Multiple Live Video Environment Map Sampling

Tomáš Nikodým
Czech Technical University,
Faculty of Electrical Engineering
Czech Republic, Prague
nikodtom@fel.cvut.cz

Vlastimil Havran
Czech Technical University,
Faculty of Electrical Engineering
Czech Republic, Prague
havran@fel.cvut.cz

Jiří Bittner
Czech Technical University,
Faculty of Electrical Engineering
Czech Republic, Prague
bittner@fel.cvut.cz

ABSTRACT

We propose a framework that captures multiple high dynamic range environment maps and decomposes them into sets of directional light sources in real-time. The environment maps, captured and processed on stand-alone devices (e.g. Nokia N900 smartphone), are available to rendering engines via a server that provides wireless access. We compare three different importance sampling techniques in terms of the quality of sampling pattern, temporal coherence, and performance. Furthermore, we propose a novel idea of merging the directional light sources from multiple cameras by interpolation. We then discuss the pros and cons when using multiple cameras.

Keywords

Image-based lighting, environment map sampling, augmented reality

1 INTRODUCTION

The *augmented reality* (AR) allows us to merge real-world images with computer generated content. AR applications have gained increasing attention over past years. We aim to enhance the believability of rendered images in AR by novel methods for real-time consistent illumination computation. We focus on methods for live acquisition of high dynamic range (HDR) environment maps in indoor environments. We compare three methods for their decomposition into sets of directional light sources by means of importance sampling.

There are three interconnected ideas discussed in this paper. Firstly, we propose a framework for live capturing of HDR environment maps and their importance sampling. Secondly, we compare three existing importance sampling techniques in terms of the quality of sampling pattern, temporal coherence, and performance. Thirdly, we introduce a novel idea of using multiple cameras to allow for the interpolation of acquired environment maps.

For efficient illumination of the scene with the acquired environment map, we approximate them by a set of directional light sources [19] with the use of importance sampling algorithms. Most of these algorithms published in the past targeted static environment maps. When dealing with dynamic environment sequences, it is important to maintain a temporal coherence of the sampling pattern to avoid visually disturbing frame-to-frame flickering. Apart from that, if the algorithm is to be used in an interactive application, the processing must run at interactive frame rates.

One of our design goals is to provide a low-cost hardware platform. Therefore we use already available mobile smart phone on the market, Nokia N900. It has a programmable camera [1] and provides sufficient computational resources to do all the processing on-chip. Thanks to the mass production of smartphones their use is cost effective compared to custom hardware. Furthermore, modern smartphones provide a broad range of connectivity (e.g.WiFi), and have a built-in camera of reasonable resolution, frame rate and quality. These features make them a suitable piece of hardware for online environment map acquisition and processing in mobile AR settings. Processing the captured images on-chip reduces the bandwidth required to transfer the data to a computer running the rendering engine. Attaching a fish-eye lens to the build-in camera provides nearly 180-degrees field of view. The smartphone can be programmed to capture a burst of varying exposure images that are fused into a single HDR image.

The framework has been proposed to be used for illumination computation in AR applications. Existing AR systems include e.g. the *Studierstube* [20] and *Spinnstube* [24]. Ideally, the virtual objects should be merged seamlessly into the real scene (see [14] for an overview of common illumination techniques). The ability to illuminate virtual objects based on the lighting conditions of the surrounding environment adds realism to the augmented scene and enhances the user experience. For a detailed discussion of existing AR for example in virtual television studios, see [9, 3].

The paper is further structured as follows. In the next section we recall the related work. In Section 3 we describe the used importance sampling algorithms in detail. The selected algorithms are then extended in Section 4 by novel techniques that allow to merge data from multiple cameras. After describing some implementation details in Section 5 we present an evaluation of the selected algorithms and their extensions in Section 6. The last section 7 concludes the paper.

2 RELATED WORK

In this section, we review existing methods for environment map sampling and discuss their suitability for sampling of live video environment maps.

Structured importance sampling proposed by Agarwal et al. [2] combines stratified and importance sampling. The proposed importance metric takes into account both surface area and integrated illumination of a hemispherical region. The number of samples can be controlled. For dynamic sequences, temporal coherence of sampling pattern is poor [12]. The computation time (tens of seconds) is far from interactive.

The algorithm based on Lloyd's relaxation, proposed by Kollig et al. [15], yields good results for static environment maps. The number of samples can be easily controlled. The time to process single environment map is dozens of seconds, too high for real-time applications. Also, the algorithm exhibits very poor temporal coherence as even local changes can cause global changes in the sampling pattern of the entire environment map.

The hierarchical importance sampling algorithm proposed by Ostromoukhov et al. [17], based on Penrose tiling, can operate at interactive frame rates. Furthermore, the sampling pattern produced by this algorithm exhibits relatively good temporal coherence. However, it is difficult to control the number of samples as it depends on the environment map being sampled.

The median cut sampling algorithm proposed by Debevec [7] is fast enough to be used in interactive systems. However, it exhibits poor temporal coherence of sampling pattern. Localized changes in illumination affect the entire sampling pattern. Another drawback of this algorithm is that the control over the number of samples is limited.

The probability density function (PDF) based importance sampling method, proposed by Havran et al. [12], targets dynamic environment maps specifically. The sampling algorithm uses an inverse transform method for hemispheres proposed by Havran et al. [11]. It is similar to the standard inversion procedure, used for importance sampling of static environment maps by Pharr and Humphreys [18] and Burke et al. [5]. However, the inverse transform method proposed by Havran et al. [12] exhibits better continuity and uniformity, which leads to better stratification of the resulting sample positions. To handle dynamic environments, the method uses two low pass filters to improve on temporal coherence. The first filter normalizes the intensity of light sources to preserve total energy of the system. The second filter suppresses high frequency movements of light sources. Invisible light source elimination and light source clustering methods are used to improve rendering performance. The method exhibits good temporal coherence, real-time performance and the number of light sources can be chosen. However, the temporal filtering causes a temporal lag for abrupt lighting changes.

The importance sampling method for dynamic environment maps, proposed by Wan et al. [23], is based on quadrilateral subdivision of a sphere. Firstly, the sphere is mapped into 2-dimensional space using the HEALPix mapping [13, 10]. A quad tree is adaptively constructed over the quadrilaterals (referred to as *quads*). At every step, the region with highest importance is further subdivided into four quads. This process is repeated until required number of samples is reached. The number of samples can be adaptively changed. The method is fast and the results for static scenes are comparable with methods such as structured sampling [2] and Penrose-based sampling [17]. It exhibits very good temporal coherence, which makes this method well suited for dynamic scene lighting.

Spatio-temporal sampling proposed by Wan et al. [22], based on Q2-Tree sampling [23], further exploits temporal and spatial coherence of environment sequences. Their method treats an environment map sequence as a volume constructed by stacking up all the frames in the chronological order. The proposed method produces a temporally coherent sampling pattern with slightly better characteristics than the original Q2-Tree sampling algorithm. A limitation of this method is that the entire environment sequence needs to be known in advance. For this reason, the method cannot be adopted for online processing of video frames in real-time.

A mobile system using environment map illumination was presented by Son et al. [21]. It uses a custom based camera. Processing of the environment maps and rendering images runs on an iPhone. The proposed system also uses markers for position tracking.

3 ENVIRONMENT MAP SAMPLING

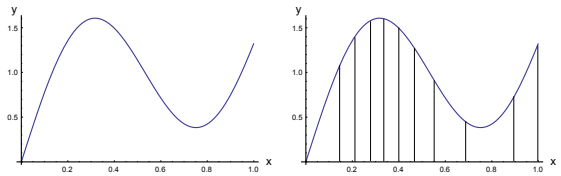
Our framework builds up on three existing environment map sampling algorithms. In this section, we provide details of the three algorithms. We then discuss our extension of these algorithms in Section 4.

Throughout this paper, we will use the following terminology. We will refer to the re-implementation of the PDF-based sampling algorithm for static environment maps [18] as *Pharr*. The extension of this algorithm for dynamic environment sequences [12], we will refer to

as *Hemigon*. The re-implementation of the algorithm based on Spherical Q2-Tree for sampling dynamic environment sequences [23], we will refer to as *Q2-Tree*.

3.1 Importance Sampling Methods based on Probability Distribution Function

Several environment map sampling algorithms based on the probability distribution function have been proposed in the past [18, 5, 12]. PDF-based importance sampling of dynamic environment maps suffers poor temporal coherence. Even if the same quaisi-random sequence is used for consecutive frames, localized changes in the environment map largely affect the global sampling pattern. As has been shown by Havran et al. [12], low pass filtering in the time domain can decrease the flickering artifacts. In this section, we recall inverse transform methods (e.g. described in [8]) based on the PDF and its application to environment maps.



(a) PDF (b) Sampled PDF

(c) CDF (d) Sampled CDF

Figure 1: Sampling of a 1-dimensional function via standard inversion method. (a) A probability distribution function (PDF), (c) corresponding cumulative distribution function (CDF). (b) and (d) are the two functions with drawn (the same) samples.

In Figure 1, we illustrate the inverse transform method on a 1-dimensional function. Firstly, a cumulative distribution function (CDF) is constructed from PDF. The construction of a CDF is computed in linear time using an iterative formulation. Then, a sequence of random numbers is drawn with uniform probability. Each of these numbers is projected via the CDF. Figure 1(d) illustrates the inversion of CDF in form $x = CDF^{-1}(y)$ that generates samples in x.

Sampling of an environment map is in principle importance sampling of a 2-dimensional discrete function, defined over hemisphere. A 2-dimensional PDF is constructed from the luminance of the pixels of the environment map. Supposing the environment map is in a latitude-longitude format, the intensity of each pixel needs to be multiplied by $sin(\theta)$ to account for smaller angular extent near the poles, where θ is the altitude angle. In practice, quasi-random number generators with uniform distribution, such as Halton sequence [8], are often used. These 2-dimensional quasi-random number vectors are then mapped via the CDFs to an altitude angle θ and an azimuth angle ϕ. This method has been successfully applied for importance sampling of static environment maps by Burke et al. [5], and Pharr and Humphreys [18].

Havran et al. [12] proposed an extension of this method to improve the temporal coherence for dynamic environment mapping. They use a mapping of PDF over hemisphere proposed by Havran et al. [11]. It avoids discontinuity for north pole and for $\phi = 0$. To decrease temporal flickering during the sampling of a dynamic environment sequence, a history of the sampling patterns for a few past frames is kept, and two low pass filters are applied, operating in the time domain.

The first low pass filter operates on the total energy of the light sources. It suppresses flickering caused by high frequency light sources, such as fluorescent tubes. The second low pass filter operates on the trajectories of the samples, suppressing high frequency movements.

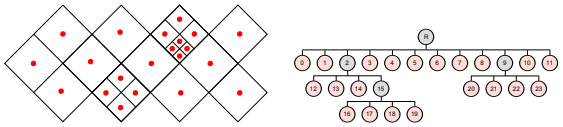
3.2 Importance Sampling Methods based on Subdivision

As opposed to PDF-based sampling methods, this class of importance sampling methods constructs a hierarchical data structure by adaptively subdividing the environment map into spherical regions [17, 23, 7, 22]. There are two approaches to the construction of such subdivisions. The first one, which we will refer to as *median split*, splits a region into two or more subregions of equal importance [7]. The second approach, referred to as *uniform split*, decides whether to further split the region or not, depending on its importance [17, 23, 22]. By region splitting two or more equally sized subregions are created. The importance metric typically takes into account the luminance of the region and its angular extent (see Equation 1). Usually one sample is placed inside each created region at the bottom of the hierarchy (leaf nodes).

Sampling patterns produced by *median split* can exhibit poor temporal coherence. This is because even local changes in the illumination affect the size or shape of subregions at the top of the hierarchy, so the sampling pattern differs dramatically. Thus, their use in dynamic environment sampling is limited. On the other hand, sampling patterns produced by *uniform split* usually exhibit good temporal coherence. This is due to the local nature of this class of subdivision methods. The size and shape of a subregion at a particular level of the hierarchy is independent of the environment map being sampled. The sampling pattern for each subregion

is not affected by illumination changes in other subregions, which stabilizes the sampling temporally.

In particular, the *Q2-Tree* sampling algorithm [23] is suitable for sampling of live video environment maps. It is fast, produces sampling patterns that exhibit strong temporal coherence and the number of samples can be adaptively changed without the impact on the temporal coherence for consecutive frames. The *Q2-Tree* algorithm is described in the rest of this section.



(a) Subdivided environment map (b) Corresponding *Q2-Tree* representation

Figure 2: Visualization of the Q2-Tree sampling algorithm.

Firstly, the sphere is mapped into 12 quadrilaterals (*quads* for short) using the HEALPix mapping [13, 10]. Then, a quad tree is iteratively constructed on the base quads. At every iteration, the quad of the highest importance is subdivided into four equally-sized subquads. All quads at the same level of subdivision have always the same surface area. The construction is complete when the required number of leaf nodes has been created. The construction is illustrated in Figure 2, showing a subdivided environment map in HEALPix mapping and a corresponding *Q2-Tree*.

The pseudo-code for *Q2-Tree* construction is given in Algorithm 1. It maintains an importance-sorted list of interior and leaf nodes. At every iteration, the left-most leaf node (highest importance) is subdivided into four quads, until the needed number of leaf nodes is created.

Algorithm 1 Sampling dynamic environment map with a *Q2-Tree*.

```
Input:
  EnvironmentMap: Spherical light probe of the environment
  RequiredNumberOfSamples: Integer

Output:
  LightSources: List of directional light sources

Algorithm:
  Q2TREE = HEALPixMappingTo12Quads( EnvironmentMap )
  EvaluateImportance( Q2TREE.Nodes )
  while(Q2TREE.NumberOfNodes < RequiredNumberOfSamples)
    NewNodes = Subdivide( Q2TREE.LeafNodes[0] )
    EvaluateImportance( NewNodes )
    Q2TREE.MoveToInterior( Q2TREE.LeafNodes[0] )
    Q2TREE.AddLeafNodes( NewNodes )
  LightSources = CreateLightSources ( Q2TREE.LeafNodes )
  return LightSources
```

During the sampling process, it is desirable to sample more densely the bright regions and less densely the dark regions. On the other hand, oversampling small bright regions should be avoided as they can be well approximated with fewer samples due to their small solid angle. The *Q2-Tree* sampling algorithm uses importance metric proposed by Agarwal et al. [2]. It combines good stratification of samples and higher density of samples in bright regions. The importance is given by the following equation.

$$p = (L)^a \cdot (\Delta\omega)^b, \tag{1}$$

where L is the total illumination of the region and $\Delta\omega$ is the solid angle. Parameters a and b are non-negative constants that are used to favor illumination (large value of a) or solid angle (large value of b) component. The result, p in the above equation, is the importance of the region. In their work, Wan et al. use constants of $a = 1$ and $b = \frac{1}{4}$ [23].

The equal solid angle property of the subdivision proposed by Wan et al. allows for computation of both terms of the importance metric in constant time. The solid angle of a quad at level i can be directly computed as $\Delta\omega = \frac{\pi}{3 \cdot 4^i}$. The illumination term is computed using a technique commonly used for texture prefiltering, known as summed area tables [6].

When the *Q2-Tree* is constructed, a single directional light source is positioned inside each of the leaf quads. The color and intensity of the light source is given by summed illumination of corresponding pixels. Wan et al. propose that the light source should be jittered deterministically around the centroid of the region to avoid regularity but maintain determinism.

For dynamic sequences of environment maps, the *Q2-Tree* need not be rebuilt from scratch for every frame. First, the importance of each region is re-evaluated, creating inconsistency in the importance-sorted list. A series of merge-and-split operations is then performed until the sorted order of the list is recovered.

4 MERGING OF SAMPLING DATA FROM MULTIPLE CAMERAS

As we have already mentioned, the acquisition and processing of the environment maps run on a smartphone. We use a wireless connection between the smartphone and the computer that runs the rendering engine (referred to as client). Such a design makes it possible to place the camera (i.e. smartphone) anywhere in the room where the lighting is to be measured. Furthermore, it is possible to place several cameras at different places. As the processing is done on-chip and the amount of data to be sent per frame is low, the number of cameras is not limited by the computational resources nor the bandwidth of the client computer. We investigated the advantages and limitations of such use of multiple cameras. We propose algorithms for merging of sampling data from multiple cameras and describe them in this section. We present the results in Section 6.

In order to compute consistent common illumination between the real and virtual objects, the illumination should be recorded at the spot where the virtual object is to be placed. This is often impractical as for example the virtual objects move in virtual television studios and it would require to move the camera together with the virtual object. Second, in AR display systems, the presence of a device inside the workspace would be disruptive for the user. It is thus inviting to approximate the illumination inside the workspace by placing several cameras around it. Furthermore, using multiple cameras could be used to improve the frame rate, temporal coherence and robustness of the system.

Before the data from multiple cameras can be merged correctly, it is necessary to perform geometric and photometric calibration. The photometric calibration requires a lux meter and is performed once for each camera. The geometric calibration needs to be repeated when the lens of the camera is repositioned. We have used external software tools based on OpenCV [4]. Multiple views of a checkerboard are taken, and the camera calibration is computed from detected corners of the checkerboard. The complete calibration of one device takes about 10 minutes to perform and does not have to be repeated unless the lens are replaced. Because of lack of space we do not discuss details in the paper.

4.1 Merging for PDF-based Methods

For the sampling algorithms based on probability distribution function (*Pharr* and *Hemigon*), the merging is straightforward. Given that the number of samples is the same for all cameras and that the samples were generated using the same sequence of quasi-random vectors, the direction, color, and intensity of each sample can be computed by linearly interpolating the direction, color, and intensity of the corresponding samples from each of the cameras.

In order for the linear interpolation of directions to be applicable, it is essential that each set of samples to be interpolated was generated from the same quasi-random vector. Typically, probability distribution function based sampling algorithms use low-discrepancy sequences, such as Halton sequence [8]. By using the same quasi-random sequence for each of the cameras, the correspondence between samples can be determined trivially from their index, supposing azimuth alignment of the cameras. Obviously, the result of such interpolation is only approximate of the real ground truth data measured in the spot, for which the interpolation was computed.

4.2 Merging for Subdivision Methods

For the *Q2-Tree* sampling algorithm, merging of data from multiple cameras is more complicated. The depth of subdivision of a particular branch can be different for each camera. Merging the Q2-Trees naively would produce non-deterministic number of samples, dependent on the environment maps. Also, the luminance of the merged tree needs to be normalized in order to preserve the total power.

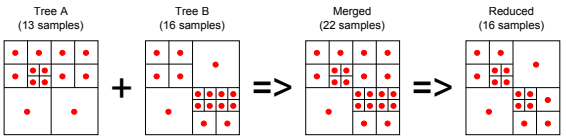


Figure 3: Visualization of the Q2-Tree merging algorithm. The two trees A and B are first merged together and later simplified to the number of samples needed.

Below, we present our *Q2-Tree* merging algorithm, given in pseudo-code in Algorithm 2. Since the original *Q2-Tree* construction algorithm uses a list-based representation of the tree, we first need to build a hierarchical structure for each of the trees to be merged. We then perform a parallel traversal of the input trees, producing a merged tree. The input trees are traversed to the maximum depth and for each node, we compute the sum of luminance of the corresponding input nodes that are available. Since the depth of a subtree can be different in each of the inputs, this step produces a tree of potentially more nodes than any of the inputs. Given camera i produced N_i samples, the merged tree can have anything between $min_{i=1}^{k}(N_i)$ samples to $\sum_{i=1}^{k} N_i$ samples. See figure 3 for illustration.

Algorithm 2 Merging of multiple Q2-Trees

```
Input:
  N: Required number of samples
  Q2[k]: List of Q2-Tree nodes for each of k cameras

Output:
  Samples[N]: List of samples
              (direction, luminance, color)

Algorithm:
  // Step 1: Tree reconstruction
  Q2Trees[] = ReconstructTrees(Q2)
  // Step 2: Merging (parallel traversal)
  MergedTree = Merge(Q2Trees)
  // Step 3: Normalization of luminance
  NormalizeLuminance(MergedTree)
  // Step 4: Importance re-evaluation (I = L^a * W^b)
  EvaluateImportance(MergedTree)
  // Step 5: Reduction to N leaves (sort-and-merge)
  FinalTree = Reduce(MergedTree, N)
  // Step 6: Sampling (Inverse HEALPix mapping)
  Samples = PlaceSamples(FinalTree)
  // Return list of samples
  return Samples[]
```

Also, the property that the luminance of an interior node equals the sum of luminances of its children no longer holds. This property is restored by the next step of our algorithm. First, the luminance of the root node is computed by averaging the luminance of the root nodes of the input trees (i.e. total luminance of the environment

map). Then, the merged tree is traversed and for each interior node, the luminance is recomputed using the following formula: $L_i' = L_i \cdot \frac{L_P}{\sum_{k=1}^{4} L_k}$, where L_i is luminance of child i, L_P is luminance of the parent, and $\sum_{k=1}^{4} L_k$ is the sum of luminances of the four children of P (including node i). Since the tree is traversed top-down and the root is already normalized, the correct value L_P is computed for each interior node before the node is visited.

We then re-evaluate the importance of each interior node, using the formula $p = (L)^a \cdot (\Delta\omega)^b$ proposed by Agarwal et al. [2]. The same values of the constants a and b are used as in the Q2-Tree construction ($a = 1$ and $b = \frac{1}{4}$). The list of interior nodes is then sorted in order of importance.

The next step reduces the number of leaf nodes to a user defined constant, using the importance-sorted list. At each iteration, we take the interior node of the lowest importance and cut off its subtree. As each interior node has four immediate children, this operation usually reduces the number of leaves by three (although a higher number is possible if the input trees differ significantly). This operation is repeated until the number of leaves drops below a user defined constant.

Now that the merged tree has approximately N leaf nodes and their luminance is correctly normalized, one sample is placed inside each leaf. The leaf nodes correspond to quadrilaterals in the HEALPix mapping [13, 10], so we use the inverse HEALPix mapping to project the sample positions back into the world coordinate system. The number of samples can be chosen with the precision of three samples, as four leaves can be always collapsed to a single leaf. The asymptotic time complexity of the proposed merging algorithm is limited by sorting of the interior nodes, $O(n \cdot \log n)$. The space complexity is linear in the number of samples.

5 IMPLEMENTATION

In this section, we present the architecture of our framework. It was one of our design goals to implement a framework that is cost-efficient, robust, extensible and easy to integrate into existing rendering engines.

The light probes (i.e. environment maps) are being captured on stand-alone devices that are capable of wireless communication. Such a device has a programmable camera and sufficient computational power to process the light probes on a chip. The on-chip processing steps include capturing a burst of varying exposure images, HDR image fusion, mapping of the captured image into the polar coordinate system, computing luminance, and importance sampling. Also, the device runs an HTTP server that provides an interface between the environment map acquisition and processing algorithm, running on a chip, and the rendering engine, running on a remote computer. Compliance with the HTTP protocol simplifies the configuration and debugging, because a standard web browser can be used to communicate with the device. The conceptual diagram is illustrated in Figure 4.

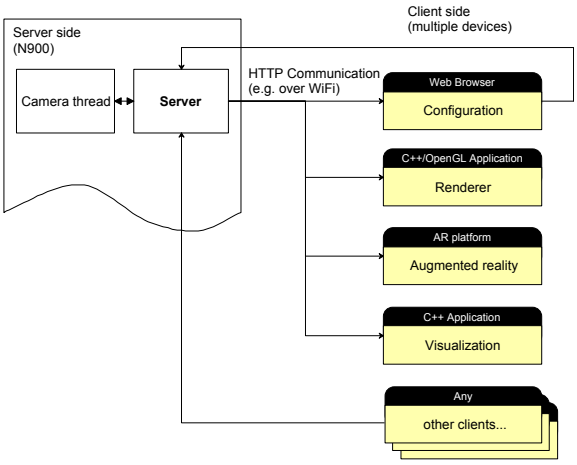


Figure 4: Conceptual diagram of our framework. The device running the sampling algorithm as a server communicates possibly with multiple clients over network.

The framework allows for multiple acquisition devices running asynchronously. The rendering engine can then communicate with all of these devices and merge the sampling data they provide. It is also possible for several client applications (rendering engines, configuration tools, etc.) to communicate simultaneously with a single acquisition device.

We have used the Nokia N900 smartphone as the environment map acquisition and processing hardware. Using a cell phone has several benefits, but also limitations. First of all, the computational resources are less powerful than those of a desktop computer. Especially the cache size is very limited. This makes it more challenging to design an efficient implementation. In particular, the benefit of accessing memory at spatially and temporally coherent locations is more pronounced. On the other hand, the small size of the device that integrates all the required features and is readily available on the market makes it a good choice for mobile augmented reality settings. It is easy to manipulate and cost-efficient.

Although we use a particular model of a cell phone (Nokia N900) in this paper, the implementation is platform independent to the extent of accessing the programmable camera and could be easily adopted to other hardware.

6 RESULTS

In this section we present the results of our work, where the test scenario is shown in Figure 5. Firstly, we

compare the three discussed importance sampling algorithms. Then, we discuss the advantages and limitations of merging the data from multiple cameras.

6.1 Comparison of Importance Sampling Algorithms

Below we summarize the results briefly for all three algorithms. Both PDF-based methods produce good results for environment sequences where the frame-to-frame changes of illumination are subtle. Temporal filtering of sample positions and intensities improves on temporal coherence but causes unwanted temporal lag if the changes are abrupt. The sampling method based on subdivision handles abrupt changes successfully, especially if the changes are local, but has problems handling subtle light source movements.

Quality of Sampling Pattern

Both implemented PDF-based importance sampling algorithms (*Pharr* and *Hemigon*) produce samples of equal power (in fact the brightness as luminance is taken as PDF). This means that the distribution of the samples is proportional solely to the distribution of the power in the captured environment map. While in general such a sampling pattern makes a good approximation of the environment map, in some cases under sampling of relatively dark regions might be a problem. Suppose we have an environment map that contains one bright light source and several much dimmer light sources. A sampling pattern that puts almost all of the samples in the small bright region would be reasonable for most views of the rendered scene. But when the main light source gets obstructed and the view being rendered is in a shadow, then the dim light sources come into play. Sampling small bright regions thoroughly and under sampling the rest of the environment produces poor results in such cases. For these reasons, it is desirable to maintain a good stratification of samples. Using such an importance metric that takes into account both brightness and angular extent of a region produces better results in these situations.

Due to the nature of HEALPix mapping, the *Q2-Tree* algorithm requires relatively many samples to approximate an environment map well (approx. 200 or more samples). The environment map is subdivided uniformly into twelve regions. An adaptive quad tree subdivision is then constructed separately on each of the twelve regions, placing one sampling into each of the leaf subregions. If we take only a few dozens of samples, the directions of light sources and thus the shadows cast in the renderer do not match the real environment and cause strong artifacts. Furthermore, for subtle movements of light sources, such as a swinging light bulb, the sampling pattern does not change, only the power of each sample changes. The rendered sequence looks unrealistic and diminishes the overall impression of the virtual environment, as the human visual system is sensitive to shadows, providing an important cue about the environment.

Figure 6 shows the sampling patterns produced by each of the three algorithms, and a scene renderer using the respective set of light sources. Note that while *Pharr* and *Hemigon* algorithms produced similar sampling pattern, the *Q2-Tree* algorithm spreads the samples across a broader area because the importance is weighted by angular extent as well as brightness.

Temporal Coherence

An important aspect of importance sampling algorithms for dynamic sequences is the temporal coherence of the sampling pattern for consecutive frames. Frame-to-frame changes of the light source positions and their intensities could cause disturbing temporal flickering in the rendered animation. The images with samples from the three algorithms are shown in Figure 7.

In the implementation of the first algorithm, *Pharr*, we used a Halton generator to generate a sequence of quasi-random vectors in two-dimensional space. Even though we use the same sequence for every frame, positions of samples change globally due to local changes. This problem is caused by the global nature of cumulative probability distribution function.

The second implemented algorithm, *Hemigon*, attempts to solve these issues by several improvements, described in Section 3.1. These processing steps increase the temporal coherence, but introduce visible temporal lag in rendered images for abrupt changes of the illumination. Despite these issues, for a typical scene with subtle frame-to-frame illumination changes, this method produces reasonable results.

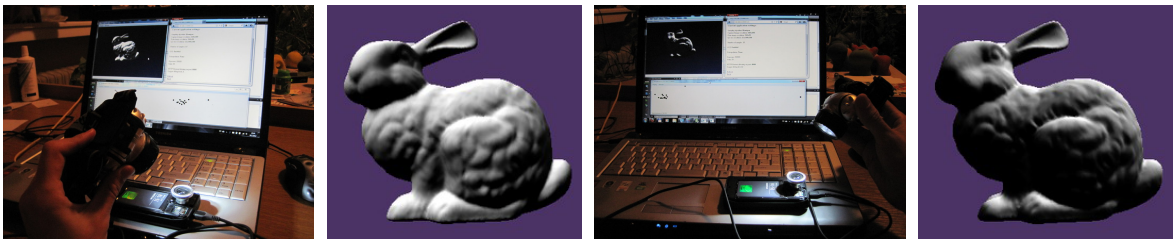
The *Q2-Tree* sampling method handles local illumination changes successfully. On the other hand, it fails to capture subtle light source movements. Furthermore, it requires more samples than the other methods to sample the environment map adequately.

We found that the PDF-based methods produce better results for scenes with subtle illumination changes and with moving light sources. The subdivision methods, on the other hand, produce better results for scenes with abrupt and localized illumination changes.

Performance

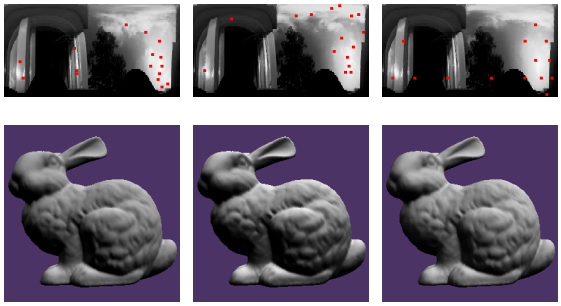
In this section, we analyze the time complexity and present the results of performance measurements of the three implemented algorithms.

The measurements were performed on a Nokia N900 smartphone. The source code was compiled in GCC compiler, version 3.4.4, with the -O3 option enabled.



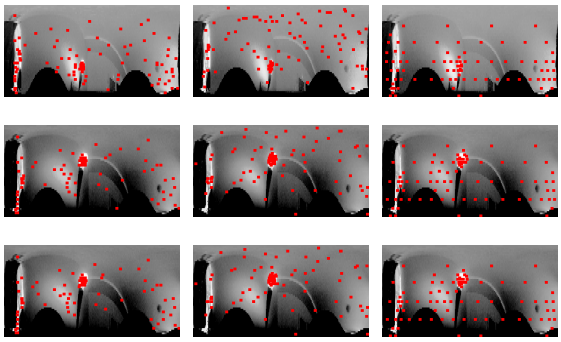
(a) Scenario 1 (photograph) (b) Scenario 1 (screenshot) (c) Scenario 2 (photograph) (d) Scenario 2 (screenshot)

Figure 5: Photographs and rendered images from a testing setup for two scenarios, when the user moves a flashlight around the camera. Changes of the illumination are interactively observed in the renderer; (a) and (b) flashlight on the left, (c) and (d) flashlight on the right.



(a) Pharr (b) Hemigon (c) Q2-Tree

Figure 6: The upper row shows the sampling patterns produced by the three algorithms for an environment map of resolution 360×90 pixels, using 16 samples; the bottom row shows a render of a bunny lit by each respective set of light sources.



Pharr *Hemigon* *Q2-Tree*

Figure 7: Snapshots of the sampling patterns produced by the three implemented sampling algorithms for three subsequent frames of a video sequence.

Three images of varying exposure were fused into one HDR image. The images were captured at resolution 640×480 pixels and mapped to a polar image of resolution one pixel per degree (i.e. 360×90 for a hemisphere). The *Q2-Tree* algorithm used a HEALPix mapping of resolution $12 \times 100 \times 100$ pixels.

The overall asymptotic time complexity is $O(w \cdot h + n \cdot (\log w + \log h))$ for the PDF-based algorithms and $O(w \cdot h + n \cdot \log n)$ for the *Q2-Tree*, where $w \times h$ is the image resolution of the environment map and n is the number of samples. The first term, $O(w \cdot h)$, accounts for the processing of the environment map prior to placement of samples. This includes, for example, the construction of the CDFs in PDF-based sampling algorithms and construction of the summed area table in the *Q2-Tree* sampling algorithm. The second term accounts for placement of samples and depends on the particular algorithm used. Post-processing of samples, including color computation, takes $O(n)$ time.

The bottleneck of the execution is the capture and processing of the HDR light probe. For a single frame, three varying exposure images are captured, fused into a single HDR image [19], and mapped into polar coordinate system. Using the configuration described above, the acquisition of a single HDR environment map takes 120 milliseconds. Table 1 shows the sampling times of the three compared algorithms, excluding the environment map acquisition time.

#samples	HDR acquisition	Sample generation		
		Pharr	Hemigon	Q2-Tree
200	120	10	150	30
1000		10	150	50
10000		80	170	430

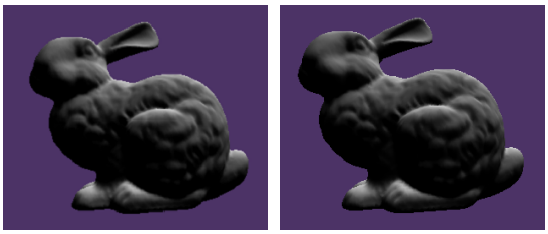
Table 1: Comparison of performance of the three discussed methods. All times are in milliseconds.

6.2 Multiple Cameras

In this section, we present the results of merging of sampling data from multiple cameras. The image renderer with light sources generated by merging the data from four displaced cameras is depicted in Figure 8 together with the reference ground truth image using samples computed from a single camera in the center. Principally the samples as a result of merging from spatially dislocated cameras can be different to the samples computed from the reference camera in dependence on the changes of indoor illumination. However, in practice we observe these differences are very small or zero. This is true only under the assumption, when

the distances between cameras and hence the environment maps are not too different.

(a)



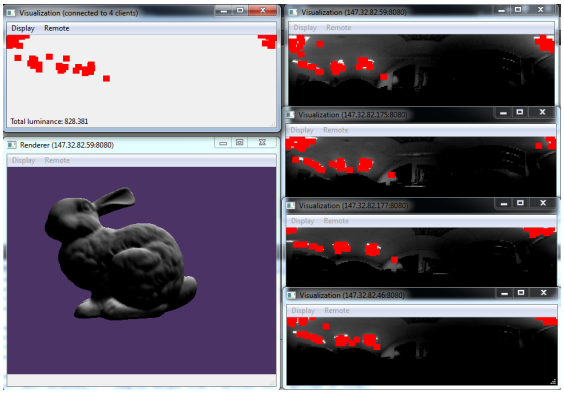
(b) (c)

Figure 8: (a) A photograph of the system in operation. The data from the four displaced cameras are merged and the results are compared against the reference camera in the center. The four outer cameras form a square sized 1000×1000mm. (b) Bunny lit by the illumination acquired on the reference camera in the center. (c) Bunny lit by the illumination computed from the four displaced cameras for *Pharr* algorithm.

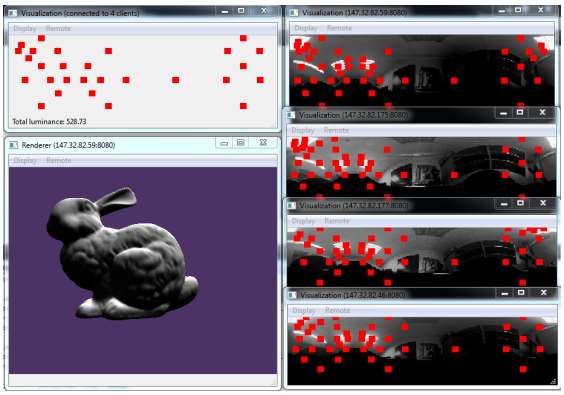
The results of merging for the algorithm *Pharr* and *Q2-tree* are shown in Figure 9. For the *Q*2-Tree sampling our merging algorithm produces the same results as averaging of the calibrated HDR light probes per pixel and building a *Q2-Tree* from scratch on the resulting image.

In order to test this hypothesis empirically, we implemented a simple testing application - it downloads the HDR light probe in .hdr format from each of the cameras, computes an average per pixel and runs the *Q2-Tree* sampling algorithm [23] on the final HDR environment map. We obtained exactly the same set of samples as produced by our *Q2-Tree* merging algorithm. Clearly, the advantage of merging *Q2-Trees* instead of averaging light probes is reduced bandwidth. The *Q2-Tree* representation of a light probe is very compact, typically not exceeding several KBytes for a reasonable number of samples, as opposed to the substantially higher memory requirements consumed by the representation of a complete HDR light probe.

In addition, the system with multiple cameras increases on robustness - it is capable of providing reasonable data to the rendering engine even if one or more of the cameras fail or has data dropout (for example, due to network problems) as long as the failure is detected. Also, the use of multiple asynchronously running cameras increases the frequency of light sources update and the data can be used to create a temporal blur, improving the temporal coherence.



(a) Pharr



(b) Q2-Tree

Figure 9: Results of merging for (a) *Pharr* and (b) *Q2-Tree* algorithm. The four windows on the right show sampling patterns and environment maps captured by four displaced cameras. The windows on the left show the resulting sampling pattern and a scene illuminated by the resulting set of directional light sources.

7 CONCLUSIONS

We presented a framework for live capturing of HDR light probes and their decomposition into sets of directional light sources. We implemented three importance sampling algorithms and compared them in terms of the quality of the sampling pattern, temporal coherence, and performance. We extended the existing techniques by merging data from multiple cameras to better approximate the lighting of the real environment, and discussed the associated advantages and limitations. We have shown a proof of concept implementation of our

ideas. Our framework with multiple cameras is useful for example in mobile setting of virtual television studios and augmented reality display systems.

As future work we want to further examine the possibilities of using multiple cameras. Existing image-based lighting methods assume only distant lighting, represented by directional light sources. This assumption is not valid, especially for indoor scenes, and causes artifacts in mixed reality applications, where the light source position is required for shadow detection and generation. By using two or more displaced cameras, it would be possible to estimate the position of point light sources and thus even improve the results of consistent illumination computation similar to the outdoor scenarios [16].

ACKNOWLEDGEMENTS

This research has been partially supported by the Czech Science Foundation under research program P202/11/1883 (ARGIE) and P202/12/2413 (OPALIS).

8 REFERENCES

[1] A. Adams, D. Jacobs, J. Dolson, et al. The Frankencamera: An Experimental Platform for Computational Photography. *ACM TOG*, 29(4):29, ACM, 2010.

[2] S. Agarwal, R. Ramamoorthi, S. Belongie, and H. Jensen. Structured importance sampling of environment maps. In *ACM TOG*, 22(3):605–612. ACM, 2003.

[3] R. Azuma. *A Survey of Augmented Reality*. Presence: Teleoperators and Virtual Environments, 6(4):355–385, August 1997.

[4] G. Bradski and A. Kaehler. *Learning OpenCV: Computer Vision with the OpenCV Library*. O'Reilly Media, Incorporated, 2008.

[5] D. Burke, A. Ghosh, and W. Heidrich. Bidirectional Importance Sampling for Illumination from Environment Maps. In *ACM SIGGRAPH 2004 Sketches*, pp. 112, ACM, 2004.

[6] F. Crow. Summed-Area Tables for Texture Mapping. *ACM SIGGRAPH Computer Graphics*, 18(3):207–212, ACM, 1984.

[7] P. Debevec. A Median Cut Algorithm for Light Probe Sampling. In *ACM SIGGRAPH 2005 Posters*, pp. 33:1-33:3, ACM, 2005.

[8] G. Fishman. *Monte Carlo: Concepts, Algorithms, and Applications*. Springer, 1996.

[9] S. Gibbs, C. Arapis, C. Breiteneder, V. Lalioti, S. Mostafawy et al. Virtual Studios: An Overview. *In IEEE MultiMedia*, 5(1):18–35, 1998.

[10] K. Górski. HEALPix. *Jet Propulsion Laboratory, California Institute of Technology, NASA*, 2012. `http://healpix.jpl.nasa.gov/`.

[11] V. Havran, K. Dmitriev, and H.-P. Seidel. Goniometric Diagram Mapping for Hemisphere. In *Short Pres. (Eurogr. 2003)*, pp. 293–300, 2003.

[12] V. Havran, M. Smyk, G. Krawczyk, K. Myszkowski, and H. Seidel. Interactive System for Dynamic Scene Lighting Using Captured Video Environment Maps. In *Proc. of EGSR'05*, pp. 43–54, Konstanz, Germany, 2005.

[13] E. Hivon and B. Wandelt. Analysis Issues for Large CMB Data Sets. *Proc. Evolution of Large-Scale Structure*, arXiv:astro-ph/9812350, 1998.

[14] K. Jacobs and C. Loscos. Classification of Illumination Methods for Mixed Reality. In *Computer Graphics Forum*, 25(1):29–52, 2006.

[15] T. Kollig and A. Keller. Efficient Illumination by High Dynamic Range Images. In *Proceedings of EGWR'03*, pp. 45–50, Leuven, Belgium, 2003.

[16] Y. Liu and X. Granier. Online Tracking of Outdoor Lighting Variations for Augmented Reality with Moving Cameras. *IEEE Trans. on Visualization and Computer Graph.*, 18(4):573–580, 2012.

[17] V. Ostromoukhov, C. Donohue, and P. Jodoin. Fast Hierarchical Importance Sampling with Blue Noise Properties. In *ACM TOG (SIGGRAPH)*, 23(3):488–495. ACM, 2004.

[18] M. Pharr and G. Humphreys. Infinite Area Light Source with Importance Sampling (plugin description). On `http://www.pbrt.org`, 2004.

[19] E. Reinhard. *High Dynamic Range Imaging: Acquisition, Display, and Image-Based Lighting*. Morgan Kaufmann, 2006.

[20] D. Schmalstieg, A. Fuhrmann, G. Hesina, Z. Szalavári, L. Encarnaçao, M. Gervautz, and W. Purgathofer. The Studierstube Augmented Reality Project. *Presence: Teleoperators & Virtual Environments*, 11(1):33–54, 2002.

[21] W. Son, B. Nam, T. Kim, and H. Hong. Using Environment Illumination Information in Mobile Augmented Reality. In *IEEE conference on Consumer Electronics 2012*, pp. 588–589, 2012.

[22] L. Wan, S.-K. Mak, T.-T. Wong, and C.-S. Leung. Spatiotemporal Sampling of Dynamic Environment Sequences. *IEEE Trans. on Visualization and Computer Graph.*, 17(10):1499–1509, 2011.

[23] L. Wan, T.-T. Wong, and C.-S. Leung. Spherical Q2-tree for Sampling Dynamic Environment Sequences. In *Proceedings of EGSR'05*, pp. 21–30, Konstanz, Germany, 2005.

[24] J. Wind, K. Riege, and M. Bogen. Spinnstube: A Seated Augmented Reality Display System. In *Proceedings of EGVE'2007*, pp. 17–23, 2007.