Implementing Temporal Photon Differentials

ARVID HANSEN DÍAZ
UNIVERSITY OF COPENHAGEN

A thesis submitted for the degree of
Bachelor of Science

February 11, 2011
## Contents

1 Introduction 2

2 Photon Mapping Basics 2
   2.1 Photon Tracing 3
      2.1.1 Reflections 5
      2.1.2 Storing Photons 6
   2.2 Rendering 6
      2.2.1 Radiance Estimate 6
      2.2.2 Visualisation 8

3 Temporal Photon Differentials 9
   3.1 Transfer 10
   3.2 Reflection and Refraction 12
   3.3 Shading 12
   3.4 Temporal Radiance Estimate 13

4 Implementation 15

5 Experiments 16

6 Conclusions 22
1 Introduction

In Computer Graphics we are looking all the time for efficient ways to recreate the world around us in a way that is indistinguishable from reality. There are a lot of available techniques, proven already in practice, e.g. in computer generated cinema pictures. But for the huge requirements of calculating these photo realistic images we still need a lot of computers and time. Apart from the fast advancement in computer hardware, it will still take some time until we can produce photo realistic images in real time, e.g. at home on a play station for games or something similar.

One of the complex tasks in computer graphics is indirect lightning. Not everything in our environment is illuminated directly by the sun or another light source, e.g. a lamp. Mostly the incoming light is reflected from other objects in the world, and they possibly receive the light from other reflecting objects as well. This is a recursive calculation problem, whose calculation costs grow with each increasing recursion step. Additionally, there can be different materials in the world, which reflect the light in many varying ways. In combination with direct lightning we call this global illumination. One algorithm which can handle global illumination, and on which my thesis is based, is called photon mapping. It is also perfectly suited for rendering caustics.

We don’t always need a high accuracy for real time applications, because the human eye can’t differentiate, if it sees for example in moving pictures a perfect reflection or just a fake reflection. Therefore, to get an immersive feeling, we can approximate the physical phenomena so far that we will believe it to be perfect.

If the function used to calculate the physical phenomena is infinite differentiable, we can approximate this function with a first order Taylor series. Therefore, it is possible to approximate the functions used in photon mapping and to extend them in time, which is called temporal photon differentials.

In thesis I will show the basic theory behind these techniques and demonstrate at the end some results on the image quality and the calculation speed of temporal photon differentials, compared to conventional photon mapping.

2 Photon Mapping Basics

In this section I want to describe the Photon-Mapping concept based on [4]. This technique is a quite straightforward way to simulate global illumination, especially indirect light and caustics.

The idea is to sample the world from two point of views, the light source and the observer. From there the photon map algorithm is build up in
two steps. In the first step, we shoot photons from the light source into the world and follow them using photon tracing. Hitting a surface, the photons can either be reflected, transmitted or absorbed, depending on the properties of the material. The information about the hitting photons is stored in a separate data structure, the photon map. A photon tracer works in the same way as a ray tracer. Instead of emitting rays from the camera into the world and gathering radiance, we will send photons which spread flux from the light source. In the second step, the image is rendered, using the information stored in the photon map, to estimate the reflected radiance on surfaces. Figure 1a shows an example caustic photon map, where each point represents a photon, and Figure 1b shows the final rendered caustic photon map. You can see how the photons in the map correspond to the caustics in the rendered image.

2.1 Photon Tracing

Photons can be emitted from any type of light sources used in computer graphics, where each photon carries a portion of the overall emitted power from the light source. If a photon hits a diffuse surface, it loses a part of its power, depending on the reflectivity of the material, and is reflected with reduced power. To reduce computational resources we can also limit the amount of photons followed, after hitting a surface with full power, using a stochastic technique, called Russian Roulette. Given a probability, $q$, that a photon is reflected and a new photon $P_n$ is traced, the random variable
$X$ denotes the subsequent weight of the photon, which is

$$P_n = \begin{cases} 
\frac{P}{q}, & \text{if } X > q \\
0, & \text{otherwise.}
\end{cases} \quad (1)$$

We can calculate the expected value of $X$ using

$$E(X) = (1 - q) \times 0 + q \times \frac{P}{q} = P \quad (2)$$

which is the original unweighed photon. This means that we can ignore most of the unimportant photons, and concentrate on the significant ones and still get a correct result. In the following, I will give an exemplary pseudo code in Figure 2, which decides if a photon hitting a material with diffuse reflectivity $d$ and specular reflectivity $s$ is either reflected specular or diffuse or if it is absorbed. The power of the reflected photon does not change.

```plaintext
pDiffuse = d  //probability of diffuse reflection
pSpecular = s  //probability of specular reflection
X = random()  //uniformly distributed random number
if ( X < pDiffuse )
    diffuse reflection
if ( X >= pDiffuse && X < pDiffuse + pSepcular )
    specular reflection
else
    absorption
```

Figure 2: pseudocode
2.1.1 Reflections

**Diffuse Reflection:** A photon is only stored when it hits diffuse surfaces. It is more accurate to render specular reflections by tracing a ray in the direction from the camera to the mirror. Diffuse surfaces have in all directions a constant reflected radiance. If it is decided through Russian Roulette to reflect a photon diffusely, a new photon is generated with the same photon power of the incoming photon. The direction of the new photon is random. Using spherical coordinates \((\theta, \phi)\) for the direction, where \(\theta\) is the surface normal and \(\phi\) indicates the rotation around the normal, and two uniformly distributed random numbers \(X_1\) and \(X_2\), we can calculate the cosine weighed reflected direction \(\vec{w}_d\) as:

\[
\vec{w}_d = (\theta, \phi) = (\arccos(\sqrt{X_1}), 2\pi X_2).
\]

(3)

**Specular Reflection:** In the case of specular reflections, photons are not stored in the photon map. The incoming photons are reflected in the mirror direction. Therefore, the angle of the incoming photon to the surface normal \(\vec{n}\) is the same as the angle between the reflected photon to \(\vec{n}\). We can calculate the reflected direction \(\vec{w}_s\) of the incoming photon with the direction \(\vec{w}_i\) by:

\[
\vec{w}_s = \vec{w}_i - 2(\vec{w}_i \cdot \vec{n})\vec{n}.
\]

(4)

**Refraction:** At the transition from one medium with an index of refraction \(\eta_1\) to another medium with an index of refraction \(\eta_2\) the light is refracted. There is a relation between the angel of incidence \(\alpha\) and the angle of the refracted light \(\beta\), it is known as Snell’s Law:

\[
\frac{\sin \alpha}{\sin \beta} = \frac{\eta_2}{\eta_1}.
\]

(5)

From this follows for the refracted light direction vector \(\vec{w}_r\) (see [10] for a detailed derivation):

\[
\vec{w}_r = \frac{\eta_1 (\vec{w}_i - \vec{n}(\vec{w}_i \cdot \vec{n}))}{\eta_2} - \vec{n} \sqrt{1 - \frac{\eta_1^2 (1 - (\vec{w}_i \cdot \vec{n})^2)}{\eta_2^2}}.
\]

(6)

**Fresnel:** When light hits the boundary surface between two mediums with indices of refraction \(\eta_1\) and \(\eta_2\), we can calculate the transmission and reflection with the Fresnel equations depending on the polarization of the light, the angle of incidence \(\alpha\) and the angle of refraction \(\beta\). For the parallel component to the plane of incidence we get the reflectance:

\[
\rho_\parallel = \frac{\eta_2 \cos \alpha - \eta_1 \cos \beta}{\eta_2 \cos \alpha + \eta_1 \cos \beta}.
\]
and for the orthogonal component:

\[ \rho_\perp = \frac{\eta_1 \cos \alpha - \eta_2 \cos \beta}{\eta_1 \cos \alpha + \eta_2 \cos \beta} \]

If we assume that the light is unpolarized, we can calculate the reflectance with:

\[ F_r(\alpha) = \frac{1}{2}(\rho_\parallel^2 + \rho_\perp^2) \]

As we are using Russian Roulette, \( F_r \) is taken as a limiting value to decide if a photon is reflected or refracted:

\[ X \in [0, F_r] \rightarrow \text{reflection} \]

\[ X \in ]F_r, 1] \rightarrow \text{refraction} \]

2.1.2 Storing Photons

Photons are stored in the photon map as soon as they are created during the photon tracing. After this, the data structure is static. Each photon contains the position, the power and the incident direction. An important property of the photon map is, that photons are separate from the geometry of the model. Jensen[4] uses a kd-tree as a data structure, which is a multidimensional binary search tree (see [2, 3, 8] for a full explanation). Since the photon map is a three-dimensional point set, we will use a three-dimensional kd-tree. Each node of the kd-tree represents a three dimensional point, the position of the photon. Every node, except the leave-nodes, splits the point set for one of the three dimensions (x,y or z) into two pieces. Hence, each one of the pieces represents a sub tree. The dimension which has the the largest maximum distance between the points is chosen as the splitting plane. The median of this dimension is defined as the root of the tree, where on the left subtree all photons are below this median, and all photons on the right subtree are above. This procedure continues recursively with each of the subtrees. For estimating the radiance at one point, we need to find the nearest photons around this position; this is similar to range searching [3, 8]. Choosing the radius well is important, because a too small radius would cause noise and a too big radius would slow down the searching and the contribution of each photon would be minimal.

2.2 Rendering

2.2.1 Radiance Estimate

In computer graphics the rendering equation is a common technique in global illumination algorithms to calculate the outgoing radiance \( L_0 \) in direction \( \omega_o \) at a surface position \( x \). It results from the sum of the emitted
radiance $L_e$ and the reflected radiance $L_r$:

$$L_0(x, \omega_o) = L_e(x, \omega_o) + L_r(x, \omega_o)$$

(7)

$L_r(x, \omega_o)$ is computed as an integral over incident radiance $L_i$ from all directions $\Omega_i$:

$$L_r(x, \omega_o) = \int_{\Omega_i} f_r(x, \omega_i, \omega_o) L_i(x, \omega_i) (\vec{n} \cdot \omega_i) d\omega_i$$

(8)

where $\omega_i$ is the incoming direction, $\vec{n}$ the normal at $x$ and $f_r$ is the Bidirectional Reflectance Distribution Function [10](BRDF), which weighs the reflected radiance. It specifies, depending on the properties of the surface material, the relationship between the reflected radiance measured in direction $\omega_o$ and the irradiance on the surface coming from $\omega_i$:

$$f_r(x, \omega_i, \omega_o) = \frac{\Delta L_0(x, \omega_o)}{\Delta L_i(x, \omega_i)}$$

(9)

In [10] several BRDF models are shown.

Due to the necessity of radiance information for the integral and the fact that the photon map stores the flux $\Phi$ of each photon, we need to transform the equation (8). We know the definition of irradiance [10]:

$$E(x, \omega_i) = \frac{d\Phi(x, \omega_i)}{dA}$$

(10)

We can now transform the definition of radiance in equation (8):

$$L_i(x, \omega_i) = \frac{d^2\Phi(x, \omega_i)}{(\vec{n} \cdot \omega_i)d\omega_i dA} = \frac{dE(x, \omega_i)}{(\vec{n} \cdot \omega_i) d\omega_i}$$

(11)

Inserting (10) in the integral from above (8) turns into:

$$L_r(x, \omega_o) = \int_{\Omega_i} f_r(x, \omega_i, \omega_o) L_i(x, \omega_i) (\vec{n} \cdot \omega_i) d\omega_i$$

$$= \int_{\Omega_i} f_r(x, \omega_i, \omega_o) \frac{dE(x, \omega_i)}{(\vec{n} \cdot \omega_i) d\omega_i} (\vec{n} \cdot \omega_i) d\omega_i$$

$$= \int_{\Omega_i} f_r(x, \omega_i, \omega_o) dE(x, \omega_i)$$

(12)

Finally we can calculate the reflected radiance using the incoming irradiance. Jensen approximates the incoming flux $\Phi$ searching the $k$ nearest photons around estimation point $x$. Since every photon represents a fraction of the light source power, it carries the power $\Delta \Phi(\omega_i)$. According to Jensen we get:

$$L_r(x, \omega_o) \approx \sum_{p=1}^{k} f_r(x, \omega_i, \omega_o) \frac{\Delta \Phi_p(x, \omega_i)}{\Delta A}$$

(13)
where $\Delta A$ is the area around $x$, which covers the $n$ nearest photons. It can be calculated as the area of a circle:

$$\Delta A = \pi r^2$$

The radius $r$ is also called the *bandwidth*. It adjusts the trade-off between the bias and the variance of the resulting image. Thus a smaller bandwidth will take less photons into account, which reduces bias but also increases variance. A higher bandwidth will lead to the contrary. With the limitation of the area by taking the $k$’th nearest neighbour photons a smoothing effect occurs automatically, which is more distinctive while photon density is low and less when the density is high. Therefore, we get a limited bias and variance in the radiance estimation. At the same time, each photon is weighed equally. This can be improved with filtering techniques where e.g. photons lying more distant from estimation point $x$ are weighed less. Furthermore, more advanced filtering techniques can be used to improve the image quality, especially at edges and corners where the estimation could be wrong, because wrong photons are taken into account.

For temporal photon differentials we are going to modify the radiance estimate later, so, at this point, I won’t further discuss the standard photon mapping.

### 2.2.2 Visualisation

In the last step, we are going to visualize the scene. For this we can choose different ways. Photon mapping is perfect to simulate caustics in a scene. Therefore, it makes sense to build a separate caustic map which inherits all photons, which have been reflected or refracted several times before they finally strike a diffuse surface. These photons can be traced further, in case the Russian roulette algorithm decides to let a diffuse reflection happen. But if they hit, while being on their way, a diffuse surface again, they are stored in a second photon map along with all the other photons, which did hit a diffuse surface after emission first. This global photon map contains all types of light that can exist in a scene: direct illumination, indirect illumination and caustics.

To visualize the scene, we could simply use a ray tracer, which stops after the first intersection and uses only the photon maps to render the scene. That would be the easiest way, but with regard to the quality of specular or glossy reflections and direct illumination, the image would be noisy with too few photons. The reason is that the photons from the light source don’t cover all the directions we can have for specular reflections, and the possibility that the photons cover all shadowed or illuminated surfaces seen directly by the observer, accurately, is low, unless we emit a lot of photons. In respect to computation efficiency Jensen uses for these phenomena a ray tracer. Indirect illumination can be rendered directly with the global map,
using the simple ray tracing method from above, or in combination with a path tracer. In the latter method, at the first intersection point with a diffuse surface, a couple of sample rays are shot in random directions to estimate the incoming light, by using the global photon map at the next intersection with a diffuse surface.

3 Temporal Photon Differentials

In this section I will explain the concept of Temporal Photon Differentials. As we follow single photons in conventional photon mapping, we can imagine very near neighbour photons following similar paths. Thus, we will trace now for each emitted photon a bundle of its next neighbours as a beam of light. It will change its shape and size while shifting through the space. More precisely, these beams of lights can be mathematically described as the differentials of the emitted photons. They can be traced just as the original photons by deriving the equations for transfer, reflection and refraction and applying them as a first order Taylor approximation.

The photons $\vec{P}$ can be considered as rays with the initial point $\vec{p}$ and a direction vector $\vec{d}$. Their differentials in correlation with the variables $u$ and $v$, which relate to the light sources whence they were emitted, look like:

$$\frac{\partial \vec{P}}{\partial u} = \left\langle \frac{\partial \vec{p}}{\partial u}, \frac{\partial \vec{d}}{\partial u} \right\rangle$$

(14)

and

$$\frac{\partial \vec{P}}{\partial v} = \left\langle \frac{\partial \vec{p}}{\partial v}, \frac{\partial \vec{d}}{\partial v} \right\rangle$$

(15)

With these photon differentials as functions of space coordinates, we can approximate close neighbour photons using the first order Taylor series:

$$\vec{P}(u + \Delta u, v) = \vec{P}(u) + \frac{\partial \vec{P}}{\partial u} \Delta u + \mathcal{O}(\Delta u^2)$$

(16)

The distance between two photons results in:

$$\vec{P}(u + \Delta u, v) - \vec{P}(u) = \frac{\partial \vec{P}}{\partial u} \Delta u + \mathcal{O}(\Delta u^2) \approx \frac{\partial \vec{P}}{\partial u}$$

(17)

Here $\mathcal{O}$ denotes the remainder in Landau notation. Owing to the assumption, that $\Delta u$ is an infinitesimal value, we can reject the even smaller remainder as well.

Positional photon differentials can be imagined as footprints in form of a parallelogram made from their positional differential vectors $\frac{\partial \vec{p}}{\partial u}$ and

$9$
They are projected onto the surface’s tangent plane at the intersection points via the transfer equation shown in the next subsection. Exactly as in the normal photon mapping algorithm, the derivatives of a photon are only stored if they hit a diffuse surface. As soon as they hit an object, it would be decided with the Russian Roulette method also, if a photon is either diffusely reflected, specularly reflected, refracted or absorbed. In case of diffuse reflection, the photon differentials are reflected in a random direction. If it is decided that a reflection or refraction is going to occur, the photon differentials transferred to the point of intersection have to be traced with the derived formulas of reflection and refraction shown in the next subsections.

We can now simply add the time to the photon differential representation:

\[
\frac{\partial \vec{P}}{\partial t} = \left\langle \frac{\partial \vec{p}}{\partial t}, \frac{\partial \vec{d}}{\partial t} \right\rangle
\]

In an entirely static scene with static light sources the time dependent differentials in \( \frac{\partial \vec{P}}{\partial t} \) will stay zero. If the photon differentials are faced with a dynamic scene, their time dependent differentials will become non-zero. The direction of the time dependent positional differential vector \( \frac{\partial \vec{p}}{\partial t} \) shows us the predicted direction where on the surface the footprint is going to or from where it was coming. The magnitude tells us how far it will move or has moved. This means that with help of \( \frac{\partial \vec{p}}{\partial t} \) we can predict the behaviour of a footprint in future frames or figure it out for former frames. The direction and the magnitude can be estimated by using the known moving direction and speed of a surface where the footprint was transferred to and calculating finite differences for each time step. If we have dynamic lights also, we can include them in the estimation.

### 3.1 Transfer

For Temporal Photon Differentials the photons are emitted as in the normal photon mapping algorithm. But in addition, the corresponding photon differentials have to be emitted with them, too. The positional differential vectors are defined as the orthogonal basis of the surface tangent plane at the emission point, and the directional differential vectors are defined as the orthogonal basis of the plane which lies orthogonally to the emitted photon direction.

To transfer these differential vectors to the point of intersection at a distance \( s \) we need to differentiate the transfer equation in correlation with \( u \) (similar for \( v \)):

\[
\vec{Q} = \vec{p} + s\vec{d}
\]
To calculate the distance $s$ between the light source and a triangular flat surface composed of the vertices $Q_0, Q_1$ and $Q_2$ we need the implicit equation of a plane through point $\bar{Q}$ with surface normal $\bar{N}_{\text{flat}}$ and a point in the plane $\bar{Q}_0$, since the interior of a triangle represents a plane, too:

$$(\bar{Q}_0 - \bar{Q})^T \cdot \bar{N}_{\text{flat}} = 0$$

(21)

Inserting (19) in (21) we get:

$$0 = (\bar{Q}_0 - \bar{p} - s\bar{d})^T \cdot \bar{N}_{\text{flat}}$$

(22)

Hence, we can determine the zero point:

$$s = \frac{(\bar{Q}_0 - \bar{p})^T \cdot \bar{N}_{\text{flat}}}{d^T \cdot \bar{N}_{\text{flat}}}$$

(23)

In (20) we need the differential $\frac{\partial s}{\partial u}$ whose full derivation can be read in [11]:

$$\frac{\partial s}{\partial u} = \frac{\bar{N}_{\text{flat}}^T}{d^T \cdot \bar{N}_{\text{flat}}} \cdot \left( \frac{\partial \bar{Q}_0}{\partial u} - \frac{\bar{N}_{\text{flat}}^T}{\bar{N}_{\text{flat}}^T \cdot d^T} \cdot \frac{(\bar{Q}_0 - \bar{p})^T}{\partial u} - s \frac{\partial \bar{d}}{\partial u} \right) + \frac{s \bar{N}_{\text{flat}}^T}{d^T \cdot \bar{N}_{\text{flat}}} \cdot \frac{\partial s}{\partial u}$$

(24)

The spatial part of the flat surface normal differential is zero, but when we consider the temporal part, which becomes non-zero in dynamic scenes, we have to differentiate the normalized normal where $I_3$ denotes the Identity matrix:

$$\bar{N}_{\text{flat}} = \frac{\bar{n}_{\text{flat}}}{||\bar{n}_{\text{flat}}||}$$

to

$$\frac{\partial \bar{N}_{\text{flat}}}{\partial t} = \frac{\bar{n}_{\text{flat}}^T \cdot \bar{n}_{\text{flat}} \cdot I_3 - \bar{n}_{\text{flat}}^T \cdot \bar{n}_{\text{flat}}^T}{(\bar{n}_{\text{flat}}^T \cdot \bar{n}_{\text{flat}})^2} \cdot \frac{\partial \bar{n}_{\text{flat}}}{\partial t}.$$ 

(25)

For the calculation of $\frac{\partial \bar{Q}_{0,f}}{\partial t}$ we need the time derivatives of the intersected triangle vertices:

$$\frac{\partial \bar{Q}_{0,f}}{\partial t} = e(Q_{0,f+1} - Q_{0,f}),$$

(26)

where $Q_{0,f}$ is a vertex of the triangle at frame step $f$ in correlation with the time. With the factor $e$, we can set an exposure in frames. It smooths the image and effects motion blur. Dividing the exposure by the frame rate, we get the exposure time.
3.2 Reflection and Refraction

After the photon has been transferred onto the surface we can calculate the reflection and refraction direction. For the photon differentials we use the derivatives of the equations (4) and (6). At first, the differential direction of the reflected photon \( \frac{\partial \vec{w}_s}{\partial u} \) is [11]:

\[
\frac{\partial \vec{w}_s}{\partial u} = (I_3 - 2\vec{N} \cdot \vec{d}^T) \cdot \frac{\partial \vec{d}}{\partial u} - 2(\vec{d}^T \cdot \vec{N} + \vec{d} \cdot \vec{d}^T) \cdot \frac{\partial \vec{N}}{\partial u} \tag{27}
\]

The differential equation of refraction \( \frac{\partial \vec{w}_r}{\partial u} \) results in [11]:

\[
\frac{\partial \vec{w}_r}{\partial u} = (\vec{d} - (d^T \cdot \vec{N} - \frac{I_3 - \vec{N} \cdot \vec{d}^T}{\eta \sqrt{\xi}}) \cdot \vec{N}) \cdot \frac{\partial \eta}{\partial u} + (\eta \cdot I_3 - \eta(1 + \frac{\eta d^T \cdot \vec{N}}{\sqrt{\xi}}) \cdot \vec{N}) \cdot \frac{\partial \vec{d}}{\partial u} - (\mu I_3 + \eta(1 + \frac{\eta d^T \cdot \vec{N}}{\sqrt{\xi}}) \cdot \vec{N}) \cdot \frac{\partial \vec{N}}{\partial u} \tag{28}
\]

3.3 Shading

For non interpolated flat shaded surfaces, the differentiated surface normal \( \frac{\partial \vec{N}_{\text{flat}}}{\partial u} \) can be used to calculate the reflected and refracted photon differentials. In order to render a more sophisticated surface look, we have to interpolate the surface normal vector across the triangle for each hitting photon. This technique is also used for phong normal interpolation [10]. In this way it is not necessary to use high resolution geometry with very small polygons to produce smooth shaded surfaces. Hence, we have to use the differentiated interpolated surface normal to calculate the derivatives \( \frac{\partial \vec{w}_s}{\partial u} \) and \( \frac{\partial \vec{w}_r}{\partial u} \), since neighbouring photons will also be reflected and refracted according to the interpolated normal. However, the transfer equation (20) will be applied with the differentiated flat normal \( \frac{\partial \vec{N}_{\text{flat}}}{\partial u} \), because the neighbouring photons will intersect the surface corresponding to its shape, which is described by its geometric normal.

After we determined the intersection point \( Q \) of the photon with the formulas (22) and (23) inside a triangle, we are able to calculate the Barycentric coordinates:

\[
Q = \lambda_0 Q_0 + \lambda_1 Q_1 + \lambda_2 Q_2 \tag{29}
\]

even if it is assumed that \( \lambda_0 + \lambda_1 + \lambda_2 = 1 \) and \( 0 \leq \lambda_i \leq 1 \). This means that \( Q \) is always inside the triangle. If e.g. one coordinate is 0 and the others are between 0 and 1, \( Q \) lies on a border. We can solve the equation to find the photon with triangle intersection with the Möller and Trumbore’s
algorithm [5]. With help of the Barycentric coordinates, we can find the interpolated normal $\vec{N}$ at $Q$ by:

$$n = \lambda_0 \vec{N}_0 + \lambda_1 \vec{N}_1 + \lambda_2 \vec{N}_2$$

$$\vec{N} = \frac{\vec{n}}{||\vec{n}||}$$

where $\vec{N}_0, \vec{N}_1$, and $\vec{N}_2$ are the normal vectors of the vertices. Further, it is now possible to guess the change of $Q$ and the interpolated normal $\vec{N}$ inside the triangle by means of the differentials $\frac{\partial \lambda_i}{\partial u}$ as follows:

$$\frac{\partial \vec{N}}{\partial u} = \frac{\vec{n}^T \cdot \vec{n} \cdot I_3 - \vec{n} \cdot \vec{n}^T}{(\vec{n}^T \cdot \vec{n})^2} \cdot \frac{\partial \vec{n}}{\partial u}$$

$$\frac{\partial \vec{n}}{\partial u} = \vec{N}_0 \cdot \frac{\partial \lambda_0}{\partial u} + \lambda_0 \frac{\partial \vec{N}_0}{\partial u} + \vec{N}_1 \cdot \frac{\partial \lambda_1}{\partial u} + \lambda_1 \frac{\partial \vec{N}_1}{\partial u} + \vec{N}_2 \cdot \frac{\partial \lambda_2}{\partial u} + \lambda_2 \frac{\partial \vec{N}_2}{\partial u}$$

While doing this, attention should be paid to the triangle borders, as $\lambda_i + \Delta \lambda_i$ could exceed them. The full derivation of $\frac{\partial \vec{n}}{\partial u}$ is quite long and I refer at this point to [11] again. The derivatives of the vertices normals can be calculated similarly as in equation (26).

### 3.4 Temporal Radiance Estimate

For estimating the radiance at a surface location with temporal photon differentials, we need to extend the equation (8) for normal photon mapping. The photon differentials are in principle thought as ellipsoids, formed by their positional differential vectors and the surface normal at the intersected object. The cross section of these ellipsoids defines an ellipse which is the photon differential footprint. The ellipse is also used as a filter kernel. It changes its size while shifting through the scene. The power of the photon, to which the photon differential corresponds, is equally distributed in this area, so that at any point in this area the irradiance has:

$$E_p = \frac{\Phi_p}{A_p}$$

The size of the initial footprint can be used to induce smoothing. A bigger initial footprint size reduces noise and increases bias, and a smaller initial footprint will lead to the contrary. In case of choosing a larger footprint size automatically, more photons are used to estimate the radiance at a surface point, so, to limit computation time, we should reduce the number of photons we want to take into account. The size of the initial footprint can also be determined by a function, which scales the differential vectors in proportion to the size of the light source and the number of emitted photons.
There are different ways to estimate radiance in time. The simplest way would be a sampling method. For each frame we create a photon map and calculate for each photon the temporal photon differentials. For the first frame the full number of photons are calculated e.g. 100 photons. In the next frame step just a percentage of the total photon number is calculated and the rest of them are taken from the photon map of the former frame. The time dependent positional differential vectors $\frac{\partial \vec{p}}{\partial t}$ of the photon differentials show us the predicted direction in which the footprint is moving. Thus, the footprints of the photon differentials chosen from the older photon map are translated along these vectors and stored in the actual photon map. A high magnitude of the vectors denotes a higher moving speed, so that the translated distances increase, too.

If we decide to reuse 50 photon differentials of the former photon map at frame step $f - 1$ and translate them, we only have to calculate 50 new photon differentials. With this method we can make sure that photon maps lose their influence on the actual photon map as they are getting older. E.g. a photon map at frame step $f - 2$ will just have an influence on 25% of all photons and so on.

To calculate the irradiance which a footprint contributes, we need to calculate the area of the ellipse:

$$A_p = \frac{1}{4} \pi \left( \left\| \frac{\partial \vec{p}}{\partial u} \times \frac{\partial \vec{p}}{\partial v} \right\|^2 \right)$$  \hspace{1cm} (35)$$

To estimate the radiance we will extend the equation (13), so that we can use a kernel filter:

$$L_r(x, \omega_o) \approx \sum_{p=1}^{k} f_r(x, \vec{\omega}_i, \vec{\omega}_o) E_p K_s((x - x'_p) M_p^T M_p(x - x'_p))$$  \hspace{1cm} (36)$$

where $x'_p$ is the translated center of a spatial kernel and $K_s$ is a bivariate kernel function, which weighs the corresponding footprint of the $p$'th photon differential proportionally to its squared distance to the position $x$ on the surface, where we want to estimate the radiance. For filtering we need to transform from world space to the spatial filter space. The footprint is centred around $x'_p$. For this we construct a transformation matrix:

$$M_p = \begin{bmatrix} \frac{1}{2} \frac{\partial \vec{p}}{\partial u} & \frac{1}{2} \frac{\partial \vec{p}}{\partial v} & \vec{n} \end{bmatrix}^{-1}$$  \hspace{1cm} (37)$$

where $\vec{n}$ is a vector which is orthogonal to the positional differential vectors.

With the simple sampling method, only the position of the footprint is translated. The shape and the angle towards the observer stays the same. In dynamic scenes it is obvious that the perspective of the moving objects
towards the observer are changing all the time, and therefore, the shape of
the footprints on the object surfaces is changing, too. For longer time
distances this could lead to falsified images. By reducing the footprint size of
the translated photon differentials for each frame step, we could limit the
trust into movement prediction and in the same step, reduce the potentially
caused error of not modifying the shape of the footprint. If we choose to in-
clude this feature to the sampling method from above, we should consider
how to distribute the energy. If we make the footprints smaller and do
not change the energy, the irradiance gets more concentrated into a smaller
area, which would result in over radiation at this surface point. Therefore,
the irradiance has to be reduced in proportion to the downsized footprint.
As consequence, we have to reuse more photons of older photon maps to
keep the energy in our image constant. Taking the same example from
above of calculating 50 new photon differentials for frame step \( f \) and as-
suming that at every frame step the footprint size is halved, the power for
this photon differential is also halved. Thus, we need twice as many photon
differentials of the photon map than in frame step \( f - 1 \).

As one can imagine, it is not difficult so far to apply other known phe-
nomena to trace photon differentials. If we know the formula of a phe-
nomenon, we can simply derive it and use it for the calculation of the
photon differentials. Even non physical effects like phong shading, from
which we used the technique of interpolating triangle surface normals,
Bump mapping and so on, can be applied to photon differentials. If we
want to predict the rotation of the objects, we need a second order deriva-
tion of the formulas, as well. But in my case, only the moving direction of
the objects can be predicted.

4 Implementation

For the implementation of temporal photon differentials I used an exten-
sive framework from the University. It already provides the functional-
ity of ray tracing, conventional photon mapping and differential photon
mapping, which I extended to support temporal photon differentials. The
framework is originally intended to render static scenes, without object
movement. I modified the code so far, that simple scenes with moving ob-
jects can be rendered, as well. For this I am using Hermite interpolation. A
moving object is assigned with controlling points and corresponding tan-
gents, with whose help an interpolated path is calculated [6]. For every
frame step, the covered object distance is calculated and then the interpo-
lated object position on the path is determined. The object is translated to
this position and loaded into the scene. In addition, the vertex and normal
derivatives for each object are calculated like in equation (26) and stored
with them, as well. After the frame is rendered, the scene is cleared and the
objects of the next frame are loaded. For the rendering I implemented the sampling technique described in the previous chapter. Two photon maps are computed, one for the actual frame and the second for the former frame. They store in addition to the photon position, power and direction, the positional photon differentials. In the photon map building phase, all photons are computed for the first frame step. In the next frame step, just a defined percentage of the overall used photons are calculated and the rest is randomly taken from the former frame, translated along the time dependent positional differential vector and stored in the actual photon map. Since only the former photon map is used to complete the actual photon map, storing two photon maps is enough, with the consequence that the older photon map of frame \( f - 2 \) is refused at every iteration step and overwritten by the new actual photon map. For the radiance estimate the equation (36) is used, which is the same as in differential photon mapping. As long as the number of photons keeps the same in each map and since we scale the photons, the amount of the overall emitted power does not vary, as old photons are reused. Therefore, the essential parts for the extension incorporate the maths for calculating the temporal differentials, the creation of dynamic scenes and the possibility to mix two photon maps. The tracing and estimating part does not change at all.

5 Experiments

In my experiments I compare the quality of the caustics produced by the algorithms of conventional photon mapping and temporal differential photon mapping. For both algorithms I used the same scene setting, in which a rotated cognac glass is flying on a curve in a Cornell box. On the floor an area light is emitting photons into the scene. The scene is rendered with 25 frames per second. The program emits photons as long as 1000000 photons are stored in the caustic map and 500 photons are taken into account for the radiance estimate. In the reference picture in Figure 3 we see a rendered caustic map with conventional photon mapping. The glass is flying on the drafted curve in the image. There is a change of altitude and direction at each point. With this distinctive movement of the glass, it is easier to evaluate the approximation of the footprint movement.
As an extreme example the Figure 4c shows a close up of a rendering using temporal differentials, where 90% of the photons were taken from the former photon map. The glass is flying after 17 frame steps into a curve, thus it changes the moving direction very fast. Apparently, we can see some turbulences in the caustics. This is due to the fact, that the former photon maps have relatively much higher influence than the new calculated photons for the actual frame. In this case, the first build photon map at frame step 1 has after 17 frames still an influence of ≈ 17% on the actual photon map. These photons are translated into a direction which differs significantly from the direction in which the glass is actually moving. This leads to a drifting of the caustics and causes variance in the image. But still, we can recognize roughly the form of the caustics. For a still image compared to the reference image using conventional photon mapping, it is not satisfying. In fast moving scenes, these differences can be missed by the observer. In the next experiment I decreased the percentage of the reused photons to 50%. The influence of the first photon map to the 17th photon map is now much smaller.
map amounts to an insignificant value well under 0%. Therefore, less former photon maps will have significant influence on the actual image. In Figure 4b, we can see the reduced variance in the caustics in a close up, and in Figure 4a we see a close up of the same region of the reference image.

![Conventional photon mapping](image1.png)

(a) Conventional photon mapping

![Reusing 50% of old photons](image2.png)

(b) Reusing 50% of old photons

![Reusing 90% of old photons](image3.png)

(c) Reusing 90% of old photons

**Figure 4: Temporal photon differentials**

The same in the second curve, in these close-ups we can see in figure 5a conventional photon mapping, in figure 5b temporal photon differentials reusing 50% of old photons, and in 5c 90% of old photons are reused.
Figure 5: Temporal photon differentials
The great advantage of temporal photon differentials is the reduced calculation time for the photon maps. The building of the photon map is working sequentially and I compared the relative speed-up of the temporal differentials method to the conventional method using the same processor. In a second long sequence of the scene above with 25 frames per second, one million photons in every photon map and reusing 50% of old photons, the average calculation time for one photon map is 111.01 seconds, for conventional photon mapping the average time is 138.99 seconds. This is almost a calculation time speed-up of a factor 1.25. The expensive photon tracing has to be computed only for half of the overall used photons, where especially the intersection calculation with the geometry costs much time. As I am using for both algorithms the same photon amount, the speed up is not that huge at first view. But since photon differentials approximate a bundle of photons, we can reduce the number of photons in temporal photon differentials and save computation time. In Figure 6b the same frame from Figure 6a is shown, but with 500000 photons in the caustic map and 250 photons in the radiance estimate. The difference is not recognisable, unless one concentrates and compares the edges of the caustics.

Figure 6: Temporal photon differentials with reduced photon number
However, using temporal photon differentials causes variance, which affects the overall image quality. Finding a good trade-off between bias and variance in the final image is a challenge for itself. As I mentioned in the radiance estimation, we can use more advanced filter kernels. Furthermore, we could use e.g. a Gaussian filter in a post process step, to slightly smooth the image and reduce variance.

I have to mention an anomaly manifesting as a black beam through the image, which occurs in some situations. In Figure 7 a close up shows this error. I could not find a good explanation for this problem, but I have some ideas about possible sources of errors. I think that a sign error arises in the code, so that the normals are showing into the opposite direction, thus the calculations of the temporal differentials get wrongly and the radiance estimate at the end too. I do not believe, that there is an error in the mathematics. I also thought, that the meshes were calculated wrong, but this was disproved, since the same meshes are used in the conventional photon mapping algorithm, which works faultless. However, the program works mostly fine and I could work with it.

Figure 7: Anomaly
6 Conclusions

In this thesis I showed the basic overview of the photon mapping concept, the mathematics behind temporal photon differentials and described the ideas on the implementation of this technique. The experiments give an impression of the potential of the method and demonstrated, that temporal photon differentials produce good and fast approximation of caustics. The algorithm I used for building the photon maps, was working sequentially, hence an increasing calculation speed can be reached by massive parallelisation of the photon tracing part, using multi-core processors or graphics hardware.

The concept could be extended with methods, which can handle the trade-off problem between bias and variance and additional filtering techniques as post process effects efficiently. Further, to provide more sophisticated animations, a second order approximation of the photon differentials would allow to render rotating objects, too.

The approximation of natural phenomena for real time applications, e.g. computer games, is used extensively. After the examination of the topic of temporal photon differentials, I believe that especially the speed benefit of predicting the behaviour of light in correlation with the time, will be used in these applications very soon.

References


