Efficiently Representing the Radiosity Kernel through Learning *

Christian-A. Bohn **

Dept. Visualization and Media Systems Design
German National Research Center for Information Technology
Schloss Birlinghoven, D-53754 Sankt Augustin, FR Germany

Abstract: A novel method for approximating the radiosity kernel by a discrete set of basis functions is presented. The algorithm is characterized by selecting samples from the geometry definition and iteratively creates a functional model instantiated by a set of Gaussian basis functions. These are supported over the whole environment and thus, surfaces are not considered separately. Together with the implicit clustering algorithm provided by the applied learning scheme, the algorithm accounts ideally for coherence in the global kernel function.
On one hand, this leads to a very sparse representation of the kernel. On the other hand, by avoiding the creation of initial basis functions for separate pairs of surfaces, the method is capable of calculating even huge geometries to a desired accuracy with a proportional amount of computing resources.
Recent results from the field of artificial neural networks (the *Growing Cell Structures*) are extended for the presented learning algorithm. This work is done in Flatland, but there are no methodical constraints which bound the application to two dimensions.

Keywords: global illumination, radiosity, clustering, visibility, finite elements, neural networks.

1 Introduction

Realistic imaging three-dimensional virtual scenes is accomplished by *global illumination* methods. The *rendering equation* [1] offers a comprehensive description of the aspects of light transport. Since it seems to be too complex for direct evaluation many methods use a simplified view of the behavior of light transfer. *Radiosity* approaches commonly reduce the light flow to the ideal diffuse phenomena formulated by the *radiosity equation*.

$$B(\mathbf{y}) = E(\mathbf{y}) + \rho(\mathbf{y}) \int_{\mathcal{S}} G(\mathbf{x},\mathbf{y}) V(\mathbf{x},\mathbf{y}) B(\mathbf{x}) d\mathbf{x} \tag{1}$$

$\mathbf{x}, \mathbf{y} \in \mathbb{R}^p$ denote points in the geometry of dimensionality p. For natural geometries p equals three. We refer to p explicitly as this work is based on, but

* published in the provisional proceedings of the '7th EUROGRAPHICS Workshop on Rendering', Porto, 1996.

** e-mail: bohn@gmd.de

not restricted to two-dimensional scenes only ($p = 2$). For an easier examination Heckbert did valuable work concerning this simplification strategy [2] and called it radiosity in *Flatland*. The radiosity B located at a point $\mathbf{y}$ from the geometry $\mathcal{S} \subset \mathbb{R}^p$ is determined by the receiving radiosity from all points $\mathbf{x}$ of the environment and weighted by a reflection coefficient ρ. E denotes the self-emittance at $\mathbf{y}$.

The transfer of energy from point $\mathbf{x}$ to point $\mathbf{y}$ is weighted by the terms G and V. $G(\mathbf{x}, \mathbf{y}) = \cos\phi_x \cos\phi_y / (2\pi r)$, defines the purely geometrical relationships in Flatland. The ϕ_x and ϕ_y are the angles between the edge $\overline{\mathbf{xy}}$ of length r and the normals of emitting and receiving edges. V is a visibility function which equals one if $\mathbf{x}$ and $\mathbf{y}$ are mutually visible, otherwise zero. See [3, 4] for its physical derivation.

We define the *kernel*, k, which describes in one term the purely geometrical proportion of equation (1) together with the reflection properties.

$$k(\mathbf{x}, \mathbf{y}) = \rho(\mathbf{x}) G(\mathbf{x}, \mathbf{y}) V(\mathbf{x}, \mathbf{y}) \tag{2}$$

Solving the radiosity equation. Analytical solutions of (1) are only known for very specific simple geometries. For numerical calculation, there exist two different approaches: *Monte Carlo* methods and *finite elements methods* (FEM). This work concentrates on FEMs which transform the radiosity equation into the linear system

$$\forall i : b_i = e_i + \sum k_{ij} b_j. \tag{3}$$

The coefficients b_i, e_i, and k_{ij} are discrete coefficients for an approximation of B, E, and k from equations (1) and (2). As early radiosity approaches favored the intuitive view of discrete patches transferring energy through *formfactors*, the FEM provides a more general view in terms of n *basis functions*, N_i, weighted by coefficients b_i which send energy through transfer coefficients. The approximated radiosity $\tilde{B}$ is calculated by the accumulation of the radiosity basis functions,

$$\tilde{B}(\mathbf{x}) = \sum_{i=1}^{n} b_i N_i(\mathbf{x}).$$

The n^2 transfer coefficients k_{ij} between each pair of basis functions N_i and N_j are calculated as follows.

$$k_{ij} = \int \int k(\mathbf{x}, \mathbf{y}) N_j(\mathbf{x}) N_i(\mathbf{y}) d\mathbf{x}\, d\mathbf{y}, \;\; i, j = 1, ..., n \tag{4}$$

To solve equation (3), several standard algorithms exist which in practice iteratively 'propagate' the energy through the discrete approximation of k. Commonly, they start with an initial energy distribution, $b_i^{(0)} = \int E(\mathbf{x}) N_i(\mathbf{x}) dx$.

2 Motivation

Numerically solving the radiosity equation (1) requires a discretization of the radiosity function B and of the kernel function k to create a linear system. On one hand, basis functions should be distributed as sparse as possible to keep the amount of computing resources small. On the other hand, a certain demanded approximation accuracy requires an efficient placement of basis functions in terms of accounting for higher variations in the kernel function with a better resolution.

Coherence. We denoted the radiosity and the kernel function as p- and $2p$-dimensional functions respectively ($\mathbf{x} \in \mathbb{R}^p$). In fact, a realistic geometry definition of virtual worlds commonly contains single discrete geometrical objects. These define points on sub-spaces of $\mathbb{R}^p$, like 2D-surfaces in 3D and 1D-edges in 2D. Thus, radiosity is a $(p-1)$-dimensional function and the kernel of dimension $2(p-1)$. Without occlusion effects, the kernel is smooth in the range of a single pair of surfaces but commonly, at the transition to another pair, sharp boundaries arise in the kernel function. These boundaries do not appear if the interaction between different pairs of surfaces is 'similar'. For example, consider two nearly equal surfaces which lie close together and send energy to another which is relatively far away.

In the preceeding section, the basis functions of the kernel were defined as the tensor product basis of a pair of radiosity basis functions (equation (4)). These commonly are generated for each separate pair of surfaces by default, and thus, the coherence of the whole kernel is not accounted for. This work concentrates on the kernel basis functions directly on a level which abstracts from the geometrical default discretization by surfaces. The algorithm examines the kernel function by considering only single kernel values ('rays'). Basis functions of the kernel are built according to the distribution of rays, independently of single surfaces.

Previous work. HR [5] and WR [6] detect coherence in the kernel between pairs of surfaces. They do not account for the whole kernel. This is crucial due to the fact that large geometries need an expensive initialization phase for creating at least one basis function for each pair of surfaces.

Smits et al. [7] and Sillion [8] propose algorithms which approximate coherence in the energy transfer by spatial coherence in the geometry definition. The problems are that spacial clusters of the surfaces do not necessarily correspond to clusters in the kernel, and that possibly spatial clusters may not exist due to the chosen cluster criterion.

3 The algorithm

A function approximation model for the radiosity kernel is developed which is based on the *supervised growing cell structures* (SGCS) approach [9]. It is derived from the field of *artificial neural networks* (ANN) and characterized by the facility of *learning* a certain functionality based on presenting sets of examples ('rays') from the *goal function* (kernel). The final model serves as an approximation of the kernel function. It is instantiated by a set of *radial basis functions* (RBF) [10], defined by its expanse and its center. The centers can be seen as centers of clusters of light transfer. The RBFs serve as kernel basis function and its coefficients as the discrete approximation coefficients for the linear system (equation (3)). The learning model adaptively finds clusters in the input space, and thus, it accounts effectively for the coherence in the kernel functions.

The iterative learning scheme is based on three essential operations. New RBFs are added according to the approximation accuracy, the centers of the RBFs are organized due to the cluster properties in the kernel, and samples are selected from the geometry automatically directed by the algorithm itself to avoid redundant examination of the kernel function.

Limitations. We would like to mention that this paper shows work in progress. We present a basic framework for an alternative representation of the radiosity kernel. A discretization of the radiosity equation is shown for the two-dimensional case. The transfer of light, i.e., the solution of equation (3) will be matter of future work. Thus, coherence originating in effects from the energy transfer is only accounted for the self-emittance of the surfaces (E from equation (1)) so far. The last section gives hints in how to go on for a completely iterative and adaptive radiosity solver.

3.1 Artificial neural networks

Instead of coding a program by hand, the functionality of an artificial neural network is trained by examples. The network finds its own internal representation according to the underlying data distribution.

ANNs consist of two basic elements. First, *units* (also called *neurons*) have the capability of summing several input values and weighting them by a thresholding-function. Second, these units are connected in a graph-structure by weighted *connections* which transport values between separate units and the network's input and output. Generally, the topology and the connection weights determine the network's functionality and thus these are the parameters which are to be modified by a learning procedure.

Basically, there exist two different learning strategies, *supervised* in contrast to *unsupervised* learning. Supervised learning means to present input/output pairs to the network. The connection weights of the network are modified such that the output units approach the presented output better (*steepest descend*). In unsupervised learning only the distribution of the input data is examined and the units are modified to represent the input space by reference vectors (*clustering*). Unsupervised learning is also known as *self-organization*, *competitive learning*, *vector quantization*, or *dimensionality reduction*.

In our work, we are looking for an algorithm which combines these two principles since we have to analyze the input distribution in terms of 'similar' light transfer (clusters in the light transfer) and the kernel value for a certain cluster has to be approximated concurrently. There are few hybrid algorithms (see [11]) which are not suitable for this task, as both strategies are applied independently. To the author's knowledge, [9] is the first method which combines both strategies in a homogeneous manner.

3.2 Supervised growing cell structures

A SGCS network contains two layers. The first is instantiated by radial basis functions, the second accumulates each of the output (*activations*) of the RBFs to form the output vector (see left side of figure 1). The network realizes a function $\tilde{f} : \mathbb{R}^n \rightarrow \mathbb{R}^m$ which serves as an approximation of a goal function $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$. Samples are drawn from f, which the network is trained with. At each state of the learning process, the network generalizes the function f over its input space to a certain accuracy.

The network is trained by presenting input/output pairs $(\xi, \zeta) \in (\mathbb{R}^n \times \mathbb{R}^m)$. The unsupervised learning part is accomplished by moving the cells of the first layer according to the input ξ to find centers of clusters in the input data. Concurrently, the second layer is adapted to deliver the intended output ζ. Moving

the RBFs is accomplished by accounting for a neighborhood relation between the single cells, which is of predefined — but not fixed — dimensionality. This relation creates an additional structural information on the training data. In our case, a two-dimensional map of clusters of light transfer is created. This will finally enable the transfer of energy, since it relates the basis functions to the geometrical relationships of the environment.

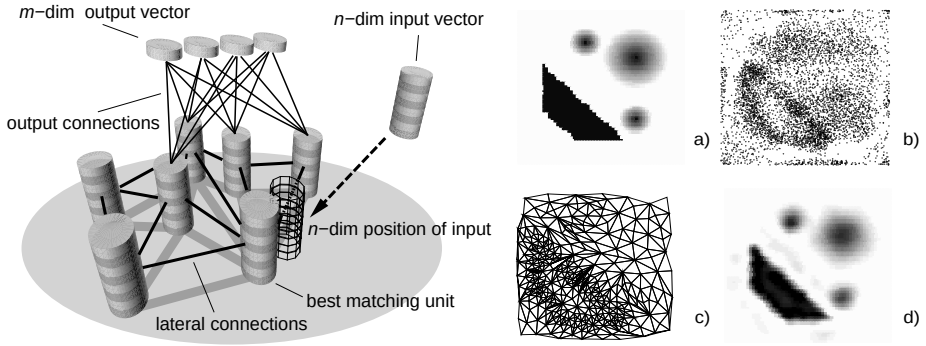

Fig. 1. On the left side, an example SGCS network is outlined. The output connections are drawn for four input cells only. On the right side, a two-dimensional example goal function (a) is approximated (d). (b) and (c) show the samples which are selected by the algorithm and the created network structure.

The initial topology of the network is a set, A, of *cells*, connected in a k-dimensional structure by lateral connections. The only basic element is a k-dimensional simplex, i.e., for $k = 2$ this is a triangle. During a self-organizing process, new cells are added to A in a way that at each time this basic structure holds.

Every cell c has attached an n-dimensional synaptic vector $\mathbf{w}_c$ which is the *position* of c in the input vector space V. A *mapping* $\phi_{\mathbf{w}} : V \to A$ is defined as

$$\begin{aligned} \phi_{\mathbf{w}} : V \to A, (\xi \in V) \mapsto (\phi_{\mathbf{w}}(\xi) \in A), \\ ||\mathbf{w}_{\phi_{\mathbf{w}}(\xi)} - \xi|| = \min_{r \in A} ||\mathbf{w}_r - \xi||, \end{aligned} \tag{5}$$

with $\mathbf{w}$ the set of all synaptic vectors $\mathbf{w}_c, c \in A$. $\phi_{\mathbf{w}}(\xi)$ is called the *winning* or the *best matching unit* (BMU) for an input ξ.

By equation (5), V is partitioned into a number of regions F_c ($c \in A$), each consisting of the locations having the common nearest synaptic vector $\mathbf{w}_c$. This is known as *Voronoi tessellation* and the regions are denoted by *Voronoi regions*. We define f_c the k-dimensional Voronoi volume of a cell and approximate it by $|F_c|$, $f_c \approx |F_c| = \overline{l_c}^l$, with $\overline{l_c}$ is the mean length of the l *edges* (the lateral connections) emanating from a cell c. The length of an edge between two cells i, j is defined as the Euclidian distance between their synaptic vectors $l_{i,j} = ||\mathbf{w}_i - \mathbf{w}_j||$.

The supervised layer of the network is defined by one additional output weight vector for each cell, $\mathbf{v}_c \in \mathbb{R}^m, c \in A$. The output of the network, $\kappa : \mathbb{R}^n \to \mathbb{R}^m$, according to an input ξ is calculated as

$$\kappa(\xi) = \sum_{c \in A} \mathbf{v}_c M_c(\xi), \quad \text{with} \quad M_c(\xi) = \exp(-\frac{||\xi - \mathbf{w}_c||^2}{\delta_c^2}). \tag{6}$$

M_c is the output of a cell c (*activation*). δ_c is assigned to f_c.

If an I/O pair is presented to the network for training, the sets $\mathbf{w}$ and $\mathbf{v}$ with $\mathbf{v} = \{\mathbf{v}_c, c \in A\}$ are modified such that $\mathbf{w}$ adapts to the input distribution by moving cells in n-space. The vectors $\mathbf{v}$ are modified to approach the intended output value ζ. We define the neighborhood N_c of a cell c as the set of directly connected cells. With the notation $X^{\text{new}} = X^{\text{old}} + \Delta X$, the adaption of the cell weights for each iteration cycle is done as follows.

$$\begin{aligned} \Delta \mathbf{w}_s &= \epsilon_b(\xi - \mathbf{w}_s), \quad \text{for the BMU } s, \\ \Delta \mathbf{w}_b &= \epsilon_n(\xi - \mathbf{w}_b), \quad \forall b \in N_s, \\ \Delta \mathbf{v}_c &= \eta(\zeta - \kappa(\xi)) \cdot M_c, \forall c \in A, \end{aligned}$$

with ϵ_b, ϵ_n, and η the learning parameters for the input weights of the BMU, its neighbors, and the output weights, respectively.

So far we described the standard SGCS approach. On the right side of figure (1) the approximation of an example function can be observed. In the following, we redefine the resource term and add the resampling scheme.

The resource term, τ, is defined in a fashion similar to the L_2 error measure. It consists of two components $\tau_{cnt,c}$ and $\tau_{err,c}$. Consider the cell s as BMU. $\tau_{err,s}$ is incremented by the error which the network compared to the actual ζ delivers, $\Delta\tau_{err,s} = \|\zeta - \emptyset\|^2$, $\tau_{cnt,s}$ is incremented by one. Concurrently for all cells $c \in A$, the terms $\Delta\tau_{cnt,c} = -\alpha \cdot \tau_{cnt,c}$ and $\Delta\tau_{err,c} = -\alpha \cdot \tau_{err,c}$ are added, which provide the weighted mean value over a certain time. The final resource value is calculated according to $\tau_c = \tau_{err,c}/\tau_{cnt,c} \cdot |F_c|$. α is a 'forgetting-parameter' in order to enable a weighted averaging over recent learning cycles.

After a certain number λ of iteration cycles the cell c with the largest τ_c is selected from A. Insertion of a new cell r is done in the middle of the longest edge originating from c. This must keep the network structure homogeneous (only simplices of dimension k). For the implementation details and the accounting for the modified Voronoi regions, see [9].

The resampling strategy. The central idea is the detection of regions in the input space which are not sufficiently represented by training samples. A high resource term is taken as criterion for the need for new samples (similar to the need for insertion of new cells). Thus, we define a relation $\sigma_\tau : (c \in A) \to \{T, F\}$, which determines, if a cell c is a *critical cell* (CC).

$$\sigma_\tau(c) = (\tau_c > \omega \cdot \overline{\tau}), \; c \in A,$$

with ω a threshold for being a high-resource cell and $\overline{\tau}$ the mean value of the resource terms of all cells from A.

For an input $\xi \in \mathbb{R}^n$, we define the overall *network activation*, $D_A : \mathbb{R}^n \rightarrow \mathbb{R}, D_A(\xi) = \sum_{c \in A} M_c(\xi)$ which is the sum of the activations of all cells of the network A, and a relation $\rho : \mathbb{R}^n \rightarrow \{T, F\}$ which defines a logical value for the fact that an input ξ lies in the range of a network A.

$$\rho_A(\xi) = (D_A(\xi) > \varphi),$$

with a threshold φ.

Further, we define the sub-network of A which consists only of critical cells as the term A_{CC}, with $A_{CC} = \{c, c \in A | \sigma_\tau(c)\}$.

These definitions enable two essential predicates. First, an input ξ lies within a critical region of the network A if $\rho_{A_{CC}}(\xi)$ is true. Second, an input ξ lies outside of the range of the network if the term $\rho_A(\xi)$ returns false.

Thus, a test sample ξ fed into the network exposes a place in the input space which is not sufficiently approximated if the relation $\theta : \mathbb{R}^n \rightarrow \{T, F\}, \theta(\xi) = \rho_{A_{CC}}(\xi) \wedge \neg \rho_A(\xi)$ returns the value *true*. In other words, $\theta(\xi)$ indicates that ξ falls into a region of the network which is not well represented because it lies either in a critical zone or at an exterior zone of the network. For the selection of new samples, the input space is scanned by evaluating θ. New samples are created and the kernel value calculated at locations where θ returns true.

The algorithm successively generates sets of sample I/O pairs, $\mathcal{S}_j \in \mathbb{R}^n \times \mathbb{R}^m : \{(\xi_i, \zeta_i)\} | \ \zeta_i := f(\xi_i), i = 1, ..., |S_j|, j = \{0, ..., q-1\}$. The value q is the actual number of objects in the whole set of all samples $\mathcal{S} = \{\mathcal{S}_j\}, j = 0, ..., p-1$. The initial set $\mathcal{S}_0$ is taken from V completely randomly. All further sets are calculated as described. The samples for training are selected from all sets from $\mathcal{S}$ with equal probability.

A new sample set is generated according to the *uncertainty principle*, i.e., if the ratio of the number of cells and the number of training samples, $\psi = |A|/|\mathcal{S}|$, with $|\mathcal{S}| = \sum_{j=0}^{p-1} |\mathcal{S}_j|$ rises above a certain threshold $\psi_{A,\mathcal{S}}$. The whole algorithm is outlined in figure 2 on the left side.

Finally, we list some experimentally determined parameters which showed a robust behavior of the algorithm: $\epsilon_b = 0.01$, $\epsilon_n = 0.001$, $\eta = 0.1$, $\alpha = 0.05$, $\lambda = 300$, $\omega = 0.6$, $\varphi = 1$, $\psi_{A,\mathcal{S}} = 0.05$.

3.3 Approximating the kernel

A one-dimensional parameterization of the geometry is chosen as described in [2] (see figure (3)). By the parameters s and t we select two points on the edges of the geometry. For training the SGCS with the kernel function, the naive way would be to feed the parameters s and t, and the kernel value directly into the network. Since the parametrization discards the geometrical relationship between separate surfaces, clusters could be created at locations which are not suitable to be represented by one reference interaction for the transfer of light. The parameters for the SGCS must somehow expose the geometrical properties of the kernel function.

Thus, we introduce the function $\gamma : \mathbb{R}^2 \rightarrow \mathbb{R}^n$ (generally, $\gamma : \mathbb{R}^{2(p-1)} \rightarrow \mathbb{R}^n$) which expands the parameters s and t to tuples (re-parametrization) which should implicitly contain the information to be examined. In this work, we set $n = 4$ and define γ such that it is composed of the four coordinates of the two points in Flatland defined by s and t.

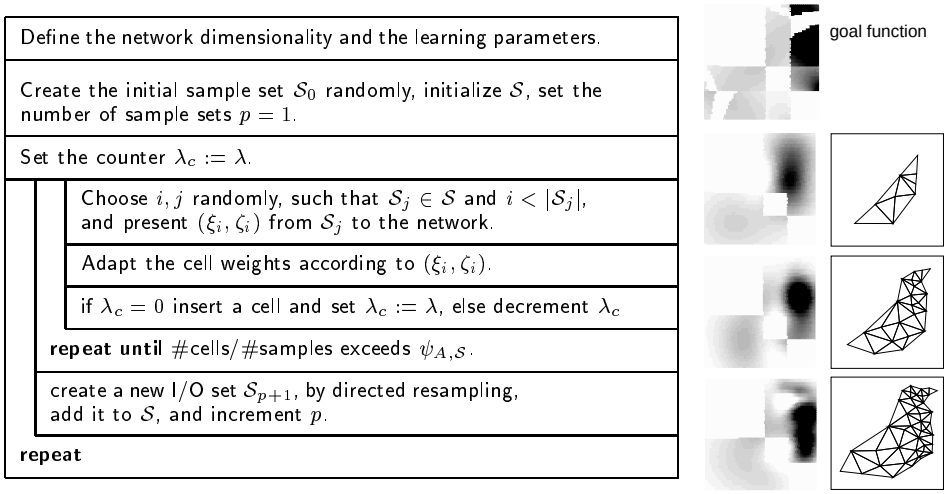

Fig. 2. The extended basic algorithm for the SGCS network, and a running example which shows growing and the adaptive function approximation of the goal function.

The output which the network should learn, $\zeta \in \mathbb{R}$, is set to the kernel function value multiplied by the emitting energy of the surface, the 'first shot' of energy (according to the initialization of the linear system (3)). We set the emitting energies of surfaces which are not defined as emitters to a small value to have a situation like after a few iteration cycles of the simulation of the energy flow.

The SGCS is fed with the I/O pairs $(\gamma(s,t), \zeta)$. These are calculated from the geometry definition in the classical manner by checking rays against all surfaces and calculating the kernel value. The first set is calculated randomly, all further automatically through the SGCS to account only for those zones which are not sufficiently examined.

The complete functional description of the approximated kernel $\tilde{k} : \mathbb{R}^2 \to \mathbb{R}$ is defined by the concatenation of the two functions, $\gamma : \mathbb{R}^2 \to \mathbb{R}^n$ and the SGCS, $\kappa : \mathbb{R}^n \to \mathbb{R}$,

$$\tilde{k} = \gamma \circ \kappa(s,t),\ 0 \leq s,t < 1.$$

κ consists of the set of z basis functions, $M_i, i = 1, ..., z$. The approximated kernel value κ is calculated as described in equation (6). The output weights of each cell, $\mathbf{v}_i \in \mathbb{R}, i = 1, ..., z$, which connect the output layer of the SGCS are considered as the transfer coefficients according to a kernel basis function M_i.

4 Results

Figure 3 shows an example Flatland geometry (a), and the kernel function (b). The blocker is not included into the parametrization. (d) is an approximation through the SGCS, and (c) exposes the centers of the RBFs. For the visualization

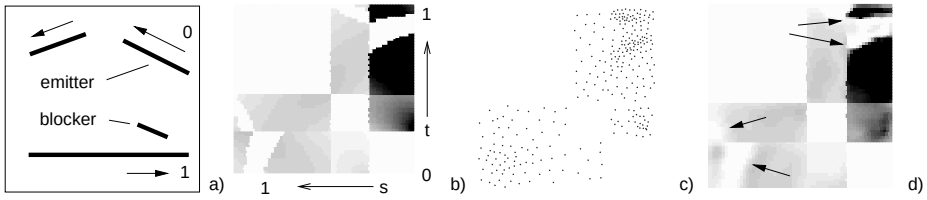

Fig. 3. An example Flatland geometry (a), the kernel (b), the distribution of the basis functions (c), and the kernel approximation (d).

in 2D (c), the four-dimensional reference vectors are re-projected according to their average parameters s and t for the selected samples.

It can be observed that the resolution of the basis functions depends on the variation in the kernel. The sharp curves in the kernel function, arising from the effect of the blocker (arrows in (d)), are represented by more RBFs than the smooth areas. Even for a small energy transfer the coherence is higher (lower arrows in (d)), and thus fewer RBFs are placed by the SGCS.

We tested the clustering and the iterative facilities with a large geometry. The edges of the example Flatland geometry were subdivided into 1000 equally spaced edges.The resulting cell distribution was the same like in the unsubdivided case. This is evident, due to the fact that the network sees nearly the same information (sample rays). The convergency of the whole algorithm was the same, in contrast to HR which would generate 10^6 default basis functions to get the first approximation. Even the approximation accuracy (L_2 error) of the SGCS were about 50% higher compared with a HR approach and for an equal number of basis functions (about 400 in this case).

The comparison of the number of samples between the GCR approach and HR is questionable, since the relationship between accuracy and the number of samples in the HR approach has to be examined further. We can state, that the ratio of the number of cells and the number of samples is about 20. We get a similar number of sample shots in HR if we use also 20 samples to determine whether to descend a level in the hierarchy (*oracle*). Thus, the number of samples does not differ in a fundamental manner.

5 Summary

We proposed a method which represents the radiosity kernel, even for huge geometries with few basis functions. By accounting for the coherence in the global kernel function, the final representation is very sparse. Thus the well-known problem of *initial linking* which creates default basis functions for each pair of surfaces could be avoided.

The representation by the model is completely adaptive, which means that the algorithm behaves robustly in case of slightly changing geometry or lighting conditions. This delivers also a promising outlook on an incremental radiosity approach.

Future work. Although this work considers only Flatland, the algorithms can be modified for calculation of realistic three-dimensional scenes. For that, the SGCS scheme has to be adapted to four dimensions according to the radiosity kernel for 3D scenes. This is described in [9]. Also [12] can be applied to account for the dimensionality of the underlying data implicitly. All assumptions in this work hold for SGCS networks of any dimension and also for the flexible approach [12].

The next step will be to incorporate the energy transfer, i.e., to find basis functions for the radiosity which suit to the generated kernel basis functions.

Acknowledgment. The author wishes to thank Heinrich Müller for valuable discussions and advises.

References

1. James T. Kajiya. The rendering equation. In David C. Evans and Russell J. Athay, editors, *Computer Graphics (SIGGRAPH '86 Proceedings)*, volume 20, pages 143–150, August 1986.
2. Paul Heckbert. Radiosity in flatland. *Computer Graphics Forum (Eurographics '92)*, 11(3):181–192, September 1992.
3. Michael F. Cohen and John R. Wallace. *Radiosity and Realistic Image Synthesis*. Academic Press Professional, San Diego, CA, 1993.
4. Francois Sillion and Claude Puech. *Radiosity and Global Illumination*. Morgan Kaufmann, San Francisco, 1994.
5. Pat Hanrahan, David Salzman, and Larry Aupperle. A rapid hierarchical radiosity algorithm. In Thomas W. Sederberg, editor, *Computer Graphics (SIGGRAPH '91 Proceedings)*, volume 25, pages 197–206, July 1991.
6. Steven J. Gortler, Peter Schröder, Michael F. Cohen, and Pat Hanrahan. Wavelet radiosity. In *Computer Graphics Proceedings, Annual Conference Series, 1993*, pages 221–230, 1993.
7. Brian Smits, James Arvo, and Donald Greenberg. A clustering algorithm for radiosity in complex environments. In Andrew Glassner, editor, *Proceedings of SIGGRAPH '94 (Orlando, Florida, July 24–29, 1994)*, Computer Graphics Proceedings, Annual Conference Series, pages 435–442. ACM SIGGRAPH, ACM Press, July 1994. ISBN 0-89791-667-0.
8. François Sillion. Clustering and volume scattering for hierarchical radiosity calculations. In *Fifth Eurographics Workshop on Rendering*, pages 105–117, Darmstadt, Germany, June 1994.
9. Bernd Fritzke. Growing cell structures - a self-organizing network for unsupervised and supervised learning. Technical Report ICSI TR-93-026, International Computer Science Institute, Berkeley, CA, May 1993.
10. J. Moody and C. Darken. Fast learning in networks of locally-tuned processing units. Technical Report YALEU/DCS/RR-654, Dept. of Computer Science, Yale University, New Haven, CT, 1989.
11. J. Hertz, A. Krogh, and R. Palmer. *Introduction to the Theory of Neural Computation*. Addison-Wesley, 1991.
12. Bernd Fritzke. Incremental learning of local linear mappings. In *Proceedings of the ICANN-95*, Paris, France, 1995.