Systems and Methods for Graphical Rendering

5-10-2011

Amar Mukherjee  
*University of Central Florida*

Weifeng Sun  
*University of Central Florida*

Find similar works at: [https://stars.library.ucf.edu/patents](https://stars.library.ucf.edu/patents)

University of Central Florida Libraries [http://library.ucf.edu](http://library.ucf.edu)

Recommended Citation

[https://stars.library.ucf.edu/patents/566](https://stars.library.ucf.edu/patents/566)

This Patent is brought to you for free and open access by the Technology Transfer at STARS. It has been accepted for inclusion in UCF Patents by an authorized administrator of STARS. For more information, please contact lee.dotson@ucf.edu.
A method of rendering a graphical scene includes determining a plurality of functions that contribute to a light transport model of a scene, factoring the plurality of functions into a set of fixed functions and one varying function, determining a first radiance transfer vector that represents the product of the fixed functions in the wavelet domain, determining a second radiance transfer vector that represents the one varying function in the wavelet domain, determining an inner product of the first and second radiance transfer vectors to approximate a radiance of a point x in the scene, and rendering the scene.

19 Claims, 25 Drawing Sheets
OTHER PUBLICATIONS

Kontkanen, et al., "Wavelet Radiance Transport for Interactive Indirect Lighting", Helsinki University of Technology; The Eurographics Association 2006.

* cited by examiner
START

PROJECTING A PLURALITY OF FUNCTIONS INTO THE WAVELET DOMAIN

REPRESENTING THE INTEGRAL AS THE SUM OF A SERIES OF CONTRIBUTING PRODUCTS THAT INCLUDE BASIS COEFFICIENTS AND ONE INTEGRAL COEFFICIENT

PLACING BASIS FUNCTIONS IN A BASIS FUNCTION TREE AND PLACING BASIS COEFFICIENTS OF EACH FUNCTION IN A BASIS COEFFICIENT TREE

DETERMINING THE INTEGRAL COEFFICIENT APPEARING IN EACH CONTRIBUTING PRODUCT

ELIMINATING THE CONTRIBUTING PRODUCT FROM THE SUM OF THE SERIES OF CONTRIBUTING PRODUCTS IF THE INTEGRAL COEFFICIENT IS ZERO

ADDING THE CONTRIBUTING PRODUCT TO THE SUM OF THE SERIES OF CONTRIBUTING PRODUCTS IF INTEGRAL COEFFICIENT IS NONZERO

END

FIG. 1
FIG. 2
Equation (2)  

\[ F_i(\nu) = \sum_{j=1}^{M} [f_{ij} \cdot b_j(\nu)] \]

FIG. 3

FIG. 4

FIG. 5

FIG. 6
FIG. 8

<table>
<thead>
<tr>
<th>Parity of the number of $\psi_1$</th>
<th>Even</th>
<th>Odd</th>
<th>Odd</th>
<th>Even</th>
<th>Even</th>
<th>Odd</th>
<th>Even</th>
<th>Odd</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parity of the number of $\psi_2$</td>
<td>Even</td>
<td>Odd</td>
<td>Even</td>
<td>Odd</td>
<td>Even</td>
<td>Odd</td>
<td>Even</td>
<td>Odd</td>
</tr>
<tr>
<td>Parity of the number of $\psi_3$</td>
<td>Even</td>
<td>Odd</td>
<td>Even</td>
<td>Odd</td>
<td>Even</td>
<td>Odd</td>
<td>Even</td>
<td></td>
</tr>
<tr>
<td>Basis function type of $P^N$</td>
<td>$\phi$</td>
<td>$\psi_1$</td>
<td>$\psi_2$</td>
<td>$\psi_3$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

FIG. 11
Equation (11)

\[ t \cdot \square = \pm |p^2| \cdot \square \]

\[ p^2 \]

FIG. 9
Equation (17)

\[ \pm \frac{1}{p^g} \]

Equation (18)

\[ \pm \frac{1}{p^{g-1}} \]

\[ \pm \frac{1}{p^g} \]

FIG. 10
DO THE BASIS FUNCTIONS LIE IN NODES ON A SINGLE DIRECT PATH THROUGH THE BASIS FUNCTION TREE?

NO

PRODUCT OF THE BASIS FUNCTIONS IS ZERO

INTEGRAL COEFFICIENT OF THE BASIS FUNCTIONS IS ZERO

PRODUCT OF THE BASIS FUNCTIONS IS NONZERO

YES

DETERMINE A BASIS FUNCTION TYPE OF A BASIS FUNCTION THAT APPEARS IN THE PRODUCT OF THE BASIS FUNCTIONS

IS THE TYPE OF THE BASIS FUNCTION SCALING?

NO

INTEGRAL COEFFICIENT OF THE BASIS FUNCTIONS IS ZERO

YES

INTEGRAL COEFFICIENT OF THE BASIS FUNCTIONS IS NONZERO; SEE EQ. (22)

FIG. 12
PLACING BASIS FUNCTIONS THAT APPEAR IN A PRODUCT IN A RANKING ORDER

DETERMINING FROM THE BASIS FUNCTIONS THAT APPEAR IN THE PRODUCT A SUBSET OF BASIS FUNCTIONS HAVING A FINEST SCALE

COUNTING AN AGGREGATE NUMBER OF TIMES EACH OF THREE WAVELET FUNCTION TYPES APPEARS IN THE SUBSET

COMPARING A PARITY OF EACH OF THE THREE AGGREGATE NUMBERS

IS THE PARITY OF EACH AGGREGATE NUMBER THE SAME?

NO

THE BASIS FUNCTION TYPE IS ONE OF THE WAVELET BASIS FUNCTIONS

YES

THE BASIS FUNCTION TYPE IS A SCALING BASIS FUNCTION

FIG. 13
Equation (22)

\[ C^N = \int \frac{1}{1} \sum_{j} \left[ |p^{N-j}| b_{pN-1} \cdot b_N \right] \]

<table>
<thead>
<tr>
<th>Case</th>
<th>( C^N ) is non-zero if:</th>
<th>( C^N ) is:</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Basis functions ( b_{pN-1} ) and ( b_N ) are both wavelet basis functions of the same type and support</td>
<td>( C^N =</td>
</tr>
<tr>
<td>2</td>
<td>Basis functions ( b_{pN-1} ) and ( b_N ) are both the mother scaling function</td>
<td>( C^N = 1 )</td>
</tr>
<tr>
<td>3</td>
<td>Basis functions ( b_{pN-1} ) is a scaling basis function and ( b_N ) is one of its parent basis functions</td>
<td>( C^N = \pm 2^j \times C^{N-1} ) ( j ) is the scale of ( b_N )</td>
</tr>
<tr>
<td>4</td>
<td>Basis function ( b_{pN-1} ) is a scaling basis function and ( b_N ) is a wavelet basis function of the same support</td>
<td>( C^N = 0 )</td>
</tr>
<tr>
<td>5</td>
<td>Basis function ( b_{pN-1} ) is a wavelet basis function and ( b_N ) is a child scaling basis function of the same support</td>
<td>( C^N = 0 )</td>
</tr>
<tr>
<td>6</td>
<td>Basis function ( b_{pN-1} ) is a wavelet basis function and ( b_N ) is a wavelet basis function of different type and the same support</td>
<td>( C^N = 0 )</td>
</tr>
<tr>
<td>7</td>
<td>Basis function ( b_{pN-1} ) is a wavelet basis function and ( b_N ) is one of its parent basis functions.</td>
<td>( C^N = 0 )</td>
</tr>
</tbody>
</table>
PROJECTING A PLURALITY OF FUNCTIONS INTO THE WAVELET DOMAIN

ENCODING BASIS COEFFICIENTS OF EACH FUNCTION IN A WAVELET TREE

PERFORMING NON-LINEAR APPROXIMATION ON EACH FUNCTION

TRAVERSING DIRECT PATHS THROUGH THE WAVELET TREES WHERE AN INTEGRAL COEFFICIENT MAY BE NON-ZERO TO DETERMINE THE INTEGRAL OF THE PRODUCT OF THE FUNCTIONS

FIG. 16
struct augtree
  dc: mother scaling coefficient;
  node: pointer to struct augnode;
end

struct augnode
  ψ[3]: wavelet coefficients;
  parentsum: signed parent summation;
  ch[4]: child pointers;
end

FIG. 17

START

INITIALLY INCREMENTING THE INTEGRAL WITH CONTRIBUTIONS OF ROOTS OF THE WAVELET TREES

SYNCHRONOUSLY PROCESSING A SET OF NODES

DETERMINING A CONTRIBUTION OF THE SET OF NODES

INCREMENTING THE INTEGRAL WITH THE CONTRIBUTION OF THE SET OF NODES

SYNCHRONOUSLY PROCESSING A CHILD SET OF NODES

END

FIG. 18
algorithm FunctionProductIntegral (W1, W2, ..., WN);
    u = W1.dc x W2.dc x ... x WN.dc;
    traverseAugTrees (1, W1.node, W2.node, ..., WN.node);
end

routine traverseAugTrees (cum, w1, w2, ..., wN);
    Rank w1, ..., wN so that w1, ..., wk are not null and wk+1, ..., wN are null;
    if k < 2 then return;
    for i = 1 to N, update wi.parentsum;
    cum = cum x [ ΠN
    u = u + [cum x getProductIntegral (w1, ..., wk)];
    for i = 0 to 3
        traverseAugTrees (cum, w1.ch[i], ... w_k.ch[i]);
end

routine getProductIntegral (w1, ..., w_m)
    if m = 1 return 0;
    return [(w_m.parentsum x getProductIntegral (w1, ..., w_{m-1}))
            + (w_m.ψ[0] x getWaveletProduct (0, 1, 2, w1, ..., w_{m-1}))
            + (w_m.ψ[1] x getWaveletProduct (1, 0, 2, w1, ..., w_{m-1}))
            + (w_m.ψ[2] x getWaveletProduct (2, 0, 1, w1, ..., w_{m-1}))];
end

routine getWaveletProduct (a, b, c, w1, ..., w_m)
    if m = 1 return w1.ψ[a];
    return [(w_m.ψ[a] x ΠN
            + (w_m.parentsum x getWaveletProduct (a, b, c, w1, ..., w_{m-1}))
            + (2^a x w_m.ψ[a] x getProductIntegral (w1, ..., w_{m-1}))
            + (2^b x w_m.ψ[b] x getWaveletProduct (c, b, a, w1, ..., w_{m-1}))
            + (2^c x w_m.ψ[c] x getWaveletProduct (b, a, c, w1, ..., w_{m-1}))];
end

FIG. 19
algorithm FunctionProductIntegral ($W_1$, $W_2$, ..., $W_N$);
    $u = W_1.dc \times W_2.dc \times \ldots \times W_N.dc$;
    traverseAugTrees (1, $W_1$.node, $W_2$.node, ..., $W_N$.node);
end

routine traverseAugTrees (cum, $w_1$, $w_2$, ..., $w_N$);
    Rank $w_1$, ..., $w_N$ so that $w_1$, ..., $w_k$ are not null and $w_{k+1}$, ..., $w_N$ are null;
    if $k < 2$ then return;
    for $i = 1$ to $N$, update $w_i$.parentsum;
    cum = cum \times \prod_{i=1}^{k} w_i$.parentsum;
    $u = u + \left[ cum \times \text{getProductIntegral} (w_1, \ldots, w_k) \right]$;
    for $i = 0$ to $3$
        traverseAugTrees (cum, $w_1$.ch[i], ..., $w_k$.ch[i]);
end

routine getProductIntegral ($w_1$, ..., $w_m$)
    if $m = 1$ return 0;
    $F'[1].\{\phi, \psi[0], \psi[1], \psi[2], \text{cum}\}$
    = \{0, $w_1$.\psi[0], $w_1$.\psi[1], $w_1$.\psi[2], $w_1$.parentsum\}
    for $i = 2$ to ($m - 1$)
        $F'[i].\text{cum} = w_i$.parentsum \times $F'[i - 1].\text{cum}$;
        $F'[i].\phi$ = getP($i$);
        $F'[i].\psi[0]$ = getW (0, 1, 2, $i$);
        $F'[i].\psi[1]$ = getW (1, 0, 2, $i$);
        $F'[i].\psi[2]$ = getW (2, 0, 1, $i$);
    return getP($m$);
end

routine getW ($a$, $b$, $c$, $i$)
    if $i = 1$ return $F'[1].\psi[a]$;
    return \[
    (w_i\.parentsum \times F'[i - 1].\phi +
    (w_i\.\psi[0] \times F'[i - 1].\psi[0])
    + (w_i\.\psi[1] \times F'[i - 1].\psi[1])
    + (w_i\.\psi[2] \times F'[i - 1].\psi[2]))
    \]
end

routine getP ($i$)
    if $i = 1$ return 0;
    return \[
    (w_i\.parentsum \times F'[i - 1].\phi +
    (w_i\.\psi[0] \times F'[i - 1].\psi[0])
    + (w_i\.\psi[1] \times F'[i - 1].\psi[1])
    + (w_i\.\psi[2] \times F'[i - 1].\psi[2]))
    \]
end

FIG. 20
FIG. 21

```
1 struct table T[N]
2   \phi: magnitude;
3   \psi[0]: magnitude;
4   \psi[1]: magnitude;
5   \psi[2]: magnitude;
6   cum: cumulative parentsum;
7 end
```

FIG. 22

SYSTEM

LOGIC CONFIGURED TO PROJECT A PLURALITY OF FUNCTIONS INTO A WAVELET DOMAIN

LOGIC CONFIGURED TO ENCODE BASIS COEFFICIENTS OF EACH FUNCTION IN A WAVELET TREE

LOGIC CONFIGURED TO PERFORM NON-LINEAR APPROXIMATION ON EACH FUNCTION

LOGIC CONFIGURED TO TRAVERSE DIRECT PATHS THROUGH THE WAVELET TREES TO DETERMINE AN INTEGRAL OF THE PRODUCT OF THE FUNCTIONS
FIG. 23
DETERMINING A PLURALITY OF FUNCTIONS THAT CONTRIBUTE TO A LIGHT TRANSPORT MODEL OF A SCENE

PROJECTING THE PLURALITY OF FUNCTIONS INTO THE WAVELET DOMAIN

ENCODING BASIS COEFFICIENTS OF EACH FUNCTION IN A WAVELET TREE

PERFORMING NON-LINEAR APPROXIMATION ON EACH FUNCTION

TRAVERSING DIRECT PATHS THROUGH THE WAVELET TREES WHERE AN INTEGRAL COEFFICIENT MAY BE NON-ZERO

RENDERING THE SCENE

FIG. 24
SYSTEM 2500

2502
LOGIC CONFIGURED TO
DETERMINE A PLURALITY OF FUNCTIONS THAT
CONTRIBUTE TO A LIGHT TRANSPORT MODEL OF A SCENE

2504
LOGIC CONFIGURED TO
PROJECT THE PLURALITY OF FUNCTIONS
INTO A WAVELET DOMAIN

2506
LOGIC CONFIGURED TO
ENCODE BASIS COEFFICIENTS OF EACH FUNCTION IN
A WAVELET TREE

2508
LOGIC CONFIGURED TO
PERFORM NON-LINEAR APPROXIMATION ON EACH FUNCTION

2510
LOGIC CONFIGURED TO
TRAVERSE DIRECT PATHS THROUGH THE WAVELET TREES

2512
LOGIC CONFIGURED TO
RENDER THE SCENE

FIG. 25
START

FACTORING A PLURALITY OF FUNCTIONS INTO A SET OF FIXED FUNCTIONS AND ONE VARYING FUNCTION

DETERMINING A FIRST VECTOR THAT REPRESENTS THE PRODUCT OF THE FIXED FUNCTIONS IN THE WAVELET DOMAIN

DETERMINING A SECOND VECTOR THAT REPRESENTS THE ONE VARYING FUNCTION IN THE WAVELET DOMAIN

DETERMINING AN INNER PRODUCT OF THE FIRST VECTOR AND THE SECOND VECTOR

END

FIG. 26
START

PROJECTING A PLURALITY OF FUNCTIONS INTO THE WAVELET DOMAIN

ENCODING BASIS COEFFICIENTS OF EACH FUNCTION IN A WAVELET TREE

PERFORMING NON-LINEAR APPROXIMATION ON EACH FUNCTION

TRAVERSING DIRECT PATHS THROUGH THE WAVELET TREES WHERE AN INTEGRAL COEFFICIENT MAY BE NON-ZERO TO DETERMINE BASIS COEFFICIENTS OF A VECTOR

END

FIG. 27
INITIALLY INCREMENTING A BASIS COEFFICIENT IN A ROOT OF AN OUTPUT WAVELET TREE WITH CONTRIBUTIONS OF ROOTS OF INPUT WAVELET TREES

SYNCHRONOUSLY PROCESSING A SET OF NODES OF THE INPUT WAVELET TREES

DETERMINING CONTRIBUTIONS OF THE SET OF NODES TO BASIS COEFFICIENTS OF OUTPUT NODES OF THE OUTPUT WAVELET TREE

INCREMENTING BASIS COEFFICIENTS OF THE OUTPUT NODES WITH THE CONTRIBUTION OF THE SET OF NODES FROM THE INPUT WAVELET TREES

SYNCHRONOUSLY PROCESSING A CHILD SET OF NODES

END
algorithm FunctionProduct(W₀, W₁, W₂, ..., Wₙ₋₁)
  W₀.dc = W₁.dc x W₂.dc x ... x Wₙ₋₁.dc;
  getCoefficients(1, W₀.node, W₁.node, W₂.node, ..., Wₙ₋₁.node);
end

routine getCoefficients (cum, w₀, w₁, w₂, ..., wₙ₋₁)
  Rank w₁, ..., wₙ₋₁ so that w₁, ..., wₖ are not null and wₖ₊₁, ..., wₙ₋₁ are null;
  if k = 0 then return;
  for i = 1 to (N - 1), update wᵢ.parentsum;
  cum = cum x \[Π}_{i=1}^{N-1} \text{w}_i.parentsum\];
  updateParents (w₀, cum x getProductIntegral(w₁, ..., wₖ));
  w₀.ψ[0] = cum x getW (0, 1, 2, k);
  w₀.ψ[1] = cum x getW (1, 0, 2, k);
  w₀.ψ[2] = cum x getW (2, 0, 1, k);
  for i = 0 to 3
    getCoefficients (cum, w₀.ch[i], w₁.ch[i], ..., wₖ.ch[i]);
end

routine updateParents (w₀, val)
  w₀.dc = w₀.dc + val;
  for scale j = 0, 1, ..., w₀.scale-1
    // lies in the quadrant (k, l) of its parent w₀ at scale j
    w₀.ψ[0] = w₀.ψ[0] + (sign (0, k, l) x 2² x val);
    w₀.ψ[1] = w₀.ψ[1] + (sign (1, k, l) x 2² x val);
    w₀.ψ[2] = w₀.ψ[2] + (sign (2, k, l) x 2² x val);
end

routine getProductIntegral (w₁, ..., wₙ)
  if m = 1 return 0;
  \( \mathcal{F}[1].\{ϕ, ψ[0], ψ[1], ψ[2], \text{cum}\} = \{0, w₁.ψ[0], ψ[1], ψ[2], w₁.parentsum\} \)
  for i = 2 to (m - 1)
    \( \mathcal{F}[i].\text{cum} = wᵢ.parentsum x \mathcal{F}[i-1].\text{cum}; \)
    \( \mathcal{F}[i].ϕ = \text{getP}(i); \)
    \( \mathcal{F}[i].ψ[0] = \text{getW}(0, 1, 2, i); \)
    \( \mathcal{F}[i].ψ[1] = \text{getW}(1, 0, 2, i); \)
    \( \mathcal{F}[i].ψ[2] = \text{getW}(2, 0, 1, i); \)
  return getP(m);
end

routine getP (i)
  if i = 1 return \( \mathcal{F}[1].ψ[a] \);
  return \( wᵢ.parentsum x \mathcal{F}[i-1].ϕ + (wᵢ.ψ[0] x \mathcal{F}[i-1].ψ[0]) \)
  \+ \( (wᵢ.ψ[1] x \mathcal{F}[i-1].ψ[1]) + (wᵢ.ψ[2] x \mathcal{F}[i-1].ψ[2]) \);
end

routine getW (a, b, c, i)
  if i = 1 return \( \mathcal{F}[1].ψ[a] \);
  return \( wᵢ.ψ[a] x \mathcal{F}[i-1].\text{cum} + (wᵢ.parentsum x \mathcal{F}[i-1].ψ[a]) \)
  \+ \( (4^i x wᵢ.ψ[a] x \mathcal{F}[i-1].ϕ) \)
  \+ \( (2^i x (wᵢ.ψ[b] x \mathcal{F}[i-1].ψ[c]) + wᵢ.ψ[c] x \mathcal{F}[i-1].ψ[b]) \);
end

FIG. 29
SYSTEM 3000

3002
LOGIC CONFIGURED TO FACTOR A PLURALITY OF FUNCTIONS INTO A SET OF FIXED FUNCTIONS AND ONE VARYING FUNCTION

3004
LOGIC CONFIGURED TO DETERMINE A FIRST VECTOR THAT REPRESENTS THE PRODUCT OF THE FIXED FUNCTIONS IN THE WAVELET DOMAIN

3006
LOGIC CONFIGURED TO DETERMINE A SECOND VECTOR THAT REPRESENTS THE ONE VARYING FUNCTION IN THE WAVELET DOMAIN

3008
LOGIC CONFIGURED TO DETERMINE AN INNER PRODUCT OF THE FIRST VECTOR AND THE SECOND VECTOR

3012
LOGIC CONFIGURED TO PROJECT THE FUNCTIONS OF THE SET OF FIXED FUNCTIONS INTO THE WAVELET DOMAIN

3014
LOGIC CONFIGURED TO ENCODE BASIS COEFFICIENTS OF EACH FIXED FUNCTION IN A WAVELET TREE

3016
LOGIC CONFIGURED TO PERFORM NON-LINEAR APPROXIMATION ON FIXED EACH FUNCTION

3018
LOGIC CONFIGURED TO TO DETERMINE BASIS COEFFICIENTS OF A VECTOR BY TRAVERSING DIRECT PATHS THROUGH THE WAVELET TREES

FIG. 30
DETERMINING A PLURALITY OF FUNCTIONS THAT CONTRIBUTE TO A LIGHT TRANSPORT MODEL OF A SCENE

FACTORING THE PLURALITY OF FUNCTIONS INTO A SET OF FIXED FUNCTIONS AND ONE VARYING FUNCTION

DETERMINING A FIRST RADIANCE TRANSFER VECTOR FOR THE FIXED FUNCTIONS

DETERMINING A SECOND RADIANCE TRANSFER VECTOR FOR THE VARYING FUNCTION

DETERMINING AN INNER PRODUCT OF THE FIRST AND SECOND RADIANCE TRANSFER VECTORS

RENDERING THE SCENE

FIG. 31
SYSTEM 3200

LOGIC CONFIGURED TO DETERMINE A PLURALITY OF FUNCTIONS THAT CONTRIBUTE TO A LIGHT TRANSPORT MODEL OF A SCENE

LOGIC CONFIGURED TO FACTOR THE PLURALITY OF FUNCTIONS INTO A SET OF FIXED FUNCTIONS AND ONE VARYING FUNCTION

LOGIC CONFIGURED TO DETERMINE A FIRST RADIANCE TRANSFER VECTOR FOR THE FIXED FUNCTIONS

LOGIC CONFIGURED TO DETERMINE A SECOND RADIANCE TRANSFER VECTOR FOR THE VARYING FUNCTION

LOGIC CONFIGURED TO DETERMINE AN INNER PRODUCT OF THE FIRST AND SECOND RADIANCE TRANSFER VECTORS

LOGIC CONFIGURED TO RENDER THE SCENE

FIG. 32
SYSTEMS AND METHODS FOR GRAPHICAL RENDERING

CROSS-REFERENCE TO RELATED APPLICATIONS

This application claims priority to copending U.S. provisional application entitled, "Generalized Wavelet Product Integral For Rendering Dynamic Glossy Objects," having Ser. No. 60/830,654, filed Jul. 13, 2006, which is entirely incorporated herein by reference.

STATEMENT REGARDING FEDERALLY SPONSORED RESEARCH OR DEVELOPMENT

This disclosure was made with government support under grant number 0312724 awarded by the National Science Foundation. The government has certain rights in the invention.

BACKGROUND

Mathematics is a powerful tool that can be used to create models, among other things. When a real world system is represented using a mathematical model, the solution to the mathematical model often represents an answer to a problem in the real world system. In some cases, due to the nature of the system, the mathematical model includes a series of functions that are multiplied together and integrated, or simply multiplied together. One example of such a mathematical model is a light transport model that represents the physics of light moving within a three-dimensional scene. The light transport model describes the radiance of objects in the scene as a function of parameters such as the viewpoint of the observer, the texture of the objects, and the lighting itself.

In cases in which the mathematical model is complex and time-consuming to solve, an approximation of the model can be employed to simplify, for example, rendering a graphical scene. However, the approximation may underestimate or ignore some variables of the model, and therefore the contribution of corresponding elements to the overall system. For example, most computer graphics rendering processes rely on simplified or approximated versions of the light transport model, but the lighting of the scenes rendered using such models is not realistic. Some simplified versions of the light transport model require objects in the scene to be static. Others cannot approximate the specular highlights that high-frequency lighting creates on glossy materials. Still others are physically accurate but are too slow for real-time rendering.

To date, a need exists for systems and methods for determining the integral of the product of a plurality of functions, or for determining the product of a plurality of functions. For example, such a need exists in the art of computer graphics rendering, where such systems and methods can be employed with reference to the light transport model.

SUMMARY

In one embodiment, a method of rendering a graphical scene includes determining a plurality of functions that contribute to a light transport model of a scene, factoring the plurality of functions into a set of fixed functions and one varying function, determining a first radiance transfer vector that represents the product of the fixed functions in the wavelet domain, determining a second radiance transfer vector that represents the one varying function in the wavelet domain, determining an inner product of the first and second radiance transfer vectors to approximate a radiance of a point x in the scene, and rendering the scene.

Other systems, devices, methods, features, and advantages of the disclosed systems and methods for determining the integral of the product of a plurality of functions will be apparent or will become apparent to one with skill in the art upon examination of the following figures and detailed description. All such additional systems, devices, methods, features, and advantages are intended to be included within the description and are intended to be protected by the accompanying claims.

BRIEF DESCRIPTION OF THE FIGURES

The present disclosure may be better understood with reference to the following figures. Matching reference numerals designate corresponding parts throughout the figures, and components in the figures are not necessarily to scale.

FIG. 1 is a block diagram illustrating an embodiment of a method for determining the integral of the product of a plurality of functions.

FIG. 2 is a diagram illustrating a two-dimensional, non-standard Haar basis set having a resolution n of 3.

FIG. 3 is a diagram illustrating the two-dimensional, non-standard Haar basis set having a resolution n of 2.

FIG. 4 is a diagram illustrating restricted basis functions of the two-dimensional, nonstandard Haar basis set as shown in FIG. 3.

FIG. 5 illustrates four example functions represented in the wavelet domain.

FIG. 6 is a diagram illustrating a wavelet domain representation of the integral of the product of the example functions of FIG. 5.

FIG. 7 is a diagram illustrating an example basis function tree for the basis set as shown in FIG. 2.

FIG. 8 is a diagram illustrating an example basis coefficient tree.

FIG. 9 is a diagram illustrating example products of two basis functions, and an example equation for calculating such products.

FIG. 10 is a diagram illustrating example products of more than two basis functions, and example equations for calculating such products.

FIG. 11 is a table that tabulates the basis function type of the basis function appearing in the product of a plurality of basis functions.

FIG. 12 is a block diagram illustrating an embodiment of a method of determining an N\textsuperscript{th} order integral coefficient C\textsuperscript{N}.

FIG. 13 is a block diagram illustrating an embodiment of a method for determining a basis function type of a basis function appearing in the product of a plurality of basis functions.

FIG. 14 is a block diagram illustrating a general-purpose computer system that can be used to implement embodiments of the systems and methods disclosed herein.

FIG. 15 is a table illustrating seven cases that facilitate determining the N\textsuperscript{th} order integral coefficient C\textsuperscript{N}.

FIG. 16 is a block diagram illustrating another embodiment of a method for determining the integral of the product of a plurality of functions.

FIG. 17 illustrates an example wavelet tree in pseudo code.

FIG. 18 is a block diagram illustrating an embodiment of a method for traversing direct paths through a plurality of wavelet trees to determine the integral of the product of the functions represented by the wavelet trees.
The wavelet transform can be a nonstandard Haar wavelet transform, in which case the wavelet basis set $\beta$ can be a two-dimensional, nonstandard Haar basis set. Such a wavelet transform and basis set are described by Stollnitz et al. in “Wavelets for Computer Graphics: A Primer,” IEEE Computer Graphics and Applications (1995), 15, 3, 76-84, which is incorporated by reference in its entirety herein. Projecting each function $F_f(v)$ onto the two-dimensional, nonstandard Haar basis set by performing the nonstandard Haar wavelet transform simplifies evaluating the integral of the product of the plurality of functions because only nonzero basis coefficients $f_{i,k}$ and nonzero $N^{th}$ order integral coefficients $C_N$ contribute to the integral and when two-dimensional, nonstandard Haar basis functions are used to represent the functions $F_f(v)$, many of the basis coefficients $f_{i,k}$ and the $N^{th}$ order integral coefficients $C_N$ are zero, as described in detail below.

Therefore, from this point forward, the term wavelet transform generally refers to the nonstandard Haar wavelet transform, the term basis function $b_\nu(v)$ generally refers to a two-dimensional, nonstandard Haar basis function and the term basis set $\mathcal{B}$ generally refers to the two-dimensional, nonstandard Haar basis set and the basis functions, of this basis set, unless otherwise indicated. Because the wavelet transform is known in the art and is explained in the Stollnitz reference,
incorporated above, a discussion of the wavelet transform is omitted here. However, a brief explanation of the basis set is provided below.

The two-dimensional, nonstandard Haar basis set B includes a plurality of basis functions, the number of basis functions varying with the resolution n of the basis set. For example, FIG. 2 is a diagram illustrating a basis set 200, having a resolution n of 3 and a total of eighty-four basis functions. In FIG. 2, each square diagram represents one of the basis functions. The basis function is positive where the diagram is white, is negative where the diagram is black, and is zero where the diagram is gray. For simplicity, a magnitude (6) of the basis function is ignored in the diagrams, but if shown, the magnitude would project out of or into the page. Each of the basis functions is normalized such that the magnitude of the basis function is one, and therefore the basis set 200 is an orthonormal basis set.

Generally speaking, the basis set 200 includes four basis function types: scaling basis functions generally denoted by \( \psi \), and three different wavelet basis functions generally denoted by \( \psi_1 \), \( \psi_2 \), and \( \psi_3 \). A mother basis function is defined for each of the four basis function types. As illustrated in FIG. 2, the mother basis functions include a mother scaling basis function \( \phi \) and three different mother wavelet basis functions \( \psi_1 \), \( \psi_2 \), and \( \psi_3 \). Each of the mother basis functions takes a unique shape within the diagram. The equations of the mother basis functions are provided below:

\[
\phi(x, y) = \begin{cases} 
1 & \text{if } (0 \leq x \leq 1) \text{ and } (0 \leq y \leq 1) \\
0 & \text{else} 
\end{cases}
\]

\[
\phi_1(x, y) = \begin{cases} 
1 & \text{if } (0 \leq x \leq \frac{1}{2}) \text{ and } (0 \leq y \leq 1) \\
-1 & \text{if } (\frac{1}{2} < x \leq 1) \text{ and } (0 \leq y \leq 1) \\
0 & \text{else} 
\end{cases}
\]

\[
\phi_2(x, y) = \begin{cases} 
1 & \text{if } (0 \leq x \leq 1) \text{ and } (0 \leq y \leq \frac{1}{2}) \\
-1 & \text{if } (0 \leq x \leq 1) \text{ and } (\frac{1}{2} < y \leq 1) \\
0 & \text{else} 
\end{cases}
\]

\[
\phi_3(x, y) = \begin{cases} 
1 & \text{if } (0 \leq x \leq \frac{1}{2}) \text{ and } (0 \leq y \leq \frac{1}{2}) \\
-1 & \text{if } (\frac{1}{2} < x \leq 1) \text{ and } (\frac{1}{2} < y \leq 1) \\
1 & \text{if } (0 \leq x \leq \frac{1}{2}) \text{ and } (\frac{1}{2} < y \leq 1) \\
0 & \text{else} 
\end{cases}
\]

Each of the remaining basis functions in the basis set 200 is a dilated and spatially translated version of one of the mother basis functions. The mother basis functions can be dilated by a scale j and can be spatially translated with respect to an x-y coordinate system by spatial translations k and l. The scale j can be any integer from 0 to (n−1), n being the resolution of the basis set 200. Each spatial translation k and l can be any integer from 0 to (2^n−1). Each combination \( <j,k,l> \) is a unique support that indicates a size and location of the basis function in the diagram, the size being a function of the scale j and the location being a function of the spatial translations k, l.

For each support \( <j,k,l> \), four normalized basis functions are defined, one for each of the basis function types. A scaling basis function \( \psi_d \) for the support \( <j,k,l> \) is expressed in terms of the mother scaling basis function \( \phi \) by:

\[
\psi_d(x, y) = 2^{j/2} \phi(2^{-j}x - k, 2^{-j}y - l)
\]

Three wavelet basis functions \( \psi_{d+1}, \psi_{d+2}, \psi_{d+3} \) for the support \( <j,k,l> \) are expressed in terms of their respective mother wavelet functions \( \psi_1, \psi_2, \psi_3 \) by:

\[
\psi_{d+1}(x, y) = 2^{j/2} \psi_1(2^{-j}x - k, 2^{-j}y - l)
\]

\[
\psi_{d+2}(x, y) = 2^{j/2} \psi_2(2^{-j}x - k, 2^{-j}y - l)
\]

\[
\psi_{d+3}(x, y) = 2^{j/2} \psi_3(2^{-j}x - k, 2^{-j}y - l)
\]

The basis set 200 therefore includes one basis function for each basis function type at each support \( <j,k,l> \), the basis function type indicating the shape of the basis function within the diagram, and the support indicating the size and location of the basis function within the diagram. Although the two-dimensional, nonstandard Haar basis set B is described above with reference to the basis set 200 shown in FIG. 2, a person of skill would understand that the basis set B can have greater or fewer basis functions depending on the resolution n. For example, FIG. 3 is a diagram illustrating the two-dimensional, nonstandard Haar basis set 300 having a resolution n of 2. The principles described with reference to the basis set 200 generally apply to the basis set 300, and to any other such basis set B regardless of the resolution n.

The basis set B includes a subset of restricted basis functions. The restricted basis function subset includes \( 2^m \times 2^n \) basis functions from the basis set, including the mother scaling basis function \( \phi \) and all of the wavelet basis functions \( \psi_{d+1}, \psi_{d+2}, \psi_{d+3} \). In other words, the restricted basis function subset includes all basis functions in the basis set except for the scaling basis functions \( \psi_d \) having scales j greater than zero. For example, the restricted basis function subset 400 of the two-dimensional, nonstandard Haar basis set 300 is shown in FIG. 4.

Returning to block 102 of FIG. 1, the nonstandard Haar wavelet transform projects the functions \( F_d(v) \) onto restricted basis functions of the basis set. Because the wavelet transform projects the function \( F_d(v) \) onto the restricted bases, the wavelet domain representation of the function \( F_d(v) \) in equation (2) is the sum of a series of \( 2^m \times 2^n \) restricted basis functions \( b_d \) scaled by \( 2^m \times 2^n \) corresponding basis coefficients \( f_{d,b} \) where n is the resolution. Therefore, from this point forward, the term basis function \( b_d \) generally refers to one of the restricted basis functions and the term basis coefficient \( f_{d,b} \) generally refers to either the mother scaling coefficient or one of the wavelet coefficients.

FIG. 4 illustrates four example functions represented in the wavelet domain. For exemplary purposes, the functions \( F_d(v) \) are shown in terms of the restricted basis functions \( b_d \) of the subset 400 shown in FIG. 4, although in most cases a basis set B of higher resolution is used. In example 502 the function \( F_d(v) \) is represented as sixteen basis coefficients \( f_{1,1} \) to \( f_{1,16} \) scaling the sixteen restricted basis functions \( b_{1} \) to \( b_{16} \). The same is true for examples 504, 506, and 508, which illustrate functions \( F_d(v) \), \( F_d(v) \), and \( F_d(v) \), respectively.

The meaning of the wavelet domain representation of the function \( F_d(v) \) will now be described. The mother basis functions are defined for the support \( <0,0,0> \), which covers the entire diagram. Therefore, the basis coefficients of the mother basis functions provide information about the entire function \( F_d(v) \). Specifically, the shape of the mother scaling basis function \( \phi \) represents an average over the entire diagram, and the
shapes of the mother wavelet functions $\psi_1^o$, $\psi_2^o$, and $\psi_3^o$ represent horizontal, vertical, and diagonal differences, respectively, from the mother scaling basis function $\phi^o$ over the entire diagram. Therefore, the basis coefficient of the mother scaling basis function $\phi^o$ provides information about an average of the function $F(v)$, while the basis coefficients of the wavelet basis functions $\psi_1^o$, $\psi_2^o$, and $\psi_3^o$ provide information about horizontal, vertical, and diagonal differences from average, respectively, over the entire function.

Supports $<j,k,l>$ that are less than the entire diagram represent distinct portions of the function $F(v)$ represented by the diagram. Therefore, the basis coefficients of basis functions for supports $<j,k,l>$ other than $<0,0,0>$ provide information about the portion of the function $F(v)$ represented by the support, and the basis function type indicates the type of information that is provided. Specifically, the basis coefficients of the wavelet basis functions $\psi_1^j$, $\psi_2^j$, $\psi_3^j$, provide information about horizontal, vertical, and diagonal differences, respectively, from the average over the portion of the function $F(v)$ represented by the support. The scaling basis functions $\phi_j$ do not have basis coefficients because the wavelet transform represents the function $F(v)$ in terms of the restricted bases, as discussed above.

While the basis coefficients of basis functions at finer or higher scales $j$ provide information about smaller portions of the function $F(v)$, the information is provided more detailed or resolved. For example, in FIG. 2, the wavelet basis function $\psi_2^1$ has a support that is smaller than the support of the mother wavelet basis function $\psi_2^0$, indicating that a basis coefficient of the wavelet basis function $\psi_2^1$ provides information about a smaller portion of the function $F(v)$ than the basis coefficient of the mother wavelet function $\psi_2^0$; however, the diagonal difference from average represented by the basis coefficient of the wavelet basis function $\psi_3^1$ is more resolved than the diagonal difference from the average represented by the basis coefficient of the mother wavelet function $\psi_3^0$, as visually indicated by the increased resolution in the diagram.

With reference back to FIG. 1, in block 104 the integral $u$ of the product of the plurality of functions is represented in the wavelet domain as the sum of a series of contributing products. Each contributing product in the series is the product of multiple basis coefficients and one $N^{th}$ order integral coefficient $C^N$, the multiple basis coefficients including one basis coefficient $\beta_h$ from each of the $N$ functions, and the $N^{th}$ order integral coefficient $C^N$ being the integral of the product of the basis functions $\beta_h$ that correspond to those basis coefficients $\beta_h$. One contributing product appears in the series for each combination of $N$ basis coefficients $\beta_h$ having one coefficient from each of the $N$ functions, such that a total of $M^N$ contributing products appear in the series. It should be noted that in the basis coefficient $\beta_h$, $h$ is an integer from 1 to $M$ and $i$ is an integer from 1 to $N$; each of $h$ and $i$ being the number of basis functions $\beta_h$ used to represent the function $F(v)$ and $N$ being the number of functions $F(v)$ whose product is being integrated.

FIG. 6 is a diagram illustrating the wavelet domain representation of the integral $u$ of the product of the example functions of FIG. 5. In example 600, the example functions $F_1$, $F_2$, $F_3$, and $F_4$ of examples 502, 504, 506, and 508 are inserted into equation (10). The integral $u$ of the product of the functions is represented in the wavelet domains the sum of a series of $16^N$ contributing products, but for illustrative purposes, only three of these contributing products 602, 604, and 606 are shown in example 600. Each product 602, 604, and 606 includes four basis coefficients 608, one from each of the functions $F_1$, $F_2$, $F_3$, and $F_4$, and one $4^{th}$ order integral coefficient 610, which is the integral of the basis functions that correspond to the basis coefficients 608. For example, the four basis coefficients 608 in the product 602 are the basis coefficients $f_{1,1}$ from the function $F_1$, the basis coefficient $f_{2,1}$ from the function $F_2$, the basis coefficient $f_{3,1}$ from the function $F_3$, and the basis coefficient $f_{4,1}$ from the function $F_4$. Each of these basis coefficients corresponds to the basis function labeled $\beta_1$ in FIG. 4, which is the mother scaling basis function $\phi^o$, as shown in FIG. 4. Therefore, the $4^{th}$ order integral coefficient 610 in the product 602 is the integral of the product of four mother scaling basis functions $\phi^o$, as shown in example 612 of FIG. 6. The example product 604 is the product for the combination of basis coefficients that includes the basis coefficient $f_{1,4}$ from the function $F_1$, the basis coefficient $f_{2,2}$ from the function $F_2$, the basis coefficient $f_{3,1}$ from the function $F_3$, and the basis coefficient $f_{4,1}$ from the function $F_4$. Therefore, as shown in example 614 of FIG. 6, the $4^{th}$ order integral coefficient 610 in the product 604 is the integral of the product of the basis functions labeled $\beta_4$, $\beta_2$, $\beta_1$, and $\beta_7$ in FIG. 4. The example product 610 follows the same form.
In block 106, the basis functions are organized in a basis function tree, and the basis coefficients of each function are placed in a basis coefficient tree. The basis function tree is described first, because its organization informs the organization of the basis coefficient tree.

The basis function tree organizes basis functions based on parent-child relationships among the basis functions. The mother scaling function \( \phi \) is defined as the parent of all other basis functions, and is defined as the immediate parent of the mother wavelet functions \( \psi_{10}^0, \psi_{12}^0, \psi_{14}^0, \) and \( \psi_{16}^0 \). For all other basis functions, the basis function is a parent basis function of a child basis function if the scale \( j \) of the parent basis function is less than the scale \( j \) of the child basis function and the support <\( j,k,l \)> of the parent basis function completely covers the support <\( j,k,l \)> of the child basis function, meaning the parent basis function is positionally located in the diagram with respect to the \( x,y \) coordinate system in every \( x,y \) position occupied by the child basis function. The parent basis function of support <\( j,k,l \)> is an immediate parent of the child basis function, indicating its status as the parent of all basis functions in the child node in which a child basis function lies. The immediate parent basis functions are the four other basis functions depending from it, and if the basis function in the child node immediately depends on the support <\( j,k,l \)> of the child basis function, meaning the immediate child includes all of the basis functions in the child node in which a child basis function lies. For example, FIG. 7 is the diagram illustrating an example basis function tree 700 for the basis set 200 shown in FIG. 2. The mother scaling basis function \( \phi \) lies in a root 702 of the tree 700, indicating its status as the parent of all basis functions in the set 200. A mother node 704 immediately depends from the root 702 and includes the three mother wavelet functions \( \psi_{10}^0, \psi_{12}^0, \psi_{14}^0, \) and \( \psi_{16}^0 \), indicating their status as the immediate children of the mother scaling basis function \( \phi \).

Below the mother node 704 in which the three mother wavelet functions \( \psi_{10}^0, \psi_{12}^0, \psi_{14}^0, \) and \( \psi_{16}^0 \) lie, basis functions are organized in child nodes 706, each child node 706 corresponding to a unique support <\( j,k,l \)> and including the four basis functions that are defined for the support. The four basis functions in the child node 706 are generally represented by \( b_{j,k,l}^i \), where \( b \) generally denotes a basis function, \( j \) denotes the scale of the basis function, and \( (k,l) \) denotes the spatial translation. For example, the basis functions in child node 710 are represented by the symbol \( b_{1,1,1}^1 \), indicating the node includes all four basis functions defined on the support <1,1,1>, as shown in the key 712.

Because the child nodes 706 are organized according to parent-child relationship, any basis function in any child node 706 is an immediate child basis function of the basis functions in the node from which the child node 706 depends. For example, each basis function in the child node 710 is the immediate child basis function of each of the mother wavelet functions in the mother node 704. Further, if any basis function in a child node 706 is a parent basis function, its corresponding child basis functions are located in child nodes 706 depending from it, and if the basis function in the child node 706 is an immediate parent basis function, its corresponding immediate child basis functions are located in the child nodes 706 immediately depending from it. For example, in FIG. 7 the child node 710 has four child nodes 706 immediately depending from it, indicating that each of the four basis functions defined on the support <1,1,1> is an immediate parent basis function of each of the basis functions defined on supports <2,2,2>, <2,3,2>, <2,2,3>, and <2,3,3>.

As shown, the basis function tree 700 includes all of the basis functions of the basis set 200 shown in FIG. 2, which has a resolution \( n \) of 3. Of course, the basis function tree 700 may have greater or fewer child nodes 706 depending on the resolution \( n \).

A direct path through the basis function tree 700 connects the node in which a child basis function lies to the node in which its parent basis function lies. The direct path traces from the child basis function to the parent basis function passing through any nodes including the immediate parent basis functions. For example, in FIG. 7, a direct path 714 exists from the child basis function \( \psi_{11,10}^1 \) in the child node 716 to the mother scaling function \( \phi \) at the root 702. The child basis function \( \psi_{11,10}^1 \) is in the child node 716 of all of the basis functions in the child node 718, including basis functions \( \psi_{10,10}^0, \psi_{12,10}^0, \psi_{14,10}^0, \) and \( \psi_{16,10}^0 \), which in turn have the mother wavelet functions \( \psi_{10}^0, \psi_{12}^0, \psi_{14}^0, \) and \( \psi_{16}^0 \) in the mother node 704 as immediate parents, which in turn having the mother scaling function \( \phi \) in the root 702 as an immediate parent. However, the direct path need not extend all the way to the mother scaling function \( \phi \) in the root 702. For example, the scaling basis function \( \phi_{10}^1 \) is a child basis function of the mother wavelet function \( \psi_{10}^0 \), as indicated by the direct path 720 connecting the child node 710 to the mother node 704.

The basis coefficient tree has the same nodes as the corresponding basis function tree. However, instead of organizing basis functions according to parent-child relationship, the basis coefficient tree organizes basis coefficients. Each basis coefficient is in the node of the basis coefficient tree that corresponds to the node occupied by its corresponding basis function in the basis function tree. Therefore, the basis coefficient tree has the same nodes positioned with respect to each other in the same manner as the basis function tree. Unlike the basis function tree, however, in the basis coefficient tree the child nodes include at most three basis coefficients, the mother scaling function not having basis coefficients because the wavelet transform projects the function onto the restricted bases.

For example, FIG. 8 is a diagram illustrating an example basis coefficient tree 800, which organizes the basis coefficients for the function \( f_1(y) \) shown in example 502 of FIG. 5. Because the basis coefficient \( f_{1,1,1} \) corresponds to the mother scaling basis function \( \phi \), the basis coefficient \( f_{1,1,1} \) lies in a root 802 of the basis coefficient tree 800. The basis coefficients \( f_{1,2,1}, f_{1,3,1}, \) and \( f_{1,4,1} \) correspond to the three mother wavelet functions \( \psi_{1,1,1}, \psi_{2,1,1}, \) and \( \psi_{3,1,1}, \) and therefore these basis coefficients lie in a mother node 804 that immediately depends from the root 802. Below the mother node 804, basis coefficients are organized in child nodes 806, each child node 806 corresponding to a unique support <\( j,k,l \)> and including the three basis coefficients that correspond to the wavelet basis functions \( \psi_{1,1,1}, \psi_{2,1,1}, \psi_{3,1,1}, \) defined for the support. For example, the child node 810 in the basis coefficient tree 800 includes the three basis coefficients \( f_{1,4,1}, f_{1,5,1}, \) and \( f_{1,6,1} \) that correspond to the three wavelet basis functions \( \psi_{1,1,1}, \psi_{2,1,1}, \psi_{3,1,1} \), defined for the support <1,1,1> in the child node 710 of FIG. 7.

As shown, the basis coefficient tree 800 includes one basis coefficient for each restricted basis function of the basis set shown in FIG. 4. Of course, the basis coefficient tree 800 may have greater or fewer child nodes 806 depending on the resolution \( n \) of the basis set, and usually a basis set of higher resolution will be used.
With reference back to block 106 of FIG. 1, the basis coefficients of the function $F_i(v)$ are placed in the basis coefficient tree. In some embodiments, one basis coefficient tree is defined for each function $F_i(v)$, such that a total of $N$ basis coefficients trees are defined for the $N$ functions whose product is being integrated. In other embodiments, a greater number of basis coefficient tree can be used.

In block 108, the $N^0$ order integral coefficients $C_N$ are determined, one $N^0$ order integral coefficient $C_N$ appearing in each contributing product in the sum of the series of contributing products that determines the integral $u$. As mentioned above, only contributing products that include nonzero $N^0$ order integral coefficients $C_N$ contribute to the result $u$ in equation (10), but when two-dimensional, nonstandard Haar basis functions are used to represent the functions $F_i(v)$, many of the $N^0$ order integral coefficients $C_N$ are zero. If the integral coefficient is zero, the entire contributing product is zero and does not contribute to the integral. If the integral coefficient is not zero, its value should be determined so that the contribution of the contributing product to the integral $u$ is considered. For example, in FIG. 6 each of the example products 602, 604, and 606 includes four basis coefficients and one $N^0$ order integral coefficients $C_N$. If the $N^0$ order integral coefficient is zero, the product 602, 604, or 606 does not contribute to the integral $u$. Therefore, a set of principles for determining the $N^0$ order integral coefficient $C_N$ are described below with reference to FIGS. 9-13.

FIG. 9 is a diagram illustrating example products of two basis functions, and an example equation for calculating such products. As shown in equation (11), a product $P^2$ of two basis functions $b_i$ and $b_j$ is another basis function $b_k$, scaled by a sign $+/-$, which may be positive or negative, and a magnitude $|P^2|$, which may be zero:

$$P^2 = b_i \cdot b_j = +/-(|P^2|) \cdot b_k$$  \hspace{1cm} (11)

The product $P^2$ of the two basis functions is not zero if the support of one of the basis functions completely covers the support of the other basis function, as shown in examples 901 through 915. As mentioned above, the support of one basis function completely covers the support of another basis function if the one basis function is positionally located in the diagram with respect to the x-y coordinate system in every (x, y) position occupied by the other basis function. In other words, the product $P^2$ of two basis functions is zero only if either there is a direct path in the basis function tree between the nodes occupied by the two basis functions, meaning the two basis functions are related as parent and child, or the two basis functions lie in the same node of the basis function tree, meaning the two basis functions have the same support. Otherwise, the support of neither basis function completely covers the support of the other basis function, the product $P^2$ of two basis functions is zero, as shown in examples 916, 917, and 918. In other words, the product $P^2$ of two basis functions is zero if the supports of the two basis functions are completely disjoint.

In cases in which the product $P^2$ is not zero, the sign, magnitude, and basis function of the product $P^2$ can be determined from the characteristics of the two basis function being multiplied together. For example, the product of two identical wavelet basis functions of the same support is the scaling basis function of the same support, scaled by a magnitude $2^{j}$, as shown in examples 904, 905, and 906 and in equations (13):

$$\psi_j \cdot \psi_j = 2^j \psi_j$$  \hspace{1cm} (13)

The product of a scaling basis function of a given support and a wavelet basis function of the same support is the wavelet basis function of the same type and support, scaled by a magnitude $2^{j}$, as shown in examples 907, 908, and 909 and in equation (14):

$$\psi_j \cdot \phi_j = 2^j \psi_j$$  \hspace{1cm} (14)

The product of two scaling basis functions of the same support is the scaling basis function of the same support, scaled by a magnitude $2^{j}$, as shown in examples 910, 911, and 912 and in equation (15):

$$\phi_j \cdot \phi_j = 2^j \phi_j$$  \hspace{1cm} (15)

The product of a child basis function and a parent basis function is the child basis function, scaled by a sign $+/-$ and a magnitude $2^{j}$, where the sign is the same as the sign of the portion of the support of the parent basis function that is covered by the child basis function covers, and where the magnitude $2^{j}$ is a function of the scale $j$ of the parent basis function, as shown in examples 913, 914, and 915 and in equation (16):

$$b_j^{+/-} = 2^j b_j$$  \hspace{1cm} (16)

For example, in example 913, the sign of the product is negative, because the support of the child basis function covers a portion of the support of the parent basis function that is negative, where the diagram is black. The magnitude of the product is one, because the scale $j$ of the parent basis function is zero. However, in example 915, the sign of the product is positive, because the support of the child basis function covers a portion of the support of the parent basis function that is positive, where the diagram is white.

The examples shown in FIG. 9 illustrate basis functions of the basis set 200 as shown in FIG. 2, but the principles outlined generally apply to all basis functions, including basis functions that are not shown in FIG. 2. Additionally, the principles outlined generally can be applied to determine the product of more than two basis functions, as described below.

FIG. 10 is a diagram illustrating example products of more than two basis functions, and example equations for calculating such products. Just as with the product $P^2$ of two basis functions, a product $P^3$ of g basis functions $b_1, b_2, \ldots, b_g$, is another basis function $b_h$, scaled by a sign, which may be positive or negative, and a magnitude $|P^3|$, which may be zero, as shown in equation (17):

$$P^3 = b_1 \cdot b_2 \cdot \ldots \cdot b_g = +/- |P^3| \cdot b_h$$  \hspace{1cm} (17)

The product $P^3$ of $g$ basis functions can be calculated by multiplying two of the basis functions together to produce a product $P^2$, as described above with reference to FIG. 9. The product $P^2$ can then be multiplied by the next basis function in the set of $g$ basis functions. After $(g-2)$ products have been calculated, the set of $g$ basis functions is reduced to a set of two basis functions: one being the product $P^{g-1}$ and the other being the basis function $b_g$, as shown in equation (18), below:

$$P^3 = (P^{g-1} \cdot b_g)$$  \hspace{1cm} (18)

The product $P^3$ is zero if the product $P^2$ of any two basis functions in the set of $g$ basis functions is zero. As described above, the product $P^2$ of any two basis functions is non-zero if
the two basis functions are related as parent and child or if the two basis functions have the same support. Therefore, the product $P^g$ is non-zero if and only if each basis function in the set of $g$ functions lies in a node that is on a direct path through the basis function tree to the node occupied by every other basis function in the set of $g$ basis functions. In other words, the product $P^g$ is non-zero if and only if all of the basis functions lie in nodes that are on a single direct path through the basis function tree. Otherwise, the product $P^g$ of $g$ basis functions is zero.

For example, an arbitrary set of basis functions 1020 is shown in example 1004 of FIG. 10. The basis functions lie on a single direct path 614 through the basis function tree 600 in FIG. 6. Therefore, the product 1022 of the set of basis functions 1020 is not zero, as shown in FIG. 10.

Determining the product $P^g$ is facilitated by placing the basis functions in a ranking order. Basis functions having finer scales are positioned in earlier positions in the ranking order than basis functions having coarser scales, and among basis functions having the same scale, wavelet basis functions of a given scale are positioned in earlier positions in the ranking order than the scaling basis function of that scale. As a result, a basis function having a finest scale $j_o$, of the set of $g$ functions appears in an earliest position in the ranking order. It should be noted that the term "finer scale" means a scale that is numerically greater than another "coarser scale", the terms finer and coarser denoting the resolution with which such basis functions are able to represent entities in the wavelet domain.

For example, the set of basis functions 1020 is organized in an arbitrary order in example 1004, and the same set of basis functions 1020 is organized in the ranking order in example 1006. An arrow 1024 moves in the direction of earlier positions in the ranking order, and terminates at an earliest position 1026 occupied by one of the basis functions having the finest scale $j_o$, which in the set 1020 happens to be a scale $j$ of 2. In the example 1006, the earliest position 1026 in the ranking order is the position that is farthest to the left on the page, although any position on the page can be defined as the earliest position as long as the basis functions are positioned relative to each other according to the rule defined above.

Because the multiplication of basis functions is commutative, organizing the set of $g$ basis functions in the ranking order does not change the product $P^g$, as can be seen by comparing the product 1022 in example 1004 with the product 1028 in example 1006. However, organizing the basis functions in the ranking order does facilitate determining the sign, the magnitude $|P^g|$, and the basis function $b_{pg}$ of the product $P^g$.

In cases in which the product $P^g$ is not zero, the magnitude of the product $P^g$ can be determined using equation (19):

$$P^g = ± \sum_{j=0}^{j_o} (-1)^j$$

where

$$\sum_{j=0}^{j_o} (-1)^j$$

is the determined by taking the sum of the scale of each basis function in the set of $g$ functions and then subtracting the finest scale $j_o$ appearing in the set of $g$ functions. As noted above, the finest scale $j_o$ is the scale of at least the basis function appearing in the earliest position 1026 in the ranking order. The sign can be determined by multiplying together a series of signs, the series of signs including one sign for each parent basis function in the product $P^g$, the sign for the parent basis function being the sign of the portion of the support of the parent basis function that is covered by its child basis functions. It should be noted that a negative magnitude effects a reversal of the colors in the diagram because, as mentioned above, the diagram is white where the magnitude is positive, black where the magnitude is negative, and gray where the magnitude is zero.

For example, in example 1006, the set of basis functions 1020 includes five basis functions having a scale $j$ of 2, two basis functions having a scale $j$ of 1, and two basis functions having a scale $j$ of 0. The finest scale $j_o$ appearing in the set of basis functions 1020 is $j$ of 2, which is the scale of the basis function in the earliest position 1026. Therefore, the magnitude of the product 1028 is $|P^g| = (-) (-) (-) (-) (+) = (+)$. A negative portion of the support of the parent basis function $\psi_{j_o}$ is covered by the supports of its child basis functions. A positive portion of the support of the parent basis function $\psi_{j_o}$ is covered by the supports of its child basis functions. A positive portion of the support of the parent basis function $\psi^{φ_0}$ is covered by the supports of its child basis functions, and a positive portion of the support of the parent basis function $φ^{j_o}$ is covered by the supports of its child basis functions. Therefore, the sign of the product 1028 is negative, or $(-)(+)(+)(+)(+) = (-)$.

The basis function $b_{pg}$ of the product $P^g$ can be determined by isolating the subset of $m$ basis functions having the finest scale $j_o$, and taking the product $P^m$ of the basis functions in the subset. In cases in which the product $P^g$ is not zero, the basis function $b_{pg}$ appearing in the product $P^g$ is the same as the basis function $b_{pm}$ appearing in the product $P^m$ of the subset of $m$ basis functions having the finest scale $j_o$. This result occurs because the product of a child basis function and a parent basis function is the child basis function, scaled by a sign and a magnitude, as explained above with reference to examples 913 through 916 of FIG. 9.

For example, in example 1006, the finest scale $j_o$ in the set of basis functions 1020 is a scale of 2, and the subset 1030 includes all of the basis functions having this scale. In example 1008, the product of the basis functions in the subset 1030 is taken. The basis function 1032 of the product 1034 that results is the scaling basis function $\phi_{j_o}$ and therefore the basis function 1036 of the product 1028 is also the scaling basis function $\phi_{j_o}$, as shown in example 1006.

Alternatively, the basis function $b_{pg}$ appearing in the product $P^g$ of $g$ basis functions can be determined from the supports $<j,k,l>$ and basis function types of the basis functions in the subset of $m$ basis functions having the finest scale $j_o$. In cases in which the product $P^g$ is not zero, the support of the basis function $b_{pg}$ appearing in the product $P^g$ is the same as the support of the $m$ basis functions having the finest scale $j_o$. Note that each basis function in the subset of $m$ basis functions having the finest scale $j_o$ has the same support $<j_o,k,l>$; otherwise, at least two basis functions would not lie on a direct path through the basis function tree and their product $P^g$ would necessarily be zero, causing the product $P^g$ of the set of $g$ basis functions to also be zero.

The basis function type of the basis function $b_{pg}$ appearing in the product $P^g$ can be determined from the parities of three numbers, each number corresponding to one wavelet function type and being the aggregate number of times the wavelet function type appears in the subset of $m$ basis functions having the finest scale $j_o$. FIG. 11 is a table that tabulates the basis function type of the basis function $b_{pg}$ appearing in the prod-
method

three, and the parity of this aggregate number is odd. The basis functions appearing in the product $P^0$ is the scaling basis function $\phi$, as shown in block 1006.

Regarding the integral of basis functions, the integral of any one of the wavelet basis functions $\psi_1, \psi_2, or \psi_3$ is zero, as shown in equation (20):

$$\int \psi_j dx dy = 0$$

However, the integral of a scaling basis function $\phi$ is $2^d$, as shown in equation (21):

$$\int \phi dx dy = 2^d$$

Returning to block 108 of FIG. 1, the $N^0$ order integral coefficient $C^N$ is determined for each contributing product appearing in the wavelet domain representation of the integral. FIG. 12 is a block diagram illustrating an embodiment of a method 1200 for determining the $N^0$ order integral coefficient $C^N$, the method employing the principles described above with reference to FIGS. 7-11. In block 1202, it is determined whether the $N$ basis functions $b_j$ that appear in the product $P^N$ lie in nodes on a single direct path through the basis function tree. If the $N$ basis functions $b_j$ do not lie in nodes on a single direct path, the product $P^N$ of the $N$ basis functions is zero, as shown in block 1204. In such case, the $N^0$ order integral coefficient $C^N$ is also zero, as shown in block 1206. If the $N$ basis functions $b_j$ do lie in nodes on a single direct path through the basis function tree, the product $P^N$ is nonzero, as shown in block 1208.

In cases in which the product $P^N$ is nonzero, the $N^0$ order integral coefficient $C^N$ is also nonzero if the basis function type of the basis function $b_j$ appearing in the product $P^N$ is a scaling basis function $\phi$, according to equation (21). Otherwise, the basis function type is one of the wavelet basis functions $\psi_1, \psi_2, or \psi_3$ and the $N^0$ order integral coefficient $C^N$ is zero, according to equation (20). Therefore, in block 1210, the basis function type of the basis function $b_j$ appearing in the product $P^N$ is determined. In block 1212, it is asked whether the basis function type of the basis function $b_j$ appearing in the product $P^N$ is a scaling basis function $\phi$.

FIG. 13 is a block diagram illustrating an embodiment of a method 1300 for determining the basis function type of the basis function $b_j$ appearing in the product $P^N$. In block 1302, the $N$ basis functions $b_j$ that appear in the product $P^N$ are placed in the ranking order, such as the ranking order described with reference to FIG. 10. In block 1304, a subset of $m$ basis functions $b_j$ having a finest scale $j_0$ are isolated from the $N$ basis functions that appear in the product $P^N$. In block 1306, the aggregate number of times each of the three wavelet function types appears in the subset of $m$ basis functions are counted. In block 1308, the parity of each of the three aggregate numbers is compared. In block 1310, it is asked whether the parities of the three aggregate numbers are the same. If the parities of any of the three aggregate numbers are different from the other numbers, then the basis function $b_j$ of the product $P^N$ is one of the wavelet basis functions, as shown in block 1312. Specifically, the basis function $b_j$ of the product is the wavelet basis function that corresponds to the aggregate number having a parity that differs from the other two aggregate numbers. If the parities of each of the three numbers are the same, the basis function $b_j$ of the product $P^N$ is the scaling basis function $\phi$, as shown in block 1314.

Returning to FIG. 12, if the basis function type of the basis function $b_j$ of the product $P^N$ is not the scaling basis function type, the $N^0$ order integral coefficient $C^N$ is zero, as shown in block 1214. If the type is the scaling basis function type, the $N^0$ order integral coefficient $C^N$ is nonzero, as shown in block 1216. In such case, the nonzero value of the $N^0$ order integral coefficient $C^N$ is found using equation (22), which results from combining equations (19) and (21):

$$C^N = \sum_{-j_0}^{j_0} \phi_{x_0}$$

In equation (22), the sign for each parent basis function in the product tree, the product $P^N$ is nonzero, as shown in block 1208.

In cases in which the product $P^N$ is nonzero, the $N^0$ order integral coefficient $C^N$ is also nonzero if the basis function type of the basis function $b_j$ appearing in the product $P^N$ is a scaling basis function $\phi$, according to equation (21). Otherwise, the basis function type is one of the wavelet basis functions $\psi_1, \psi_2, or \psi_3$ and the $N^0$ order integral coefficient $C^N$ is zero, according to equation (20). Therefore, in block 1210, the basis function type of the basis function $b_j$ appearing in the product $P^N$ is determined. In block 1212, it is asked whether the basis function type of the basis function $b_j$ appearing in the product $P^N$ is a scaling basis function $\phi$. FIG. 13 is a block diagram illustrating an embodiment of a method 1300 for determining the basis function type of the basis function $b_j$ appearing in the product $P^N$. In block 1302, the $N$ basis functions $b_j$ that appear in the product $P^N$ are placed in the ranking order, such as the ranking order described with reference to FIG. 10. In block 1304, a subset of $m$ basis functions $b_j$ having a finest scale $j_0$ are isolated from the $N$ basis functions that appear in the product $P^N$. In block 1306, the aggregate number of times each of the three wavelet function types appears in the subset of $m$ basis functions are counted. In block 1308, the parity of each of the three aggregate numbers is compared. In block 1310, it is asked whether the parities of the three aggregate numbers are the same. If the parities of any of the three aggregate numbers are different from the other numbers, then the basis function $b_j$ of the product $P^N$ is one of the wavelet basis functions, as shown in block 1312. Specifically, the basis function $b_j$ of the product is the wavelet basis function that corresponds to the aggregate number having a parity that differs from the other two aggregate numbers. If the parities of each of the three numbers are the same, the basis function $b_j$ of the product $P^N$ is the scaling basis function $\phi$, as shown in block 1314.

Returning to FIG. 12, if the basis function type of the basis function $b_j$ of the product $P^N$ is not the scaling basis function type, the $N^0$ order integral coefficient $C^N$ is zero, as shown in block 1214. If the type is the scaling basis function type, the $N^0$ order integral coefficient $C^N$ is nonzero, as shown in block 1216. In such case, the nonzero value of the $N^0$ order integral coefficient $C^N$ is found using equation (22), which results from combining equations (19) and (21):

$$C^N = \sum_{-j_0}^{j_0} \phi_{x_0}$$

In equation (22), the sign for each parent basis function in the product tree, the product $P^N$ is nonzero, as shown in block 1208.

In cases in which the product $P^N$ is nonzero, the $N^0$ order integral coefficient $C^N$ is also nonzero if the basis function type of the basis function $b_j$ appearing in the product $P^N$ is a scaling basis function $\phi$, according to equation (21). Otherwise, the basis function type is one of the wavelet basis functions $\psi_1, \psi_2, or \psi_3$ and the $N^0$ order integral coefficient $C^N$ is zero, according to equation (20). Therefore, in block 1210, the basis function type of the basis function $b_j$ appearing in the product $P^N$ is determined. In block 1212, it is asked whether the basis function type of the basis function $b_j$ appearing in the product $P^N$ is a scaling basis function $\phi$. FIG. 13 is a block diagram illustrating an embodiment of a method 1300 for determining the basis function type of the basis function $b_j$ appearing in the product $P^N$. In block 1302, the $N$ basis functions $b_j$ that appear in the product $P^N$ are placed in the ranking order, such as the ranking order described with reference to FIG. 10. In block 1304, a subset of $m$ basis functions $b_j$ having a finest scale $j_0$ are isolated from the $N$ basis functions that appear in the product $P^N$. In block 1306, the aggregate number of times each of the three wavelet function types appears in the subset of $m$ basis functions are counted. In block 1308, the parity of each of the three aggregate numbers is compared. In block 1310, it is asked whether the parities of the three aggregate numbers are the same. If the parities of any of the three aggregate numbers are different from the other numbers, then the basis function $b_j$ of the product $P^N$ is one of the wavelet basis functions, as shown in block 1312. Specifically, the basis function $b_j$ of the product is the wavelet basis function that corresponds to the aggregate number having a parity that differs from the other two aggregate numbers. If the parities of each of the three numbers are the same, the basis function $b_j$ of the product $P^N$ is the scaling basis function $\phi$, as shown in block 1314.

Returning to FIG. 12, if the basis function type of the basis function $b_j$ of the product $P^N$ is not the scaling basis function type, the $N^0$ order integral coefficient $C^N$ is zero, as shown in block 1214. If the type is the scaling basis function type, the $N^0$ order integral coefficient $C^N$ is nonzero, as shown in block 1216. In such case, the nonzero value of the $N^0$ order integral coefficient $C^N$ is found using equation (22), which results from combining equations (19) and (21):
basis function tree, as shown in example 614. Therefore, the product of the basis function is zero, and the corresponding 4th order integral coefficient in the product 604 is zero, meaning the product 604 does not contribute to the integral and can be eliminated from the series of contributing products.

In product 606, the basis coefficients correspond to basis functions that lie on a single direct path through the basis function tree. Therefore, the product of the basis function is nonzero. However, the aggregate numbers of each wavelet function type are not all of the same parity, as shown in example 616. Therefore, the 4th order integral coefficient appearing in the product 606 is zero, meaning the product 606 does not contribute to the integral and can be eliminated from the series of contributing products.

In some embodiments, the systems and methods disclosed herein can be implemented in software, firmware, hardware, or combinations thereof. Furthermore, the components of the systems and methods can reside on one computer system, or can be distributed among more than one computer system. In some embodiments, the systems and methods are implemented in software, as an executable program or programs, and are executed by a special or general-purpose digital computer, or combination of computers, such as a personal digital assistant (PDA) or personal computer (PC).

Reference is now made to FIG. 14, which is a block diagram illustrating a general-purpose computer system 1400 that can be used to implement embodiments of the systems and methods disclosed herein. Generally, in terms of hardware architecture, the computer 1401 includes a processor 1402, memory 1403, and one or more input or output (I/O) devices or peripherals 1404 that are communicatively coupled via a local interface 1405. The local interface 1405 can be, for example but not limited to, one or more buses or other wired or wireless connections, as is known in the art. The local interface 1405 may have additional elements (omitted for simplicity), such as controllers, buffers, drivers, repeaters, and receivers, to enable communications. Further, the local interface 1405 may include address, control, and data connections to enable appropriate communications among the aforementioned components.

The processor 1402 is a hardware device for executing software, particularly that stored in memory 1403. The processor 1402 can be any custom made or commercially available processor, a central processing unit (CPU), an auxiliary processor among several processors resulting from the combination of hardware elements (e.g., a personal computer system, system, device, or processor, or a combination thereof). The processor 1402 can be any hardware device for executing instructions, particularly that stored in memory 1403, to communicate with other devices or systems.

The memory 1403 can include any one or combination of volatile memory elements (e.g., random access memory (RAM), such as DRAM, SDRAM, SDRAM, etc.) and nonvolatile memory elements (e.g., ROM, hard drive, tape, CDROM, etc.). Moreover, the memory 1403 may incorporate electronic, magnetic, optical, or other types of storage media. Note that the memory 1403 can have a distributed architecture, where various components are situated remote from one another, but can be accessed by the processor 1402.

The software in memory 1403 may include one or more separate programs, each of which comprises an ordered list of executable instructions for implementing logical functions. In the example of FIG. 14, the software in the memory 1403 includes one or more components of the systems and methods disclosed here, and a suitable operating system 1406. The operating system 1406 essentially controls the execution of other computer programs, such as the systems and methods disclosed herein, file and data management, memory management, and communication control and related services.

The systems and methods disclosed herein may be a source program, executable program (object code), script, or any other function comprising a set of instructions to be performed. When a source program, the program needs to be translated via a compiler, assembler, interpreter, or the like, which may or may not be included within memory 1403, as so to operate properly in connection with the operating system 1406.

The peripherals 1404 may include input devices, for example but not limited to, a keyboard, mouse, scanner, microphone, etc. Furthermore, the peripherals 1404 may also include output devices, for example but not limited to, a printer, display, facsimile device, etc. Finally, the peripherals 1404 may further include devices that communicate both inputs and outputs, for instance but not limited to, a modulator/demodulator (modem; for accessing another device, system, or network), a radio frequency (RF) or other transceiver, a telephone interface, a bridge, a router, etc.

If the computer 1401 is a PC, workstation, or the like, the software in the memory 1403 may further include a basic input output system (BIOS). The BIOS is a set of essential software routines that initialize and test hardware at startup, start the operating system 1406, and support the transfer of data among the hardware devices. The BIOS is stored in the ROM so that the BIOS can be executed when the computer 1401 is activated.

When the computer 1401 is in operation, the processor 1402 is configured to execute software stored within the memory 1403, to communicate data to and from the memory 1403, and to generally control operations of the computer 1401 in accordance with the software. The systems and methods disclosed herein, in whole or in part, are read by the processor 1402, and perhaps buffered within the processor 1402, and then executed.

It should be noted that the systems and methods disclosed herein can be stored on any computer readable medium for use by or in connection with any computer related system or method. In the context of this document, a “computer-readable medium” can be any means that can store, communicate, propagate, or transport the program for use by or in connection with the instruction execution system, system, or device. The computer-readable medium can be, for example but not limited to, an electronic, magnetic, optical, electromagnetic, infrared, or semiconductor system, system, device, or propagation medium. A non-exhaustive example set of the computer-readable medium would include the following: an electrical connection having one or more wires, a portable computer diskette, a random access memory (RAM), a read-only memory (ROM), an erasable programmable read-only memory (EPROM, EEPROM, or Flash memory), and a portable compact disc read-only memory (CDROM).

In an alternative embodiment, in which the systems and methods disclosed herein are implemented in hardware, they can be implemented with any or a combination of the following technologies, which are each well known in the art: a discrete logic circuit(s) having logic gates for implementing logic functions upon data signals, an application specific integrated circuit(s) (ASIC) having appropriate combinatorial logic gates, a programmable gate array(s) (PGA), a field programmable gate array(s) (FPGA), etc.

Any process descriptions or blocks in flowcharts should be understood as representing modules, segments, or portions of code which include one or more executable instructions for implementing specific logical functions or steps in the pro-
The first considerations used to represent each function and $N$ is the number of relatively faster when implemented by computer.

Implementing the method 100 described above using a computer may be computationally expensive, for example, on the order of $O(M^N)$, where $M$ is the number of basis functions used to represent each function and $N$ is the number of functions whose product is being integrated. FIG. 15 is a table illustrating seven cases that facilitate determining the $N^0$ order integral coefficient $C^0$ using a recursive approach. The seven cases are used to construct tree-traversal algorithms for determining the integral $u$ of the product of the plurality of functions. The tree-traversal algorithms may be relatively less computationally complex than the method 100, and therefore relatively faster when implemented by computer.

The seven cases of FIG. 15 apply in cases in which the product of more than two functions $N$ is being integrated. In such cases, the $N^0$ order integral coefficient $C^0$ can be determined by placing the basis functions appearing in the product $P^0$ in the ranking order, determining a product $P^{N-1}$ of the first $(N-1)$ basis functions appearing in the ranking order, and comparing the basis function type and scale of the product $P^{N-1}$ to the basis function type and scale of the basis function $b^0$. In such a case, the $N^0$ order integral coefficient $C^0$ is the integral of the $N^0$ order basis product $P^0$:

$$
C^0 = \int P^0 \cdot b^0
$$

where $P^{N-1}$ is the product of the first $(N-1)$ basis functions appearing in the ranking order and $b^0$ is the $N^0$ basis function appearing in the ranking order.

The $N^0$ order integral coefficient $C^0$ is non-zero if and only if the basis function type of the product $P^0$ is a scaling basis function, which only occurs if one of the following cases hold for the ranked basis product $P^0 = P^{N-1} \cdot b^0$:

1. $b^{p^0}$ and $b^0$ are wavelet basis functions of the same type, and both have identical support. In this case, $C^0 = \|P^{N-1}\|$.  
2. Both $b^{p^0}$ and $b^0$ are both the mother scaling function. In this case, $C^0 = 1$.  
3. $b^{p^0}$ is a scaling basis function, and $b^0$ is a parent basis function of $b^{p^0}$. In this case, $C^0 = \pm 2\|C^{N-1}\|$ where $j$ is the scale of $b^0$, and where the sign is the sign of the portion of the support of $b^0$ that is covered by the support of $b^{p^0}$.  

Otherwise, the basis function type of the product $P^0$ is a wavelet basis function and therefore, the $N^0$ order integral coefficient $C^0$ is zero. In such a case, one of the following rules describes the ranked basis product $P^0 = P^{N-1} \cdot b^0$:

4. $b^{p^0}$ and $b^0$ have identical supports. $b^{p^0}$ is a scaling basis and $b^0$ is a wavelet basis function. In this case, $\|P^{N-1}\| = \|P^{N-1}\|$.  
5. $b^{p^0}$ and by have identical supports. $b^{p^0}$ is a wavelet basis function, $b^0$ is a child scaling basis function. In this case, $\|P^{N-1}\| = 2\|C^{N-1}\|$.  
6. $b^{p^0}$ and $b^0$ have identical supports. Each are wavelet basis functions of different basis function types. In this case, $\|P^{N-1}\| = 2\|P^{N-1}\|$ and $b^{p^0}$ is the wavelet basis function of the third type and the same support.

7. $b_{p^0}$ is a child basis function and $b^0$ is its parent basis function. $b_{p^0}$ is a wavelet basis function. In this case, $\|P^{N-1}\| = \|P^{N-1}\|$ where $j$ is the scale of $b^0$, and the sign is the sign of the portion of the support of $b^0$ that is covered by the support of $b_{p^0}$. $b_{p^0}$ is a wavelet basis function of the same type and support $b_{p^0}$.

FIG. 16 is a block diagram illustrating another embodiment of a method 1600 for determining the integral of the product of a plurality of functions. Like the method 100, the method 1600 is employed with reference to equation (10) above. Therefore, the method 1600 determines the integral $u$ of a plurality of functions that includes $N$ functions $F_i(v), i$ being an integer from 1 to $N$.

In block 1602, each of the plurality of functions $F_i(v)$ whose product is being integrated is projected into the wavelet domain. Projecting the functions $F_i(v)$ into the wavelet domain comprises, for example, performing the wavelet transform on each function to project the function onto the basis set $B$. The wavelet transform is the two-dimensional, non-standard Haar transform and the basis set is the two-dimensional, non-standard Elard-Haar basis set. Each of the functions $F_i(v)$ is then represented in the wavelet domain as the series of basis functions $f_j$, scaling basis functions $b_j$ of the basis set $B$. Specifically, the basis functions $b_j$ are the restricted basis functions of the basis set and the basis coefficients $f_j$ are the wavelet coefficients of the wavelet basis functions $\psi_j^0, \psi_j^1$ and $\psi_j^2$ (except for the basis coefficient of the mother scaling function $\phi$).

In block 1604, the basis coefficients $f_j$ of each function $F_i(v)$ are recorded in a wavelet tree $W_i$. The wavelet tree $W_i$ is, for example, a tree-shaped data structure that stores the basis coefficients $f_j$ of function $F_i(v)$, organizing the basis coefficients according to the parent-child relationships described above with reference to the basis function tree 700 and the basis coefficient tree 800. The wavelet tree $W_i$ has the coefficient of the mother scaling function $\phi$ at a root of the tree, and nodes $w_i$ depend from the root. Each node $w_i$ includes the basis coefficients $f_j$ of the restricted basis functions $b_j$ defined for the support $<j, k, l>$, and each node points to its immediate child nodes having supports $<j+1, k, l>$. For example, a mother node depending from the root stores the mother wavelet coefficients, and four immediate child nodes having supports $<1, k, l>$ depend from the mother node. The basis coefficients $f_j$ stored in the nodes $w_i$ are wavelet coefficients, because each of the restricted basis functions $b_j$ is of a wavelet type except for the mother scaling function $\phi$, the coefficient of which is stored in the root. Each node $w_i$ also has a variable that stores a signed parent summation of the node.

FIG. 17 illustrates an example wavelet tree $W_i$ in pseudo code 1700. In line 1, the data structure augtree is defined, the data structure augtree being one wavelet tree $W_i$. In line 2, a variable dc is defined to store the mother scaling coefficient.

In line 3, a pointer node is defined to point to a data structure augnode, which stores one node $w_i$ of the wavelet tree $W_i$. In line 5, the data structure augnode is defined. In line 6, an array $\psi[3]$ is defined to store the three wavelet coefficients of the node $w_i$. In line 7, a variable parentsum is defined to store the signed parent summation of the node $w_i$, which is described below. In line 8, an array of four pointers ch[4] is defined, the pointers of the array pointing to the node’s four immediate child nodes $w_i$, each of which also is stored using the data structure augnode.

A person of skill may note that the wavelet tree $W_i$ is similar to other data structures for encoding wavelet coefficients, such as the quadtree structure described by Berman et al. in "Multiresolution painting and compositing", in Proc.
trees

the insignificant basis coefficients. In such an embodiment, trees

within the set of nodes through the wavelet trees

wavelet trees

wavelet trees

results for each of the N functions whose product is being

SIGGRAPH 94, 85-90, 1994, or the zero-tree structure
described by Shapiro in “Embedded image coding using
zero-trees of wavelet coefficients”, IEEE Transactions on Sig-
nal Processing SP, 41 (Dec.), 3445-3462, 1993, both of which are
incorporated by reference in their entirety.

With reference back to FIG. 16, in block 1604 the basis coefficients \( f_{j,k} \) of the function \( F_i(v) \) are encoded in the wave-

let tree \( W_i \). In some embodiments, each function \( F_i(v) \) is encoded into its own wavelet tree \( W_i \) such that one wavelet tree \( W_i \) results for each of the N functions whose product is being integrated, i being an integer from 1 to N. In other embodiments, each function \( F_i(v) \) may be encoded in a plurality of wavelet trees \( W_i \).

In block 1606, non-linear approximation is performed on each function \( F_i(v) \) to discard insignificant basis coefficients \( f_{j,k} \). Non-linear approximation is known in the art, and therefore a detailed discussion is omitted here. For example, non-linear approximation is described in Devore, “Non-linear-approximation,” Acta Numerica 7, 51-150 (1998). Generally, however, non-linear approximation presupposes that if a basis coefficient \( f_{j,k} \) is insignificant with respect to a given threshold, then all of the basis coefficients corresponding to basis functions of higher scales \( j \) are also likely to be insignificant with respect to that threshold. Therefore, the insignificant basis coefficients \( f_{j,k} \) can be discarded.

In embodiments in which the non-linear approximation is performed after the basis coefficients \( f_{j,k} \) are encoded, the insignificant basis coefficients are discarded by removing the insignificant basis coefficients from the wavelet tree \( W_i \). In other embodiments, the non-linear approximation is performed before the basis coefficients \( f_{j,k} \) are encoded into the wavelet tree \( W_i \). In such embodiments, the insignificant basis coefficients \( f_{j,k} \) are discarded by only encoding the significant basis coefficients into the wavelet tree \( W_i \) without encoding the insignificant basis coefficients. In such an embodiment, blocks 1604 and 1606 of FIG. 16 are reversed. In still other embodiments, the non-linear approximation is omitted. In embodiments in which the non-linear approximation is omitted, determining the integral \( u \) of the plurality of functions \( F_i(v) \) is relatively more computationally complex and is relatively slower; however, the value determined for the integral may be more accurate. In still other embodiments, the non-linear approximation is performed on some but not all of the wavelet trees \( W_i \), such that some of the functions \( F_i(v) \) are approximated but others are not.

In block 1608, the integral \( u \) of the product of the plurality of functions \( F_i(v) \) is determined by traversing direct paths through the wavelet trees \( W_i \) that represent the functions, along which direct paths the Nth order integral coefficients \( C_N^m \) may be non-zero. Traversing direct paths through the wavelet trees \( W_i \) comprises synchronously traversing a plurality of wavelet trees \( W_i \) of the wavelet trees \( W_i \) storing the basis coefficients of the functions whose product is being integrated, at least one wavelet tree being traversed for each function \( F_i(v) \) whose product is being integrated. For example, exactly N wavelet trees \( W_i \) are synchronously traversed in embodiments in which each function \( F_i(v) \) is represented by exactly one wavelet tree \( W_i \).

As the wavelet trees \( W_i \) are traversed, a set of nodes \( \{w_1, \ldots, w_N\} \) are synchronously processed. The set of nodes \( \{w_1, \ldots, w_N\} \) includes one node \( w_i \) from each of the wavelet trees \( W_i \). The term “traversing direct paths through the wavelet trees \( W_i \)” denotes that the nodes \( w_i \) that are synchronously processed correspond to nodes in the basis function tree that lie on a single direct path through the basis function tree, as described above with reference to FIG. 7. In other words, within the set of nodes \( \{w_1, \ldots, w_N\} \), no two nodes \( w_i \) correspond to nodes of the basis function tree that are not on a direct path with each other. Alternatively stated, no two nodes \( W_i \) correspond to supports \( \{j,k,l\} \) that are completely disjoint from each other. Instead, for any two nodes \( W_i \) in the set of nodes \( \{w_1, \ldots, w_N\} \), the supports \( \{j,k,l\} \) to which the nodes \( W_i \) correspond are either the same or are related as parent and child. In other words, for any two nodes \( W_i \) in the set, the support \( \{j,k,l\} \) of one of the nodes \( W_i \) completely covers the support \( \{j,k,l\} \) of the other node \( W_i \).

Traversing the wavelet trees \( W_i \) on the direct paths employs the principles described above: the Nth order integral coefficient \( C_N^m \) of basis functions may be nonzero if each of the basis functions lie on a single direct path through the basis function tree; however, the Nth order integral coefficient \( C_N^m \) is zero if the basis functions do not lie on a single direct path through the basis function tree. Recall that the integral \( u \) of the product of the plurality of functions \( F_i(v) \) is the sum of a series of contributing products, as shown in equation (10). Each contributing product includes one basis coefficient from each function \( F_i(v) \) and one Nth order integral coefficient \( C_N^m \), which is the integral of the corresponding basis functions. Because the basis coefficients are stored in nodes \( w_i \) of the wavelet trees \( W_i \), the integral \( u \) can be determined by synchronously processing one node \( w_i \) from each wavelet tree \( W_i \). However, synchronously processing nodes \( w_i \) from disparate wavelet trees \( W_i \) that do not correspond to nodes on a single direct path through the basis function tree is of little value, because in such case, the Nth order integral coefficient \( C_N^m \) is necessarily zero. Therefore, confining the traversal to direct paths through the wavelet trees \( W_i \) while avoiding the indirect paths enables accumulating contributing products that could contribute to the integral \( u \) while avoiding those that necessarily do not contribute to the integral \( u \). As a result, traversing the wavelet trees \( W_i \) on direct paths determines the integral \( u \) with the same accuracy but with fewer computations than other systems and methods.

FIG. 18 is a block diagram illustrating an embodiment of a method 1800 for traversing direct paths through a plurality of wavelet trees \( W_i \) to determine the integral \( u \) of the product of the plurality of functions \( F_i(v) \), which can be employed in block 1608 of FIG. 16. In block 1802, the integral \( u \) is initially incremented with the contribution of the roots of the wavelet trees \( W_i \). The contribution of the roots is determined by multiplying together the mother scaling coefficient \( \phi \) stored in the root of each wavelet tree \( W_i \), because in such case the Nth order integral coefficient \( C_N^m \) is one. In block 1804, a set of nodes \( \{w_1, \ldots, w_N\} \) is synchronously processed. The set of nodes \( \{w_1, \ldots, w_N\} \) includes one node \( w_i \) from each of the wavelet trees \( W_i \). No two nodes \( w_i \) in the set correspond to supports \( \{j,k,l\} \) that are completely disjoint from each other, because the wavelet trees \( W_i \) are traversed on direct paths. In block 1806, a contribution to the integral \( u \) is determined for the set of nodes \( \{w_1, \ldots, w_N\} \). In some embodiments, the contribution of the set of nodes \( \{w_1, \ldots, w_N\} \) is determined by separating the nodes into a group of null nodes and a group of the non-null nodes, determining a contribution of the null nodes, and determining a contribution of the non-null nodes. The contribution of the null nodes is limited to the contribution of their parent nodes, which is not ignored even though the nodes themselves are null. The contribution of the non-null nodes is determined either by recursively expanding the non-null nodes to calculate the contribution of subsets of the non-null nodes, or by iterating through the non-null nodes to build a table that accumulates the magnitude of the contribution of the non-null nodes. Regardless of how the contribution of the set of nodes \( \{w_1, \ldots, w_N\} \) is determined, in block 1808 the integral \( u \) is incremented with the contribution determined.
in block 1806. In block 1810, a set of child nodes $[w_1, \ldots, w_N]$ is synchronously processed. The set of child nodes $[w_1, \ldots, w_N]$ includes one node $w_0$ from each of the wavelet trees $W_N$, the one node being the immediate child of the node most recently processed. Because the child nodes $w_0$ immediately depend from the most recently processed nodes, the wavelet trees $W_N$ are traversed along direct paths. The contribution of the set of child nodes $[w_1, \ldots, w_N]$ may then be determined in the manner described above, and the integral may be incremented, and the process may be repeated for subsequent sets of child nodes until the last set of nodes is reached.

In at least some embodiments, traversing direct paths through wavelet trees $W_N$ in block 1608 of FIG. 16 comprises applying a tree-traversal algorithm to the wavelet trees. FIG. 19 illustrates in pseudo code 1900 an example tree-traversal algorithm 1902 for determining the integral $u$ and embodiments of component routines, such as routine 1904, routine 1906, and routine 1908. The algorithm 1902 is defined as FunctionProductIntegral in line 1. The algorithm 1902 accepts as input a set of wavelet trees $(W_1, \ldots, W_N)$, one for each function $F(v)$ whose product is being integrated. In line 2, the integral $u$ is initially incremented to the product of the mother scaling functions, by multiplying together the variable $W_Ndc$ of each wavelet tree in the set $(W_1, \ldots, W_N)$. In such case, the $N^{th}$ order integral coefficient $C_N$ is one, because regardless of $N$ the integral coefficient of a series of mother scaling functions $\Psi^0_j$ is one and need not be determined to increment the integral $u$. In line 3, a routine traverseAugTrees is called, which is the routine 1904. The routine 1904 is configured to simultaneously process a set of nodes $(w_1, \ldots, w_N)$ that includes one node $w_0$ from each wavelet tree $W_N$ and to increment the integral $u$ as a result of its processing of the set of nodes. For its initial call, the routine 1904 processes the set of nodes $(W_1, \ldots, W_N)$, which are the mother nodes that include the mother wavelet coefficients. The routine 1904 then iteratively calls itself to capture the contributions of later sets of nodes, traversing the set of wavelet trees $(W_1, \ldots, W_N)$ from the top down along direct paths through the wavelet trees.

Before the routine 1904 is described in detail, the signed parent summation of the node $w_0$, which is stored in the variable parentsum of the node $w_0$, is described. The mother node $w_0$ that holds the mother wavelet coefficients has the value of the mother scaling coefficient $dc$ as its signed parent summation. For all other nodes $w_i$, the signed parent summation is the signed parent summation of its immediate parent node $p_i$ plus the sum of the wavelet coefficients stored in the immediate parent node $p_i$, scaled by the magnitude of the product of two basis functions, one in the node $w_i$ and the other in the immediate parent node $p_i$, as shown in equation (23):

$$w_{i, \text{parentsum}} = p_{i, \text{parentsum}} +$$

$$+ \prod_{j=1}^{i-1} [p_{j, \alpha} \cdot \text{sign}(q_j, q_j)] + p_{i, \beta} \cdot \text{sign}(1, q_i, q_i)] +$$

$$\prod_{j=1}^{i-1} [p_{j, \beta} \cdot \text{sign}(2, q_i, q_i)]$$

where $p_i$ is the immediate parent of the node $w_i$, $p_i, \alpha$ is the wavelet coefficient in the parent node corresponding to the basis function of type $\Psi_1$, $p_i, \beta$ is the wavelet coefficient in the parent node corresponding to the basis function of type $\Psi_2$, $p_i, y$ is the wavelet coefficient in the parent node corresponding to the basis function of type $\Psi_3$, $(q_i, q_i)$ is the quadrant of the immediate parent node $p_i$ that is covered by the node $w_i$, as described below with reference to equation (24), and the product is an array described below with reference to equation (25).

The node $w_0$ corresponds to a unique support $(j,k,l)$ that covers a quadrant $(q_1, q_2)$ of its immediate parent node $p_i$, where the values $q_1$ and $q_2$ are:

$$q_{i, k} \mod 2$$

This quadrant $(q_1, q_2)$ is used to determine the sign of the signed parent summation in conjunction with the product array that stores the signs of the four quadrants of the three mother wavelet functions $\Psi_1^0, \Psi_2^0, \Psi_3^0$, respectively, as shown in equation (25):

$$\text{sign}(0, 1, 1, 1, 1, 1, 1, 1)$$

A person of skill may note that the variable $w_0, \text{parentsum}$ is inspired by Ng in “Triple product wavelet integrals for all-frequency relighting”, ACM Transactions on Graphics (SIGGRAPH '04) 23, 3, 477-487, which is incorporated by reference herein in its entirety.

An accumulated parent summation is stored in a variable cum. Specifically, the variable cum is initially set to one, which is the $N^{th}$ order integral coefficient $C_N$ of a series of mother scaling functions $\Psi^0_j$, regardless of $N$. As the set of wavelet trees $(W_1, \ldots, W_N)$ are traversed, the accumulated parent summation is incremented as described below.

The routine 1904 will now be described. In line 1, the routine 1904 is defined as traverseAugTrees, accepting as inputs the variable cum and the sets of nodes $(w_1, \ldots, w_N)$. In line 2, the nodes of the set $(w_1, \ldots, w_N)$ are reorganized into a two sets of nodes: one being a set of non-null nodes $(w_1, \ldots, w_k)$ and the other being a set of null nodes $(w_{k+1}, \ldots, w_{N})$. Up to this point, the subscript $i$ has denoted that a specific wavelet tree $W_i$ corresponds to a specific function $F_j(v)$, or alternatively that a specific node $w_i$ is a node of a specific wavelet tree $W_i$. For the remainder of the discussion of the algorithm 1902, the subscript $i$ merely indicates whether the node $w_i$ is null or non-null, with non-null nodes having subscripts $w_i$ to $w_N$ and null nodes having subscripts $w_{k+1}$ to $w_N$. It is likely that some of the nodes $w_i$ are null in embodiments in which non-linear approximation is performed.

In line 3, the routine 1904 returns if at most one of the nodes in the set $(w_1, \ldots, w_N)$ is null. In such case the $N^{th}$ order integral coefficient $C_N$ is zero, because each node $w_i$ only includes wavelet coefficients and the integral of each wavelet basis functions $\Psi_1, \Psi_2$, and $\Psi_3$ is zero.

The set of non-null nodes $(w_1, \ldots, w_k)$ can contribute to the integral $u$, however, the contribution of the set of null nodes $(w_{k+1}, \ldots, w_N)$ is limited to the contribution of their parent nodes $p_i$. Therefore, the contribution of the nodes $(w_1, \ldots, w_k)$ to the integral $u$ is determined by first updating the signed parent summation $w_i, \text{parentsum}$ for each node $(w_1, \ldots, w_k)$ in line 4, updating the accumulated parent summation cum with the contributions of only the null nodes $(w_{k+1}, \ldots, w_N)$ in line 5, and calling a routine getProductIntegral to determine the contributions of the non-null nodes $(w_1, \ldots, w_k)$ in line 6. The integral $u$ is then incremented with the product of the accumulated parent summation cum and the return of getProductIntegral, described below, such that the contributions of both the non-null nodes and the parents of the null nodes are captured.

In lines 7 and 8, the routine traverseAugTrees 1904 then iteratively calls itself four times to independently process the next four sets of nodes, which are the immediate child nodes of the non-null nodes. In other words, the routine 1904 calls
itself for the nodes (w1, ch[i], ..., wC, ch[i]) in a loop, where ch[i] is the pointer to the child node and i is an integer from 0 to 3.

The routine getProductIntegral 1906, which is called by the routine 1904 to determine the contribution of the non-null nodes (w1, ..., wC), recursively calculates the non-zero Nth order integral coefficients C^N using the first three cases of FIG. 15. In line 1, the routine 1906 accepts a plurality of nodes w as input, initially processing the set of non-null nodes (w1, ..., wC). In lines 3-6 the routine 1906 recursively calls itself and three instances of a routine getWaveletProduct 1908, passing in one fewer nodes with each call, and returning the sum of the four routines scaled by variables of the node not passed in for the call. Specifically, the routine 1906 is scaled by the parentsum of the node w, not passed in and each return of the routine 1908 is scaled by one of the three wavelet coefficients of the node w, not passed in. In line 2, the routine returns zero if only one node w, is passed in.

The routine getWaveletProduct 1908 uses cases 4, 6, and 7 of FIG. 15 to evaluate the magnitude of the product where the type of the basis function appearing in the product is one of the wavelet basis functions. In line 1, the routine getWaveletProduct 1908 accepts as input a plurality of nodes w, and three input parameters a, b, and c that differentiate the three different wavelet types. In lines 3-7, the routine getWaveletProduct 1908 recursively calls the routine getProductIntegral 1906 and three instances of itself, passing in one fewer nodes with each call, and returning the sum of the four routines scaled by variables of the node not passed in for the call, plus the parent variables of the fewer nodes scaled by one of the wavelet coefficients of the node not passed in for the call. In line 2, the routine getWaveletProduct 1908 returns one of the wavelet coefficients if only one node w, is passed in.

The embodiment of the tree-traversal algorithm Function-ProductIntegral 1902 may have a computational complexity on the order of O(mN^4), where m is the number of significant basis coefficients f_m retained after the nonlinear approximation, and N is the number of functions F_j(x) whose product is being integrated. By eliminating repetitive operations, the algorithm 1902 can be optimized such that the computational complexity is on the order of O(mN^4). For example, the embodiment of the routine getProductIntegral 1906 traverses the wavelet trees W_k from the top down, recursively expanding the nodes w_k, to calculate the product of one fewer nodes, but in other embodiments of the routine the recursive expansion is eliminated to increase the speed of the algorithm.

For example, FIG. 20 illustrates in pseudo code a 2000 an optimized tree-traversal algorithm 2002 for determining the integral u, and embodiments of component routines, such as the routine 2004, the routine 2006, the routine 2008, and the routine 2010. As shown, the algorithm 2002 is the same as the algorithm 1902 and the routine 2004 is the same as the routine 1904 described above. However, the routine getProductIntegral 2006 differs from the routine getProductIntegral 1906. Specifically, instead of recursively expanding the nodes w_k from the top down to determine the Nth order integral coefficients C^N, the routine getProductIntegral 2006 employs an intermediate table T [N], an embodiment of which is illustrated with example pseudo code 2100 in FIG. 21.

In line 1 of FIG. 21, the table T [N] is defined to have N records, the fields of which are described with reference to a subset of nodes in line 2, the field \( \phi \) is defined, which stores the magnitude of the product of a subset of the nodes w that have a scaling basis function \( \phi \) appearing in the product. In line 3, the field \( \psi[0] \) is defined. The field \( \psi[0] \) stores the magnitude of the product of a subset of the nodes w that have a wavelet basis function of type \( \psi \) appearing in the product.
which employs the table, may have computational complexity on the order of $O(n^2)$. Of course, the computational complexity of the algorithm affects the speed with which the integral $u$ is determined, and therefore reducing the computation complexity is desirable.

The computational complexity can be further controlled by varying the traversal depth of the tree-traversal algorithm. Traversing at higher traversal depths denotes traversing subsets of nodes that have relatively lower scales and are located relatively closer to the root of the wavelet tree, while traversing at lower traversal depths denotes traversing subsets of nodes that include relatively higher scales and are located relatively farther from the root node. In some embodiments, the traversal depth of the algorithm may be controlled, such that the need for fast computation can be balanced against the need for accurate computation on a case by case basis.

FIG. 22 is a block diagram illustrating an embodiment of a system 2200 for determining the integral $u$ of the product of a plurality of functions $F_i(v)$. The system includes logic 2202 configured to project the functions $F_i(v)$ of the plurality of functions into the wavelet domain. Logic 2204 configured to encode basis coefficients $f_{jk}$ of each function in a wavelet tree $W_n$. Logic 2206 configured to perform non-linear approximation on the functions $F_i(v)$ to discard insignificant basis coefficients $f_{jk}$, and logic 2208 configured to traverse direct paths through the wavelet trees $W_n$, where the unit order integral coefficients are non-zero, to determine the integral $u$ of the product of the functions $F_i(v)$ represented by the wavelet trees $W_n$.

In some embodiments, the systems and methods described above are employed by graphics rendering applications to re-light and shade objects in a scene. Typically, the graphics rendering application models objects in the scene, and the lighting and shading of the objects is determined using a light transport model that simulates the interaction of light with the objects. FIG. 23 is a diagram illustrating an example scene 2300. An object 2302 in the scene 2300 has a point $x$ on a surface 2304 of the object, and a radiance $B$ exiting the point $x$ is a function of a direction of view $\theta$. Generally, determining the radiance of the point $x$ requires integrating the product of a plurality of functions that contribute to the model, as shown in equation (26):

$$B(x, \theta) = \int_{-1}^{1} L(p, \phi)p(x, \phi \leftrightarrow \theta)O_i(x, \phi)(D \cdot \phi) \int_{-1}^{1} O_i(x, \phi) d\phi$$  \hspace{1cm} (26)

FIG. 24 is a block diagram illustrating an embodiment of a method of rendering a graphical scene in which the radiance of a point $x$ is in the scene is determined by integrating the product of a plurality of functions contributing to a light transport model of the scene. The method 2400 will be described with reference to the example scene 2300 shown in FIG. 23. In block 2402, the plurality of functions that contribute to the light transport model of the scene 2300 are determined. The plurality of functions include, for example, a distant environment lighting function $L(\phi)$, a bi-directional reflectance distribution function (BRDF) $p(x, \phi \leftrightarrow \theta)$, a local visibility function $O_i(x, \phi)$, a cosine term $(D \cdot \phi)$ and a plurality of dynamic occlusion functions $O_i(x, \phi)$. The distant environment lighting function $L(\phi)$ represents distant environment lighting as a function of a vector $\phi$, the vector $\phi$ representing the incident direction of the light. The BRDF $p(x, \phi \leftrightarrow \theta)$ models the interaction of the incident light $I$ with the surface 2304 of the object 2302. The local visibility function $O_i(x, \phi)$ represents the local visibility at the point $x$ due to self-occlusion.

The cosine term $(D \cdot \phi)$ represents Lambert’s law: the incident radiance $B$ is directly proportional to the dot product of a vector $D$ and the vector $\phi$, the vector $D$ being a vector that is normal to the surface 2304 of the object 2302 at the point $x$. Each dynamic occlusion function $O_i(x, \phi)$ represents the dynamic occlusion at the point $x$ caused by an $i$th neighboring object 2306 in the scene 2300, where $i$ can be any integer from 1 to $g$, being the number of neighboring objects 2306 that contribute to the light transport model. Three neighboring objects 2306 are shown in FIG. 23 for illustrative purposes, although greater or fewer neighboring objects could be included.

In some embodiments, the cosine term $(D \cdot \phi)$ can be combined with the local visibility $O_i(x, \phi)$ term as shown in equation (27):

$$B(x, \theta) = \int_{-1}^{1} L(p, \phi)p(x, \phi \leftrightarrow \theta)O_i(x, \phi)(D \cdot \phi) \int_{-1}^{1} O_i(x, \phi) d\phi$$

Thus, the exitant radiance $B$ at the fixed point $x$ is the integral of the product of $g+3$ functions, $g$ being the number of neighboring objects 2306 that could dynamically occlude and cast a shadow upon the point $x$. To determine the exitant radiance $B$, the product of the functions is integrated with respect to all incident directions $\phi$ surrounding the point $x$.

For example, in FIG. 23 a cubemap $S$ surrounds the point $x$, and in equation (27) the integral is over the cubemap $S$. However, in equation (28) the integral is over one face of the cubemap $S$. Therefore, to determine the exitant radiance $B$ using equation (28), the integral is taken six times, once for each face of the cubemap $S$, and the six iterations are summed together. In such an embodiment, each function that contributes to the light transport model of the scene is represented as a cubemap function 2310, and in block 2402 determining the plurality of functions comprises determining the cubemap functions.

Example cubemap functions 2310 are shown in FIG. 23, where the distant environment lighting function $L(\phi)$ is represented by the cubemap function 2312, the BRDF $p(\phi)$ is represented by the cubemap function 2314, the local visibility function $O_i(\phi)$ is represented by the cubemap function 2316, and three dynamic occlusion functions $O_i(\phi)$, $O_0(\phi)$, and $O_3(\phi)$ are represented by the cubemap functions 2318, 2320, and 2322, respectively, one for each neighboring object 2306 in the scene 2300. Again, three neighboring objects 2306 are illustrated by way of example. Although the cubemaps functions 2310 can be determined in a variety of ways that are known or will later be known to a person of skill in the art, one example embodiment of determining the cubemap functions is described below.

In the example embodiment, data is collected about the scene by sampling the objects and lighting within the scene. The sampled data is then processed to create the cubemap functions 2310. The sampled data may compiled interactively.
or in advance of run-time, and in cases in which the sampled data is compiled in advance, the cubemap functions 2310 may be pre-computed in advance or computed interactively.

The lighting cubemap function 2312 is interactively determined from interactively sampled data of the high dynamic range illumination, as described in Debevec et al., “Recovering high dynamic range radiance maps from photographs,” In Proc. SIGGRAPH ’97, 360-378, which is incorporated by reference herein in its entirety. For example, the illumination may be sampled at a resolution of 6x6x6x64, where 6 denotes the number of faces 2308 of the cubemap and 64x64 denotes the number of samples taken per face. Sampling the illumination at a resolution that is the same as the resolution used to sample the visibility and BRDF may be desirable because, sampling at a higher resolution merely captures basis coefficients of a finer scale that will not contribute to the light transport model if comparable basis coefficients are not captured for other functions.

The BRDF cubemap function 2314 is interactively determined from pre-computed data. Prior to run-time, the pre-computed data is tabulated by, for example, sampling Phong BRDF’s with a resolution in θ, ϕ of up to (6x6x6x64)x(6x64x64), where θ, is a reflection vector of the direction of view θ about the vector D that is normal to the surface. The Phong BRDF’s can have a shininess of up to, for example, 200. In some embodiments, the pre-computed BRDF data can be gathered as described in the Ng paper previously incorporated by reference. At run-time, the BRDF cubemap 2314 is interactively interpolated from the tabulated data.

The local visibility cubemap function 2316 is pre-computed using pre-computed data, and the dynamic occlusion cubemap functions 2318, 2320, and 2322 are determined interactively using pre-computed data. For each object 2302, 2306 in the scene 2300, a local visibility field of the object is sampled at points x on its surface 2304. A global visibility field is sampled in nearby surrounding regions, such as on a virtual ground plane of the object that is a lower plane of a bounding box of the object. Sampling the global visibility field is accomplished using schemes such as a planar sampling scheme and/or a spherical sampling scheme. The planar sampling scheme samples the visibility in the form of concentric circles on each object’s virtual ground plane. The center of each concentric circle is the projection of the object center on the virtual ground plane, the radius of the concentric circles varying from about 0.02r to about 10r, r being a radius of the projection of the object on the virtual ground plane. For example, one hundred concentric circles may be used, with about 200 sample points being collected per concentric circle. The spherical sampling scheme samples the visibility of the region surrounding each scene entity in the form of concentric spheres. The center of the concentric spheres may coincide with the centers of the concentric circles. Twenty concentric spheres may be used, the radius of the concentric spheres varying from about 0.2r to about 6r, R being the radius of the bounding sphere. Each concentric sphere is sampled at about 6x9x9. Such a sampling scheme is similar to the object occlusion field described in Zhou et al., “Precomputed shadow field for dynamic scenes,” ACM Transactions on Graphics (SIGGRAPH ’05), 24, 3, 1196-1201, which is incorporated by reference herein in its entirety. The difference between the radiiuses of neighboring concentric circles and spheres, in the planar and spherical sampling schemes respectively, increases linearly with increasing distance from the projection of the object center on the virtual ground plane.

The local visibility cubemap function 2316 and the dynamic occlusion cubemap functions 2318, 2320, and 2322 are pre-computed from the sampled data by rasterizing the coarse model using graphics hardware. Each cubemap is rasterized at a resolution of 6x6x6x64. In some embodiments, multiple objects 2302, 2306 in the scene 2300 share the same geometry and visibility field, in which case each of the objects is represented by the same cubemap function 2310. For example, if a scene includes multiple identical chairs, the cubemap function 2310 of the chair is determined only once.

As can be seen from equation (28), the light transport model is a specific implementation of equation (10). Equation (10) generically represents determining the integral u of the product of a plurality of functions including N generic functions F(v). In equation (28), the N functions F(v) are the functions that contribute to the light transport model, including the distant environment lighting function L(ϕ), the bi-directional reflectance distribution (BRDF) function R(ϕ), the local visibility function combined with the cosine term O1(x, ϕ), where i is an integer from 1 to g.

Because the light transport model is known in the art, a through discussion of the model is omitted. However, a person of skill would understand that the above equations of the light transport model are merely examples, and in other embodiments, the model may take other forms. For example, in some embodiments some of the functions that contribute to the light transport model may be omitted, or additional functions may be included. Additionally, the functions that contribute to the light transport model need not be represented as cubemap functions 2310, or the cubemap functions can be determined in manners other than those described in the example embodiment above.

Once the functions that contribute to the light transport model are determined in block 2402, the method 2400 is similar to the method 1600. This is because the method 1600 was employed with reference to equation (10) and because the light transport model of equation (28) is a specific implementation of equation (10), as described above. In block 2404, each of the plurality of functions that contribute to the light transport model is projected into the wavelet domain, as described above with reference to block 1602 of FIG. 16. In embodiments in which the functions are represented as cubemaps 2310, the sampling resolution used to determine the cubemap function may determine the resolution of the basis set B onto which the cubemap function is projected. For example, in embodiments in which the sampling resolution of 6x6x6x64 is used, the basis set B may have a resolution of 7, although other resolutions may be used.

In block 2406, the basis coefficients of each function are encoded in a wavelet tree W, in embodiments in which the functions are represented as cubemap functions 2310, encoding the basis coefficients of the functions in the wavelet trees W comprises encoding the basis coefficients of each face 2308 of each cubemap function 2310 in a separate wavelet tree W. Thus, six wavelet trees W are encoded for each cubemap function 2310, one per face 2308 of the cubemap function 2310. In other embodiments, each function can be represented with greater of fewer wavelet trees W.

In block 2408, non-linear approximation is performed on each function to discard insignificant basis coefficients, as
described above with reference to block 1606 of FIG. 16. In some embodiments, the non-linear approximation block 2408 is performed before the encoding block 2406, the non-linear approximation block 2408 is performed on only a subset of the functions, or the non-linear approximation block 2408 is omitted completely. In embodiments in which all-frequency lighting is desirable, the distance environment lighting function \(L(q)\) is approximated using a relatively larger number of basis coefficients than the other functions, such that the high-frequency components of the lighting are represented. For example, in embodiments in which about 60 to 120 basis coefficients are retained per face 2308 of the cubemap functions 2314 to 2322, up to 300 basis coefficients may be retained per face 2308 of the lighting cubemap function 2312, each face corresponding to one wavelet tree \(W,\).

In block 2410, the radiance \(B\) of the point \(x\), which is the integral of the product of the functions that contribute to the light transport model, is determined by traversing direct paths through the wavelet trees \(W,\) where the \(N^n\) order integral coefficients \(C_n\) may be nonzero, as described above with reference to block 1606 of FIG. 6. For example, traversing direct paths through the wavelet trees \(W,\) may comprise performing the method 1800 described above with reference to FIG. 18 or applying a tree-traversal algorithm to the wavelet trees \(W,\) such as either the algorithm 1902 or the algorithm 2002 described above with reference to FIG. 19 and FIG. 20.

In embodiments in which the functions are represented as cubemap functions 2310, traversing the wavelet trees \(W,\) may determine the radiance of the point \(x\) over one face 2308 of the cubemap \(S\). In some such embodiments, each function is represented by a plurality of wavelet trees \(W,\) as a result of the faces 2308 of the cubemap functions 2310 being encoded in separate wavelet trees \(W,\). In such cases, traversing the wavelet trees \(W,\) comprises traversing sets of wavelet trees to determine the radiance \(B\) over face 2308, each set corresponding to one face 2308 of the cubemap and each set including one wavelet tree \(W,\) for each function. In other words, the wavelet trees \(W,\) that correspond to a given face 2308 are traversed to determine the radiance \(B\) per face, and the six radiance \(B\) per face 2308 are summed to determine the radiance \(B\) over the entire cubemap \(S\).

To render the point \(x\) in block 2412 the graphics rendering application may require three colors, each for one of three independent color channels. However, the radiance \(B\) over the entire cubemap \(S\) may represent one color value for the point \(x\). This is because, for example, the lighting function \(L(q)\) and BRDF function \(p(q)\) may be different for different color channels at a single point \(x\). In such cases, traversing the wavelet trees \(W,\) comprises iteratively traversing the wavelet trees to determine three color values for the point \(x\). Therefore, the tree traversal is iterated six times, once per face of the cubemap to determine the first color, and the six iterations are repeated for the second and third color values. As a result, the tree traversal is iterated eighteen times per point \(x\), once per cubemap face 2308 per color channel. In some embodiments, the depth to which the wavelet trees \(W,\) are traversed is interactively controlled, the purpose of which is described below.

In block 2412, the scene is rendered. Because graphics rendering is known, a thorough discussion is omitted here. Rendering generally comprises, for example, using a graphics rendering application to rasterize the scene using graphics hardware. The graphics rendering application sets the color of the point \(x\) using the color values determined in block 2410. The algorithm then iterates for the next point \(x\), each visible point \(x\) being processed, a plurality of points \(x\) producing objects, and a plurality of objects producing the rendered scene.

The method 2400 can be used to interactively render dynamic, high-glossy objects 2302 with realistic, all-frequency shadows. The shadows cast by neighboring objects 2306 on the object 2302 appear in the rendered scene due to the dynamic occlusion functions \(O(x,q)\), even if the objects are moving. The shadows cast on an object 2302 by itself also appear in the scene, due to the local visibility function \(O(x,q)\). Further, the shadows cast on the virtual ground plane of the objects are represented, because the visibility on the virtual ground plane was considered in creating the cubemap functions. The BRDF \(p(q)\) enables reproducing the lighting effects caused by materials and textures in the scene, including glossy materials, and the distant environment lighting function \(L(q)\) can describe all-frequency lighting, such that specular highlights and non-diffuse shadows, including material specific highlights and shadows, appear in the rendered scene. Both the lighting and the direction of view can be varied, and the objects can be interactively manipulated. Note that any or all of the above-described effects can be simultaneously or individually produced, because each of the above-described functions simultaneously contributes to the integral \(B\). Therefore, the method 2400 has applicability in fields such as industrial lighting and computer gaming, among others, in which fields interactive lighting is desirable.

Further, the speed of rendering and/or richness of the lit scene can be interactively controlled by varying the traversal depth. Limiting the traversal to relatively higher (lower scale) nodes \(w,\) in the wavelet trees \(W,\) reduces the computational complexity, which is related, such as linearly related, to the number of nodes \(w,\) processed. However, in cases in which the traversal depth is limited, the relatively lower (higher scale) nodes \(w,\) may not be traversed, and therefore the contributions of these nodes to the lighting of the scene are not considered. Because the lower nodes \(w,\) correlate to higher scales, generally representing more resolved or higher-frequency information, not traversing the lower nodes \(w,\) may eliminate higher-frequency components of shadows and view-dependent specularities from the light transport model and therefore the rendered scene. Conversely, traversing relatively lower nodes \(w,\) enables the contributions of relatively more basis coefficients to be considered, the additional basis coefficients including higher-frequency and/or more resolved lighting information. Therefore, the rendered scene may have more detail but may take longer to render due to the increased complexity of determining the lighting.
In block 2602, the plurality of functions $F_i(v)$ whose product is being integrated are factored into sets of functions. Generally speaking, the plurality of functions $F_i(v)$ includes $N$ functions, and the $N$ functions can be factored into a plurality of sets. For example, in equation (29) the $N$ functions are factored into two sets, a first set having functions $F_i(v)$ for $i$ from 1 to $p$, and a second set having functions $F_i(v)$ for $i$ from $(p+1)$ to $N$:

$$u = \int \left[ \prod_{i=1}^{N} F_i(v) \right] dv = \int \left[ \prod_{i=1}^{p} F_i(v) \right] \cdot \int \left[ \prod_{i=p+1}^{N} F_i(v) \right] dv$$

For each set, the product of the functions $F_i(v)$ in the set can be represented in the wavelet domain as a vector $T$. Continuing the above example, the product of the functions of the first set can be represented as a vector $T_1$, and the product of the functions of the second set can be represented as a vector $T_2$, as shown in equation (30):

$$T_1 = \prod_{i=1}^{p} F_i(v) \text{ and } T_2 = \prod_{i=p+1}^{N} F_i(v)$$

In cases in which the products of the functions in the set are represented as vectors, the integral $u$ of the product of the functions is the integral of the product of vectors, or alternatively, the inner product of the vectors. For example, the integral of the product of the functions shown in equation (29) can alternatively be expressed as shown in equation (31):

$$u = \langle T_1, T_2 \rangle$$

where $T_1$ and $T_2$ are the vectors defined in equation (29), and $\langle T_1, T_2 \rangle$ denotes the inner product of the vectors.

The integral $u$ can be approximated by fixing at least one of the vectors in advance while allowing at least one of the vectors to vary. In cases in which at least one of the vectors is fixed, approximating the integral is relatively less computationally complex, and therefore relatively faster, than in cases in which none of the vectors are fixed. For example, in equation (31), the vector $T_1$ can be fixed while the vector $T_2$ varies, or vice versa.

Although in equations (29) through (31) the $N$ functions are factored into two sets and one of the sets is fixed, a person of skill would understand that the $N$ functions can be factored into more than two sets, each of the sets having one or more functions, and that one or more of the sets can be fixed or can vary. The remainder of the discussion is directed to an embodiment in which all of the functions $F_i(v)$ are fixed except for one function that is permitted to vary, as shown in equations (32) to (34). In such an embodiment, the relative computational complexity is further decreased, and the relative speed of computation is further increased.

In block 2602, the plurality of functions $F_i(v)$ whose product is being integrated are factored into a set of fixed functions and one variable function as shown in equation (32):

$$u = \int \prod_{i=1}^{N} F_i(v) dv = \int \left[ \prod_{i=1}^{p} F_i(v) \right] \cdot F_K(v) dv$$

In equation (32), the set of fixed functions is the set $[F_1(v), \ldots, F_{N-1}(v)]$, and the one varying function is the function $F_K(v)$.

In block 2604, a first vector $T$ is determined. The first vector $T$ represents the product of the set of fixed functions $[F_1(v), \ldots, F_{N-1}(v)]$ in the wavelet domain as a series of basis coefficients $(t_1, \ldots, t_M)$, or alternatively as the sum of the series of basis coefficients $t_k$ scaling corresponding basis functions $b_k$ as shown in equation (33):

$$T = \{ t_1, \ldots, t_{N-1}, t_M \}$$

where

$$t_k = \sum_{h=1}^{M} [ f_h \cdot b_h(v)]$$

In equation (33), $f_h$ is the basis coefficient corresponding to the $h^{th}$ basis function in the basis set $b$, and $M$ is the number of basis functions used to represent the product of the set functions in the wavelet domain. The first vector $T$ is determined using a method described below with reference to FIG. 27, which is a block diagram illustrating an embodiment of a method for determining the basis coefficients of a vector representing the product of a plurality of functions in the wavelet domain. In some embodiments, determining the first vector $T$ comprises pre-computing the first vector $T$ prior to run-time. The first vector $T$ can be pre-computed because the functions whose product is represented by the first vector $T$ are fixed in advance of run-time.

In block 2606, a second vector $F$ is determined. The second vector $F$ is the wavelet domain representation of the one varying function $F_K(v)$, expressing the function $F_K(v)$ as a series of basis coefficients $(f_1, \ldots, f_M)$, or alternatively as the sum of a series of basis coefficients $f_h$ scaling corresponding basis functions $b_h$, as shown in equation (34):

$$F = \{ f_1, \ldots, f_{N-1}, f_M \}$$

where

$$F_K(v) = \sum_{h=1}^{M} [ f_h \cdot b_h(v)]$$

Determining the second vector $F$ can comprise, for example, projecting the one varying function $F_K(v)$ into the wavelet domain by performing a wavelet transform, encoding the basis coefficients $(f_1, \ldots, f_M)$ of the second vector $F$ in a wavelet tree $W_K$, and performing non-linear approximation on the one varying function $F_K(v)$, as described above with reference to blocks 1602 through 1606 of FIG. 16. In some embodiments, determining the second vector $F$ comprises computing the second vector $F$ at run-time. Computing the second vector at run-time, whether interactively or in real-time, may be desirable because the function $F_K(v)$ is permitted to vary, in which case the second vector $F$ varies also.

In block 2608, the inner product of the first vector $T$ and the second vector $F$ is determined to approximate the integral $u$ of the product of the functions represented by the vectors, as shown in equation (35):

$$u = \langle T, F \rangle = \langle t_1, f_1 \rangle + \ldots + \langle t_M, f_M \rangle$$

In equation (35), the integral $u$ is an approximation because the first vector $T$ is fixed and does not vary. However, because the first vector $T$ is fixed, approximating the integral $u$ using the method 2600 is relatively less computationally complex, and therefore relatively faster, than other methods. For example, in embodiments in which the first vector $T$ is pre-
computed in advance of run-time, the method 2600 can be employed to approximate the integral \( u \) interactively and/or in real-time. In such embodiments, both the second vector \( T \) and the inner product of the vectors are computed interactively and/or in real-time.

FIG. 27 is a block diagram illustrating a method 2700 for determining the basis coefficients of a vector representing the product of the functions in the wavelet domain. For illustrative purposes, the method 2700 is described with reference to an example embodiment, in which the plurality of functions is the set of fixed functions \( \{F_1(v), \ldots, F_{N-1}(v)\} \) shown in equation (33). However, a person of skill would understand that the method 2700 generally determines a vector that represents the product of a plurality of functions in the wavelet domain, independent of the method 2600 and regardless of the functions whose product is being represented.

As shown in equation (33), the product of a plurality of functions \( F_i(v) \) can be expressed as a series of basis coefficients \( \{t_1, \ldots, t_N\} \) corresponding basis functions. The series of basis coefficients \( \{t_1, \ldots, t_N\} \) constitutes a vector that represents the product of the plurality of functions \( F_i(v) \) in the wavelet domain. The basis coefficients \( \{t_1, \ldots, t_N\} \) of the vector \( T \) can be determined by projecting the vector onto the basis functions of the basis set \( B \). In such case, the \( k^{th} \) basis coefficient \( t_k \) of the vector \( T \) is determined by taking the dot product of the basis vector and the corresponding basis function \( b_k \) of the basis set \( B \) as shown in equation (36):

\[
t_k = (F_k, b_k)
\]  

Replacing the vector \( T \) in equation (36) with the product of the plurality of functions from equation (33) yields equation (37):

\[
b_k = \int \prod_{h=1}^{N} \sum_{i=1}^{M} (f_{ih}, b_{ih}) \cdot b_k \, d\nu
\]

In other words, the \( k^{th} \) basis coefficient \( t_k \) of the vector \( T \) is the sum of a series contributing of products. Each contributing product in the series is the product of multiple basis coefficients \( f_{ih} \) and one \( N^{th} \) order integral coefficient \( CN \), the multiple basis coefficients including one basis coefficient \( f_{ih} \) from each of the \( N-1 \) functions, and the \( N^{th} \) order integral coefficient \( CN \) being the integral of the product of the basis functions \( b_{ih} \) that correspond to those basis coefficients \( f_{ih} \) along with the basis function \( b_k \) that corresponds to basis coefficient \( t_k \), being determined. It should be noted that \( h \) is an integer from 1 to \( M \) and \( i \) is an integer from 1 to \( N-1 \), \( M \) being the number of basis functions \( b_{ih} \) used to represent a function \( F_i(v) \) and \( (N-1) \) being the number of functions \( F_i(v) \) whose product is represented by the vector \( T \).

Because the similarities between equation (37) and equation (10) are immediately apparent, the differences between these equations are now described. Equation (10) determines the integral \( u \) of the product of a plurality of functions \( F_i(v) \), while equation (37) determines one basis coefficient \( t_k \) of a vector \( T \) that represents the product of a plurality of functions \( F_i(v) \) in the wavelet domain. Although either equation generally applies to any number of functions, equation (10) is expressed in terms of a plurality of functions \( F_i(v) \) that includes \( N \) functions, and equation (37) is expressed in terms of a plurality of functions \( F_i(v) \) that includes \( N-1 \) functions.

Both equations require taking the sum of a series of contributing products, each contributing product one including one basis coefficient \( f_{ih} \) from each function \( F_i(v) \) and one \( N^{th} \) order integral coefficient \( CN \). In equation (10), each contributing product in the series includes \( N \) basis coefficients and in equation (37), each contributing product in the series includes \( (N-1) \) basis coefficients. While the number of basis coefficients in a contributing product depends on the equation, in either case the contributing product includes the \( N^{th} \) order integral coefficient \( CN \). This means that the order of the \( N^{th} \) order integral coefficient \( CN \) matches the number of basis coefficients in equation (10), but is one greater than the number of basis coefficients in equation (37). In equation (10), the \( N^{th} \) order integral coefficient \( CN \) is the integral of the \( N \) basis functions that correspond to the basis coefficients appearing in the product. The \( N^{th} \) order integral coefficient \( CN \) in equation (37) is similar except that the basis function \( b_k \) corresponding to the basis coefficient \( t_k \) being determined is also included. Thus, while the two equations represent different problems, the efficient solution to both lies in determining the \( N^{th} \) order integral coefficient \( CN \). In both cases the principles described above with reference to FIGS. 1-15 are applied.

Returning to the method 2700, in block 2702 each function \( F_i(v) \) whose product is to be represented in the wavelet domain is projected into the wavelet domain. In the example embodiment, the plurality of functions includes the functions \( F_i(v) \) of the set of fixed functions \( \{F_1(v), \ldots, F_{N-1}(v)\} \). Projecting the functions \( F_i(v) \) into the wavelet domain comprises, for example, performing the wavelet transform on each function, as described above with reference to block 1602 of FIG. 16. Each function \( F_i(v) \) of the set is then represented as the series of basis coefficients \( f_{ih} \) scaling basis functions \( b_{ih} \).

In block 2704, the basis coefficients \( f_{ih} \) of each function \( F_i(v) \) are encoded in a wavelet tree. Encoding the basis coefficients \( f_{ih} \) is described above with reference to block 1604 of FIG. 16, and the wavelet trees \( W_i \) are described above with reference to FIG. 17. In some embodiments, each function \( F_i(v) \) is encoded into one wavelet tree \( W_i \), although in other embodiments each function \( F_i(v) \) can be represented using more than one wavelet tree. In the example embodiment, each function \( F_i(v) \) of the set of fixed functions \( \{F_1(v), \ldots, F_{N-1}(v)\} \) is encoded, and if one wavelet tree is used to represent each function in the set, then \( (N-1) \) wavelet trees \( W_i \) result.

In block 2706, non-linear approximation is performed on each wavelet tree \( W_i \) to discard insignificant basis coefficients \( f_{ih} \). Non-linear approximation is described above with reference to block 1606 of FIG. 16. In some embodiments, the non-linear approximation block 2706 is performed before encoding block 2704, the non-linear approximation block
In block 2708, the wavelet trees \( W_t \) are traversed on direct paths, along which an \( N^0 \)-order integral coefficient \( C^0 \) may be non-zero, to determine the basis coefficients \( t_k \) of the vector \( T \) representing the product of the functions \( F_t(v) \) in the wavelet domain. Traversing direct paths through the wavelet trees \( W_t \) is generally described above with reference to block 1608 of FIG. 16. However, unlike the embodiment of FIG. 16 in which the integral \( t \) is incremented as the wavelet trees \( W_t \) are traversed, in this embodiment the basis coefficients \( t_k \) of the vector \( T \) are determined. For example, an output wavelet tree \( W_o \) may be encoded as the input wavelet trees \( W_t \) are traversed, the output wavelet tree \( W_o \) having nodes \( w_0 \) into which the corresponding basis coefficients \( t_k \) of the vector \( T \) are encoded. Therefore, in this embodiment, traversing direct paths comprises synchronously traversing a plurality of wavelet trees \( W_t \), the wavelet trees \( W_t \) storing the basis coefficients of the functions \( [F_1(v), ..., F_{N-1}(v)] \) whose product is being represented in the wavelet domain, simultaneously processing a set of nodes \( [w_1, ..., w_{N-1}] \) that includes one node \( w_t \) from each wavelet tree \( W_t \) incrementing affected basis coefficients \( t_k \) and encoding the basis coefficients \( t_k \) in output nodes \( w_0 \) of an output wavelet tree \( W_o \). For example, \( (N-1) \) wavelet trees \( W_t \) may be synchronously traversed in embodiments in which each function \( F_t(v), ..., F_{N-1}(v) \) is represented by exactly one wavelet tree \( W_t \).

As described above, the term “traversing direct paths through the wavelet trees \( W_t \)” denotes that the nodes \( w_t \) that are synchronously processed correspond to nodes in the basis function tree that lie on a single direct path through the basis function tree, as described above with reference to FIG. 7. Traversing the wavelet trees \( W_t \) only on the direct paths employs the principles described above: the \( N^0 \)-order integral coefficient \( C^0 \) of basis functions may be nonzero if each of the basis functions lie on a single direct path through the basis function tree; however, the \( N^0 \)-order integral coefficient \( C^0 \) is zero if the basis functions do not lie on a single direct path through the basis function tree. Recall that each basis coefficient \( t_k \) is the sum of a series of contributing products, as shown in equation (37), each contributing product including one basis coefficient from each function \( F_1(v), ..., F_{N-1}(v) \) and one \( N^0 \)-order integral coefficient \( C^0 \), which is the integral of the corresponding basis functions and the basis function \( b_k \) whose basis coefficient \( t_k \) is being determined. Confining the traversal to direct paths through the wavelet trees \( W_t \) while avoiding the indirect paths enables accumulating contributing products that could contribute to the basis coefficient \( t_k \) while avoiding those that necessarily do not contribute to the basis coefficient \( t_k \). As a result, traversing the wavelet trees \( W_t \) on direct paths determines the basis coefficients \( t_k \) of a vector \( T \) with the same accuracy but with fewer computations than other systems and methods.

FIG. 28 is a block diagram illustrating an embodiment of a method 2800 for traversing direct paths through a plurality of wavelet trees \( W_t \) to determine the basis coefficients \( t_k \) of a vector \( T \) representing the product of the functions \( F_t(v) \) in the wavelet domain, which can be employed in block 2708 of FIG. 27. In block 2802, a basis coefficient in a root of an output wavelet tree \( W_o \) is initially incremented with contributions of basis coefficients in roots of input wavelet trees \( W_t \). The contribution to the root of the output wavelet tree \( W_o \) is determined by multiplying together the mother scaling coefficient stored in the root of each input wavelet tree \( W_t \), because in such case the \( N^0 \)-order integral coefficient \( C^0 \) is one. In block 2804, a set of nodes \( [w_1, ..., w_{N-1}] \) of the input wavelet trees \( W_t \) are synchronously processed. The set of nodes \( [w_1, ..., w_{N-1}] \) includes one node \( w_i \) from each of the input wavelet trees \( W_t \). No two nodes \( w_i \) in the set correspond to supports \( j, k, l \) that are completely disjoint from each other, because the input wavelet trees \( W_t \) are traversed on direct paths. In block 2806, contributions of the set of nodes \( [w_1, ..., w_{N-1}] \) to the basis coefficients \( t_k \) of output nodes \( w_0 \) of the output wavelet tree \( W_o \) are determined. In some embodiments, the contribution of the set of nodes \( [w_1, ..., w_{N-1}] \) is determined by separating the nodes into a group of null nodes and a group of non-null nodes, determining a contribution of the null nodes, and determining a contribution of non-null nodes. The contribution of null nodes is limited to the contribution of their parent nodes, which is not ignored even though the nodes themselves are null. The contribution of the non-null nodes is determined by iterating through the set of nodes to build a table that accumulates the magnitude of the contribution of the non-null nodes. In block 2808, basis coefficients \( t_k \) of output nodes \( w_o \) are incremented with contributions of the set of nodes \( [w_1, ..., w_{N-1}] \). In some embodiments, both the basis coefficients \( t_k \) of both the output node \( w_o \) and the parent of the output node \( w_o \) are updated. In block 2810, a set of child nodes is synchronously processed. The set of child nodes includes one node \( w_i \) from each of the input wavelet trees \( W_t \), the one node \( w_i \) being the immediate child of the node most recently processed. Because the child nodes immediately depend from the most recently processed nodes, the input wavelet trees \( W_t \) are traversed along direct paths. The contribution of the set of child nodes may then be determined in the manner described above, and the basis coefficients \( t_k \) of the output nodes \( w_o \) may be incremented, and the process may be repeated for subsequent sets of child nodes until the last set of nodes is reached.

In at least some embodiments, traversing direct paths through the wavelet trees \( W_t \) in block 2708 of FIG. 27 comprises applying a tree-traversal algorithm to the wavelet trees. FIG. 29 illustrates in pseudo code 2900 an example tree-traversal algorithm 2902 for determining the basis coefficients of a vector representing the product of a plurality of functions in the wavelet domain, and embodiments of component routines such as a routine 2904, a routine 2906, a routine 2908, a routine 2910, and a routine 2912. The algorithm 2902 is defined as FunctionProduct in line 1, accepting as input a set of wavelet trees \( (W_0, ..., W_{N-1}) \). The wavelet trees \( (W_0, ..., W_{N-1}) \) include one wavelet tree for each function in the set of fixed functions \( [F_1(v), ..., F_{N-1}(v)] \) whose product is being represented in the wavelet domain. The wavelet tree \( W_0 \) is an output wavelet tree into which basis coefficients determined by the algorithm 2902 are encoded, the basis coefficients being the basis coefficients \( t_k \) of the vector \( T \) representing the product of a plurality of functions in the wavelet domain.

In line 2, the product of the mother scaling coefficients of each of the fixed functions \( [F_1(v), ..., F_{N-1}(v)] \) is encoded in the variable \( W_0.de \), which is determined by taking the product of the variables \( W.t.de \) for wavelet trees \( W_t \) for \( i \) from 1 to \( (N-1) \). In line 3, the routine getCoefficients 2904 is called. The routine 2904 is configured to simultaneously process a set of nodes \( (W_1, ..., W_{N-1}) \) that includes one node \( w_i \) from each wavelet tree \( W_i \) and to encode the result of processing the nodes in the corresponding node \( w_o \) of the output wavelet tree \( W_o \). For its initial call, the routine 2904 processes the set of mother nodes \( (W_0, ..., W_{N-1}, node) \) that include the mother wavelet coefficients. The routine 2904 then iteratively calls itself to process sets of nodes corresponding to higher scales and located at lower traversal depths, traversing the set of wavelet trees \( (W_1, ..., W_{N-1}) \).
along the direct paths to determine and encode the basis coefficients in the output nodes \( w_0 \). It is worth noting that in a single call to the algorithm 2902, all of the basis coefficients of the vector are determined and encoded in the output wavelet tree \( W_0 \).

The routine getCoefficients 2904 is defined in line 1 to accept as input the variable cum, and a set of nodes \( (w_0, \ldots, w_{n-1}) \). When the routine is initially called in line 3 of algorithm 2902, the value one is passed into the routine for the variable cum. Note that the variable cum is described above with reference to the pseudo code 1900 and 2000, and therefore a complete discussion is omitted here. In line 2, the nodes of the set of nodes \( (w_1, \ldots, w_{n-1}) \) are reorganized into a two sets of nodes: one being a set of non-null nodes \( (w_1, \ldots, w_k) \) and the other being a set of null nodes \( (w_{k+1}, \ldots, w_{n-1}) \). Up to this point, the subscript \( i \) has denoted that a specific wavelet tree \( W_i \) corresponds to a specific function \( F_i(v) \), or alternatively that a specific node \( w_i \) is a node of a specific wavelet tree \( W_i \). For the remainder of the discussion of the algorithm 2902, the subscript \( i \) merely indicates whether the node \( w_i \) is null or non-null, with non-null nodes having subscripts \( w \) to \( w_i \) and null nodes having subscripts \( w_{k+1} \) to \( w_{n-1} \). It is likely that some of the nodes \( w_0 \) are null in embodiments in which non-linear approximation is performed.

In line 3, the routine 2904 returns all of the nodes in the set \( (w_1, \ldots, w_{n-1}) \) are null. This is because if all of the nodes are null, then all of the nodes of higher scales at lower traversal depths will also be null as a result of the non-linear approximation. In line 4, the variable \( w_0 \) is updated for each node in the set \( (w_1, \ldots, w_{n-1}) \). The variable parentsum is described above with reference to the pseudo code 1900 and 2000, and therefore a complete discussion is omitted here. In line 5, the variable cum is updated for the set of null nodes \( (w_{k+1}, \ldots, w_{n-1}) \). In line 6, the routine updateParents 2906 is called, as described in detail below. In lines 7-9, the three wavelet coefficients \( \psi[0], \psi[1], \) and \( \psi[2] \) of the output node \( w_0 \) are updated with the value of cum multiplied by the return of the routine getW(i) 2912. In lines 10-11, the routine 2904 then iteratively calls itself to four times to independently process the next four sets of nodes, which are the immediate child nodes of the non-null nodes. In other words, the routine 2904 calls itself for the nodes \( (w_1, ch[i], \ldots, w_k, ch[i]) \) in a loop, where \( ch[i] \) is the pointer to the child node and \( i \) is an integer from 0 to 3.

The routine updateParents 2906 is defined in line 1 to accept as inputs the output node \( w_0 \) corresponding to the set of nodes \( (w_1, \ldots, w_{n-1}) \) currently being processed, and the variable cum multiplied by the return of getProductIntegral 2908 for the non-null nodes \( (w_1, \ldots, w_k) \) of the set currently being processed. In line 2, the output node \( w_0 \) of the output wavelet tree \( W_0 \) is updated with a variable val, which is the product of the variable cum and the return of the routine getProductIntegral 2908 described below. In other words, output node \( w_0 \) is updated with the contributions of the non-null nodes \( (w_1, \ldots, w_k) \) in the set of nodes currently being processed, and the contributions of the parents of the null nodes \( (w_{k+1}, \ldots, w_{n-1}) \) in the set currently being processed. In lines 5-7, the three basis coefficients \( t_k \) in the parent of the output node \( w_0 \) is also updated with the product of the variable val, a sign, and the magnitude of a parent basis function multiplied by an intermediate child basis function.

The routine getProductIntegral 2908 is the exact same as the routine getProductIntegral 2006 described above. Further, the routines that it calls, including the routine getT(i) 2910 and the routine getW(i) 2912, are the exact same as the routines getP(i) 2908 and getW(i) 2010 described above. Therefore, the reader is referred to the prior discussion. Note that the output of the algorithm 2902 is the wavelet tree \( W_0 \), and the nodes \( w_0 \) of the wavelet tree \( W_0 \) include the basis coefficients \( t_k \) of the vector \( T \) that represents the product of the plurality of functions \( (F_1(v), \ldots, F_{n-1}(v)) \) in the wavelet domain.

The computational complexity of the algorithm 2902 is the order of \( O(mn) \), where \( m \) is the number of basis coefficients retained after the non-linear approximation and \( N \) is the number of functions \( F_i(v) \) whose product is being represented in the wavelet domain. Note that the algorithm 2902 determines all of the basis coefficients in a single call and need not be applied iteratively.

FIG. 30 is a block diagram illustrating a system 3000 for determining the wavelet domain representation of the product of a plurality of functions. The system 3000 includes logic 3002 configured to factor a plurality of functions into a set of fixed functions and one varying function, logic 3004 configured to determine a first vector that represents the product of the fixed functions in the wavelet domain, logic 3006 configured to determine a second vector that represents the one varying function in the wavelet domain, and logic configured to determine an inner product of the first vector and the second vector.

As shown, the logic 3004 configured to determine a first vector that represents the product of the fixed functions in the wavelet domain may include logic 3012 configured to project the functions of the set of fixed functions into the wavelet domain, logic 3014 configured to encode basis coefficients of each fixed function in a wavelet tree, logic 3016 configured to perform non-linear approximation on each fixed function, and logic 3018 configured to determine basis coefficients of a vector by traversing direct paths through the wavelet trees.

The computational complexity of the algorithm 2902 is the order of \( O(mn) \), where \( m \) is the number of basis coefficients retained after the non-linear approximation and \( N \) is the number of functions \( F_i(v) \) whose product is being represented in the wavelet domain. Note that the algorithm 2902 determines all of the basis coefficients in a single call and need not be applied iteratively.

FIG. 30 is a block diagram illustrating a system 3000 for determining the wavelet domain representation of the product of a plurality of functions. The system 3000 includes logic 3002 configured to factor a plurality of functions into a set of fixed functions and one varying function, logic 3004 configured to determine a first vector that represents the product of the fixed functions in the wavelet domain, logic 3006 configured to determine a second vector that represents the one varying function in the wavelet domain, and logic configured to determine an inner product of the first vector and the second vector.

As shown, the logic 3004 configured to determine a first vector that represents the product of the fixed functions in the wavelet domain may include logic 3012 configured to project the functions of the set of fixed functions into the wavelet domain, logic 3014 configured to encode basis coefficients of each fixed function in a wavelet tree, logic 3016 configured to perform non-linear approximation on each fixed function, and logic 3018 configured to determine basis coefficients of a vector by traversing direct paths through the wavelet trees.

Such systems and methods are applied to determine the radiance \( B \) of a point \( x \) in a scene using a light transport model of the lighting and objects in a scene, with reference to FIGS. 23-25. Also described above are systems and methods for approximating the integral of the product of a plurality of functions by fixing at least some of the functions whose product is being integrated, with reference to FIGS. 26-30. Such systems and method for approximating the integral are employed to approximate the radiance \( B \) of the point \( x \) in the scene, with reference to FIGS. 31-32 below.

FIG. 31 is a block diagram illustrating an embodiment of a method 3100 for rendering a graphical scene by approximating the radiance \( B \) of a point \( x \) in the scene using a light transport model of the scene. The scene is a scene such as the scene 2300 described above with reference to FIG. 23, and the light transport model of the scene is represented, for example, in equation (28), which is repeated below:

\[
B = \int L(\phi) \rho_{\phi} O_{\phi}(\psi) d\phi
\]  

(28)

In block 3102, a plurality of functions \( F_\psi(\phi) \) that contribute to the light transport model of the scene are determined. For example, the functions \( F_\psi(\phi) \) can include the distant environment lighting function \( L(\phi) \), the BRDF \( \rho_{\phi} \), the local visibility function \( O_{\phi}(\psi) \), and one or more dynamic occlusion functions \( O_{\psi}(\phi) \). Determining the plurality of functions \( F_\psi(\phi) \) is described above with reference to block 2402 of FIG. 24. In some embodiments, determining the functions \( F_\psi(\phi) \) com-
method 2800 described above with reference to FIG. 28 or applying the tree-traversal algorithm 2902, described above with reference to FIG. 29. In such case, the output wavelet tree \( W_c \) of the tree-traversal algorithm 2902 stores the basis coefficients \( (t_1, \ldots, t_n) \) of the first radiance transfer vector \( T \).

In embodiments in which the functions \( F_s(\phi) \) are cubemap functions 2310, the first radiance transfer vector \( T \) corresponds to one face 2306 of the cubemap \( S \). Therefore, determining the first radiance transfer vector \( T \) comprises determining a set of first radiance transfer vectors \( [T_1, \ldots, T_n] \) with respect to each face of the cubemap \( S \).

In some embodiments, determining the first radiance transfer vector \( T \) comprises pre-computing the first radiance transfer vector \( T \) prior to run-time, although in other embodiments the first radiance transfer vector \( T \) is computed interactively. Also, in some embodiments, determining the first radiance transfer vector \( T \) comprises determining a plurality of first radiance transfer vectors \( T \) corresponding to distinct sets of fixed functions \( [F_1(\phi), \ldots, F_{N_s}(\phi)] \). For example, if the lighting function \( L(\phi) \) is permitted to vary, the functions in the set of fixed functions are different than if the BRDF \( \rho(\phi) \) is permitted to vary. Especially in embodiments in which the first radiance transfer vector \( T \) is pre-computed but the one varying function \( F_s(\phi) \) is interactively selected to vary, it may be desirable to determine a plurality of first radiance transfer vectors \( T \) corresponding to distinct sets of fixed functions \( [F_1(\phi), \ldots, F_{N_s}(\phi)] \) so that regardless of the function \( F_s(\phi) \) interactively selected to vary, the corresponding first radiance transfer vector \( T \) is already pre-computed and ready for use in real-time. In embodiments in which the functions \( F_s(\phi) \) are cubemap functions 2310, determining the plurality of first radiance transfer vectors \( T \) comprises determining a plurality of sets of first radiance transfer vectors \( [T_1, \ldots, T_n] \) for each vector \( T_i \) of the set corresponding to one face 2308 of the cubemap \( S \), and each set \( [T_{1j}, \ldots, T_{nj}] \) of the plurality of sets corresponding to one of the potential sets of fixed functions \( [F_1(\phi), \ldots, F_{N_s}(\phi)] \).

In block 3108, a second radiance transfer vector \( F \) is determined. The second radiance transfer vector \( F \) represents the one varying function \( F_s(\phi) \) in the wavelet domain as a series of basis coefficients \( (f_1, \ldots, f_m) \), or alternatively, as a series of basis coefficients \( f_b \) scaling a series of basis functions \( b_{\phi} \), as shown in equation (40):

\[
F = (f_1, f_2, \ldots, f_m) \quad \text{where} \quad f_s(\phi) = \sum_{k=1}^m b_k(\phi)\]

Determining the second radiance transfer vector \( F \) is generally described above with reference to block 2606 of FIG. 26 and can comprise projecting the one varying function \( F_s(\phi) \) into the wavelet domain by performing a wavelet transform, encoding the basis coefficients \( (f_1, \ldots, f_m) \) of the second vector \( F \) in a wavelet tree \( W \), and performing non-linear approximation on the one varying function \( F_s(\phi) \).

In embodiments in which the functions \( F_s(\phi) \) are cubemap functions 2310, the second radiance transfer vector \( T \) corresponds to one face 2308 of the cubemap function. Therefore, determining the second radiance transfer vector \( T \) comprises determining a set of second radiance transfer vectors \( [F_1(\phi), \ldots, F_n(\phi)] \), each vector \( F_i \) of the set corresponding to one face 2308 of the cubemap function.

Therefore, determining the first radiance transfer vector \( T \) comprises determining a series of basis coefficients \( (t_1, \ldots, t_m) \). Using the method 2700 described above with reference to FIG. 27, determining the series of basis coefficients \( (t_1, \ldots, t_m) \) generally comprises projecting each function \( F_s(\phi) \) in the set of fixed functions \( [F_1(\phi), \ldots, F_{N_s}(\phi)] \) into the wavelet domain (block 2702), encoding each function \( F_s(\phi) \) in a wavelet tree \( W \) (block 2704), subjecting each function \( F_s(\phi) \) to non-linear approximation (block 2706), and traversing the wavelet trees \( W \) along direct paths to determine the basis coefficients of the vector (block 2708). Traversing the wavelet trees \( W \) along direct paths may comprise performing the
In at least some embodiments, determining the second radiance transfer vector \( F \) comprises iteratively determining the second radiance transfer vector \( F \) in response to changing conditions in the scene \( 2300 \). While the first radiance transfer vector \( T \) can be fixed because the functions \([T_{1}(\phi), \ldots, T_{N_{c}}(\phi)]\) represented by the vector \( T \) do not change, the same is not true for the second radiance transfer vector \( F \), which represents the one varying function \( F_\alpha(\phi) \). Therefore, it may be desirable to determine the first radiance transfer vector \( T \) once, while determining the second radiance transfer vector \( F \) iteratively and interactively, in response to changing conditions in the scene \( 2300 \).

In block \( 3110 \), an inner product of the first radiance transfer vector \( T \) and second radiance transfer vector \( F \) is determined. Taking the inner product of the first radiance transfer vector \( T \) and second radiance transfer vector \( F \) approximates the radiance \( B \) of the point \( x \), as shown by re-writing equation (38) in terms of equation (39) and equation (40):

\[
\begin{align*}
B &= \int \left[ L_{\alpha}(\phi) \cdot \frac{\partial}{\partial \phi} \mathcal{D}_{\alpha}(\phi) \mathcal{D}_{\alpha}(\phi) d\phi \right]
\end{align*}
\]

In equation (41), the radiance \( B \) is an approximation and does not vary. However, because the first radiance transfer vector \( T \) is fixed, approximating the radiance \( B \) using the method \( 3100 \) is relatively less computationally complex, and therefore relatively faster, than other methods. For example, in embodiments in which the first radiance transfer vector \( T \) is pre-computed and the second radiance transfer vector \( F \) is determined interactively, the method \( 3100 \) can be employed to approximate the radiance \( B \) interactively and/or in real-time by taking the inner product of the two vectors interactively and/or in real-time.

In embodiments in which the functions \( F_\alpha(\phi) \) are cubemap functions \( 2310 \), the inner product of the first radiance transfer vector \( T \) and second radiance transfer vector \( F \) corresponds to one face \( 2308 \) of the cubemap \( S \). Therefore, determining the inner product comprises determining inner products of a set of first radiance transfer vectors \([T_{1}, \ldots, T_{N_{c}}] \) and a set of second radiance transfer vectors \([F_{1}, \ldots, F_{\alpha}] \) and summing the inner products together, each inner product corresponding to one face \( 2308 \) of the cubemap \( S \) and the sum of the inner products representing the radiance \( B \) over the entire cubemap \( S \).

In block \( 3112 \), the scene is rendered, as described above with reference to block \( 2412 \) of FIG. 24. The radiance \( B \) for the point \( x \) determined in block \( 3110 \) is used by the graphics rendering application to set the color of the point \( x \). In at least some embodiments, the radiance \( B \) of the point \( x \) represents one color value for the point, but to render the scene the graphics rendering application requires three color values, one for each of three independent color channels. Therefore, in such embodiments the above blocks are iterated to independently process each of the three independent color channels, such that three color values are available to the graphics rendering application for the point \( x \). The above steps are also iterated for each visible point \( x \) before the final scene is rendered, because a plurality of points \( x \) are combined to produce objects, and a plurality of objects are combined to produce the rendered scene.

The method \( 3100 \) can be employed to generate all-frequency shadows in real-time. In embodiments in which the first radiance transfer vector \( T \) is pre-computed but the second radiance transfer vector \( F \) is determined interactively, the method \( 3100 \) can be considered a method of just-in-time radiance transfer (JRT), unlike prior pre-computed radiance transfer (PRT) methods, such as those disclosed by Sloan et al. in “Precomputed radiance transfer for real-time rendering in dynamic-low-frequency lighting environments,” ACM Transactions on Graphics (SIGGRAPH 02) 21, 3, 527-536, which is incorporated by reference herein in its entirety.

Allowing only one function to vary is reasonable in some cases, such as in lighting design systems, in which the designer typically adjusts only one variable at a time. For example, the designer may experiment with the lighting by varying the lighting function \( L(\phi) \) without varying the direction of view \( \theta \), or the objects \( 2302, 2306 \) in the scene \( 2300 \). The designer may also render the scene from different directions of view \( \theta \) while maintaining the current lighting \( L(\phi) \) and objects \( 2302, 2306 \) in the scene \( 2300 \). In other cases, the designer may change the location of one object \( 2302 \) or \( 2306 \) in the scene \( 2300 \) while holding the lighting function \( L(\phi) \) and direction of view \( \theta \) constant. As long as only one varying function \( F_\alpha(\phi) \) is selected, the method \( 3100 \) can be employed to generate all-frequency shadows in real-time.

Because the method \( 3100 \) employs pre-computed data on individual objects \( 2302, 2306 \) in the scene \( 2300 \) instead of pre-animated models of the scene as a whole, the objects in the scene can be manipulated interactively, such as by cloning, scaling, and/or translating the objects. Even in such cases of interactive manipulation, the method \( 3100 \) can be employed to render the objects with all-frequency shadows in real-time. Glossy materials are also supported, such that dynamic high-glossy objects can be rendered in real-time with realistic, all-frequency shadows.

The method \( 3100 \) supports interactively changing the function selected as the one varying function \( F_\alpha(\phi) \). In embodiments in which a plurality of first radiance transfer vectors \( T \) are pre-computed in block \( 3106 \) for different sets of fixed functions \([F_{1}(\phi), \ldots, F_{\alpha}(\phi)]\), the one varying function \( F_\alpha(\phi) \) can be changed in real-time without repeating block \( 3106 \). For example, if the distant environment lighting function \( L(\phi) \) is selected as the one varying function \( F_\alpha(\phi) \), rendering the scene for a variety of lighting conditions in real-time reduces to iteratively determining the second radiance transfer vector \( F \) of the lighting function in real-time and taking the inner product of the two radiance transfer vectors \( T \) and \( F \) in real-time, because the first radiance transfer vector \( T \) for the set of fixed functions \([\rho(\phi), O_{1}(\phi), O_{2}(\phi), \ldots, O_{N_{c}}(\phi)]\) was determined in advance. If the one varying function \( F_\alpha(\phi) \) is interactively changed from the distant environment lighting function \( L(\phi) \) to the first dynamic occlusion function \( O_{1}(\phi) \), rendering the scene for the moving object in real-time reduces to iteratively determining the second radiance transfer vector \( F \) of the dynamic occlusion function in real-time and taking the inner product of the two radiance transfer vectors \( T \) and \( F \) in real-time, because the first radiance transfer vector \( T \) for the set of fixed functions \([\rho(\phi), O_{1}(\phi), O_{2}(\phi), \ldots, O_{N_{c}}(\phi)]\) was pre-computed in advance.

FIG. 32 is a block diagram illustrating an embodiment of a system \( 3200 \) for rendering a graphical scene by approximating the radiance of the point \( x \) in a scene using a light transport model of the scene. The system \( 3200 \) includes logic \( 3202 \) configured to determine a plurality of functions that contribute to a light transport model of a scene, logic \( 3204 \) configured to factor the plurality of functions into a set of fixed functions and one varying function, logic \( 3206 \) configured to
determine a first radiance transfer vector for the set of fixed functions, logic 3208 configured to determine a second radiance transfer vector for the one varying function, logic 3210 configured to determine the inner product of the first and second radiance transfer vectors, and logic 3212 configured to render the scene.

While particular embodiments of systems and methods for determining the integral of the product of a plurality of functions, and for determining the product of a plurality of functions, have been disclosed in detail in the foregoing description and figures for purposes of example, those skilled in the art will understand that variations and modifications may be made without departing from the scope of the disclosure. All such variations and modifications are intended to be included within the scope of the present disclosure, as protected by the following claims.

At least the following is claimed:

1. A method of rendering a graphical scene using a computer, the method comprising:
   determining with the computer a plurality of functions that contribute to a light transport model of a scene; factoring with the computer the plurality of functions into a set of fixed functions and one varying function, the varying function being interactively determinable and therefore permitted to vary during the rendering, and the fixed functions being computed in advance of the rendering so as to be invariable during the rendering; determining with the computer a first radiance transfer vector that represents a product of the fixed functions in the wavelet domain; determining with the computer a second radiance transfer vector that represents the one varying function in the wavelet domain; determining with the computer an inner product of the first and second radiance transfer vectors to approximate a radiance of a point x in the scene; and rendering with the computer the scene.

2. The method of claim 1, wherein determining the plurality of functions that contribute to the light transport model of the scene comprises determining a distant environment lighting function, a bi-directional reflectance distribution (BRDF), a local visibility function, and one or more dynamic occlusion functions.

3. The method of claim 2, wherein determining the plurality of functions that contribute to the light transport model of the scene further comprises collecting sampled data about the scene, and processing the sampled data to create a plurality of cubemap functions, one cubemap function being created for each function that contributes to the light transport model of the scene.

4. The method of claim 1, wherein determining the plurality of functions comprises pre-computing a cubemap function for each of the fixed functions, and interactively determining a cubemap function for the one varying function.

5. The method of claim 1, wherein determining a first radiance transfer vector that represents the product of the fixed functions in the wavelet domain comprises:
   projecting each function of the set of fixed functions into the wavelet domain;
   encoding each function of the set of fixed functions into a wavelet tree;
   traversing direct paths through the wavelet trees, along which direct paths an integral coefficient may be non-zero, to determine the basis coefficients of the first radiance transfer vector.

6. The method of claim 5, wherein projecting each function of the set of fixed functions into the wavelet domain comprises performing a two-dimensional nonstandard Haar wavelet transform on each fixed function.

7. The method of claim 5, wherein encoding the basis coefficients of the function comprises encoding the basis coefficients of each function into six wavelet trees, each function being represented as a cubemap function and each face of each cubemap being encoded into a separate wavelet tree.

8. The method of claim 5, wherein traversing direct paths through the wavelet trees to determine basis coefficients of the first radiance transfer vector comprises synchronously processing a set of nodes that includes one node from each wavelet tree, determining a contribution of the set of nodes to basis coefficients of the first radiance transfer vector, incrementing the basis coefficients, encoding the basis coefficients in an output wavelet tree, and synchronously processing a child set of nodes.

9. The method of claim 5, wherein traversing the wavelet trees comprises:
   separately traversing six sets of wavelet trees, each set corresponding to one face of a cubemap and each wavelet tree of the set including basis coefficients that represent one face of one cubemap function, such that traversing each one set of wavelet trees determines the first radiance transfer vector for one face of the cubemap and traversing each of the six sets of wavelet trees separately determines a set of six first radiance transfer vectors that represent the entire cubemap.

10. The method of claim 5, further comprising performing non-linear approximation on each fixed function to discard insignificant basis coefficients.

11. The method of claim 1, wherein determining the second radiance transfer vector comprises projecting the one varying function into the wavelet domain, encoding the basis coefficients of the one varying function into a wavelet tree, and performing non-linear approximation on the one varying function to discard insignificant basis coefficients.

12. The method of claim 11, wherein projecting the one varying function into the wavelet domain comprises performing a two-dimensional nonstandard Haar wavelet transform on the one varying function.

13. The method of claim 12, wherein encoding the basis coefficients of the one varying function comprises encoding the basis coefficients of the one varying function into six wavelet trees, the one varying function being represented as a cubemap function and each face of the cubemap being encoded into a separate wavelet tree.

14. The method of claim 1, wherein:
   each function is represented as a cubemap function;
   determining the first radiance transfer vector comprises determining a set of first radiance transfer vectors, each first radiance transfer vector of the set corresponding to one face of a cubemap that represents the product of the fixed functions in the wavelet domain;
   determining a second radiance transfer vector comprises determining a set of second radiance transfer vectors, each second radiance transfer vector corresponding to one face of the cubemap function that represents the one varying function; and
   determining the inner product of the first and second radiance transfer vectors comprises determining inner products of the two sets of first and second radiance transfer vectors, and summing the inner products together to determine the radiance of the point x.
15. The method of claim 1, wherein the first radiance transfer vector is pre-computed in advance of run-time, the second radiance transfer vector is determined interactively, and the inner product of the first and second radiance transfer vectors is determined interactively.

16. The method of claim 13, wherein the second radiance transfer vector and the inner product are determined in real time.

17. The method of claim 1, wherein the first radiance transfer is pre-computed in advance of run-time, the second radiance transfer vector is iteratively determined in real time in response to changing scene conditions, and the inner product is iteratively determined in real time.

18. The method of claim 1, wherein:
   factoring the functions into the set of fixed functions and the one varying function comprises interactively selecting the one varying function; and
   determining the first radiance transfer vector comprises pre-computed a plurality of first-radiance transfer vectors in advance of run-time, such that the first radiance transfer vector for the set of fixed functions has already been pre-computed regardless of which function is interactively selected as the one varying function.

19. The method of claim 1, further comprising iteratively determining three color values for each visible point in the scene before rendering the scene.
In the Specification

Column 1, line 14 through 20 should read,

This invention was made with Government support under Agency contract 0312724 awarded by the National Science Foundation. The Government has certain rights in this invention.