Radiosity on Evolving Networks

Dissertation
zur Erlangung des Grades eines
Doktors der Naturwissenschaften
der Universität Dortmund
am Fachbereich Informatik

von
Christian-Arved Bohn

Dortmund
2000

Tag der mündlichen Prüfung:	31. Januar 2000
Dekan:	Professor Dr. Bernd Reusch
Vorsitzender des Promotionsausschusses:	Professor Dr. Ingo Wegener
Vorsitzender der Prüfungskommission:	Professor Dr. Claudio Moraga
1. Gutachter:	Professor Dr. Heinrich Müller
2. Gutachter:	Professor Dr. Markus Gross (ETH Zürich)
3. Gutachter:	Professor Dr. Wolfgang Banzhaf

Abstract

Radiosity approaches — and generally, *finite element methods* (FEM) — are mainly driven by a *discretization* of the domain under consideration which is necessary for the numerical solution of the application specific differential or integral equations.

This discretization can easily be accomplished in those practical problems which "live" in more or less regular domains like, for example, a fluid in a cylindric container, or an electrical field in outer space. But with the arising of *boundary conditions* the medium under consideration is prevented from a unique homogeneous behavior, and thus, the demand for elaborated algorithms as well as for higher computing resources is increased.

Radiosity approaches compute the light transfer inside of virtual architectural scene definitions. Here, the domain consists of arbitrary located geometric elements with sharply bounded edges and abruptly varying surface properties. They limit the virtually smooth propagation of light. Moreover, todays claims to virtual scene descriptions like the possibility of handling several thousands of single patches, dramatically increase the problem of boundaries.

This work presents an alternative way for the discretization of the environment for a radiosity finite element solution. Based on sample "rays" drawn from the geometry, a pair of *artificial neural networks* (ANN) is trained and their inner structures are interpreted as FEM meshing. Representations of the *light flow* as well as the *scene geometry* are generated. Based on this *virtual geometry* the required FEM integration operations are accomplished analytically. The network structure approaches the given geometry with the progress of *learning* and delivers a close functional description of the light flow in the virtual scene.

The Monte Carlo sampling and the simultaneously executed FEM support each other. Thus, samples are drawn directedly by utilizing the information derived from the internal structure of the ANN, and vice versa, an efficient meshing of the scene results from the non-redundant distribution of samples to which the network adjusts.

The main feature of this work is the new kind of generation of an FEM meshing for radiosity, which is completely independent from the scene objects.

The ANN generates its own internal representation of the geometry, and thus, the approach is capable of detecting coherency or clusters of the underlying light flow in an optimal way. Due to the independence from the scene definition, *initial linking* is avoided and rough, fast solutions for even huge geometries are possible. A further essential feature is the combination of sampling and the FEM — importance sampling is driven by the light flow instead of examining the geometry relationships only.

The work is divided into two parts. First, a new ANN algorithm is developed, which is capable of being trained to deliver a general efficient approximation model for arbitrary functions of any dimensionality. Here, emphasis is laid on an automatic importance sampling of the underlying general *goal function* to enable its non-redundant examination. The approximation facility of the resulting network is equivalent to a linear function base of Gaussian *radial basis functions* (RBF) with infinite support. This work's second part describes how this algorithm is applied to the radiosity problem, i.e., to the approximation of the light flow and the scene geometry. A new kind of FEM model for radiosity is developed based on the neural network ("neural meshing").

Mathematical Notations

Notations

When writing **vector** notation, it is implicitly assumed that vectors are columns. Thus, a vector $\mathbf{x} \in \mathbb{R}^n$, *written* by its components would deliver $(x_1, x_2, \ldots x_n)$, although virtually, $(x_1, x_2, \ldots x_n)^\top$ is *meant*.

Considering vectors $\mathbf{y}, \mathbf{z} \in \mathbb{R}^n$, a matrix $\mathbf{M} \in \mathbb{R}^n \times \mathbb{R}^n$ with rows $\mathbf{m}_k \in \mathbb{R}^n$, $k = 1 \ldots n$, and a scalar $v \in \mathbb{R}$, then the multiplication of two vectors is written $\mathbf{z} = \mathbf{x}\,\mathbf{y} = (x_1 y_1, x_2 y_2, \ldots x_n y_n)$, and the inner product $v = \mathbf{x} \cdot \mathbf{y} = \sum_{i=1}^{n} x_i y_i$. Multiplication of a matrix and a vector is denoted $\mathbf{z} = \mathbf{M}\mathbf{y} = (\mathbf{m}_1 \cdot \mathbf{y}, \mathbf{m}_2 \cdot \mathbf{y}, \ldots \mathbf{m}_n \cdot \mathbf{y})$.

Most **characters** in this text are used uniquely. Besides enumeration variables like i, j, etc., or function parameters like s or t, one particular term denotes the same object through the whole text, as follows.

Terms

Radiosity

f_{BSDF}	bidirectional scattering distribution function (BSDF)
ϕ	angles between received and scattered light ray and surface normals
$\mathcal{S}$	point domain (surfaces) of the scene geometry
$\mathbf{x}, \mathbf{y}$	points ($\in \mathbb{R}^3$) of the surface domain
x_i, y_i	components of $\mathbf{x}, \mathbf{y}$
B	radiosity function $\mathcal{S} \to \mathbb{R}$ (monochrome light phenomena assumed)
$B^{(n)}$	radiosity function after the n^{th} integration of the kernel operator
E	emittance function $\mathcal{S} \to \mathbb{R}$
ρ	reflectance function $\mathcal{S} \to \mathbb{R}$
V	visibility function $\mathcal{S} \to \{0, 1\}$

G	geometric term $\mathcal{S} \times \mathcal{S} \rightarrow \mathbb{R}$
K	kernel function $(\rho \cdot G)$ of the radiosity integral equation
n	number of radiosity base function components
$\hat{B}$	radiosity approximation $\mathcal{S} \rightarrow \mathbb{R}$
$\hat{E}$	approximation $\mathcal{S} \rightarrow \mathbb{R}$ of the radiosity emittance
$\mathcal{K}$	operator notation of kernel K
$\mathbf{K}$	matrix notation $(\in \mathbb{R}^n \times \mathbb{R}^n)$ of integration operator $\mathcal{K}$
k_{ij}	components of $\mathbf{K}$
$\mathbf{e}$, e_i	vector $(\in \mathbb{R}^n)$ of emittance coefficients and i^{th} component
$\mathbf{b}$, b_i	vector $(\in \mathbb{R}^n)$ of radiosity coefficients and i^{th} component
ϱ, ϱ_i	vector $(\in \mathbb{R}^n)$ of absorption coefficients (function ρ) and i^{th} component
N_i	orthonormal radiosity base function component
$\tilde{N}_i$	the dual function of N_i
$\mathbf{N}$	vector $(\in \mathbb{R}^n)$ of all $N_i,\ i = 1 \ldots n$
$\hat{n}$	number of segments in $\hat{\mathcal{S}}$

Artificial Neural Networks

n	network input dimension (number of "units" of the 0^{th} layer)
m	network output dimension (number of units of the output layer)
a	number of units in a network
ξ, ξ_i	general input vector (*sample*, $\in \mathbb{R}^n$) and i^{th} component
ζ, ζ_i	general output vector (for supervised learning, $\in \mathbb{R}^m$) and i^{th} component
$\hat{\zeta}$	training (*goal*) sample vector $(\in \mathbb{R}^m)$
f, g	general notation for goal functions
F	general notation for the approximation of a goal function
p	number of layers (0^{th} layer not counted)
m'_k	output dimension of the k^{th} layer
$\boldsymbol{\Phi}^{(k)}$	vector of network unit functions of the k^{th} layer, $k = 1 \ldots p$
$\mathit{\Phi}_i^{(k)}$	i^{th} component of $\boldsymbol{\Phi}^{(k)}$, $i = 1 \ldots m'_k$

$\mathbf{W}$	weight matrix of all weights of a network
$\mathbf{W}^{(i)}$	weight matrix ($\in \mathbb{R}^{m'_{k+1}} \times \mathbb{R}^{m'_k}$) between the i^{th} and the $(i+1)^{\text{th}}$ layer
$\mathbf{w}_j^{(i)}$	the j^{th} row ($\in \mathbb{R}^{m'_{k+1}}$) of $\mathbf{W}^{(i)}$
$w_{jk}^{(i)}$	the k^{th} component of $\mathbf{w}_j^{(i)}$
$\mathcal{Q}$	a set of training samples
$\lvert\mathcal{Q}\rvert$	number of samples in $\mathcal{Q}$

Growing Cell Structures

k	topological dimension of an ISGCS
$\widetilde{k}$	dimension of the distribution of training samples
z	number of hidden units
$\mathcal{A}$	set of cells $\{c_i : i = 1 \ldots z\}$ of a network
c_i	i^{th} cell of $\mathcal{A}$
$\mathit{\Omega}_i$	network unit function (Gaussian) of the first layer, attached to c_i
$\mathit{\Sigma}_j$	network unit function (summation) connecting the first layer and the j^{th} output of an ISGCS
$\mathbf{\Omega}$, $\mathbf{\Sigma}$	vector notation of the sets $\{\mathit{\Omega}_i :\ i = 1 \ldots z\}$ and $\{\mathit{\Sigma}_j :\ j = 1 \ldots m\}$
d_i	standard deviation of $\mathit{\Omega}_i$
$\mathbf{W}$	weight matrix ($\in \mathbb{R}^n \times \mathbb{R}^z$) between the input and the first hidden layer of an ISGCS
$\mathbf{w}_i$	center vector ($\in \mathbb{R}^n$) of a cell c_i (the i^{th} row of $\mathbf{W}$)
$\mathbf{V}$	weight matrix ($\in \mathbb{R}^z \times \mathbb{R}^m$) between the first hidden and the output layer of an ISGCS
$\mathbf{v}_j$	j^{th} row ($\in \mathbb{R}^z$) of $\mathbf{V}$
$\mathbf{\Psi}$	one-dimensional output of an ISGCS , $\mathbf{\Psi} : \mathbb{R}^n \to \mathbb{R}$
θ	denotes the function which searches the best matching unit
$\mathcal{N}_i$	set of cells in the direct neighborhood of a cell c_i
$\mathcal{N}_i^k$	set of cells in the k-neighborhood of c_i ($\leq k$ jumps between a $c_j \in \mathcal{N}_i^k$ and c_i)
f_i	Voronoi volume of a cell c_i

Γ_i	ratio for the change of the Voronoi volume of a cell c_i when adjusting the cell positions in the network
$\hat{f}_i$	Voronoi volume estimation of a cell c_i
h_i	relative signal frequency of a cell i, i.e., frequency of being a BMU
$\hat{p}_i$	probability density estimate of a cell i being a BMU
l	length of an edge between cells
τ_i^{USL}	unsupervised resource term at the i^{th} cell
τ_i^{SL}	supervised resource term at the i^{th} cell
τ_i^{X}	extended resource term of a cell c_i (combination of τ_i^{USL} and τ_i^{SL})
τ_i	general notation for a resource term of a cell c_i
ϵ_{BMU}	moving strength for a BMU during unsupervised learning
$\epsilon_{\mathcal{N}}$	moving strength for the direct neighbors of a BMU during unsupervised learning
η	learning rate of the output layer
α	decreasing rate of the resource terms τ_c^{SL} and τ_c^{USL}
λ	number of iterations until a new cell is inserted
κ	number of iterations until a possible deletion of cells is tested
ε	threshold for deletion of cells
υ_ξ	relation determines if an input ξ is located inside of a network
φ	threshold for being inside
$\varUpsilon_\xi$	network activation for an input ξ (accumulated activation of all cells)
σ_i	relation determines if a cell c_i is "critical"
ω	threshold for being a high resource cell
$\mathcal{A}_{CR}$	set of "critical cells" of a network $\mathcal{A}$
φ_ξ	relation determines if an input ξ lies in a region which needs to be resampled
ν	resampling counter

Growing Cell Structures and Radiosity

$\mathit{\Xi}$	ISGCS function of the kernel approximation
$\mathit{\Omega}_i$	Gaussian function ($\mathbb{R}^n \rightarrow \mathbb{R}$ attached to c_i of the kernel approximation ISGCS ("reference ray"))
$\boldsymbol{\Omega}$	vector notation of the set $\{\mathit{\Omega}_i : i = 1 \ldots z\}$
$\mathbf{w}_i$	center vector ($\in \mathbb{R}^6$) of the kernel network's i^{th} Gaussian
$w_{i,k}$	k^{th} component of $\mathbf{w}_i$
d_i	standard deviation of the kernel network's i^{th} Gaussian
$\mathbf{v}$, v_k	output weight vector ($\in \mathbb{R}^z$) of the kernel ISGCS and k^{th} component
h_{ij}	transfer coefficient multiplied with the emitted radiosity
$\mathbf{H}$	matrix ($\in \mathbb{R}^n \times \mathbb{R}^n$) of the h_{ij}
$\hbar_{pqo}$	fractional transfer coefficient between two base and one kernel Gaussian
$\hat{\hbar}_{pqo}$	approximation of $\hbar_{pqo}$ for flat geometries
$\hbar^{FL}_{pqo}$	fractional transfer coefficient for two-dimensional geometries
$\hat{\hbar}^{FL}_{pqo}$	approximation of $\hbar^{FL}_{pqo}$
$\hat{\mathcal{S}}$	approximation of the scene definition $\mathcal{S}$ through the shading network
$\hat{n}$	number of segments in $\hat{\mathcal{S}}$
$\hat{\mathcal{S}}_i$	i^{th} $(i = 1 \ldots \hat{n})$ segment (*simplex*, triangle) of $\hat{\mathcal{S}}$
$\mathit{\Theta}$	radiosity function represented by the ISGCS shading network
$\hat{z}$	size of the shading network (number of cells)
$\mathit{\Lambda}_i$	Gaussian function ($\mathbb{R}^{\hat{z}} \rightarrow \mathbb{R}$ attached to c_i of the shading ISGCS ("reference vertex")
$\boldsymbol{\Lambda}$	vector notation of the set $\{\mathit{\Lambda}_i : i = 1 \ldots \hat{z}\}$
$\mathcal{M}$	set of bases with components defined by linear combinations of the Gaussians $\mathit{\Lambda}_i$
$\hat{\mathbf{w}}_i$	center vector ($\in \mathbb{R}^3$) of the shading network's i^{th} Gaussian
$\hat{w}_{i,k}$	k^{th} component of $\hat{\mathbf{w}}_i$
$\hat{d}_i$	standard deviation of i^{th} Gaussian component of the shading network

$\mathbf{u}$, u_k output weight vector ($\in \mathbb{R}^3$) of the shading ISGCS ("one-dimensional matrix") and its k^{th} component

$\hat{\mathbf{n}}_i$ normal ($\in \mathbb{R}^3$) of i^{th} base component of the shading network

$\mathbf{A}$ "orthonormality matrix" ($\in \mathbb{R}^{\hat{z}} \times \mathbb{R}^{\hat{z}}$) of the shading network Gaussians

$\mathbf{a}_k$ k^{th} line ($\in \mathbb{R}^{\hat{z}}$) of $\mathbf{A}$

a_{ki} i^{th} component of $\mathbf{a}_k$

SR_i local surface roughness value for a shading network cell i

FSR_i blurred surface roughness value in the environment of a shading network cell i

DF_i depth parameter of a shading network cell i

$B^{(k)}$ radiosity approximation after the k^{th} integration of the kernel network

$s_i^{(k)}$ i^{th} training sample calculated through $B^{(k)}$

$\tilde{\mathcal{I}}_{\text{FI}}$ inner product estimate on an assumed flat shading network

$\mathcal{I}_{\text{FI}}$ inner product estimate on the shading network

$\mathcal{E}_{\text{FI}}$ error of an approximated inner product (calculated from $\tilde{\mathcal{I}}_{\text{FI}}$ and $\mathcal{I}_{\text{FI}}$)

$\tilde{\mathcal{I}}_{\text{FT}}$ transfer coefficient estimate on an assumed flat shading network

$\mathcal{I}_{\text{FI}}$ transfer coefficient estimate

$\mathcal{E}_{\text{FI}}$ error estimate of an approximated transfer coefficient (calculated from $\tilde{\mathcal{I}}_{\text{FT}}$ and $\mathcal{I}_{\text{FT}}$)

$\mathcal{I}_{\text{DI}}$ inner product estimate assuming Gaussians being sufficiently inside of a shading network

$\mathcal{I}_{\text{RI}}$ inner product error estimate of the region of a Gaussian lying outside the shading network

$\mathcal{E}_{\text{DI}}$ error function for the symbolically calculated inner product of two Gaussians depending on the depth of theirs position in the network

$\mathcal{I}_{\text{DT}}$ transfer coefficient estimate assuming Gaussians being sufficiently inside of a shading network

$\mathcal{I}_{\text{RT}}$ transfer coefficient error estimate of the region of a Gaussian lying outside the shading network

$\mathcal{E}_{\text{DT}}$ error function for the symbolically calculated transfer coefficient depending on the depth of its position in the network

Contents

Chapter 1

Introduction

1.1 Digital Image Synthesis

Central challenges of *digital image synthesis* are extraction, interpretation, and visualization of information abstractly stored in a computer memory. In the area of *realistic image synthesis* or *rendering*, these data mostly consist of geometrically defined three-dimensional objects of three-dimensional artificial worlds. They are generated, for example, in fields like architectural design or to explore three-dimensional environments in *Virtual Reality* applications.

In many applications, pictures of artificial worlds show highest expressiveness if they are computed in a way that they look like virtual photographs of virtual scenes as they might exist in reality. Maximizing the degree of realism in these pictures is an important goal in the image synthesis of three-dimensional environments.

Since real photographs are results of the interaction of light with its real environment and with the camera film, rendered images are simulations of virtual light emitted from virtual light sources into virtual environments. The higher the accuracy of this simulation, the higher is the impression of realism which the viewer gets, but the bigger is the amount of required computing resources. Thus, research in digital image synthesis focuses on the development of algorithms which calculate the flow of light with sufficient accuracy, on the one hand, but, on the other hand, by using only a moderate amount of computing resources.

Physics of Light Flow. A *light transport model* or *illumination model* describes how light interacts with its environment. The model may include detailed definitions of the light propagating through a medium, of its reflection from opaque objects, and its transmittance through transparent objects. But even the simplified view on light as an everywhere existing constant brightness value might be taken as a valid illumination model.

In other words, there *does not exist a unique model of light.* Physicists may be interested in the effect of light particles on the aggregate of molecules or its resonance with *light waves.* Opticians would only account for *light rays* which propagate from one light source through different ideally transparent media until they are absorbed by a retina, and architects may like to see their concepts regarding the global model of the whole flow of light of an arbitrary set of light sources, with the phenomena of reflection and re-reflection.

The complexity of these models differ tremendously and each particular user attempts to use the simplest model which suffices the specific application needs.

The application of *rendering* with the goal of generating realistically looking pictures commonly assumes *geometrical optics.* Here, geometric rays are traced through the virtual environment. They originate at light sources and are scattered at surfaces of objects defined by discrete geometric elements like polygons in three-dimensional space. Pictures are drawn by calculating those ray energies which are transported from the point space on the surfaces directly into the virtual viewer's eye. Vacuum is assumed in the space between objects, and thus we are able to formulate a complete light model through the following two phenomena.

1. The kind of scattering[1] a ray at a certain point on the surfaces, mostly referred to as *shading* [20] is commonly described by the *bidirectional scattering distribution function* (BSDF), $f_{\text{BSDF}} : \mathbb{R}^4 \rightarrow \mathbb{R}$. The BSDF is locally defined on each surface point, i.e., depends on local surface properties. It delivers the ratio for the differential portion of the received energy which is scattered into a certain sphere direction.

2. The linkage of single points in terms of the portion of differential energy moving from one to another point — the *flow* of light — is formulated through a directional integral of the energy scattered or emitted[2] from the surface points into the environment.

This is one of the most comprising light transport formulations as it is required for the challenges arising in the field of computer graphics. It is the basis of the following simplification, called the *radiosity illumination model*, but for the comprehensiveness of this work we refer to the constitutive work, the *rendering equation* [30], to an outstanding comprehensive book [20], and to a recent attempt at solving the general problem [63].

[1] Transparency effects are ignored in this work.

[2] in cases where the surface is a light source

The following section describes the radiosity illumination model which this work is based on. This kind of simplification of the general rendering equation enables very short calculation times and also offers several practical advantages described below.

1.2 The Radiosity Illumination Model

When simplifying the general light model, we must be aware of the fact that commonly the demanded degree of realism is also reduced. The term radiosity describes the certain class of algorithms which simplify the scattering properties on the surfaces to *ideally diffuse* reflection and emission (*Lambertian reflection*). This means that the intensity of a surface point is equal for all scattering directions and it is determined by the accumulation of the incoming light at this point, only. On the one hand, the remaining degree of realism of the resulting pictures is high enough due to the fact that by far the most objects in realistic environments mainly consist of ideally diffuse reflecting and emitting objects, on the other hand, the gain of complexity reduction over the general model is tremendous because of two points. First, the rendering equation itself is substantially simplified, second, due to the fact that a point looks the same independently from the observer's direction, commonly available hardware graphics accelerators can effectively be utilized for the final display.

The flow of light in the radiosity illumination model is described by the *radiosity integral equation*,

$$B(\mathbf{y}) = E(\mathbf{y}) + \rho(\mathbf{y}) \int_{\mathcal{S}} G(\mathbf{x}, \mathbf{y})\ B(\mathbf{x})\ d\mathbf{x}. \tag{1.1}$$

$\mathbf{x}, \mathbf{y} \in \mathbb{R}^3$ are points which lie in the two-dimensional surface space $\mathcal{S}$ defined through the scene description[3]. The radiosity value $B : \mathbb{R}^3 \to \mathbb{R}$ at a point $\mathbf{y}$ is the value of its light intensity[4] (energy per unit time per area) determined by the weighted accumulation (integral) of the radiosities of all points $\mathbf{x}$ which emit light into the environment.

[3] Deliberately, the placements of single terms in the following formulations of the radiosity integral equation differ from classical literature, like they also do within these previous works. For example, the locations of π and ρ are, for consistency, sometimes switched between places inside or outside of the integral. This is a valid approach, since they do not depend on the integration parameter.

[4] For simplicity, B is assumed being a one-dimensional intensity value sufficient for the description of pure monochrome light. The transition to colors is explained in subsequent paragraphs.

The *geometric term* $G : \mathbb{R}^3 \times \mathbb{R}^3 \rightarrow \mathbb{R}$,

$$G(\mathbf{x}, \mathbf{y}) = \frac{\cos\phi_{\mathbf{x}} \cos\phi_{\mathbf{y}}}{\pi \left\|\mathbf{x} - \mathbf{y}\right\|^2} \cdot V(\mathbf{x}, \mathbf{y}) \tag{1.2}$$

describes the geometric properties of the light transport through a vacuum. It depends on the distance between $\mathbf{x}$ and $\mathbf{y}$ and the angles of the surface normals at $\mathbf{x}$ and $\mathbf{y}$ with the direction of the connecting ray (see figure 1.1). For its derivation, see for example [58, 20]. $V : \mathbb{R}^3 \times \mathbb{R}^3 \rightarrow \{0, 1\}$ is called the *visibility term*. Its value depends on the mutual visibility of the points $\mathbf{x}$ and $\mathbf{y}$ and delivers the value zero if the direct view between $\mathbf{x}$ and $\mathbf{y}$ is occluded by a scene object, otherwise one.

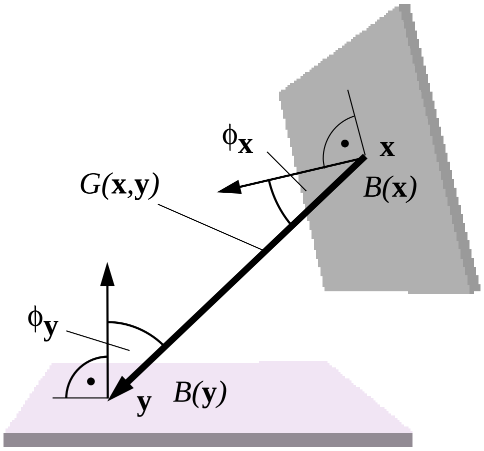


Figure 1.1: The geometric term of the radiosity integral equation (eq. 1.2).

$\rho : \mathbb{R}^3 \rightarrow \mathbb{R}$ is called the *reflectance term*. It is the reduced form of the general f_{BSDF} and determines which portion of light is absorbed for one reflection at a point $\mathbf{y}$. ρ can be interpreted as the brightness of the particular surface, and finally, the term $E : \mathbb{R}^3 \rightarrow \mathbb{R}$ defines the self-emittance at each point of the environment (light sources).

Intuitively spoken, equation (1.1) evaluated at all points $\mathbf{y} \in \mathcal{S}$ for given values for B at all $\mathbf{x} \in \mathcal{S}$ describes one propagation of the recent energy state into the environment. Its repetition accounts for several reflections according to the simulation of the whole flow of light until it is absorbed to a certain degree through the terms ρ and G.

Up to now, we have been speaking just about "intensities". This is sufficient for examining "monochrome geometries", like for example only gray objects resulting in black-and-white pictures. Examining colored environments, the light spectrum must be taken into account by utilizing a well-known short-cut — the separate calculation of the intensities of three different color bands, and the summation of the results. In this case, the reflectance term and the radiosity are functions $B, \rho, E : \mathbb{R}^3 \rightarrow \mathbb{R}^3$ with each output component defining a value for a separate color band.

If not mentioned explicitly, we describe this work related to one general intensity value only, and do not account for several color bands. Thus, B, ρ and E are univariate functions at a first attempt. Its extension to an arbitrary number of color bands is described in chapter 4.

1.3 Solving the Radiosity Equation

B, E, ρ, and G are continuous functions defined on an infinite number of points which describe the surface domain. Central idea in solving the radiosity equation is a *discretization of the environment* and herewith the mentioned functions to turn the integration into a summation.

Intuitive explanations of this discretization have widely been used, for example, in the classical approaches [21, 44] and [12], but also in more recent work like [25]. Here, surfaces are cut into n subpatches and single radiosity constants $b_i : i = 1 \dots n$ are taken as approximation of the average intensity shading over the patches' range. Then, the numerical solution of the radiosity equation is performed by initializing the radiosities b_i with its average emittances derived from E and by exchanging these energies between each pair (i, j) of subpatches weighted by the related *formfactor* k_{ij}, $i, j = 1 \dots n$. The formfactor is calculated by integrating the term G over the domains of the i^{th} and j^{th} subpatch. The radiosity is accumulated on each of the subpatches and then repropagated several times until the energy which is not absorbed is driven below a certain threshold.

In contrast to this more intuitive formulation, it has been recognized that radiosity belongs to the large class of general *finite element* problems. In particular, equation (1.1) has the shape of a *Fredholm integral equation* of the second kind (see [20] for a comprehensive classification), which has been used in several fields of physics. For consistency, we define the *kernel* $K : \mathbb{R}^3 \times \mathbb{R}^3 \rightarrow \mathbb{R}$ like

$$K(\mathbf{x}, \mathbf{y}) = \rho(\mathbf{y})\ G(\mathbf{x}, \mathbf{y}) \tag{1.3}$$

and modify the notation of the radiosity integral equation (eq. 1.1) to

$$B(\mathbf{y}) = E(\mathbf{y}) + \int_S K(\mathbf{x}, \mathbf{y})\ B(\mathbf{x})\ d\mathbf{x}, \tag{1.4}$$

which delivers a convenient transition to operator notation

$$B = E + \mathcal{K}B, \tag{1.5}$$

with the *radiosity transport operator* $\mathcal{K}$ defined according to equation (1.3) by (see also [60], page 182)[5]

$$(\mathcal{K}B)(\mathbf{y}) = \rho(\mathbf{y}) \int_{\mathcal{S}} G(\mathbf{x}, \mathbf{y}) B(\mathbf{x})\, d\mathbf{x}. \tag{1.6}$$

Its solution is primarily an integration task and analytical solutions for the radiosity problem have not yet been found for the general case.

In this work, solving the radiosity integral equation is based on a *projection method* leading to a discretization of equation (1.5) through its representation by a finite set of base functions. In the following, we do not regard Monte Carlo methods. For their description see [30, 53, 63, 20].

Finite Elements and Radiosity

Projection methods under the general framework of finite element methods have been applied in most radiosity approaches. The same holds for the presented work which is based on a general FEM approach — the *Galerkin method* [66, 54]. It assumes the radiosity B represented by an approximation

$$\hat{B}(\mathbf{x}) = \sum_{i=1}^{n} b_i N_i(\mathbf{x}) \tag{1.7}$$

through a linear function space $\{N_i : \mathbb{R}^3 \to \mathbb{R},\ i = 1 \dots n\}$ with the unknown *radiosity coefficients* $b_i, i = 1 \dots n$. Similarly, E can be written as linear combination of the same base weighted by coefficients $e_i, i = 1 \dots n$ like

$$\hat{E}(\mathbf{x}) = \sum_{i=1}^{n} e_i N_i(\mathbf{x}). \tag{1.8}$$

For the following derivation, we use vector notation changing equations (1.7) and (1.8) to

$$\hat{B}(\mathbf{x}) = \mathbf{N}(\mathbf{x})\, \mathbf{b} \tag{1.9}$$

and

$$\hat{E}(\mathbf{x}) = \mathbf{N}(\mathbf{x})\, \mathbf{e}, \tag{1.10}$$

[5] Note that equations (1.5) and (1.6) are short forms of $B(\mathbf{y}) = E(\mathbf{y}) + \mathcal{K}(B(\mathbf{x}))$ and $(\mathcal{K}(B(\mathbf{x})))(\mathbf{y}) = \rho(\mathbf{y}) \int_{\mathcal{S}} G(\mathbf{x}, \mathbf{y}) B(\mathbf{x})\, d\mathbf{x}$, respectively.

respectively, with a vector $\mathbf{b} = (b_1, b_2 \dots b_n)$ of the radiosity coefficients, a vector $\mathbf{e} = (e_1, e_2 \dots e_n)$ of coefficients for the emitted energy, and a vector of the base function components $\mathbf{N} = (N_1, N_2 \dots N_n)$.

Through equations (1.9) and (1.10) the radiosity integral equation (eq. 1.4) can be reformulated as

$$\mathbf{N}(\mathbf{y})\mathbf{b} = \mathbf{N}(\mathbf{y})\mathbf{e} + \mathcal{K}(\mathbf{N}(\mathbf{x})\mathbf{b})(\mathbf{y}). \tag{1.11}$$

In order to get a discrete form of equation (1.11), we eliminate the function base components on both sides by applying the linear *projection operator* $\left[\left\langle \widetilde{\mathbf{N}} \middle| \cdot \right\rangle\right]$[6]. $\widetilde{\mathbf{N}} = (\tilde{N}_1, \tilde{N}_2 \dots \tilde{N}_n)$ is the dual vector of $\mathbf{N}$ defined through the dual functions $\tilde{N}_i$, where $\left\langle N_i \middle| \tilde{N}_j \right\rangle = \delta_{ij}$ holds and δ_{ij} denotes the *Kronecker delta*. The inner product $\langle \cdot | \cdot \rangle$ of two functions f and g is defined as $\langle f | g \rangle = \int_\infty f \cdot g$, (see also [60] and [2]). It follows

$$\left[\left\langle \widetilde{\mathbf{N}} \middle| \mathbf{N}\mathbf{b} \right\rangle\right] = \left[\left\langle \widetilde{\mathbf{N}} \middle| \mathbf{N}\mathbf{e} \right\rangle\right] + \left[\left\langle \widetilde{\mathbf{N}} \middle| (\mathcal{K}\mathbf{N})\mathbf{b} \right\rangle\right]. \tag{1.12}$$

Since $\left[\left\langle \widetilde{\mathbf{N}} \middle| \mathbf{N} \right\rangle\right] = \mathbf{I}$, where $\mathbf{I}$ denotes the identity matrix, equation (1.12) can be reformulated like

$$\mathbf{b} = \mathbf{e} + \mathbf{K}\mathbf{b}, \tag{1.13}$$

with the discrete transport operator — the *transport matrix* — $\mathbf{K}$, consisting of single *transport coefficients* k_{ij} between a pair of function components $\tilde{N}_i$ and N_j, with $\mathbf{K} = \{k_{ij} : i, j = 1 \dots n\}$ defined by

$$\begin{aligned} k_{ij} &= \left\langle \tilde{N}_i(\mathbf{y}) \middle| \int_{\mathcal{S}} K(\mathbf{x}, \mathbf{y}) N_j(\mathbf{x}) \, d\mathbf{x} \right\rangle \\ &= \left\langle \tilde{N}_i(\mathbf{y}) \middle| \rho(\mathbf{y}) \int_{\mathcal{S}} G(\mathbf{x}, \mathbf{y}) N_j(\mathbf{x}) \, d\mathbf{x} \right\rangle \\ &= \int_{\mathcal{S}} \tilde{N}_i(\mathbf{y}) \rho(\mathbf{y}) \int_{\mathcal{S}} G(\mathbf{x}, \mathbf{y}) N_j(\mathbf{x}) \, d\mathbf{x} \, d\mathbf{y}. \end{aligned} \tag{1.14}$$

The shape of equation (1.13) even proposes a method for its solution — the calculation of a finite Neumann series

$$\mathbf{b} = (\mathbf{I} + \sum_{k=1}^{\infty} \mathbf{K}^{(k)}) \, \mathbf{e}, \tag{1.15}$$

where the components of $\mathbf{e}$ are defined by

$$e_i = \langle E | N_i \rangle, \ \forall i = 0 \dots n.$$

[6] "[...]" denotes the creation of a vector or a matrix.

Intuitively, equation (1.15) is evaluated by executing equation (1.13) through the recursion

$$\begin{aligned} \mathbf{b}^{(0)} &= \mathbf{e} && (1.16)\\ \mathbf{b}^{(k+1)} &= \mathbf{e} + \mathbf{K}\mathbf{b}^{(k)},\ k = 1 \ldots \infty && (1.17) \end{aligned}$$

which converges due to the *spectral radius* of $(\mathbf{I} - \mathbf{K})$ being less than one (see, for example, [13], pgs. 110-111).

Every $\mathbf{b}^{(k)}$ defines a new approximate solution of the radiosity, which the coefficients $\mathbf{b}$ on the right side is replaced with. Each evaluation of equation (1.17) — each element of the Neumann series — can be interpreted as another reflection of the propagated light at the geometry.

This type of solution by a *relaxation method* is most common in the field of radiosity since general inversion techniques mostly fail due to the size of the linear system

$$(\mathbf{I} - \mathbf{K})\mathbf{b} = \mathbf{e}. \qquad (1.18)$$

Equation (1.14) is the general formulation of a transport coefficient between two function base components, from which the transfer coefficients of classical radiosity approaches (for example the formfactor) can be derived.

Generally the radiosity base does not require to be orthogonal. In this case, the linear system must be modified, i.e., single components of the radiosity base influence each other, and even the transfer matrix must be regarded under this influence. In this case, computational resources are increased and the simple form of a linear system must be extended aggravating the general view on this work. Thus, in the following, we assume the radiosity problem being focused on orthonormal bases, where all N_i equal their duals $\tilde{N}_i$.

Several authors assume an orthonormal function base [12, 66, 25] — transfer coefficients change to

$$k_{ij} = \int_S N_j(\mathbf{y})\rho(\mathbf{y}) \int_S G(\mathbf{x},\mathbf{y}) N_i(\mathbf{x})\ d\mathbf{x}\ d\mathbf{y}.$$

The approaches [21, 44, 12] utilize a further implicitly simplified form of equation (1.14) like

$$k_{ij} = \int_S \rho(\mathbf{y}) \int_S G(\mathbf{x},\mathbf{y})\ d\mathbf{x}\ d\mathbf{y}.$$

Here, constant base function components with a value of 1 have been defined implicitly supporting the surfaces only locally, i.e., the base components do not overlap.

The definition of the classical formfactor (see [13]),

$$k_{ij} = \rho_j \iint_S G(\mathbf{x},\mathbf{y})\ d\mathbf{x}\ d\mathbf{y},$$

originates from the two simplifications that, first, the absorption term $\rho(\mathbf{y})$ is assumed being constant for one patch, and second, the function base components N_j do not reach over more than one patch. Thus, ρ does not vary on points $\mathbf{y}$ when integrating over one N_j, and it can be pulled out of the integral into a constant coefficient ρ_j.

1.4 Motivation

The radiosity function must be adequately represented through the distribution of radiosity base components on the geometry. This implies, on the one hand, that the amount of them suffices for mimicking the shape of the unknown radiosity, on the other hand, the number of bases must be kept as small as possible, since, with them, the required computing resources grow drastically (eq. 1.13).
These goals are challenged by the following two issues.

a) Default discretization. It is hardly possible to invent a function base with its components reaching over separate surface elements.

Thus, surface boundaries, more or less randomly generated through a modeler program, are commonly tolerated for settling down of at least one single base component each. This *default discretization* (*initial linking*) does not account for any efficiency criterion, and thus, nowadays geometry requirements of up to several tenths of thousands of polygons quickly set limits to solving the radiosity equation.

b) Sequencing issue. Many classical radiosity methods regard the discretization task as two separate processes like, first, creating a radiosity base, and second, calculating the corresponding transfer coefficients (eq. 1.14). Although, for efficiency reasons, most approaches slightly couple these separate tasks, an ideal way to solve the problem seems to be generally impossible, due to the fact that as long as a solution of B is not known, a sufficient criterion for its approximation is not available — and, vice versa, as long as the radiosity base and with it the transfer coefficients are not determined, a solution for B can not be computed.

These two facts have been directing this work to, on the one hand, focusing attention neither on B nor on K, separately, but on the combined term of the *light flow* KB, and on the other hand, to not accounting for the surface geometry explicitly, but instead, to examining the goal function KB implicitly *by sampling*.

Previous Work. Consider classical radiosity approaches and their regard to the term KB.

- In classical radiosity (CR) [21, 44] an arbitrary base on the surfaces is created (subdivision into subpatches). Since the subdivision does not regard the intensity bleeding (which is unknown yet), a common aid has been to estimate[7] the solution B — which virtually can be seen like an estimation of the light flow KB — and then, to develop the subdivision accordingly.

- Progressive radiosity approaches (PR) [12], like they have been implemented in many commercially available software packages [61, 3], compute direct shadows of the light sources (examination of EK) from which an initial discretization is derived.

- Hierarchical approaches (HR,WR) [25, 22] are up to now the most promising attempts of regarding KB, since they start with a coarse discretization of the surface domain, but then, refine patches depending on the actual size of the transfer coefficients multiplied by the emitted radiosity. Additionally, the coherence of KB is estimated by computing the degree of mutual visibility between pairs of radiosity base elements. The resulting representation of the function KB is very compact.

- *Multigridding* methods [25, 13] are commonly applied to all of the above methods. In principle, they can be seen like assuming an arbitrary base at the start of the iteration process on which a preliminary solution B is calculated. From the result an alternate base is derived which is assumed to be more suitable to represent the final bleeding on the surfaces, and then, the iteration process is started again. Here, KB is accounted for implicitly, since its analysis is based on a particular approximation of B on the surfaces.

In these examples, KB has been regarded in a more or less indirect way, and the focus is commonly directed on the surface geometry. The reason is self-evident. Ideally accounting for KB is hardly possible if the geometrical definition of surfaces must be regarded. The existence of polygons hinders the application of well-known efficient mathematical approximation methods for analyzing KB directly, and thus, it prohibits a completely free examination of the light transfer relationships.

[7] intuitively, with the knowledge of an experienced user

1.5 Overview

In order to ideally focusing on the light flow and not only on the geometric term K of the radiosity integral equation, this work is centered around the development of a functional model of the term KB, the actual energy flow through the scene. In the following, we denote KB simply as "kernel" in contrast to the kernel definition used in finite element methods where it corresponds to the geometric term K of the radiosity equation only. The basic idea of this work is instantiated by the following simultaneously executed steps.

a) The term KB is assumed like a general independent goal function $f : \mathbb{R}^6 \to \mathbb{R}$ of directed rays between two points ($\in \mathbb{R}^3$) of the scene geometry. Sets of single rays are drawn from the scene definition[8], and an adaptive function approximation model is *trained* [9] by them. The model adjusts to the *training samples* by creating a compact internal representation through *reference samples* ("reference rays"[10]). These reference rays are taken as centers of infinitely supported Gaussian radial basis functions (RBF), and the whole construct can be seen as a linear function base over Gaussians which approximates the term KB.

   In the following, we denote this approximation model as *kernel network*, which is reasonable due to the algorithm's origin residing in the field of artificial neural networks.

   The approximation scheme is capable of importance sampling of the goal function under examination. This means, starting at an initial arbitrary sample set, it is capable of resampling the scene and creating new sets of samples repeatedly. The scheme accounts for the variance of the goal function.

   The main advantage of this approach comes from the fact that the model is developed virtually independently from the geometry. Thus, it can strictly be separated from classical methods by the fact that it uses a *general* approximation algorithm which does not account for an explicit geometry description like single surfaces. Emphasis is laid on the optimal representation of the kernel function. Clustering and coherence detection facilities of the model, which are described below, can efficiently be utilized.

[8] in the classical way, by testing visibility and calculating G (eq. 1.1)

[9] which means "iteratively adjusted". We stay with this terminology due to the particular approximation model which stems from the field of artificial neural networks.

[10] comparable to *links* in HR

b) The sets of sample values, created while generating the kernel network, ideally account for the coherence in the function KB. This fact is utilized to train a second approximation model, the *shading network*, which is located on the surface domain. The end points of the kernel network's sample rays serve as training samples for the shading network, and thus, due to this indirect connection to KB, it ideally suits to represent the result of the light flow KB, the radiosity.

c) With the described pair of separate networks, the determination of B to get the training value KB and also the final radiosity result have been left. B is derived from an FEM integration at certain time steps during the simultaneous training of the kernel and the shading networks. It is computed based on the internal representations of the two approximation models. For that, the shading network's *reference points* [11] are taken as centers of three-dimensional radial basis functions such that the network defines a linear function base for the radiosity.

   With this radiosity base, the integration operations "through the kernel network" are executed and result in values for coefficients of the radiosity base. The coefficients together with the base deliver a radiosity approximation which is used to adjust the training samples. Finally, the kernel network adapts, in the subsequent training iterations, to the modified training samples and delivers an updated approximation of the light flow. Virtually, the new training set are values $K(E + B^{(1)})$, whereas at the initial state the energy is given by the first element of the Neumann series $KE = KB^{(0)}$.

Consider figure 1.2. The center of the algorithm is a permanently growing sample set of values of $KB^{(n)}$ through which the kernel network as well as the shading network is trained. Resampling is based on examining the geometry and on an approximation accuracy criterion of the kernel network. At certain time steps, the kernel network and the shading network are utilized for an FEM computation whose resulting $B^{(n+1)}$ is taken to adjust the samples already drawn from the geometry. The sample set is constantly adjusted to the actual $KB^{(n)}$ and thus, the light flow is the basic criterion for the evolving of the two networks.

At each time step, the sample set consists of example values of $KB^{(n)}$ with a distribution suiting to the coherence of the light flow. The kernel network contains reference rays of the samples set, which are centers of clusters in

[11] It is generally the same model as the kernel model but approximates a function $g : \mathbb{R}^3 \to \mathbb{R}$.

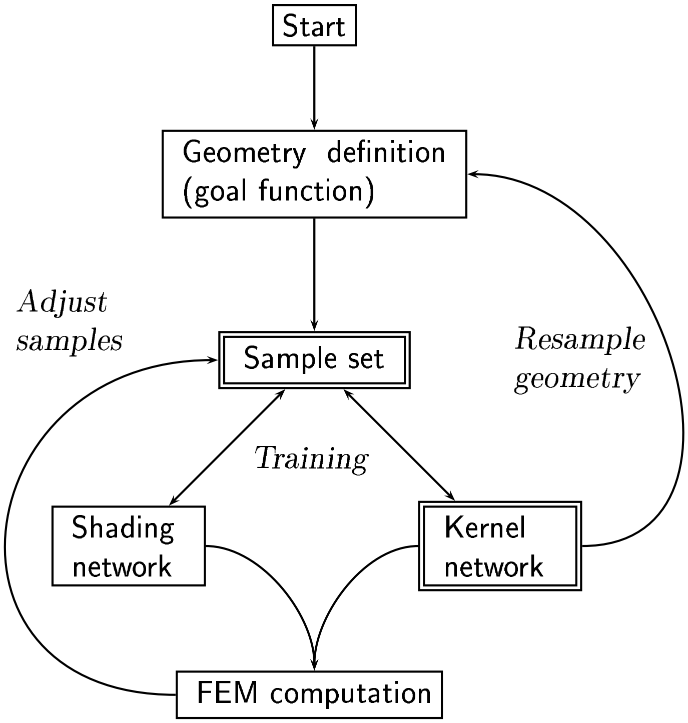


Figure 1.2: The overview of the approach. Kernel and shading networks are trained by the kernel sample set which is adjusted by intermediate FEM computations. The sample set is extended by resampling processes based on the geometry and the FEM result.

the sample distribution. In other words, the kernel network is a compressed instantiation of the light flow.

For the realization of this work, we developed the general neural network function approximation method, described in chapter 2. Its capabilities are proven in chapter 4. For the FEM simulation, a new kind of representation of the surface domain was invented — another ANN (section 3.2). Chapter 3 describes the developments required for connecting both networks. It determines a solution for the analytical integration operations for the FEM. Finally, chapter 4 presents possibilities to reduce the complexity of the whole approach, and shows final results.

Chapter 2

Growing Cell Structures

The *incremental supervised growing cell structures network* (ISGCS) [8, 17, 18] is described in the following. It is regarded as a specific sort of general artificial neural network for clustering and function approximation.

The algorithm initially bases on Fritzke's work on growing cell structures [18]. In section 2.4 we adjust its supervised learning facilities for the ability of general function approximation [8], and in section 2.5 we extend it by a resampling strategy [8] which enables importance sampling of the goal function under examination.

In this work, the method serves as an approximation of the radiosity kernel and delivers the required light flow discretization [6, 7] which we have been asking for in the previous chapter. Additionally, the basic method is utilized for the representation of the radiosity itself — the network topology is taken as a discretization (meshing) of the geometry.

2.1 Approximation Theory

How to *approximate* or *interpolate* a given continuous, multivariate *goal function* $f : \mathbb{R}^n \to \mathbb{R}^m$ by an *approximation model* $F : \mathbb{R}^n \to \mathbb{R}^m$ is the central question in the area of approximation theory. The development of F is justified by its increased tractability and paid by the reduced accuracy of the model if compared to evaluating the original function.

A common application — and the focus of this work — is the approximation of a goal function which is defined *implicitly* by a set $\mathcal{Q}$ of single input-/output values (*samples*) $\mathcal{Q} = \{(\xi_i, \hat{\zeta}_i) \in (\mathbb{R}^n \times \mathbb{R}^m),\ i = 1 \ldots |\mathcal{Q}|\}$. The entirety of the samples characterizes an *abstract* functionality which is *concretized* by the approximation model.

In case of a small sample set, classical interpolation techniques can be chosen, which are characterized by solving a system of linear equations in a way that the final interpolation model matches the sample points exactly. If the number of samples is large, then classical matrix inversion techniques are not convenient. Moreover, if the number of samples varies during the development of the interpolation model (resampling) or if the sample values change (incremental methods), then the mentioned techniques mostly fail. In these cases, like in this work, heuristic approximation methods are more likely. They have been regarded under several different terms like *system identification* or *inductive learning* [45]. The notion *nonparametric regression* has been used in statistics and is probably the best-known paraphrase for this subject.

Searching for such a kind of approximation technique can be characterized by, first, choosing one class from available approximation models, and then, to adjust a set of model parameters or *weights* through the application of a model- and task-specific *weight tuning* algorithm (*learning* or *training*).

Approximation and Networks. Commonly, approximation models can be seen as a conglomerate of a set of (simple) base functions. Describing these combinations in terms of (through the topology of) networks has been introduced to many different areas, due to its representative view on certain classes of algorithms. In fact, the incorporation of networks, and especially artificial neural networks [28, 51, 50], has led to a bunch of new algorithms for solving mathematical problems, which, in many cases, outperform the well-known classical approaches.

This work is based on the extension of a classical ANN algorithm for a general approximation model, and thus, we adopt the ANN terminology, in the following.

2.1.1 Artificial Neural Networks

The essentials of artificial neural networks [52, 28, 50] lie in the fact that they consist of several single *processing units* which execute a simple *activation function* (*combination function* [52]) like for example summation or comparison of its input signals. *Units* are connected in a network of uni-directional weighted *communication channels*. Tuning these *connection weights* means training the neural network, and the entireness of all connections defines its functionality.

Terminology. Artificial neural networks are commonly instantiated by single *layers* which contain several units of the same type of activation function.[1] Layers are interconnected in a *feed-forward* manner[2], one layer can be seen as connected to the successor by a *connection matrix* (see figure 2.1). This

[1] Principally, there may exist different types of activation functions in one layer (see [28]).

[2] There also exist *recurrent networks* [28].

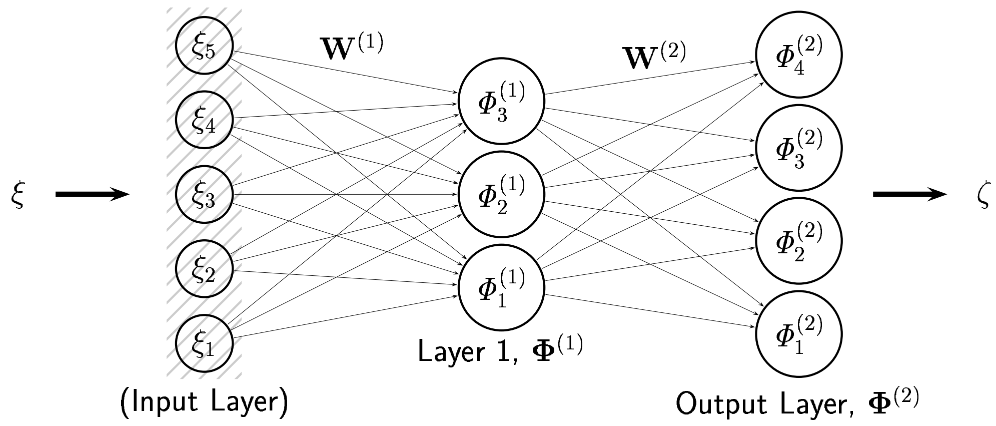


Figure 2.1: An example 5-3-4 feed-forward network with one hidden layer, activation functions $\mathit{\Phi}_i^{(k)}$, and connection matrices $\mathbf{W}^{(k)}$, $k = 1..2$.

connection matrix formalizes the weighted coupling of each of the unit outputs to an arbitrary number of unit inputs (multiple *fan-out*).

Consider an ANN having p layers. For notational consistency, a 0^{th} layer is added [28] carrying identity functions as activation functions. The p^{th} layer is called the *output layer*. *Input-* and output layers connect the network to the "outer world", and the remaining layers 1 to $p-1$ are named *hidden layers* consisting of *hidden units*. An input is propagated through the network from the 0^{th} to the p^{th} layer, weighing the signals by the connection weight matrices and evaluating the activation function at each unit.

The k^{th} layer consists of $m'_k,\ k = 0 \ldots p$, units. The dimensions of the 0^{th} and the p^{th} layer are identical with the input and output dimensions of the network, respectively. Two layers $k-1$ and k are connected by a connection matrix $\mathbf{W}^{(k)} = \{w_{ij}^{(k)}\},\ i = 1 \ldots m'_k,\ j = 1 \ldots m'_{k-1},\ k = 1 \ldots p$. One unit $i,\ i = 1 \ldots m'_k$ of the k^{th} layer contains the activation function $\mathit{\Phi}_i^{(k)} : \mathbb{R}^{m'_{k-1}} \times \mathbb{R}^{m'_{k-1}} \to \mathbb{R}$, of the output signals from the preceding layer, which are weighted by a line $\mathbf{w}_i^{(k)}$ of the layer connection matrix $\mathbf{W}^{(k)}$. The *layer function* $\mathbf{\Phi}^{(k)} : (\mathbb{R}^{m'_k} \times \mathbb{R}^{m'_{k-1}}) \times \mathbb{R}^{m'_{k-1}} \to \mathbb{R}^{m'_k}$, the multivariate output of an array of unit functions of a layer k, $k = 1 \ldots p$, is denoted as the concatenation $\mathbf{\Phi}^{(k)}(\mathbf{W}^{(k)}, \mathbf{x}^{(k-1)}) = (\mathit{\Phi}_1^{(k)}(\mathbf{w}_1^{(k)}, \mathbf{x}^{(k-1)}),\ \mathit{\Phi}_2^{(k)}(\mathbf{w}_2^{(k)}, \mathbf{x}^{(k-1)}) \ldots \mathit{\Phi}_{m'_k}^{(k)}(\mathbf{w}_{m'_k}^{(k)}, \mathbf{x}^{(k-1)}))$, with an input vector $\mathbf{x}^{(k-1)} \in \mathbb{R}^{m'_{k-1}}$ from the preceeding layer's output.

We denote the network output $\zeta \in \mathbb{R}^m$ as function $\mathbf{\Psi} : \mathbb{R}^n \to \mathbb{R}^m$ with $\zeta = \mathbf{\Psi}(\xi)$ for a given input $\xi \in \mathbb{R}^n$. $\mathbf{\Psi}$ equals the output of the p^{th} layer, $\mathbf{\Psi} = \mathbf{\Phi}^{(p)}$, and it is computed through the recursion

$$\begin{aligned} \mathbf{x}^{(0)} &= \xi, \\ \mathbf{x}^{(k)} &= \mathbf{\Phi}^{(k)}(\mathbf{W}^{(k)}, \mathbf{x}^{(k-1)}),\ k = 1 \ldots p. \end{aligned}$$

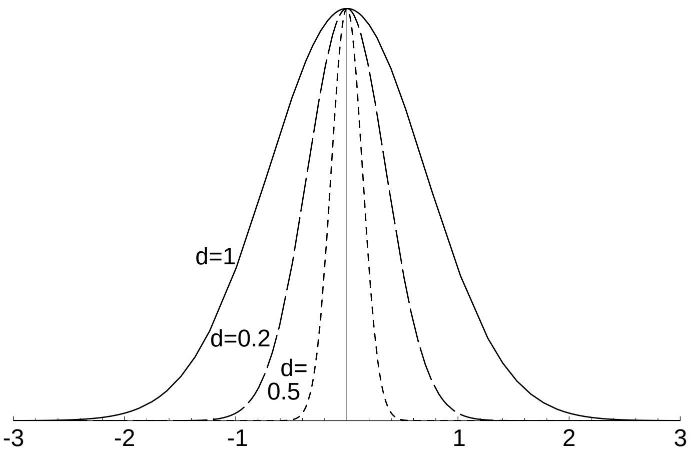


Figure 2.2: Gaussian e^{-dr^2} for varying values of the expansion d

2.2 Basic Topology of an ISGCS

An ISGCS is a two-layered[3] feed-forward network with n units in the 0th layer (the input layer), which are called *cells*, and m output units. In the following, we denote an ISGCS by the set $\mathcal{A}$ of cells $\{c_i,\ i = 1 \dots z\}$.

The first layer of an ISGCS consists of z hidden units, the 0th and the 1th layers are connected by the connection matrix $\mathbf{W}$, and the first and the output layer by $\mathbf{V}$. Here, for notational convenience, we changed the names of the matrices $\mathbf{W}^{(1)}$ and $\mathbf{W}^{(2)}$ to $\mathbf{W}$ and $\mathbf{V}$, respectively. z is equivalent to m_1' from the preceding section.

The first layer consists of z Gaussian radial basis functions [43], $\Omega_i : \mathbb{R}^n \rightarrow \mathbb{R},\ i = 1 \dots z$,

$$\Omega_i(\xi) = e^{-d_i^{-2} r^2} \quad \text{with} \quad r = \|\mathbf{w}_i - \xi\| \,. \tag{2.1}$$

The RBF's output activation depends on the distance of an n-dimensional input vector ξ from an RBF's *center vector* $\mathbf{w}_i$ — rows of the matrix $\mathbf{W}$ (see figure 2.2). In contrast to the preceding section, here, we modified the parametrization of the activation functions in a way that the center vectors $\mathbf{w}_i$ are seen as being a part of the definition of the Ω_i — attached to particular cells c_i of the network. The $\mathbf{w}_i$ are interpreted as the locations of the RBFs, and the RBFs' activations originate just from their distances from the network

[3] input layer ignored

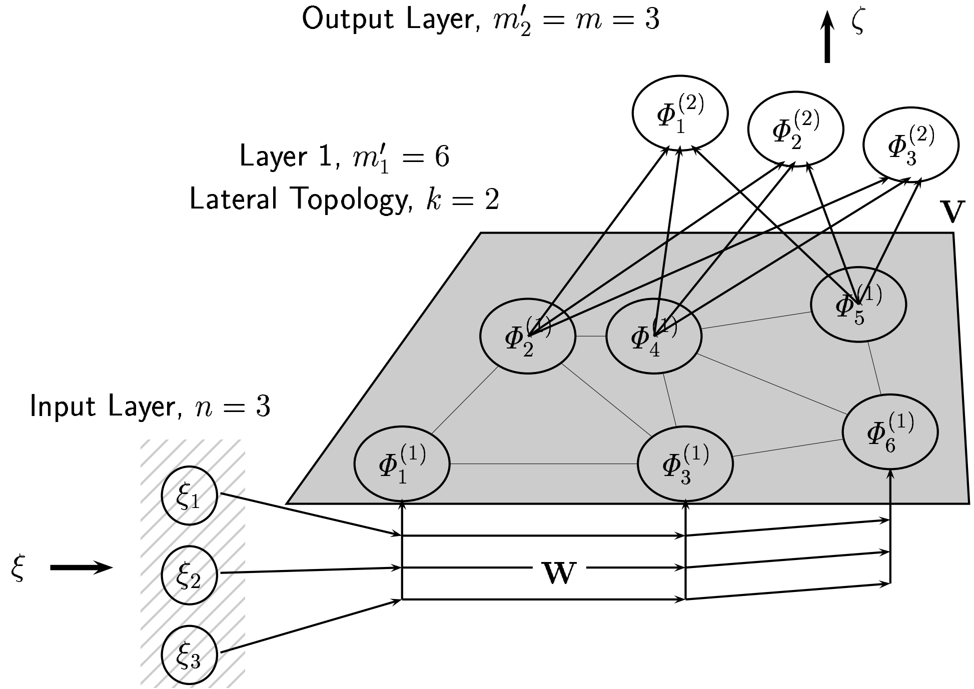


Figure 2.3: A general network, similar to figure 2.1, drawn in a way that eases the interpretation as growing cell structures. The units of layer 1 are instantiated by their weight vectors, rows of the weight matrix $\mathbf{W}$. They are interpreted as the cell positions in the three-dimensional input space. Connections are only drawn partly. A two-dimensional lateral topology (triangulation) can be observed (grey area)[6]. This is explained in section 2.3

input vector. The units can be seen as n-dimensional "representatives" for particular regions in the input space (*reference cells*).

The RBF's *standard deviation* or *smoothness parameter* d_i defines the *expansion* or *range* of an Ω_i and is fixed, i.e., it is not part of the network weights which are tuned in the following learning algorithms[4].

The layer's output function is defined as $\mathbf{\Omega} : \mathbb{R}^z \times \mathbb{R}^n \to \mathbb{R}^z$,

$$\mathbf{\Omega}(\mathbf{W}, \xi) = (\Omega_1(\xi), \Omega_2(\xi) \dots \Omega_z(\xi)). \tag{2.2}$$

Figure 2.3 shows an ISGCS drawn in a style which is common for general *mapping networks*. The cells of layer 1 are three-dimensional units ($n = 3$) — the network input connections. The units are placed regarding an areal topology (exposed as grey area together with embedded connections between

[4] In fact, also the d_i are modified regarding an additional network topology, which is explained in section 2.3. For consistency, this is decoupled from the general ANN training.

their two-dimensional positions). This topological information is explained in section 2.3.

The second layer, the output layer, consists of m functions $\mathit{\Sigma}_j : \mathbb{R}^z \times \mathbb{R}^z \to \mathbb{R}$, $j = 1 \dots m$,

$$\mathit{\Sigma}_j(\mathbf{v}_j, \mathbf{x}) = \mathbf{v}_j \cdot \mathbf{x}, \tag{2.3}$$

which is simply a summation of the activations of the first layer's units $\mathbf{x} \in \mathbb{R}^z = \mathbf{\Omega}(\mathbf{W}, \xi)$ weighted by the connection matrix $\mathbf{V}$. The layer output $\mathbf{\Sigma} : (\mathbb{R}^m \times \mathbb{R}^z) \times \mathbb{R}^z \to \mathbb{R}^m$ is

$$\mathbf{\Sigma}(\mathbf{V}, \mathbf{x}) = (\mathit{\Sigma}_1(\mathbf{v}_1, \mathbf{x}), \mathit{\Sigma}_2(\mathbf{v}_2, \mathbf{x}) \dots \mathit{\Sigma}_m(\mathbf{v}_m, \mathbf{x})) \tag{2.4}$$

and the whole network output $\zeta \in \mathbb{R}^m = \mathbf{\Psi} : \mathbb{R}^n \to \mathbb{R}^m$ is defined as

$$\mathbf{\Psi}(\xi) = \mathbf{\Sigma}\left(\mathbf{V}, \mathbf{\Omega}\left(\mathbf{W}, \xi\right)\right). \tag{2.5}$$

Like the $\mathbf{w}_i$, the $\mathbf{v}_i$ are attached to each of the cells c_i. In the following, we apply this attachment strategy to almost all objects which define local properties of a cell or of its activation function.

Training the ISGCS is divided into two separate parts[6]. The first layer's weights are adjusted according to an *unsupervised learning* strategy, and the output weights are changed in a *supervised* manner. Both strategies are explained in the next section, they can also be seen as the general learning strategies for the training of artificial neural networks. We explain them regarding the particular ISGCS needs.

For later studies we define[7]

Definition 1. *An arbitrary set of $a = |\mathcal{A}|$, $a > 0$ cells and every subset $\mathcal{B}$ of $\mathcal{A}$, with $b = |\mathcal{B}|$ and $0 < b \leq a$ is a valid ISGCS network.*

In the following, when writing the term ΔX regarding a placeholder X for a general variable, then we implicitly assume that this is an alteration value used in a way like

$$X^{(\text{new})} = \Delta X + X^{(\text{old})}.$$

2.2.1 Unsupervised Learning

Unsupervised learning (USL) or *general competitive learning* is applied to the first layer of the ISGCS according to the formula

$$\Delta \mathbf{w}_{\text{BMU}} = \epsilon_{\text{BMU}}(\xi - \mathbf{w}_{\text{BMU}}), \tag{2.6}$$

[6] Although, they are *executed concurrently.*

[7] This definition is different if compared to the classical work [17].

with $\epsilon_{\text{BMU}} \in \mathbb{R}$, $\epsilon_{\text{BMU}} > 0$, a global parameter for the adaption strength and $\mathbf{w}_{\text{BMU}}$ the weight vector of a particular cell c_{BMU} which is determined by the following definition.

Definition 2. *For a particular input $\xi \in \mathbb{R}^n$ of a network $\mathcal{A}$, the* best matching unit *(BMU) is defined as the cell with the index θ where*

$$\begin{aligned}\theta : \mathbb{R}^n \to \{1, 2 \ldots z\},\ (\xi \in \mathbb{R}^n) \mapsto (\theta(\xi) \in \{1, 2 \ldots z\}),\\ \left\|\mathbf{w}_{\theta(\xi)} - \xi\right\| = \min_{i \in \{1,2 \ldots z\}} \|\mathbf{w}_i - \xi\|\end{aligned} \tag{2.7}$$

is the so-called matching function and z the number of cells in $\mathcal{A}$.

Equation (2.7) picks the unit from the network which lies closest[8] to the actual input sample, and equation (2.6) moves it into the direction of the input sample in the n-dimensional space. This action can be interpreted as competing for the best matching of a cell's position against the actual input sample. The cell which wins the competition is moved towards the input to decrease its *matching uncertainty* for the next time when a similar input is presented to the network. It leads to an assignment of cells to particular regions of the input space (*classification*), and the continuous adjustment of these reference cells (*clustering*) delivers an optimal positioning in a cluster's center, in a way that the error of *misclassification* is minimized.[9]

For example, if the input samples presented to the first layer are equally distributed, then, through equation (2.6), the cells will adjust to "span" the distribution uniformly — approximately the same amount of input samples lies inside of one *Voronoi region* [64] of a specific cell. If the input distribution is not of uniform type — clusters of samples in the input space exist — then the cells will adapt to these clusters, i.e., reference cells will arise which stay for different accumulations of samples in certain regions (classification).

This type of learning is called "unsupervised", since the weights of the layer *self-organize* according to an unknown sample distribution.

The presented competitive learning algorithm, principally, is applied in most iterative clustering approaches, like, for example, *k-means clustering* or the *Linde-Buzo-Gray algorithm* [35], or the prominent *Kohonen feature map* [31]. For an overview of classical literature, see [15].

[8] in respect to the Euclidian distance

[9] This scheme is also used in the *Kohonen feature map* algorithm. We refer to it in later sections.

2.2.2 Supervised Learning

Whereas the criterion for unsupervised learning is the distance of a training sample input $\xi \in \mathbb{R}^n$ to a cell weight vector $\mathbf{w}_i \in \mathbb{R}^n$, *supervised learning* (SL) is controlled by the distance of the training sample output $\hat{\zeta} \in \mathbb{R}^m$ to the network output $\zeta \in \mathbb{R}^m = \mathbf{\Psi}(\xi)$, i.e., the error which the network generates by comparing its output with the training sample goal value (*gradient descend learning* [28, 50]).

In the same manner like the misclassification error is minimized in USL by moving some vectors $\mathbf{w}_i$ to approach the position of an input ξ in the n-dimensional input space, the network output vector is moved into the direction of the output training vector in the m-dimensional output space. This is accomplished according to

$$\Delta \mathbf{v}_i = \eta(\hat{\zeta} - \zeta) \cdot \Omega_i(\xi) \quad \text{with} \quad \zeta = \mathbf{\Psi}(\xi),\ \forall i \in \{1 \dots |\mathcal{A}|\}, \tag{2.8}$$

with an adaption strength parameter $\eta \in \mathbb{R}$, $\eta > 0$ and the definition of Ω_i from equation (2.1). Consider, with $\mathbf{v}_i$, we denote in the following those weights of the output matrix which are connected from a particular cell c_i to the output layer (one row of $\mathbf{V}$). Equation (2.8) can be explained like moving the cell output weights into the direction of $\hat{\zeta}$. The strength of this movement depends on the calculated error $(\hat{\zeta} - \zeta)$ and the responsibility of a certain cell for that error — the *cell activation* $\Omega_i(\xi)$.

2.3 Lateral Topology

2.3.1 Training

Up to now, the topology and the learning scheme of the ISGCS do not essentially differ from classical ANN approaches. But now, an additional topological information, a *lateral cell topology* is introduced. It is defined between the cells of the network in a way that the cells build a graph of a predefined fixed *lateral dimension k*. It is predefined in the sense that it stays the same through the whole training process and after training. Thus virtually, the term ISGCS designates a *class of networks* of certain dimensions.

Cells are connected forming single geometric base elements (*simplices*) depending on k. Setting $k = 2$, the simplices are triangles. In case of $k = 1$, the simplest base elements are lines, for $k = 3$ tetrahedrons, and generally, they are called *k-simplices* which span the corresponding k-space.

Moving cells by paying attention to the underlying lateral topology becomes a part of the unsupervised training (see section 2.2.1). A cell c_i in the k-dimensional neighborhood $\mathcal{N}_{\text{BMU}}$ of a best matching unit c_{BMU} is moved according to

$$\Delta\mathbf{w}_i = \epsilon_{\mathcal{N}}(\xi - \mathbf{w}_{\text{BMU}}),\ \forall c_i \in \mathcal{N}_{\text{BMU}}, \tag{2.9}$$

with the moving strength parameter for neighboring cells $\epsilon_{\mathcal{N}} \in \mathbb{R}$, $\epsilon_{\mathcal{N}} > 0$. The neighborhood $\mathcal{N}_i$ of a cell c_i is defined by the lateral network topology, i.e., it is the set of cells with a direct connection c_i.

Definition 3. *The* neighborhood set $\mathcal{N}_i$ *of a cell* $c_i \in \mathcal{A}$ *is the set of those cells which are directly connected to* c_i *through lateral connections.*

On page 35, this definition is extended to the general term "k-neighborhood" which includes all cells located in a particular *topological distance.*

"Three dimensionalities". It follows a more detailed discussion of the mentioned three different types of dimensions.

n — is the **literal dimension** of the input data. It is the dimension of the sample vector and identical to the input dimension of the network. For example, if the network learns a function over three-dimensional points, then the input dimension equals three and the samples are three-dimensional vectors.

$\widetilde{k}$ — For example, although three-dimensional points are represented by three-dimensional vectors, in our case, they lie on two-dimensional surfaces in the three-dimensional space. Thus, their **implicit dimension** $\widetilde{k}$ equals 2.

k — The existence of the topological **lateral dimension** as explained in this section, restricts the cells from completely free distributing in the n-dimensional input space. In case of an implicit dimension which equals the topological dimension, this is not a real restriction, and additionally, a topological structure is created on the reference cells and herewith on the sample data. This structure may help in analyzing the data, i.e., topological information contained in the sample data is implicitly extracted and memorized in the structure of the ISGCS .

One typical application is *dimensionality reduction* where the data are assumed as being of lower implicit dimension than it looks like (due to n). In this type of applications, sample data of very high literal dimension can easily be structured to regard its "real dimension".

In other words, the topological information hidden in the training data is exposed by its "projection" on the internal (lateral) structure of the network — a *topological mapping* [31, 32, 33] is generated.

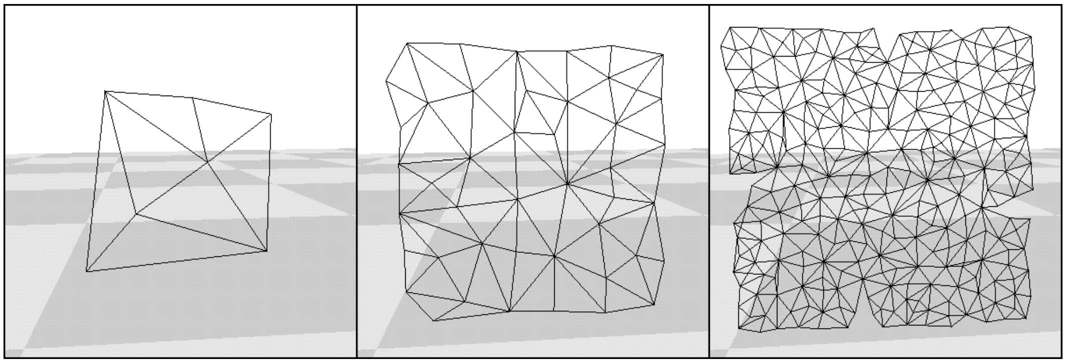


Figure 2.4: A $(k = 2)$-dimensional network trained with $(n = 3)$-dimensional data uniformly distributed on a $(\widetilde{k} = 2)$-dimensional area embedded in $\mathbb{R}^3$. The scheme of insertion of cells will be explained in section 2.3.2.

See for example figure 2.4 and figure 2.5 which show the training process[10] of $(k = 2)$-dimensional networks trained with $(n = 3)$-dimensional data (points) distributed in $(\widetilde{k} = 2)$-dimensional space. Figure 2.5 is created by a nonuniform data distribution.

Besides the application of dimensionality reduction, one can also imagine the inverse case where $\widetilde{k}$ is larger than k — higher dimensional data are projected onto a network of lower dimension. Figure 2.6 shows an example of two networks trained with data for which $\widetilde{k} > k$ holds. On the rightmost side, a one-dimensional network adapts to a two-dimensional space, the left and the mid picture (stereo representation) expose the same for a three dimensional data distribution. Further example applications can be found in [33, 50].

[10] The training, i.e., the growing scheme is explained in section 2.3.2.

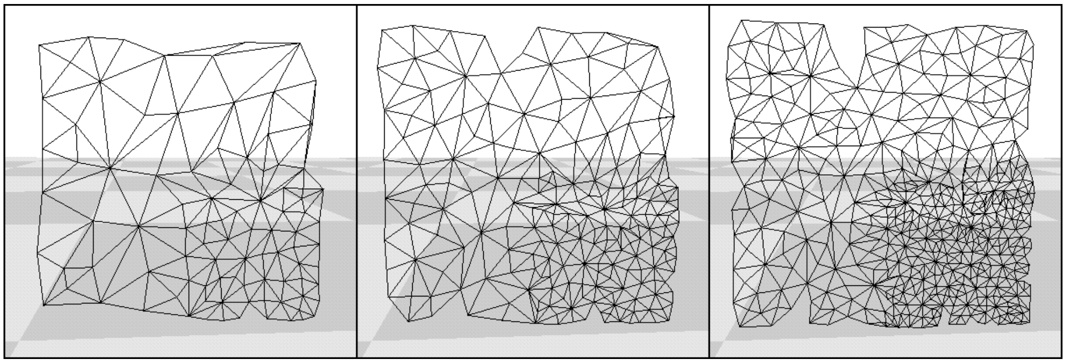


Figure 2.5: The same as in figure 2.4 but with a doubled sample density in the lower right quarter.

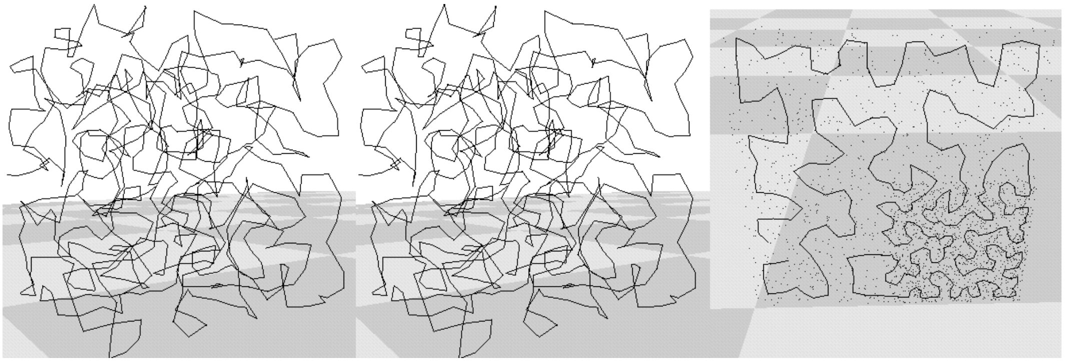


Figure 2.6: In the rightmost picture, a one-dimensional network is trained with two-dimensional data ($n = 3,\ k = 1,\ \widetilde{k} = 2$), in the left and mid picture, with a three-dimensional ($\widetilde{k} = 3$) data set (drawn in stereo representation).

Now, *learning* of an ISGCS (training the connection weights) has been described completely. We add the announced definition of the remaining unknown parameter d_i (eq. 2.1) of the Gaussian of a cell c_i — the expansion coefficient. It is adapted each time in advance to the evaluation of the Gaussians and it is defined by the average length $\overline{l_i}$ of the edges emanating from cell c_i as

$$d_i = \overline{l_i} = \frac{1}{|\mathcal{N}_i|} \sum_{j \in \mathcal{N}_i} (\|\mathbf{w}_i - \mathbf{w}_j\|). \tag{2.10}$$

Equation (2.10) is a simple heuristics which showed practical benefits in [18, 17, 19, 8], and thus, it is adopted in this work. d_i is adjusted each time the unit weights change, i.e., after each training step where the units' positions change.

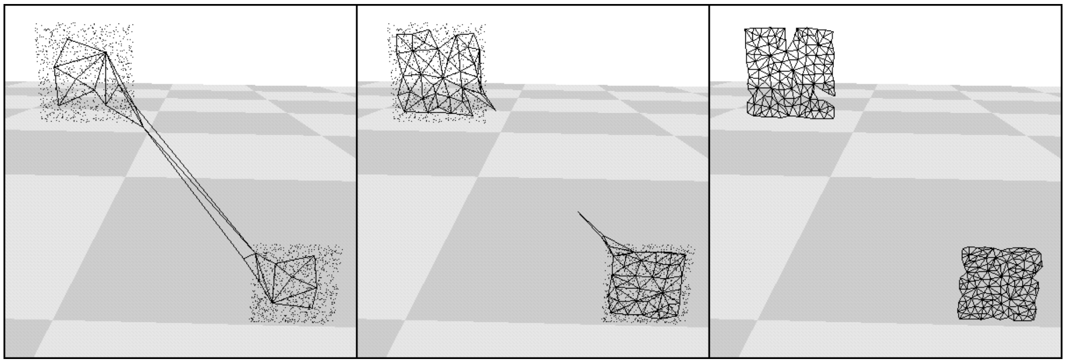


Figure 2.7: Two separate two-dimensional clusters in $\mathbb{R}^3$. The algorithm deletes cells which represent a small sample distribution.

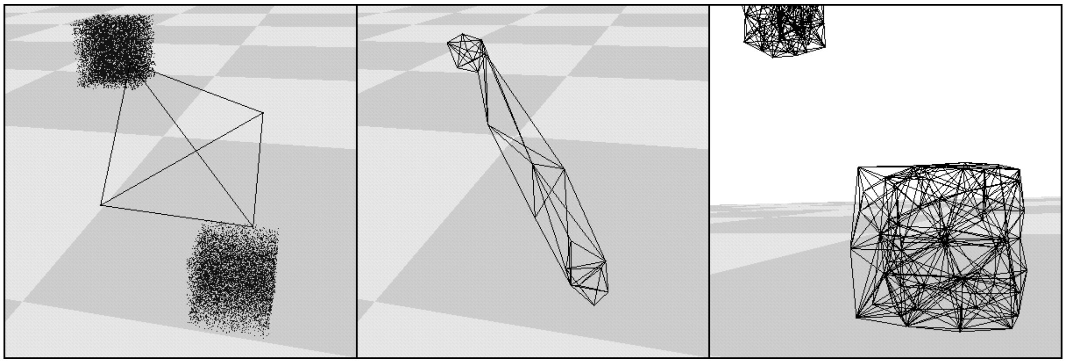


Figure 2.8: A three-dimensional network trained with a three-dimensional separated data set.

With the presented schemes for adjusting the connection weights and d_i, the algorithm got the essential features necessary for supervised and unsupervised learning.

What follows is one of the most important advantages of an ISGCS — *growing* and *shrinking* of the network through *insertion* and *deletion* of cells (see figures 2.7 and 2.8).

Training always starts with the simplest element of the underlying dimension k — one simplex. Then, according to a certain error criterion which is accumulated during the training cycles, new cells are inserted or deleted, and whenever this happens, the basic topology of the network is kept homogeneous, i.e., the network must consist of simplices of the same predefined dimension k, only. The next section shows how this is accomplished.

2.3.2 Insertion of Cells

Adjusting the First Layer's Weights

Insertion of cells is always done by splitting one edge of the network, inserting a new cell in the middle and connecting the surrounding edges such that the topology stays a valid network. We assume a criterion[11] which determines the *edge* to be split between two cells c_q and c_f. The new cell c_r is inserted by the formula

$$\mathbf{w}_r = \frac{1}{2}\left(\mathbf{w}_q + \mathbf{w}_f\right).$$

Each time the units' weights change, commonly the cells' positions in the lateral topology change, and with it, also the term d_i (equation (2.10)) is adapted.

[11] to be explained in section 2.4

Adjusting the Output Layer's Weights

Each cell contributes to the output of the network by its output weights $\mathbf{v}_i$. If a new cell is inserted into the network topology, then intuitively formulated, the contribution of the affected cells must be redistributed according to the new network topology. For that we define a parameter Γ_i for each cell $c_i \in \mathcal{A}$, which can be seen as the ratio of the change of a cell's scope in the k-dimensional network space.

The scope of a cell is given by the cell's Voronoi volume. Since the computation of the Voronoi volume is too complex for $k > 2$ [17], we approximate it by the term $\hat{f}_i$.

$$\hat{f}_i \approx \left(\overline{l_i}\right)^{\widetilde{k}}, \tag{2.11}$$

where $\widetilde{k}$ is the implicit dimension of the training data and $\overline{l_i}$ the mean length of the edges emanating from a cell c_i (eq. 2.10). The parameters of a particular cell which are affected by the alteration of the network topology are adjusted according to its Voronoi volume before and after the insertion or deletion of cells. This relationship is determined by the coefficient

$$\Gamma_i = \frac{\hat{f}_i^{(\text{new})} - \hat{f}_i^{(\text{old})}}{\hat{f}_i^{(\text{old})}}, \ \forall c_i \in \mathcal{A}. \tag{2.12}$$

If a cell c_r is inserted, the Voronoi volumes of the cells in its neighborhood change and equation (2.12) is utilized for an initial approximate determination of the output weight vectors $\mathbf{v}_r$ and that of the neighboring cells $\mathbf{v}_j$, $c_j \in \mathcal{N}_r$ with the equations

$$\begin{aligned} \Delta \mathbf{v}_j &= \Gamma_j \cdot \mathbf{v}_j, \ \forall c_j \in \mathcal{N}_r, & (2.13)\\ \mathbf{v}_r &= -\sum_{j \in \mathcal{N}_r} \Delta \mathbf{v}_j. & (2.14) \end{aligned}$$

Adjusting the Network Topology

As mentioned, the lateral topology of the network must be changed in a way that the defined basic structure will hold. We outline the basic strategy from [17] without going further into detail, since the algorithm strongly depends on the particular implementation-specific data structures. We assume a data structure based on simplices and edges, and modify them as follows.

- **Adjust edges**: connect c_r with all cells which are neighbors of both, c_q *and* c_f;
- **Adjust simplices**: duplicate each k-simplex which contains an edge $\overline{qf}$ from c_q to c_f. Replace $\overline{qf}$ in one of the k-simplices by $\overline{qr}$ and in the other by $\overline{rf}$.

2.3.3 Deletion of Cells

The principal scheme for the deletion of cells is based on the cell's relative signal frequency. It is calculated through tracking the average number of *hits* of a cell, where "hits" means the number of times of the cell c_i being a best matching unit, i.e., the *relative signal frequency* h_i. The *probability density* estimate $\hat{p}_i$ of a cell is calculated according to $\hat{p}_i = h_i / \hat{f}_i$.

The probability density can be taken as a measure for the importance of a cell regarding its contribution to the network task. In case of a low probability density the cell has only little influence and can be deleted from the network. The usage of the probability density, i.e., its extended usage concerning the network's approximation accuracy is explained in section 2.4[12].

We assume a cell c_d is detected for deletion, then deletion is accomplished by deletion of

- **simplices** and **edges** which include cell c_d,
- **cells** which are not member of any simplex.

Parameters and weights of the remaining cells, edges, and simplices are not touched since their influence on the network is small due to their small relative signal frequency.

The described algorithms for modifying the lateral topology should serve as a rough proposal, only, since they strongly depend on the data structures of the particular implementation. For example, on the one hand, it is possible to neglect an explicit edge structure, on the other hand, adding more elaborated structures (like for example *winged edges*, etc.) can increase efficiency. Even the usage of public software libraries might require a reformulation of the demanded tasks in order to fit into the corresponding data structures.

2.4 Resource Term

Up to now, we described *how* to delete and to insert cells, this section explains *if* and *where* to trigger such actions. The scheme is derived from the originating work [17] like the preceeding sections of the growing cell structures approach. In the following, we extend the basic scheme of a resource term for the purposes of this work, enabling general function approximation and a resampling mechanism.

[12] Due to its extension, there, it is renamed to *resource term*.

The decision of when and where to insert new cells in the network is made with the help of the *resource term* attached to each of the cells. It is the extension of the term "probability density" defined in the last section. The resource term is updated each time a sample is presented to the network and it can be seen (to be explained in the following) like a cell-local parameter for a particular "need" of either inserting new cells in that regions, or increasing the number of training samples there (by resampling, see section 2.5).

We differentiate between three different types of resource terms:

1. The *supervised resource term* τ_i^{SL} of a cell c_i is defined as

$$\Delta\tau_{\text{BMU}}^{\text{SL}} = \left\|\hat{\zeta} - \zeta\right\|, \quad \text{with} \quad \zeta = \mathbf{\Psi}(\xi), \tag{2.15}$$

$$\Delta\tau_i^{\text{SL}} = -\alpha \cdot \tau_i^{\text{SL}}, \ \forall c_i \in \mathcal{A}. \tag{2.16}$$

   Presenting a sample I/O pair $(\xi, \hat{\zeta})$ to the network, the supervised resource term τ_i^{SL} is the difference between the goal value $\hat{\zeta}$ to be learned and the output value of the network ζ — the training error. It is accumulated only at the best matching unit c_{BMU} of the network.

   Equation (2.16) provides the network cells with an *aging functionality*. The parameter $\alpha \in \mathbb{R}$, $\alpha > 0$, weighs the influence of more recent events stronger. It determines how fast earlier accumulations "vanish", and virtually, it calculates an average over all τ_i^{SL} during the whole training process.

2. The *unsupervised resource term* (pure probability density) τ_i^{USL} of a cell c_i, principally, is used as a counter for the number of times when a cell was selected as BMU. It is accomplished by equation (2.17),

$$\Delta\tau_{\text{BMU}}^{\text{USL}} = 1, \tag{2.17}$$

$$\Delta\tau_i^{\text{USL}} = -\alpha \cdot \tau_i^{\text{USL}}, \ \forall c_i \in \mathcal{A}. \tag{2.18}$$

   The reason for applying equation (2.18) is the same as for equation (2.16). Counting serves for determining the distribution of the samples which lie in a particular cell's Voronoi region.

3. The *extended resource term* τ_i^{X} of a cell c_i is essentially a specific combination of τ_i^{SL} and τ_i^{USL}. For example, we outline three kinds of usage of the three resource terms to create networks with different functionalities.

   a) *Pure USL* — only τ_i^{USL} is taken as criterion for insertion and deletion of cells. The distribution of the underlying training samples exposes the need for cells, due to the assumption that a high sample distribution should also be represented by a higher distribution of reference cells. The output layer is ignored. Pure clustering of the training data is performed.

b) *USL with an output layer* — the input layer is treated as described in (a). The output layer is trained independently. Similar to point (a), the RBF layer performs clustering, additionally an output function approximation facility is utilized.

The parallel training of the topology and the output layer is the essential advantage of the ISGCS compared to similar methods like hybrid schemes of the *Kohonen feature map* [28, 33]. There, the two training schemes must be executed sequentially, due to the fixed topology of the network. This fact does not allow for a flexible consistent combination of both training schemes.

c) *Combined USL and SL* — the input sample distribution and the approximation error are combined by the definition of an extended resource term like

$$\tau_i^{\mathrm{X}} = \frac{\tau_i^{\mathrm{SL}}}{\tau_i^{\mathrm{USL}}} \cdot \hat{f}_i. \tag{2.19}$$

Equation (2.19) can be seen similar to an L_1 error measure and emulates a very efficient function approximation facility (shown in [8]). The resource term is influenced by the approximation accuracy as well as the sample distribution. This is the essential extension to the classical algorithm [17], since there, only the sample distribution influences insertion and deletion of cells. Here, also the approximation error is utilized, which leads to inserting cells at those places of the goal function which show a higher variance in its function value. Thus, reference cells are inserted and deleted in a way that the goal function is represented more efficiently — more cells are located at regions of a complicated function shape. In case of smooth zones, less cells are positioned according to less function representation requirements.

Point (c) in the last paragraph is only one example for a valid resource term, which has shown sufficient robustness for almost all approximation tasks. Principally, many other combinations are also imaginable. For consistency, in the following we mostly refer to one *general resource term* τ_i which then is a particular definition of τ_i^{X}.

2.4.1 Insertion of Cells

The resource term defined in the last section enables the determination of a certain "need" for a better representation of the goal function by the approximation model — for example, by more cells. In this case, a new cell is inserted into the neighborhood of the cell with the highest resource term in the network.

Paying attention to the lateral topology when inserting cells has been explained in section 2.3.2. Now, we focus on how the resource term of a new cell has to be derived. Due to the fact that an inserted cell c_r has no history in terms of the past iteration process, it must extract a resource term portion from the cells c_i which are already existing in its environment. This affects τ_i^{SL} and τ_i^{USL} only, since these are the only terms which accumulate information during the training process.

Since the Voronoi volumes of the cells are proportional to the resource terms τ_i^{USL} and τ_i^{SL}, they are also used for the resource term modification in case of changing the network topology — similar to modifying the output layer weights (equations 2.13 and 2.14),

$$\tau_r^{\mathrm{SL}} = -\sum_{c_i \in \mathcal{N}_r} \Delta\tau_i^{\mathrm{SL}} \quad \text{and} \quad \Delta\tau_i^{\mathrm{SL}} = \Gamma_i \cdot \tau_i^{\mathrm{SL}},\ \forall c_i \in \mathcal{N}_r, \tag{2.20}$$

$$\tau_r^{\mathrm{USL}} = -\sum_{c_i \in \mathcal{N}_r} \Delta\tau_i^{\mathrm{USL}} \quad \text{and} \quad \Delta\tau_i^{\mathrm{USL}} = \Gamma_i \cdot \tau_i^{\mathrm{USL}},\ \forall c_i \in \mathcal{N}_r, \tag{2.21}$$

with Γ_i like defined in equation (2.12). In other words, the neighboring cells donate a portion of its accumulated resource to the new cell c_r in order to approximate its amount of resource in a way such as if c_r would already have been existing at this place earlier.

The determination of the cell c_r with the highest resource term and the insertion of a new cell is accomplished after $\lambda \in \mathbb{N}$ training steps.

2.4.2 Deletion of Cells

A cell's resource term which has been fallen below a certain threshold $\varepsilon \in \mathbb{R}$, $\varepsilon > 0$, exposes a certain amount of a cell's redundancy in terms of not significantly supporting the representation facilities of the network. Testing this case is done after $\kappa \in \mathbb{N}$ training steps, similar to the cell insertion facility.

If a cell is removed from the network, also the adjacent simplices are removed. Since the cell's Voronoi regions completely vanish, the neighboring cells' resource terms and weights do not need to be touched. The lateral topology is modified according to section 2.3.3.

2.5 Resampling

In this concluding section, we propose a resampling strategy which prevents the goal function under examination from arbitrary sampling. It enables taking a moderate amount of samples at the start, and then to incrementally resample the goal function (importance sampling) according to a criterion derived from the actual training accuracy — the resource term. In the following, we refer to the general resource term τ_i of a cell c_i (see page 2.4), since even for the resampling, the resource can be defined in several ways.

Definition 4. *The* network activation $\Upsilon_{\mathcal{A}} : \mathbb{R}^n \to \mathbb{R}$ *for a set of cells* $\mathcal{A}$ *and an input* $\xi \in \mathbb{R}^n$ *is*

$$\Upsilon_{\mathcal{A}}(\xi) = \sum_{c_i \in \mathcal{A}} \Omega_i(r), \quad \text{with} \quad r = \|\xi - \mathbf{w}_i\| \,. \tag{2.22}$$

□

Definition 5. *An input sample* $\xi \in \mathbb{R}^n$ *is located* inside of *a set of cells* $\mathcal{A}$ *if the network activation* $\Upsilon_{\mathcal{A}}(\xi)$ *exceeds a threshold* φ*,*

$$\upsilon_{\mathcal{A}} : \mathbb{R}^n \to \{T, F\},\ \upsilon_{\mathcal{A}}(\xi) = \left\{ \begin{array}{ll} \mathbf{T} & : \ \text{if } \Upsilon_{\mathcal{A}}(\xi) \geq \varphi \\ \mathbf{F} & : \ \text{else.} \end{array} \right.$$

□

Definition 6. *A cell* c_i *is a* critical cell *if its resource term exceeds the average of all resource terms of a set of cells* $\mathcal{A}$ *to a certain fraction* ω*,*

$$\sigma_{\mathcal{A}} : \{1, 2 \ldots z\} \to \{T, F\},\ \sigma_{\mathcal{A}}(i) = \left\{ \begin{array}{ll} \mathbf{T} & : \ \text{if } \tau_i \geq \omega \cdot \overline{\tau_{\mathcal{A}}} \\ \mathbf{F} & : \ \text{else,} \end{array} \right.$$

$$\text{with} \quad \overline{\tau_{\mathcal{A}}} = |\mathcal{A}|^{-1} \sum_{c_i \in \mathcal{A}} \tau_i .$$

The critical cells of a network $\mathcal{A}$ *define a* critical sub-network $\mathcal{A}_{CR}$ *of* $\mathcal{A}$*,*

$$\mathcal{A}_{CR} = \{c_i : \ c_i \in \mathcal{A}, \sigma_{\mathcal{A}}(i) = \mathbf{T}\}.$$

□

With these three definitions, a sample can be identified as lying “in the range” of a network, and we utilize this capability to detect critical regions in a network, i.e., to recognize if an input ξ lies in a critical region.

Resampling is accomplished by scanning the overall input space with a fixed step size. At each step, the generated input sample is fed into the network without evaluating the goal function but only testing equation (2.22). Depending on the following definition, the network activation triggers the sample as critical or not.

Definition 7. *A position $\xi \in \mathbb{R}^n$ in the input domain lies inside of a* critical region *(CR) if it is located in an environment of critical cells or in an outside region of the network,*

$$\varphi_{\mathcal{A}} : \mathbb{R}^n \to \{T, F\},\ \varphi_{\mathcal{A}}(\xi) = \begin{cases} \mathbf{T} & : \ \text{if } (\upsilon_{\mathcal{A}_{CR}}(\xi)) \ \vee \ \neg\ (\upsilon_{\mathcal{A}}(\xi)) \\ \mathbf{F} & : \ \text{else.} \end{cases} \tag{2.23}$$

□

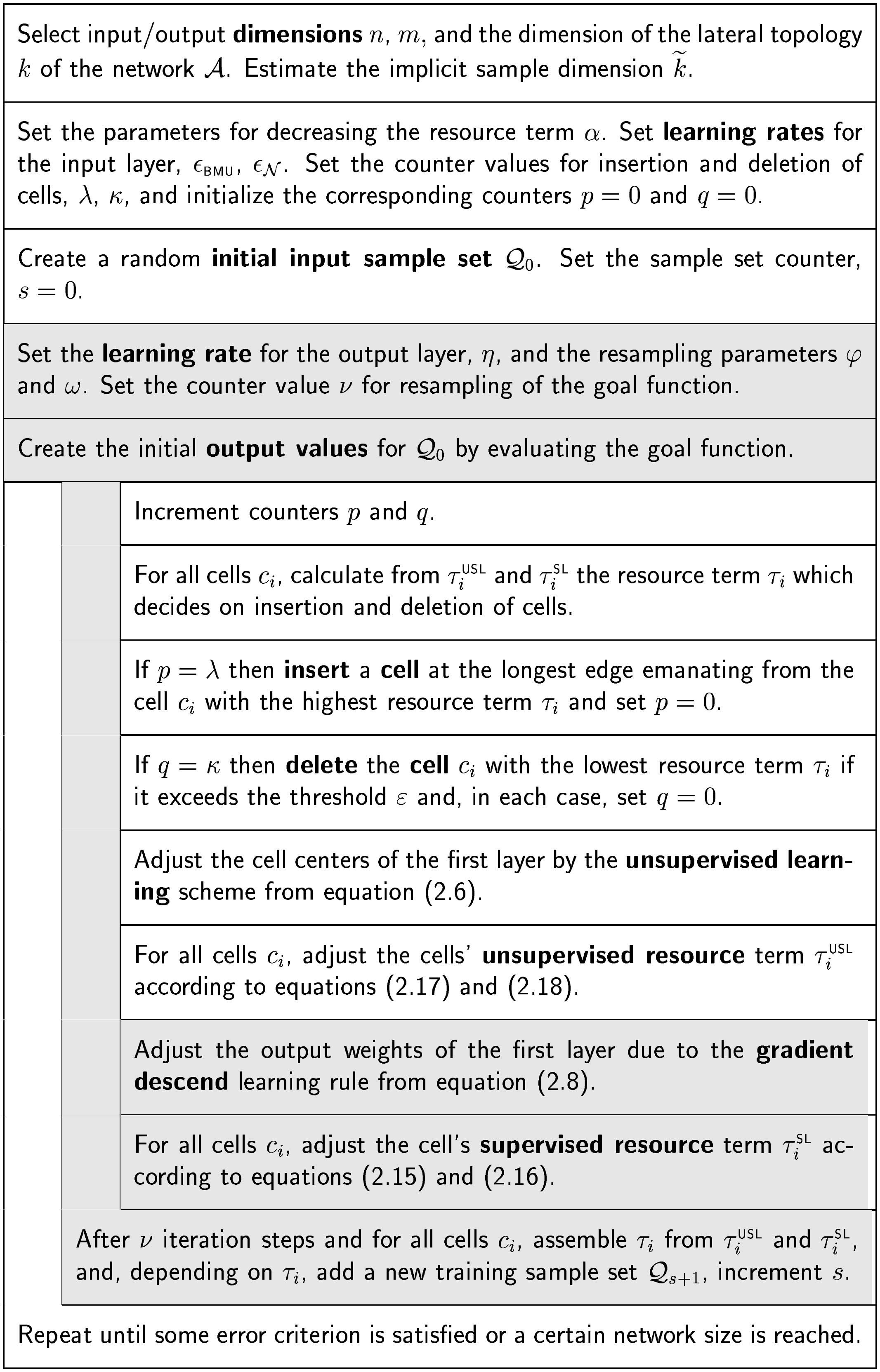
If equation (2.23) holds, the actual input has detected a critical region. The scanning step size, now, is decreased, and successively new samples are evaluated from the goal function — a new set of samples is generated and added to the existing training set.

Equation (2.23) designates a critical region, first, as such being represented mainly by critical cells, or second, being "represented by no cells". The first case simply denotes that the resource term in this region — for example, the approximation error — is extraordinarily high, the second fact detects regions where the input domain is defined, but where the ISGCS training did not create an instantiation by cells. The latter case typically appears in general self-organizing mapping approaches and happens at *boundary regions* of the input space. Due to the missing samples outside of the network, the sample distribution can be seen as artificially modified when applying the proposed learning scheme. This leads to a shrinking of the network apart from the boundaries. We avoid this drawback to a certain degree [8] by artificially increasing the distribution at these places, i.e., by creating additional samples at these outside regions.

The overall ISGCS algorithm is outlined in figure 2.9. For pure unsupervised learning, the gray boxes can be removed. See also figure 4.7.d-f as an example for a function approximation task of the two-dimensional goal function from figure 4.6.b.

2.6 Implementation Details and Complexity

On a first attempt, the presented training algorithm seems to be quite complex and dependent on the number of cells. In the following, we show that this is



```
Select input/output **dimensions** $n$, $m$, and the dimension of the lateral topology $k$ of the network $\mathcal{A}$. Estimate the implicit sample dimension $\widetilde{k}$.
Set the parameters for decreasing the resource term $\alpha$. Set **learning rates** for the input layer, $\epsilon_{\text{BMU}}$, $\epsilon_{\mathcal{N}}$. Set the counter values for insertion and deletion of cells, $\lambda$, $\kappa$, and initialize the corresponding counters $p = 0$ and $q = 0$.
Create a random **initial input sample set** $\mathcal{Q}_0$. Set the sample set counter, $s = 0$.
Set the **learning rate** for the output layer, $\eta$, and the resampling parameters $\varphi$ and $\omega$. Set the counter value $\nu$ for resampling of the goal function.
Create the initial **output values** for $\mathcal{Q}_0$ by evaluating the goal function.
        Increment counters $p$ and $q$.
        For all cells $c_i$, calculate from $\tau_i^{\text{USL}}$ and $\tau_i^{\text{SL}}$ the resource term $\tau_i$ which decides on insertion and deletion of cells.
        If $p = \lambda$ then **insert** a **cell** at the longest edge emanating from the cell $c_i$ with the highest resource term $\tau_i$ and set $p = 0$.
        If $q = \kappa$ then **delete** the **cell** $c_i$ with the lowest resource term $\tau_i$ if it exceeds the threshold $\varepsilon$ and, in each case, set $q = 0$.
        Adjust the cell centers of the first layer by the **unsupervised learning** scheme from equation (2.6).
        For all cells $c_i$, adjust the cells' **unsupervised resource** term $\tau_i^{\text{USL}}$ according to equations (2.17) and (2.18).
        Adjust the output weights of the first layer due to the **gradient descend** learning rule from equation (2.8).
        For all cells $c_i$, adjust the cell's **supervised resource** term $\tau_i^{\text{SL}}$ according to equations (2.15) and (2.16).
    After $\nu$ iteration steps and for all cells $c_i$, assemble $\tau_i$ from $\tau_i^{\text{USL}}$ and $\tau_i^{\text{SL}}$, and, depending on $\tau_i$, add a new training sample set $\mathcal{Q}_{s+1}$, increment $s$.
Repeat until some error criterion is satisfied or a certain network size is reached.
```

Figure 2.9: The complete ISGCS algorithm. The light boxes describe the ISGCS algorithm if pure clustering is demanded. Adding the gray boxes enables supervised learning additionally.

not the case, moreover, through utilizing *lateral locality of cells and samples* we can reach *constant complexity* for the cycle of one iteration.

Locality of Cells

Evaluating the whole network output has to be accomplished for the adaption of the output weights and the cells' resource terms. This influences the amount of time resources crucially, since, while training, it has to be executed every time a sample is presented to the network. Evaluation of the network output consists of evaluating and accumulating each of the unit activations weighted by the output weights (equations 2.2 and 2.3). This process is $O(z)$, with z the number of cells in the network.

To reduce this work we profit from the locality of each of the cells' Gaussians defined through the standard deviation, i.e., the distance of the direct neighbors (eq. 2.10). For further explanations we define the k-neighborhood of each cell.

Definition 8. *The* k-neighborhood $\mathcal{N}_i^k$ *of a cell* c_i *of an ISGCS* $\mathcal{A}$ *is the set of cells* $\{c_j : c_j \in \mathcal{A}\}$ *which can be reached from* c_i *on the shortest way by traversing not more than* k *edges of the network.*

In other words, the k-neighborhood of a cell c_i is the set of those cells which lie in a *topological distance* from c_i which is smaller than or equals k. Reaching a cell $c_j \in \mathcal{N}_i^k$ from c_i must be possible by passing not more than $k-1$ other cells. It holds $\mathcal{N}_i^0 = \{c_i\}$ and $\mathcal{N}_i^1 = \mathcal{N}_i$ (see page 23).

Since the value of a Gaussian decreases with its center's distance to the argument, the influence of Gaussians which lie in a large distance from the input vanish. Only the neighbors in the k-neighborhood of a cell have a significant contribution to the accumulation result and the others can be neglected by accepting a small error.

Thus, instead of calculating all cell's output activation, the BMU of a certain input sample is determined and only the activations of the cells in the k-neighborhood are accumulated.

Locality of Samples

Beside the evaluation of the network output, the search for the BMU is the second most time consuming operation since it also has to be done for every iteration step.

One characteristic feature of an ISGCS is the fact that a cell's position adapts slightly to a certain sample input. A sample which is presented to the network, again, is not likely to change its affiliation to a certain Voronoi volume very often. This feature is utilized by storing for each sample a pointer to the corresponding BMU. This pointer is used as a BMU estimate for the next time when this sample is selected for training. Based on this BMU, only its k-neighborhood is searched for a better matching cell instead of scanning the whole network.

Thus, through the locality of samples and cells, "touching" the network is a cell-local task, only depending on the number of cells in a cell's neighborhood, which is constant.

Own experiments for several example goal functions have shown that a depth of 3 for the evaluation of the Gaussians (i.e., only evaluating the cells of $\mathcal{N}^{k=3}_{\text{BMU}}$) creates an error which lies several orders in magnitude below the overall error.

For the search of the BMU, a depth of $k = 1$ suffices in most cases, for $k = 3$ the results are identical with a global search.

Chapter 3

Radiosity with Cell Structures

We follow up the application of the results of chapter 2, divided into three parts. First, an approximation model of the radiosity kernel is generated instantiated by an ISGCS [6], second, with the generated sample set, concurrently, a further growing cell structures network is trained by the end points of the sample set generated with the kernel net, and third, based on both network structures, the integration method required by the FEM is formulated.

3.1 Kernel Base (Light Flow Approximation)

Goal Function

Since we want to account for the whole flow of light, the goal function f to be approximated is the geometric kernel multiplied by the energy emitted from the patches. Recalling equation (1.1),

$$B(\mathbf{y}) = E(\mathbf{y}) + \int_S \underbrace{K(\mathbf{x},\mathbf{y})\, B(\mathbf{x})}_{\text{goal function } f \approx \boldsymbol{\Psi}} \; d\mathbf{x}, \tag{3.1}$$

the whole energy transfer will be approximated by an ISGCS $\boldsymbol{\Psi}$ which is defined in the following. Training sample rays are derived, first, from the geometry and the emittance of the light sources, second, after the first FEM computation, additionally, from its results (see section 3.4).

Approximation Model

The goal function $f : \mathbb{R}^6 \to \mathbb{R},\ f(\mathbf{x},\mathbf{y}) = K(\mathbf{x},\mathbf{y})B(\mathbf{x})$ is approximated by the network $\boldsymbol{\Psi}$,

$$\boldsymbol{\Psi}(\xi) \approx K(\mathbf{x},\mathbf{y})\, B(\mathbf{x}), \quad \text{with} \quad \xi = (x_1, x_2, x_3, y_1, y_2, y_3), \tag{3.2}$$

and $\boldsymbol{\Psi}$ is a four-dimensional ISGCS ($k = 4$) with a six-dimensional input layer and a one-dimensional output layer, $\boldsymbol{\Psi} : \mathbb{R}^6 \to \mathbb{R}$. $\boldsymbol{\Psi}$ is trained and evaluated by input 6-tuples defined from two points (ray) $\mathbf{x}$ and $\mathbf{y}$ lying on the surfaces of the scene definition. The lateral dimension of $k = 4$ is chosen due to the fact that the kernel is a four-dimensional function, i.e., a function of two points (a ray) on the two-dimensional surface space. The instantiation of the ISGCS (see section 2.2) is

$$\boldsymbol{\Psi}(\xi) = \sum_{m=1}^{z} v_m \, \Omega_m(r), \quad \text{with} \quad r = \|\mathbf{w}_m - \xi\|\,, \tag{3.3}$$

and z the number of cells in the network. The functions Ω_m are the RBFs of the first layer, v_m the one-dimensional output weight of a cell m, and $\mathbf{w}_m$ its 6-dimensional center. The matrix $\mathbf{V}$ in this case is just a vector $\mathbf{v} \in \mathbb{R}^z$, and each component v_i is the coefficient of the cell's support to the actual output. We reformulate equation (3.3) into matrix notation.

$$\begin{aligned} &\boldsymbol{\Psi}(\xi) = \mathbf{v}\,\boldsymbol{\Omega}(\mathbf{r}), \\ &\qquad \text{with} \quad \mathbf{r} = (r_1, r_2 \ldots r_z),\ r_m = \|\mathbf{w}_m - \xi\|\,,\ m = 1 \ldots z, \end{aligned} \tag{3.4}$$

and $\boldsymbol{\Omega} : \mathbb{R}^6 \to \mathbb{R}^z$ is a vector of the RBFs, and $\mathbf{r}$ a vector of distances between the input ξ and the according center $\mathbf{w}_m$ of the RBF m.

Intuitively, the kernel operator described in section 1.2 as a function operating on points $\mathbf{x}$ to deliver points $\mathbf{y}$, here, is assumed as a function with an input of two points and an output of the "operation strength". This is a slightly different but valid view on the meaning of the kernel operator.

The resulting approximation interpolates reference rays (RBFs) in a four-dimensional space. These rays can be seen similar to links from the HR approach, but their relation to the geometry is much weaker, since they are not developed by focusing on patches, but on the general approximation facilities of a set of arbitrary rays. Thus, displaying them in a three-dimensional space would not lighten one's imagination. Instead we refer to the *Flatland* [1] examples presented in chapter 4.1, where two-dimensional kernels are displayed as two-dimensional pictures with the reference rays marked.

[1] Flatland is the two-dimensional analogue of the three-dimensional space. It is commonly used to ease the understanding of relations in space by investigating its two-dimensional translation (see [26]).

3.2 Radiosity Mesh

Classical radiosity approaches are characterized by the following steps.

1. Define an orthonormal radiosity function base $N_i : \mathbb{R}^3 \to \mathbb{R},\ i = 1 \dots n$, of size n and the position of its component functions somewhere on the surfaces of the geometry, such that it enables an approximation of the radiosity by a linear combination $B \approx \sum_{i=1}^{n} b_i N_i$.

2. Transfer the energy between these base components by executing the radiosity kernel operator on the radiosity base, i.e., calculate the n^2 transfer coefficients $k_{ij} = \iint_{\mathcal{S}} N_i N_j \, d\mathbf{x} \, d\mathbf{y},\ i, j = 1 \dots n$, between the base components, and accumulate the energy on the surfaces through the accumulation of the base function coefficients, $\Delta b_i := \sum_{j=1}^{n} k_{ij} b_j,\ \forall i = 1 \dots n$.

3. Go to step 2 until the transferred energy falls below a certain threshold, or go to step 1 to adjust number or shape of the base functions according to some criterion defined by the actual solution — the coefficients b_j (multigridding).

In this work, instead of focusing on the approximation of the radiosity function, we start with the development of the kernel approximation model $\mathbf{\Psi}$ (see section 3.1), and then, a suitable radiosity base is derived to perform the integration tasks.

Although there is *some* representation of the kernel in the kind of a function base (the RBFs), this base has nothing in common with the usual classical representation of a kernel, i.e., it is not a linear combination of the tensor products of the radiosity base components. Thus, since the kernel model is not able being interpreted as a tensor product base, we need to invent an independent base which exists on the surface domain. On the one hand, it appears self-evident to derive this base from the kernel model topology directly, for example, by placing its components at the start and end points of the kernel cell centers. But, on the other hand, this kind of arbitrary definition would lead to a function base which generally is unfavorable concerning its approximation facilities.

Thus, we utilize the growing cell structures' facility of finding adequate distributions of reference objects and create *another* two-dimensional growing cell structures network, the *shading network* (SN). It is trained by points lying on the surface geometry and it must be instantiated by a two-dimensional lateral topology. Its simplices are triangles ($k = 2$) and establish a homogeneous global triangulation over the geometry, driven by the distribution of sample points. The single cells can be seen as reference points which "interpolate" the shape of the geometry. The shading network is trained and evolves concurrently to the training of the kernel network and guarantees the geometry independence and the iterative character of the complete approach.

The shading network is trained in an unsupervised manner, and additionally, a three-dimensional output layer learns the normal vector of the presented point. This is required for the integration tasks to be explained in chapter 3. The training samples for the shading network are derived directly from the kernel samples, i.e., for a sample ray like $(x_1, x_2, x_3, y_1, y_2, y_3)$, the shading network sample is the vector (y_1, y_2, y_3). This ensures that the shading samples and with it the final network account for the light flow, i.e., the distribution of base centers suits to the distribution of the kernel samples, moreover, inherits its efficiency.

Although the base components derived from the shading network are trained by points from the geometry definition, they only *approximate* the original surface domain $\mathcal{S}$. Thus, besides the original scene definition $\mathcal{S}$, we denote the approximate geometry $\hat{\mathcal{S}}$, instantiated by the areas of the single simplices (triangles) $\hat{\mathcal{S}}_k : k = 1 \dots \hat{n}$ of the shading network. See figure 4.12.b for an example of a shading network.

The next section describes the generation of a linear function base attached to the network mesh.

3.3 Radiosity Basis Functions

Consider the common definition of the radiosity integral equation

$$B(\mathbf{y}) \quad = \quad E(\mathbf{y}) + \int_{\mathcal{S}} K(\mathbf{x}, \mathbf{y})\, B(\mathbf{x})\; d\mathbf{x},$$

reformulated by replacing the ANN kernel approximation model,

$$\begin{aligned} B(\mathbf{y}) \quad &= \quad E(\mathbf{y}) + \int_{\mathcal{S}} \mathbf{\Psi}(\xi)\; d\mathbf{x} \qquad (3.5) \\ &\qquad \text{with} \quad \xi = (x_1, x_2, x_3, y_1, y_2, y_3). \end{aligned}$$

In the following, the exact notion of the arguments of $\mathbf{\Psi}$ is ignored, for example $\mathbf{\Psi}(\xi) \equiv \mathbf{\Psi}(\mathbf{x}, \mathbf{y})$. We assume the radiosity base being an orthogonal vector $\mathbf{N} = (N_1, N_2 \dots N_{\hat{n}})$ of base components N_i. The components are defined focusing on the mesh which is generated in the preceeding section. The detailed algorithm follows in the next section.

In equation (3.5), there is no explicit representation of $B(\mathbf{x})$, which, for example, could be transformed into an approximation through a linear function base, and virtually, the equation even differs from the general shape of a Fredholm integral equation.

For the discretization of equation (3.5), similar to section 1.3, we first apply the linear projection operator $[\langle \mathbf{N} | \cdot \rangle]$ which leads to a "nearly discrete"[2] system,

$$\mathbf{b} = \mathbf{e} + \left[\left\langle \mathbf{N}(\mathbf{y}) \middle| \int_{\mathcal{S}} \boldsymbol{\Psi}(\mathbf{x}, \mathbf{y})\, d\mathbf{x} \right\rangle \right], \tag{3.6}$$

with lines

$$\begin{aligned} b_i &= e_i + \left\langle N_i(\mathbf{y}) \middle| \int_{\mathcal{S}} \boldsymbol{\Psi}(\mathbf{x}, \mathbf{y})\, d\mathbf{x} \right\rangle \\ &= e_i + \int_{\hat{\mathcal{S}}} N_i(\mathbf{y}) \int_{\mathcal{S}} \boldsymbol{\Psi}(\mathbf{x}, \mathbf{y})\, d\mathbf{x}\, d\mathbf{y}, \\ & \qquad\qquad i = 1 \ldots \hat{n}. \end{aligned} \tag{3.7}$$

It remains a discretization of the domain $\{\mathbf{x}\}$ from which the outgoing radiosity has to be accumulated by the integration operator approximation $\boldsymbol{\Psi}$.

One can think of the fact that in equation (3.7) there already exists some kind of discretization — by the kernel base components. Replacing equation (3.3) for $\boldsymbol{\Psi}$ in equation (3.7) delivers

$$b_i = e_i \int_{\hat{\mathcal{S}}} N_i(\mathbf{y}) \int_{\mathcal{S}} \sum_{m=1}^{z} v_m\, \Omega_m(\mathbf{x}, \mathbf{y})\, d\mathbf{x}\, d\mathbf{y},\ i = 1 \ldots \hat{n},$$

and a system like

$$\mathbf{b} = \mathbf{e} + \mathbf{K}'\, \mathbf{v},$$

with the *non-quadratic transfer array* $\mathbf{K}' \in \mathbb{R}^z \times \mathbb{R}^{\hat{n}}$ and coefficients $\mathbf{K}' = \{k'_{ij}\}$, $i = 1 \ldots z$, $j = 1 \ldots \hat{n}$ defined like

$$k'_{ij} = \int_{\hat{\mathcal{S}}} N_i(\mathbf{y}) \int_{\mathcal{S}} \Omega_m(\mathbf{x}, \mathbf{y})\, d\mathbf{x}\, d\mathbf{y}. \tag{3.8}$$

We will not investigate this method due to two issues. First, the integration operation (eq. 3.8) would be difficult, since the integration domain of the outgoing radiosity (points $\mathbf{x}$ on $\mathcal{S}$) must be accessed through accessing the surface geometry directly. This is what we wanted to avoid from the beginning of this work[3]. Second, in the way described above, integrating equation (3.8) needs to be accomplished numerically.

[2] Only B on the left side of the equation and the emittances are converted to coefficients. Here, the new integration domain is the shading network $\hat{\mathcal{S}}$ leading to different integration domains in equation (3.7).

[3] This is also the reason for using the shading network to define a radiosity base even at points $\mathbf{y}$ (equations (3.6) and (3.7)).

In our solution, we suggest describing the surface domain regarded from points $\mathbf{x}$ only by using the shading network. By the facility of the basis components of having a local support (the base will be discussed in more detail in later sections) and by using Gaussians, we are able to find an analytical solution of equation (3.8) and thus avoid its numerical evaluation.

In the following, we define the second base $\bar{\mathbf{N}}$ while keeping it separate from the definition of $\mathbf{N}$. This degree of freedom is lifted in later paragraphs.

We assume the orthonormal base function vector $\bar{\mathbf{N}} = (\bar{N}_1, \bar{N}_2 \dots \bar{N}_{\bar{n}})$ and a vector $\bar{\mathbf{b}} = (\bar{b}_1, \bar{b}_2 \dots \bar{b}_{\bar{n}})$ of $\bar{n}$ coefficients $\bar{b}_j$ for approximating a *void* function $\bar{B} : \mathbb{R}^3 \to \mathbb{R}$ with $\bar{B}(\mathbf{x}) \equiv 1$. In other words, we add the base in equation (3.5) at points $\mathbf{x}$. The coefficients of $\bar{\mathbf{b}}$ serve for trimming the components' weights in a way that the function base approximates the value one.

Adding $\bar{B}$ into equation (3.7) delivers

$$\begin{aligned} b_i &= e_i + \int_{\hat{\mathcal{S}}} N_i(\mathbf{y}) \int_{\hat{\mathcal{S}}} \mathbf{\Psi}(\mathbf{x},\mathbf{y}) \bar{B}(\mathbf{x})\, d\mathbf{x}\, d\mathbf{y} && (3.9) \\ &= e_i + \int_{\hat{\mathcal{S}}} N_i(\mathbf{y}) \int_{\hat{\mathcal{S}}} \mathbf{\Psi}(\mathbf{x},\mathbf{y}) \sum_{j=1}^{\bar{n}} \bar{b}_j \bar{N}_j \, d\mathbf{x}\, d\mathbf{y} \\ &= e_i + \int_{\hat{\mathcal{S}}} N_i(\mathbf{y}) \int_{\hat{\mathcal{S}}} \mathbf{\Psi}(\mathbf{x},\mathbf{y})\, \bar{\mathbf{b}}\bar{\mathbf{N}}\, d\mathbf{x}\, d\mathbf{y} && (3.10) \\ & \qquad i = 1 \dots \hat{n}. \end{aligned}$$

Due to $\mathbf{\Psi}(\mathbf{x},\mathbf{y}) \approx KB$, equation (3.10) remains valid if $\bar{B}(\mathbf{x}) = 1$ holds for all $\mathbf{x} \in \hat{\mathcal{S}}$. If $\bar{B}(\mathbf{x})$ only approximates a value of one, then a certain error is introduced[4]. Equation (3.10) delivers a scheme to completely transform equation (3.6) into a discrete form like

$$\mathbf{b} = \mathbf{e} + \mathbf{H}\,\bar{\mathbf{b}}, \tag{3.11}$$

with a matrix $\mathbf{H} \in \mathbb{R}^{\bar{n}} \times \mathbb{R}^{\hat{n}}$, $\mathbf{H} = \{h_{ij}\}$, $i, = 1 \dots \hat{n}$, $j = 1 \dots \bar{n}$ of coefficients h_{ij} defined like (eq. 3.10)

$$h_{ij} = \int_{\hat{\mathcal{S}}} N_i(\mathbf{y}) \int_{\hat{\mathcal{S}}} \mathbf{\Psi}(\mathbf{x},\mathbf{y}) \bar{N}_j(\mathbf{x})\, d\mathbf{x}\, d\mathbf{y}. \tag{3.12}$$

Obviously, this is not equivalent to the classical definition of a discrete linear radiosity system. The differences are, on the one hand, the vector $\bar{\mathbf{b}}$ with fixed predefined components forcing $\bar{B}(\mathbf{x}) = 1$ and defined by $\bar{b}_j = \int_{\hat{\mathcal{S}}} \bar{N}_j(\mathbf{x})\, d\mathbf{x}$, $j = 1 \dots \bar{n}$, and on the other hand, the second base $\bar{\mathbf{N}}$ is denoted differently from $\mathbf{N}$ reinforcing that the base components not necessarily need to be identical[5].

[4] to be discussed later

[5] Consider, we are even not able to write the linear system in the classical notation like $(\mathbf{I} - \mathbf{K})\mathbf{b} = \mathbf{e}$.

In classical radiosity approaches, using two different function bases would expose disadvantages, since there is an additional base transformation needed for transferring coefficients of the approximation of $B(\mathbf{y})$ on the right side of the radiosity equation to coefficients of the representation of $B(\mathbf{x})$ on the left side.

In contrast to classical approaches, in this work, the conversion of the coefficients of the approximation of $B(\mathbf{y})$ into single sample values is required only to get new training values for the kernel network. These values are independent from a particular function base, which relieves from any base transformations. It delivers a new degree of freedom in the definition of radiosity bases, but nevertheless, for an efficient integration algorithm, later in this work, one base is used for both representations of the radiosity, $B(\mathbf{x})$ and $B(\mathbf{y})$.

The next two paragraphs discuss the convergence of the whole approach focusing on the mentioned difference between classical approaches if compared to this work. Then, in section 3.6, two instantiations of a base are discussed, a constant, more common base and a base created by a set of Gaussians.

3.4 Iterating the Linear System

The preceeding paragraph defines how to operate the kernel network once to get radiosity coefficients $\mathbf{b}$ related to a base $\mathbf{N}$. Feeding the computed radiosity back into the system to simulate further propagations of light will obviously be accomplished implicitly through training the kernel network by values KB. Each time a new approximation of the radiosity is calculated and represented through the coefficients $\mathbf{b}$, the training samples are adjusted by evaluating the approximation $\mathbf{bN}$ at the start points of the sample rays. The following training of the kernel network incrementally adjusts the approximation to match the actual sample set.

In other words, the whole process of finding the solution of the radiosity (see also figure 1.2) is centered around training both networks, where the values KB are adjusted through single evaluations of equation (3.11). Concurrently, the shading network topology evolves, since it depends on the distribution of the kernel samples, which changes through the resampling processes of the ANN training. Each time a single integration operation is triggered, first, a new base is generated regarding the actual shape of the shading network, and then, the integration of the transfer coefficients and the accumulation of the result at the vector $\mathbf{b}$ is accomplished.

This approach is quite different to usual iteration techniques. We investigate this property in detail in section 3.5. Now, the description of its algorithmical realization follows.

Consider the classical finite Neumann series (see equation (1.3)) with k elements,

$$\mathbf{b}^{(k)} = \mathbf{e} + \mathbf{K}^{(1)}\mathbf{e} + \mathbf{K}^{(2)}\mathbf{e} + ... + \mathbf{K}^{(k)}\mathbf{e}. \tag{3.13}$$

The k^{th} element of the series can be seen as the emitted energy which is propagated k-times through the geometry. The final result is the sum of all these bounces on the surfaces. Practical issues of iterating the radiosity equation can be represented in a more convenient way by

$$\mathbf{b}^{(k)} = \mathbf{K}(\mathbf{K} \cdots k \cdots (\mathbf{K}(\mathbf{K}\mathbf{e} + \mathbf{e}) + \mathbf{e}) \cdots k \cdots + \mathbf{e}) + \mathbf{e} \tag{3.14}$$

which directs to the following algorithm for adjusting the kernel training samples (see also figure 1.2).

An existing set of samples $\{\xi_i \in \mathbb{R}^6 = (\mathbf{x}_i, \mathbf{y}_i),\ i = 1...\}$ is assumed with $\mathbf{x}_i \in \mathbb{R}^3$ and $\mathbf{y}_i \in \mathbb{R}^3$ are start and end points of the i^{th} sample (reference ray), respectively. Each of them has an *actual radiosity value* $s_i^{(k)}$ attached, which contains the most recently computed radiosities $B^{(k)}(\mathbf{x}_i) + E(\mathbf{x}_i)$ emitted from the point $\mathbf{x}_i$ after the k^{th} FEM integration. The term $s_i^{(k)}$ multiplied with the geometry term $K(\mathbf{x}_i, \mathbf{y}_i)$ is the actual training value for the network output. Additionally, each sample has stored the emittance $E(\mathbf{x}_i)$ to access it when recalculating $B^{(k+1)}$ after another FEM iteration step, or in cases where new sample sets are created. See an overview of these aspects in figure 3.1 which outlines the complete iteration algorithm.

Updating the values $s_i^{(k)}$ might be seen as "base transform from $\mathbf{N}$ to points". The reason for creating the base, instead of integrating the kernel network directly onto the sample start points $\mathbf{x}_i$, is the fact that the integration is not required for each single sample value but only for each component of the function base with much fewer elements. Thus, just evaluating $B^{(k)}$ if an actual $s_i^{(k)}$ must be known (after the FEM or after the resampling mechanism of the ISGCS method) needs far less amount of computing resources than the integration of the whole environment through the kernel network would require.

```
Set $k = 0$. For all samples $(\mathbf{x}_i, \mathbf{y}_i)$, set $s_i^{(0)} = B^{(0)}(\mathbf{x}_i) = E(\mathbf{x}_i)$.
    Repeat training of the kernel network with the goal value $K(\mathbf{x}_i, \mathbf{y}_i) \cdot s_i^{(k)}$
    (eq. 1.3) (and repeat training of the shading network with $\mathbf{x}_i$ and the normal
    at $\mathbf{x}_i$) until one integration of the FEM is triggered.
    Calculate a new approximation $B^{(k+1)}$ by executing the integration like
    described in section 3.2.
    For all samples, set the new emission $s_i^{(k+1)} = B^{(k+1)}(\mathbf{x}_i) + E(\mathbf{x}_i)$.
    Increment $k$.
Repeat until the discretization of the geometry reaches a predefined size limit.
```

Figure 3.1: Complete algorithm for iterating the modified radiosity integral equation of this work (propagating energy through the kernel network), based on adjusting the emittances $s_i^{(k)}$ of the sample values (see figure 1.2).

3.5 Convergence

Convergence and accuracy of general Fredholm integral equations have widely been investigated in the literature (for example, see [54]). In [36] error bounds for the specific case of radiosity are analyzed, whereas Arvo et al. [2] have studied this issue for general global illumination approaches. There, it has been stated that an error bound for a radiosity system like

$$(\mathbf{I} - \mathbf{K})\mathbf{b} = \mathbf{e}, \tag{3.15}$$

is defined by the sum of separate error sources from which we, in the following, outline those three which are crucial for this work. First, the *discretization error* arises from the projection of B to a finite linear function space, second, the *perturbation error* arises from imprecisely evaluating the integral operator $\mathbf{K}$, which perturbs the whole linear system, and third, the *computation error* comes from those elements of the Neumann series, which are not evaluated due to lacking computational resources (i.e., time (for example in PR), or memory (for example in CR, HR)).

In this work, we have developed a different integral equation (eq. 3.11). To profit, nevertheless, from the results of [2], first, we explain where the two differences arise, and second, we show that the equation, virtually, is identical with the classical one.

1. Consider the developed radiosity integral equation

$$\begin{aligned} B(\mathbf{y}) &= E(\mathbf{y}) + \int_{\hat{S}} \mathbf{\Psi}(\xi)\, d\mathbf{x} \\ &= E(\mathbf{y}) + \int_{\hat{S}} [\tilde{K}(\mathbf{x},\mathbf{y})\tilde{B}(\mathbf{x})]\, \bar{B}(\mathbf{x})\, d\mathbf{x}. \end{aligned} \tag{3.16}$$

   $\tilde{K}$ and $\tilde{B}$ are the explicitly written content of the approximation kernel network $\mathbf{\Psi}$. On the one hand, a function $\bar{B}(\xi) \equiv 1$ has been introduced to accomplish efficient integration operations, on the other hand, since $\bar{B}$ is represented by its approximation through components of $\mathbf{N}$[6], it introduces an additional perturbation of $\tilde{B}$.

2. Virtually, the presented algorithm looks in a way like it would never calculate more than one element of the Neumann series, at all. This might be derived from the fact that, since the integration results are used to retrain the networks, this also implies a change of both network topologies, and thus, a change of the radiosity base $\mathbf{N}$ for each following FEM computation. Thus, in fact, the discrete linear system (eq. 3.11) is evaluated for a certain function base only once.

In order to account for the first point from above we reformulate equation (3.16) to

$$B(\mathbf{y}) = E(\mathbf{y}) + \int_{\hat{S}} \tilde{K}(\mathbf{x},\mathbf{y})\, \bar{B}(\mathbf{x})\tilde{B}(\mathbf{x})\, d\mathbf{x} \tag{3.17}$$

which delivers a linear system like in equation (3.15) but with different transfer coefficients,

$$k_{ij} = \int_{\hat{S}} N_i(\mathbf{y}) \int_{\hat{S}} \tilde{K}(\mathbf{x},\mathbf{y})\bar{b}_j N_j(\mathbf{x})\, d\mathbf{x}\, d\mathbf{y}. \tag{3.18}$$

In equation (3.17), $\tilde{B}$ is taken out of the closed network approximation and its approximation coefficients are used like in the usual linear system. This is a valid procedure due to the fact that even $\tilde{B}$ is calculated through the base $\mathbf{N}$, and thus, its re-projection can be accomplished without loss of accuracy. With equation (3.18), we can rewrite equation (3.16) in the common form (equation (3.15)), and thus, eliminate the first difference (point 1) from above. The term $\bar{B}$ is included at the transfer coefficients directing the view from a perturbed radiosity to a perturbed transfer coefficient. This reduces the original problem to just an additional perturbation error (like defined in [2]) whose amount is constant for a certain base and which does not extend the given error bounds essentially.

[6] Note, from this point, we assume the bases $\mathbf{N}$ and $\bar{\mathbf{N}}$ as being identical.

Regarding the second point from above, its elimination is accomplished in a more intuitive way. The radiosity error bounds have already been defined in [2] for a static base $\mathbf{N}$ — a static instantiation of $\mathbf{K}$ and a constant approximation facility of $\mathbf{N}$. If we assume that our system (eq. 3.15) even has a static base but simply increases the approximation facility of $\mathbf{N}$ and approximation accuracy of $\mathbf{K}$ during the network training, then we can state, while ignoring the change of the function base, that this work's error is at least smaller than that of a classical static linear system. In other words, changing the accuracy of the kernel operator during training does not affect the error bounds formulated in [2].

3.6 Gaussian Radiosity Base

Constant Base Components

One possible choice of a suitable base which is organized regarding the shading network topology could be defined through constant components $\bar{N}_j : \mathbb{R}^3 \to \mathbb{R}$,

$$\bar{N}_j(\mathbf{x}) = \begin{cases} 1 & : \text{ if } \mathbf{x} \in \hat{\mathcal{S}}_j \\ 0 & : \text{ else.} \end{cases} \tag{3.19}$$

In equation (3.19) separate box function components are defined, which support each triangle $\hat{\mathcal{S}}_j$, $j = 1 \ldots \hat{n}$ of the shading network. The base is orthogonal regarding the inner product $\langle \bar{N}_x | \bar{N}_y \rangle = \int_{\hat{\mathcal{S}}} \bar{N}_x \cdot \bar{N}_y \; d\mathbf{x}$ of two arbitrary base components $\bar{N}_x$ and $\bar{N}_y$, and delivers an average radiosity value which is constant over the area of a separate triangle — similar to base definitions in CR, PR, and HR.

Gaussian Base Components

The point of this work is the usage of Gaussian base components, like they are suggested in the general ISGCS approximation model and like we have been using them already in the kernel approximation model. On the one hand, this ensures its applicability as a function base if attached to a growing cell structures network. On the other hand, the next paragraphs will show how the integration operations to derive the inner product and the transfer coefficients of such a base will profit, i.e., a formula is derived, which allows for solving the integrals analytically. This frees from an expensive numerical integration like it is needed in almost all classical radiosity approaches.

In the following, together with a suitable inner product, the Gaussian base is defined and its validity is proved.

The linear base components used in this work are a set of Gaussian functions $\mathcal{P} = \{\Lambda_k : \mathbb{R}^3 \to \mathbb{R}\}, \;\; k = 1 \dots \hat{z}$ with centers $\hat{\mathbf{w}}_k \in \mathbb{R}^3$ at the cell center coordinates of the shading network,

$$\Lambda_k(\mathbf{x}) = e^{-\hat{d_k}^{-2} s^2}, \quad \text{with} \quad s = \|\mathbf{x} - \hat{\mathbf{w}}_k\|, \, \mathbf{x} \in \mathbb{R}^3. \tag{3.20}$$

One Gaussian component, Λ_k, is centered at a common vertex of a set of adjacent triangles. The $\hat{d_k}$ are assigned the average distance of the neighboring cells in the network topology. Consider, $\mathcal{P}$ does not play the same role as the N_i from the preceeding section. Due to the missing orthogonality, the $\mathcal{P}$ are just the components from which, later in this work, an orthonormal base $\mathbf{N}$ is built — each N_i defined by a linear combination of the Λ_i. The orthogonal base, then, consists of base components each defined by a set of Gaussians with according coefficients. These coefficients account for the *overlap* of neighboring Gaussians.[7]

Before testing the validity of the proposed radiosity base, regard the following notes concerning the base's unusual properties.

The connection between the geometry surfaces (instantiated by the shading network) and the components Λ_k is quite unusual if compared to classical approaches. There is no unique domain belonging to each of them. Thus, handling one Λ_k in the following integration operations (sections 3.6.1 and 3.8) means accounting for each of the surrounding simplices explicitly due to its specific embedding in the three-dimensional space.

The Gaussians Λ_k are defined for any choice of dimensionality, only basing on some distance from its n-dimensional center. Even in our case, on the one hand, the Gaussians are valid at all points in the three-dimensional space, on the other hand, they are regarded as two-dimensional functions on the two-dimensional shading network domain.

Validity of the Gaussian Function Base

We have been proposing the set $\mathcal{P}$ of functions $\Lambda_k, \; k = 1 \dots \hat{z}$ on the $\hat{z}$ vertices of the shading network and assume that it is suitable to span the space of possible radiosity solutions on the geometry $\hat{\mathcal{S}}$ defined by $\hat{n}$ triangles. In the following, we derive an orthonormal radiosity function base $\{N_k, k = 1 \dots \hat{z}\}$ by a linear combination of elements from $\mathcal{P}$ and prove the mathematical validity of such a base. Since the base is a linear subspace of the space of all linear function bases, it remains to prove existence and validity of an inner product and of a norm, and finally, linear dependency are derived and proven.

[7] The single Gaussians have infinite expansion. Thus, the term 'neighboring' relates to the fact that the influence (concerning the inner product) of components which lie far from each other can be ignored.

For comprehensiveness, in this work, we omit the prove of the numerical validity of the proposed base, like stability conditions, for example. Instead, we rely on work from [18], in which it has been proven that a Gaussian base on a growing cell structures network is an efficient approximation model. See also [46, 48, 49].

Inner Product and Norm

We assume three elements x, y, z of the set $\mathcal{M}$, the space of linear function bases created from linear combinations of Gaussians $\mathit{\Lambda}_k$ over scalars $a_k, b_k, c_k \in \mathbb{R}$, $k = 1 \ldots \hat{z}$, respectively. For example, $x \in \mathcal{M} \rightarrow \exists \{a_k,\ k = 1 \ldots \hat{z}\} : x = a_1 \mathit{\Lambda}_1 + a_2 \mathit{\Lambda}_2 + \ \ldots a_{\hat{z}} \mathit{\Lambda}_{\hat{z}}$.

Defining a *norm* $\|\cdot\|$ for a linear system can be considered as associating a "size" to each of the elements. The inner product $\langle\cdot|\cdot\rangle$ defines a kind of "influence" between a pair of elements of $\mathcal{M}$. We adopt the common definition of the inner product of two functions x and y,

$$\langle x|y\rangle = \int_{\hat{\mathcal{S}}} x \cdot y \, d\mathbf{x}, \tag{3.21}$$

with $\mathbf{x}$ points on the triangles $\hat{\mathcal{S}}$ of the shading network, and equation (3.21) must satisfy the following conditions.

I_1: $\langle x|y\rangle = \langle y|x\rangle$.

I_2: $\langle(\alpha x + \beta y)|z\rangle = \alpha \langle x|z\rangle + \beta \langle y|z\rangle$.

I_3: $\langle x|x\rangle \geq 0,\ \forall x \in \mathcal{M}$.

I_4: $\langle x|x\rangle = 0$ if and only if $x = \emptyset$.

I_5: The norm is defined as $\|x\| = \langle x|x\rangle^{\frac{1}{2}}$.

Condition I_5 leads to the definition of the norm of an element x,

$$\|x\| = \sqrt{\int_{\hat{\mathcal{S}}} x^2 \, d\mathbf{x}}. \tag{3.22}$$

With $x, y, z \in \mathcal{M}$ and two scalars $\alpha, \beta \in \mathbb{R}$, the following must hold.

N_1: $\|x\| \geq 0$ and $\|x\| = 0$ if and only if $x = \emptyset$.

N_2: $\|\alpha x\| = |\alpha| \|x\|$.

N_3: $\|x+y\| \leq \|x\| + \|y\|$.

Equations (3.21) and (3.22) can be rewritten like

$$\begin{aligned}\langle x|y\rangle = \int_{\hat{S}} x \cdot y \, d\mathbf{x} &= \int_{\hat{S}} \left(\sum_{i=1}^{\hat{z}} a_i \Lambda_i\right)\left(\sum_{j=1}^{\hat{z}} b_j \Lambda_j\right) d\mathbf{x} \\ &= \sum_{1\leq i,j\leq \hat{z}} a_i b_j \int_{\hat{S}} \Lambda_i \Lambda_j \, d\mathbf{x}, \qquad (3.23)\end{aligned}$$

and

$$\begin{aligned}\|x\|^2 = \int_{\hat{S}} x^2 \, d\mathbf{x} &= \int_{\hat{S}} \left(\sum_{i=1}^{\hat{z}} a_i \Lambda_i\right)^2 d\mathbf{x} \\ &= \sum_{i=1}^{\hat{z}} a_i^2 \int_{\hat{S}} \Lambda_i^2 \, d\mathbf{x} + 2 \sum_{\substack{1\leq i,j\leq \hat{z} \\ j>i}} a_i a_j \int_{\hat{S}} \Lambda_i \Lambda_j \, d\mathbf{x}. \qquad (3.24)\end{aligned}$$

In section 3.6.1, the computation of the inner product of two Gaussians Λ_i and Λ_j[8] is described. For the following, we anticipate the existence of this result and proceed with the proof of the linear independency of the chosen base.

Linear Independency

Consider a set of elements $\{x_1, x_2, \ldots, x_{\hat{z}}\} : x_k \in \mathcal{M}, \forall k = 1 \ldots \hat{z}$ with a set of vectors $\{\mathbf{a}_1, \mathbf{a}_2 \ldots \mathbf{a}_{\hat{z}}\}, \mathbf{a}_k \in \mathbb{R}^{\hat{z}}, \forall k = 1 \ldots \hat{z}$ where $x_k = \sum_{i=1}^{\hat{z}} a_{ki} \Lambda_i$. With scalars $\alpha_1, \alpha_2, \ldots \alpha_{\hat{z}}$, the base $x_1, x_2, \ldots x_{\hat{z}}$ is linear independent if

$$\alpha_1 x_1 + \alpha_2 x_2 + \ldots + \alpha_{\hat{z}} x_{\hat{z}} = \emptyset \qquad (3.25)$$

implies that $\alpha_1 = \alpha_1 = \ldots = \alpha_{\hat{z}} = 0$. In other words, the x_k are *linear dependent* if equation (3.25) holds with at least one coefficient being different from zero. Linear independency is the basic demand for a stable algorithm for the development of approximation models. In the following, we prove the linear independency of the chosen base.

If the x_k, $k = 1 \ldots \hat{z}$ are linear independent, then this holds also for a base $e_k \in \mathcal{M} : e_k = \alpha_k \sum_{i=1}^{\hat{z}} \delta_{ik} \Lambda_i$, $k = 1 \ldots \hat{z}$, with the Kronecker delta δ_{ik}. Linear independency of a combination of Gaussians with equal expansion $\hat{d}$ has been examined, for example, in [42]. In the case of this work, the $\hat{d}_i$, $i = 1 \ldots \hat{z}$, differ from each other and we propose a proof based on a conversion

[8] i may equal j for the calculation of the norm.

of the approximation term into a polynomial form over the Gaussians' center coordinates. It is assumed that $\Delta_i \in \mathbb{Q}^+$, $\Delta_i \equiv \hat{d}_i^{-2}$, $\mathbf{x}_i, \mathbf{x}, \mathbf{r} \in \mathbb{Q}^2$, and $\mathbf{r} \equiv (\mathbf{x} - \mathbf{x}_i)^2$.

Due to comprehensiveness, the proof follows for two-dimensional spaces, but nevertheless, it holds even for higher dimensions.

$$\begin{aligned}
\sum_{i=1}^{\hat{z}} \alpha_i \varLambda_i &= \sum_{i=1}^{\hat{z}} \alpha_i e^{-\Delta_i ||\mathbf{x}-\mathbf{x}_i||^2} \qquad (3.26)\\
&= \sum_{i=0}^{n} \alpha_i \, e^{-\Delta_i (r_1^2 + r_2^2)}\\
&= \sum_{i=0}^{n} \alpha_i \, e^{-\Delta_i (x_1^2 - 2x_1 x_{i,1} + x_{i,1}^2 + x_2^2 - 2x_2 x_{i,2} + x_{i,2}^2)}\\
&= \sum_{i=0}^{n} \alpha_i \, \left(e^{x_1^2}\right)^{-\Delta_i} \left(e^{x_1}\right)^{\Delta_i 2 x_{i,1}} \, e^{-\Delta_i x_{i,1}^2} \left(e^{x_2^2}\right)^{-\Delta_i} \left(e^{x_2}\right)^{\Delta_i 2 x_{i,2}} \, e^{-\Delta_i x_{i,2}^2}.
\end{aligned}$$

With the substitutions

$$\begin{aligned}
v &= e^{\log^2 u} \qquad (3.27)\\
t &= e^{\log^2 s} \qquad (3.28)
\end{aligned}$$

and $a_i = \alpha_i \, e^{-\Delta_i x_{i,1}^2} \, e^{-\Delta_i x_{i,2}^2}$, equation (3.26) can be written like

$$\sum_{i=1}^{\hat{z}} \alpha_i \varLambda_i = \sum_{i=0}^{n} a_i \, v^{-\Delta_i} \, u^{\Delta_i 2 x_{i,1}} \, t^{-\Delta_i} \, s^{\Delta_i 2 x_{i,2}}. \qquad (3.29)$$

The above assumptions of the coordinates and expansions being elements of $\mathbb{Q}$ allow for the transformation of the exponents in equation (3.29) forming terms with common denominators $\alpha'', \beta'', \gamma'', \delta''$, and its related nominators for each Gaussian $\alpha_i', \beta_i', \gamma_i', \delta_i'$,

$$\sum_{i=1}^{\hat{z}} \alpha_i \varLambda_i = \sum_{i=0}^{n} a_i \, v^{\frac{\alpha_i'}{\alpha''}} \, u^{\frac{\beta_i'}{\beta''}} \, t^{\frac{\gamma_i'}{\gamma''}} \, s^{\frac{\delta_i'}{\delta''}},$$

which exposes a polynomial with the substitutions $\tilde{v} = v^{\frac{1}{\alpha''}}$, $\tilde{u} = u^{\frac{1}{\beta''}}$, $\tilde{t} = t^{\frac{1}{\gamma''}}$, $\tilde{s} = v^{\frac{1}{\delta''}}$ like

$$\sum_{i=1}^{\hat{z}} \alpha_i \varLambda_i = \sum_{i=0}^{n} a_i \, \tilde{v}^{\alpha_i'} \, \tilde{u}^{\beta_i'} \, \tilde{t}^{\gamma_i'} \, \tilde{s}^{\delta_i'}. \qquad (3.30)$$

Considering $v = \tilde{v}^{\alpha''}$, $u = \tilde{u}^{\beta''}$, $t = \tilde{t}^{\gamma''}$, $\tilde{s} = \tilde{v}^{\delta''}$ derived from equations (3.27) and (3.28), there arise the auxiliary conditions

$$\tilde{v}^{\alpha''} = e^{\log^2 \tilde{u}^{\beta''}}, \; \tilde{v} = e^{\frac{\alpha''}{\beta''} \log^2 \tilde{u}}$$

and

$$\tilde{t}^{\gamma''} = e^{\log^2 \tilde{s}^{\delta''}}, \; \tilde{t} = e^{\frac{\gamma''}{\delta''} \log^2 \tilde{s}}.$$

In equation (3.30) the final polynomial is shown, which is zero *only if* all coefficients a_i, $i = 1 \ldots \hat{z}$, i.e., α_i equal zero, and thus the approximation (eq. 3.26) is linear independent.

3.6.1 Calculating Inner Product and Norm

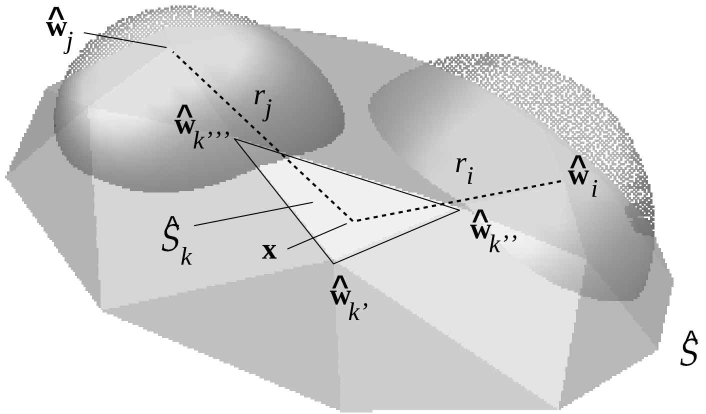


Figure 3.2: Parameterization of an integration point $\mathbf{x}$ on a triangle of the shading network (one simplex) to calculate the concerning proportion of the inner product of two RBFs, $\Lambda_i(r_i), \Lambda_j(r_j)$, each centered at $\hat{\mathbf{w}}_i$ and $\hat{\mathbf{w}}_j$, respectively.

Section 3.6 shows that inner product and norm of two elements $x, y \in \mathcal{M}$ can be reduced to the combination of terms of the form $\int_{\hat{\mathcal{S}}} \Lambda_i \Lambda_j$. Considering figure 3.2, we assume the Λ_i, Λ_j located at two cell centers $\hat{\mathbf{w}}_i$ and $\hat{\mathbf{w}}_j$. Integration has to be done over the environment $\hat{\mathcal{S}}$ — all separate triangles $\hat{\mathcal{S}}_k, k = 1 \ldots \hat{n}$ of the shading network where $\hat{n}$ is the total number of its simplices.

$$\langle \Lambda_i | \Lambda_j \rangle = \int_{\hat{\mathcal{S}}} \Lambda_i \Lambda_j \, d\mathbf{x} = \sum_{k=1}^{\hat{n}} \int_{\hat{\mathcal{S}}_k} \Lambda_i \Lambda_j \, d\mathbf{x}, \tag{3.31}$$

In figure 3.2 one simplex k is depicted, circumscribed by the cell centers $\hat{\mathbf{w}}_{k'}$, $\hat{\mathbf{w}}_{k''}$, and $\hat{\mathbf{w}}_{k'''}$. Integration is accomplished over a point $\mathbf{x} \in \mathbb{R}^3$ on $\hat{\mathcal{S}}_k$. We define an integration substitution, $\mathbf{x} : \mathbb{R}^2 \to \mathbb{R}^3$, which runs through the area of one triangle $\hat{\mathcal{S}}_k$, by barycentric coordinates like

$$\mathbf{x}(s,t) = (1-s-t)\,\hat{\mathbf{w}}_{k'} + s\,\hat{\mathbf{w}}_{k''} + t\,\hat{\mathbf{w}}_{k'''}, \quad \text{with} \quad s+t \leq 1. \tag{3.32}$$

From equation (3.32), the arguments of Λ_i, Λ_j (eq. 3.20), $r_i, r_j : \mathbb{R}^2 \to \mathbb{R}$ — the distances from the centers $\hat{\mathbf{w}}_i$, $\hat{\mathbf{w}}_j$ of the Gaussians — are given by

$$r_i(s,t) = \|\hat{\mathbf{w}}_i - \mathbf{x}(s,t)\|, \quad \text{and} \quad r_j(s,t) = \|\hat{\mathbf{w}}_j - \mathbf{x}(s,t)\|. \tag{3.33}$$

Substituting the three components of equation (3.32) into equation (3.31) leads to

$$\begin{aligned} \langle \Lambda_i | \Lambda_j \rangle &= \sum_{k=1}^{\hat{n}} \int_{\hat{\mathcal{S}}_k} \Lambda_i \Lambda_j \; d\mathbf{x} \\ &= \sum_{k=1}^{\hat{n}} \int_0^1 \int_0^{1-t} e^{-\hat{d}_i^{-2} r_i^2}\, e^{-\hat{d}_j^{-2} r_j^2} \left\| \frac{\partial \mathbf{x}}{\partial s} \times \frac{\partial \mathbf{x}}{\partial t} \right\| \, ds\, dt, \end{aligned} \tag{3.34}$$

with $J_{ijk} = \left\| \frac{\partial \mathbf{x}}{\partial s} \times \frac{\partial \mathbf{x}}{\partial t} \right\|$ is constant[9] due to the linearity of equation (3.32) and can be moved out of the integral. Finally inserting equation (3.33) into equation (3.34) delivers

$$\boxed{\langle \Lambda_i | \Lambda_j \rangle = \sum_{k=1}^{\hat{n}} J_{ijk} \int_0^1 \int_0^{1-t} e^{P_2(s,t)} \; ds\, dt,} \tag{3.35}$$

where $P_n(x)$ denotes a polynomial in x of degree n of the form $P_2(s,t) = A\,s^2 + B\,t^2 + C\,s + D\,t + E\,st + F$ with coefficients $A...F$. Due to its obviousness, we omit the lengthy derivation of these coefficients and suggest using reliable automatic tools like *Mathematica* [65].

Now, we are able to calculate the inner product of two Gaussians Λ_i and Λ_j on $\hat{\mathcal{S}}$, and herewith, the inner product and norm of two elements of $\mathbf{N}$. We continue with the proof of the remaining axioms from page 49.

Assuming two elements x and y of the set $\mathcal{M}$.

[9] J_{ijk} is derived by applying the integration substitution rule, which is explained in the appendix on page 101 in more detail.

Proof.

- I_1: From Equations (3.23) and (3.31) follows

$$\begin{aligned}\langle x|y\rangle &= \int_{\hat{\mathcal{S}}} x\,y\;d\mathbf{x}\\ &= \sum_{1\le i,j\le\hat{z}} a_i b_j \int_{\hat{\mathcal{S}}} \Lambda_i\Lambda_j\;d\mathbf{x} = \sum_{1\le i,j\le\hat{z}} a_i b_j \sum_{k=1}^{\hat{n}} \int_{\hat{\mathcal{S}}_k} \Lambda_i\Lambda_j\;d\mathbf{x},\end{aligned}$$

  which is commutative since summation in $\mathbb{R}$ is commutative and $\int_{\hat{\mathcal{S}}_k} \Lambda_i\Lambda_j\;d\mathbf{x} = \int_{\hat{\mathcal{S}}_k} \Lambda_j\Lambda_i\;d\mathbf{x}$ holds for all $k = 1 \ldots \hat{n}$.

- I_2:

$$\begin{aligned}&\langle(\alpha x+\beta y)|z\rangle\\ &\quad= \left\langle\left(\alpha\sum_{i=1}^{\hat{z}} a_i\Lambda_i + \beta\sum_{i=1}^{\hat{z}} b_i\Lambda_i\right)\middle|\sum_{j=1}^{\hat{z}} c_j\Lambda_j\right\rangle\\ &\quad= \left\langle\sum_{i=1}^{\hat{z}}(\alpha a_i+\beta b_i)\Lambda_i\middle|\sum_{j=1}^{\hat{z}} c_j\Lambda_j\right\rangle\\ &\quad= \sum_{1\le i,j\le\hat{z}}(\alpha a_i+\beta b_i)c_j\int\Lambda_i\Lambda_j\;d\mathbf{x}\;\;\text{(see equation (3.23))}\\ &\quad= \sum_{1\le i,j\le\hat{z}}\alpha a_i c_j\int_{\hat{\mathcal{S}}}\Lambda_i\Lambda_j\;d\mathbf{x} + \sum_{1\le i,j\le\hat{z}}\beta b_i c_j\int_{\hat{\mathcal{S}}}\Lambda_i\Lambda_j\;d\mathbf{x}\\ &\quad= \alpha\,\langle x|z\rangle + \beta\,\langle y|z\rangle\,.\end{aligned}$$

- I_3: From equation (3.24) and I_5 follows

$$\langle x|x\rangle = \|x\|^2 = \int_{\hat{\mathcal{S}}}\left(\sum_{i=1}^{\hat{z}} a_i\Lambda_i\right)^2 d\mathbf{x}, \tag{3.36}$$

  which is nonnegative since integration is done over a nonnegative function.

- I_4: Since

$$\langle x|x\rangle = \sum_{i=1}^{\hat{z}} a_i^2\int_{\hat{\mathcal{S}}}\Lambda_i^2\;d\mathbf{x} + 2\sum_{\substack{1\le i,j\le\hat{z}\\ j>i}} a_i a_j\int_{\hat{\mathcal{S}}}\Lambda_i\Lambda_j\;d\mathbf{x}, \tag{3.37}$$

  if $x=\emptyset$ then $\langle x|x\rangle$ equals 0. Otherwise, if $x\neq 0$ at at least one location, then, due to the continuity of x, $\langle x|x\rangle\neq 0$ holds.

- ∘ N_1: From $\|x\| = \sqrt{\langle x|x\rangle}$ and I_3 and I_4 follows N_1.
- ∘ N_2: Since $\alpha x = \sum_{i=1}^{\hat{z}} \alpha a_i \Lambda_i$ holds and coefficients a_i and herewith α always appear pairwise in equation (3.24), N_2 holds.
- ∘ N_3: Squaring $\|x+y\| \leq \|x\| + \|y\|$ leads to

$$\|x+y\|^2 \leq \|x\|^2 + \|y\|^2 + 2\|x\| \cdot \|y\|,$$

where we insert equation (3.24),

$$\begin{aligned}\sum_{i=1}^{\hat{z}} (a_i+b_i)^2 \int_{\hat{\mathcal{S}}} \Lambda_i^2 \, d\mathbf{x} + 2 \sum_{\substack{1\leq i,j\leq \hat{z}\\ j>i}} (a_i+b_i)(a_j+b_j) \int_{\hat{\mathcal{S}}} \Lambda_i \Lambda_j \, d\mathbf{x} \leq \\ \sum_{i=1}^{\hat{z}} a_i^2 \int_{\hat{\mathcal{S}}} \Lambda_i^2 \, d\mathbf{x} + 2 \sum_{\substack{1\leq i,j\leq \hat{z}\\ j>i}} a_i a_j \int_{\hat{\mathcal{S}}} \Lambda_i \Lambda_j \, d\mathbf{x} + \\ \sum_{i=1}^{\hat{z}} b_i^2 \int_{\hat{\mathcal{S}}} \Lambda_i^2 \, d\mathbf{x} + 2 \sum_{\substack{1\leq i,j\leq \hat{z}\\ j>i}} b_i b_j \int_{\hat{\mathcal{S}}} \Lambda_i \Lambda_j \, d\mathbf{x} + 2\|x\| \cdot \|y\|.\end{aligned}$$

Expanding and collecting the terms on both sides leads to

$$\begin{aligned}\sum_{i=1}^{\hat{z}} a_i b_i \int_{\hat{\mathcal{S}}} \Lambda_i^2 \, d\mathbf{x} + \left[\sum_{\substack{1\leq i,j\leq \hat{z}\\ j>i}} (a_i b_j + a_j b_i) \int_{\hat{\mathcal{S}}} \Lambda_i \Lambda_j \, d\mathbf{x} \right] &\leq \|x\| \cdot \|y\| \\ \cdots \left[\sum_{1\leq i,j\leq \hat{z}} a_i b_j \int_{\hat{\mathcal{S}}} \Lambda_i \Lambda_j \, d\mathbf{x} - \sum_{i=1}^{\hat{z}} a_i b_i \int_{\hat{\mathcal{S}}} \Lambda_i^2 \, d\mathbf{x} \right] &\leq \cdots \\ \langle x|y\rangle &\leq \|x\| \cdot \|y\|,\end{aligned}$$

and $\langle x|y\rangle^2 \leq \|x\|^2 \cdot \|y\|^2$ rewritten like $\left(\int x \cdot y \, d\mathbf{x}\right)^2 \leq \int x^2 \, d\mathbf{x} \cdot \int y^2 \, d\mathbf{x}$ holds (see [11], page 176). This also holds for surface integrals, like used in this work, due to the fact that the reparameterization just adds a constant J_{ijk} to the integral (see equation (3.35) and section A.1).

□

3.7 Constant Base versus Gaussian Base

We introduced the representation of the emitted radiosity $B(\mathbf{x})$ at the surfaces by a linear function base of a set of Gaussians, instead of assuming a

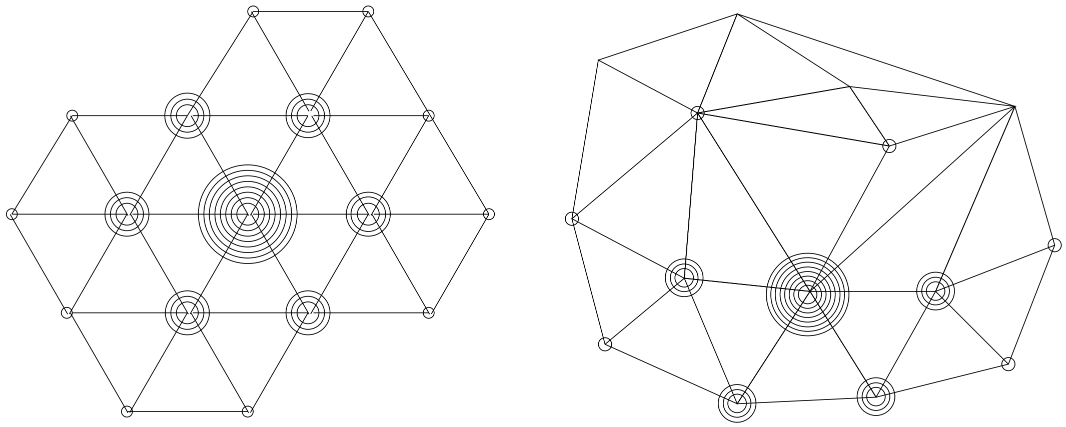


Figure 3.3: On the left, the ideal distribution of Gaussian centers to reach the best approximation accuracy of a constant function. On the right, a snapshot from a typical shading network topology, the edge lengths emanating from a certain cell vary stronger. Only one base function is shown on each side, the sizes of the slices go with the according inner product with the center Gaussian.

constant base like proposed in section 3.6, and instead of just numerically integrating over the shading network triangles. Its advantage is the possibility of an analytical integration of the transfer coefficients. Disadvantages arise in the induced approximation error, which is the topic of this paragraph.

We stated on page 42 that the additional base should equal the constant 1 at all surface points. Thus, we investigate the general error arising through approximating an exact constant by a set of Gaussians. The approximation error is characterized by two criterions.

a) The best approximation is given if the network cells, i.e., the positions of the Gaussians, are ideally distributed. In this case, all edges emanating from a cell have equal lengths. Thus, all Gaussian centers are positioned in the same distance from its neighbors, and the remaining error comes from the general approximation facility regarding the shape of the chosen function base components.

b) The distribution of Gaussians is less uniform. This is the case if the expansion ($\hat{d}_k$ of a Gaussian $e^{-\hat{d}_k{}^{-2} s^2}$) which is calculated by the average edge length emanating from a cell (see page 25), differs significantly from the single edge lengths.

See figure 3.3 as an example. On the left, an optimal uniform distribution of the shading network which mimics the two-dimensional surface space is shown. It corresponds to the locally uniform distribution of training samples.

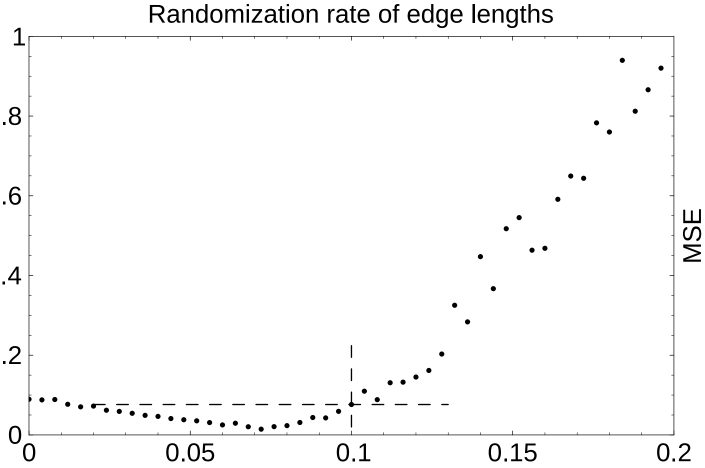


Figure 3.4: The error of a Gaussian function base approximating a constant. On the left, the error of the ideal case of uniformly distributed Gaussians, increasing to the right side of the diagram, where the positions of the cells are randomized at a certain rate. Experiments have shown that usually the network centers are randomized by a rate of about 10% which leads to an average MSE of about 10%.

On the right, a more realistic shape of the shading network is exposed, where the positions jitter around their optimal centers. In the following, we estimate the error by an experiment.

A network is created of the size of 20 × 20 Gaussians. The Gaussians are distributed with equal distance to their direct neighbors (like in figure 3.3 on the left side). The base is orthonormalized and projected on the constant 1. Then, samples from inside of one triangle in the center of the whole network are taken and the network output at these sample positions is compared to 1. Assuming the described constellation of Gaussians which are equally distributed in the function space is optimal, now, the error we can achieve in approximating a constant is minimal.

Since Gaussians in the shading network usually are not distributed in that equability, we generate an artificial jittering of the Gaussians from their ideal place, and draw the error against the jittering rate in figure 3.4.

From the diagram, we are able to derive the error generated through the use of the Gaussian base, since, in experiments, it has been measured that the shading network edges of one Gaussian center vary by about 10%. Thus, an average error increase by about 10% in the whole radiosity approximation is assumed.

3.8 Operating the Radiosity

The kernel is a linear operator operating on the surface space $\hat{\mathcal{S}}$. In a finite element approach an approximation of the radiosity is given through a linear function system and the projection of a function at $\hat{\mathcal{S}}$, here, turns into the projection of the approximation's base components. Herewith, a base component N_k is projected onto its *dual* $\tilde{N}_k$ and the transfer coefficient is calculated by the inner product of a component with its dual. Since orthonormal bases are *self-dual*, calculating the dual of each of the base components is equivalent to *orthonormalizing* the particular base, which is described in the following section.

3.8.1 Orthonormalizing the Gaussian Base

Orthogonalization can be seen as a "strong form of linear independence" [60]. It is based on the inner product of two functions which can be regarded as the *overlap*, the *support*, or the "*influence*" which they exercise over each other. This influence must be accounted for if, for example, a base transformation is accomplished.

We assume a vector $\mathbf{\Lambda}$ of non-orthonormal, linear independent function base components Λ_k, $i = 1 \ldots \hat{z}$ and our goal of an orthogonal function base vector $\mathbf{N} = (N_1, N_2 \ldots N_{\hat{z}})$.

Definition 9. *A set of n functions f_i, $i = 1 \ldots n$ is orthogonal if $\langle f_i | f_j \rangle = 0$, $\forall i, j : i \neq j$, $i, j = 1 \ldots n$ holds. It is orthonormal if, additionally, the functions' norms $\|f_i\|$ equal 1 for all $i = 1 \ldots n$.*

In this work, $\mathbf{N}$ is calculated by the *Schmidt orthogonalization* rule,

$$N_1 := \Lambda_1, \quad N_k := \Lambda_k - \sum_{i=1}^{k-1} \frac{\langle \Lambda_k | N_i \rangle}{\langle N_i | N_i \rangle}, \; N_i, \; k = 2 \ldots \hat{z}, \tag{3.38}$$

which delivers a sequence of orthogonal functions $N_1, N_2 \ldots N_{\hat{z}}$ which finally are normalized by

$$N_k := \|N_k\|^{-1} N_k, \; k = 1 \ldots \hat{z}. \tag{3.39}$$

Practically spoken, scheme (3.38) delivers successively orthogonal functions N_k, $k = 1 \ldots \hat{z}$ which are combinations of the $k - 1$ foregoing developments N_j, $j = 1 \ldots k - 1$ and another new Λ_k. A coefficient matrix $\mathbf{A}$,

$$\begin{array}{c|cccc}
\mathbf{A} & \mathit{\Lambda}_1 & \mathit{\Lambda}_2 & \cdots & \mathit{\Lambda}_{\hat{z}} \\
\hline
N_1 & a_{11} & 0 & \cdots & 0 \\
N_2 & a_{21} & a_{22} & \ddots & \vdots \\
\vdots & \vdots & & \ddots & 0 \\
N_{\hat{z}} & a_{\hat{z}1} & & \cdots & a_{\hat{z}\hat{z}}
\end{array}$$

is created consisting of rows $\mathbf{a}_k$ of coefficients $a_{k1}, a_{k2} \dots a_{k\hat{z}}$, which define the resulting orthonormal base components

$$N_k = \sum_{i=1}^{\hat{z}} a_{i\hat{z}} \mathit{\Lambda}_i = \mathbf{a}_k \, \mathbf{\Lambda} = \sum_{i=1}^{k} a_{i\hat{z}} \mathit{\Lambda}_i . \tag{3.40}$$

The described scheme robustly generates the coefficients of $\mathbf{A}$. In some cases, i.e., for some critical topologies of the shading network (see section 4.4), there arise "nearly linear dependence" which means that Gaussians of a new base component, generated in equation (3.38), are very similar to Gaussians of a formerly created N_k, i.e., there are similar centers and expansions. These cases are detected by the above scheme through a very small value $\|N_k\|$. In the implementation, these cases are intercepted by testing a threshold. If $\|N_k\|$ falls below the threshold, then the particular N_k is rejected from the base and the corresponding $\mathit{\Lambda}_k$ which is taken to generate the new N_k (eq. 3.38) is not longer considered in the following development.

3.8.2 Integration

Up to now, the radiosity base and the kernel model has been defined. It remains the integration operation, the calculation of the transfer coefficients h_{ij} derived in section 3.3 (eq. 3.12).

We assume the vector of orthonormal radiosity base components $\mathbf{N} = (N_1, N_2 \dots N_n)$ and recall the definition of the transfer coefficients,

$$h_{ij} = \int_{\hat{\mathcal{S}}} N_i(\mathbf{y}) \int_{\hat{\mathcal{S}}} \mathit{\Psi}(\mathbf{x}, \mathbf{y}) N_j(\mathbf{x}) \; d\mathbf{x} \; d\mathbf{y}. \tag{3.41}$$

Resolving the RBFs in equation (3.41) by applying equation (3.40) and the kernel representation from equation (3.4) leads to

$$\begin{aligned}
h_{ij} &= \int_{\hat{\mathcal{S}}} \mathbf{\Lambda}(\mathbf{y}) \, \mathbf{a}_i \int_{\hat{\mathcal{S}}} \mathbf{v} \, \mathbf{\Omega}(\mathbf{x}, \mathbf{y}) \, \mathbf{\Lambda}(\mathbf{x}) \, \mathbf{a}_j \; d\mathbf{x} \; d\mathbf{y} \\
&= \sum_{p,q=1}^{\hat{z}} a_{ip} \, a_{jq} \sum_{o=1}^{z} v_o \iint_{\hat{\mathcal{S}}} \mathit{\Lambda}_p(\mathbf{x}) \mathit{\Omega}_o(\mathbf{x}, \mathbf{y}) \mathit{\Lambda}_q(\mathbf{y}) \; d\mathbf{x} \; d\mathbf{y}
\end{aligned} \tag{3.42}$$

For calculating equation (3.42) for one particular pair of bases (i, j), all combinations of three RBFs, those of the kernel and a pair of radiosity base components must be integrated. For notational convenience, we define coefficients $\hbar_{pqo}$ representing these combinations by

$$\hbar_{pqo} = \iint_{\hat{\mathcal{S}}} \Lambda_p(\mathbf{x})\ \Lambda_q(\mathbf{y})\ \Omega_o(\mathbf{x}, \mathbf{y})\ d\mathbf{x}\ d\mathbf{y}, \tag{3.43}$$

containing the integrands of a pair of radiosity RBFs (Λ_p and Λ_q) and one kernel Gaussian (Ω_o). This *fractional transfer coefficient* looks, from the computational point of view, similar to the integration task explained in section 3.6.1 by equation (3.34). The difference is the additional transfer Gaussian Ω_o. The term $\hbar_{pqo}$ (eq. 3.43) can be regarded like the transfer coefficient between the two components Λ_p and Λ_q through a kernel component Ω_o. See figure 3.5 for the following explanations.

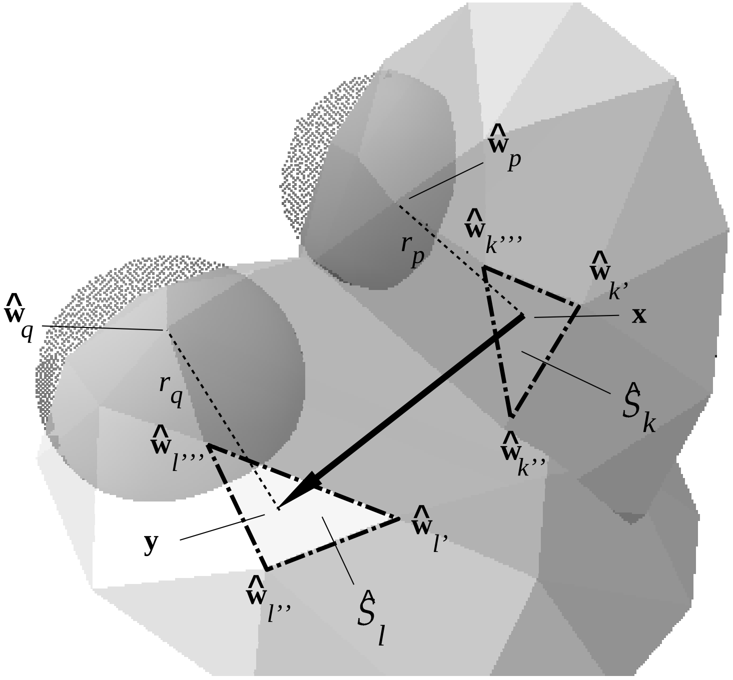


Figure 3.5: Parameterization of the integration points $\mathbf{x}$ and $\mathbf{y}$ on two triangles (simplices) of the shading network to calculate the concerning proportion of the transfer coefficient of two RBFs, $\Lambda_i(r_p), \Lambda_j(r_q)$, centered at $\hat{\mathbf{w}}_p$ and $\hat{\mathbf{w}}_q$.

Considering two components Λ_p and Λ_q centered at the corresponding cell centers $\hat{\mathbf{w}}_p$ and $\hat{\mathbf{w}}_q$. Integration must be accomplished over all pairs of triangles $\{(\hat{\mathcal{S}}_k, \hat{\mathcal{S}}_l) : \hat{\mathcal{S}}_k, \hat{\mathcal{S}}_l \in \hat{\mathcal{S}}\}$ of the approximate geometry $\hat{\mathcal{S}}$ defined by the shading network.

$$\hbar_{pqo} = \iint_{\hat{\mathcal{S}}} \mathit{\Lambda}_p \, \mathit{\Lambda}_q \, \mathit{\Omega}_o \, d\mathbf{x} \, d\mathbf{y} = \sum_{1 \le k,l \le \hat{n}} \int_{\hat{\mathcal{S}}_k} \int_{\hat{\mathcal{S}}_l} \mathit{\Lambda}_p \, \mathit{\Lambda}_q \, \mathit{\Omega}_o \, d\mathbf{x} \, d\mathbf{y} \tag{3.44}$$

One pair is denoted by k and l in figure 3.5, and it is bounded by the vertices $\hat{\mathbf{w}}_{k'}$, $\hat{\mathbf{w}}_{k''}$, $\hat{\mathbf{w}}_{k'''}$, and $\hat{\mathbf{w}}_{l'}$, $\hat{\mathbf{w}}_{l''}$, and $\hat{\mathbf{w}}_{l'''}$, respectively. We assume two integration points $\mathbf{x}$ on $\hat{\mathcal{S}}_k$ and $\mathbf{y}$ on $\hat{\mathcal{S}}_l$, and, similar to section 3.6.1, we define a two-dimensional parameterization, $\mathbf{x}, \mathbf{y} : \mathbb{R}^2 \to \mathbb{R}^3$, over both triangles,

$$\mathbf{x}(s,t) = (1-s-t)\,\hat{\mathbf{w}}_{k'} + s\,\hat{\mathbf{w}}_{k''} + t\,\hat{\mathbf{w}}_{k'''}, \quad \text{with} \quad s+t \le 1, \tag{3.45}$$

$$\mathbf{y}(u,v) = (1-u-v)\,\hat{\mathbf{w}}_{l'} + u\,\hat{\mathbf{w}}_{l''} + v\,\hat{\mathbf{w}}_{l'''}, \quad \text{with} \quad u+v \le 1. \tag{3.46}$$

The distances $r_p, r_q : \mathbb{R}^2 \to \mathbb{R}$ of $\mathbf{x}$ and $\mathbf{y}$ from the centers $\hat{\mathbf{w}}_p$ and $\hat{\mathbf{w}}_q$, respectively, can be written like

$$r_p(s,t) = \|\hat{\mathbf{w}}_p - \mathbf{x}(s,t)\|, \quad \text{and} \quad r_q(u,v) = \|\hat{\mathbf{w}}_q - \mathbf{y}(u,v)\|, \tag{3.47}$$

and the distance $r_o : \mathbb{R}^4 \to \mathbb{R}$ of the concatenation $(\mathbf{x}, \mathbf{y}) \in \mathbb{R}^6$ from the center $\mathbf{w}_o \in \mathbb{R}^6$ of the kernel Gaussian $\mathit{\Omega}_o$ is

$$r_o(s,t,u,v) = \sqrt{r_p^2(s,t) + r_q^2(u,v)}. \tag{3.48}$$

Similar to page 53, replacing equations (3.47) and (3.48) in equation (3.44) leads to the solution

$$\begin{aligned} &\hbar_{pqo} = \sum_{1 \le k,l \le \hat{n}} \int_{\hat{\mathcal{S}}_k} \int_{\hat{\mathcal{S}}_l} \mathit{\Lambda}_p \, \mathit{\Lambda}_q \, \mathit{\Omega}_o \, d\mathbf{x} \, d\mathbf{y} = \\ &\sum_{1 \le k,l \le \hat{n}} \int_0^1 \int_0^{1-v} \int_0^1 \int_0^{1-t} e^{-\hat{d}_p^{-2} r_p^2} \, e^{-\hat{d}_q^{-2} r_q^2} \, e^{-d_o^{-2} r_o^2} \, \cdot \\ &\qquad \left\| \frac{\partial \mathbf{x}}{\partial s} \times \frac{\partial \mathbf{x}}{\partial t} \right\| \left\| \frac{\partial \mathbf{y}}{\partial u} \times \frac{\partial \mathbf{y}}{\partial v} \right\| \, ds \, dt \, du \, dv, \end{aligned} \tag{3.49}$$

where the constant[10] $J_{pqokl} = \|\frac{\partial \mathbf{x}}{\partial s} \times \frac{\partial \mathbf{x}}{\partial t}\| \, \|\frac{\partial \mathbf{y}}{\partial u} \times \frac{\partial \mathbf{y}}{\partial v}\|$ can be drawn out of the integral, resulting in a term of the form

$$\boxed{\hbar_{pqo} = \sum_{1 \le k,l \le \hat{n}} J_{pqokl} \int_0^1 \int_0^{1-v} \int_0^1 \int_0^{1-t} e^{P_2(s,t,u,v)} \, ds \, dt \, du \, dv,} \tag{3.50}$$

with a polynomial $P_2(s,t,u,v) = A\,s^2 + B\,t^2 + C\,u^2 + D\,v^2 + E\,s + F\,t + G\,u + H\,v + I\,st + J\,uv + K$. We recommend to support the calculation of $\hbar_{pqo}$ by using Mathematica [65].

[10] For its derivation, see the appendix on page 101.

3.9 Approximate Integrations

Computation of the integration operations for the inner product (section 3.6.1) and the transfer coefficient (section 3.8.2) are quite costly due to two reasons. First, the integration has to be done over each triangle of the shading network separately, second, the integration itself must be done numerically since an analytical solution is not available.

In the following, we propose an integration shortcut which is based on the property that, in certain regions of the shading network, the triangles are positioned in a way that they form a unique, flat surface — a *unique domain*. In these cases, integration does not need to be done over each simplex separately, and moreover, if integration is done over the complete support of the Gaussians, then an analytical solution (described in sections 3.9.2 and 3.9.3) is available.

3.9.1 Criterion for Symbolic Integration

For the decision when to apply symbolic or numerical integration, two conditions must be verified. First, the domain of integration must *exist*, i.e., the support of a participating Gaussian must not exceed the shading network boundaries, and second, the Gaussian must lie in a *flat domain*. In later sections, the mentioned properties are characterized by continuous parameters which measure a certain degree of how much they apply, and a threshold decides if the criteria hold or not.

1) The **existence** condition is satisfied if a certain Λ_i supports *mainly* inner pieces of the shading network topology. In other words, the approximate geometry, represented by the shading network, must exist under a particular Gaussian — the expansion $\hat{d}_i$ should not exceed the range of the network significantly.

2) The determination of a **unique domain** requires the following two conditions.

   a) The set of those triangles which are supported by a certain Gaussian describes a *flat area* at average. This arises, for example, at locations which lie "central" to a geometry surface, since the shading network adapts to the geometry, and thus to its flatness.

   b) Two Gaussians Λ_i, Λ_i lie in the same domain if, first, their normals point into the same direction, and second, if they belong to the same closed network portion.[11]

[11] The shading network usually is split into several subnetworks through the training (see

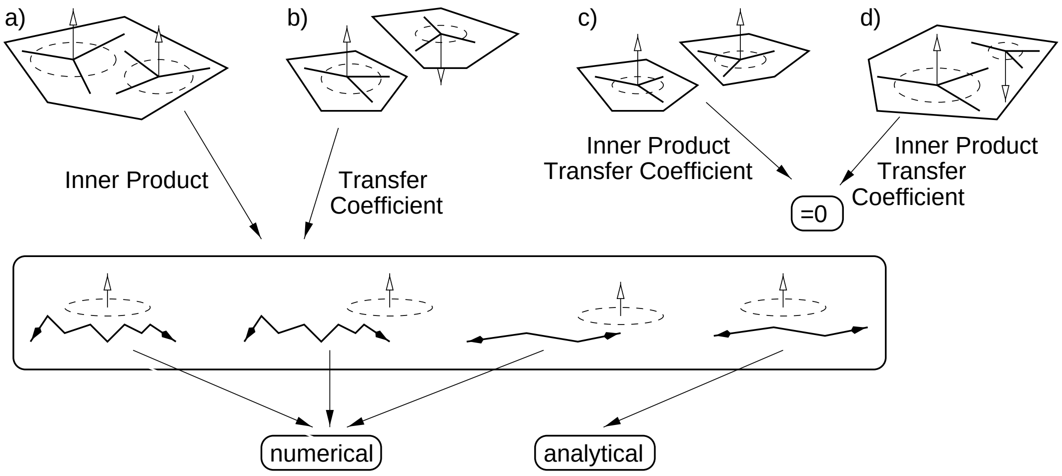


Figure 3.6: Algorithm for deciding when to use numerical instead of analytical integration of the transfer coefficient and of the inner product, respectively.
The upper row shows the general possibilities of how the participating two base components can be located on the shading network. For example, calculation of the inner product is not accomplished (set to zero) if the components are located on the same sub-network with opposed normals (d) or if they are located on separate sub-networks (c). The transfer coefficient is set to zero, for example, if the normals of the participating components do not point to each other (c). In cases (a) and (b) integrations are done (lower box) according to the network shape — numerical integration is accomplished if the participating base components lie too close to the network boundaries or if the network roughness supported by the base components is too high, otherwise analytical integration is accepted.

Depending on these conditions, we decide about the kind of computation for the following tasks.

I) **Inner product:** If point 2b) is not fulfilled, the inner product is set to zero. Otherwise symbolic integration is triggered in cases where points 1) and 2a) are satisfied for the participating Λ_i. For all other cases, numerical integration is applied.

II) **Transfer coefficient:** If point 2b) is satisfied the transfer coefficient is set to zero[12]. Otherwise symbolic integration is accomplished if points 1) and 2a) hold for both RBFs, if not, numerical integration is applied.

section 2.2), each of these networks is defined as belonging to different domains, and each contained cell is marked with the corresponding network identifier.

[12] This serves for efficiency reasons only, since the transfer coefficient also equals zero if it is calculated through the kernel network.

For an overview of the algorithm, see figure 3.6. There, each arrow exposes the only case in which the integrations are performed for the different tasks instead of assigning the results to zero.

The exact definition of the "existence" and the "being a flat domain" property which finally enable their implementation is explained in section 3.9.4.

Enhanced radiosity base shape. We like to mention that setting the inner product in a way described above virtually changes the described base definition but it does not change its principle properties and especially its validity which has been proven in section 3.2. Moreover, the base qualifying for an instantiation of a radiosity base is optimized due to the normal criterion (point 2b) which is an additional facility suitable for modeling edges in the network topology. By cutting the influence between base components at sharp edges, an oracle is defined which "seems to detect" an edge of the underlying geometry, modeled by the shading network function base.

3.9.2 Approximating the Inner Product

Presupposing a unique domain, the single participating elements $\hat{\mathcal{S}}_k$ are assumed instantiating a coherent surface $\hat{\mathcal{S}}$. Thus, the inner product can be integrated over a flat domain and without limits as follows.

Consider again equations (3.31) and (3.34). The integral over all $\hat{\mathcal{S}}_k$ now can simply be written as an *area* integral over the common parameter r,

$$\begin{aligned}\langle \Lambda_i | \Lambda_j \rangle &= \sum_{k=1}^{\hat{n}} \int_{\hat{\mathcal{S}}_k} \Lambda_i \Lambda_j \; d\mathbf{x} \\ &\approx \int_{\mathbb{R}^2} e^{-\hat{d}_i^{-2}\, r^2} \cdot e^{-\hat{d}_j^{-2}\, (r-d)^2} \; dr \qquad (3.51)\\ &\qquad\qquad = \widehat{\langle \Lambda_i | \Lambda_j \rangle},\end{aligned}$$

with d as the distance between the centers $\hat{\mathbf{w}}_i$ and $\hat{\mathbf{w}}_j$. Equation (3.51) can be solved analytically like

$$\boxed{\widehat{\langle \Lambda_i | \Lambda_j \rangle} = \pi \, \frac{\hat{d}_i^2 \hat{d}_j^2}{\hat{d}_i^2 + \hat{d}_j^2} \, e^{-\frac{d^2}{\hat{d}_i^2 + \hat{d}_j^2}}.} \qquad (3.52)$$

From equation (3.52) the solution of the norm $\|\Lambda_i\| = \langle \Lambda_i | \Lambda_i \rangle^{\frac{1}{2}}$ of a Λ_i can be derived like

$$\boxed{\widehat{\|\Lambda_i\|} = \hat{d}_i^2 \pi.} \qquad (3.53)$$

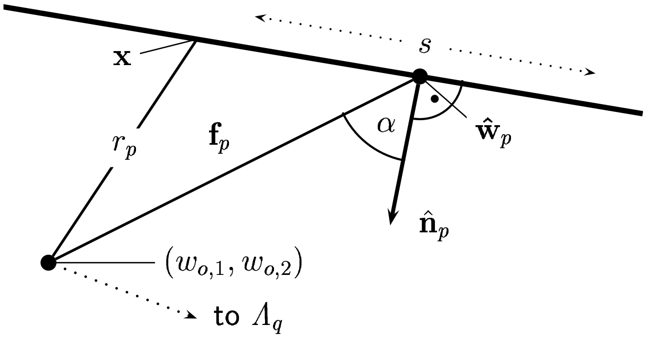


Figure 3.7: The Flatland geometry for integration. Only the upper part of a kernel reference ray is shown.

3.9.3 Approximating the Transfer Coefficient

With the same assumptions concerning the integration domain (the $\hat{\mathcal{S}}_k$ and $\hat{\mathcal{S}}_l$ approach $\mathcal{S}$), equation (3.44) can be rewritten over a homogeneous domain,

$$\begin{aligned} \hbar_{pqo} &= \sum_{1\le k,l\le \hat{n}} \int_{\hat{\mathcal{S}}_k} \int_{\hat{\mathcal{S}}_l} \Lambda_p \, \Lambda_q \, \Omega_o \, d\mathbf{x} \, d\mathbf{y} \\ &\approx \iint_{\mathbb{R}^2} e^{-\hat{d}_q^{-2}\, t^2} \cdot e^{-\hat{d}_p^{-2}\, s^2} \cdot e^{-d_o{}^{-2}\, r^2} \, ds \, dt \qquad (3.54) \\ & \qquad\qquad = \hat{\hbar}_{pqo}, \end{aligned}$$

with the parameterization $r = \|(x_1, x_2, x_3, y_1, y_2, y_3) - \mathbf{w}_o\|$. In the following derivation, we assume Λ_p and Λ_q being one-dimensional functions defined on lines of a two-dimensional scene geometry (Flatland).

The parameters s and t are the distance arguments of the functions Λ_p and Λ_q, respectively. The normals $\hat{\mathbf{n}}_p$ and $\hat{\mathbf{n}}_q$ are given by the average of the normals of the neighboring simplices (see figure 3.7).

The radius r is derived from $\mathbf{x}$ and $\mathbf{y}$ as follows. Letting s and t move on two lines, the distance r from the center $\mathbf{w}_o$ is

$$r = r_p^2 + r_q^2 \quad \text{with} \quad r_{\{p,q\}}^2 = f_{\{p,q\}}^2 + \{s,t\}^2 - 2\{s,t\} f_{\{p,q\}} \cdot \sin \alpha_{\{p,q\}}. \qquad (3.55)$$

$\alpha_{\{p,q\}}$ are the angles between the "surface" normals $\hat{\mathbf{n}}_{\{p,q\}}$ and the vectors $\mathbf{f}_p = \hat{\mathbf{w}}_p - (w_{o,1}, w_{o,2})$ and $\mathbf{f}_q = \hat{\mathbf{w}}_q - (w_{o,3}, w_{o,4})$ with $f_{\{p,q\}} = \|\mathbf{f}_{\{p,q\}}\|$, and $\mathbf{f}_p$ and $\mathbf{f}_q$ are the vectors between the centers of the surface RBF and the top and bottom of the kernel RBF center (ray), respectively.

Inserting equation (3.55) into equation (3.54)[13] delivers an equation of the form

$$\hbar_{pqo}^{FL} \approx \hat{\hbar}_{pqo}^{FL} = \iint_{\mathbb{R}} e^{P_2(s,t)}\, ds\, dt, \tag{3.56}$$

with a polynomial $P_2(s,t) = -A_p\, s^2 - A_q\, t^2 + B_p\, s + B_q\, t - C$, and the substitutions

$$\begin{aligned} A_{\{p,q\}} &= d_o^{-2} + \hat{d}_{\{p,q\}}^{-2}, & (3.57)\\ B_{\{p,q\}} &= d_o^{-2} \cdot f_{\{p,q\}} \cdot \sin \alpha_{\{p,q\}}, & (3.58)\\ C &= d_o^{-2} \cdot (f_p^2 + f_q^2). & (3.59) \end{aligned}$$

With the common analytical solution

$$\int_{\mathbb{R}} \exp\left(-As^2 + Bs - C\right)\, ds = \frac{\exp\left(\frac{B^2}{4A} - C\right)\sqrt{\pi}}{\sqrt{A}}$$

equation (3.56) can be solved to

$$\hat{\hbar}_{pqo}^{FL} = \frac{\exp\left(\frac{B_p^2}{4A_p} + \frac{B_q^2}{4A_q} - C\right)\pi}{\sqrt{A_p}\sqrt{A_q}}. \tag{3.60}$$

Switching to three-dimensional geometries can be assumed as integrating twice over s and t with different parameters but the same coefficients $A_{\{p,q\}}$, $B_{\{p,q\}}$, and C. Thus, the non-constant arguments of the exponential function in the solution (eq. 3.60) are doubled and the coefficients squared leading with substitutions (3.57), (3.58), and (3.59) to the following solution.

$$\boxed{\hat{\hbar}_{pqo} = \iint_{\mathbb{R}^2} e^{P_2(s,t)}\, ds\, dt = \frac{\exp\left(\frac{B_p^2}{2A_p} + \frac{B_q^2}{2A_q} - C\right)\pi^2}{A_p A_q}} \tag{3.61}$$

3.9.4 Triggering Symbolic Integration

This section describes the estimation of the numerical error which is created by integrating over an approximate ideal surface like described in the preceding paragraph instead of over the triangles of the shading network separately (exact solution). Depending on this error and on a related threshold, then, the decision whether integrating numerically or accepting the symbolic integration is made. On page 69, we begin the examination of four cases following the possible combinations of calculating the inner product or the transfer coefficient by assuming a flat network and an infinite network.

[13] their two-dimensional equivalents

Criterion for the Network Flatness

Figure 3.8 shows how the modification of the network geometry in case of an approximate integration can be thought of. Thick lines stand for the existing shading network topology in a Flatland environment, and the dashed lines for the approximate geometry of a flat shading network which is implicitly assumed when applying the symbolical integration algorithm.

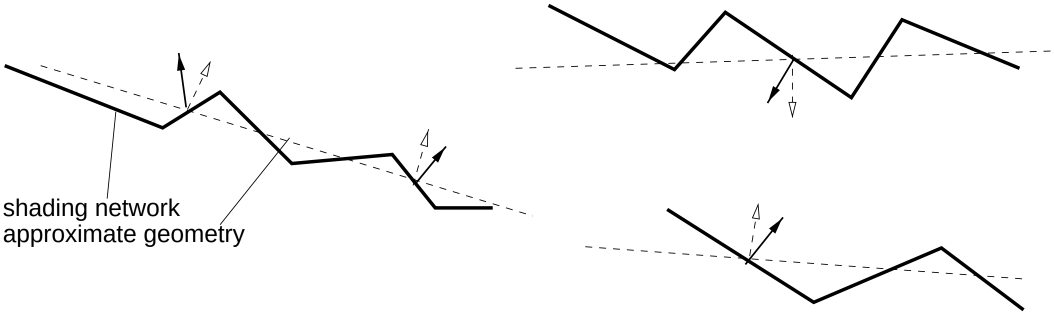


Figure 3.8: Comparison of an example shading network geometry (in Flatland) with the approximation by assuming a flat network (dashed lines). On the left, the geometry for calculating the inner product, on the right, the geometry for the transfer coefficient approximation are shown.

On the left hand side of figure 3.8, the determination of the inner product of two Gaussians is shown. An approximate flat domain is implicitly assumed by the direct connection of the Gaussians' centers. On the right hand side, the geometry for calculating a transfer coefficient $\hbar_{pqo}$ is drawn. In this case, the network domains supported by the Gaussians each are assumed being the infinite continuation of those areas (lines) which are defined by the RBFs' centers and their normals[14]. Note that the approximate geometries are not explicitly generated. Figure 3.8 only explains in which way an error is introduced into the calculation.

Estimating the error must be based on those properties of the underlying network, which are able, on the one hand, to easily be calculated on the fly (during training the network and without requiring much computing resources), and, on the other hand, which allow for calculation of an adequate — at least, worst case — error estimate.

We define a parameter for the *surface roughness*, SR_i, attached to each cell c_i. It figures a value for the *roughness* of the surrounding triangle structure. Each time a sample is presented to the network, the SR_i of the BMU (see section 2.2.1) is calculated from the variance of the normals of the surrounding

[14] defined through training a shading network output layer (see section 3.2)

simplices and it can be seen as an average angle over which the single patches are rotated.

Since these roughness values are locally defined only, we add an algorithm which computes an SR_i which spans a larger environment of the shading network, called *fuzzy surface roughness*, FSR_i, as follows.

At certain time steps, the actual SR_i of each cell is written into the cell's FSR_i and interpreted like a *potential.* Then, this local amount is distributed through the cell edges to the neighboring SR_i with a certain strength. This can be seen like letting a certain amount of fluid flow (*diffuse*) through tubes (*edges*) of a certain thickness (*strength factor*), and it arises a mutual compensation of the potentials between the shading network cells. The process is stopped if the average flow (created by the *potential difference*) which is propagated in the last step falls below a certain threshold.

The result (contained in FSR_i) is a blurring of the SR_i into the FSR_i which then denominate a larger area of the network and which are taken as approximation of a local surface roughness which spans an environment extending that of one cell only.

Criterion for the Network Support ("Existence")

Similar to the roughness, the existence of the network is evaluated for each cell when it is triggered as a BMU in the network training.

The existence of the underlying network is satisfied at those Gaussians which lie, to a certain degree, far from a boundary of the shading network. To determine this property, each cell c_i contains a *depth flag* DF_i which stores the topological distance of the cell to the network boundary. To track the depth of the cells, at the start of the network training (only one simplex exists), the DF_i of all cells contain the value 0 (boundary). Each time a new cell is inserted at an edge, it inherits the smallest DF_i of those cells which define the edge to be split. If cells of the network are deleted such that new subnetworks arise, then each of them is traversed from the newly arisen boundaries by updating the depth flags.

Thus, with each Λ_i a value of its depth in the network is stored serving as a measure for its support on the network. If this support is complete, i.e., if the Λ_i lie sufficiently deep inside of the network, then the existence condition is fulfilled, since the expansion of a Gaussian is proportional to the average edge length of the surrounding edges.

We proposed the integration trigger as a boolean function of two parameters DF_i and FSR_i depending on which the algorithm starting on page 62 is steered. The thresholds which decide if the accuracy of a symbolical integration would

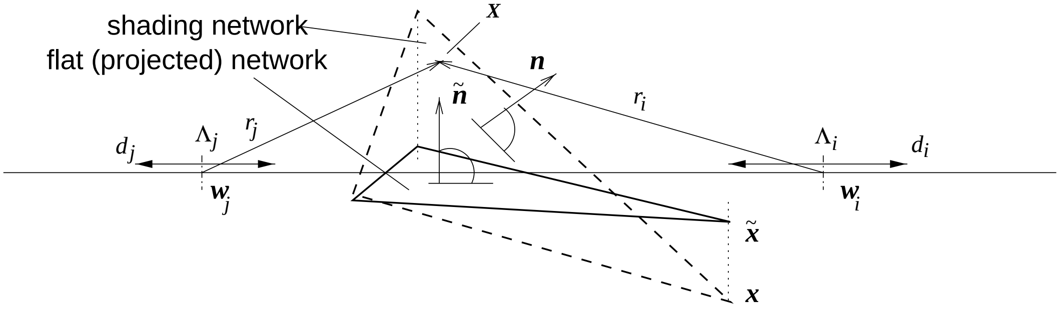


Figure 3.9: Example constellation for calculating the inner product of a pair Λ_i, Λ_j.

be sufficient are derived from the arising integration errors to be determined in the following four sections.

Inner Product Error on a Flat Domain

Figure 3.9 describes the integration task arising in cases of finding the inner product of two functions Λ_i and Λ_j. The integration domain of the network is described by the infinite point set $\{\mathbf{x}\}$, that of the assumed flat projection by points $\{\hat{\mathbf{x}}\}$. r_i and r_j denote the distances of the integration point $\mathbf{X}$ on the network triangle from the Gaussian centers $\mathbf{w}_i, \mathbf{w}_j$. The definition of the integration triangle (by its vertices) is not displayed.

The vectors $\mathbf{n}$ and $\hat{\mathbf{n}}$ are the triangle normals which are taken for averaging a cell's normal and then used for determining the values SR_i.

Regarding all parameters depicted in figure 3.9, calculating the difference between the integrals over $\{\mathbf{x}\}$ and $\{\hat{\mathbf{x}}\}$ is as difficult as computing the integral itself. Thus, a simplification of the integration problem is required, which reduces the number of free parameters to one single angle of the normal variance (SR_i).

Taking care of keeping a worst case scenario and assuming that the integration error increases with the integral itself, the centers of the Gaussians are set being identical maximizing the inner product. Furthermore, the center of the integration area is placed at the center of the Gaussians and the triangle is approximated by a slice. Since the error to be calculated is the ratio of the difference to the correct value, we can neglect the factors of the Gaussians (the output layer weights u_i, u_j), set the expansions d_i, d_j to 1 each and integrate over a slice of radius 1.

In figure 3.10 this reduced problem is drawn in Flatland. Integration is done over the dashed line $\{\hat{\mathbf{x}}\}$ for the approximation case and over the thick line $\{\mathbf{x}\}$ for the numerical integration. With the presented simplifications we

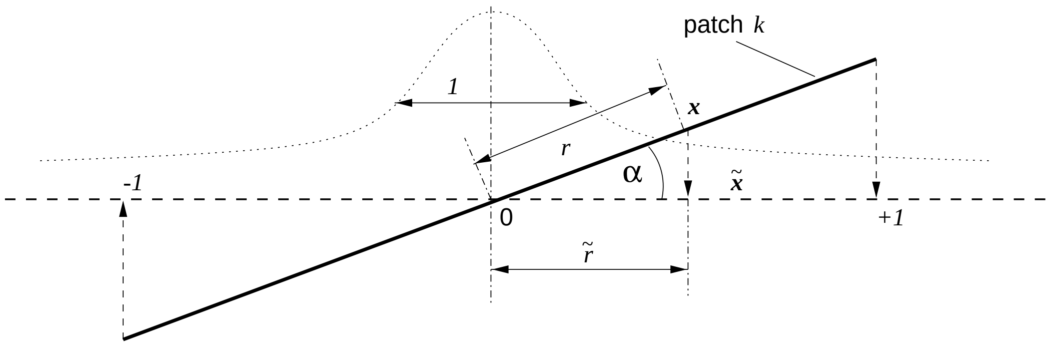


Figure 3.10: Principle scheme for integrating over the network geometry (thick line) compared to the approximate geometry (dashed line).

are able to create a function depending on one parameter α which stands for the angle over which a patch is rotated.

The area integral over the flat domain ($\{\hat{\mathbf{x}}\}$) of the three-dimensional environment is

$$\begin{aligned}\tilde{\mathcal{I}}_{\text{FI}} &= \pi \int_{-1}^{1} \left(e^{-2\hat{r}^2}\right)^2 \, d\hat{r} \\ &= \pi \int_{0}^{2} e^{-t^2} dt, \end{aligned} \tag{3.62}$$

and with $a = \sqrt{1 + \tan^2 \alpha}$ its exact value (integrating over $\{\mathbf{x}\}$)

$$\begin{aligned}\mathcal{I}_{\text{FI}} &= \pi \int_{-a}^{a} \left(e^{-2r^2}\right)^2 \, dr \\ &= \pi \int_{0}^{2a} e^{-t^2} dt. \end{aligned} \tag{3.63}$$

With equations (3.62) and (3.63), we define the resulting error at a certain cell i given by the function $\mathcal{E}_{\text{FI}} : \mathbb{R} \to \mathbb{R}$ of the roughness value at that cell, $\alpha = \text{FSR}_i$, like

$$\mathcal{E}_{\text{FI}}(\alpha) = (\tilde{\mathcal{I}}_{\text{FI}} - \mathcal{I}_{\text{FI}})^2 / \mathcal{I}_{\text{FI}}^2. \tag{3.64}$$

Transfer Coefficient Error on a Flat Domain

Again, we create a constellation which, in practical cases, shows an upper limit of the generated error. The shading network Gaussians are centered around two integration patches p and q (see figure 3.43 for example) and the transfer Gaussian (the Gaussian of the kernel network) is set to reaching from the center of Gaussian p to that of Gaussian q. Further, since the expansion of the kernel Gaussian might be quite arbitrary compared to the shading network Gaussians, we accept it being infinite which leads to a constant 1 instead of a Gaussian, and thus, the third Gaussian in equation (3.43) is omitted. The resulting integral approximation over the two assumed flat regions (use figure 3.10 for both patches p and q each by indexing $\mathbf{x}, \hat{\mathbf{x}}, r, \tilde{r}$ with p and q, respectively) is

$$\begin{aligned} \tilde{\mathcal{I}}_{\text{FT}} &= \pi^2 \int_{-1}^{1} \int_{-1}^{1} \left(e^{-\tilde{r}_p^2 - \tilde{r}_q^2} \right)^2 \, d\,\tilde{r}_p d\,\tilde{r}_q \\ &= 2\pi^2 \left(\int_0^{\sqrt{2}} e^{-t^2} dt \right)^2 , \end{aligned} \qquad (3.65)$$

and with $a = \sqrt{1 + \tan^2 \alpha}$ its exact solution

$$\begin{aligned} \mathcal{I}_{\text{FT}} &= \pi^2 \int_{-a}^{a} \int_{-a}^{a} \left(e^{-r_p^2 - r_q^2} \right)^2 \, d\,r_p d\,r_q \\ &= 2\pi^2 \left(\int_0^{\sqrt{2}a} e^{-t^2} dt \right)^2 . \end{aligned} \qquad (3.66)$$

The resulting worst case error for two cells p and q where we assume $\alpha = \max(\text{FSR}_p, \text{FSR}_q)$ is

$$\mathcal{E}_{\text{FT}}(\alpha) = (\tilde{\mathcal{I}}_{\text{FT}} - \mathcal{I}_{\text{FT}})^2 / \mathcal{I}_{\text{FT}}^2. \qquad (3.67)$$

Inner Product Error on an Infinite Network

See figure 3.11 as an example. Now, we are interested in the relation of the depth flag DF_i and the error which arises if the inner product is calculated approximately, i.e., if it exceeds the network boundaries. For simplicity a straight shading network boundary is assumed (fg. 3.11).

To simplify the integration task, again, the centers of the two Gaussians are set being identical, such that the integral becomes a maximum. The expansions d_i are set to 1. Considering figure 3.11, the error is generated by the integral taken at the outside region of the network (grey area), with the outside boundary assumed being a straight line at the distance DF_i (i.e., $d_i \cdot \text{DF}_i$, with $d_i = 1$, since the depth flag is proportional to the number of triangles to be passed to reach the boundary, and the diameter of a triangle approximates $\delta = \text{DF}_i$). The approximate integral at a cell i, which includes the integration over the area exceeding the boundaries of the network (integrating infinitely) is given by

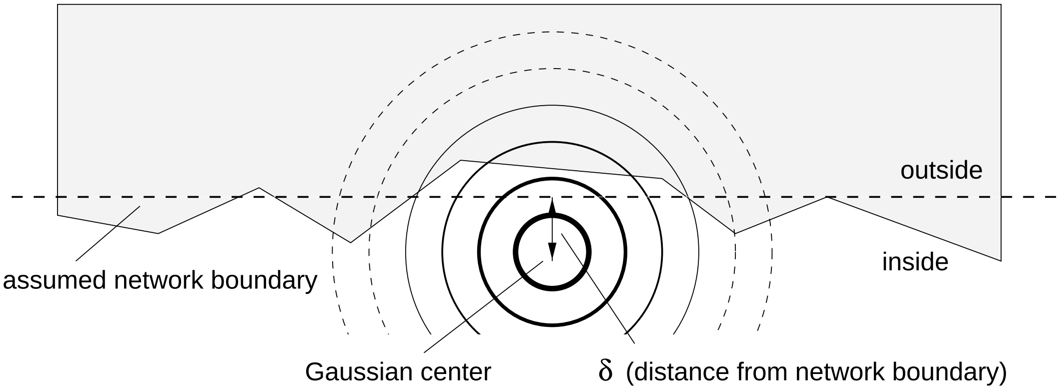


Figure 3.11: The range of a Gaussian is significantly exceeding the network boundary. For the approximate integration, the error generated at the outside region is calculated assuming a straight boundary (dashed line) with distance being equal to the depth flag of a Gaussian, which is multiplied by its expansion.

$$\begin{aligned}\mathcal{I}_{\text{DI}} &= \int_{\mathbb{R}^2} e^{-2(s^2+t^2)}\, ds\, dt \\ &= \frac{\pi}{2},\end{aligned} \tag{3.68}$$

and the absolute error is defined by the integral from the beginning of the boundary at the distance $\delta = \text{DF}_i$ to infinity,

$$\begin{aligned}\mathcal{I}_{\text{RI}} &= \int_{\delta}^{\infty} \int_{\mathbb{R}} e^{-2(s^2+t^2)}\, ds\, dt \\ &= \frac{\pi}{4}\left(1 - \frac{2}{\sqrt{\pi}} \int_0^{\delta\sqrt{2}} e^{-t^2} dt\right).\end{aligned} \tag{3.69}$$

From equations (3.68) and (3.69) an error term can be derived like

$$\mathcal{E}_{\text{DI}}(\delta) = \mathcal{I}_{\text{RI}}^2 / \mathcal{I}_{\text{DI}}^2. \tag{3.70}$$

Transfer Coefficient on an Infinite Network

The same assumptions like in the last paragraph (omitting the kernel Gaussian) is applied here, and leads to the approximation of the transfer coefficient like

$$\begin{aligned}\mathcal{I}_{\text{DT}} &= \int_{\mathbb{R}^4} e^{-s^2-t^2-u^2-v^2}\, ds\, dt\, du\, dv \\ &= \pi^2.\end{aligned} \tag{3.71}$$

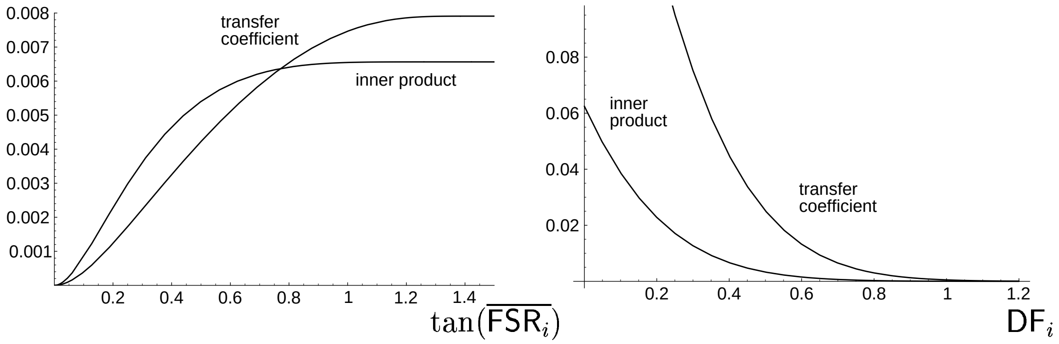


Figure 3.12: On the left side, the errors generated when calculating the inner product and the transfer coefficient approximately regarding a particular roughness value ($\tan alpha$ with α is the average angle over which the triangles in the near environment are rotated, the inner product curve is scaled by a factor of 300 to fit into the diagram). The same on the right side regarding a certain depth of a Gaussian in the shading network.

The absolute error if the Gaussians p and q exceed the shading network to the same degree is

$$\begin{aligned}\mathcal{I}_{\text{RT}} &= \int_{\delta}^{\infty}\int_{\mathbb{R}}\int_{\delta}^{\infty}\int_{\mathbb{R}} e^{-s^2-t^2-u^2-v^2}\, ds\, dt\, du\, dv \\ &= \frac{\pi^2}{4}\left(\frac{2}{\sqrt{\pi}}\int_0^{\delta} e^{-t^2} dt - 1\right)^2 \end{aligned} \tag{3.72}$$

and the relative error term follows like

$$\mathcal{E}_{\text{DT}}(\delta) = \mathcal{I}_{\text{RT}}^2 / \mathcal{I}_{\text{DT}}^2. \tag{3.73}$$

In figure 3.12 a plot of the four different error functions is shown. On the left side, the two functions $\mathcal{E}_{\text{FI}}$ (eq. 3.64) and $\mathcal{E}_{\text{FT}}$ (eq. 3.67) are drawn. They show the error for calculating the inner product and the transfer coefficient, respectively, related to a particular roughness value.

The right side of figure 3.12 shows the errors for approximating the inner product $\mathcal{E}_{\text{DI}}$ (eq. 3.70) and the transfer coefficient $\mathcal{E}_{\text{DT}}$ (eq. 3.73) in relation to a particular depth value in the shading network.

It can be seen that the errors are very small and the thresholds for the angle and the depth can be chosen very generously, leading to the avoidance of the costly numerical integration in almost all cases.

Chapter 4

Application

The content of this chapter is divided into two parts. First, we focus on the approximation of the kernel BK, second, the base function distribution derived from the shading network is discussed.

The parameter settings determined from the following experiments are described in section 4.4.1.

4.1 2D Kernel Approximation

The kernel of a three-dimensional environment is a 4-dimensional function over six-dimensional rays. For visualization reasons, in this section, we refer to two-dimensional environments where surfaces change to edges and the three-dimensional space to just an area on which the edges define the *flat* scene geometry. Thus, radiosity changes to a one-dimensional and the kernel to a two-dimensional function describing the interaction between two line points. Heckbert [26] called this constellation Flatland and introduced a parameterization of the edges which helps to visualize the kernel in an intuitive way. In this parameterization, the edges of the scene are represented by consecutive sub-intervals of the interval $[0, 1]$. A parameter value $s \in [0, 1]$ uniquely identifies an edge and a point on it. A pair of points as required for the kernel is given by two parameter values s and t. Thus, the kernel is a real function over the unit square in the s-t-coordinate system. The assignment of the edges to intervals is arbitrary.

According to that, the goal function to be approximated by an ISGCS is defined as two-dimensional function accessible by two parameters s and t. Its tasks are, first, it delivers the de-parameterized two-dimensional points $\mathbf{x}$ and $\mathbf{y}$ (see figure 1.1) related to s and t, second, it calculates the geometric term G and the visibility relation V, and third, it weighs them by the emittance E, defined with the scene definition, and a recent radiosity approximation value

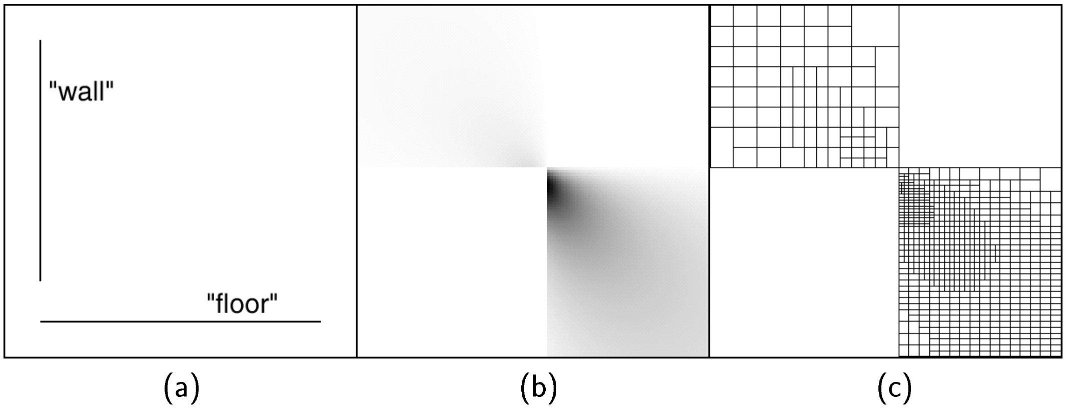


Figure 4.1: A two-dimensional example geometry (a), its kernel (b), and an example discretization done by the hierarchical radiosity approach.

$B^{(k)}(\mathbf{x})$ at $\mathbf{x}$ which is taken from the shading network (see also section 3.4). The resulting value $V(\mathbf{x},\mathbf{y})G(\mathbf{x},\mathbf{y})B^{(k)}(\mathbf{x})$ is taken as an i^{th} sample $s_i^{(k)}$ for the training of the kernel network. At the initial state and in the following examples we assume the pure emittance as radiosity approximation, which is equivalent to the first shot of energy.

In the following, we draw the two-dimensional kernels as black-and-white pictures. A bright grey value stands for a small kernel value, dark points denote high values.

We accomplished the following approximation tests with two different Flatland geometries shown in figure 4.1.a and figure 4.6.a. The kernel goal functions are drawn in figure 4.1.b and figure 4.6.b, respectively. Consider the high-valued area in figure 4.1.b in the center of the picture. It results from a high kernel value at the lower left corner of the geometry in figure 4.1.a and features the chosen parameterization, beginning with the value 1 at the upper end of the vertical edge (*"wall"*) and ending at 0 at the right end of the *"floor"* edge. The wall is defined as light source. The floor has also assigned a moderate amount of emitting light, mimicking the approach in an intermediate state where energy has already been reflected to be repropagated in the environment. Consider the upper left quarter of figure 4.1.b which is similar to its lower left. They differ in their brightness and orientation and might be interpreted like the portion of light energy emitted from the floor to the wall in contrast to the lower right quarter denoting the energy emitted from the light source to the floor only. The same holds for the geometry from figure 4.6.a, which has its light source defined at the ceiling edge. Consider the blocking edge compared to its respective sharp edged shape arising in figure 4.6.b.

Hierarchical Radiosity and its Kernel Representation

Figure 4.1.c and figure 4.6.c show the discretization of the kernel, if it is calculated by an hierarchical radiosity approach. We continue this chapter by comparing our results with the HR approach, due to the fact that it is most similar to the presented one since it also focuses on the search for a compact representation of the kernel explicitly (see also section 1.4). A constant hierarchical base has been used through all the tests.

An ideal compact representation of the radiosity kernel could theoretically be found by a standard compression algorithm applied to an instance of a kernel calculated to its full accuracy at “each single” domain location. Since this is generally not practicable, approximation methods which do not need to have full access to the complete goal function are required for radiosity. The compression method used in HR assumes a set of initial arbitrary average values over the kernel domain. Each integral of a portion of the domain defined by a pair of separate surfaces of the geometry is calculated and taken as average of the underlying kernel piece. This value is denoted the transfer coefficient of a *link* between the two patches. These *initial* links are predetermined and thus arbitrary. The HR approach starts on the initial subdivisions of the kernel and adaptively refines links into sets of finer links which approximate the underlying kernel domain on a more granular level.

A certain criterion, called *oracle*, determines if a particular link needs to be subdivided, and it is commonly based on the energy which is transferred by the according link (the value of the transfer coefficient), and/or on an estimation of the coherence of the kernel function inside of the domain related to that link. The degree of coherence is approximated by the number of those rays, randomly “shot” inside of that domain, which are occluded by a blocker between the corresponding pair of patches.

The oracle directly determines the resulting efficiency of the kernel, and it is obvious that the mentioned HR oracle is only suboptimal in its compression capabilities.

Figure 4.1.c shows an example discretization of the geometry where blocking effects do not arise. It can be observed how the oracle subdivides the kernel at the high-energy areas, at the corner of the edges. Each *block* in the kernel representation outlines a link of HR. At figure 4.6.c also the zones with a high transfer coefficient are represented by a finer link resolution, but additionally, links which “carry a blocking effect” have been detected by the coherency part of the oracle, and thus, they have been subdivided at a finer level.

Figure 4.2 displays an example solution of an HR approach of the geometry from figure 4.1.a. Figure 4.2.a is the computed discretization and figure 4.1.b the approximation of the kernel goal function, repeated in figure 4.1.c. For the

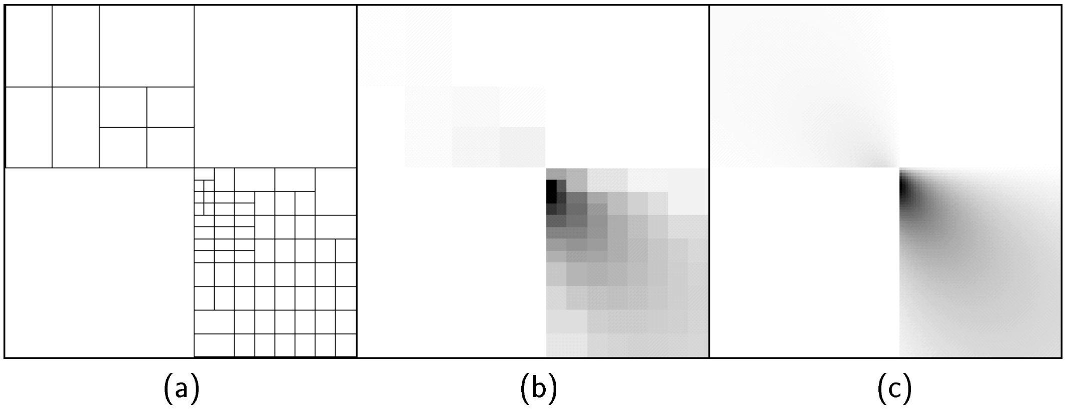


Figure 4.2: Discretization of the kernel from figure 4.1.a done by the hierarchical radiosity approach (a), the calculated transfer coefficients (b), and the goal function (c).

generation of figure 4.1.b the transfer coefficients of the links corresponding to each block are divided by the area (length) of the corresponding patches and drawn as grey values.

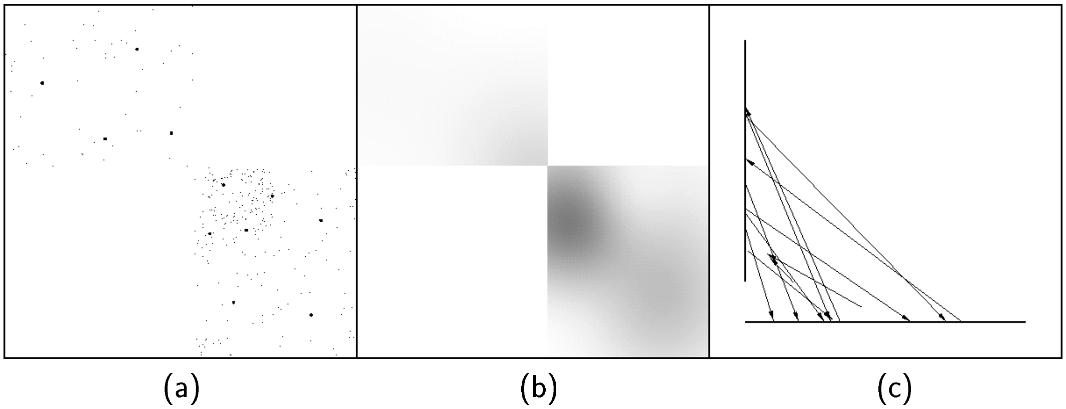


Figure 4.3: In the middle, the approximation of the kernel from figure 4.1.b by just 11 Gaussian RBFs. The RBFs are drawn in (a) as large points, and the generated samples as small points. (c) shows the distribution of RBFs drawn as rays into the geometry.

Approximation without Blockers

Figure 4.3 shows an example approximation by the presented ISGCS approach of the geometry from figure 4.1.a. On the left, the black points mark the centers of the radial basis functions of the kernel network — the reference rays. They

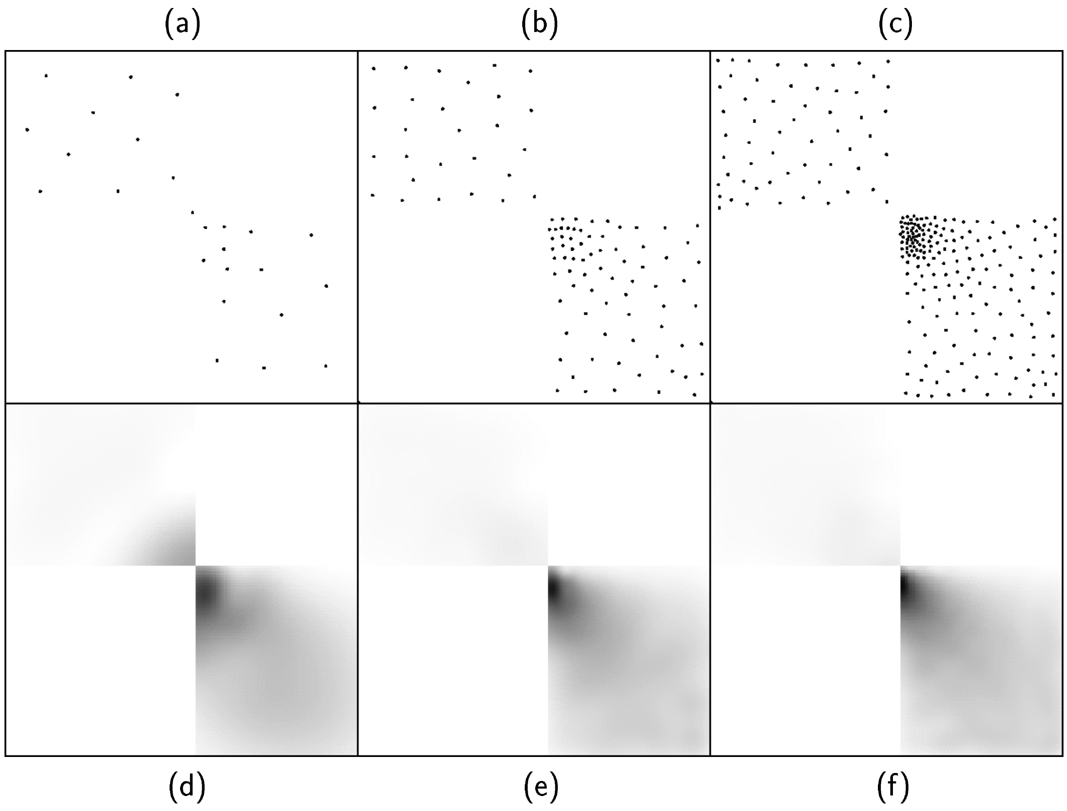


Figure 4.4: Snapshots of a complete learning process of the geometry from figure 4.1.a. At the top, the distribution of Gaussian RBFs from the beginning (a) to the end (c), at the bottom, the according plots of the kernel approximated by the kernel network.

stand for the discretization by the ISGCS approach and, in the following, they are compared with the links of the HR approach. In the middle of figure 4.3 the approximation of the kernel through the network is plotted. Figure 4.3.c shows a different representation of figure 4.3.a — the network function centers drawn as reference rays into the geometry.

Figure 4.4 presents approximations of the kernel through different numbers of reference links each, and these results are compared with the corresponding discretization by an equal number of HR links shown in figure 4.5.

The number of links for figures 4.4.a and 4.4.c and figures 4.5.a and 4.5.c from left to right are about 16, 100, and 500 and the resulting root mean squared error (RMS) has been calculated to 0.3%, 0.1%, and 0.07%, respectively, which lies about 30% below that of the HR approach for the equivalent number of links (see figure 4.8.a).

On a 180 Mhz, *MIPS R5000* CPU, the time required for the solutions from

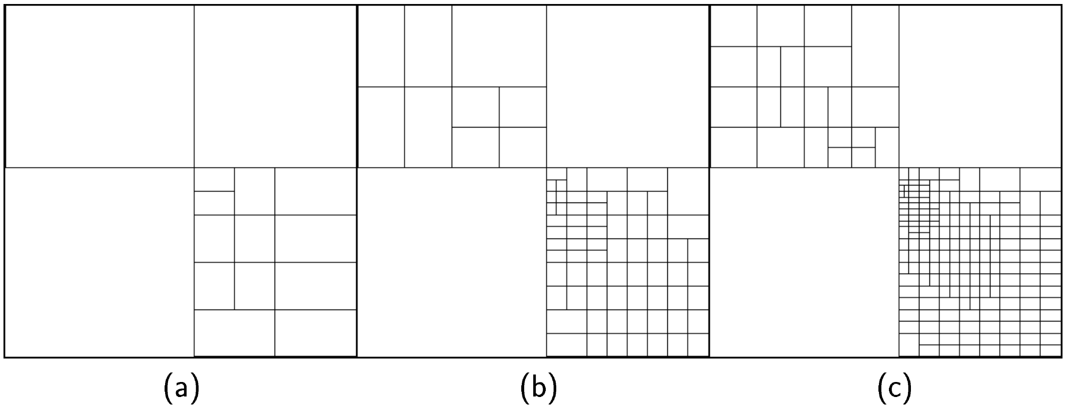


Figure 4.5: Discretization of the kernel from figure 4.1.a by the HR approach. Each block relates to a certain link.

figure 4.4 is about 10 seconds (figure 4.4.a), one minute (figure 4.4.b), and five minutes (figure 4.4.c). These results show this approach's facility of finding rough and fast solutions better than exact solutions of complex scenes. See also the discussion in section 4.4.

Links versus Samples

There are two obvious criteria which describe the quality of a radiosity approach adequately — the compactness of the generated approximation model (for example, the number of links), and the number of samples required for its generation. Whereas the first determines the amount of storage space needed for memorizing the kernel, the second shows its main effect on the computing time. In HR, the number of samples drawn from the geometry commonly is proportional to the number of generated links. Each link requires a set of about 10 to 200 test samples to determine the visibility information between the corresponding pair of patches.

The ISGCS algorithm creates reference cells which directly correspond to the number of underlying samples. Thus, there is a strong correspondence between the number of samples and the number of links for both approaches.

Tests have proven that a ratio of 10 of samples per generated links is sufficient in almost all cases. In ongoing comparisons we concentrate on the number of links only, keeping in mind that the number of required samples is not higher than that of the HR approach.

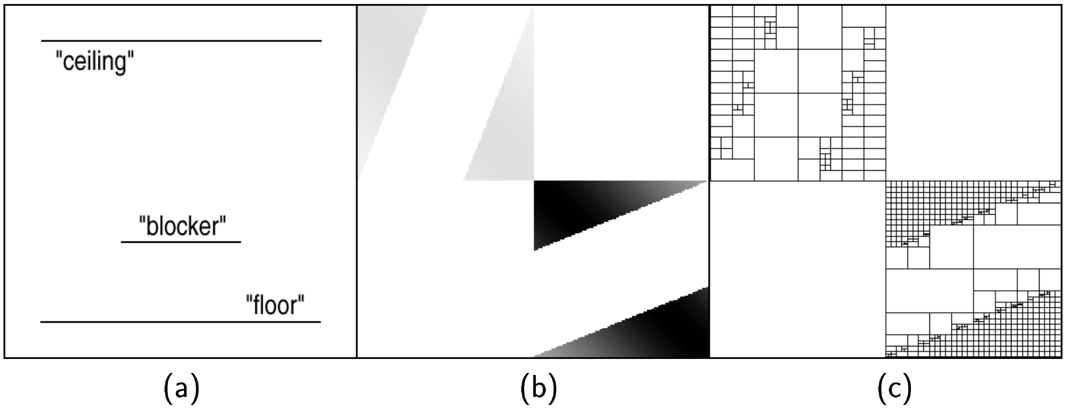


Figure 4.6: In (a), a geometry with a blocking edge between the light source (ceiling) and the floor. In the middle, the according kernel, on the right, an example HR discretization.

Approximation with Blockers

Approximating the kernel of a geometry which contains a blocker shows sightly different results. In figure 4.6 it can be observed that the shape of the kernel is characterized by a sharp edged band coming from the blocker effects between the light source, the ceiling patch, and the floor. Approximating this band requires a high resolution of reference cells or HR links as, for example, figure 4.6.c shows.

Figure 4.7 shows a result for the ISGCS approximation. Remarkable are two observations. First, to get a moderate approximation accuracy, in this case, just 11 links are sufficient (see figures 4.7.a and 4.7.d). The approximation error for this compact representation is half of that for the equal number of links in HR.

The described results are summarized in the diagrams of figure 4.8. On the left hand, the comparison between the first geometry is plotted, on the right hand, the comparison related to the blocker geometry. Besides the approximation capability, the fast convergence is remarkable even for a small number of links (see also figure 4.7.d). In early stages of the computation, the error lies significantly below that of the HR approach (up to 60%). This proves the practicability of the algorithm especially for fast initial approximations of the solution. Even this fact emphasizes the advantages of this work concerning huge geometries. Execution times go from a low of ten seconds for few cells on the left side of the diagrams up to about 20 minutes for the final solution with about 800 cells, on a 180 Mhz, *MIPS R5000* CPU.

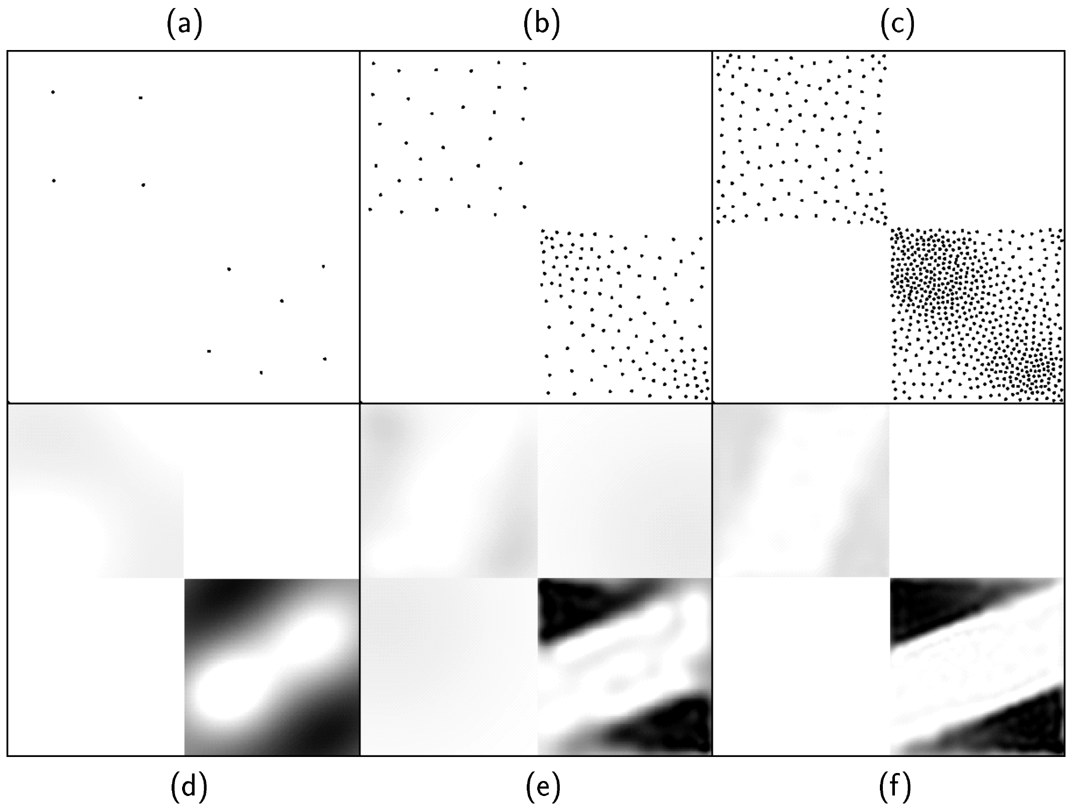


Figure 4.7: Snapshots of a complete learning process of the geometry from figure 4.6.a. At the top, the distribution of Gaussian RBFs from the beginning (a) to the end (c), at the bottom, the according plots of the kernel approximated by the kernel network. Execution times from the left to the right side are 5 seconds, 120 seconds, and 20 minutes on a 180 Mhz, *MIPS R5000* CPU.

4.2 2D Shading

Surfaces in Flatland are edges, and the one-dimensional shading network can be seen like a *string* of Gaussian radial base functions. Figure 4.9 shows the floor patch of figure 4.1.a with the centers of links/RBFs drawn as short bar. Consider the distribution of them. The bottom line has been generated by a typical HR subdivision. The middle line shows the distribution of the end points of reference rays of the kernel approximation. It can be observed that, although this kind of distribution might be the best solution of matching the results of the kernel training onto the surfaces, the distribution lacks of regularity which makes it difficult to get a reasonable function base like mentioned in section 3.2. Moreover, there is no overlayed structural information available, like the direct neighborhood of each center which could define the expansion

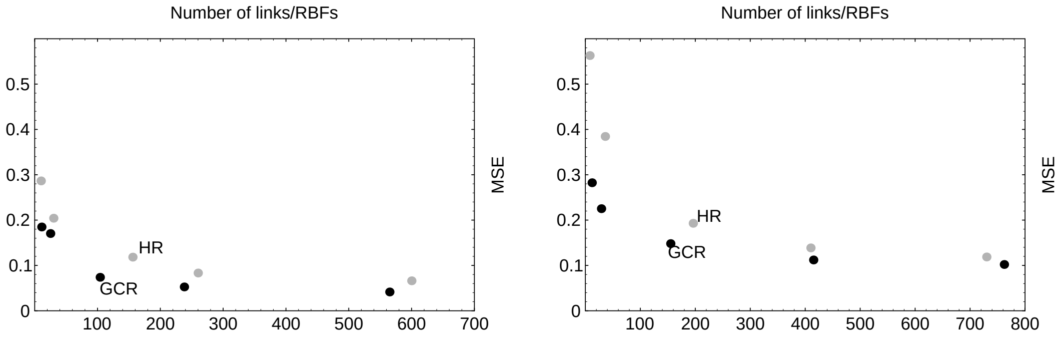


Figure 4.8: Comparison of the kernel MSEs generated with the ISGCS approach and with the HR approach in Flatland. (a) shows the kernel approximation error of the geometry from figure 4.1.a (no blockers), (b) that from figure 4.6.a (one blocker).

of the corresponding RBF.

In contrast to that, the top line displays the distribution of the shading network, which is developed based on the samples generated by training the kernel network. It is much more regular and it also accounts for the energy distribution. The resolution increases going from right to left from lower to higher energy (see the high energy corner in figure 4.1.b).

It can also be observed that the distribution on the right side is slightly higher although energy decreases here. This is related to the boundary matching of the ISGCS network explained in section 2.5.

Figure 4.10 shows the results for the distribution of radiosity bases from the shading network concerning the second geometry which contains a blocker. Again, only the floor edge is drawn, and the blocker boundaries are marked with two horizontal lines. Remarkable is the matching of the distribution with the boundaries of the blocker. According to a higher ray end point distribution,

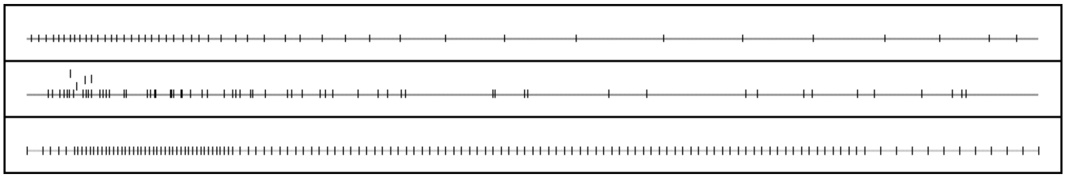


Figure 4.9: The floor edge of the geometry from figure 4.1.a (without blocker), from top to bottom, the center distribution of the shading network, the kernel ray end point distribution, and the classical subdivision derived from the HR kernel.

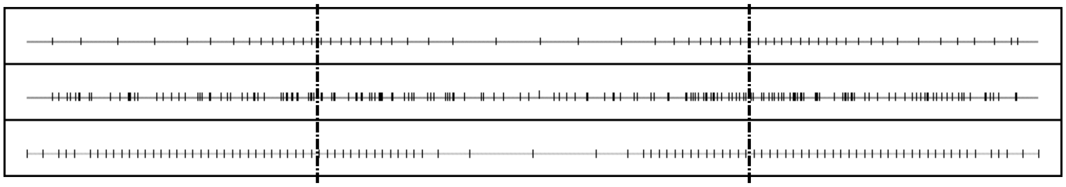


Figure 4.10: The floor edge of the geometry from figure 4.6.a (with blocker boundaries marked by the horizontal bars), from top to bottom, the center distribution of the shading network, the kernel ray end point distribution, and the classical subdivision derived from the HR kernel.

drawn in the middle, the resolution of the shading bases is more granular than at the shadow positions of the patch. Although this can also be observed at the bottom of the figure for the HR solution, here, the resolution is much more restricted to the kernel shape and matches the final shading requirements slightly worse.

4.3 3D

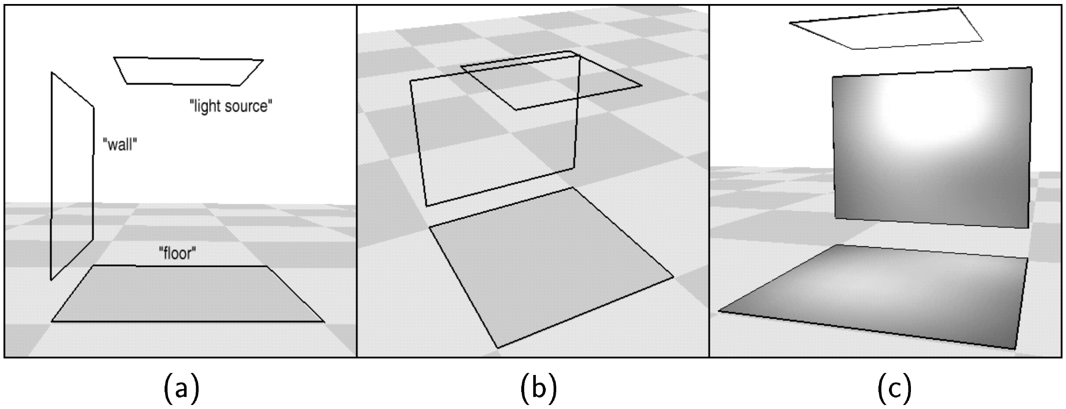


Figure 4.11: Example scene (a,b) and the radiosity solution (c) calculated by the ISGCS approach.

Sections 4.1 and 4.2 have proved this work's capabilities in case of two-dimensional environments. Kernel training results have been exposed through two-dimensional pictures of grey shades. Going from Flatland to realistic three-dimensional scenes, the kernel changes to a four-dimensional function of two points in three-dimensional space, and since an adequate two-dimensional visualization technique is hardly available, we mainly discuss the approximation error which arises during the kernel training and the triangulation derived from the shading network representation.

Figure 4.11 shows a test geometry of two perpendicular planes, a "wall" patch and a "floor" which are illuminated by a light source from the top. Figure 4.11.c shows a solution calculated with the ISGCS approach. Instead of visualizing the kernel representation, figure 4.12.a shows the distribution of receiver points of the sample rays which are generated during the kernel training. It is proportional to the distribution of reference rays in the kernel network (not displayed) and proves that end points of rays focus on locations with a high energy distribution.

As mentioned, the end points of the kernel samples are used for the training of the shading network, since these pay most attention to the final intensity bleeding on the patches. An example network like in figure 4.12.b is generated simultaneously to the kernel training. Figure 4.11.c shows the shading which is generated by a final triangulation path explained in the following section.

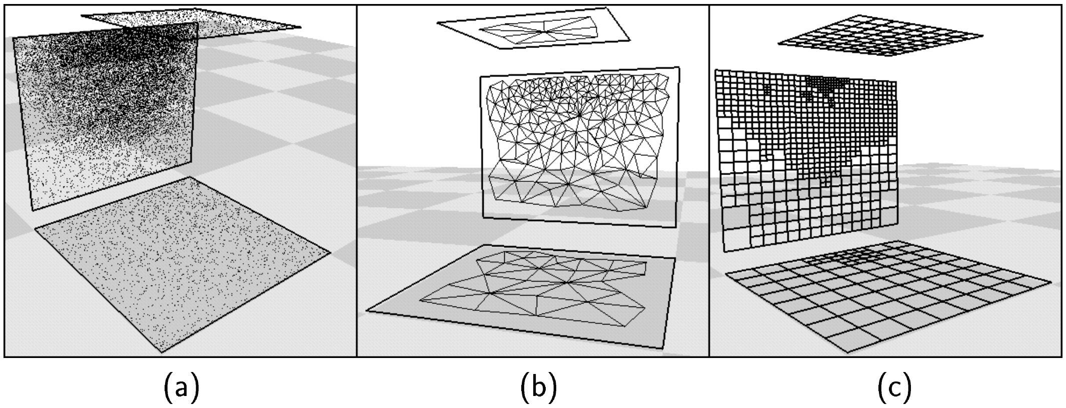


Figure 4.12: Kernel training sample distribution (a), shading network topology (b), and the re-triangulation (c) of the geometry from figure 4.11.

4.3.1 Triangulation

Since the shading network model generally forms an adequate triangulation of the underlying data set (see section 2.2), it seems obvious to apply it directly as triangulation of the surface space. Nevertheless, considering figure 4.12.b, it is quite clear where the problems of such an algorithm would arise. Since the network shape is generally an approximation of the underlying domain, it is not suitable for a realistic rendering of the geometry. Figure 4.12.b exposes the problem of the under-representation of boundary regions by the network's clustering capabilities referring to section 2.5. Further disadvantages are described below. As result, a re-triangulation like in figure 4.12.c is needed,

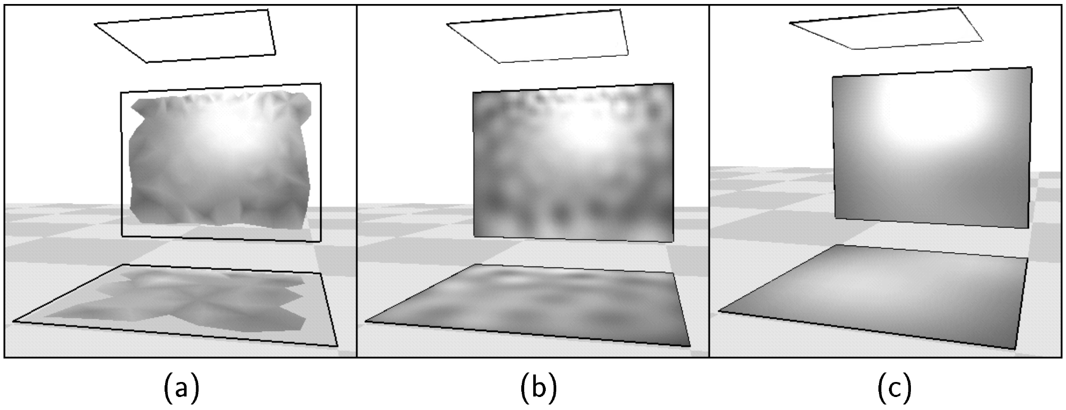


Figure 4.13: In (a), the linear radiosity function base is used for triangulation and calculation of intensity values on the shading network cell centers (vertices in figure 4.12.b). In (b), vertex intensities generated by re-triangulation are calculated from the radiosity base. (c) neglects the base and computes the vertex intensities through a final "collecting" step through the kernel approximation model.

which adopts the discretization determined by the shading network. Consider the following algorithm.

The original surfaces are subdivided recursively. In case of a quadrangle, its center point is calculated and four new quads containing the new point are generated, whereas the originating one is deleted. The algorithm is repeated with the four newly generated quads until the size of a subpatch approximately equals the size of the simplex of the shading network which lies closest to it. In this work, a reference point of a particular subpatch is determined and a reference edge like the diagonal is compared to the average edge length of that cell which matches best the reference point (the best matching unit, see section 2.2).

In other words, the patch is subdivided as long as it is larger than the underlying simplices of the shading network.

4.3.2 Shading

The radiosity base presented in the preceding chapters serves for the light flow FEM simulation and for the updating of the samples selected from the geometry. For rendering purposes, it seems obvious to derive the radiosity directly from this base creating the possibility of calculating arbitrary points on the surfaces.

Figure 4.13.a shows such a solution. The radiosity is evaluated at each cell center and the network is directly used for the triangulation. In contrast, figure 4.13.b shows the results after the described re-triangulation. Radiosity values

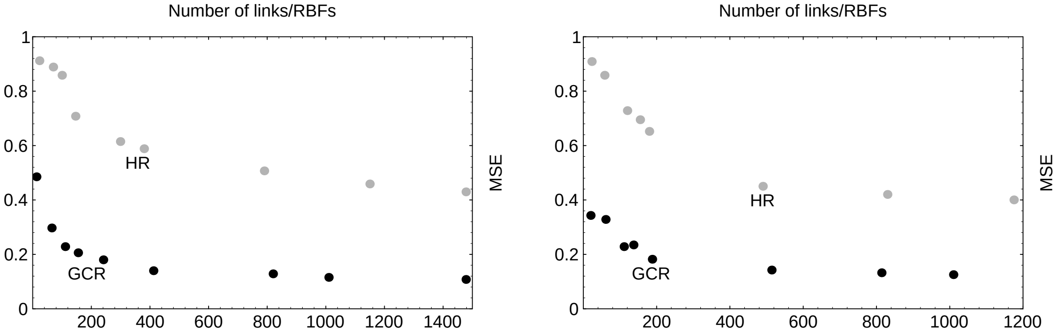


Figure 4.14: Approximation accuracy of the ISGCS kernel model if compared with the kernel of the HR approach for equal numbers of links/RBFs and for two different geometries without blockers.

are computed at each of the generated vertices, and expose, that, although the base's accuracy in terms of a particular error criterion might be sufficient for the FEM, it is not well suited to satisfy one's demands of a realistic color bleeding. Difficulties arise, on the one hand, with the small distribution of bases at certain locations, on the other hand, with the infinite support of each of the base components leading to the downy look of the radiosity shading.

Thus, for the final rendering, we neglect the base and instead propagate the energy once again through the kernel approximation onto each of the vertices, resulting in figure 4.13.c. The method is common to the most classical radiosity approaches and it is known as a final collection path to the geometry vertices.

The diagram on the left hand side of figure 4.14 exposes the calculated error in approximating the kernel network if compared with a hierarchical radiosity approach. Remarkable is the small error even in case of few cells (below 200). The right hand diagram shows the error for another example model of two perpendicular planes, one of them defined as a light source. Execution times go from a low of ten seconds for few cells on the left side of the diagrams up to about 42 minutes for the final solution with about 1400 cells, on a 180 Mhz, *MIPS R5000* CPU.

We follow up with two example geometries containing two blockers, each shown on top of figure 4.15 and in figures 4.19.a and 4.19.b. Figure 4.15.d gives an example of the network cell distribution, figure 4.15.e the triangulation, and figure 4.15.f the final shading of the solution.

The errors compared with the HR approach are plotted in figure 4.16. Execution times are nearly similar to that from figure 4.14, since they mainly depend on the number of generated links. The size of the error is about half

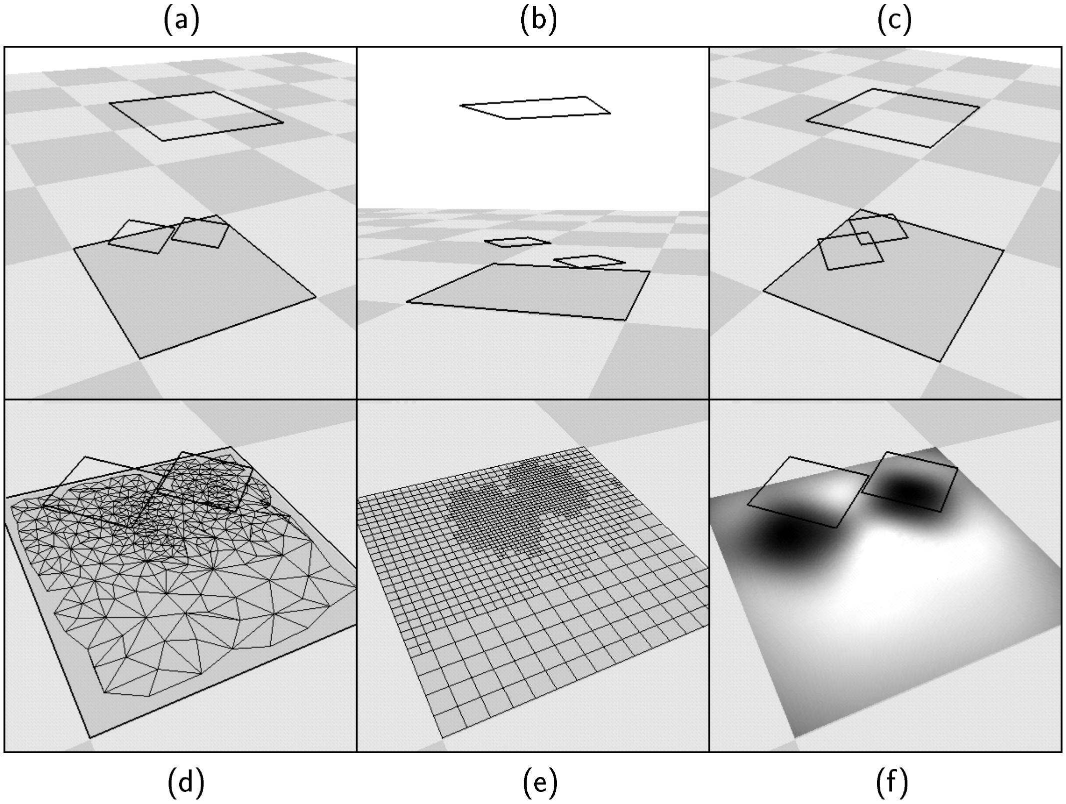


Figure 4.15: A geometry with two blockers (a-c). (d-f) show the shading network topology, the triangulation, and the final radiosity solution, respectively.

of that of the HR approach.

4.4 Results, Discussion

4.4.1 Parameter Settings

In figure 4.17, we list some typical parameters which were used for the network training in the preceeding sections. It has been shown that suitable parameters often depend on the geometry under investigation. Choosing “slightly” wrong parameters led to a slower learning rate, in almost all cases. Of course, parameters which differ strongly from the listed ones led to a collapse of the network training, generally, i.e., in these cases, the ISGCS is not longer able to create a consistent structure over the input sample domain. In [8], these effects have been investigated in several experiments for the two-dimensional case. The parameters' ranges listed in figure 4.17 can be taken as robust clues.

Additionally, consider the following application rules.

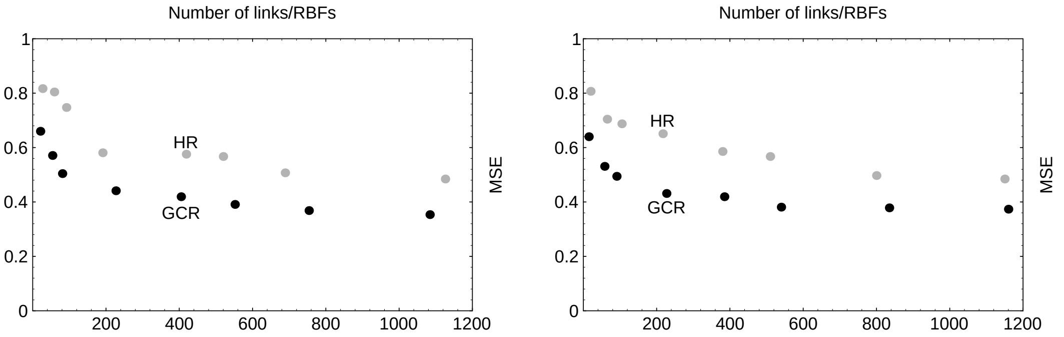


Figure 4.16: Approximation accuracy of the ISGCS kernel model compared with the kernel of the HR approach for equal numbers of links/RBFs and for two different geometries with blockers.

- Taking the number of cells which have been inserted in the kernel network as a *termination condition* has shown practical benefits in almost all cases.
- The *insertion counter of cells* λ in the kernel and the shading network were always kept equal.
- The *triggering of the FEM* integration calculations is done after each completed insertion of about 20 to 50 cells in the kernel network.
- Thresholds for the roughness of the network and the depth of a cell in the network to decide if an analytical integration is sufficient can be chosen generously (0.4 and 1, respectively), due to the small arising error (see figure 3.12).

4.4.2 Practicability, Problems

We have been proving the validity of the whole approach, but, admittedly, there are some difficulties in applying the method to arbitrary geometries. We summarize these handicaps, in the following, then, in section 4.4.3, we summarize the error sources of this approach.

Topology Preserving Networks

Consider figure 4.18. It shows the problems which arise from the adaption of a two-dimensional network to the surfaces of the scene. Figure 4.18.a is an example for the case where network cells are not deleted at the right time (see

Parameter	min	max	Description	Page ref.
ϵ_{BMU}	0.01	0.05	Weight vector adaption strength for a BMU.	21
$\epsilon_{\mathcal{N}}$	0.001	0.009	Weight vector adaption strength for the direct neighbors of a BMU.	23
α	0.01	0.05	Decreasing rate of the resource terms τ_i^{SL} and τ_i^{USL}.	29
η	0.01	0.1	Learning rate of the output layer.	22
ω	0.4%	0.6%	Average resource percentage for the threshold for being a high resource cell.	32
ε	0.001	0.01	Deletion threshold for the resource term τ_i of a cell c_i.	31
λ	300	600	Number of iterations before a new cell is inserted.	31
κ	400	1000	Number of iterations before a possible deletion of cells is tested.	31
φ	0.4	0.6	Threshold which decides if an input sample ξ is inside of a network. It is tested against the network activation Υ_ξ.	32

Figure 4.17: Recommended parameter settings.

section 2.3). The network is spread over the gap between the wall and the floor. This happens accidently depending on not sensitively chosen training parameters. The error which is added to the radiosity base is exposed in figure 4.18.b.

The example also points to another problem. There is no sufficient criterion for modeling the surface boundaries of the goal geometry. The model virtually "sees" a distribution of points in space from which no information about the affiliation to surfaces can be derived. On the one hand, this fact is desirable for curved surfaces or patch clusters which describe a unique surface, on the other hand, edges in the geometry are not sufficiently accounted for. Some sort of "breaking operator" which splits the network into several independent pieces at these edges would be necessary.

Generally, the topology preserving facilities of the ISGCS network hinders a completely free adaption to the underlying sample distribution. Even the training parameters essentially influence the final approximation result, moreover, in some cases, they depend on the scene geometry.

These problems are obviously visible for the two-dimensional shading net-

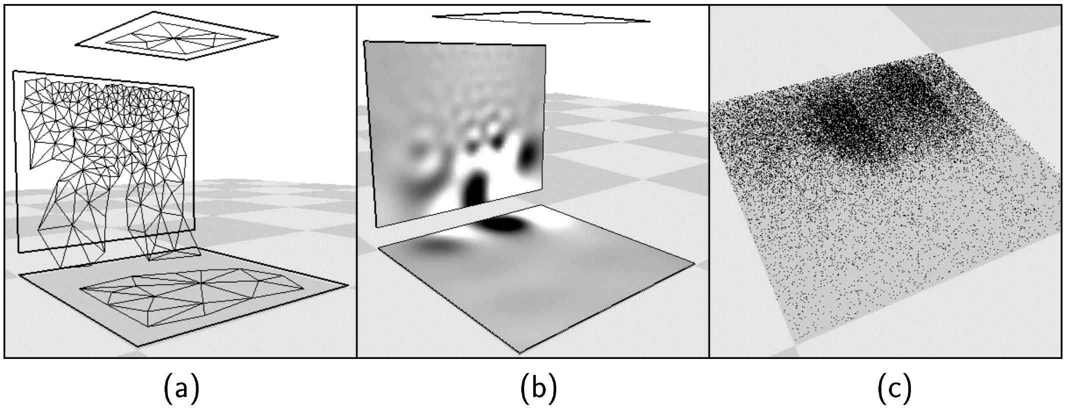


Figure 4.18: Example problems arising from the ISGCS approach, which originate the topology preserving features of the ISGCS network.

work but they also arise in the four-dimensional kernel network observable through the slight discontinuity of the sample distribution, for example, in figure 4.18.c.

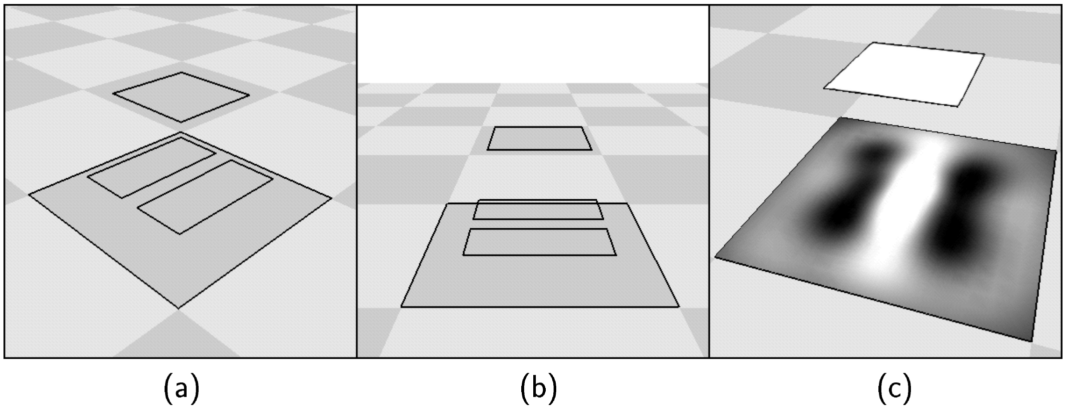


Figure 4.19: Example for the lack of approximation capability for a highly varying kernel (in case of blocking effects).

Kernel Approximation Accuracy

As mentioned, the presented supervised function approximation model shows its drawbacks when modeling functions with sharp variations. Thus, shadow generating blocking effects are less likely to be represented through an RBF network. The radiosity solution (fg. 4.19.c) of the geometry described in figures 4.19.a and 4.19.b visualizes this effect. Here, the virtually sharp edges are bleary and do not model the expected shadow boundaries sufficiently.

Time Resources, Complexity

The kernel approximation clearly outperforms the HR approach in its accuracy for a certain number of network cells versus the same number of links. The complexity of the approach equals $O(z \cdot \hat{z}^2)$ with $\hat{z}$ as the number of cells in the shading network and z as the size of the kernel network.

On the one hand, this is worse if compared with GR and HR, on the other hand, by the avoidance of initial linking through clustering, an outstanding small amount of time for a fast rough solution with up to several hundreds of network cells even for huge geometries is possible. In this case, it outperforms classical approaches — rough solutions even for millions of polygons are possible in a few minutes. These solutions can be taken as a first estimate but also to support classical approaches with information about clusters in the geometry or to deliver a basic triangulation usable as discretization for an exact classical approach.

Besides the issues of complexity, the time needed to calculate a rough solution, where the networks consist of up to 100 cells, lies in a range of 10 seconds to few minutes. This is comparable with the HR approach for the presented test scenes. Time resources rise drastically if an exact solution or a more complex scene is required to be calculated. Basically, this results from the network training time and is a common problem of almost all approaches using neural networks.

4.4.3 Error Summary

Errors in the computations of this approach arise in four ways — first, in the approximation of the kernel by the kernel network, second, in the usage of an additional surface function base, third, in the short-cut of using the analytical integration of the transfer coefficients and the inner products instead of the numerical exact solution, and fourth, in the difference between the scene geometry and the SN geometry which mimics the scene geometry

The first and the second errors seem to be identical with classical approaches, where, on the one hand, a surface base generates an error, due to the inaccuracy of modeling the radiosity function, and on the other hand, there is also an defective approximation of the kernel by the tensor product base of the surface base.

In fact, classical approaches do not have an additional error infusion since the kernel base is the tensor product base of the radiosity base, and thus, transfer coefficients exchange energy between a pair of radiosity base components without a loss of accuracy[1].

[1] except for the error arising from the numerical calculation of the transfer coefficients

Kernel Error

The error of the kernel approximation, in this work, has already been documented in, for example, diagrams of figure 4.8 and of figures 4.14 and 4.16. It converges (which has been shown in section 3.5) to an MSE between 10% and 40% depending on the presented test geometries, i.e., on the scene complexity. In these simple test cases, the presented approach outperforms, for example, the HR approach, significantly, whose error amount is about as twice as large.

Surface Base Error

Whereas in classical approaches, no additional error is introduced by the application of a surface base, this work uses a different function base to represent the radiosity on the surfaces — the Gaussians of the shading network (SN). Assuming the SN itself as the geometry under investigation[2], we add the error of the Gaussian base in representing a constant as a bias to all further considerations[3]. Figure 3.4 demonstrates that the surface base's error is about 10%.

Integration Error

An integration error is introduced by assuming the analytical solution instead of the numerical solution when calculating the inner products and the transfer coefficients of the radiosity base. By assuming the decision technique from section 3.9.4, which avoids accepting the analytical solution if it is inaccurate, the creation of an integration error is limited to only few cases where it is neglectable small (see diagrams from figure 3.12).

Geometry Error

The geometry error is characterized by two effects.

- The case that the shading network parts do not support the given geometry (for example at geometry rifts where the network covers domains which virtually do not exist), or the case that the network models sharp edges of the given geometry $\mathcal{S}$ not exactly.

- The inaccuracy stemming from the fact that we assume the shading network domain $\hat{\mathcal{S}}$ as geometry under investigation, and then, for final

[2] and thus, ignoring the error which comes from the fact that the initial geometry definition is not identical to the geometry created by the shading network

[3] This is a rough approximation of the error of a Gaussian base, but considering the following more fatal errors, this approximation should be reasonable.

shading the SN is evaluated at the originating geometry surfaces $\mathcal{S}$. In other words, virtually, the shading base is only valid for points lying on $\hat{\mathcal{S}}$.

Overall Error

On the one hand, accumulating the overall effects of these errors is difficult, since it is hardly possible to analyze the geometry error in a reasonable way. On the other hand, the geometry error is the one with the biggest effect on the intuitive quality of the whole approach. Assumptions can not be accomplished adequately when using large geometries, moreover, from practical experiences, we assume the geometry error to be very large if the geometries become complex. This comes from the fact already mentioned that the ISGCS are hardly capable of mimicking geometric objects in a sufficient accuracy.

The effects will be visible in the case of discontinuous function shapes, like shadows[4]. Besides its physical correct simulation of the light flow, people commonly judge quality through looking at sharp shadows, and the impression of the error occurring at these discontinuous locations is intuitively essentially higher — the physical correctness does not count in such cases.

Thus, we refer to the general practicability discussion on the subsequent pages, and summarize here the known errors by applying the technique from section 3.5, i.e., from [2]. Neglecting the geometry error, we add the described surface base errors to the kernel approximation error[5] which is increased by 50% at average and thus, nevertheless, it lies significantly below that of the HR approach.

4.4.4 Practicability, Benefits

Despite the presented issues from the preceeding paragraphs of a large geometric error, problems in calculating complex geometries, and long execution times, we consider the following benefits.

Main advantage of the ISGCS approach is its flexibility and generality arising from the development of an approximation method which is derived from a neural network algorithm. The two networks, for the kernel representation and for the discretization of the radiosity function, are trained independently from each other.

This is generally different if compared to classical approaches, since there, there exists a fixed connection between the kernel and the radiosity representation — via the approximation of the kernel by the radiosity base components'

[4] in the kernel and in the surface function representation

[5] the perturbation error in [2]

tensor products. Thus, in the presented approach, there arise a novel topological freedom in developing the discretization of the radiosity equation.

Another practical benefit arises from the capability of easily predetermining the up-running computing resources by simply choosing limits for the network sizes and letting the algorithm run until these limits are hit. The aspired maximum kernel storage or the number of polygons which should finally be created (for example to match the limitations of a real time application) may be criteria. This is an advantage if compared to classical approaches where, due to the indirect effects of the available parameters, it mostly has been difficult to determine the resulting resources easily. In HR, for example, an energy criterion for the kernel representation determines the final number of polygons for the shading.

Finally, advantages arise from the fact that the kernel model is memorized in the network representation. This enables an easy and fast reiteration of both networks. A quick adaption is possible and provides the possibility for incrementally retraining slightly changed geometries like dynamical scenes. Rays which are effected by changes in the geometry trigger a local resampling.

Chapter 5

Conclusion

Growing cells radiosity (GCR) presents an approach which combines Monte Carlo like sampling with the finite element scheme for solving the radiosity integral equation. A new artificial neural network is developed which is capable of efficient function approximation, and it is adapted to mimic finite elements through its topology, suitable for an FEM radiosity algorithm.

Emphasis is laid on abstraction from geometric objects, generally, to be able to freely develop a compact storage model of the light flow through a neural network representation. Through examining the geometry by sampling instead of accounting for discrete polygons, the neural network's outstanding clustering and function approximation facilities are utilized for an efficient representation of the energy flow in the scene. This representation is instantiated by a linear function base of six-dimensional Gaussian radial basis functions.

A second network of similar type approximates the scene geometry. From the internal structure of both networks a finite element representation is derived and an FEM computation executed simultaneously, in order to update the radiosity during the training process. Due to the homogeneous structure of both networks, the integration operations required for the FEM calculation are accomplished analytically.

During the learning process and the concurrent FEM computations the geometry network approximates the actual results of the radiosity computation in terms of a linear base of three-dimensional Gaussian radial basis functions with local expansion but infinite support of the geometry.

Advantages. The algorithm runs completely without accounting for surfaces or geometric objects at all. Based on the radiosity network, a triangulation of the scene can finally be derived in a straight-forward manner. Thus, avoidance of initial linking and an ideal clustering of the light flow arise. It enables, principally, to calculate geometries of arbitrary sizes similar to pure Monte Carlo approaches. Additionally, due to the interpolation facilities of the approximation networks, the approach is capable of calculating fast, rough solutions without the classical noise effects, and moreover, an efficient non-redundant importance sampling method is presented.

The final representation of the radiosity by a base of Gaussian radial basis functions enables a smooth color bleeding on the surfaces with a very small number of base components.

The kernel and the representation of the radiosity function are stored in a flexible, adaptive data structure of a neural network. If compared to conventional approaches, a new degree of topological freedom has been shown by the iterative development of the discretization of the radiosity equation. There are no limitations in modeling the geometry through surface base components, like there are no restrictions in approximating the kernel function, due to the fact that the usual connection of both representations via the tensor product base is not existent.

The flexible data structure even enables a reiteration of an already existing solution and thus provides the possibility for an *incremental radiosity approach*.

Problems. One the one hand, like almost all adaptive approximation approaches of this kind, the algorithm ideally accounts for smooth transitions of the goal function. In contrast, high variances in the light flow which result in sharply bounded shadows can only be represented with a high amount of computing resources. On the other hand, fast and rough solutions are easily possible, and thus, one way of benefitting from this approach could be to make an analysis about the training behavior and the preliminary results to support classical approaches with information which helps making preliminary decisions concerning an efficient meshing, for example.

Additionally (concluding from the section 4.4.3), the geometry approximation by a neural network is questionable due to the inflexibility of the network in case of modeling virtual geometries, generally. Thus, this work should be seen from a more or less idealistic, academic point of view. Practical applications for realistic geometries are not recommended even due to the long training times of the principal ISGCS network.

In the author's opinion, the main attraction of the presented scheme is its generality — two general goal functions (the kernel and the radiosity) are approximated by a general artificial neural network, and thus, the approach is able to profit from the typical outstanding neural network features of analyzing arbitrary data distributions. It results in an radiosity approach which is independent from the geometry and from the definition of fixed surface base functions — these are generated in a fluent, smooth adaption of two neural networks from which later the final radiosity is derived. Thus, this approach is new in its kind of thinking of geometry, generally, it turns the light flow in a self-organizing, esthetic conglomerate of neural networks which automatically and autonomously learn the complex geometric relationships — there is is no need for adapting and converting the initial geometry or for making assumptions like discontinuity meshing to keep computing resources small.

Growing cells radiosity, in its entirety, can be seen like a special designed neural network for solving the radiosity problem in a self-organizing, automatic manner.

Appendix A

Mathematical Derivations

A.1 Integrating over $\mathbb{R}^2$ in $\mathbb{R}^3$

In sections 3.6.1 and 3.8, integration over a two-dimensional function f is required. According to the general substitution rule for a one-dimensional function $f : \mathbb{R} \to \mathbb{R}$ with a parameterization $g : \mathbb{R} \to \mathbb{R}$,

$$\int_{x_1}^{x_2} f(x)\,dx = \int_{g^{-1}(x_1)}^{g^{-1}(x_2)} f(g(s))\,\frac{\partial g}{\partial s}\,ds,$$

there exists its two-dimensional analog,

$$\int_{x_1}^{x_2}\int_{y_1}^{y_2} f(x,y)\,dx\,dy = \int_{g^{-1}(x_1)}^{g^{-1}(x_2)}\int_{h^{-1}(y_1)}^{h^{-1}(y_2)} f(g(s,t),h(s,t))\,\frac{\partial(g,h)}{\partial(s,t)}\,ds\,dt \tag{A.1}$$

with $f : \mathbb{R}^2 \to \mathbb{R}$, $g, h : \mathbb{R} \to \mathbb{R}$, and the *Jacobian* [10, 16]

$$\frac{\partial(g,h)}{\partial(s,t)} = \begin{vmatrix} \frac{\partial g}{\partial s} & \frac{\partial g}{\partial t} \\ \frac{\partial h}{\partial s} & \frac{\partial h}{\partial t} \end{vmatrix}. \tag{A.2}$$

The required bijectivity of the substitution (eq. A.1) implies a quadratic shape of the Jacobian, which is not given in this work. Here, a two-dimensional function f is defined on an area in $\mathbb{R}^3$ described by three separate parameterization terms $x_1, x_2, x_3 : \mathbb{R}^2 \to \mathbb{R}$, forming a vector parameterization $\mathbf{x} : \mathbb{R}^2 \to \mathbb{R}^3$, $\mathbf{x}(s,t) = (x_1(s,t), x_2(s,t), x_3(s,t))$. The Jacobian of this parameterization, $\frac{\partial \mathbf{x}}{\partial(s,t)} = \frac{\partial(x_1,x_2,x_3)}{\partial(s,t)}$, would be $\in \mathbb{R}^3 \times \mathbb{R}^2$.

To create a quadratic form of the Jacobian, one way is to introduce a third variable u, and a replacement of the parameterization $\mathbf{x}$ by $\mathbf{X} : \mathbb{R}^3 \to \mathbb{R}^3$, defined like

$$\mathbf{X} = \mathbf{x} + u \, \frac{\frac{\partial \mathbf{x}}{\partial s} \times \frac{\partial \mathbf{x}}{\partial t}}{\left\| \frac{\partial \mathbf{x}}{\partial s} \times \frac{\partial \mathbf{x}}{\partial t} \right\|}, \tag{A.3}$$

with the normal $\frac{\frac{\partial \mathbf{x}}{\partial s} \times \frac{\partial \mathbf{x}}{\partial t}}{\left\| \frac{\partial \mathbf{x}}{\partial s} \times \frac{\partial \mathbf{x}}{\partial t} \right\|}$ of the area spanned by $\frac{\partial \mathbf{x}}{\partial s}$ and $\frac{\partial \mathbf{x}}{\partial t}$ and the quadratic Jacobian

$$\frac{\partial \mathbf{X}}{\partial (s,t,u)} = \begin{vmatrix} \frac{\partial \mathbf{x}}{\partial s} & \frac{\partial \mathbf{x}}{\partial t} & \frac{\frac{\partial \mathbf{x}}{\partial s} \times \frac{\partial \mathbf{x}}{\partial t}}{\left\| \frac{\partial \mathbf{x}}{\partial s} \times \frac{\partial \mathbf{x}}{\partial t} \right\|} \end{vmatrix}$$

which is equivalent to the triple product of $\frac{\partial \mathbf{x}}{\partial s}$, $\frac{\partial \mathbf{x}}{\partial t}$ and $\frac{\partial \mathbf{x}}{\partial s} \times \frac{\partial \mathbf{x}}{\partial t}$ — the differential volume element when integrating over s, t and u — divided by $\left\| \frac{\partial \mathbf{x}}{\partial s} \times \frac{\partial \mathbf{x}}{\partial t} \right\|$. The division leads to a normalization in the "u-direction", and since this direction is "orthogonal to s and t", du does not affect the result. It follows that for an integration over the area at $u = 0$,

$$\frac{\partial \mathbf{X}}{\partial (s,t,u)} = \left\| \frac{\partial \mathbf{x}}{\partial s} \times \frac{\partial \mathbf{x}}{\partial t} \right\| ds \, dt$$

holds, and an integration over an area $\mathcal{A} \in \mathbb{R}^3$ defined by the substitution $\mathbf{x}(s,t)$ and boundaries s_1, s_2, t_1, t_2 delivers

$$\int_{s_1}^{s_2} \int_{t_1}^{t_2} f(\mathbf{x}(s,t)) \left\| \frac{\partial \mathbf{x}}{\partial s} \times \frac{\partial \mathbf{x}}{\partial t} \right\| ds \, dt.$$

Appendix B

Implementation Details

In this chapter we concentrate on few selected (not self-evident) algorithms which have central meaning for this work. For managing general graph structures and matrix operations required for the ISGCS algorithm, we refer to existing public-domain libraries.

The following extractions from the implementation are printed as *pseudo code* listings derived from *C++* language. Language-specific commands or control structures are denoted with slanted fonts, user-defined objects in typewriter style, and comments begin with the character combination "//" and are typed in Roman style.

B.1 Main Loop (ANN Training)

The program is centered around one main task of continuously selecting samples from the actual sample set and training the kernel and the shading network (see section 1.5).

At certain time steps, the main loop (fg. B.1) is interrupted and either a resampling process or an FEM simulation is triggered which *extends* or *adjusts* the existing sample set, respectively. Thereafter, the algorithm restarts at the

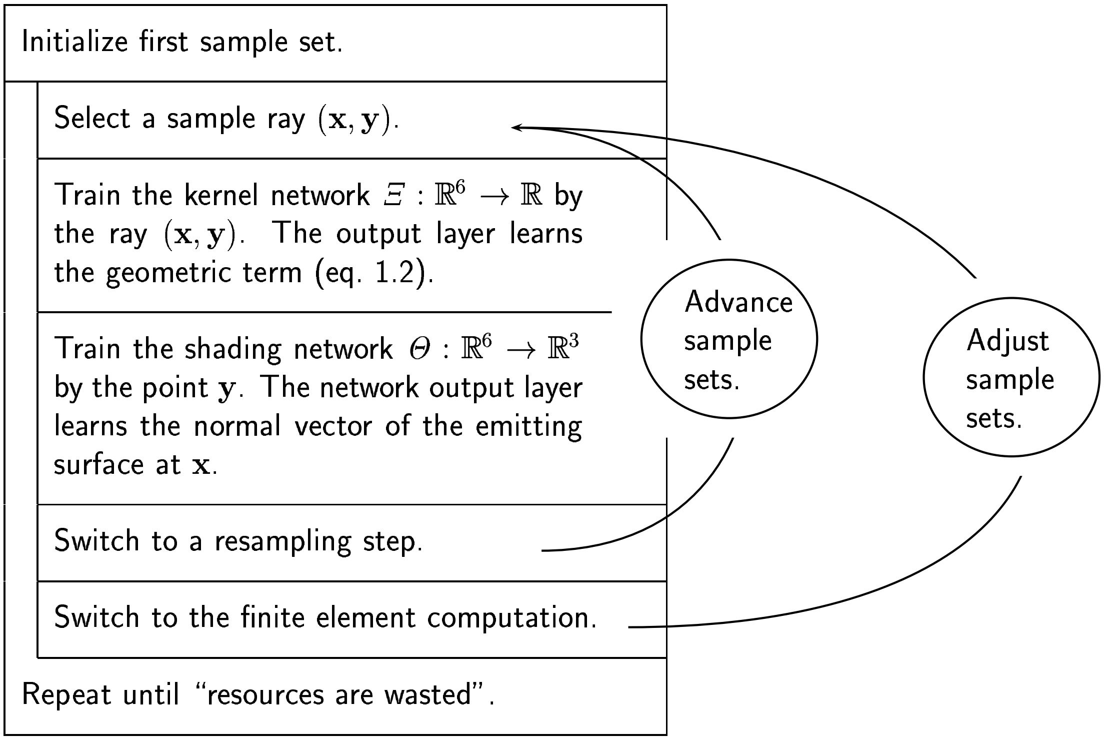

Initialize first sample set.
Select a sample ray $(\mathbf{x}, \mathbf{y})$.
Train the kernel network $\Xi : \mathbb{R}^6 \to \mathbb{R}$ by the ray $(\mathbf{x}, \mathbf{y})$. The output layer learns the geometric term (eq. 1.2).
Train the shading network $\Theta : \mathbb{R}^6 \to \mathbb{R}^3$ by the point $\mathbf{y}$. The network output layer learns the normal vector of the emitting surface at $\mathbf{x}$.
Switch to a resampling step.
Switch to the finite element computation.
Repeat until "resources are wasted".
Advance sample sets.
Adjust sample sets.

Figure B.1: The main loop of the whole algorithm, consisting of adapting the kernel and the shading network to the sample sets.

interrupt point.

Extension is accomplished by creating new sample sets which examine the goal function — the kernel (see section 1.2) — at those places where the approximation accuracy is low. The general ISGCS approach suitable for arbitrary function approximation tasks is described in chapter 2.1.1.

Adjustment means adding propagated energy to the kernel samples. The samples are drawn directly from the geometry definition and are multiplied with the reflection coefficient and the actual emission energy after the k^{th} FEM calculation, $K \cdot B^{(k)}$ (see section 3.1).

As termination condition of the main loop (fg. B.1), several instantiations are possible. For example, the available memory resources, i.e., the number of cells in the shading network, might be a useful criterion, since the complexity of the final triangulation is proportionally related to the number of cells in the shading network, which, again, relates to the number of cells in the kernel network. Thus, this criterion is suitable to limit the final number of polygons, and it is essential for suiting the rendering resources to the radiosity result. A further example for the termination condition might be a simple time criterion.

B.2 Growing Cell Structures

Training both networks — adapting the cell positions by simple vector operations and inserting or deleting cells — is explained in section 2.3. The ISGCS part is encapsulated as one library block with a resampling functionality included. The library interfaces the goal function in an abstract way which means that it does not have any other kind of access to the goal function than simply reading an n-dimensional input vector and calculating an output vector which, then, is interpreted by the according application class. The goal function in this work, which calculates the geometric term defined by the geometry and two points $\mathbf{x}, \mathbf{y} \in \mathbb{R}^3$, is a four-dimensional function. The ISGCS library gets the parameter vector s, t, u, v as input. Based on these parameters, the calculation of the goal function and also the resampling arise.

B.2.1 Sampling the Geometry

For enabling the sampling of the four-dimensional goal function, two tasks must be accomplished — the parameterization (see page 3 and section 4.3) of the four-dimensional input vector to two three-dimensional points, and the calculation of the differential formfactor and the visibility.

Parameterization is proposed as described in the algorithm of figure B.2. After initializing the patch areas (*init()*), the algorithm (*2dTo3d()*) is able to return one three-dimensional surface point corresponding to its two input parameters s and t.

For further explanations, see the comments of the according pseudo code lines. The search routine (*2dTo3d()*) in the implementation initializes a hash table (not shown) which then is accessed in linear time.

Visibility is calculated in the common way by testing a ray against the hierarchically organized surfaces of the whole geometry.

B.3 Orthonormalization

In figure B.3, the orthogonalization like described in section 3.8.1 is outlined. '`a`' denotes the resulting matrix $\mathbf{A}$, which transfers the Λ_i to the orthonormal N_i. '`s`' is an array which stores the calculated values $\int \Lambda_i \Lambda_j$, and `absVal[i]` contains the norms of the orthogonal functions N_i. The functions `scal` calculate the inner product between two $\Lambda_i \Lambda_j$, $N_i \Lambda_j$, or $N_i N_j$. These different

```
init() {  // initialize patch dimensions
    global int geomArea=0;  // reset whole surfaces' area
    for (int  i=0; i<N; i++) {  // for all patches
        geomArea+=patch[i].area;  // add patch size
        patch[i].areaSum=geomArea;  // store 'area position'
    }
}


Vec 2dTo3d(float s, float t) {
    // calculate a point from two parameters
    float areaPos=t·geomArea;
    int i =0;
    do
        i++ ;
    while (i<n) ∧ (patch[i].areaSum  ≤ areaPos);

    if (i>0)
        areaPos-=patch[i-1].areaSum;

    t=areaPos/patch[i].area;

    return s·patch[i].xVec +t·patch[i].yVec;
}
```

Figure B.2: Parameterizing two-dimensional surface coordinates to three-dimensional point coordinates.

functionalities are based on the inner product of two $\Lambda_i \Lambda_j$ which has been explained in section 3.6.1.

B.4 FEM

For the propagation of the energy between all base components N_i and the kernel, first a propagation matrix 'P' between the Λ_i, is calculated by `init()` in figure B.4. Through `calcH` the matrix $\hbar_{pqo}$ is initialized like described in section 3.8.2. Then, `propagate()` is executed which runs through the array of transfer coefficients 'P' and weighs the energy (`P[].val`) by the coefficients of all emitting N_i and all receiving N_j (matrix 'a'). The result is added to the particular radiosity coefficients 'b'.

```
a[N,N];  // result matrix $\mathbf{A}$, see section 3.8.1
s[N,N];  // container for $\int \Lambda_i \Lambda_j$
absVal[N];  // container for $\|\Lambda_i\|$

a[0,0]=1;
s[0,0]=1;
for (int  k=1; k<N; k++) {  // generate $N_k$

    for (int  i=0; i $\leq$ k; i++)
        // calculate $\int \Lambda_k \Lambda_i$ with already used $\Lambda_k$
        s[k][i]=scal($\Lambda_k, \Lambda_i$);

    for (int  i=0; i<k; i++)
        // reset $\mathbf{A}$ in line k
        a[k][i]=0;

    for (int  kk=0; kk<k; kk++) {
        // components are calculated from the influence on $N_k$
        // and on the preceding $N_{kk}$
        float  tmp=scal($N_{kk}, N_k$);
        for (int  j=0; j $\leq$ kk; j++)
            a[k][i]-=tmp/abs[kk]$\cdot$a[kk][j];
    }
    a[k][k]=1; // coefficient of $\Lambda_k$ in $N_k$ equals 1
    absVal[k]=norm($N_k$);
}

for (int  k=0; k<N; k++)  // normalization
    for (int  i=0; i $\leq$ k; i++)
        a[k][i] /= absVal[k];
```

Figure B.3: Schmidt orthogonalization like described in section 3.8.1.

B.5 Triangulation

The last section contains a description of the re-triangulation algorithm mentioned in section 4.3.1. The pseudo code which can be found in figure B.5 is an example limited to polygons with four vertices. The originating patches (named `patch`) are subdivided at the rectangle described by the points `pt1, pt2, pt3, pt4`. If this is not sufficiently small regarding the underlying network topology the function is called recursively (`sub`) for further subdivisions. Otherwise the color results at the vertices are calculated (`collectRadiosity`) and the rectangle is left for the final rendering.

```
init() {  // calculate propagation matrix P between the Λi
    float  P[N·N];
    int  m=0;  // counts the transfer coefficients
    for  (int   p=0; p<N; p++)
        for  (int   q=0; q<N; q++) {
            float  sum = 0;
            for  (int   o=0; o<Z; o++)
                 // for all kernel cell components Ωz
                 // compute transfer coefficients from equation (3.56)
                sum+=calcH(p,q,o);  // see section 3.8.2
            P[m].val = sum;  // store the transfer coefficient
            P[m].from = p;
            P[m].to = q;
            m++;
        }
}


propagate() {
     // calculate propagation between the Ni
     // include the orthogonality matrix A

    for  (int   i=0; i<N; i++)  // reset radiosity coefficients
        b[i]=0;

    for  (int   o=0; o<m; o++)  // for all transfer coefficients from P
        for  (int   i=P[o].from; i<N; i++) {
             // for all non-zero elements from A in line k
            float  ai=a[i,P[o].from];
            for  (int   j=0; j<N; j++) {
                 // for all non-zero elements from A in line o
                float  aj=a[j,P[o].to];
                float  val=P[o].val·ai·aj;
                b[j]+=val; // accumulate radiosity coefficient of Nj
            }
        }
    }
}
```

Figure B.4: Calculation of the transfer factors P in `init()` and the propagation of energy by `propagate`.

```
subdivide(patch, pt1,pt2,pt3,pt4) {
   for  (int   i=0; i<N; i++)
        // for all surfaces of the geometry definition
        // recursively subdivide, pt1...pt4 are the patch vertices
       sub(patch[i],pt1,pt2,pt3,pt4);
}


sub(patch, pt1,pt2,pt3,pt4) {
    // subdivide patch at quad pt1...pt4
   float  patchSize=(patch.xLen+patch.yLen)/2;
    // reference value as "average length" for comparison with network
   Cell cell=bestMatch(shadingNetwork);
    // determine the cell "under" this patch ...
   float  edgeLen=averageEdgeLen(cell);
    // ... and the average emanating edge length
   if  (edgeLen<patchSize) {
        // subdivision granularity not yet reached
       Point ctr=(pt1+pt2+pt3+pt4)/4;  // new vertex
       sub(patch, ctr, pt2, pt3, pt4);
       sub(patch, pt1, ctr, pt3, pt4);
       sub(patch, pt1, pt2, ctr, pt4);
       sub(patch, pt1, pt2, pt3, ctr);
   }
   else  {
        // subdivision granularity reached,
        // calculate radiosity at vertices (collect energy)
       collectRadiosity(pt1);
       collectRadiosity(pt2);
       collectRadiosity(pt3);
       collectRadiosity(pt4);
   }
}
```

Figure B.5: The final triangulation algorithm based on the generated shading network topology, and limited to rectangles.

Bibliography

[1] Hans Wilhelm Alt. *Lineare Funktionalanalysis.* Springer-Verlag, Berlin, Heidelberg, Germany, 1992.

[2] James Arvo, Kenneth Torrance, and Brian Smits. A framework for the analysis of error in global illumination algorithms. In Andrew Glassner, editor, *Proceedings of SIGGRAPH '94 (Orlando, Florida, July 24–29, 1994)*, Computer Graphics Proceedings, Annual Conference Series, pages 75–84. ACM SIGGRAPH, ACM Press, July 1994. ISBN 0-89791-667-0.

[3] auto·des·sys, Inc. form·Z, 1998.

[4] Christopher M. Bishop. *Neural Networks for Pattern Recognition.* Oxford University Press Inc., New York, 1995.

[5] Hans Hermann Bock. *Automatische Klassifikation.* Vandenhoeck & Ruprecht, Göttingen, 1974.

[6] Christian-A. Bohn. Efficiently representing the radiosity kernel through learning. In Xavier Pueyo and Peter Schröder, editors, *Rendering Techniques '96*, pages 123–132, Wien, Austria, 1996. Springer-Verlag Wien New York. Proceedings of the 7th EUROGRAPHICS workshop on rendering in Porto, Portugal, June 17-19, 1996.

[7] Christian-A. Bohn. Finite element mesh generation using growing cell structures networks. In A.B. Bulsari and S. Kallio, editors, *Neural Networks in Engineering Systems*, pages 241–244, Turku, Finland, 1997. Systems Engineering Association.

[8] Christian-A. Bohn. An incremental unsupervised learning scheme for function approximation. In *Proceedings of the 1997 IEEE International Conference on Neural Networks*, volume 3, pages 1792–1797, Piscataway, NJ, June 1997. IEEE Service Center.

[9] Christian-A. Bohn. A neural network representation of three-dimensional geometrical scene descriptions. In *Mathematical Modelling And Scientific*

Computing, volume 8. Principia Scientia, Inc.-International Publishers, October 1997.

[10] S. Brehmer and H. Haar. *Differentialformen und Vektoranalysis.* VEB Deutscher Verlag der Wissenschaften, Berlin, German Democratic Republic, 1973.

[11] I.N. Bronstein and K.A. Semendjajew. *Taschenbuch der Mathematik.* B.G. Teubner Verlagsgesellschaft, Leipzig, 20 edition, 1983.

[12] Michael F. Cohen, Shenchang Eric Chen, John R. Wallace, and Donald P. Greenberg. A progressive refinement approach to fast radiosity image generation. In John Dill, editor, *Computer Graphics (SIGGRAPH '88 Proceedings)*, volume 22, pages 75–84, August 1988.

[13] Michael F. Cohen and John R. Wallace. *Radiosity and Realistic Image Synthesis.* Academic Press Professional, San Diego, CA, 1993.

[14] Christian Darken and John E. Moody. Fast, adaptive K-means clustering: Some empirical results. In *International Joint Conference on Neural Networks*, volume 2, pages 233–238, San Diego, CA, June 1990. IEEE Press.

[15] B. S. Duran and P. L. Odell. *Cluster Analysis (A Survey)*, volume 100 of *Lecture Notes in Economics and Mathematical Systems.* Springer, Berlin/New York, 1974.

[16] Friedhelm Erwe. *Differential- und Integralrechnung.* Bibliographisches Institut Mannheim, Hochschultaschenbücher Verlag, Mannheim, 1962.

[17] 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.

[18] Bernd Fritzke. Kohonen feature maps and growing cell structures - a performance comparison. In L. Giles, S. Hanson, and J. Cowan, editors, *Advances in Neural Information Processing Systems 5*, pages 123–130. Morgan Kaufmann, San Mateo, CA, 1993.

[19] Bernd Fritzke. Incremental learning of local linear mappings. In *Proceedings of the ICANN-95*, Paris, France, 1995.

[20] Andrew S. Glassner. *Principles of Digital Image Synthesis.* Morgan Kaufmann, San Francisco, CA, 1995.

[21] Cindy M. Goral, Kenneth E. Torrance, Donald P. Greenberg, and Bennett Battaile. Modelling the interaction of light between diffuse surfaces. In *Computer Graphics (SIGGRAPH '84 Proceedings)*, volume 18, pages 212–22, July 1984.

[22] 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.

[23] Richard A. Graff. *Elements of non-linear functional analysis*, volume 16. American Mathematical Society, Providence, Rhode Island, 1978.

[24] Charles W. Groetsch. *Elements of applicable functional analysis.* Marcel Dekker, Inc., New York, 1980.

[25] 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.

[26] Paul Heckbert. Radiosity in flatland. *Computer Graphics Forum (Eurographics '92)*, 11(3):181–192, September 1992.

[27] Paul S. Heckbert and James M. Winget. Finite element methods for global illumination. Technical Report UCB/CSD 91/643, Computer Science Division (EECS), University of California, July 1991.

[28] J. Hertz, A. Krogh, and R. Palmer. *Introduction to the Theory of Neural Computation.* Addison-Wesley, 1991.

[29] Einar Hille and Ralph S. Phillips. *Functional Analysis and Semi-Groups.* American Mathematical Society, Providence, R. I., 1957.

[30] 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.

[31] Teuvo Kohonen. Self-organized formation of topologically correct feature maps. *Biological Cybernetics, 43*, pages 59–99, 1982.

[32] Teuvo Kohonen. *Self-Organization and Associative Memory.* Springer-Verlag, Berlin, third edition, 1989.

[33] Teuvo Kohonen. *Self-organizing maps.* Springer Verlag, New York, 1997.

[34] K.H. Körber and E.A. Pforr. *Integralrechnung für Funktionen mit mehreren Variablen.* Verlag Harri Deutsch, Frankfurt/Main, 1980.

[35] Y. Linde, A. Buzo, and R. M. Gray. An algorithm for vector quantizer design. *IEEE Trans. on Communications*, COM-28(1):84–95, January 1980.

[36] Dani Lischinski, Brian Smits, and Donald P. Greenberg. Bounds and error estimates for radiosity. In Andrew Glassner, editor, *Proceedings of SIGGRAPH '94 (Orlando, Florida, July 24–29, 1994)*, Computer Graphics Proceedings, Annual Conference Series, pages 67–74. ACM SIGGRAPH, ACM Press, July 1994. ISBN 0-89791-667-0.

[37] S.P. Lloyd. Least squares quantization in PCM. Technical report, Bell Labs., 1957.

[38] M. Loeve. *Probability Theory.* Springer, New York, 4 edition, 1977.

[39] Paul Lorenzen. *Differential und Integral.* Akademische Verlagsgesellschaft, Frankfurt/Main, 1965.

[40] Wilhelm Maak. *Differential- und Integralrechnung.* Vandenhoeck & Ruprecht, Göttingen, 1969.

[41] J. MacQueen. Some methods for classification and analysis of multivariate observations. In *Mathematics, Statistics and Probability: 5th Berkeley Symposium*, volume I, pages 281–297, Berkeley, California, 1967. University of California Press.

[42] Charles A. Micchelli. Interpolation of scattered data: Distance matrices and conditionally positive definite functions. In *Constructive Approximation*, pages 11–22, New York, 1986. Springer Verlag.

[43] 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.

[44] Tomoyuki Nishita and Eihachiro Nakamae. Continuous tone representation of three-dimensional objects taking account of shadows and interreflection. In B. A. Barsky, editor, *Computer Graphics (SIGGRAPH '85 Proceedings)*, volume 19, pages 23–30, July 1985.

[45] Mark J.L. Orr. Introduction to radial basis function networks. World Wide Web, URL: http://www.cns.ed.ac.uk/people/mark.html, April 1996.

[46] J. Park and I. W. Sandberg. Universal approximation using radial-basis-function networks. *Neural Computation*, 3(2):246–257, 1991.

[47] T. Poggio and F. Girosi. A theory of networks for approximating and learning. Artificial Intelligence Lab. Memo 1140, MIT, 1989.

[48] T. Poggio and F. Girosi. Networks for approximation and learning. *Proceedings of the IEEE*, 78:1481–1497, 1990.

[49] M. J. D. Powell. Radial basis functions for multivariable interpolation: A review. In *Proc. IMA Conf. on "Algorithms for the approximation of functions and data"*, Shrivenham, 1985. RMCS.

[50] Helge Ritter, Thomas Martinez, and Klaus Schulten. *Neuronale Netze - eine Einfuehrung in die Neuroinformatik selbstorganisierender Netzwerke.* Addison-Wesley, Bonn, 1990.

[51] R. Rojas. *Theorie der neuronalen Netze: eine systematische Einfuehrung.* Springer-Verlag, Berlin, 1993.

[52] W.S. Sarle. Neural network FAQ. Periodic posting to the Usenet newsgroup comp.ai.neural-nets,
URL: `ftp://ftp.sas.com/pub/neural/FAQ.html`, 1997.

[53] Mateu Sbert, Xavier Pueyo, Lazlo Neumann, and Werner Purgathofer. Global multipath monte carlo algorithms for radiosity. *The Visual Computer*, 12(2):47–61, 1996. ISSN 0178-2789.

[54] Hans Rudolf Schwarz. *Methode der finiten Elemente.* B.G. Teubner, Stuttgart, Germany, 1991.

[55] Robert Siegel and John R. Howell. *Thermal Radiation Heat Transfer.* Hemisphere Publishing Corp., Washington, DC, 1981.

[56] François Sillion. Clustering and volume scattering for hierarchical radiosity calculations. In *Fifth Eurographics Workshop on Rendering*, pages 105–117, Darmstadt, Germany, June 1994.

[57] François Sillion and George Drettakis. Feature-based control of visibility error: A multi-resolution clustering algorithm for global illumination. In Robert Cook, editor, *SIGGRAPH 95 Conference Proceedings*, Annual Conference Series, pages 145–152. ACM SIGGRAPH, Addison Wesley, August 1995. held in Los Angeles, California, 06-11 August 1995.

[58] François Sillion and Claude Puech. *Radiosity and Global Illumination.* Morgan Kaufmann, San Francisco, 1994.

[59] 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.

[60] Eric J. Stollnitz, Tony D. DeRose, and David H. Salesin. *Wavelets for Computer Graphics: Theory and Applications.* Morgan Kaufmann, San Francisco, CA, 1996.

[61] Lightscape Technologies. Lightscape$^{\mathrm{TM}}$, 1998.

[62] Seth Teller and Pat Hanrahan. Global visibility algorithms for illumination computations. In *Computer Graphics Proceedings, Annual Conference Series, 1993*, pages 239–246, 1993.

[63] Eric Veach. *Robust Monte Carlo Methods for Light Transport Simulation.* PhD thesis, Stanford University, CA, 1997.

[64] G. Voronoi. Nouvelles applications des paramètres continus à la théorie des formes quadratiques. Premier mémoire: Sur quelques proprieteés des formes quadratiques positives parfaites. *Reine angewandte Mathematik*, 133:97–178, 1907.

[65] Stephen Wolfram. Mathematica, 1988.

[66] Harold R. Zatz. Galerkin radiosity: A higher order solution method for global illumination. In *Computer Graphics Proceedings, Annual Conference Series, 1993*, pages 213–220, 1993.

List of Figures