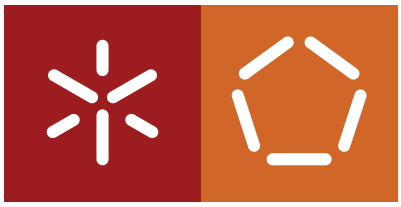
Universidade do Minho
Escola de Engenharia
Departamento de Informática

César Morais Perdigão

Advanced Light Transport Algorithms on Heterogeneous Platforms: Evaluation of the DICE Framework

September 2015



Universidade do Minho
Escola de Engenharia
Departamento de Informática

César Morais Perdigão

Advanced Light Transport Algorithms on Heterogeneous Platforms: Evaluation of the DICE Framework

Dissertação de Mestrado
Mestrado em Engenharia Informática

Trabalho realizado sob a orientação de
Luís Paulo Peixoto dos Santos
Alberto José G. de C. Proença

September 2015

AGRADECIMENTOS

Ao meu orientador, Luís Paulo Santos, pela excelente orientação, disponibilidade e pelo rigor exigido na análise do trabalho realizado. Ao meu co-orientador, Alberto Proença, pelos desafios apresentados, pelas críticas construtivas que fez ao longo to trabalho.

Aos colegas do LabCG, pelo ambiente de trabalho familiar que proporcionaram, que foi uma grande ajuda no desenvolvimento deste trabalho.

À Regina, por estar sempre ao meu lado a apoiar-me e pelo suporte incondicional que tem sido para mim ao longo destes anos.

À Helena por ser sempre a melhor amiga que poderia ter.

À família da residência, que me deu uma segunda casa ao longo destes cinco anos, em especial ao Pedro Maia, ao Luís Caseiro, Francisco Neves e ao João Pinto.

Por fim, um agradecimento especial à minha mãe, pelos sacrifícios, pelo apoio incondicional que tornaram possível tudo isto.

Work funded by the Portuguese agency FCT, *Fundação para a Ciência e Tecnologia*, under the program UT Austin — Portugal.

ABSTRACT

Efficient physically based rendering is still an actual challenge due to both the wide variety of available algorithms and their heavy computational demands. The performance of such techniques can be enhanced by combining two complementary approaches: using the most accurate and robust light transport algorithms and an efficient use of all available computing resources. Although most modern computers have both powerful multicore CPU devices coupled to manycore GPU devices, most current implementations of light transport algorithms only address one of these types of computing units and thus not using all available resources efficiently. This dissertation work addresses these efficiency issues by evaluating the most recent global illumination algorithms based on path space integration, and evaluating a framework that automatically takes full advantage of an heterogeneous computing system, based on CPU and GPU devices: the DICE framework. The outcomes from our scalability test on such heterogeneous computing platforms show promising results and a path worth pursuing.

RESUMO

A síntese de imagens fotorrealistas eficiente é ainda um desafio real devido à variedade de algoritmos disponíveis, bem como às exigências computacionais destes tipos de algoritmos. O desempenho destas técnicas pode ser melhorado usando duas abordagens complementares: utilizar os algoritmos de transporte de luz mais eficientes e robustos bem como usando todos os recursos computacionais da melhor forma possível. Apesar da maioria dos computadores modernos ter disponível poderosos CPU's *multicore* bem como GPU's *manycore*, a maioria das implementações apenas utiliza um dos tipos de unidades computacionais, não usando eficientemente todos os recursos disponíveis. Este trabalho tem como alvo estas falhas de eficiência avaliando os mais recentes algoritmos de iluminação global baseados em *path space integration*, e avaliando a uma *framework* que automaticamente tira total partido de sistemas computacionais heterogéneos: a *framework* DICE. Os resultados de escalabilidade neste tipo de plataformas heterogéneas mostram resultados promissores e um caminho que vale a pena explorar.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

BPM	Bidirectional Photon Mapping
BPT	Bidirectional Path Tracing
CPU	Central Processing Unit
GPU	Graphics Processing Unit
HT	Hyper Threading
MCMC	Markov Chain Monte Carlo
MIC	Many Integrated Core
MIS	Multiple Importance Sampling
NUMA	Non Uniform Memory Access
PM	Photon Mapping
PPM	Progressive Photon Mapping
PT	Path Tracing
RMSE	Root Mean Squared Error
SIMD	Single Instruction Multiple Data
VCM	Vertex Connection and Merging

1

INTRODUCTION

1.1 CONTEXT

One of the main challenges in computer graphics is physically based rendering, the synthetic creation of images that are perceptually indistinguishable from real world views based on a geometric description of the scene, materials and light sources. There are several algorithms that try to solve this problem, although none of them is yet robust enough to handle every possible situation.

One of the first solutions, Path Tracing (PT) proposed by Kajiya (1986), aims to solve this problem by tracing light transport paths starting from the camera until a light source is hit.

One improvement upon this algorithm was Bidirectional Path Tracing (BPT), developed independently by Lafortune and Willems (1993) and Veach (1998). Although with different mathematical background, the goal is to sample more light transport paths by connecting sub-paths generated from the camera and the light source. This allows for a much more efficient rendering of effects like caustics, although effects like reflected caustics are still too difficult for a bidirectional path tracer to handle robustly.

One completely different approach developed by Jensen (1996) was instead of trying to find paths from the light source to the camera to just trace a packets of photons throughout the scene and store them in an acceleration structure referred to as Photon Map. In a second pass, the rays would start from the camera and consult the photon map in the vicinity of the intersections and calculate the expected radiance through a density estimation. Unlike all the previously presented methods, photon mapping introduces bias, that is, it may not converge to the correct result of the rendering equation. However, it is consistent, and by diminishing the search radius on the photon map and increasing the number of traced photons, the bias reduces to zero in the limit (Hachisuka et al., 2008).

Most recently, an attempt to combine these two approaches was proposed by Georgiev et al. (2012). In this algorithm, Vertex Connection and Merging (VCM), photon mapping and bidirectional path tracing are combined, taking advantage of each of the algorithms strong points: the high convergence rate from bidirectional path tracing and the better handling of caustics from photon mapping. These two algorithms are combined by reducing photon mapping to a path sampling technique in the path integration space formulation and combines it with bidirectional path tracing using Multiple Importance Sampling (MIS).

In spite of the differences between the different techniques, what they all have in common is the need for computational power, since all of these algorithms are based on ray tracing, a computationally expensive operation.

Although most computers nowadays contain both multicore chips with multiple Central Processing Unit (CPU) and a Graphics Processing Unit (GPU), most commonly seen implementations only use one of these types of computational devices, and so waste useful computational power. Developing for these heterogeneous platforms, such that all different devices are effectively and efficiently used by the application, raises a number of challenges. In fact, besides all issues commonly associated with parallel processing (e.g., workload decomposition, communication overheads, load balancing), the heterogeneity of the different devices often implies maintaining different implementations for each architecture as well as dealing with separate memory address spaces. Frameworks, such as StarPU (Augonnet et al., 2011) and DICE (Barbosa et al., 2015) have been proposed to alleviate the application programmer from the challenges risen by heterogeneous systems.

1.2 THE PROBLEM AND ITS CHALLENGES

Light transport algorithms are computationally demanding, with solutions resorting to (homogeneous) parallel computing being common. It is, however, reasonable to assume that many of these algorithms have the potential to exhibit significant efficiency levels on heterogeneous computing platforms.

With this, the main goals for this project are to evaluate the suitability of the above described path integration space rendering algorithms for parallel heterogeneous systems, such as those commonly found on current desktops, i.e., multicore CPU and a GPU. The DICE framework has been selected for this evaluation, because it has been developed in close cooperation with our research group and we are interested on identifying its strong and weak points, regarding both performance and usability.

One of the main challenges present when addressing heterogeneous systems is the management of multiple implementations, one for each architecture. In order to minimize this difficulty, both implementations use most of the basic structures and use specialized ray tracing frameworks, which simplifies development. Another important aspect to take into account is to minimize communication across devices, which can be costly.

Other problem specific challenges is load balancing, which is an issue with Monte Carlo methods because the workloads may be highly irregular, issue that should be addressed by the DICE scheduler.

1.3 STRUCTURE OF THE DISSERTATION

This dissertation is structured in four chapters, being the first the introduction where the problem and the main goals of the project are presented, as well as the structure of the dissertation.

The second chapter presents a background in computation light transport, namely a review of various light transport algorithms and a review on heterogeneous systems programming.

The third chapter presents the experimental work, explaining the methodologies used, the results obtained and finalizing with a discussion of those results.

The fourth and final chapter contains the conclusions, as well as the limitations of this study and possible future work.

2

BACKGROUND ON COMPUTATIONAL LIGHT TRANSPORT AND HETEROGENEOUS SYSTEMS

Physically based rendering is the process of generating synthetic images indistinguishable from the perception of the real world. These images are generated according to a geometric model of the scene and models for every material and light source present. With this information is possible to simulate the light transport process from the light sources to the camera through the scene in order to synthesize an image. With this simulation, the radiance for each pixel of the image is determined, being radiance the radiant power incident on a given region per area and solid angle unit.

Physically based rendering is a computationally intensive process, and the use of heterogeneous systems may provide additional computation power, at the cost of a more difficult development. This issue is addressed by frameworks such as DICE (Barbosa et al., 2015) and StarPU (Augonnet et al., 2011) that aim to ease the development of heterogeneous applications as well as scheduling and memory management across devices.

2.1 PATH TRACING

One of the first solutions to the global illumination was introduced by Kajiya (1986) when he defined the problem of rendering as the resolution of an integral equation, also know as the rendering equation.

$$I(x,x') = g(x,x')\left[e(x,x') + \int_S \rho(x,x',x'')I(x',x'')dx''\right] \tag{1}$$

Where:

$I(x,x')$	is related to the intensity of light passing from point x to x'
$g(x,x')$	is the geometric term
$e(x,x')$	is related to the intensity of emitted light from x to x'
$\rho(x,x',x'')$	is related to the intensity of light scattered from x'' to x through x'

This equation translates to how much light arrives at a given point from a given direction. This integral equation can not be calculated analytically, so its expected value is calculated through Monte Carlo Integration. Based on this, the goal is to trace a light transport path, with points x_0 .. x_n in which x_0 is a point on a light source and x_n is a point on the camera lens. Commonly, these paths are traced from the camera to the light source, to ensure that paths sampled fall in the image plane and to

capture reflection and refraction paths. However caustic paths have a very low sampling probability, reaching zero if using point light sources. Tracing paths form the light source to the camera however sample caustics more effectively but specular paths can not be sampled.

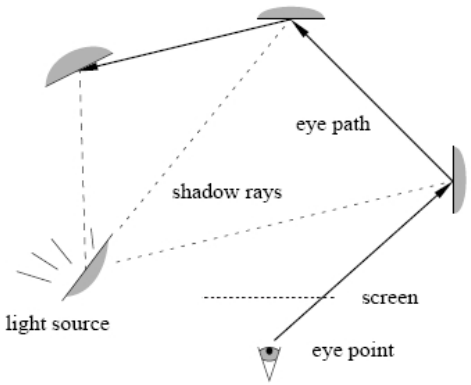


Figure 1: Path Tracing Algorithm

2.2 BIDIRECTIONAL PATH TRACING

BPT, proposed independently by Veach (1998) and Lafortune and Willems (1993) combines both forward and backwards path tracing in one single method that can become more robust than the previous two alone. Although with a different mathematical background, both authors propose that the method should sample pairs of sub-paths containing a light sub-path and a camera sub-path. Then, each vertex of one sub-path is explicitly connected to all vertices of the other one, generating a new set of light transport paths.

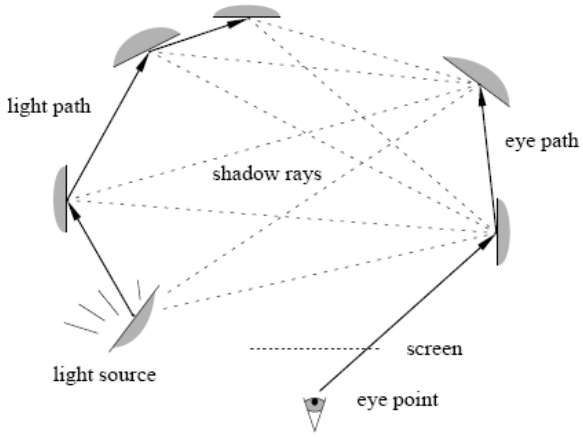


Figure 2: Bidirectional Path Tracing Algorithm

In order to make the algorithm more robust, Veach (1998) proposed the Path Space Integration Framework and MIS. The first consists in converting the problem of light transport from a recursive integral equation (1) to a simple integral over the space of all light transport paths.

$$I_j = \int_\Omega f_j(x) d\mu(x) \tag{2}$$

where Ω is the set of all light transport paths of all lengths, μ is a measure on this space of paths and f_j is the measurement contribution function.

Multiple Importance Sampling is a technique that attempts to combine multiple sampling techniques in a probably good way in order to minimize variance.

$$F = \sum_{i=1}^{n} \frac{1}{n_i} \sum_{j=1}^{n_i} \omega_i(X_{i,j}) \frac{f(X_{i,j})}{p_i(X_{i,j})} \tag{3}$$

Being n the number of sampling techniques and n_i the number of samples for technique i, this estimator attributes a weight ω to each sample. The balance heuristic is a simple and robust way to calculate these weights and is demonstrated that no other heuristic is much better (Veach, 1998, p. 264).

$$\omega_i(x) = \frac{n_i p_i(x)}{\sum_k n_k p_k(x)} \tag{4}$$

In order to use MIS in Bidirectional Path Tracing one must consider that each path of a given length can be sampled in several ways: a path of length n can be sampled by using whatever light and camera sub-paths of s and t vertices as long as $n = s + t + 1$. Given this it is just needed to apply the heuristic above to calculate the respective weight for that path.

$$I = \sum_{s \geq 0} \sum_{t \geq 0} \omega_{s,t}(x_{s,t}) \frac{f_j(x_{s,t})}{p_{s,t}(x_{s,t})} \tag{5}$$

This algorithm is much more robust than path tracing, as it can sample many light transport paths efficiently and robustly. Nonetheless, this algorithm is not perfect for every situation, as it has difficulty sampling reflected caustics (Georgiev et al., 2012).

2.3 PHOTON MAPPING

The previous algorithms attempted to solve the problem of global illumination by sampling light transport paths from the light source to the camera. Photon Mapping (PM) however, attempts to calculate the radiance at any given point by estimating a photon density (Jensen, 1996).

In a first phase the algorithm traces packets of photons from the light sources into the scene and stores them in a range search acceleration structure, the photon map. In a second phase, rays are traced from the camera and the photon map is consulted to calculate the photon density in the neighborhood

of the hit points. One technique used to reduce the variance of this algorithm is the use of distinct photon maps for caustics and diffuse indirect illumination and consult these maps differently: the caustics map is accessed directly while for diffuse illumination a final gathering step is performed. In final gathering, rays are traced from the hit points into the scene and in the intersections of these rays the diffuse map is consulted.

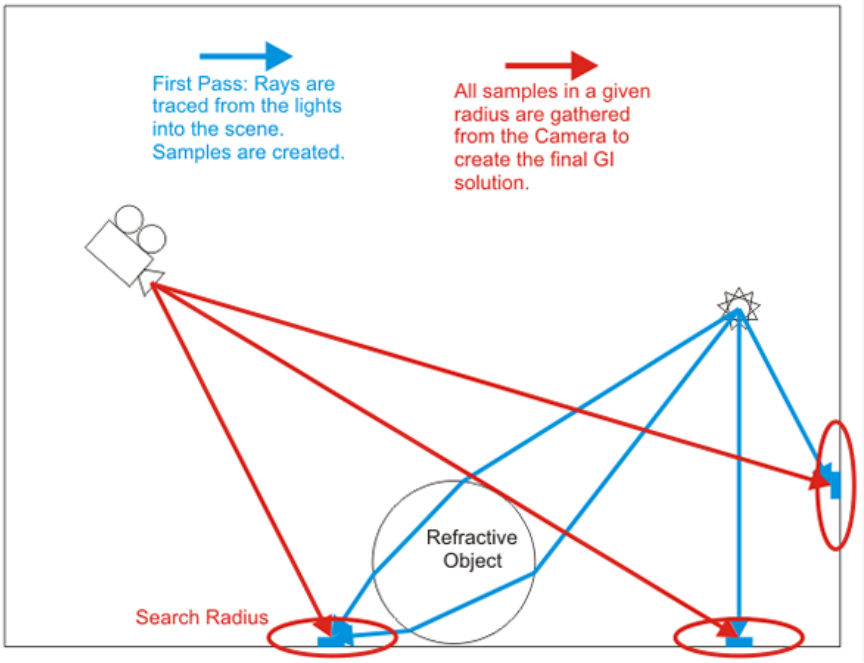


Figure 3: Photon Mapping Algorithm

This algorithm, unlike all the previously mentioned algorithms, introduces bias, because there is an approximation of the photon map search area to an infinity small area. This causes the algorithm to not converge to the correct result but to an approximation with an unknown error (bias).

Although biased, this algorithm performs well in the handling of caustics, even reflected and refracted caustics. However, the convergence rate for the overall diffuse illumination is smaller than that of Bidirectional Path Tracing (Georgiev et al., 2012). Another back draw of this algorithm is memory usage, since the more photons used to rendering, the larger the photon map has to be.

2.3.1 *Progressive Photon Mapping*

Photon Mapping is a biased algorithm, although if the number of photons used tends to infinity and the search radius in the photon map tends to zero, the bias is zero in the limit. This makes the bias in the algorithm consistent and this property can be used to improve the quality of the algorithm.

Instead of storing the positions of photons in the photon map, the hit points from the camera rays are stored in an acceleration structure. Then, when photons are traced from the light sources, a range

search looks for any hit point in the neighborhood and the contribution of that photon is added to the corresponding pixels. In each iteration of the algorithm the search radius is decreased according to a constant parameter of the algorithm (Hachisuka et al., 2008).

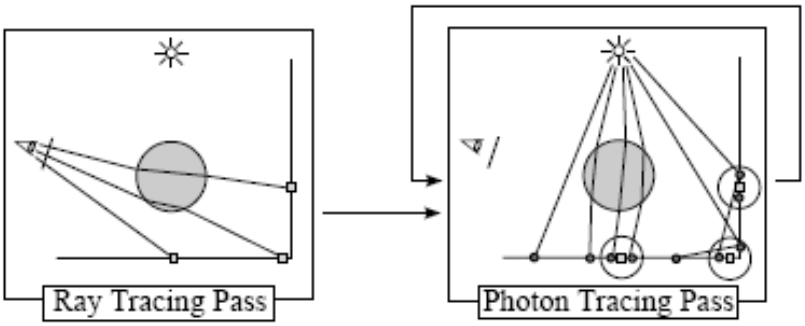


Figure 4: Progressive Photon Mapping Algorithm

This alteration to the algorithm improves the efficiency of Photon Mapping, namely in terms of memory usage, now fixed proportional to the size of the image, instead of proportional to the number of photons used. Although with Progressive Photon Mapping (PPM) the bias may be zero in the limit, this algorithm also produces more useless work the longer it runs, since as the search radius decreases with the execution, more photons are traced that do not contribute to the final image.

2.3.2 *Bidirectional Photon Mapping*

In order to increase the robustness of the Photon Mapping technique, Vorba (2011) proposed the Bidirectional Photon Mapping (BPM) algorithm. Instead of using a heuristic to determine the use of final gather and direct photon map consult, this algorithm uses MIS in order to increase the robustness of the algorithm, mostly when glossy materials are present.

This algorithm operates mostly like PM but using only one photon map. In the rendering phase, instead of tracing only primary rays, a whole path is traced, and at every intersection of this path the photon map is consulted. During this step every photon contribution is independently weighted according to the probability it was generated.

This algorithm can also be progressive by reducing the search radius of the photon map at each iteration.

2.4 VERTEX CONNECTION AND MERGING

This new algorithm proposed by Georgiev et al. (2012), combines two complementary algorithms in order to create a more robust rendering algorithm. As seen previously, Bidirectional Path Tracing has high convergence rate and can handle most types of illumination effects, however it has problem

rendering reflected or refracted caustics. On the other hand, the family of Photon Mapping algorithms has no problem dealing with caustics but has an overall low convergence rate. The combination of these two algorithms into a new one, Vertex Connection and Merging, through MIS is much more robust than the original algorithms alone.

The combination of Photon Mapping and Bidirectional Path Tracing required the reformulation of one of these algorithms, since they calculate the image values through different mathematical frameworks: while Bidirectional Path Tracing calculates an integral on the path space, Photon Mapping calculates a photon density estimation. To achieve this, Photon Mapping was reformulated as a path sampling technique, where a path is successfully sampled if the last vertex of the light sub-path falls into the search area of a vertex from the camera sub-path. These two vertices are effectively merged so the last vertex in the light sub-path is not considered part of the generated path itself but only as a conditional test of the acceptance of this new path. For this new path sampling technique, called Vertex Merging, an associated probability density function is needed to use multiple importance sampling.

$$p_{VM} = [\pi r^2 p(x_{s-1} \rightarrow x_s^*)] p_{VC}(x) \tag{6}$$

where $p_{VC}(x)$ is the probability of generating path x through Bidirectional Path Tracing and the initial part is the probability density function for the last light sub-path vertex to intersect the merging area of the last camera sub-path vertex.

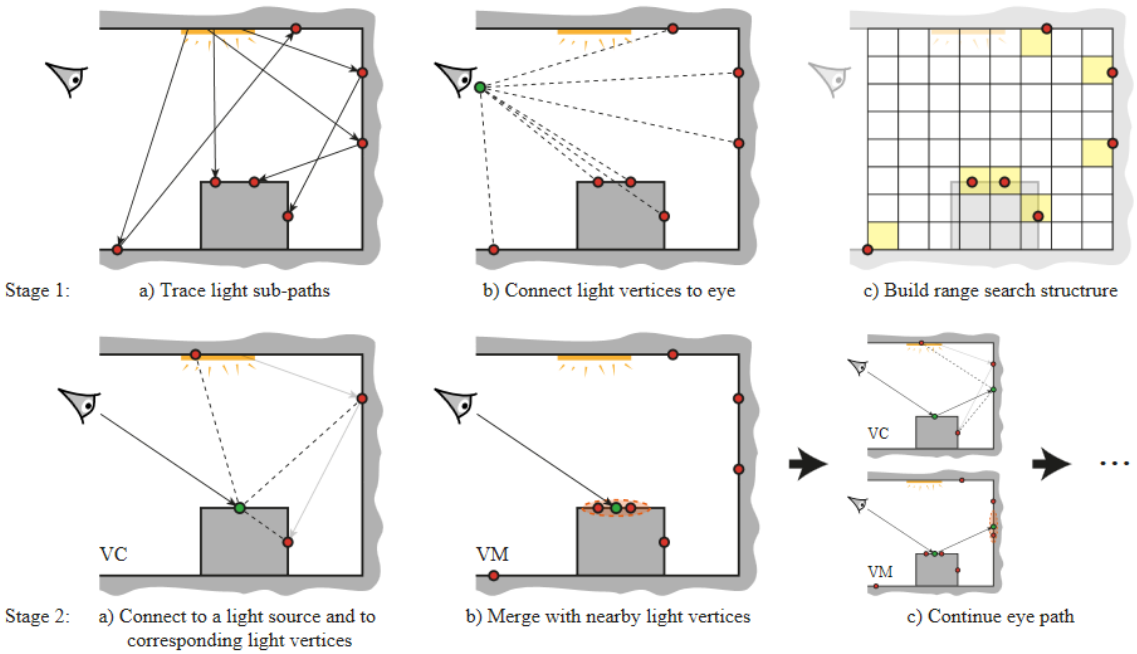


Figure 5: Vertex Connection and Merging Algorithm

With a new path sampling technique, Vertex Merging, in conjunction with Bidirectional Path Tracing, or Vertex Connection, it is possible to combine these two techniques using MIS to achieve ro-

bustness. This new combined algorithm starts like Bidirectional Path Tracing by tracing a set of paths from the light sources, and stores the vertices in a range search acceleration structure. Then when the camera sub-paths are being traced, each vertex of the camera sub-path is connected to all the vertices of the respective light sub-path, and searches the acceleration structure for any vertices from any light sub-path viable for merging. The search radius used in VCM should be smaller than the radius used in other photon mapping based algorithms.

This combined light transport algorithm maintains the high convergence rate of Bidirectional Path Tracing, while being able to handle caustics efficiently like Photon Mapping. The efficiency of this algorithm can still be improved by applying Markov Chain Monte Carlo (MCMC) techniques on top of it (Georgiev et al., 2012).

2.5 HETEROGENEOUS SYSTEMS

An heterogeneous system is a computing system with different processing devices with different architectures. Indeed, most average computers are heterogeneous systems that contain a multicore CPU used for most tasks and a massively parallel GPU used mainly for graphics rasterization. In order to achieve peak performance on each type of device, there are specific architecture dependent optimizations that must be implemented by programmers. This imposes difficulties to developers and inhibits portability of the code across different platforms (Kunzman and Kalé, 2011). However there are potential performance gains in using both devices since there is additional computational power available

When programming CPU's, most parallelism is implicit, in the form of instruction parallelism and the use of vectorial instructions(Single Instruction Multiple Data (SIMD)). The use of vectorial instructions is critical in order to achieve peak performance, and although the compiler may perform the automatic vectorization of the code, the code must follow a strict pattern for this to happen. CPU performance is also dependent on its memory hierarchy model with few registers and a large main memory, with some levels of cache memory to amortize the time cost of accessing the main memory. It is though imperative to use memory locality to avoid cache misses and accessing the main memory which is much slower than accessing the cache memory. Further issues arise with memory hierarchy when introducing thread level parallelism. When multiple cores have a cache line stored in their private cache and that line is altered by one of them, that cache line has to be invalidated and reloaded on all other caches in order to avoid a incoherent state, process that is costly (Hennessy and Patterson, 2011).

GPU architecture is completely different from the CPU. Instead of only having few powerful cores, GPU has a large number of less complex processors, excelling at massively parallel tasks. This level of parallelism is expressed using a different programming model. Using kernels, data parallel functions that deploy a large number of threads that perform a given task. These threads are grouped in blocks that have access to a fast memory that is shared across all threads within the same block. With

a completely different memory model, with a great number of registers, reduced cache memory, and a programmer controlled shared memory, which use is essential in most cases to achieve high performance. Other specific problem in GPU programming is the control of divergence. In a GPU threads execute in warps, sets of threads running the same instruction. When the code reaches a branching instruction if threads follow different paths, each of the branches is executed separately, deactivating the correspondent threads (Kirk and Wen-mei, 2010).

Beyond all the specific challenges of programming each device, there are even more difficulties when using them together, as these two processors have separate addressing spaces, communication between them is costly, and even the programming model is different between them. Furthermore, it is necessary an efficient scheduler that distributes the workload across the devices and deals with load balancing issues. Managing all this manually is impractical, thus making the use of a specialized framework for heterogeneous systems imperative to speed the development process.

2.5.1 *Heterogeneous System Frameworks*

There is a variety of frameworks that aim to solve the issues that arise when developing for heterogeneous systems, namely HMPP (Dolbeau et al., 2007), Harmony (Diamos and Yalamanchili, 2008), Legion (Bauer et al., 2012), StarPU (Augonnet et al., 2011) and DICE (Barbosa et al., 2015). These frameworks aim to provide high level abstractions of an heterogeneous system, providing data management facilities and enhanced scheduling mechanisms. These frameworks also provide automatic work decomposition and transparent data transfers between devices.

Although other frameworks like StarPU are more widely known used and by the scientific community, DICE offers an additional set of features, like the scheduler based on granularity refinement and the hybrid programming model, that differentiate it from the others, thus the choice of DICE for this work.

DICE

DICE (Barbosa et al., 2015) is an application framework for developing heterogeneous applications that aims to ease the process of development for heterogeneous platforms containing both CPU and GPU, with an emphasis on irregular workloads.

DICE uses a task driven model that schedules work across processing devices through a granularity refinement. When a task is submitted to the scheduler, sub-tasks of finer granularity are generated and executed on the available devices. This scheduler analyses the execution time of the task on each device to enhance the workload distribution in order to attain maximum performance. DICE also provides a virtual shared memory manager that abstracts different addressing spaces by presenting the developer only one address space and transparently performing copies between different addressing spaces whenever required.

DICE tasks support a variety of execution modes. The developer can provide a single kernel and the task can execute automatically on CPU and GPU, while it is also possible to provide the task with specialized code for each architecture. The specialized GPU implementation can be provided as a CUDA kernel or as a call to specialized libraries. These possibilities allow a simple programming model while retaining the possibility of refinement, architecture specific optimizations and the use of external optimized libraries.

3

EXPERIMENTAL WORK

To evaluate the suitability of advanced light transport algorithms to heterogeneous systems, optimized implementations of all algorithms were developed for both CPU and GPU. These implementations were developed using specialized ray tracing libraries, namely Nvidia Optix (Parker et al., 2010) and Intel Embree (Wald et al., 2014).

There are two possible schemes of work decomposition across devices. One is image space decomposition, where each device processes a portion of the image pixels. Another possible alternative is to distribute the algorithm iterations across the devices. In PT and BPT the choice between these two approaches is almost irrelevant, however in photon based algorithms (BPM and VCM) it is not that simple. If image plane division was applied, it would force communication between the CPU and the GPU in every iteration since the photon map, which would be generated concurrently across the devices, has to be shared among them. If an iteration division was applied, for every CPU core a complete photon map would be stored in the main memory. When the number of such cores is significant, the memory requirements are too high for most computing systems.

To solve these issues, DICE was configured to consider all the CPU cores as a monolithic device, leaving the distribution of work in the CPU to the application code. The work between the CPU and the GPU is then distributed autonomously by the DICE scheduler as a set of independent iterations, eliminating all communication needs. Between the CPU cores the image plane is divided, eliminating the excessive memory use. This approach was applied to every developed algorithm in order to ease development and to allow an algorithm independent DICE configuration. The Intel Thread Building Blocks library was used to provide thread parallelism in the CPU implementation, due to the ease of use as well as for compatibility with the DICE framework.

3.1 METHODOLOGY

To test the scalability and efficiency of the studied algorithms, the execution times of the studied algorithms were measured using only the GPU, and while using only the CPU or both, with a varying number of CPU cores and threads per core, in order to evaluate the gain of using Hyper Threading (HT). Every time measurement was executed five times, selecting the best time as the final result. The workload distribution between the devices was also measured by the DICE scheduler at the end of

the five time measurements, being the final result the best possible distribution calculated by DICE. These results represent the fraction of iterations assigned to each device, which may not be a good performance evaluation metric since the iteration time is not uniform, specially in photon mapping based algorithms, in which the iteration time is dependent on the search radius, which decreases in every iteration. These tests were executed using the Living Room Scene with a resolution of 1024x768 and 50 samples per pixel.

To evaluate the image quality produced by the studied algorithms, three scenes were used for testing, each with a different goal. The Sponza scene, courtesy of Crytek, is an outdoor scene with only diffuse materials, but complex geometry nonetheless. The Kitchen scene from Lux Renderer, is an indoor scene with glossy materials. The final and most complex scene, the Living Room, courtesy of Iliyan Georgiev, is an indoor scene as well but with an emphasis on reflected caustics and complex lighting. For every scene a reference image was rendered with VCM with 100000 samples per pixel. Then for every algorithm a set of images with an approximate rendering time were produced. All these images were compared to the reference image using the Root Mean Squared Error (RMSE) metric.

Figure 6: Sponza scene reference image.

Figure 7: Kitchen scene reference image.

Figure 8: Living Room scene reference image.

All experiments were executed on a dual-socket computer with two 10 core Intel Xeon E5-2670 v2 at the frequency of 2.50 GHz, 64 GB of RAM in a Non Uniform Memory Access (NUMA) configuration and a Nvidia Tesla K20 GPU.

Every software used was updated to the versions listed in table 1.

Software	Version
Linux	2.6.32-279
GCC	4.8.2
CUDA Toolkit	5.5
Optix	3.7
Embree	2.5.1
Intel TBB	4.3

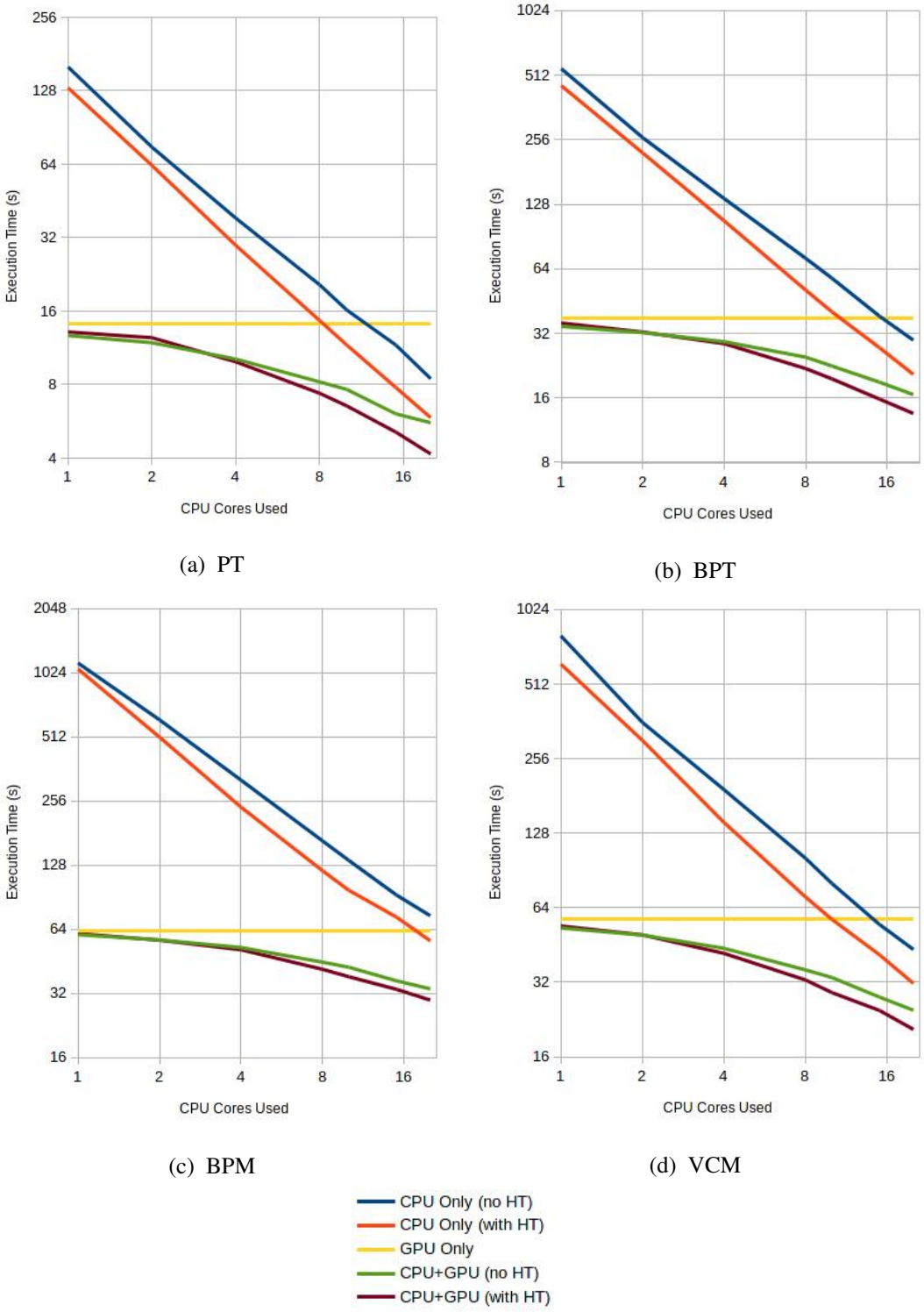
Table 1: Software Used

3.2 EXPERIMENTAL RESULTS

3.2.1 *Execution Time Results*

Figure 9 presents the execution times of the tested algorithms using different device sets. All algorithms show a proportional reduction in execution times with the number of CPU's used. Please note that the "GPU only" yellow horizontal line represents a constant value, since no CPU cores are used.

3.2. Experimental Results



(a) PT

(b) BPT

(c) BPM

(d) VCM

Figure 9: Execution Times of the Algorithms

3.2.2 *CPU Speedup Results*

Due to the lack of a clear speedup definition for heterogeneous systems, figure 10 presents only the speedup for the CPU implementation. However, in section 3.2.5 there is presented an heterogeneous analysis. Every algorithm achieves an almost linear speedup, with the exception BPM that only achieves a speedup of 15.4 using 20 cores. Although not present in this plot, all algorithms take advantage of Hyper Threading, which increases the overall performance between 30% to 50%.

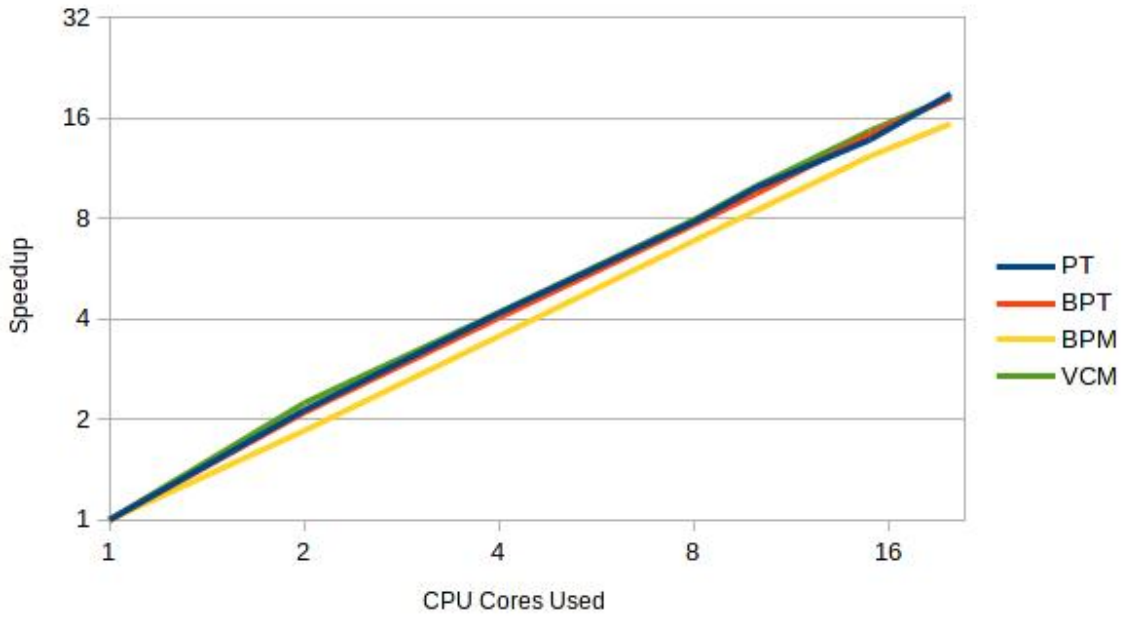


Figure 10: Speedup Analysis

3.2.3 *CPU Parallelization Efficiency Results*

Due to the lack of a clear efficiency definition for heterogeneous systems, figure 11 presents only the efficiency for the CPU implementation. However, in section 3.2.5 there is presented an heterogeneous analysis. Every algorithm has a parallelization efficiency above 0.9 independently of the number of cores used, except BPM that only achieves an efficiency of 0.77 using 20 cores.

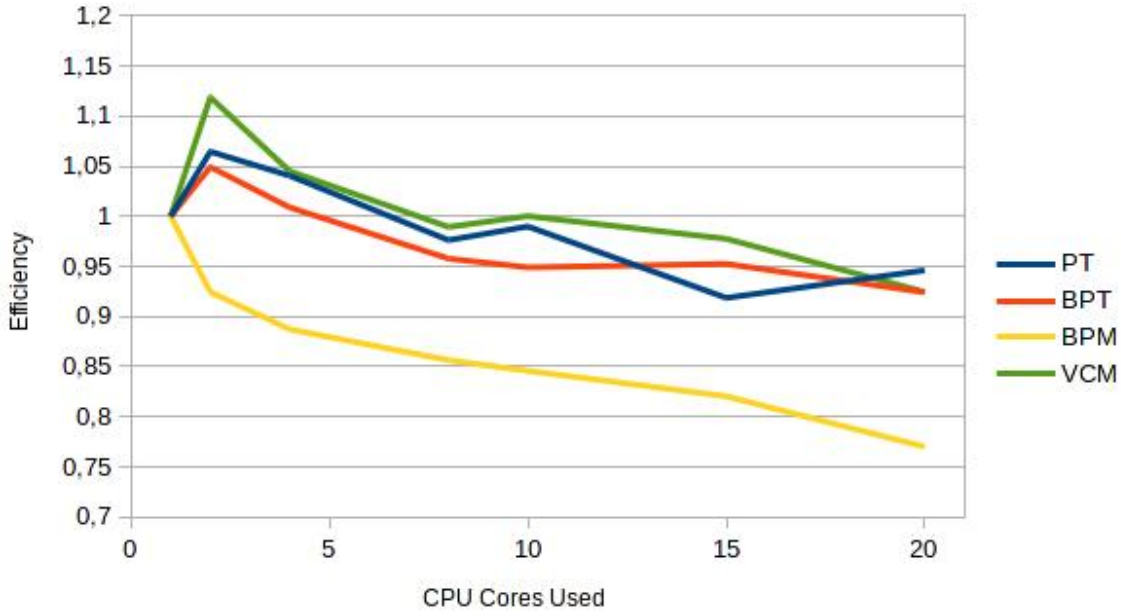
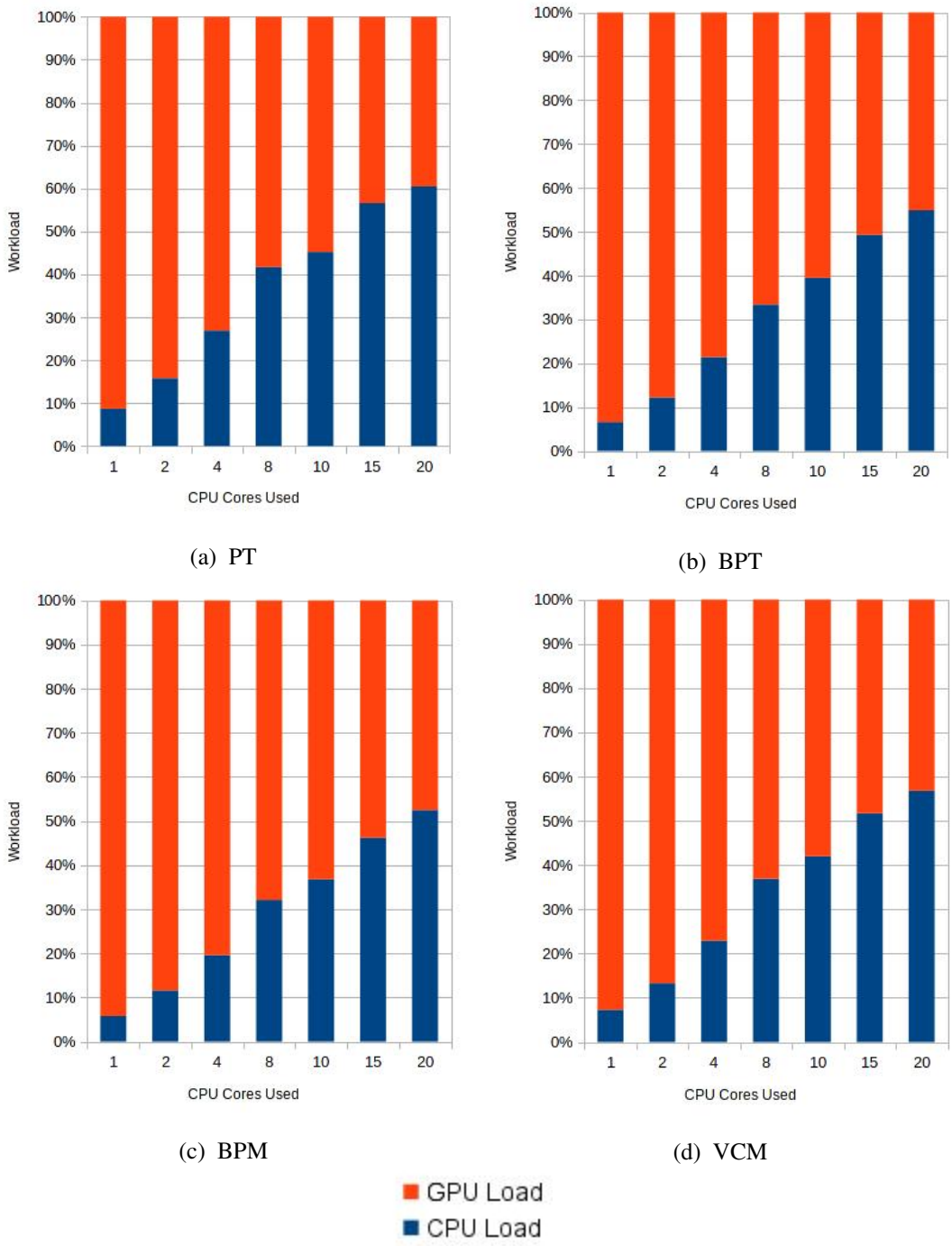


Figure 11: Efficiency Analysis

3.2.4 *Workload Distribution Results*

Figure 12 presents the workload distributions between the CPU's and the GPU. Remember that this is measured as the fraction of iterations processed by each device and that DICE sees the CPU as a single device (image space decomposition being then performed among cores by the rendering application itself). The relative workload processed by the CPU implementation increases with the number of cores used. This behavior is similar for all four algorithms.



(a) PT

(b) BPT

(c) BPM

(d) VCM

Figure 12: Workload Distribution

3.2.5 *Heterogeneous Efficiency*

Although all previous metrics may give a notion of how well the heterogeneous implementations have performed, there was no absolute measurement of how well the resources were used.

3.2. Experimental Results

Chamberlain et al. (1998) defined heterogeneous speedup according to equation 7, where T_{ref} is the execution time on a reference device selected according to some criterium and T_D is the execution time on an heterogeneous system constituted by the set D of devices.

$$S_h(D) = \frac{T_{\text{ref}}}{T_D} \tag{7}$$

The optimal heterogeneous speedup, $S_h^*(D)$, can not be defined as function of the number of used devices, because different devices offer different amounts of computational power. Chamberlain et al. (1998) expressed this ideal speedup as a ratio of computing rates.

Let the computing rate for the device $d \in D$ for a given workload W be $R_d = \frac{W}{T_d}$, then the ideal computing rate is the sum of all the computing rates of all the devices $d \in D$.

$$R_D^* = \sum_{d \in D} R_d = W \sum_{d \in D} \frac{1}{T_d} \tag{8}$$

Chamberlain et al. (1998) define $S_h^*(D)$ as the ratio between available computing rate and the computing rate of the reference device:

$$S_h^*(D) = \frac{R_D^*}{R_{\text{ref}}} = T_{\text{ref}} \sum_{d \in D} \frac{1}{T_d} \tag{9}$$

Given the definitions of heterogeneous speedup and optimal heterogeneous speedup (equations 7 and 9), heterogeneous efficiency can thus be defined as the ratio of both

$$E_h(D) = \frac{S_h(D)}{S_h^*(D)} = \frac{R_D}{R_D^*} = \frac{\frac{1}{T_D}}{\sum_{d \in D} \frac{1}{T_d}} \tag{10}$$

Given these definitions, it is possible to calculate the efficiency of the heterogeneous implementations using one GPU and a varying number of CPU's, as depicted in the figure 13. All algorithms achieve an heterogeneous efficiency above 0.9, except BPM.

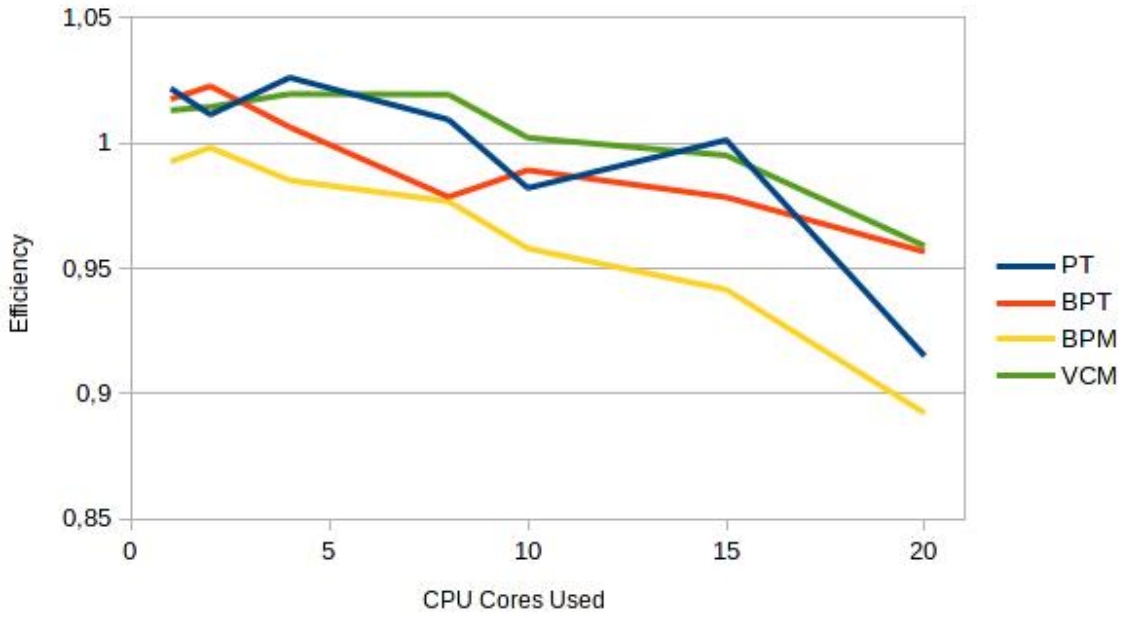


Figure 13: Heterogeneous Efficiency Analisys

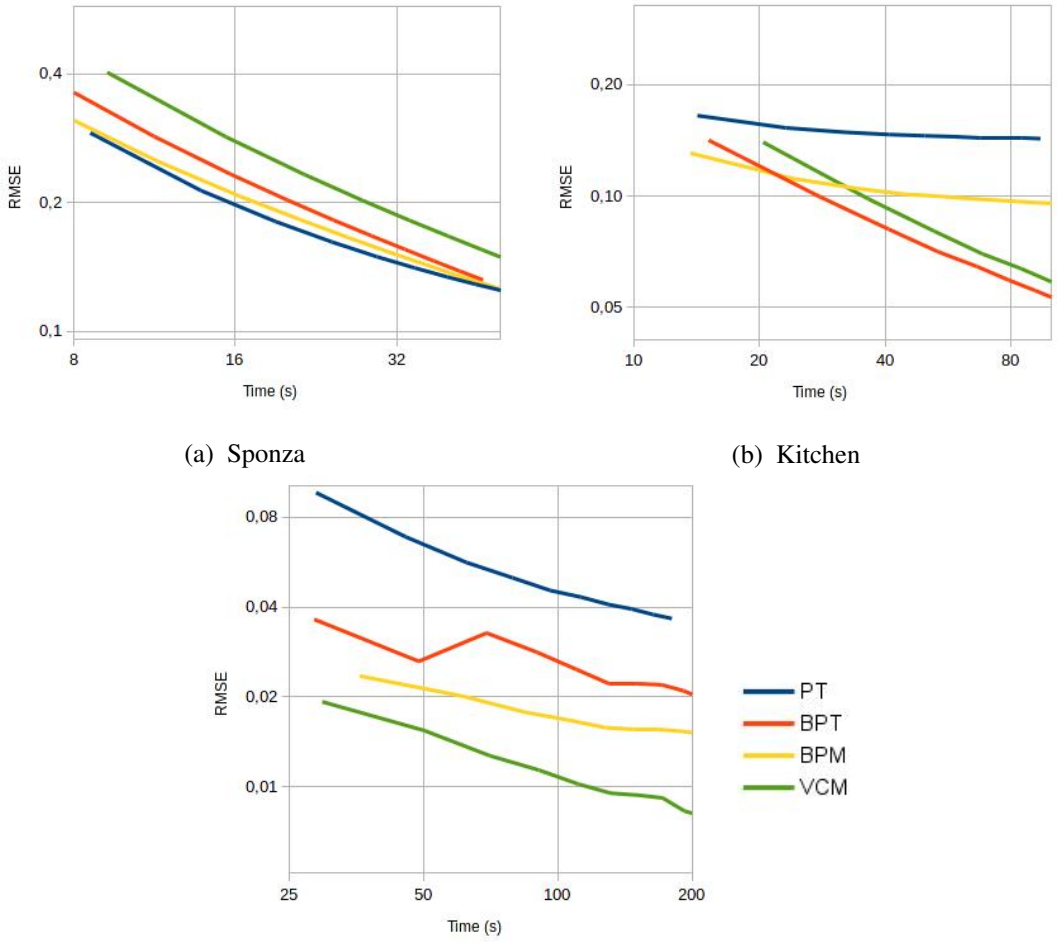
3.2.6 *Image Quality Results*

Figure 14 presents the image quality measures in function of the rendering time for the various tested algorithms and test scenes.

For Sponza the Path Tracer outperforms the remaining algorithms.

In the Kitchen scene, both Bidirectional Path Tracer and Vertex Connection and Merging algorithms converge to the expected result, although the former is faster for this scene. Both Path tracer and Bidirectional Photon Mapping present lower convergence rates.

In the Living Room scene, Vertex Connection and Merging presents the fastest convergence, while the remaining algorithms converge much more slowly.



(a) Sponza

(b) Kitchen

(c) Living Room

Figure 14: Image Quality Comparison

3.3 DISCUSSION OF RESULTS

The execution times measures (subsection 3.2.1) indicate that all algorithms studied are scalable and take advantage of both devices in the rendering process. Note that no inflection point was reached, i.e., a point above which execution time starts increasing with additional processing devices. In fact, all the curves present a negative slope, meaning that not even an horizontal asymptote (i.e., constant execution time with increasing devices) has been reached. This trend suggests that the above referred inflection point is far from being reached and all algorithms should scale well with additional devices. It was not possible to further increase the number of devices in order to detect such inflection point. Speedup and efficiency results (subsections 3.2.2 and 3.2.3) also show an almost perfect scalability in the CPU implementations, suggesting that the algorithms would scale well for additional CPU cores. This hypothesis would have, however, to be verified experimentally.

3.3. Discussion of Results

Previous research (Palhas, 2013) has shown scalability issues when using the second processor socket in a NUMA architecture with PPM. Although with a slightly lower efficiency, the photon mapping based algorithms developed scale across both sockets. This behavior may be due to the usage of specialized ray tracing and threading libraries, such as Embree and Intel Thread building Bocks.

The developed BPM implementation has a lower efficiency than all the remaining algorithms. This may be due to the high memory access this algorithm has compared to all the other algorithms, as BPM needs to gather lighting from a large number of photons (much more that the photons used in VCM and more than the number of vertices accessed by BPT during the connection phase) due to the higher search radius needed to reliably gather photons.

Both workload distribution and heterogeneous efficiency (subsections 3.2.4 and 3.2.5) results demonstrate that the developed heterogeneous implementation is efficient and that the DICE overhead is minimal and that it always manages to find the optimal workload distribution for every situation in order to maximize the efficiency of the resources used. These results clearly show, for all algorithms, that the workload is distributed among all devices, with the fraction assigned to the CPU increasing with the number of cores. This suggests that no devices starve for work, which is a good explanation of why performance scales so well for the number of tested devices. Since these algorithms and the used workload decomposition do not resort to global communications and also do not require work replication across different devices, the other overhead which could be responsible for non-scalable results would be a poor workload distribution; this is clearly not the case. These results are promising in that it makes heterogeneous computing a viable way of reducing the rendering time needed to achieve a given result.

The image quality results (subsection 3.2.6) show that each algorithm is best suited for specific situations. In the Sponza scene the best algorithm is Path Tracing due to the outdoor scene configuration, which is detrimental in bidirectional methods due to many of the light paths traced are lost and miss the scene, as well as the simple materials, causing the most relevant light transport paths to be direct illumination and diffuse inter-reflections, which are completely captured by Path Tracing, dispensing with more complex solutions. In the Kitchen scene the best algorithm is the Bidirectional Path Tracing, while Path tracer and Bidirectional Photon Mapping fail to converge at all due to the high variance of the lighting effects present, namely caustics incident on glossy surfaces. In the Living Room scene the Best algorithm is VCM due to the presence of many specular-diffuse-specular paths (namely the reflected caustics in the mirror), which VCM was specially designed to capture. Although not always the best algorithm, VCM is able to converge in every scene (w.r.t. RMSE), making it the more robust algorithm present among the studied techniques, as shown by previous research (Davidovič et al., 2014).

3.4 DICE EVALUATION

One of the main goal of this work is to evaluate the DICE framework, regarding both performance and usability. After this work, it is possible to say that this framework excels in both points.

Regarding performance, as shown by the previous results (subsections 3.2.4 and 3.2.5), DICE manages to find the optimal workload distribution for every algorithm and device configuration used, always reaching an heterogeneous efficiency above 85%. This shows that the framework has a low performance overhead as well as an optimal (or close to optimal) scheduling policy.

Regarding usability, DICE presents to the user a simple and intuitive C++ interface. It was simple to integrate the existing rendering code with DICE due to the possibilities this framework offers regarding how to specify the task to perform: in this work, specialized implementations were passed to the DICE framework. Other features like the hybrid specification of tasks and the memory manager were not tested.

Overall, DICE proves to be a good solution to take full advantage of heterogeneous systems, being both simple and efficient.

4

CONCLUSIONS

This work main goal was to evaluate the suitability of a set of state of the art light transport algorithms, based on path space integration, for execution on parallel heterogeneous systems using the DICE framework. Simultaneously, we were interested on identifying DICE's strong and weak points, regarding both performance and usability.

Four path space integration algorithms were selected (path tracing, bidirectional path tracing, bidirectional photon mapping and vertex connection and merge) based on their relevance and robustness for solving the light transport problem. Robust and efficient kernels were developed for each of these algorithms using specialized and highly optimized ray tracing libraries, namely Optix for the GPU and Embree for the CPU. DICE was then used to manage execution on a heterogeneous platform with 20 CPU cores and a GPU.

Results demonstrated that all algorithms are able to efficiently exploit the available set of heterogeneous devices. In fact, even when using all the devices, heterogeneous efficiency is well above 85% for all algorithms. Execution times also show that the algorithms scale well within the evaluated heterogeneous system. No inflection point (i.e., a number of devices above which execution times starts to increase) is reached.

DICE is a complete framework for programming heterogeneous systems, however during this work, only a small set of the available features were used. As specialized implementations for each device were used, and memory transfers were inexistent, only the DICE scheduler was used. The scheduler was efficient and distributed workload in a way that maximized performance and efficiency. From an usability point of view, DICE uses a solid and simple C++ generic interface, making it and easy to program and interact with the existing rendering code, although there was the need to reconfigure some settings, namely the way DICE views the various available devices. Being able to consider the CPU as a monolithic device or as a set of devices (CPU cores) allows more sophisticated work decomposition schemes at the cost of developing simplicity. This configuration possibility was essential in photon mapping based algorithms as it allows a hybrid work decomposition, thus avoiding performance issues regarding communication.

With respect to the light transport algorithms, image quality measurements show that different algorithms are more appropriate (i.e., converge faster) for different scene characteristics. VCM, being the most complete algorithm, is the more robust, always converging to the expected solution; VCM

excels when specular-diffuse-specular paths are present, which have zero or very low probability of being sampled by the remaining algorithms.

4.1 FUTURE WORK

Current results do not exhaustively evaluate scalability on a heterogeneous system, because one single GPU is used. Extending the code to a multi-GPU environment is a priority, even though not trivial due to limitations of the Optix framework. Another interesting possibility is to adapt the developed code to the Intel Many Integrated Core (MIC) architecture using Embree as well.

A very interesting line of research would be to understand and define metrics that characterize performance on an intuitive manner on heterogeneous systems. In fact, metrics such as speedup, efficiency, iso-efficiency, among others, are well understood for homogeneous systems, both for the fixed size and fixed time cases. There is clearly a need to define metrics for heterogeneous systems, which are well accepted and understood by the community.

Lastly, a comparative study between DICE and StarPU in terms of performance and usability seems important in order to determine which is the best heterogeneous system development framework.

More sophisticated rendering algorithms, such as those based on Metropolis light transport combined with VCM, have to be assessed with respect to both performance on heterogeneous systems and rate of convergence.

BIBLIOGRAPHY

C. Augonnet, S. Thibault, R. Namyst, and P. Wacrenier. Starpu: a unified platform for task scheduling on heterogeneous multicore architectures. *Concurrency and Computation: Practice and Experience*, 23(2):187–198, 2011.

J. Barbosa, S. Keely, R. Ribeiro, D. S. Fussel, C. Lin, A. Proença, and L. P. Santos. Automatic granularity control for irregular workloads on heterogeneous systems. Submitted to the 20th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming, PPoPP, 2015.

M. Bauer, S. Treichler, E. Slaughter, and A. Aiken. Legion: expressing locality and independence with logical regions. In *Proceedings of the international conference on high performance computing, networking, storage and analysis*, page 66. IEEE Computer Society Press, 2012.

R. Chamberlain, D. Chacey, and A. Patel. How are we doing? an efficiency measure for shared, heterogeneous systems. In *Proc. of the ISCA 11th International Conference on Parallel and Distributed Computing Systems*, pages 15–21, 1998.

T. Davidovič, J. Křivánek, M. Hašan, and P. Slusallek. Progressive light transport simulation on the gpu: Survey and improvements. *ACM Transactions on Graphics (TOG)*, 33(3):29, 2014.

G. F. Diamos and S. Yalamanchili. Harmony: an execution model and runtime for heterogeneous many core systems. In *Proceedings of the 17th international symposium on High performance distributed computing*, pages 197–200. ACM, 2008.

R. Dolbeau, S. Bihan, and F. Bodin. Hmpp: A hybrid multi-core parallel programming environment. In *Workshop on General Purpose Processing on Graphics Processing Units (GPGPU 2007)*, 2007.

I. Georgiev, J. Křivánek, T. Davidovič, and P. Slusallek. Light transport simulation with vertex connection and merging. *ACM Trans. Graph.*, 31(6):192:1–192:10, November 2012. ISSN 0730-0301.

T. Hachisuka, S. Ogaki, and H. W. Jensen. Progressive photon mapping. *ACM Trans. Graph.*, 27(5): 130:1–130:8, December 2008. ISSN 0730-0301.

J. L. Hennessy and D. A. Patterson. *Computer architecture: a quantitative approach*. Elsevier, 2011.

H. W. Jensen. Global illumination using photon maps. In *Rendering Techniques' 96*, pages 21–30. Springer, 1996.

J. T. Kajiya. The rendering equation. *SIGGRAPH Comput. Graph.*, 20(4):143–150, August 1986. ISSN 0097-8930.

D. B. Kirk and W. H. Wen-mei. *Programming massively parallel processors: a hands-on approach*. Elsevier, 2010.

D. M. Kunzman and L. V. Kalé. Programming heterogeneous systems. In *Parallel and Distributed Processing Workshops and Phd Forum (IPDPSW), 2011 IEEE International Symposium on*, pages 2061–2064. IEEE, 2011.

E. P. Lafortune and Y. D. Willems. Bi-directional path tracing. In *Proceedings of Third International Conference on Computational Graphics and Visualization Techniques (Compugraphics '93)*, pages 145–153, 1993.

M. Palhas. An evaluation of the gama/starpu frameworks for heterogeneous platforms: the progressive photon mapping algorithm. Master's thesis, Universidade do Minho, 2013.

S. G. Parker, J. Bigler, A. Dietrich, H. Friedrich, J. Hoberock, D. Luebke, D. McAllister, M. McGuire, K. Morley, A. Robison, et al. Optix: a general purpose ray tracing engine. *ACM Transactions on Graphics (TOG)*, 29(4):66, 2010.

E. Veach. *Robust Monte Carlo Methods for Light Transport Simulation*. PhD thesis, Stanford, CA, USA, 1998. AAI9837162.

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

I. Wald, S. Woop, C. Benthin, G. S. Johnson, and M. Ernst. Embree: a kernel framework for efficient cpu ray tracing. *ACM Transactions on Graphics (TOG)*, 33(4):143, 2014.