, *Eurographics Rendering Workshop 1997*, pages 193–204, June 1997.
- [17] Greg Turk. Generating random points in triangles. In Andrew Glassner, editor, *Graphics Gems I*, pages 24–28. Academic Press Professional, Boston, MA, 1990.

## A Derivative computation details

This appendix shows how to compute derivatives for several sampling events:

- Pixel sampling
- Ray transfer
- Direction sampling (Phong lobe sampling)
- Light source sampling

The following notation will be used throughout this appendix (see also figure 2):

- $\mathbf{D}'$ ,  $\mathbf{V}'$ : Newly sampled direction and vertex.
- $\mathbf{D}$ ,  $\mathbf{V}$ : Previously sampled direction or vertex.
- $x, y$ : Unit random variables used in the current sampling event (e.g. random numbers used in BRDF sampling).
- $x_j$ : Unit random variables used for previous sampling events in the path.
- $\mathbf{N}$ : Shading normal.
- $\mathbf{N}_g$ : Geometric normal.
- $\mathbf{R}$ : Perfectly specular reflection vector (given an incoming direction  $\mathbf{D}$  in a path vertex  $\mathbf{V}$ ).

### A.1 Pixel Sampling

Pixel sampling constructs a direction  $D'$  by sampling a uniform point on a certain pixel. The derivative computation is similar to pixel sampling in [8] except for a scale factor because we compute the derivatives with respect to unit (random) variables.

Given 2D image coordinates  $(u, v)$  and a camera coordinate system (**View**, **Right**, **Up**), a non normalized viewing direction is given by:

$$d(x, y) = \mathbf{View} + u\mathbf{Right} + v\mathbf{Up}$$

Uniform sampling of a pixel using random variables  $x, y$  gives image coordinates:

$$\begin{aligned} u &= x \text{pix}_w + \text{pix}_l \\ v &= y \text{pix}_h + \text{pix}_t \end{aligned}$$

with  $\text{pix}_{w,h}$  the width and height of the pixel, and  $\text{pix}_{l,t}$  the left and top coordinate of the pixel under consideration.

The normalized sampled direction is:

$$\mathbf{D}' = \frac{d}{(d \cdot d)^{1/2}}$$

The derivative for  $u$  is:

$$\frac{\partial \mathbf{D}'}{\partial x} = \frac{\partial \mathbf{D}'}{\partial u} \frac{\partial u}{\partial x} = \frac{\partial \mathbf{D}'}{\partial u} \text{pix}_w$$

This differs from [8] only by a scale factor  $\text{pix}_w$ . A similar expression can be computed for  $\frac{\partial \mathbf{D}'}{\partial y}$ .

## A.2 Transfer

*Transfer* computes a new path vertex  $\mathbf{V}'$  given a vertex  $\mathbf{V}$  and an outgoing direction  $\mathbf{D}$ . Transfer does not involve new sampling, and the result from [8] can be used. Of course derivatives have to be computed for all random variables used in the path up till the vertex  $\mathbf{V}'$ .

$$\mathbf{V}' = \mathbf{V} + t\mathbf{D}$$

The derivatives are:

$$\frac{\partial \mathbf{V}'}{\partial x_j} = \left( \frac{\partial \mathbf{V}}{\partial x_j} + t \frac{\partial \mathbf{D}}{\partial x_j} \right) + \frac{\partial t}{\partial x_j} \mathbf{D}$$

with

$$\frac{\partial t}{\partial x_j} = \frac{\left( \frac{\partial \mathbf{V}}{\partial x_j} + t \frac{\partial \mathbf{D}}{\partial x_j} \right) \cdot \mathbf{N}_g}{\mathbf{D} \cdot \mathbf{N}_g}$$

Note that  $\frac{\partial \mathbf{V}'}{\partial x_j}$  is always perpendicular to the geometric normal  $\mathbf{N}_g$ . This can be shown by computing the dot product using the formulas above.

## A.3 Direction sampling

Direction sampling involves choosing a new direction  $\mathbf{D}'$  given an incoming direction  $\mathbf{D}$  in a vertex  $\mathbf{V}$ . In our rendering framework, a transformation to a sampling coordinate frame is applied first. A new direction  $\mathbf{D}'_s$  is generated in the sampling frame and transformed back into world coordinates giving the direction  $\mathbf{D}'$ .

The derivation is given for Phong lobe sampling, but it is similar for other direction sampling methods (e.g. uniform sampling, cosine sampling).

Section A.3.1 shows how the transformation is handled when computing derivatives. Phong lobe sampling in the local frame is detailed in section A.3.2.

### A.3.1 Coordinate transformation

Figure 2 shows such a phong lobe (in 2D). Given the incoming direction  $\mathbf{D}$ , a new direction  $\mathbf{D}'$  is sampled proportional to the  $\cos^S(\alpha)$  lobe, where  $S$  is the sharpness of the lobe and  $\alpha$  is the angle between the perfectly reflected direction  $\mathbf{R}$  and  $\mathbf{D}'$ .

First a coordinate transformation  $T_s$  (a rotation) is performed making  $\mathbf{R}$  the new Z-axis. The sampling itself is done in this shading frame, and the resulting vector  $\mathbf{D}_s$  is transformed back using  $T_s^{-1}$  ( $= T_s^\top$ ).

Computing the derivatives follows the same steps. As said,  $x_j$  will be used to denote variables introduced by previous sampling events and  $x, y$  are the new variables used in the actual sampling of the lobe.

The reflected direction  $\mathbf{R}$  is given by:

$$\mathbf{R} = \mathbf{D} - 2(\mathbf{D}\mathbf{N})\mathbf{N}$$

To construct a rotation matrix  $T_s$  two other axes  $X_s, Y_s$  of the sampling frame need to be constructed, yielding:

$$T_s = [X_s Y_s \mathbf{R}]^\top \text{ and } T_s \mathbf{D}' = \mathbf{D}'_s \quad (2)$$

**Derivatives for  $x_j$**  Since  $x_j$  is introduced by previous sampling, derivatives exist for  $\mathbf{D}$ ,  $\mathbf{V}$  and also  $\mathbf{N}$ . To compute the derivative of  $\mathbf{D}'$  we derive equation 2:

$$\frac{\partial T_s}{\partial x_j} \mathbf{D}' + T_s \frac{\partial \mathbf{D}'}{\partial x_j} = \frac{\partial \mathbf{D}'_s}{\partial x_j} \quad (3)$$

$$\frac{\partial \mathbf{D}'}{\partial x_j} = T_s^{-1} \left( \frac{\partial \mathbf{D}'_s}{\partial x_j} - \frac{\partial T_s}{\partial x_j} \mathbf{D}' \right) \quad (4)$$

where:

$$\frac{\partial T_s}{\partial x_j} = \left[ \frac{\partial X_s}{\partial x_j} \frac{\partial Y_s}{\partial x_j} \frac{\partial \mathbf{R}}{\partial x_j} \right]^T$$

Computation of  $\frac{\partial \mathbf{R}}{\partial x_j}$  is given in [8]. Derivatives of  $X_s$  and  $Y_s$  depend on how they were chosen. This will not be detailed here.

For phong lobe sampling  $\mathbf{D}'_s$  only depends on new variables  $x, y$  so equation 4 reduces to:

$$\frac{\partial \mathbf{D}'}{\partial x_j} = T_s^{-1} \left( - \frac{\partial T_s}{\partial x_j} \mathbf{D}' \right)$$

**Derivatives for  $x, y$**  Since  $x, y$  are newly introduced, we know that the shading frame does not depend on them ( $\frac{\partial T_s}{\partial x, y} = 0$ ), so that:

$$\frac{\partial \mathbf{D}'}{\partial x, y} = T_s^{-1} \frac{\partial \mathbf{D}'_s}{\partial x, y}$$

Now we only need to compute the derivatives of  $\mathbf{D}'_s$  in the (convenient) sampling frame.

### A.3.2 Sampling $\mathbf{D}'_s$ using $x, y$

After transformation, we need to sample a direction that has a  $\cos^S(\theta)$  distribution given  $x, y \in [0, 1]$  and  $\theta$  as the angle with the Z-axis (In fact  $\theta = \alpha$ , but  $\theta$  is used to indicate that we work in a local shading frame).

This is a well known importance sampling procedure (see e.g [11]). Angles  $\theta$  and  $\phi$  are determined as follows:

$$\begin{aligned} \cos(\theta) &= x^{1/(S+1)} \\ \sin(\theta) &= \sqrt{1 - \cos^2(\theta)} \\ \phi &= 2\pi y \end{aligned}$$

The direction  $\mathbf{D}'_s$  is now:

$$\mathbf{D}'_s = (\cos(\phi)\sin(\theta), \sin(\phi)\sin(\theta), \cos(\theta))$$

To compute  $\mathbf{D}'_s$  we just need to derive these equations for  $x$  and  $y$  (only non zero derivatives are shown):

$$\begin{aligned} \frac{\partial \cos \theta}{\partial x} &= \frac{1}{S+1} x^{-s/(s+1)} = \frac{1}{(S+1)\cos^S \theta} \\ \frac{\partial \sin \theta}{\partial x} &= -\cos \theta * \frac{\partial \cos \theta}{\partial x} / \sin \theta \\ \frac{\partial \phi}{\partial y} &= 2\pi \end{aligned} \quad (5)$$

The derivatives for  $\mathbf{D}'_s$  are easily computed from this. Note again that only elementary calculus is needed to turn a sampling procedure into derivative computations.

Two remarks:

- Near the pole of the lobe,  $\frac{\partial \sin\theta}{\partial x}$  becomes large as  $\sin\theta$  goes to 0. Limiting  $\sin\theta$  to a certain small  $\epsilon$  is sufficient to prevent this problem.
- Note that  $c = \cos\theta$  is chosen with a pdf  $p(c) = (S+1)c^S$ . With  $x$  uniform over  $[0, 1]$  and  $P$  the cumulative distribution ( $\int p$ ), importance sampling tells us that  $c = P^{-1}(x)$  is distributed according to the desired pdf.  
The derivative  $\frac{\partial c}{\partial x}$  is now simply  $1/p(c)$  as can be verified in equation 5. For non-separable two dimensional pdf's the equivalent relationship is more complex, but can still be useful.

## A.4 Light sampling

This section details derivative computations for light source sampling.

Sampling a path vertex on a light source involves two steps, choosing a light source  $l$  (A.4.2) and sampling a point on this light source (A.4.1).

### A.4.1 Choosing a point on a light source

We will demonstrate the computation of derivatives for a regular quadrilateral and a triangle. Other geometry can be handled similarly by differentiating their sampling procedures.

**Quadrilateral** Given a regular quadrilateral light source (i.e. a parallelogram) with vertices  $\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}$  and given random numbers  $x, y$ , a uniformly sampled vertex  $\mathbf{V}'$  and its derivatives are given by:

$$\begin{aligned}\mathbf{V}' &= \mathbf{A} + x(\mathbf{B} - \mathbf{A}) + y(\mathbf{D} - \mathbf{A}) \\ \frac{\partial \mathbf{V}'}{\partial x} &= \mathbf{B} - \mathbf{A} \\ \frac{\partial \mathbf{V}'}{\partial y} &= \mathbf{D} - \mathbf{A}\end{aligned}$$

For irregular quadrilaterals, the sampling procedure and its derivatives are more complex. First a mapping from uniform  $x, y$  to bilinear coordinates must be performed. The point is then sampled using the standard bilinear mapping:  $\mathbf{V}' = x(\mathbf{B} - \mathbf{A}) + y(\mathbf{D} - \mathbf{A}) - xy(\mathbf{B} - \mathbf{C} + \mathbf{D} - \mathbf{A})$ . See [2] for more information.

**Triangle** A simple procedure for sampling a uniform point in a triangle  $\mathbf{A}, \mathbf{B}, \mathbf{C}$  is given by [17]:

$$\begin{aligned}\text{if } (x + y > 1.) \text{ then } x &= 1 - x; y = 1 - y \\ \mathbf{V}' &= \mathbf{A} + x(\mathbf{B} - \mathbf{A}) + y(\mathbf{C} - \mathbf{A})\end{aligned}$$

In fact a point is sampled in a parallelogram, and one half ( $x + y > 1$ ) is mirrored back into the triangle. This doubles the density of points compared to the density in the parallelogram. Multiplying the derivatives by a factor  $\sqrt{2}$  corrects for this higher density. The derivatives are:

$$\begin{aligned}\frac{\partial \mathbf{V}'}{\partial x} &= (\mathbf{B} - \mathbf{A})/\sqrt{2} \\ \frac{\partial \mathbf{V}'}{\partial y} &= (\mathbf{C} - \mathbf{A})/\sqrt{2}\end{aligned}$$

Suppose for instance that  $N$  uniform samples are generated in the triangle. The expected density  $\rho$  of points in a point  $\mathbf{V}(x,y)$  in the triangle is<sup>2</sup>:

$$\rho(\mathbf{V}) = \frac{N}{\frac{\partial \mathbf{V}'}{\partial x} \times \frac{\partial \mathbf{V}'}{\partial x}} = \frac{N}{\mathbf{AB} \times \mathbf{AC} / 2} = \frac{N}{\text{Area}(\mathbf{A}, \mathbf{B}, \mathbf{C})}$$

#### A.4.2 Choosing the light source

Suppose there are a number of light sources that emit in total a flux  $\Phi_{\text{tot}}$ . Usually light sources are chosen according to their emitted power, resulting in a probability  $P_l$  for choosing  $l$ :

$$P_l = \frac{\Phi_l}{\Phi_{\text{tot}}}$$

Note that this is a discrete sampling event, and no derivatives can be computed. We can however account for this probability by scaling the derivatives of the sampled points themselves with a factor  $\sqrt{P_l}$ . This factor accounts for the lower density of points on a single light source  $l$ .

Note that other discrete sampling events can also be handled by scaling the derivatives (cfr. Russian roulette).

---

<sup>2</sup>Note that other procedures for sampling uniform points in a triangle are possible (cfr. [17]). These result in other partial derivatives, but the density (i.e. the vector product of the derivatives) stays the same for all procedures.

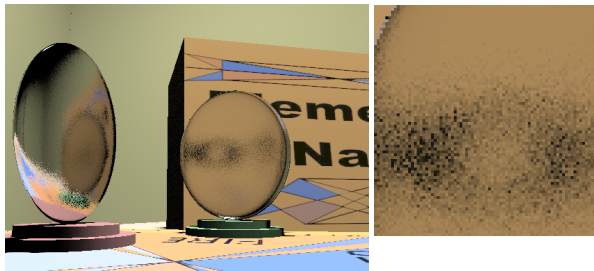


Figure 5: Image of a scene with glossy materials without the use of texture filtering. Glossy reflections and refractions are noisy.

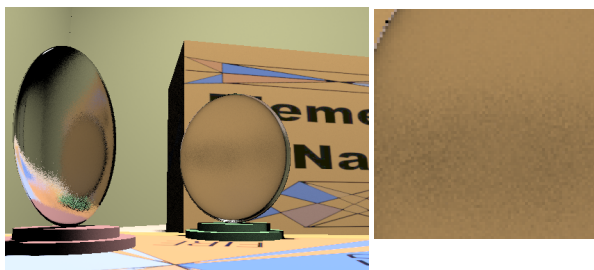


Figure 6: For this image texture filtering is used, based on the footprint of the path. In some regions the footprint is over-estimated, causing too much blurring (especially the glossy transparent squashed sphere, as shown in the magnification).

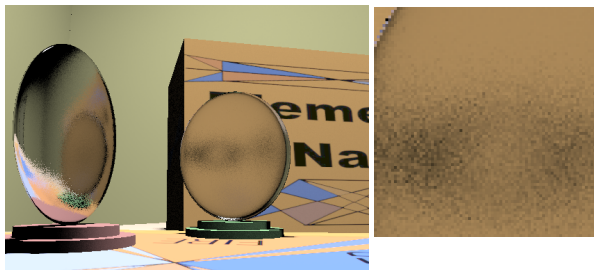


Figure 7: For this image the path gradient was used to reduce over-estimated footprints and the excessive blurring is effectively reduced.

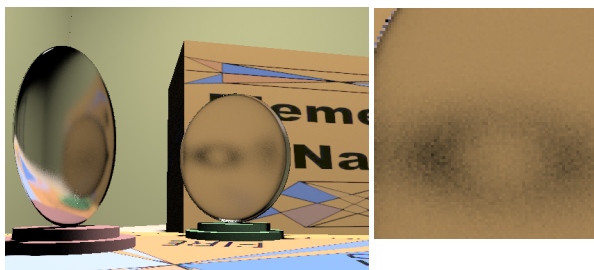


Figure 8: Reference image using many more samples.