ТРАССИРОВКА ЛУЧЕЙ МЕТОДОМ МОНТЕ–КАРЛО ЧЕРЕЗ ОСЕСИММЕТРИЧНЫЕ ОПТИЧЕСКИЕ ЭЛЕМЕНТЫ С ИСПОЛЬЗОВАНИЕМ K-МЕРНОГО ДЕРЕВА

Е.С. Андреев [1,2], М.А. Моисеев [1,2], К.В. Борисова [1,2], Л.Л. Досколович [1,2]
[1] Институт систем обработки изображений РАН, Самара, Россия,
[2] Самарский государственный аэрокосмический университет имени академика С.П. Королёва (национальный исследовательский университет) (СГАУ), Самара, Россия

Аннотация

Предложена модификация процедуры трассировки лучей методом Монте–Карло для осесимметричных поверхностей. Модификация заключается в представлении оптических поверхностей в виде набора усечённых конусов и использовании оригинального k-мерного дерева для упорядочивания конических примитивов в пространстве, что позволяет значительно сократить время поиска точки пересечения луча и оптической поверхности. Результаты моделирования подтверждают, что модифицированный метод трассировки лучей работает в 3–12 раз быстрее по сравнению с традиционными методами трассировки лучей, использующими представление оптических поверхностей в виде набора треугольников.

Ключевые слова: трассировка лучей Монте–Карло, трассировка пути, геометрическая оптика, вычислительные методы, системы с особой симметрией.

Цитирование: **Андреев, Е.С.** Трассировка лучей методом Монте–Карло через осесимметричные оптические элементы с использованием k-мерного дерева / Е.С. Андреев, М.А. Моисеев, К.В. Борисова, Л.Л. Досколович // Компьютерная оптика. – 2015. – Т. 39, № 3. – С. 357-362.

Введение

Трассировка лучей методом Монте–Карло [1] используется при решении прямых и обратных задач геометрической оптики [2, 3], в том числе для моделирования распределений освещённости, интенсивности или яркости, формируемых оптическими системами. Процедура трассировки лучей подразумевает многократный поиск точек пересечения лучей и оптических поверхностей, последующее определение нормалей в полученных точках поверхностей и вычисление преломлённых и отражённых лучей. При этом основная вычислительная сложность приходится на поиск точек пересечения лучей с оптическими поверхностями.

Как правило, для описания оптических поверхностей используются различные сплайновые формы представления, что делает невозможным точное аналитическое решение задачи трассировки лучей. Для расчёта точек пересечения геометрия поверхностей аппроксимируется набором простых фигур (примитивов), например, треугольников [4–6]. Развитие методов трассировки лучей, использующих треугольные примитивы, привело к появлению большого класса ускоряющих структур данных [7–12], таких как октодеревья [7], деревья ограничивающих объёмов (BVH) [8], k-мерные деревья [9]. Работа таких ускоряющих структур данных основывается на упорядочивании примитивов в пространстве. В работах [10–13] описываются методы эффективного поиска пересечений луча с примитивами, а также алгоритмы построения k-мерных деревьев с целью минимизации времени поиска и эффективного использования оперативной памяти.

В данной работе предложен эффективный метод поиска точки пересечения луча с оптической поверхностью, обладающей осевой симметрией. Примером задачи с такой геометрией является задача расчёта оптических элементов для светодиодов, в ходе решения которой оптимизационными методами процедура трассировки лучей может вызываться сотни раз [14]. Свойство осевой симметрии позволяет использовать в качестве примитивов усечённые конусы, что значительно увеличивает точность трассировки лучей, при этом существенно сокращая вычислительные затраты. Для подтверждения эффективности модифицированной процедуры трассировки лучей результаты работы разработанного метода были сравнены с результатами традиционной трассировки лучей. Выигрыш по времени составил от 3 до 12 раз, при этом результаты расчёта световых распределений практически совпадают.

1. Аппроксимация поверхности и поиск точки пересечения

Сформулируем задачу поиска точки пересечения луча с моделируемой оптической системой. Пусть луч задан в следующем виде:

$$\mathbf{R}(t) = \mathbf{O} + t\mathbf{D}, \quad t \geq 0, \tag{1}$$

где вектор $\mathbf{R}$ – вектор-функция точек луча, $\mathbf{O} = (x_0, y_0, z_0)$ и $\mathbf{D} = (s_x, s_y, s_z)$ – вектора, определяющие начало и направление луча, а t – лучевой параметр. Введём осесимметричную оптическую поверхность S, образованную вращением профиля γ вокруг оси Oz в пространстве. Взаимное расположение луча и поверхности в пространстве изображено на рис. 1. Необходимо найти точку первого пересечения луча с данной поверхностью и рассчитать нормаль в этой точке.

Аппроксимируем гладкий профиль γ ломаной γ_Δ с узлами (x_i, z_i) с точностью Δ. Поверхность S_Δ,

образованная вращением ломаной γ_Δ вокруг оси Oz и состоящая из набора усечённых конусов $C_i, i=1,...,N$, также аппроксимирует исходную поверхность S с точностью Δ (рис. 2).

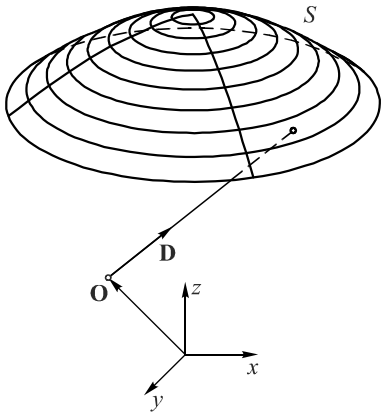

Рис. 1. Взаимное расположение луча и поверхности в пространстве

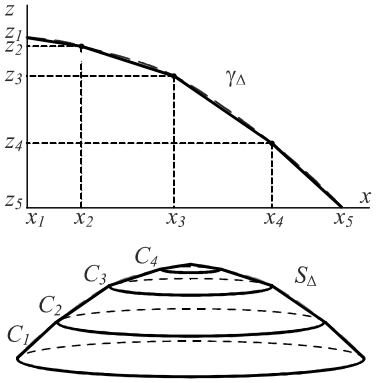

Рис. 2. Аппроксимация осесимметричной поверхности набором усечённых конусов

Каждый из усечённых конусов C_i, входящих в поверхность S_Δ, описывается следующим уравнением:

$$\begin{aligned} &x^2+y^2=\left(k_i z+b_i\right)^2,\\ &z_i\le z\le z_{i+1},\\ &k_i=\frac{x_{i+1}-x_i}{z_{i+1}-z_i},\\ &b_i=x_i-k_i z. \end{aligned} \qquad (2)$$

Таким образом, задача поиска точки пересечения луча с поверхностью S сводится к задаче поиска пересечения луча с набором усечённых конусов. Лучевой параметр t, соответствующий точке пересечения отдельного конического примитива C_i и луча (1), определяется из следующего квадратного уравнения:

$$\begin{aligned} &At^2+Bt+C=0,\\ &A=s_x^2+s_y^2-k_i^2 s_z^2,\\ &B=2\left(x_0 s_x+y_0 s_y-k_i^2 z_0 s_z-k_i b_i s_z\right),\\ &C=x_0^2+y_0^2-k_i^2 z_0^2-b_i^2-2k_i b_i z_0. \end{aligned} \qquad (3)$$

Отметим, что при трассировке большого количества лучей основные временные затраты приходятся не на непосредственное вычисление точек пересечения, а на проверку наличия пересечений лучей с примитивами. Несмотря на то, что проверка пересечения луча и усеч`нного конуса требует от 39 до 64 операций с плавающей запятой (флопа), а аналогичная проверка для треугольника [6] – только от 16 до 45 флопов, использование усечённых конусов является более выгодным, так как количество таких примитивов, необходимых для аппроксимации поверхности, как правило, отличается на несколько порядков в меньшую сторону по сравнению с треугольными примитивами. В качестве примера, подтверждающего этот факт, на рис. 3 приведён рассчитанный профиль стандартного коллиматора, работающего по принципу полного внутреннего отражения, а в табл. 1 указано количество конических и треугольных примитивов, необходимых для аппроксимации поверхности оптического элемента с заданной точностью.

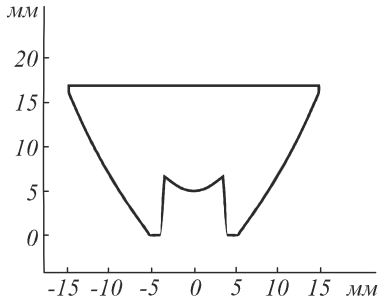

Рис. 3. Профиль коллиматора

Табл. 1. Количество примитивов, необходимых для аппроксимации поверхностей оптического элемента

Точность аппроксимации, мм	Количество конических примитивов	Количество треугольных примитивов
0,01	18	1782
0,001	44	13770
0,0001	125	123246

2. K-d дерево для конических примитивов

Простейший алгоритм поиска пересечения луча с поверхностью, представленной набором усечённых конусов, представляет собой последовательную проверку пересечения луча с каждым из примитивов и расчёт точек пересечения с помощью уравнения (3). Вычислительная сложность такого алгоритма составляет $O(N)$, где N – количество примитивов.

Для уменьшения вычислительной сложности используется упорядочивание примитивов в пространстве. Такой подход позволяет эффективно исключать при поиске примитивы в областях, через которые не проходит луч, что значительно сокращает количество вызовов процедуры проверки пересечения.

В данной работе предлагается использовать упорядочивание примитивов с помощью k-мерного дерева [9]. Данная структура данных представляет собой бинарное дерево, соответствующее иерархии вложенных друг в друга пространственных областей, содержащих примитивы. K-мерное дерево позволяет производить поиск пересечения луча и примитивов со средней вычислительной сложностью $O(\log N)$ [10], а

сложность построения самой структуры данных составляет $O(N \log N)$ [12].

2.1. Построение k-мерного дерева

Построение k-мерного дерева представляет собой рекурсивную процедуру разбиения пространства на вложенные друг в друга элементарные области с последующим сохранением информации о примитивах, лежащих в данных областях (в том числе частично). В случае бинарного дерева каждая область разбивается на две подобласти. Данная процедура выполняется до тех пор, пока не будет достигнута определённая глубина дерева или количество примитивов в узле дерева не окажется менее заданного значения.

Особенностью при построении k-мерного дерева, содержащего усечённые конусы, является выбор элементарных областей, на которые разбивается пространство, содержащее примитивы. Параллелепипеды со сторонами, параллельными осям координат, традиционно используемые при упорядочивании треугольных примитивов, не позволяют эффективно упорядочить осесимметричные объекты. В данной работе в качестве элементарных областей предлагается использовать цилиндрические слои, которые являются осесимметричным аналогом параллелепипедов. Цилиндрический слой $V(R_{in}, R_{out}, z_{down}, z_{up})$ представляет собой область, заключённую между двумя коаксиальными цилиндрическими поверхностями и двумя плоскостями, перпендикулярными оси вращения Oz, здесь R_{in}, R_{out} – радиусы внутренней и внешней цилиндрических поверхностей, z_{down}, z_{up} – координаты ограничивающих плоскостей.

Рассмотрим построение дерева для набора конических примитивов. Сначала определим минимальный цилиндрический слой $V(R_{in}, R_{out}, z_{down}, z_{up})$, полностью содержащий все примитивы набора. Данный цилиндрический слой можно разбить на два подслоя следующими способами: цилиндрической поверхностью с радиусом $R_{split} \in \left(R_{in}, R_{out}\right)$ или плоскостью, перпендикулярной оси вращения с координатой $z_{split} \in \left(z_{down}, z_{up}\right)$. После разбиения получаются два цилиндрических слоя: $V(R_{in}, R_{out}, z_{down}, z_{up})$ и $V_R(R_{split}, R_{out}, z_{down}, z_{up})$ в случае разбиения цилиндрической поверхностью (рис. 4*а*); или $V_L(R_{in}, R_{out}, z_{down}, z_{split})$ и $V_R(R_{in}, R_{out}, z_{split}, z_{up})$ при разбиении плоскостью (рис. 4*б*).

Полученные после разбиения области V_L и V_R соответствуют левому и правому узлам дерева. В узлы заносится информация о примитивах, которые хотя бы частично содержатся в соответствующих областях.

Дальнейшее разбиение производится рекурсивно для каждого из полученных цилиндрических слоёв, пока не будет выполнено условие остановки. Выбор типа и значения параметра разделяющей поверхности для каждого шага алгоритма определяется конкретной реализацией.

2.2. Поиск пересечений луча и примитивов в k-мерном дереве

Для поиска пересечений луча с коническими примитивами, упорядоченными с помощью описанной выше процедуры, необходимо совершить рекурсивный обход построенного k-мерного дерева. На каждом шаге алгоритма «посещаются» только те узлы дерева, которым соответствуют цилиндрические слои, пересекаемые лучом. Конечные узлы, достигнутые при обходе дерева, содержат списки только тех примитивов, с которыми в принципе может пересечься луч. Для каждого из этих примитивов выполняется проверка условия пересечения и при необходимости рассчитывается лучевой параметр точки пересечения t_k, $k = 1,\ldots,M$. Ближайшая точка соответствует минимальному лучевому параметру $t = \min t_k$.

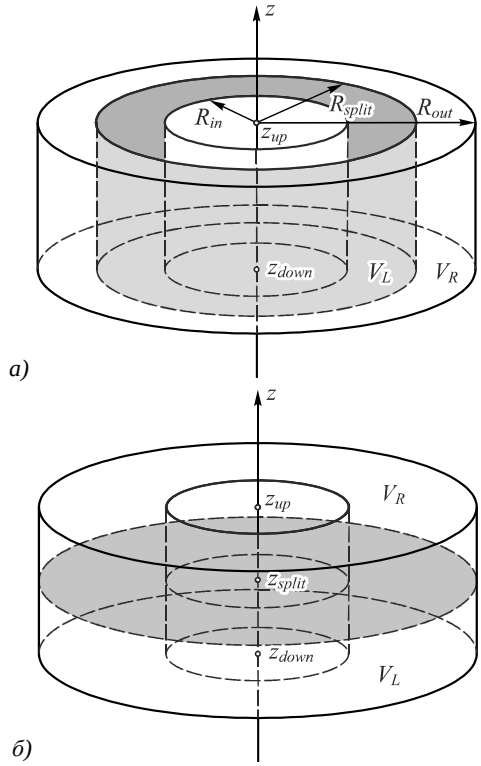

Рис. 4. Разбиение цилиндрического слоя в процессе построения k-мерного дерева: а) цилиндрической поверхностью; б) плоскостью

2.3. Поиск пересечений луча и примитивов в k-мерном дереве

Для поиска пересечений луча с коническими примитивами, упорядоченными с помощью описанной выше процедуры, необходимо совершить рекурсивный обход построенного k-мерного дерева. На каждом шаге алгоритма «посещаются» только те узлы дерева, которым соответствуют цилиндрические слои, пересекаемые лучом. Конечные узлы, достигнутые при обходе дерева, содержат списки только тех примитивов, с которыми в принципе может пересечься луч. Для каждого из этих примитивов выполняется проверка условия пересечения и при необходимости рассчитывается лучевой параметр точки пересечения t_k, $k = 1,\ldots,M$. Ближайшая точка соответствует минимальному лучевому параметру $t = \min t_k$.

Рассмотрим отдельный шаг рекурсивного алгоритма обхода дерева. На каждом шаге мы работаем с текущим цилиндрическим слоем $V(R_{in}, R_{out}, z_{down}, z_{up})$,

разбитым на вложенные области V_L, V_R. Найдём все точки пересечения луча с границами этого цилиндрического слоя. Точки пересечения луча с любой из цилиндрических поверхностей определяются из следующего квадратного уравнения:

$$\begin{cases} At^2 + Bt + C = 0, \\ A = s_x^2 + s_y^2, \\ B = 2\left(x_0 s_x + y_0 s_y\right), \\ C = x_0^2 + y_0^2 - R^2, \\ \dfrac{z_{down} - z_0}{s_z} \le t \le \dfrac{z_{up} - z_0}{s_z}, \end{cases} \quad (4)$$

а точка пересечения луча с любым из плоских колец определяется решением линейного уравнения:

$$\begin{cases} t = (z_f - z_0)/s_z, \\ R_{in} \le \left(x_0 + ts_x\right)^2 + \left(y_0 + ts_y\right)^2 \le R_{out}. \end{cases} \quad (5)$$

Решение двух уравнений типа (6) и двух уравнений типа (9) даёт до четырёх различных лучевых параметров, соответствующих точкам пересечения луча с границами цилиндрического слоя. Для определения потомков V_L, V_R, в которых необходимо продолжить обход дерева, проанализируем полученные точки пересечения и их расположение относительно разбивающей поверхности.

В случае разбиения слоя плоскостью необходимо вычислить координаты $z_i = z_0 + t_i s_z$ всех точек пересечения. Если все значения z_i меньше z_{split}, то луч пересекает только цилиндрический слой V_L и обход необходимо продолжить через левого потомка. Если все значения z_i больше координаты z_{split}, то пересечение произошло только со слоем V_R и обход необходимо продолжить через правого потомка. В остальных случаях выполняется ветвление обхода по обоим потомкам V_L, V_R.

В случае разбиения слоя цилиндрической поверхностью необходимо выполнить сравнение радиальных координат точек пересечения со значением R_{split}. Если все радиальные координаты меньше значения R_{split}, обход продолжается по левому потомку V_L, если больше – то по правому потомку V_R. В противном случае выполняется ветвление обхода дерева по обоим потомкам. Отметим, что для уменьшения вычислительных затрат имеет смысл выполнять сравнение не радиальных координат, а их квадратов, что позволит избежать ресурсоёмкой операции вычисления квадратного корня.

Обход дерева прекращается при отсутствии пересечений луча с текущим цилиндрическим слоем или по достижении самого нижнего в иерархии слоя. В последнем случае выполняется последовательная проверка и вычисление точек пересечения для всех примитивов, содержащихся в данном слое.

3. Экспериментальная проверка метода

Приведённый в данной работе метод поиска пересечения луча с осесимметричной поверхностью был реализован на языке программирования C++ и интегрирован в процедуру трассировки лучей Монте–Карло, разработанную в программной среде MATLAB®. Вычисления производились на компьютере с процессором Intel Core i7-3770, при этом использовалось только одно вычислительное ядро.

Для проведения численного эксперимента были рассчитаны и промоделированы следующие оптические элементы:

- аксикон, представленный одним коническим примитивом;
- элемент с двумя асферическими поверхностями [15], формирующий равномерное распределение освещённости в круглой области с радиусом 2 м на расстоянии 3 м (рис. 5);
- элемент с поверхностью, работающей по принципу полного внутреннего отражения (TIR-линза) [14], формирующий равномерное распределение освещённости в круглой области с радиусом 375 мм на расстоянии 1000 мм (рис. 6).

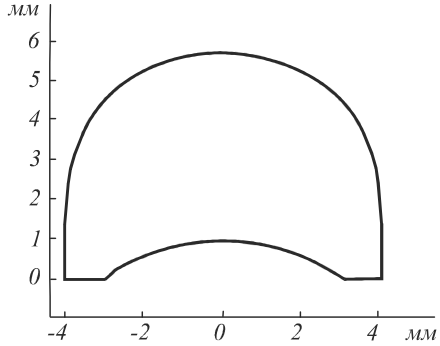

Рис. 5. Оптический элемент с двумя асферическими поверхностями

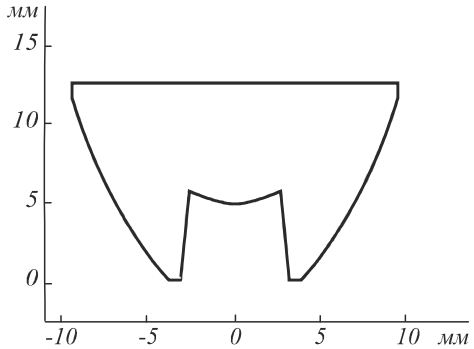

Рис. 6. Оптический элемент с поверхностью, работающей по принципу полного внутреннего отражения

В табл. 1 представлены сводные данные по времени трассировки 500 000 лучей от протяжённого источника 1×1 мм через представленные оптические элементы, аппроксимированные с точностью 1 мкм как треугольными, так и коническими примитивами. Для всех трёх примеров аппроксимация осесимметричной поверхности с помощью усечённых конусов обеспечила многократный прирост скорости трассировки (от 3 до 12 раз).

Табл. 2. Результаты моделирования оптических элементов

Оптический элемент	Время трассировки с коническими примитивами		Количество конических примитивов	Время трассировки с треугольными примитивами, с	Количество треугольных примитивов
	k-мерное дерево, с	последовательный поиск, с			
Аксикон	0,931	0,931	1	11,378	994
Две асферические поверхности	4,618	8,905	217	13,059	150 030
TIR-линза	4,814	6,602	125	30,355	123 246

Оптический элемент, изображённый на рис. 6, также был промоделирован в коммерческом программном обеспечении TracePro®. Время трассировки в TracePro® для 500 000 лучей составило 205 секунд, при этом среднеквадратичное отклонение распределения освещённости, полученного разработанным методом (рис. 7*а*), от распределения, полученного в TracePro (рис. 7*б*), составляет всего лишь 2,9 %, что свидетельствует о работоспособности предложенного метода трассировки лучей.

Заключение

В данной работе представлена модификация метода трассировки лучей Монте–Карло для осесимметричных поверхностей. Модификация основана на аппроксимации поверхностей набором усечённых конусов и последующем их упорядочивании в пространстве с помощью k-мерного дерева с цилиндрическими слоями. Данная модификация позволяет существенно сократить время трассировки лучей (от 3 до 12 раз для приведённых в работе примеров), при этом повышается точность аппроксимации поверхности по сравнению с традиционными способами представления поверхности треугольными примитивами.

Благодарности

Исследование выполнено за счёт гранта Российского научного фонда, проект № 14-19-00969.

Литература

1. **Lafortune, E.** Mathematical models and Monte Carlo algorithms for physically based rendering / E. Lafortune // Department of Computer Science, Faculty of Engineering, Katholieke Universiteit Leuven. – 1996.
2. **Hokr, B.H.** Modeling focusing Gaussian beams in a turbid medium with Monte Carlo simulations / B.H. Hokr, J.N. Bixler, G. Elpers, B. Zollars, R.J. Thomas, V.V. Yakovlev, M.O. Scully // Optics Express. – 2015. – Vol. 23(7). – P. 8699-8705.
3. **Zhdanov, D.D.** Indeterminate ray tracing in problems of the analysis of light scattering and the design of illuminating systems / D.D. Zhdanov, A.A. Garbul, V.A. Maïorov, I.S. Potemin, V.G. Sokolov // Journal of Optical Technology. – 2014. – Vol. 81(6). – P. 322-326.
4. **Boissonnat, J.D.** Provably good sampling and meshing of surfaces / J.D. Boissonnat, S. Oudot // Graphical Models. – 2005. – Vol. 67(5). – P. 405-451.
5. **Rineau, L.** A generic software design for Delaunay refinement meshing / L. Rineau, M. Yvinec // Computational Geometry. – 2007. – Vol. 38(1). – P. 100-110.
6. **Möller, T.** Fast, minimum storage ray-triangle intersection / T. Möller, B. Trumbore // Journal of Graphics Tools. – 1997. – Vol. 2(1). – P. 21-28.

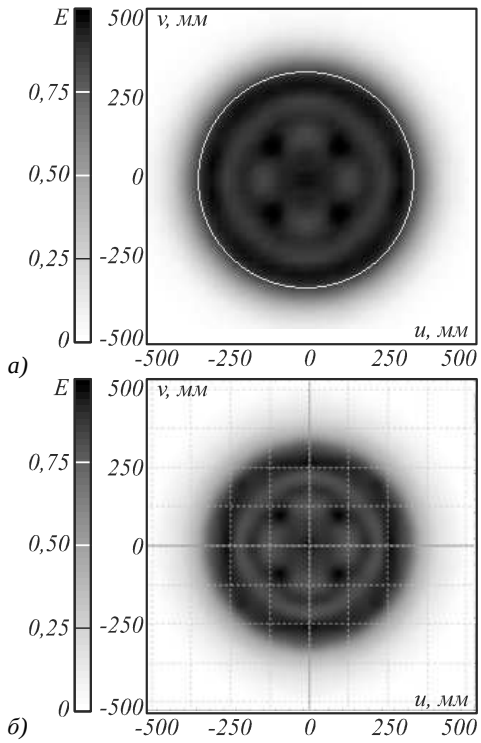

Рис. 7. Распределения освещённости, полученные в результате моделирования оптического элемента с поверхностью, работающей по принципу полного внутреннего отражения: а) предложенным методом; б) в коммерческом программном обеспечении TracePro®

7. **Meagher, D.** Geometric modeling using octree encoding / D. Meagher // Computer Graphics and Image Processing. – 1982. – Vol. 19(2). – P. 129-147.
8. **Gunther, J.** Realtime Ray Tracing on GPU with BVH-based Packet Traversal / J. Gunther, S. Popov, H.P. Seidel, P. Slusallek // Proceedings of the IEEE Symposium on Interactive Ray Tracing. – 2007. – P. 113-118.
9. **Bentley, J.L.** Multidimensional binary search trees used for associative searching / J.L. Bentley // Communications of the ACM. – 1975. – Vol. 18(9). – P. 509-517.
10. **Havran, V.** Fast robust BSP tree traversal algorithm for ray tracing / V. Havran, T. Kopal, J. Bittner, J. Žára // Journal of Graphics Tools. – 1997. – Vol. 2(4). – P. 15-23.
11. **Havran, V.** Heuristic ray shooting algorithms / V. Havran // Faculty of Electrical Engineering, Czech Technical University – 2000.
12. **Wald, I.** On building fast kd-trees for ray tracing, and on doing that in O (N log N) / I. Wald, V. Havran // Interactive Ray Tracing. – 2006. – P. 61-69.
13. **MacDonald, J.D.** Heuristics for ray tracing using space subdivision/ J.D. MacDonald, K.S. Booth // The Visual Computer. – 1990. – Vol. 6(3). – P. 153-166.
14. **Moiseev, M. A.** Fast and robust technique for design of axisymmetric TIR optics in case of an extended light source /

M.A. Moiseev, L.L. Doskolovich, K.V. Borisova, E.V. Byzov // Journal of Modern Optics. – 2013. – Vol. 60(14). – P. 1100-1106.

15. **Кравченко, С.В.** Расчёт осесимметричных оптических элементов с двумя асферическими поверхностями для формирования заданных распределений освещённости / С.В. Кравченко, М.А. Моисеев, Л.Л. Досколович, Н.Л. Казанский // Компьютерная оптика. – 2011. – Т. 35, № 4. – С. 467–472.

References

[1] Lafortune E. Mathematical models and Monte Carlo algorithms for physically based rendering. Department of Computer Science, Faculty of Engineering, Katholieke Universiteit Leuven 1996.

[2] Hokr BH, Bixler JN, Elpers G, Zollars B, Thomas RJ, Yakovlev VV, Scully MO. Modeling focusing Gaussian beams in a turbid medium with Monte Carlo simulations. Optics Express 2015; 23(7): 8699-705.

[3] Zhdanov DD, Garbul AA, Maĭorov VA, Potemin IS, Sokolov VG. Indeterminate ray tracing in problems of the analysis of light scattering and the design of illuminating systems. Journal of Optical Technology 2014; 81(6): 322-26.

[4] Boissonnat JD, Oudot S. Provably good sampling and meshing of surfaces. Graphical Models 2005; 67(5): 405-51.

[5] Rineau L, Yvinec M. A generic software design for Delaunay refinement meshing. Computational Geometry 2007; 38(1): 100-10.

[6] Möller T, Trumbore B. Fast, minimum storage ray-triangle intersection. Journal of Graphics Tools 1997; 2(1): 21-8.

[7] Meagher D. Geometric modeling using octree encoding. Computer Graphics and Image Processing 1982; 19(2): 129-47.

[8] Gunther J, Popov S, Seidel HP, Slusallek P. Realtime Ray Tracing on GPU with BVH-based Packet Traversal. Proceedings of the IEEE Symposium on Interactive Ray Tracing 2007; 1: 113-8.

[9] Bentley JL. Multidimensional binary search trees used for associative searching. Communications of the ACM 1975; 18(9): 509-17.

[10] Havran V, Kopal T, Bittner J, Žára J. Fast robust BSP tree traversal algorithm for ray tracing. Journal of Graphics Tools 1997; 2(4): 15-23.

[11] Havran V. Heuristic ray shooting algorithms. Faculty of Electrical Engineering, Czech Technical University 2000.

[12] Wald I, Havran V. On building fast kd-trees for ray tracing, and on doing that in O (N log N). Interactive Ray Tracing 2006; 1: 61-9.

[13] MacDonald JD, Booth KS. Heuristics for ray tracing using space subdivision. The Visual Computer 1990; 6(3): 153-66.

[14] Moiseev MA, Doskolovich LL, Borisova KV, Byzov EV. Fast and robust technique for design of axisymmetric TIR optics in case of an extended light source. Journal of Modern Optics 2013; 60(14): 1100-6.

[15] Kravchenko SV, Moiseev MA, Doskolovich LL, Kazanskiy NL. Design of axis-symmetrical optical element with two aspherical surfaces for generation of prescribed irradiance distribution [In Russian]. Computer Optics 2011; 35(4): 467–72.

MONTE-CARLO RAY TRACING FOR AXISYMMETRICAL OPTICAL ELEMENTS

E.S. Andreev[1,2], M.A. Moiseev[1,2], K.V. Borisova[1,2], L.L. Doskolovich[1,2]
[1] Image Processing Systems Institute, Samara, Russia,
Russian Academy of Sciences,
[2] Samara State Aerospace University, Samara, Russia

Abstract

A modification of the Monte-Carlo ray tracing procedure for axisymmetric surfaces is proposed. The main idea consists in the approximation of optical surfaces by truncated cones and the use of an unconventional k-d tree. The simulation results show high performance of the proposed method, with the ray tracing procedure working 3-12 times faster than conventional algorithms based on a triangle approximation.

Keywords: Monte-Carlo ray tracing, path tracing, geometric optics, computation methods, systems with special symmetry.

Citation: Andreev ES, Moiseev MA, Borisova KV, Doskolovich LL. Monte-Carlo ray tracing for axisymmetrical optical elements. Computer Optics 2015; 39(3): 357-2.

Сведения об авторах

Андреев Евгений Сергеевич, 1992 года рождения. В 2014 году с отличием окончил Самарский государственный аэрокосмический университет имени академика С.П. Королёва (СГАУ) по направлению «Прикладные математика и физика». Магистрант первого года обучения магистерской программы «Математическое моделирование и информационные технологии в фотонике», СГАУ.

E-mail: *gsomix@gmail.com* .

Evgeniy Sergeevich Andreev (b. 1992) graduated with honors (2014) from Samara State Aerospace University (SSAU), majoring in Applied Mathematics and Physics. First year master's student with major in "Mathematical Modelling and Information Technologies in Photonics" at SSAU.

*Сведения об авторах **Досколович Леонид Леонидович** и **Моисеев Михаил Александрович** – см. стр. 345 этого номера.*

*Сведения об авторе **Борисова Ксения Валерьевна** – см. стр. 356 этого номера.*

Поступила в редакцию 2 июля 2015 г.
Окончательный вариант – 9 июля 2015 г.