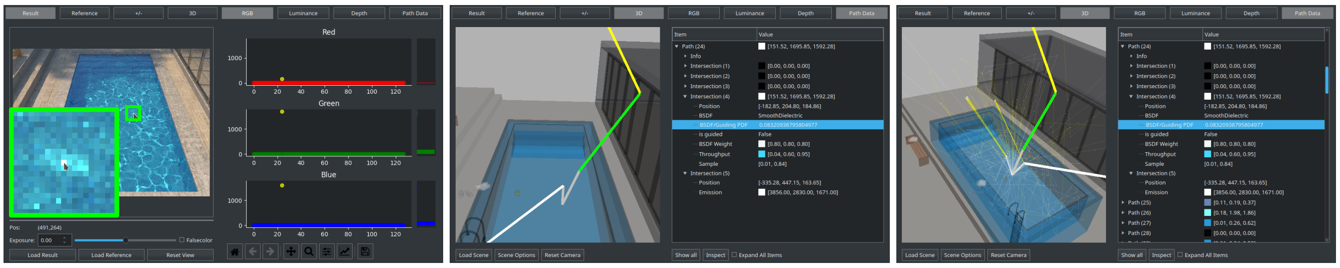
Vision, Modeling, and Visualization (2021)
B. Andres, M. Campen, and M. Sedlmair (Eds.)

EMCA: Explorer of Monte Carlo based Algorithms

L. Ruppert[1†] and C. Kreisl[2†] and N. Blank[3] and S. Herholz[4] and H. P.A. Lensch[1]

[1]University of Tübingen, Germany [2]Robert Bosch GmbH [3]Karlsruhe Institute of Technology, Germany [4]Intel Corporation



```
Item                      Value
▼ Path (24)               [151.52, 1695.85, 1592.28]
   ▸ Info
   ▸ Intersection (1)     [0.00, 0.00, 0.00]
   ▸ Intersection (2)     [0.00, 0.00, 0.00]
   ▸ Intersection (3)     [0.00, 0.00, 0.00]
   ▼ Intersection (4)     [151.52, 1695.85, 1592.28]
       Position           [-182.85, 204.80, 184.86]
       BSDF               SmoothDielectric
       BSDF/Guiding PDF   0.08320938795804977
       is guided          False
       BSDF Weight        [0.80, 0.80, 0.80]
       Throughput         [0.04, 0.60, 0.95]
       Sample             [0.01, 0.84]
   ▼ Intersection (5)
       Position           [-335.28, 447.15, 163.65]
       Emission           [3856.00, 2830.00, 1671.00]
```

```
Item                      Value
▼ Path (24)               [151.52, 1695.85, 1592.28]
   ▸ Info
   ▸ Intersection (1)     [0.00, 0.00, 0.00]
   ▸ Intersection (2)     [0.00, 0.00, 0.00]
   ▸ Intersection (3)     [0.00, 0.00, 0.00]
   ▼ Intersection (4)     [151.52, 1695.85, 1592.28]
       Position           [-182.85, 204.80, 184.86]
       BSDF               SmoothDielectric
       BSDF/Guiding PDF   0.08320938795804977
       is guided          False
       BSDF Weight        [0.80, 0.80, 0.80]
       Throughput         [0.04, 0.60, 0.95]
       Sample             [0.01, 0.84]
   ▼ Intersection (5)
       Position           [-335.28, 447.15, 163.65]
       Emission           [3856.00, 2830.00, 1671.00]
▸ Path (25)               [0.11, 0.19, 0.37]
▸ Path (26)               [0.18, 1.98, 1.86]
▸ Path (27)               [0.01, 0.26, 0.62]
▸ Path (28)               [0.00, 0.00, 0.00]
```

Figure 1: *Exploration of a single high-contribution path using our proposed visualization framework. The path collects a high contribution from the sun while the BSDF of the tinted glass sampled this path with low probability, resulting in a firefly. It stands out compared to all other paths contributing to the pixel.*

Abstract

Debugging or analyzing the performance of global illumination algorithms is a challenging task due to the complex path-scene interaction and numerous places where errors and programming bugs can occur. We present a novel, lightweight visualization tool to aid in the understanding of global illumination and the debugging of rendering frameworks. The tool provides detailed information about intersections and light transport paths. Users can add arbitrary data of their choosing to each intersection, based on their specific demands. Aggregate plots allow users to quickly discover and select outliers for further inspection across the globally linked visualization views. That information is further coupled with 3D visualization of the scene where additional aggregated information on the surfaces can be inspected in false colors. These include 3D heat maps such as the density of intersections as well as more advanced colorings such as a diffuse transport approximation computed from local irradiance samples and diffuse material approximations. The necessary data for the 3D coloring is collected as a side-product of quickly rendering the image at low sample counts without significantly slowing down the rendering process. It requires almost no pre-computation and very little storage compared to point cloud-based approaches. We present several use cases of how novices and advanced rendering researchers can leverage the presented tool to speed up their research.

CCS Concepts

• ***Computing methodologies*** → *Ray tracing;* • ***Human-centered computing*** → *Visualization toolkits; Heat maps;*

1. Introduction

Simulating physically-based light transport is a challenging task where researchers continue to deliver amazing progress in the form of novel, ever more efficient rendering algorithms, e.g. [KGV*20]. Developing these algorithms is complicated by the fact that rendered images are the result of millions of independently traced light transport paths. The computation of these images generally takes minutes to hours or even days to produce the final noise-free result and is heavily parallelized to speed up computation. Tracking down issues with individual components of the rendering framework or interactions thereof only by inspecting the resulting image can seem downright impossible under these circumstances. Similarly, running the renderer in a regular general-purpose debugger is impractical as one would need to carefully step through the millions of light paths one by one while only ever having access to the current state (usually just the current intersection). Additional tooling is required to aid in this process.

A key element is the visualization of light transport paths and associated intersection data. In addition, we propose the use of

† equal contribution

DOI: 10.2312/vmv.20211377

quickly computed, yet detailed 3D heat maps on surfaces to further improve visibility and interpretability of developer-controlled aspects of the high dimensional light transport problem.

In particular, our tool "EMCA" supports path tracing [Kaj86], which remains the primary rendering method in production [FHH*19] due to its generality and simplicity. In path tracing, light transport paths are randomly sampled in a 3D scene. Starting from a sampled location within the pixel's footprint on the camera's sensor, rays are traced recursively to create paths that connect to emitters which then contribute to the computed image. At each intersection, local sampling decisions determine the outgoing ray directions. The choice of those local sampling distributions greatly influences the noise in the resulting image or more precisely the variance of the Monte Carlo estimator [SW92]. To minimize variance, one usually uses multiple importance sampling (MIS) [VG95] to combine sampling strategies that construct high throughput paths by sampling the bidirectional scattering distribution function (BSDF), e.g. [MU12], with sampling strategies that connect directly to light sources, e.g. [SWZ96].

While, at its core, the path tracing algorithm has long since remained the same and is thoroughly covered by standard works [Vea97, PJH16], advances in its key components like BSDF sampling strategies [Hei18] or MIS weights [IVG*19] are made to this day. These components introduce additional complexity and affect the path tracing behavior globally and locally in non-obvious ways. Their particular influence is hard to grasp without visualization tools that provide immediate feedback during development cycles. We demonstrate possible visualizations for these tasks in Figure 2.

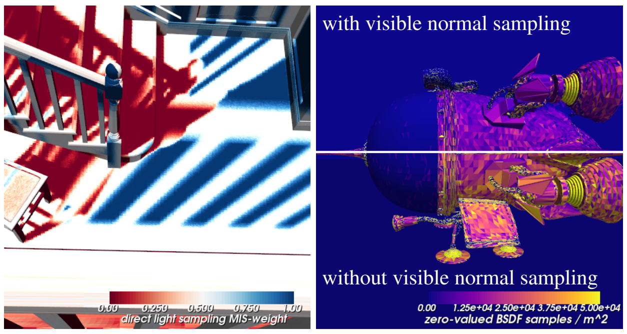

0.00
0.250
0.500
0.750
1.00
direct light sampling MIS-weight
with visible normal sampling
without visible normal sampling
0.00
1.25e+04
2.50e+04
3.75e+04
5.00e+04
zero-valued BSDF samples / m^2

Figure 2: *EMCA can visualize various three-dimensional data such as MIS weights (left, reproducing a result from [IVG*19], shown from a novel viewpoint), or the density of zero-valued samples (right) which increase variance by unintentionally terminating paths early when sampling invalid ray directions below the surface normal.*

EMCA allows for detailed user-controlled data which can be displayed and plotted out of the box. In addition to data per path and intersection, EMCA supports the collection of data on 3D surfaces in the scene which may either be scalar heat maps or arbitrary RGB values. A flexible plugin interface readily supports additional special needs. To ease navigation, we follow the brushing and linking paradigm [EW95, TSWB94] and fully link the individual views of the light transport data via the selected paths and intersections.

While modern hardware and tooling would allow to capture all path data in advance, only a small subset of that data is ever viewed by the user. As one does not know which data the user will look at, existing methods (see Section 2) rely on heavy pre-computation or only provide sparse data to reduce pre-processing times and memory consumption. Instead, we re-compute the necessary visualization data on-demand, where we build on a deterministic, repeatable rendering process. In the rendered image, users can select individual pixels which are then re-computed and data is collected for that pixel only. Only for the 3D visualization, some pre-processing is necessary. For that, we present an efficient method to integrate it at low cost in the anyway required rendering process.

EMCA is publicly available online (`https://github.com/cgtuebingen/emca`). We hope that the focus on fast yet detailed visualization and inspection capabilities encourages developers to use it regularly during development rather than relying on gut feelings, lengthy code reviews, or simplistic logging and quickly assembled hand-crafted tools that lack the vast context and rich visualization capabilities of EMCA. As an example, in Figure 1, the root cause of an outlier path is determined by drilling down from pixel to path and intersection. The detailed user-supplied data identifies the issue as the insufficient sampling probability for the direction towards the sun. Further use-cases are presented in Section 5.

2. Related Work

The visualization of light transport has a long history, including real-world visualization of individual light transport paths in fluorescent fluids [HFA*08].

Digital methods for visualizing light transport paths include the *Ray Tracing Visualization Toolkit* rtVTK [GFE*12] which supports the visualization of individual paths and spatial acceleration structures on top of rendered images or the 3D scene. It supports an immediate mode that displays rays while stepping through the actual rendering process and a read/write mode, where all ray data needs to be written to disk in advance and can then be stepped through later. Its primary use case is education.

The framework by Lesev and Penev [LP14] records, filters, and analyzes massive amounts of ray data. At the core of their approach stands a cloud-based logging API that records detailed information about the entire rendering process including the line numbers in the source code. Before visualization, the recorded data needs to be analyzed in a pre-processing step and multiple acceleration structures are built to facilitate interactivity later on. A major use-case of their approach is to serve as an execution log that allows for precisely tracking down issues after rendering but they also target education.

Simons et al. [SAH*16, SHP*19] apply visual analytics to light transport. Their primary point of interaction is a parallel coordinate plot which allows for the selection of paths based on commonalities in a few coarse quantities such as radiance, throughput, depth, or location. They further specialize in highlighting differences between two similar scene configurations, provide 2D and 3D heat map visualization of basic quantities and the pixel locations of selected paths. All path and intersection data of the entire image need to be pre-computed. For 3D heat maps, the individual intersection point samples need to be converted into a detailed voxel representation.

When it comes to visualizing light transport in a scene rather than individual paths, the tool by Reiner et al. [RKRD12] provides several interactive visualizations with a progressive photon mapper at its core. Their target audience is artists who are interested in the flow of light to understand which adjustments need to be made to achieve the desired target illumination. To that end, the tool renders in false color, visualizes incident radiance via the displacement and coloring of a sphere, visualizes coarse summary light paths for simple transports and entire volumes representing the light paths contributing to a certain effect. They also inject particles into the scene that follow the propagation direction of the light.

For visualizing the flow of light, Kartashova et al. [KPRP19] introduce light shapes: Meshes of arrows, ellipsoids, and tubes that fill the 3D space to indicate the origin of the light.

Going one step further, Schmidt et al. [SNM*13] present a method to manipulate light transport by bending or otherwise retargeting light transport paths that match a set of rules. These rules can be extracted by simply selecting light effects in their visualization tool which simultaneously displays the responsible light transport paths. Their light transport manipulation enables the re-rendering of the image with full global illumination where certain portions of the path space are manipulated to achieve the desired effects.

3. The Explorer of Monte Carlo based Algorithms

We present EMCA, the Explorer of Monte Carlo based Algorithms, a visualization tool aimed at light transport researchers that require quick visualizations as part of their development cycles. It provides an interactive 3D scene with light transport paths and various plots for detailed user-defined data. All necessary data is provided on-demand by a remote renderer via TCP (see Section 4).

For an overview of the GUI, see Figure 1. In the following, we present the various display options and capabilities in detail.

3.1. Image View

In the image view (Figure 3), the high-dynamic range image created by the render is displayed. Our tool provides the basic capabilities one has come to expect from tools like *tev* [Mül17]. Namely, changing exposure, false-color display, setting a reference image, and comparing against said reference in a green/red +/- difference view. The difference view can be the first indicator of bias or outliers which may otherwise be difficult to catch after tone mapping.

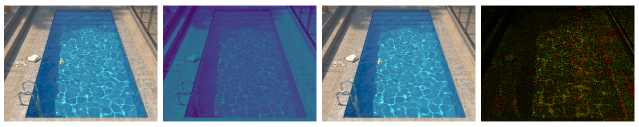


Figure 3: *The image view displays the rendered result image (1), optionally in false color (2). When setting a reference image (3), it can also display the difference between result and reference (4).*

Regardless of the display setting, one can here select pixels of interest for which to retrieve the detailed path and intersection data for further visualization and inspection. Via the pixel history, one can quickly switch between several pixels of interest.

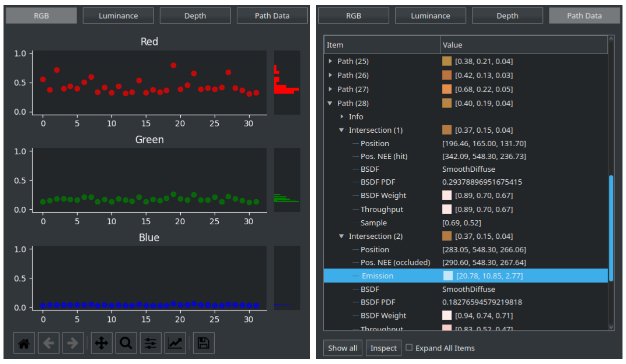
3.2. 3D Visualization

EMCA provides a VTK-based [SML06] 3D view of the entire scene geometry, wherein selected light transport paths are displayed. A rectangular selection tool enables selecting subsets of the visible paths or individual intersections. Optionally, the scene geometry can be colored with additional 3D data which may visualize arbitrary RGB or heat map data (see Section 4.1). By default, scene geometry is displayed at reduced opacity to make paths more visible. For inspecting the 3D data, the opacity can be increased.

Displayed light transport paths are colored to provide additional information. By default, all path segments are white. The segment leading to the current intersection is colored green, segments connected to an emitter are colored yellow, next event estimations (if shown) are blue if successful and red if occluded.

3.3. Inspecting Path Data

Access to detailed path data is the key ingredient in understanding or debugging rendering algorithms. By default, the paths that make up the selected pixel are displayed by their contribution in stacked scatter plots for red, green, and blue (Figure 4, left) where one can further select individual paths for display in the 3D view. Where the distinction by color channel is not relevant, one can instead plot just the luminance. Selecting paths and intersections will highlight them in the 3D view and focus the camera on them. A rectangular selection tool allows for quick extraction of similar paths.



Item	Value
Path (25)	[0.38, 0.21, 0.04]
Path (26)	[0.42, 0.13, 0.03]
Path (27)	[0.68, 0.22, 0.05]
Path (28)	[0.40, 0.19, 0.04]
Info	
Intersection (1)	[0.37, 0.15, 0.04]
Position	[196.46, 165.00, 131.70]
Pos. NEE (hit)	[342.09, 548.30, 236.73]
BSDF	SmoothDiffuse
BSDF PDF	0.29378896951675415
BSDF Weight	[0.89, 0.70, 0.67]
Throughput	[0.89, 0.70, 0.67]
Sample	[0.69, 0.52]
Intersection (2)	[0.37, 0.15, 0.04]
Position	[283.05, 548.30, 266.06]
Pos. NEE (occluded)	[290.60, 548.30, 267.64]
Emission	[20.78, 10.85, 2.77]
BSDF	SmoothDiffuse
BSDF PDF	0.18276594579219818
BSDF Weight	[0.94, 0.74, 0.71]
Throughput	[0.83, 0.52, 0.47]

Figure 4: *Path data can be inspected in summarizing plots or in great detail in a tree view with access to all user-defined values.*

In the "Path Data" view, the entire data of the selected path and its intersections are shown in a tree view (Figure 4, right). Each path and intersection shows the color of the contribution it has gathered up to that point and by that allows one to quickly determine interesting paths and intersections (e.g. those causing fireflies) without having to go through their potentially extensive data individually. The data presented in this setting is completely user-controlled by the data set within the renderer and may thus contain as much or as little data as deemed necessary for the task at hand.

Paths can also be plotted by their depth, where paths with similar depth often show similar behavior, e.g. Figure 5. The path depth may also expose issues that lead to premature path termination, e.g. due to too aggressive Russian roulette, or, on the flip side, paths that reach maximum depth, e.g., as a result of total internal reflection.

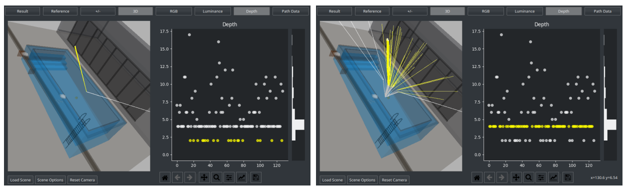
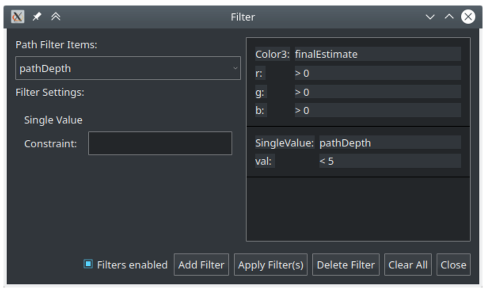


Figure 5: *Path depth often correlates with their behavior in 3D space. Here, one can quickly select all paths that have been reflected immediately by the water surface and thus almost pointing into the same direction, as well as those that bounced on the pool floor once before reaching the sky.*

3.4. Filtering

With the included filtering tool (Figure 6), one can automatically select a subset of paths through a pixel based on path-dependent criteria. For instance, one might be interested in displaying only paths with a non-zero contribution or vice versa to visually check for similarities. Or, one might be interested only in those paths intersecting an object with a specific material one is currently implementing. Similarly, the outlier detector automatically selects firefly paths based on their values after selecting a pixel in the image.


Filter

Path Filter Items: pathDepth

Filter Settings:

Single Value

Constraint:

Color3: finalEstimate
r: > 0
g: > 0
b: > 0

SingleValue: pathDepth
val: < 5

Figure 6: *Filtering allows one to automatically select paths based on a simple set of rules.*

3.5. Inspecting Intersection Data

When the context between the intersections of a single path matters, the "Path Data" view can become tedious to use. Therefore, EMCA provides the "Intersection Data" view (Figure 7) which provides helpful scatter plots for all user-supplied intersection data in the selected path. Here, one can quickly see how the path throughput develops with each intersection, one can pinpoint intersections with extreme values in the BSDF, the sampling PDF, emitted radiance, and many others. Not only scalar data can be plotted over the path depth, but of course also RGB-valued, 2D, and textual data. Each plot is accompanied by a simple histogram for cases where one expects a certain distribution and wishes to quickly check for violations of that assumption.

For 2D data, such as random samples, a 2D scatter plot is used with a histogram on both axes which contains the values of all intersections. While one could display this data in a 3D plot with the path depth values, we found it difficult to extract precise values or concrete meaning from loose points in a 3D projection. In 2D, while missing the depth information, one can at least check for obvious issues such as extreme correlations between the raw 2D samples that are used across intersections.

Again, this view is linked to the global state of selected paths and intersections. Selecting an intersection in the plot will directly focus the 3D and "Path Data" views on the selected intersection.

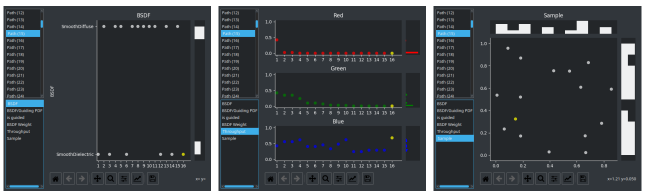


Figure 7: *The intersection data view provides datatype-specific plots of all user-supplied quantities within a path.*

3.5.1. Light Probe

Additional context about the global illumination helps to judge local sampling decisions. The "Light Probe" view (Figure 8), when activated, asks the renderer to render a small image with a spherical camera at the current intersection point. As additional context, the incident and outgoing ray directions are overlaid on that image, as well as the direction of the next event estimation, if available. While the cylindrical projection of the spherical camera image may need some getting used to, this view makes it easy to judge sampling decisions by displaying outgoing ray directions in the context of the incident radiance.

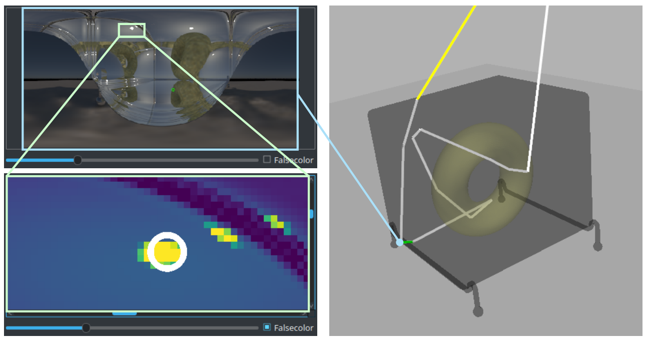


Figure 8: *The light probe view (left) visualizes the incident radiance at a selected intersection point in the 3D scene view (right). While the complete incident radiance on the top shows the sheer complexity of this illumination scenario, the zoomed-in false color view explains why this path results in a firefly: The outgoing ray direction leads right into one of the many reflections of the sun.*

When encountering an outlier, this view can be used to pinpoint not only its immediate origin but also its context, i.e., shape and appearance of the emitting or reflecting object, even through chains of specular or near-specular reflections or refractions that otherwise complicate the visualization. It is a valuable tool for exploring small but important features in complex light transport situations.

3.6. Plugins

EMCA's visualization client is implemented in Python, which makes it particularly easy to add additional features such as plugins. The existing plugin API provides full access to all path and intersection data. Furthermore, plugins are linked with all other views, i.e., they are notified of path and intersection selections and can also select paths and intersections themselves. In terms of display capabilities, they are free to place arbitrary display elements in a Qt widget and may also request additional plugin-specific data from the renderer. In fact, the "Intersection Data" and "Light Probe" views are both implemented as plugins.

4. Integration into the Rendering Framework

The integration into existing rendering frameworks is made easy by bundling the entire data collection mechanism into a single C++ library. One only needs to implement a small wrapper that will on request render an image or a single pixel and which exposes the 3D meshes and camera projection required for visualization.

All data necessary for visualization is transmitted to the client via TCP. This also enables connections to large scale remote renderers while the lightweight visualization data can still be inspected locally without any delay or compression artifacts.

Path Data Collection. To record the actual path and intersection data, the path tracer itself needs to interface with the library as well. We show an example for a simplistic path tracer in Algorithm 1.

First, one needs to set the current path and intersection indices. Afterwards, special functions record data for specific visualizations such as intersection points (including camera position and next event estimation targets) as well as intermediate and final radiance estimates. Arbitrary data can be recorded per path and intersection using general-purpose type-agnostic functions.

Since a single pixel only represents about one millionth of a regular image, we can afford a fairly simple data collection mechanism. All paths of the pixel are computed sequentially by a single thread, where data is gathered in a vector of paths which themselves contain a vector of intersections. Paths and intersections contain, in addition to the core data, vectors of pairs of strings and a union of the supported data types (scalars, points, vectors, colors, strings). To allow the same code to be used for efficient rendering, data collection can be disabled and all functions return immediately.

Determinism. At the core of our lightweight design stands a deterministic rendering process. As long as everything is repeatable, we do not have to store all path data for the entire image in advance but can instead selectively replay and record only the relevant events.

Many renderers already support determinism for debugging and reproduction purposes. It is easily achieved by deterministically seeding the pseudo-random number generators used to sample the light transport paths for each pixel or by using one of the many samplers that operate on fixed low-discrepancy sampling patterns for Quasi Monte Carlo rendering, e.g. [KK02]. For those, one instead needs to deterministically control their decorrelation mechanisms.

```
import emca
foreach pixel(x,y) in the image do
    L = Color(0)
    for sample=1 to samples per pixel do
        emca->setPathIndex(sample)
        ray = sample camera ray for pixel (x, y)
        emca->setPathOrigin(ray.origin)
        L += trace(ray)
    end
    L /= sample count
    pixel(x, y).setColor(L)
end
function trace(ray):
    L = Color(0), T = Color(1)
    for bounce=1 to max bounces do
        its = scene.intersect(ray)
        if no intersection then
            emca->addPathData("left scene", true)
            break
        end
        emca->setDepthIndex(bounce)
        emca->setIntersectionPos(its.pos)
        L += T · $L_e(\dots)$
        $\omega_o$ = sample outgoing ray direction
        $p$ = PDF of outgoing ray direction
        emca->addIntersectionData("PDF", $p$)
        T *= $f_s(\dots) \cdot \cos\theta_o / p$
    end
    emca->setFinalEstimate(L)
    return L
end function
```

Algorithm 1: Integration of the server-side library into a simplistic path tracer. Note that the existing code does not need to be modified. Only additional instructions for data collection need to be inserted. See Section 4 for details.

4.1. 3D Data Collection

The data for 3D visualization is collected per face on dynamically subdivided meshes while rendering the image. Generally, low sample counts (e.g., 32spp) suffice, as each intersection of each path can provide a sample. As a consequence, computing the resulting image and the 3D data usually just takes around 30 seconds.

As the resolution of the captured data depends on the mesh resolution, dynamic mesh subdivision is necessary during data collection to adapt to the spatially varying density of the sample data. Once a face has gathered several hundred samples, we apply a simple 1 to 4 subdivision, where the center face is constructed from the midpoints of the face's edges. Newly created midpoints are referenced in a hash map and reused for subdivisions in neighboring faces. One mutex per mesh is used to prevent race conditions when performing the subdivision. As subdivisions become rarer and rarer with each performed subdivision, this does not hinder performance.

As part of the path tracing process, the renderer already determines the face id in the initial mesh at each intersection. When collecting sample data, the face in the subdivided mesh is efficiently determined by recursively checking if there is a subdivision and then evaluating the barycentric coordinates only for the center face to determine the intersected child face.

We use atomics to update the individual face values as an incremental mean [Fin09] (or a simple sum, when computing densities), making data collection almost entirely lock-free, as to not hinder the parallelism of the rendering process. To prevent dynamic re-allocations, we pre-allocate for the maximum allowed subdivision count. Should the pre-allocated storage run out, the collected data gracefully degrades to a low-resolution approximation.

We perform a short post-processing step, where subdivided faces contribute their values to their child faces proportionally to their sample density. Passing the value to the child faces immediately upon subdivision would propagate low-resolution data to all child faces, including those which do not receive any further samples and would thus introduce aliasing issues. Another issue with collecting samples on mesh faces can be that the initial mesh might be too detailed. We propagate data from neighboring faces to faces which did not receive any samples during data collection to fill gaps.

Compared to collecting point cloud data, this approach has several benefits: Memory consumption is generally much lower and limited from the start. Data collection is fully parallel without requiring a final merging step. Finally, the data can immediately be rasterized efficiently on GPUs without costly conversions.

5. Use-Cases

EMCA supports a wide range of use-cases and can be used by novices as well as advanced rendering engineers. In some cases, it may also provide useful information for artists tasked with modeling, texturing, and lighting detailed 3D scenes.

Starting with basic tasks, one can check the image for outliers or bias in the image view (Section 3.1). One can follow paths through 3D space in the 3D view (Section 3.2), where one can visually check for issues with intersection routines or mentally re-trace and re-compute the path's image contribution to understand exactly how the path's total contribution came to be. In the path data view (Section 3.3), all user-supplied data can be inspected. This allows, for instance, to understand the deficiencies of existing path tracing algorithms that lead to fireflies, e.g. Figures 1 and 8.

For fireflies to occur, a path needs to have a high throughput when intersecting a strong emitter. The throughput is the product of all reflection terms divided by the probability of constructing the given path. When a path with very low probability encounters an emitter, it will most likely cause a firefly. In the intersection data view (Section 3.5), one can see for instance how the throughput develops as the path progresses through the scene. A low-probability sampling decision will result in a sudden, drastic increase in throughput. The responsible intersection can immediately be selected here, presenting the user with the location in 3D space and the user-defined path data, which may include the intersected material type responsible for making that sampling decision.

The light probe (Section 3.5.1) shows the incident radiance one can expect from each direction and by that gives an idea about which directions need to be sampled more often to reduce fireflies.

Aggregate data over all intersection points can be inspected in the form of 3D data collected on mesh surfaces (Section 4.1), this ranges from simple heat maps where one can display intersection densities – optionally filtered by certain conditions such as path termination as the result of a bad sampling decision – or other spatial quantities such as MIS weights, e.g., Figure 2. Even RGB-valued quantities can easily be inspected in scene space, e.g., Figure 10.

Understanding Image Errors. In combination, high-level tasks can be tackled, such as determining the cause of errors in the rendered image. Path tracers consist of many individual components that need to correctly work together. Image artifacts can be caused by issues in individual components as well as by components not interfacing correctly. Especially for novices, it can be quite challenging to figure out where the error lies, as the list of potential reasons is large, e.g., the camera points the wrong way, samples are warped incorrectly, the emission is not set by the emitter, the outgoing ray points into the wrong direction (e.g. using world space instead of local space), terms are evaluated using the primary camera ray's direction rather than the current ray's direction, self-intersection due to missing minimal ray distance $\varepsilon > 0$, etc.. All these problems can quickly be detected with the provided tools.

As a concrete example, we show how improperly re-using a sample to select both a triangle in a mesh and a point on the selected triangle can distort direct illumination calculations in Figure 9.

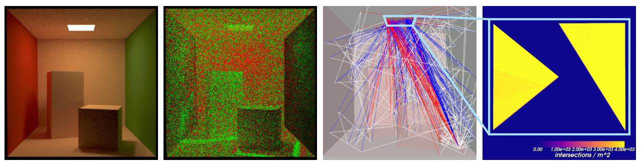


Figure 9: *This rendered image of the Cornell box (1) might appear correct on first glance. However, the difference to the reference (2) shows strong errors near the edges of shadows. Inspecting the paths of an erroneous pixel (3) hints at a non-uniform distribution of samples on the emitter based on the next event estimation path segments, but is too cluttered to be conclusive. Visualizing the density of direct light sampling points (4) confirms this suspicion.*

Understanding global illumination and image contributions. In complex scenes, it can be difficult to judge the global illumination that results from a specific scene configuration. EMCA can approximate global illumination in the 3D scene by visualizing the mean reflected radiance (Figure 10, left). Non-trivial local light transport effects can easily be explored in the interactive 3D scene and provide additional context when inspecting light transport paths.

When further multiplied with the throughput of the camera path up to each point (Figure 10, right), one can judge how much different parts of the scene contribute to the rendered image for the given camera configuration. Each location's brightness corresponds to the average contribution still made to the rendered image when continuing the path. This information can be useful to artists who are tasked with adding detail to a scene with a known camera configuration. Scene parts which do not contribute much to the image even through multiple bounces of light transport can simply remain at a low level of detail in terms of textures and geometry without noticeable differences to the rendered image compared to a fully detailed 3D scene. Similarly, rendering frameworks may want to spatially adapt the level of detail in a scene for efficiency reasons.

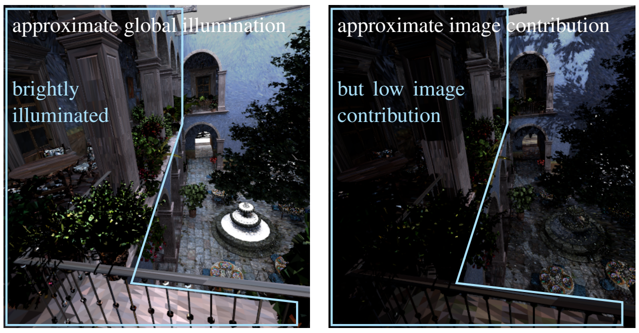

approximate global illumination
brightly illuminated
approximate image contribution
but low image contribution

Figure 10: *Mean reflected radiance approximates global illumination at low cost (left). With additional weighting by the throughput (right), contributing parts of the scene appear brighter, while parts not relevant for the given camera configuration appear darker.*

Understanding the sampling quality of an integrator. Particularly challenging for path tracers are scene configurations as seen in Figure 11. The camera points at a scene which is only illuminated by an area light hidden behind an ajar door. Since the path tracer does not know how to efficiently construct paths through the gap, it primarily explores the room on the front, resulting in a fairly noisy image. Instead, with a guided path tracer, local sampling decisions are made such that paths are more likely to reach the room on the back, where they can collect a contribution on the emitter. Still, the guided path tracer seems to misguide many samples to intersect with the dark front-facing side of the door. Where previously this issue did only manifest itself in higher levels of noise and may have gone unnoticed, with the help of this visualization, further work can be done to improve the sampling distributions.

6. Discussion

While fast and lightweight, EMCA lacks the capability of simultaneously displaying path data from different pixels. While collecting data for multiple pixels would be possible, the required computation grows linearly with the number of pixels to inspect and one would need additional filtering capabilities to deal with the growing set of light transport paths.

Rendering frameworks that do not compute pixels independently or parallelize within pixels, such as wavefront path tracers commonly used by GPU path tracers, require custom data collection mechanisms to be efficient, which we leave for future work.

We measured the render times without computing 3D heat map data and with computing heat map data. Even in the worst case of

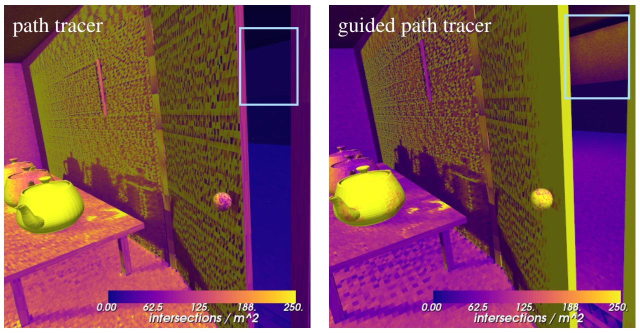


Figure 11: *Comparing the intersection density of path tracing (left) and guided path tracing (right), one can see that only the guided path tracer has a high intersection density on the emitter hidden behind the door. Many intersections also result from the primary camera rays and reflections in the glass teapot closest to the door.*

adding a sample to the heat map at each intersection, render times increased only by about 50% and remained still within seconds.

We coarsely compare the features provided by EMCA against previous methods in Table 1. Additionally, each tool comes with some unique features. Those include BVH visualization [GFE*12], large-scale data logging and filtering [LP14], scene comparison [SHP*19], particle flow visualization [RKRD12], and light transport classification and manipulation [SNM*13]. What makes EMCA unique is its minimal pre-computation, detailed access to user-defined intersection data, and its versatile set of included tools.

Table 1: *Feature comparison of different visualization methods capable of visualizing individual light transport paths.*

Capability	[GFE*12]	[LP14]	[SHP*19]	[RKRD12]	[SNM*13]	EMCA
light transport paths	✓	✓	✓	✓	✓	✓
path filtering	✗	✓	(✓)	(✓)	✓	✓
user data visualization	(✗)	(✗)	(✗)	✗	✗	✓
online data collection	(✗)	(✗)	✗	✓	✗	✓
fast 3D heat maps	✗	✗	(✓)	✗	✗	✓

7. Conclusion and Future Work

We presented a novel visualization tool "EMCA" that helps rendering researchers in developing new, advanced light transport algorithms. Similarly, novices can use the tool to quickly pinpoint issues in their implementation and gain further insight into the light transport process as a whole.

Supported visualizations include a 3D scene view that displays light transport paths as well as heat maps or arbitrary RGB-colored quantities on mesh surfaces. The result image can be viewed in false color and compared to a reference image without resorting to external tools. Individual paths can be inspected in great detail by providing full access to all user-defined data of all light transport paths that make up the entire image without having to pre-compute any path data. An overview of path and intersection data is provided in the form of scatter plots. A light probe provides additional context when inspecting intersection points. A filter tool, as well as the individual and rectangular selection of points in scatter plots as well as paths and intersections in 3D space, allow for quickly selecting important light transport paths. All views are linked and the selection of paths and intersections in one view will update all other views and focus the 3D camera on the selected intersection.

We presented a highly efficient way of collecting detailed 3D data on mesh surfaces for quick visualization that can be transparently included into the highly parallel rendering process without severely impacting its performance. It uses an adjustable but fixed amount of memory and its results can be efficiently rasterized without any further processing.

We presented several use-cases for novices and advanced rendering researchers alike, which can use the presented tool to discover and investigate issues, as well as track their progress at fixing them.

Interesting avenues for future work include the visualization of arbitrary additional mesh data such such as spatial acceleration structures or various other spatial caching structures, as well as proper visualization of volumetric data. A more challenging problem is to add support for rendering algorithms where paths may contribute to arbitrary pixels, making them more difficult to select on a per-pixel basis. This includes light tracers, bidirectional path tracers, and Markov Chain Monte Carlo approaches.

Acknowledgments

This work has been partially funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy – EXC number 2064/1 – Project number 390727645 and SFB 1233, TP 02 – Project number 276693517. It was supported by the German Federal Ministry of Education and Research (BMBF): Tübingen AI Center, FKZ: 01IS18039A.

References

[EW95] EICK S. G., WILLS G. J.: High interaction graphics. *European Journal of Operational Research 81*, 3 (1995), 445–459. doi:10.1016/0377-2217(94)00188-I. 2

[FHH*19] FASCIONE L., HANIKA J., HECKENBERG D., KULLA C., DROSKE M., SCHWARZHAUPT J.: Path tracing in production: Part 1: Modern path tracing. In *ACM SIGGRAPH 2019 Courses* (New York, NY, USA, 2019), SIGGRAPH '19, Association for Computing Machinery. doi:10.1145/3305366.3328079. 2

[Fin09] FINCH T.: Incremental calculation of weighted mean and variance. *University of Cambridge 4*, 11-5 (2009), 41–42. 6

[GFE*12] GRIBBLE C., FISHER J., EBY D., QUIGLEY E., LUDWIG G.: Ray tracing visualization toolkit. In *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games* (New York, NY, USA, 2012), I3D '12, Association for Computing Machinery, p. 71–78. doi:10.1145/2159616.2159628. 2, 7

[Hei18] HEITZ E.: Sampling the ggx distribution of visible normals. *Journal of Computer Graphics Techniques Vol 7*, 4 (2018). 2

[HFA*08] HULLIN M. B., FUCHS M., AJDIN B., IHRKE I., SEIDEL H.-P., LENSCH H. P.: Direct visualization of real-world light transport. In *Proceedings of Vision, Modeling, and Visualization (VMV)* (2008), Citeseer, pp. 363–371. 2

[IVG*19] IVO K., VÉVODA P., GRITTMANN P., SKRIVAN T., SLUSALLEK P., KRIVANEK J.: Optimal multiple importance sampling. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2019) 38*, 4 (July 2019), 37:1–37:14. doi:10.1145/3306346.3323009. 2

[Kaj86] KAJIYA J. T.: The rendering equation. In *Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1986), SIGGRAPH '86, Association for Computing Machinery, p. 143–150. doi:10.1145/15922.15902. 2

[KGV*20] KELLER A., GRITTMANN P., VORBA J., GEORGIEV I., ŠIK M., D'EON E., GAUTRON P., VÉVODA P., KONDAPANENI I.: Advances in monte carlo rendering: The legacy of jaroslav křivánek. In *ACM SIGGRAPH 2020 Courses* (New York, NY, USA, 2020), SIGGRAPH 2020, Association for Computing Machinery. doi:10.1145/3388769.3407458. 1

[KK02] KOLLIG T., KELLER A.: Efficient multidimensional sampling. *Computer Graphics Forum 21*, 3 (2002), 557–563. doi:10.1111/1467-8659.00706. 5

[KPRP19] KARTASHOVA T., PAS S. F. T., RIDDER H. D., PONT S. C.: Light shapes: Perception-based visualizations of the global light transport. *ACM Trans. Appl. Percept. 16*, 1 (Jan. 2019). doi:10.1145/3232851. 3

[LP14] LESEV H., PENEV A.: A framework for visual dynamic analysis of ray tracing algorithms. *Cybernetics and Information Technologies 14*, 2 (2014), 38–49. doi:10.2478/cait-2014-0018. 2, 7

[MU12] MONTES R., UREÑA C.: An overview of brdf models. *University of Grenada, Technical Report LSI-2012-001* (2012). 2

[Mül17] MÜLLER T.: tev – the exr viewer, 2017. URL: https://github.com/Tom94/tev. 3

[PJH16] PHARR M., JAKOB W., HUMPHREYS G.: *Physically based rendering: From theory to implementation*. Morgan Kaufmann, 2016. 2

[RKRD12] REINER T., KAPLANYAN A., REINHARD M., DACHSBACHER C.: Selective inspection and interactive visualization of light transport in virtual scenes. *Computer Graphics Forum 31*, 2pt3 (2012), 711–718. doi:10.1111/j.1467-8659.2012.03050.x. 3, 7

[SAH*16] SIMONS G., AMENT M., HERHOLZ S., DACHSBACHER C., EISEMANN M., EISEMANN E.: An Interactive Information Visualization Approach to Physically-Based Rendering. In *Vision, Modeling & Visualization* (2016), Hullin M., Stamminger M., Weinkauf T., (Eds.), The Eurographics Association. doi:10.2312/vmv.20161356. 2

[SHP*19] SIMONS G., HERHOLZ S., PETITJEAN V., RAPP T., AMENT M., LENSCH H., DACHSBACHER C., EISEMANN M., EISEMANN E.: Applying visual analytics to physically based rendering. *Computer Graphics Forum 38*, 1 (2019), 197–208. doi:10.1111/cgf.13452. 2, 7

[SML06] SCHROEDER W., MARTIN K., LORENSEN B.: *The Visualization Toolkit*, 4th ed. Kitware, 2006. 3

[SNM*13] SCHMIDT T.-W., NOVÁK J., MENG J., KAPLANYAN A. S., REINER T., NOWROUZEZAHRAI D., DACHSBACHER C.: Path-space manipulation of physically-based light transport. *ACM Trans. Graph. 32*, 4 (July 2013). doi:10.1145/2461912.2461980. 3, 7

[SW92] SHIRLEY P., WANG C.: Distribution ray tracing: Theory and practice. In *Proceedings of the Third Eurographics Workshop on Rendering* (1992), Citeseer, pp. 200–209. 2

[SWZ96] SHIRLEY P., WANG C., ZIMMERMAN K.: Monte carlo techniques for direct lighting calculations. *ACM Transactions on Graphics (TOG) 15*, 1 (1996), 1–36. 2

[TSWB94] TWEEDIE L., SPENCE B., WILLIAMS D., BHOGAL R.: The attribute explorer. In *Conference companion on Human factors in computing systems* (1994), pp. 435–436. 2

[Vea97] VEACH E.: *Robust monte carlo methods for light transport simulation*. No. 1610. Stanford University PhD thesis, 1997. 2

[VG95] VEACH E., GUIBAS L. J.: Optimally combining sampling techniques for Monte Carlo rendering. In *Annual Conference Series (Proceedings of SIGGRAPH)* (Aug. 1995), vol. 29, Association for Computing Machinery, pp. 419–428. doi:10/d7b6n4. 2