STEFAN SCHLECHTWEG

# INFORMATIONSDARSTELLUNG MIT BILDERN

# INFORMATIONSDARSTELLUNG MIT BILDERN

Habilitationsschrift

zur Erlangung der Venia Legendi für
**Informatik**
angenommen durch die Fakultät für Informatik
der Otto-von-Guericke-Universität Magdeburg

von: Dr.-Ing. Stefan Schlechtweg
geb. am 10. Februar 1971 in Bad Salzungen

Gutachterinnen/Gutachter:
Prof. Dr. rer. nat. habil. Thomas Strothotte, PhD, MBA (Universität Magdeburg)
Prof. Dr.-Ing. habil. Heidrun Schumann (Universität Rostock)
Univ.-Prof. Mag. Dr. Silvia Miksch (Donau-Universität Krems)
Ao. Univ.-Prof. Dipl.-Ing. Dr. techn. Eduard Gröller (Technische Universität Wien)

Magdeburg, 07. März 2007

## INHALTSVERZEICHNIS

# 1 EINLEITUNG

Bilder dienen seit jeher zur Darstellung von Informationen. Beispiele hierfür finden sich in allen kulturgeschichtlichen Epochen, angefangen von frühen Zahlzeichen bis hin zu heutigen computergenerierten interaktiven Darstellungen hochdimensionaler Informationsräume. Bilder bieten als zweidimensionales Medium weit mehr Gestaltungsspielräume als beschreibende Texte und durch die Auswahl verschiedener Darstellungstechniken ergibt sich eine Vielzahl an Möglichkeiten, Informationen graphisch zu kodieren.

## 1.1 ZIELE UND ERGEBNISSE

Ziel der vorliegenden Arbeit ist es, ein vertieftes Verständnis der Informationsdarstellung durch Bilder zu erlangen. Dies geschieht hauptsächlich in zwei Richtungen:

1. die Darstellung geometrischer Informationen,
2. die Darstellung abstrakter Informationen.

In beiden Bereichen sollen anhand einzelner Beispiele die Eigenschaften solcher Darstellungen untersucht und für konkrete Anwendungsfälle Techniken entwickelt werden. Aber auch das Gebiet als Ganzes soll eine weitere Untersuchung und Strukturierung erfahren.

Im Ergebnis entstanden mehrere Arbeiten sowohl theoretischer Natur als auch anwendbare Methoden. Basierend auf obiger Einteilung handelt es sich dabei:

1. zur Darstellung geometrischer Informationen um:
   a) eine umfassende Betrachtung der nicht-photorealistischen Computergraphik in Form eines Lehrbuches (siehe Anhang A) sowie eine Einführung des Stil-Begriffs in die Computergraphik (Anhang B),
   b) eine Übersicht über verschiedenste Verfahren, Silhouetten aus geometrischen Modellen oder Bildern zu berechnen und darzustellen (Anhang C),
   c) eine neue Technik zur Erzeugung von Schraffurdarstellungen, die es erstmalig ermöglicht, Bilder abhängig vom intendierten Ausgabegerät in unterschiedlichen angepaßten Qualitätsstufen zu generieren (Anhang D) und
   d) eine neuartige Technik des Stroke-Based Rendering, die es erstmalig ermöglicht, verschiedenste Stile mit einer einheitlichen Technik zu erzeugen und diese auch in einem Bild zu mischen (Anhang E);
2. zur Darstellung abstrakter Informationen um:

- eine Visualisierungstechnik, die Erkenntnisse des nicht-photorealistiechen Renderings in der Informationsvisualisierung anwendet (Anhang F) sowie ein System, das verschiedene Verfahren der Informationsvisualisierung kombiniert, um aussagekräftige Darstellungen von Behandlungsplänen für Intensivtstationen zu erzeugen (Anhang G),
- eine neuartige interaktive Visualiserungstechnik zur Schlüsselwortsuche in digitalen Dokumenten, die die Darstellung der Suchergebnisse mit der Darstellung der Dokumentstruktur verbindet und dadurch ein neues Navigationswerkzeug für digitale Dokumente bereitstellt (Anhang H)
- eine neuartige Technik zur Navigation in digitalen Dokumenten basierend auf Relationen zwischen Dokumentbestandteilen (Anhang I) sowie
- eine neue Technik zur Navigation in großen, stark vernetzten Datenbeständen, wie diese beispielsweise in Literaturdatenbanken oder digitalen Bibliotheken vorliegen (Anhang J).

Weitere, im Umfeld dieser oben genannten Arbeiten entstandene Techniken und Systeme werden im Kapitel 5 angesprochen und sind im Literaturverzeichnis enthalten.

## 1.2 Aufbau der Arbeit

Das folgende zweite Kapitel enthält eine kurze Einführung in die Thematik der Arbeit und stellt den thematischen Rahmen vor. Die beiden hauptsächlichen Richtungen der Informationsdarstellung mit Bildern, die in dieser Arbeit betrachtet werden, sind in den dann folgenden Kapiteln drei und vier beschrieben. Hier werden jeweils zu Beginn einige Grundlagen angegeben und verwandte Arbeiten betrachtet, bevor die im Anhang angefügten Veröffentlichungen hier eingeordnet und kurz vorgestellt werden. Das abschließende fünfte Kapitel beinhaltet eine Zusammenfassung.

Die Anhänge A bis J enthalten die in dieser Arbeit besprochenen Veröffentlichungen. Verweise hierauf sind im Text durch gesonderte Randmarkierungen kenntlich gemacht. Verweise auf weitere eigene Veröfentlichungen, die im Umfeld dieser Arbeit entstanden sind, sind im Text durch Unterstreichungen gekennzeichnet.

## 1.3 Danksagung

Diese Arbeit entstand am Institut für Simulation und Graphik der Otto-von-Guericke-Universität Magdeburg. Hier habe ich eine angenehme und produktive Arbeitsatmosphäre gefunden und mit Prof. Thomas Strothotte einen Betreuer der mir viel Freiraum gelassen hat. Ich möchte mich an dieser Stelle besonders bei ihm und allen Kollegen am ISG bedanken. Ein spezieller Dank geht an das Technik-Team sowie das Sekretariat, ohne die das ISG in dieser Form sicher nicht arbeitsfähig wäre.

Diese Habilitationsschrift ist das Ergebnis der wissenschaftlichen Zusammenarbeit mit vielen Kolleginnen und Kollegen nicht nur aus Magdeburg. Diesen gilt mein

besonderer Dank für das gemeinsam Ideen ausbrüten, Themen diskutieren, Veröffentlichungen schreiben und präsentieren. Besonders bedanken möchte ich mich hier bei Dr. Marcel Götze, der nicht nur zum Thema digitale Dokumente mit mir zusammenarbeitete sondern auch als Freund meine Arbeit immer vorangebracht hat. Frau Prof. Heidrun Schumann und ihrer Arbeitsgruppe in Rostock bin ich ebenfalls zu Dank verpflichtet. Viele Ideen im Umfeld dieser Arbeit entstammen fruchtbaren Diskussionen mit den Rostocker Kollegen.

Nicht zuletzt waren es auch die Studierenden, die durch Fragen und Diskussionen in Lehrveranstaltungen oder durch ihre Studien- und Diplomarbeiten Beiträge geleistet haben. Dafür möchte ich mich ebenfalls bedanken und ich hoffe, daß ich durch meine Lehrveranstaltungen auch zu deren beruflichem Erfolg beitragen konnte.

# 2 INFORMATIONSDARSTELLUNG MIT BILDERN

Der Kern dieser Arbeit beschäftigt sich mit der Darstellung von Informationen. Der Begriff der *Information* ist sehr weitläufig verwendet und daher auch schwer abzugrenzen. Hier soll deshalb keine Definition angegeben werden, sondern im Kontext dieser Arbeit wird die sehr weit gefaßte, im allgemeinen Sprachgebrauch verwendete Bedeutung genutzt. Information ist damit „zweckorientiertes bzw. zielgerichtetes Wissen“ [SH05] bzw. „Kenntnis über Sachverhalte oder Vorgänge“ [Nie77].

Wichtig im Zusammenhang dieser Arbeit ist die Übertragung von Informationen, die auch als *Kommunikation* bezeichnet wird. In der Informationstheorie und in anderen Wissenschaften, die sich der Informationstheorie bedienen, wird die Kommunikation üblicherweise mit Hilfe des Sender-Empfänger-Modells beschrieben [SW69]. Der Sender will Informationen an einen Empfänger übetragen. Dazu werden die Informationen nach einem entsprechenden Schema kodiert und die so entstehende Nachricht über einen Kanal (mit Hilfe eines Trägermediums) an den Empfänger übertragen. Dieser dekodiert die Nachricht, um die in ihr verschlüsselten Informationen zu erhalten (siehe Abbildung 2.1).

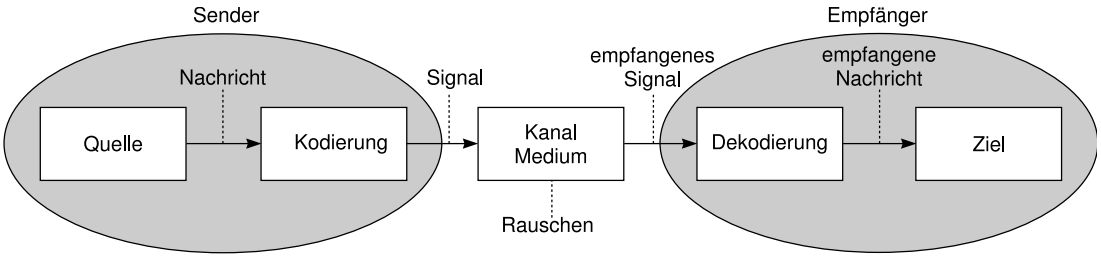



Abbildung 2.1. Kommunikationsprozeß nach Shannon und Weaver (nach [SS97]).

Die Kommunikation zwischen Menschen weist dabei eine Besonderheit auf. Der Empfänger kann und wird aus den erhaltenen Informationen weitere Informationen durch verschiedenste Schlußfolgerungsprozesse erzeugen. Diese Schlußfolgerungen beruhen auf Informationen, die nicht direkt in der Nachricht kodiert sind bzw. auch auf der Verknüpfung von in der Nachricht direkt kodierten Informationen. Goffman [Gof82] unterscheidet daher in die folgenden beiden Klassen:

- *Information given* sind die tatsächlich vom Sender bereitgestellten und damit übertragenen Informationen. Sie bestehen aus „Wortsymbolen und ihren Substituten, die der einzelne eingestandenermassen und ausschliesslich dazu verwendet, diejenigen Informationen zu vermitteln, die er und die anderen mit diesen Symbolen verknüpfen.“ [Gof82]

- *Information given off* sind die Informationen, die diesen Schlußfolgerungsprozeß unterstützen. Sie betreffen weitere Informationen, die vom Sender bereitgestellt werden und vom Empfänger dazu genutzt werden können, Schlußfolgerungen zu ziehen; vorausgesetzt, diese Informationen wurden vom Sender für andere als der Informationsübertragung dienenden Zwecke bereitgestellt (nach [Gof82]).

Strothotte und Strothotte [SS97] nähern sich diesem Modell auf eine ähnliche Art und Weise und unterscheiden in

- *gelieferte Informationen* als die in der Nachricht tatsächlich enthaltenen Informationen und
- *mitgelieferte Informationen* als die aus der Nachricht abgeleiteten Informationen.

Art und Umfang der mitgelieferten Informationen hängen einerseits natürlich von den Fähigkeiten von Sender und Empfänger ab, eine entsprechende Kodierung/Dekodierung vorzunehmen und insbesondere von den Fähigkeiten des Empfängers, die entsprechenden Schlußfolgerungsprozesse auszuführen. Andererseits spielen die Auswahl des Trägermediums und dessen Eigenschaften eine ebenso große Rolle. In dieser Arbeit wird von Bildern als Trägermedium ausgegangen. Die Kodierung der Informationen erfolgt bei Bildern durch Geometrie und Farbe, also durch die Anordnung geometrischer Elemente, die dann durch die menschliche Wahrnehmung zu einem Bild zusammengesetzt werden. Die Extraktion von Informationen aus Bildern wird dabei durch verschiedene psychologische Prozesse gesteuert, wie beispielsweise die Gestaltgesetze [ZG99]. Sollen Bilder zur Vermittlung von Informationen eingesetzt werden, die darüber hinausgehen, das tatsächliche Aussehen eines Gegenstandes wiederzugeben, ist eine Kenntnis dieser psychologischen Prinzipien wichtig, um eine fehlerfreie (bzw. fehlerarme) Kodierung und Dekodierung sicherzustellen.

Mackinlay beschreibt in [Mac86] Kriterien für die graphische Präsentation relationaler Informationen, die von vielen Autoren als Kriterien für die Gestaltung von Visualisierungen im Sinne von Kapitel 4 dieser Arbeit genutzt und erweitert werden (siehe auch [SM00]). Tatsächlich lassen sich die Kriterien von Mackinlay (und deren Erweiterung nach Schumann und Müller) für graphische Darstellungen aller Art verallgemeinern:

Expressivität: Die Expressivität oder Ausdrucksfähigkeit einer graphischen Darstellung beschreibt die unverfälschte Wiedergabe der darzustellenden Informationen. Das bedeutet, daß die und nur die in den Daten enthaltenen Informationen dargestellt werden.

Effektivität: Die Effektivität beschreibt die Fähigkeit einer graphischen Darstellung, die in ihr enthaltenen Informationen zu veranschaulichen und auf intuitive Weise an den Betrachter zu vermitteln. Um eine effektive Darstellung zu erreichen, muß unter den Randbedingungen Zielsetzung, Anwendungskontext, visuelle Fähigkeiten des Betrachters und Eigenschaften des die Graphik erzeugenden Systems ein optimales Verfahren zur Generierung der graphischen Darstellung eingesetzt werden.

Angemessenheit: Die Angemessenheit einer graphischen Darstellung setzt den Aufwand und Nutzen in Relation zueinander, wobei unter den Faktoren, die den Aufwand beschreiben, nicht ausschließlich der technische und zeitliche Aufwand zur Erzeugung der Darstellung zu verstehen ist, sondern hier auch der zeitliche, physische und psychische Aufwand des Betrachters einfließt, die Informationen aus der graphischen Darstellung zu extrahieren.

Bei der Gestaltung und beim Einsatz von Bildern zur Informationsdarstellung stellen diese Kriterien eine Diskussionsgrundlage dar, um genau die Eigenschaften des Bildes festzulegen, die die bei Strothotte und Strothotte [SS97] angesprochenen Schlußfolgerungsprozesse anstoßen, die zur Ableitung der mitgelieferten Informationen führen.

In den folgenden Kapiteln werden zwei Gruppen von Bildern bzw. graphischen Darstellungen näher untersucht, die zur Visualisierung geometrischer und nichtgeometrischer Daten eingesetzt werden. Dabei werden jeweils allgemein obige Kriterien diskutiert und deren Umsetzung anhand von Beispielen gezeigt.

# 3 RENDERING UND NICHT-PHOTOREALISTISCHE COMPUTERGRAPHIK

Die *Computergraphik* (oder *Bildsynthese*, siehe [WP98] oder [Sch05a]) als ein Teilgebiet der Informatik beschäftigt sich mit der algorithmischen Erzeugung von Bildern. Die Breite des Gebietes erstreckt sich dabei von Algorithmen zur Rasterkonvertierung zweidimensionaler geometrischer Primitive (Linien, Kreise, etc.) bis hin zur Abdeckung des gesamten Prozesses der Erzeugung eines zweidimensionalen Bildes aus einem gegebenen dreidimensionalen Geometriemodell. Dieser Prozess wird allgemein mit dem Begriff *Rendering* bezeichnet.

Ziel der Computergraphik war und ist es, eine möglichst realitätsgetreue und geometrisch korrekte Wiedergabe der modellierten Geometrie zu erreichen. Dieses Ziel wurde in Anlehnung an eine insbesondere in Nordamerika populäre Kustrichtung als *Photorealismus* bekannt. Dadurch, daß die photorealistische Computergraphik die Arbeitsweise einer photographischen Kamera simuliert, müssen sämtliche geometrischen Informationen sowie die Beleuchtungs- und Materialcharakteristika auf die Farbe der einzelnen Pixel abgebildet werden. Dies schränkt die Ausdrucksfähigkeit solcher Bilder stark ein. Um die Ausdrucksfähigkeit computergraphischer Darstellungen zu erhöhen, entwickelte sich seit den 1980er Jahren ein weiterer Zweig der Computergraphik, der sich mit *nicht-photorealistischer* Bilderzeugung beschäftigt (siehe [GG01] und [SS02]). Ziel hierbei ist es, unterschiedliche *Stile* zu verwenden, um damit sowohl geometrische Eigenschaften als auch nicht-geometrische, zusätzlich gegebene Eigenschaften (wie beispielsweise die Wichtigkeit eines Objektes im Dialogkontext), darstellen zu können.

→ Anhang A Non-Photorealistic Computer Graphics. Modelling, Rendering, and Animation

Im folgenden wird ein Überblick über die photorealistische und nicht-photorealistische Bilderzeugung gegeben, bevor auf einige Aspekte der nicht-photorealistischen Darstellung insbesondere geometrischer Informationen näher eingegangen wird.

## 3.1 PHOTOREALISTISCHE COMPUTERGRAPHIK

Der gesamte Prozeß der photorealistischen Bilderzeugung setzt sich aus verschiedenen Teilprozessen zusammen, zu denen unter anderem die folgenden gehören (siehe auch [FvDFH90] und [Wat02a]):

- Geometrische Modellierung und Modelltransformationen,
- Geometrische Transformationen und Projektion,
- Entfernen unsichtbarer Modellteile,
- Beleuchtungsberechnungen,

- Rasterisierung,
- Texturierung und Shading.

Durch die Simulation der Arbeitsweise einer photographischen Kamera werden Geometrie, Beleuchtung und Materialien insgesamt durch die Färbung der einzelnen Bildpunkte (Pixel) wiedergegeben. Daher spielen für die Bilderzeugung zwei Punkte eine wesentliche Rolle. Zum einen betrifft dies die korrekte Modellierung des sichtbaren Bereichs der Szene. Dies schließt die eigentliche Modellierung der geometrischen Infomationen, die korrekte Abbildung der dreidimensionalen Geometrie auf die zweidimensionale Bildebene und auch die Bestimmung des sichtbaren Anteils der Objekte ein. Der zweite wesentliche Bereich besteht in der Beleuchtungsberechnung, die zu einer gegebenen Beleuchtungssituation aus den Reflexionseigenschaften der Objektoberflächen und den Strahlungseigenschaften der Lichtquellen die Farbe der einzelnen Bildpunkte bestimmt. Abbildung 3.1 zeigt zwei Beispiele photorealistischer Darstellungen, die mittels verschiedener Verfahren erzeugt wurden.

(a) Computergenerierte Landschaft. Copyright: Oliver Deussen, Bernd Lintermann

(b) „Steel Mill“ Quelle: Stephen Spencer, SIGGRAPH 1993 Education Slide Set

Abbildung 3.1. Photorealistische Darstellungen

Ausgehend von den Kriterien aus Kapitel 2 lassen sich die Qualitätsmerkmale für computergenerierte photorealistische Darstellungen wie folgt diskutieren:

Expressivität: Die unverfälschte Wiedergabe der darzustellenden Informationen bedeutet bei pohotorealistischen Computergraphiken insbesondere die Beleuchtung und die Geometrie bzw. das Zusammenspiel der beiden korrekt wiederzugeben. Für eine Darstellung von darüber hinaus gehenden Informationen sind photorealisitische Computergraphiken weniger bis gar nicht geeignet.

Effektivität: Die bei photorealistischen Computergraphiken zur Verfügung stehenden Mittel zur Wiedergabe von Geometrie und Beleuchtung entsprechen denen der Photographie. Hierunter fallen z. B. Kameraparameter und Farbraum. Werden diese Mittel entsprechend dem Anwendungskontext und dem Kommunikationsziel eingesetzt, ergeben sich effektive graphische Darstellungen.

Angemessenheit: Der technische Aufwand zur Erstellung einer photorealistischen Computergraphik kann – je nach verwendetem Verfahren – sehr hoch sein. Demgegenüber ist der mentale Aufwand für den Betrachter, geometrische Informationen aus einem photorealistischen Bild zu dekodieren, sehr gering, da dies den normalen Sehgewohnheiten entspricht.

Moderne computergraphische Systeme und Anwendungen sind in der Lage, diese Qualitätskriterien zu einem sehr hohen Grad zu erfüllen. Ein Indiz dafür ist auch der immer stärker steigende Einsatz computergraphischer Techniken im Medienbereich, insbesondere bei Filmproduktionen.

## 3.2 nicht-photorealistische computergraphik

Für einen Großteil der möglichen Anwendungsgebiete computergraphischer Techniken ist eine realitätsgetreue und geometrisch korrekte Wiedergabe des Modells alleine allerdings nicht ausreichend oder nicht zielführend. Ein Beispiel soll dies illustrieren. Abbildung 3.2 zeigt zwei Darstellungen aus medizinischen Lehrbüchern, die dem Zweck der Informationsvermittlung dienen. Eine photographische Darstellung des Inneren des menschlichen Körpers wäre hier schwierig, da sich die Strukturen durch ihre Färbung kaum unterscheiden würden. Gerade in diesem Bereich haben sich Techniken und Konventionen herausgebildet, die gezielt eingesetzt werden, um spezielle auch nicht-geometrische Informationen zu vermitteln.

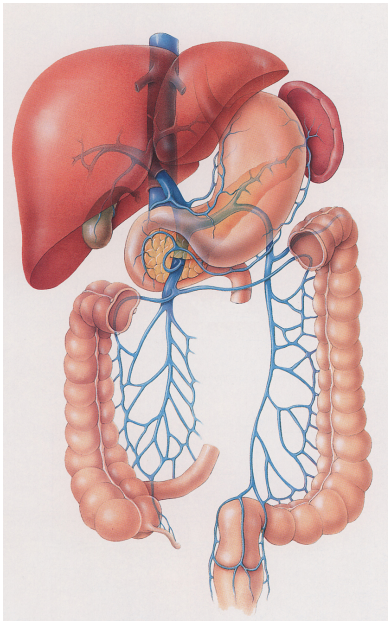
Weiterhin zeigen bildliche Darstellungen aus sämtlichen (kultur-)historischen Epochen, daß der Mensch das Ziel hatte und auch in der Lage war, sehr selektiv auf graphischem Weg Informationen zu vermitten, die über die geometrisch korrekte Wiedergabe des Modells im Sinne von Form und Farbe hinausgehen. Das Ziel der graphischen Vermittlung von Informationen war und ist nicht an photorealistische Darstellungen und technische Möglichkeiten ihrer Erzeugung gekoppelt. Vielmehr eröffnet gerade die Abkehr hiervon Möglichkeiten, mehr Informationen oder aber andere Informationen in einem Bild darzustellen und somit dieses Bild zielgerichtet im Kommunikationsprozeß einzusetzen (siehe hierzu insbesondere [S+98] und [SS02]).

→ Anhang A Non-Photorealistic Computer Graphics. Modelling, Rendering, and Animation

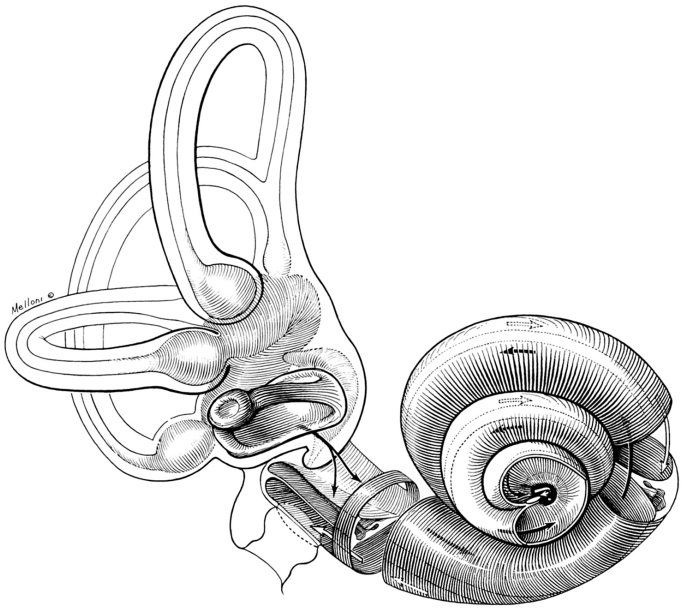
Diese Beobachtungen führen seit den 1980er Jahren zur Entwicklung der *nicht-photorealistischen Computergraphik (NPR)*, deren Hauptziele es sind:

1. computergraphische Darstellungen zu erzeugen, die effektiv zur Informationsvermittlung eingesetzt werden können,
2. die Ausdruckskraft computergraphischer Darstellungen durch die Einbeziehung verschiedener Stile zu erweitern und
3. die Arbeitsweise von Werkzeugen nachzuahmen, die zur manuellen Bilderzeugung eingesetzt werden.

Um diese Ziele zu erreichen, wird eine erstaunliche Breite an Kenntnissen aus anderen Bereichen der Informatik und auch aus anderen Wissenschaften genutzt. Damit stellt NPR sowohl als Forschungsrichtung als auch in seinen Anwendungen ein inter-



(a) Illustration von Leber und Magen. Transparenz, Schnitte und Farbkodierungen wurden hier zur Informationsvermittlung eingesetzt. Quelle: unbekannt.



(b) Illustration des Innenohrs. Geometrische und nicht-geometrische Informationen werden lediglich durch Schraffuren, Umrisse und abstrakt-graphische Symbole dargestellt. Quelle: G. J. Tortora: *Introduction to the Human Body*, 1997.

Abbildung 3.2. Medizinische Illustrationen

disziplinäres Gebiet dar. Betrachtet man dies und die Ziele der nicht-photorealistischen Computergraphik ergibt sich die folgende Diskussion der Qualitätskriterien (nach Kapitel 2):

Expressivität: Die Vielzahl der zur Verfügung stehenden darstellerischen Mittel und Techniken erschwert die Erzeugung einer expressiven Darstellung. Andererseits stehen dadurch im Gegensatz zur photorealistischen Computergraphik auch erst die Möglichkeiten bereit, expressive Darstellungen zu erzeugen.

Effektivität: Durch die Anwendung von Abstraktionen können insbesondere nicht-geometrische Informationen sehr effektiv im Kontext mit geometrischen Informationen vermittelt werden. Auch bei der Darstellung geometrischer Informationen bietet die Abstraktion eine Möglichkeit, die Effektivität der Darstellung zu steigern, da nicht mehr alle Informationen über Farbe und Beleuchtung kodiert werden müssen.

Angemessenheit: Die Darstellungen orientieren sich in den meisten Fällen nicht an den normalen Sehgewohnheiten und setzen Techniken ein, die durch den Betrachter dekodiert werden müssen, um die Informationen zu erhalten. Daher ist der mentale Aufwand größer als bei photorealistischen Computergraphiken. Trotzdem werden Abstraktionen sehr schnell und effektiv wahrgenommen, da der menschliche Wahrnehmungsprozeß durch die Aufdeckung von Invarianzen

und Stabilitäten unvollständige Bilder zu schlüssigen Sinneseindrücken verarbeitet [ZG99]. Der technische Aufwand zur Erzeugung nicht-photorealistischer Darstellungen kann als vergleichbar zu photorealistischen Darstellungen eingeschätzt werden. Vorteilhaft bei nicht-photorealistischen Darstellunge ist oftmals ihre bessere Reproduzierbarkeit in Print-Medien.

## 3.3 Datenstrukturen und Algorithmen für NPR

Ausgehend von den obigen Überlegungen zeigt sich, daß die zur Erzeugung nicht-photorealistischer Graphiken genutzten Algorithmen nicht auf die Standardalgorithmen der generativen Computergraphik beschränkt bleiben können. Das allgemeine Ziel ist es, die Ausdruckskraft computergenerierter Darstellungen dadurch zu erweitern, daß diese Graphiken in verschiedenen *Stilen* erzeugt werden können [Sch05b]. Eine direkte Umsetzung der Geometrieinfomationen aus dem dreidimensionalen Modell in die Farbinformationen einzelner Pixel reicht nicht mehr aus, um die zusätzlich zu vermittelnden Informationen im zu erzeugenden Bild zu kodieren. Diese Informationen verändern sowohl das Modell als auch den Prozeß der Bilderzeugung auf verschiedenen Ebenen.

→ Anhang B
Stile in der Computergraphik, oder: Können Rechner malen?

Eine Untersuchung der in die nicht-photorealistische Computergraphik einzuordnenden Literatur ergibt insbesondere vier verschiedene Bereiche, die einen deutlichen Unterschied zwischen photorealistischer und nicht-photorealistischer Bilderzeugung zeigen (siehe auch [HIR+03]):

1. *Modellmanipulationen:* Das dreidimensionale Geometriemodell als Ausgangspunkt der Bilderzeugung wird häufig Manipulationen unterworfen, um einerseits zusätzliche Informationen darin zu kodieren, es andererseits aber auch durch die zusätzlichen Informationen gesteuert zu verändern.

2. *Zusätzliche Graphikprimitive:* Zwischen dem dreidimensionalen Geometriemodell und der Beschreibung eines Bildes als zweidimensionale Anordnung von Pixeln gibt es weitere zweidimensionale Graphikprimitive, die *Strokes*. Diese sind dadurch charakterisiert, daß sie eine von Pixeln unabhängige Größe besitzen und vielfältig attribuitiert werden können.

3. *Bildmanipulationen:* Das Ergebnis des Renderings ist oft nicht das Endprodukt der nicht-photorealistischen Bilderzeugung. Bildverarbeitungs- und -manipulationstechniken werden häufig angewendet, um zusätzliche Informationen im Bild darzustellen bzw. die Darstellung des Bildes anhand zusätzlicher Informationen zu ändern. Einige NPR-Techniken verzichten auch gänzlich auf ein dreidimensionales Geometriemodell und sehen ein zweidimensionales Bild als Eingabedatum vor.

4. *Neue Transformationen*: Durch die Einführung der Strokes als zusätzliche Datenstruktur sowie die Einbeziehung von Modell- und Bildmanipulationen ergeben sich zusätzliche Transformationen, die zu einem integralen Bestandteil der Bilderzeugung werden.

Abbildung 3.3 zeigt diese Unterschiede in einem Überblick über den Prozeß der nicht-photorealistischen Bilderzeugung. Der Begriff „rendering“ steht dabei für jeweils spezifische Umwandlungsalgorithmen. Strokes und Bilder werden häufig als Texturen verwendet (texturing). Strokes können auch direkt auf die Modelloberfläche aufgebracht werden (painting). Zu einem Gesamtsystem gehören neben den Datenstrukturen und den entsprechenden Transformationen noch die standardmäßigen Viewing- und Rendering-Techniken.

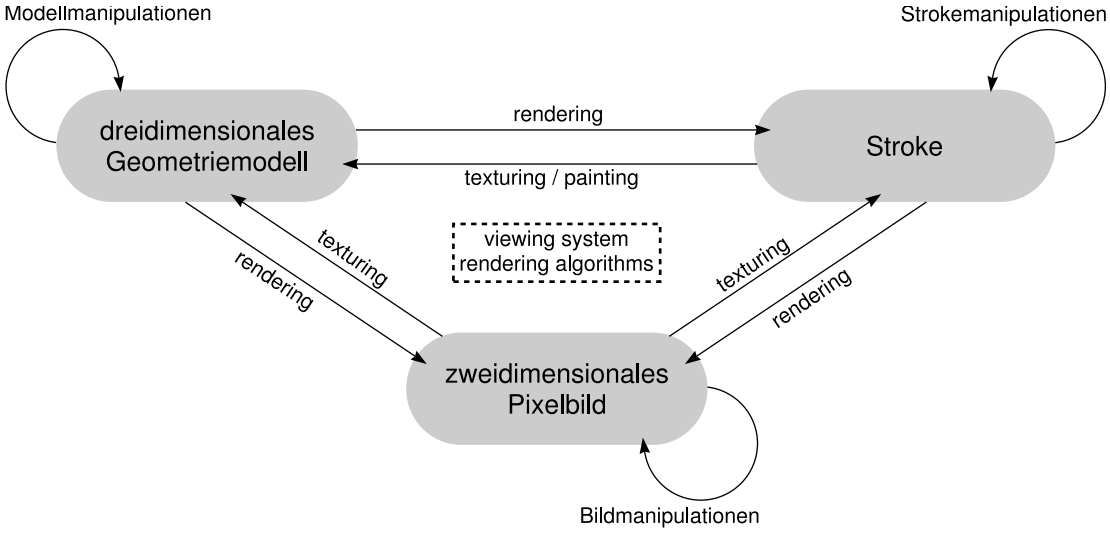



Abbildung 3.3. Datenstrukturen und Transformationen bei der nicht-photorealistischen Bilderzeugung.

Die in der Literatur vorgestellten Techniken der nicht-photorealistischen Computergraphik lassen sich nun den einzelnen Transformationen in Abbildung 3.3 zuordnen. Interessanterweise gibt es häufig viele verschiedene Methoden (=Transformationen), um einen speziellen visuellen Effekt zu erzeugen. Gleichzeitig ist es aber auch möglich, mit einer Methode und verschiedenen Parametern verschiedene visuelle Effekte zu generieren. Damit ergibt sich eine Erweiterung der Ausdrucksfähigkeit computergraphischer Techniken, die als ein Ziel des nicht-photorealistischen Renderings in Abschnitt 3.2 genannt wurde und die im folgenden am Beispiel des Stroke-Based Rendering illustriert werden soll.

## 3.4 STROKE-BASED RENDERING

Wie bereits in Abschnitt 3.3 dargelegt, werden in der nicht-photorealistischen Computergraphik *Strokes* als Komponenten eines Bildes bzw. als neue Datenstrukturen eingeführt. Strokes sind dabei Datenstrukturen, die zweidimensionale diskrete Bildelemente beschreiben, die typischerweise größer sind als ein Pixel auf dem Ausgabemedium [Her03]. *Stroke-Based Reendering* als eine nicht-photorealistische Basistechnik beschreibt dabei sowohl Algorithmen und Techniken, die das Aussehen der Strokes bestimmen als auch solche, die ihre Plazierung festlegen.

### 3.4.1 *Verwandte Arbeiten und Einordnung*

Stroke-Based Rendering umfaßt nach Hertzmann [Her03] automatische Techniken, nicht-photorealistische Bilder durch die Plazierung von Strokes verschiedener Art zu erzeugen. In dieser Arbeit wird der Begriff weiter gefaßt und umschließt alle Stroke-basierten Techniken unabhängig davon, ob die Plazierung der Strokes automatisch, manuell oder aufgrund zusätzlicher Informationen geschieht.

Die Vielzahl an möglichen Stilen bestimmt auch die Vielzahl an Techniken, die dem Stroke-Based Rendering zugeordnet werden. Diese unterscheiden sich zunächst nach dem Aussehen der Strokes und nach dem Plazierungsalgorithmus. Automatische Methoden versuchen hier, anhand von Optimierungsalgorithmen eine Verteilung der Strokes so zu finden, daß sowohl Helligkeits- als auch Textureigenschaften wiedergegeben werden. Eine Möglichkeit hierzu ist die Verteilung von Strokes über das gesamte Bild und eine nachfolgende Optimierung der Verteilung mit Hilfe von Voronoi-Diagrammen und dem Lloyd-Algorithmus. Deussen und Hiller [DHvS00, HHD03] sowie Secord [Sec02] zeigen dies für Stippling, Hausner für Mosaike [Hau01]. Diese Voronoi-basierten Methoden ermöglichen eine sehr gute geometrische Verteilung sich nicht überlappender Strokes aufgrund von Bildhelligkeiten.

Überlappen sich Strokes oder bestimmen Farbinformationen die Plazierung und Ausrichtung, können Verfahren herangezogen werden, die die Position eines Strokes zunächst vorschlagen und dieser Vorschlag wird dann dahingehend geprüft, ob er zu einer Optimierung der Verteilung führt. Ist dies der Fall, wird der Stroke ausgeführt. Diese Verfahren lag Haeberli's „Paint by Numbers" [Hae90] zugrunde, einer der ersten Arbeiten auf dem Gebiet. Hertzmann beschreibt ein Verfahren, mit dem auf diese Weise Bilder erstellt werden, die einem Gemälde ähnlich sehen [Her01]. Die vorgeschlagene Position des nächsten Strokes läßt sich sehr gut durch Greedy-Algorithmen ermitteln, indem die Stelle im Bild gesucht wird, bei der eine größtmögliche Änderung wahrscheinlich ist. Diese Technik wird unter anderem von Litwinowicz [Lit97] verwendet, um impressionistische Stilelemente in Stillbildern und Video zu erzeugen. Eine Kombination dieses Greedy-Ansatzes mit einer vorgegebenen Reihenfolge der Strokes untereinander, um auch Texturen zu erzeugen, wird von Salisbury et al. [SABS94] für zweidimensionale Ausgangsdaten (Bilder) und von Winkenbach und Salesin [WS94] für dreidimensionale Modelle vorgestellt. Die hier beschriebenen *(Prioritized) Stroke-Textures* bilden die Grundlage für weitere Anwendungen auf unterschiedliche Ausgangsdaten [WS96] sowie für die Behandlung weiterer Probleme im Zusammenhang mit der Ausrichtung der Strokes [SWHS97] und bei der Verwendung unterschiedlicher Ausgabegeräte [SALS96]. Die Positionierung der Strokes ist generell ein schwieriges Problem, wenn eine Verteilung erreicht werden soll, die manuell erstellten Bildern nahe kommt. Barla et al. [BBT+06] schlagen hierzu eine Technik vor, die eine handgezeichnete Graphik analysieren, die Charakteristika der enthaltenen Strokes extrahieren und über Texture Synthesis-Methoden eine neue Darstellung mit ähnlicher Textur erzeugen kann.

Der hauptsächliche Nachteil einer bildbasierten Positionierung der Strokes ist die

schlechte Eignung für Animationen, da mit dieser Techniken keine Frame-Kohärenz erzielt werden kann. Hier bieten Methoden Abhilfe, die die Strokes im Objektraum auf dem dreidimensionalen Modell positionieren und dann nach der Projektion im Bildraum zeichnen. Die erste Anwendung dieser Technik erfolgte für Pinselstriche durch Meier in [Mei96]. Meruvia et al. stellen in [MPFS03] ein Verfahren für animierte Stippling-Bilder vor, das sich durch die Positionierung der Stipple-Strokes an den Eckpunkten des polygonalen Geometriemodells und die Verwendung von Subdivision-Algorithmen auch für Echtzeit-Animationen eignet.

Die meisten der oben angeführten Arbeiten eignen sich insbesondere für kürzere Strokes, deren Zweck es ist, Flächen zu füllen, sei dies, um Helligkeiten, Farbe, Textur oder eine Kombination davon darzustellen. Oft sind aber auch lange Strokes notwendig, um Konturen darzustellen sowie weitere Kanten, die der Strukturierung des Modells bzw. des Bildes dienen. Sollen bildbasierte Kanten mit langen Strokes erzeugt werden oder sollen generell lange Strokes eingesetzt werden, um einen bestimmten Stil zu erzeugen, müssen andere als die bisher vorgestellten Techniken angewendet werden. Hertzmann stellt in [Her98] (siehe auch [Her03]) eine Technik vor, um beliebig lange „Pinselstriche“ erzeugen zu können. Ein genereller Ansatz zur Lösung des Problems der Plazierung langer Strokes benutzt Vektorfelder und daraus berechnete Streamlines, wie dies von Jobard und Lefer in [JL97] gezeigt wird.

Wichtige Kanten im dreidimensionalen Modell, die durch Strokes dargestellt werden sollen, sind insbesondere Silhouetten. Zur Berechnung von Silhouetten stehen nach [IFH+03] drei Klassen von Algorithmen zur Verfügung. Bildbasierte Algorithmen verwenden G-Buffer (vgl. [ST90]), um Diskontinuitäten in geometrischen Merkmalen, die durch die Anwesenheit einer Silhouette entstehen, zu bestimmen. Diese Algorithmen zeichnen sich insbesondere dadurch aus, daß sie für Real-Time-Anwendungen verwendet werden können, wie dies von Mitchell et al. [MBC02] sowie Loviscach [Lov02] gezeigt wird. Hybride Algorithmen modifizieren das Modell oder dessen Transformationen im Objektraum und verwenden mehrere Render-Durchgänge, um durch Überzeichnen der unterschiedlich transformierten Modelle bildbasierte Silhouetten zu erhalten. Ein Beispiel hierfür ist die Verwendung des Stencil-Buffers bei einer Translation des Objektes in die vier möglichen Richtungen im Bild pro Render-Durchgang [Rus89]. Der $z$-Buffer in Kombination mit einer Verschiebung des Modells in $z$-Richtung zwischen zwei Render-Durchgängen wird von Rossignac zur Silhouetten-Bestimmung verwendet [RvE92], während Raskar und Cohen den $z$-Buffer mit unterschiedlichen Eigenschaften der Polygone (front- und back-facing) kombinieren [RC99]. Die dritte Gruppe der Algorithmen arbeitet komplett im Objektraum und bestimmt die Silhouette in dreidimensionalen Koordinaten. Verbesserungen des trivialen Ansatzes, Kanten zwischen einem dem Betrachter zugenwandten und einem vom Betrachter abgewandten Polygon als Silhouetten zu verwenden, stellen Buchanan und Sousa unter Verwendung einer speziellen Datenstruktur vor [BS00], sowie Card und Mitchell unter Benutzung programmierbarer Graphikhardware [CM02]. Gooch et al. [GSG+99] sowie Benichou und Elber [BE99] bestimmen die Silhouetten eines Objektes durch Projektion der Flächennormalen auf eine Gauss-Kugel und entsprechende Berechnungen in diesem Dualraum. Hertz-

→ Anhang C
A Developer's Guide to Silhouette Algorithms for Polygonal Models

MANN und ZORIN stellen einen ähnlichen Ansatz vor unter Verwendung eines vierdimensionalen Dualraumes [HZ00].

In den letzten Jahren standen insbesondere zwei Probleme beim Stroke-Based Rendering im Mittelpunkt. Einerseits sollten solche Algorithmen entworfen werden, die die Eigenschaften moderner Graphikhardware ausnutzen und damit an Geschwindigkeit gewinnen. Andererseits soll die Qualität der Darstellungen verbessert werden, was durch spezielle Antialiasing-Verfahren gelingt, durch die direkte Verwendung von Vektorgraphiken [IBSC05] oder durch extra abgestimmte Rendering-Techniken, wie sie von ZANDER et al. vorgestellt wurden [ZISS04b, ZISS04a].

→ Anhang D
High Quality Hatching

### 3.4.2 *Informationsdarstellung mittels Strokes*

Strokes eignen sich sehr gut, um sowohl geometrische als auch weitere zusätzliche Informationen in Bildern darzustellen. Position und Verlauf der Strokes können genutzt werden, um geometrische Merkmale zu kodieren, wie beispielsweise den Verlauf von Silhouetten oder anderer Kanten im dreidimensionalen Modell. Bildbasiert lassen sich hier Diskontinuitäten in Farbe und Helligkeit graphisch hervorheben. Die breite Attributierbarkeit von Strokes, die weit über die Änderung der Farbe eines Pixels hinausgeht, erlaubt es, nicht-geometrische Informationen darzustellen, wenn diese entsprechend auf die Attribute der Strokes abgebildet werden. Im folgenden werden drei Arbeiten vorgestellt, die diese These belegen.

#### 3.4.2.1 *Silhouetten*

Silhouetten bilden eine Grundlage für die Wahrnehmung geometrischer Eigenschaften eines dreidimensionalen Objektes in einem zweidimensionalen Bild. Daher sind sie ein wichtiger Bestandteil bildlicher Darstellungen. Durch die Anwendung Strokebasierter Techniken kann die Darstellung von Silhouetten an die wiederzugebenden Informationen und an die zu unterstützende Anwendung angepaßt werden.

In einer Übersicht über Verfahren zur Berechnung und Darstellung von Silhouetten von polygonalen Modellen (siehe [IFH+03]) werden verschiende Silhouetten-Algorithmen untersucht und eine Klassifikation in drei Gruppen vorgeschlagen:

→ Anhang C
A Developer's Guide to Silhouette Algorithms for Polygonal Models

- Objektraum-Algorithmen,
- Bildraum-Algorithmen und
- hybride Algorithmen.

Jede dieser Gruppen hat Vor- und Nachteile, die die Auswahl eines entsprechenden Algorithmus abhängig von der gewünschten Anwendung betimmt. In Tabelle 3.1 sind diese Eigenschaften und ihre Anwendung zusammengefaßt.

Tabelle 3.1. Silhouetten-Algorithmen

| | | |
|---|---|---|
| Bildraum | $\oplus$ | • schnell, Komplexität nicht abhängig von Größe des Modells<br>• existierende Hard- und Software nutzbar<br>• Hardwarebeschleunigung möglich<br>• Hidden Line Removal simultan zur Silhouettenerkennung<br>• zusätzlich automatische Erkennung von Feature-Kanten |
| | $\ominus$ | • beschränkt auf Pixelgenauigkeit<br>• Ergebnis wenig beeinflußbar<br>• Weiterverarbeitung schwierig<br>• kaum Möglichkeiten zur Stilisierung |
| Objektraum | $\oplus$ | • sehr genaue Ergebnisse (bis auf Sub-Polygon-Genauigkeit)<br>• Beschleunigung durch Preprocessing möglich<br>• Silhouetten als vektorielle Daten<br>• Rendering unabhänging von Silhouettenerkennung<br>• sehr leichte Stilisierung und Weiterverarbeitung |
| | $\ominus$ | • oftmal hohe Renderingzeiten<br>• anspruchsvollere Algorithmen<br>• kaum Hardwarebeschleunigung möglich<br>• Hidden Line Removal als zusätzlicher Schritt notwendig |
| Hybrid | $\oplus$ | • existierende Hard- und Software nutzbar<br>• relativ schnelle Berechnung<br>• erweiterter Einfluß auf Ergebnisse |
| | $\ominus$ | • beschränkt auf Pixelgenauigkeit<br>• Weiterverarbeitung schwierig<br>• kaum Möglichkeiten zur Stilisierung |

#### 3.4.2.2 *Schraffuren*

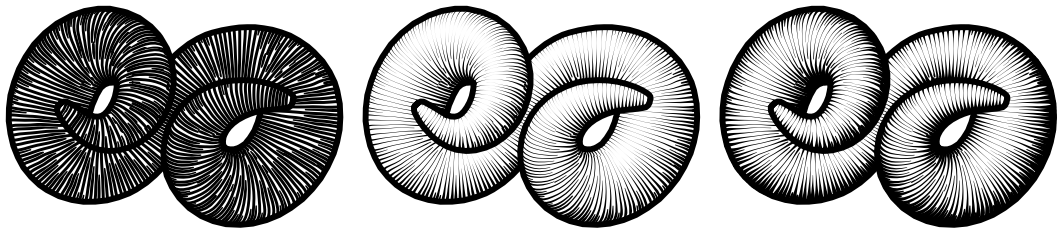
Neben Silhouetten spielen Schraffuren eine wesentliche Rolle bei nicht-photorealistischen Darstellungen. Im Gegensatz zu Silhouetten besteht ihre Funktion darin, Helligkeits- und Farbinformationen sowie Texturen abzubilden. Daher werden Schraffuren als flächenfüllende Technik eingesetzt. Die Erzeugung und spezielle Algorithmen zur Darstellung von Schraffuren wird in [ZISS04b] vorgestellt.

→ Anhang D
High Quality Hatching

Die grundlegende Idee besteht darin, Krümmungsinformationen zur Darstellung der Form eines Objektes zu benutzen. Dazu wird ein zweistufiges Verfahren vorgeschlagen, das zunächst potentielle Schraffurlinien im Objektraum bestimmt und diese dann im Bildraum visualisiert. Die Berechnung der Schraffurlinien erfolgt auf Basis eines Vektorfeldes, das für jeden Eckpunkt des gegebenen polygonalen Modells die Hauptkrümmungsrichtung enthält. Dieses Vektorfeld wird dann einer Optimierung unterzogen, um Artefakte (z. B. durch Rauschen und Inhomogenitäten) zu verringern. Abschließend werden durch Integration aus dem Vektorfeld Strömungslinien erzeugt, die die Basis für die Schraffurlinien bilden.

Zur Visualisierung dieser Schraffurlinien werden zunächst die verdeckten Linien und Linienteile unter Benutzung eines hybriden Algorithmus zur Bestimmung sichtbarer Linienteile entfernt. Die resultierenden Linien werden dann in einer Stilisierungsphase mit Shading-Informationen versehen und schließlich angepaßt für ein spezielles Ausgabegerät dargestellt. Das Shading erfolgt abhängig von verschiedenen Beleuch-

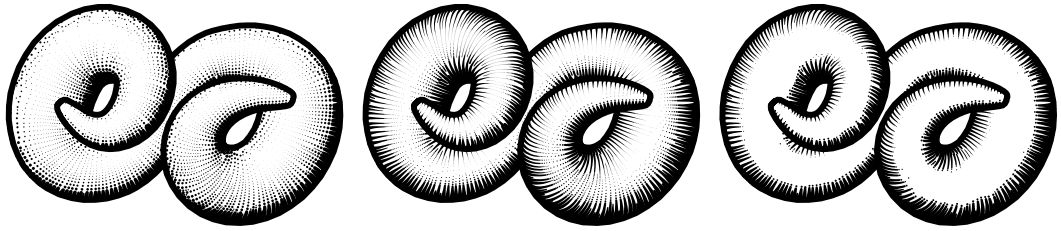
tungsparametern, die per Vertex zur Laufzeit berechnet werden und durch die dann Breite und Grauwert der Linien über ihren Verlauf gesteuert werden. Für Liniengraphiken typische Stile, wie Punktierungen oder Strichelungen können ebenfalls auf diese Weise erzeugt werden, wie Abbildung 3.4 zeigt.



(a) Line width = 1
Line density = 1

(b) Line width = $\mathtt{light}$
Line density = 1

(c) Line width = $\mathtt{light} + \mathtt{rim}$
Line density = 1



(d) Line width = $\mathtt{light} + \mathtt{rim}$
Line density = $\mathtt{light}$

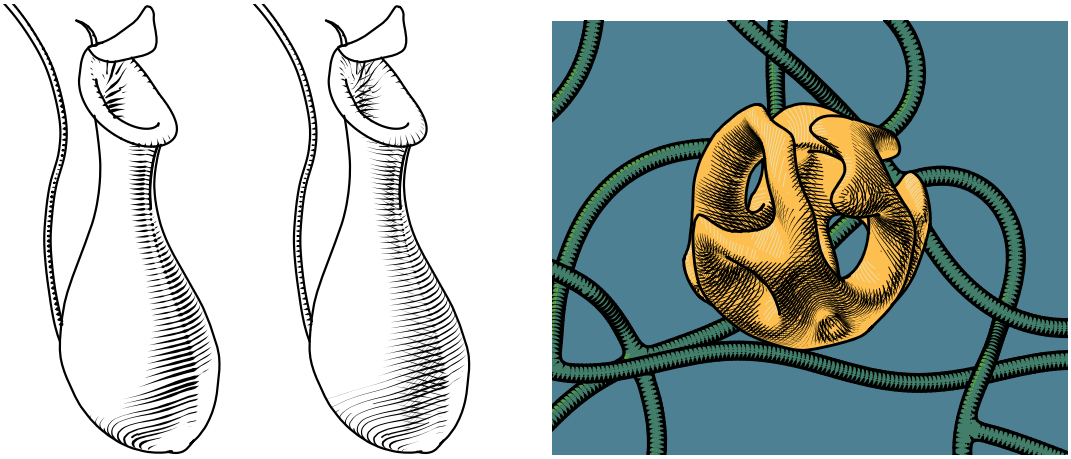
(e) Line width = $\mathtt{light} + \mathtt{rim}$
Line density = $5\mathtt{rim}$

(f) Line width = $\mathtt{light} + \mathtt{rim}$
Line density = $(5\mathtt{rim})^{50}$

Abbildung 3.4. Anpassung der Stile unter Nutzung verschiedener Beleuchtungsinformationen (*light* entspricht dem diffusen Anteil des Phong-Beleuchtungsmodells, *rim* entspricht dem diffusen Anteil des Phong-Beleuchtungsmodells mit einer Lichtquelle zwischen Betrachter und Objekt). Die Linienbreite und -dichte (entspricht der Stärke der Stippling-Effektes) wird modifiziert. Allen Bildern liegt die gleiche Menge an aus dem Krümmungsfeld berechneten Linien zu Grunde.

Die Berechnung des Bildes erfolgt abhängig vom Ausgabemedium, so daß für eine interaktive Bildschirmdarstellung eine qualitativ niedrigere, aber ressourcenschonendere und schnellere Ausgabe erfolgt, wohingegen die Ausgabe für Drucker qualitativ hochwertig und an die Eigenschaften des Mediums angepaßt berechnet wird. Hier ist auch eine Kombination mehrerer Ebenen möglich, die durch Rotation des initialen Vektorfeldes zu Kreuzschraffuren führt (siehe Abbildung 3.5(a)). Die Ausgabe für den Druck kann auch durch verschieden farbige Ebenen unter Anwendung von Volltonfarben realisiert werden (siehe Abbildung 3.5(b)).

Schraffuren und Silhouetten sind ein wirksames Mittel, um geometrische Informationen über die Form eines Objektes darzustellen. Dabei bietet die Oberflächenkrümmung kombiniert mit Beleuchtungsinformationen ein sehr hohes Potential. Techniken aus dem Bereich des Stroke-Based Rendering bieten hier ein effektives Werkzeug zur Umsetzung, da die einzelnen Strokes vielfältig attributiert werden können und somit nicht nur durch ihre Farbe, sondern auch durch Position, Verlauf und Stil zur Visualisierung eingesetzt werden können.



(a) Schraffuren aus einer Ebene (links) und drei Ebenen (rechts), die durch Rotation des initialen Vektorveldes entstanden sind.

(b) Komposition aus sechs Ebenen und fünf Farben sowie schwarz.

Abbildung 3.5. High Quality Hatching

#### 3.4.2.3 *Bildbasierte Strokes*

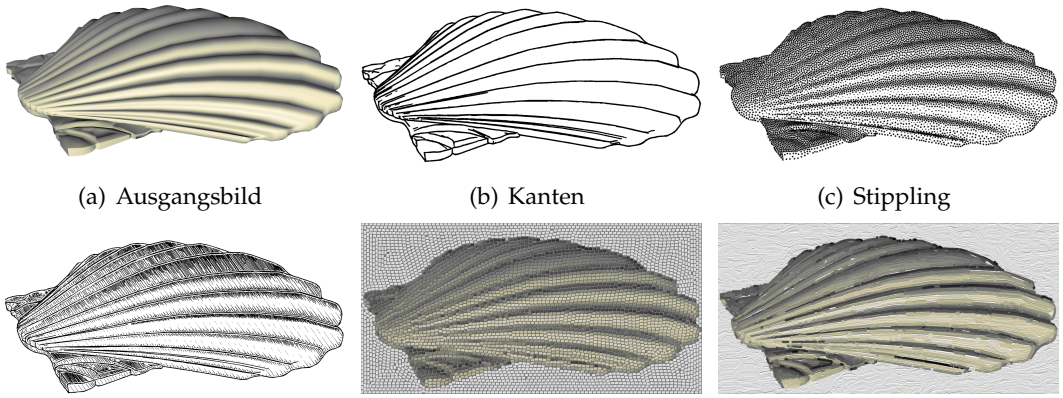
Sowohl Silhouetten als auch Schraffuren stellen dreidimensionale Informationen dar, die sich durch die Geometrie der Objekte oder die Beleuchtungsverhältnisse ergeben. Stroke-basierte Rendering-Algorithmen können auch bildbasiert angewendet werden. Dabei werden dann im Bild kodierte Informationen, wie z. B. Farbe, Helligkeit oder Diskontinuitäten dargestellt.

→ Anhang E RenderBots – Multi Agent Systems for Direct Image Generation

In [SGS05] wird ein Verfahren vorgestellt, solche Stroke-basierten Abbildungen unter Verwendung von Multiagentensystemen zu erzeugen. Damit wird eine einheitliche Behandlung aller Stroke-basierten Stile erreicht. Das grundlegende Problem bei bildbasierten Algorithmen ist die Verteilung der Strokes gemäß den Eigenschaften des Ausgangsbildes, während gleichzeitig die Charakteristiken des zu emulierenden Stils beachtet werden müssen. Der vorliegende Ansatz verwendet relativ einfache autonome Agenten (*RenderBots*), die sich in einer künstlichen Umgebung bewegen. Die Agenten repräsentieren oder erzeugen die dem Stil entsprechenden Strokes. Die Umgebung besteht aus dem Ausgangsbild und eventuellen zusätzlichen Informationen in Form von G-Buffers.

Für unterschiedliche Stile werden unterschiedliche Klassen von RenderBots verwendet, die sich in ihrem Schwarmverhalten und ihren Fähigkeiten zur Bildgenerierung unterscheiden. Die Verteilung der Strokes im Bild wird über das Schwarmverhalten realisiert, das sich an das von Multiagentensystemen zur Simulation von Vogelschwärmen und ähnlichem im Bereich der Computeranimation anlehnt [Rey87]. Alle RenderBot-Klassen basieren auf den selben grundlegenden Verhaltensmustern und können ihre Umgebung (d. h. das zu reproduzierende Ausgangsbild sowie andere RenderBots in ihrer Nähe) wahrnehmen. Um unterschiedliche Stile zu erzeugen, be-

sitzen die RenderBot-Klassen unterschiedliche Fähigkeiten, die benötigten Strokes zu generieren. Während bei Stippling und Mosaik ein einzelner RenderBot jeweils einen Stroke repräsentiert, werden die anderen Stile dadurch erzeugt, daß RenderBots bei ihrer Bewegung in der künstlichen Umgebung je nach vorgefundenen Bedingungen Spuren hinterlassen. Abbildung 3.6 zeigt einige mit dieser Technik aus einem Ausgangsbild (Abbildung 3.6(a)) entstandene Darstellungen.



(a) Ausgangsbild (b) Kanten (c) Stippling

(d) Schraffuren (e) Mosaik (f) Pinselstriche

Abbildung 3.6. Unterschiedliche durch RenderBots aus einem Modell erzeugte Stile

Werden zusätzliche Informationen, wie eine Segmentierung des Ausgangsbildes, Informationen über Kanten oder Krümmungseigenschaften (wenn das Ausgangsbild aus einem dreidimensionalen Modell entstanden ist) hinzugenommen, läßt sich das Verhalten der RenderBots sehr detailliert parametrisieren und dadurch die Qualität der erzeugten Graphiken weiter steigern. Eine Segmentierung des Ausgangsbildes erlaubt beispielsweise das Mischen unterschiedlicher Stile in einer Darstellung, wie dies in Abbildung 3.7 zu sehen ist.

Auch für die Darstellung bildbasierter Informationen stellt Stroke-Based Rendering also einen effektiven Ansatz dar. Hier ist ebenfalls durch die weitgehende Parametrisierbarkeit der Darstellung die Möglichkeit gegeben, sehr pointiert und selektiv unterschiedliche Aspekte der Darstellung in den Mittelpunkt zu rücken.

## 3.5 ZUSAMMENFASSUNG

Die (generative) Computergraphik stellt Algorithmen und Techniken zur Verfügung, um geometrische Daten effektiv zu visualisieren. Betrachtet man phororealistische Rendering-Verfahren, ist deren Ziel einzig und allein, die Geometrie und Beleuchtungsverhältnisse einer Szene so realitätsgetreu wie möglich darzustellen. Die dabei verwendeten Mittel sind demetsprechend limitiert und beschränken die Ausdrucksfähigkeit computergraphischer Darstellungen, die sich nur dieser Mittel bedienen. Mit dem Aufkommen nicht-photorealistischer Techniken wurde die Ausdrucksfähigkeit computergraphischer Darstellungen stark erweitert, so daß jetzt auch die Vi-

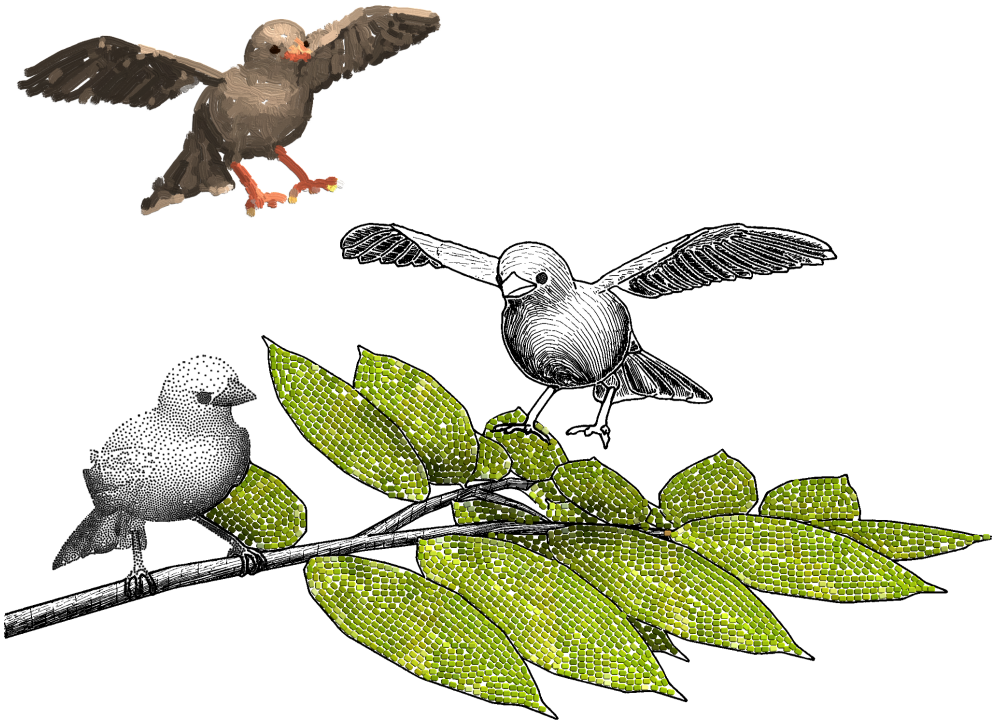


Abbildung 3.7. Mischen verschiedener Stile durch Beschränkung der RenderBots auf einzelne Objekte.

sualisierung von Informationen möglich ist, die über Geometrie oder Beleuchtung hinausgehen.

Die Entwicklungen im Bereich des NPR lassen sich dabei einerseits nach dem Stil gruppieren, der erzeugt wird, andererseits nach der verwendeten Technik. Danach ist es möglich den gleichen Stil mit verschiedenen Techniken zu erreichen bzw. durch die Parametrisierung einer gegebenen Technik verschiedene Stile zu erzeugen (siehe Abbildung 3.7 als Beispiel). Der Gewinn an ausdrucksstarken Techniken in der Computergraphik geht allerdings mit einem Zuwachs an Freiheitsgraden beim Erstellen von Bildern einher. Das bedeutet, daß ein Mehr an Möglichkeiten, expressive Darstellungen zu erzeugen, eventuell mit einer Verringerung der Effektivität und Angemessenheit verbunden ist. Hier ist der Ersteller der Graphiken noch stärker in den Bilderzeugungsprozeß einzubeziehen, um eine korrekte Kommunikation mit dem Bild als Medium sicherzustellen.

# 4 INFORMATIONSVISUALISIERUNG

Die *Visualisierung* beschäftigt sich mit der bildlichen Veranschaulichung von Daten und ihrer relevanten Aspekte. Das Ziel der Visualisierung ist es nach Schumann und Müller [SM00], geeignete visuelle Repräsentationen einer gegebenen Datenmenge zu erzeugen, um damit eine effektive Auswertung zu ermöglichen. In diesem Sinne ist die im Kapitel 3 beschriebene Computergraphik eigentlich auch als Visualisierung – nämlich als Visualisierung geometrischer Daten – zu verstehen. Im Rahmen der Visualisierung haben sich insbesondere zwei Teilgebiete herausgebildet, wobei die Grenze zwischen ihnen nicht eindeutig klar definiert ist:

1. Die *wissenschaftliche Visualisierung* (engl.: scientific visualization, oft auch als Datenvisualisierung bezeichnet) beschäftigt sich hauptsächlich mit der graphischen Darstellung von Meßwerten und Daten aus wissenschaftlichen Experimenten und (Computer-)Simulationen.
2. Die *Informationsvisualisierung* beschäftigt sich hauptsächlich mit der graphischen Darstellung abstrakter Daten.

Ziel beider Zweige ist es, den kognitiven Zugang zu den Daten zu erleichtern. Im Rahmen von interaktiven Systemen bieten visuelle Darstellungen der zugrundeliegenden Daten häufig eine Grundlage zur Interaktion mit der Darstellung und/oder mit den Daten.

Im folgenden werden zunächst einige allgemeine Aspekte der (Informations-)Visualisierung besprochen, bevor ein besonderer Zweig, die Visualiserung im Umfeld von digitalen Dokumenten, im Mittelpunkt steht. Hier werden dann auch einzelne Arbeiten vorgestellt.

## 4.1 VISUALISIERUNGS-PIPELINE UND BEISPIELE

Der Prozeß des Erstellens einer Visualisierung kann grob in drei Schritte unterteilt werden, die sich in der Visualisierungs-Pipeline niederschlagen (siehe Abbildung 4.1, nach [SM00]) Für eine graphische Darstellung abstrakter Daten müssen diese zunächst aufbereitet werden (Filtering). Dabei kommen Operationen zur Vervollständigung, Reduzierung und/oder Konvertierung der Rohdaten zum Einsatz. Anschließend werden die so behandelten Daten in ein Modell für die Darstellung überführt (Mapping). Dabei handelt es sich um eine Beschreibung der darzustellenden Geometrie in unterschiedlichen Ausprägungen. Dieses Modell wird in einem abschließenden Schritt (Rendering) in ein Bild überführt. Hier besteht ein weiterer Zusammenhang zwischen Visualisierung und Computergraphik. In diesem letzten Schritt der Visualisierungs-Pipeline werden Algorithmen der Computergraphik genutzt, um

aus dem nun vorliegenden geometrischen Modell ein Bild zu erzeugen. Dieser Schritt kann durchaus den kompletten photorealistischen oder nicht-photorealistischen Bilderzeugungsprozeß umfassen.

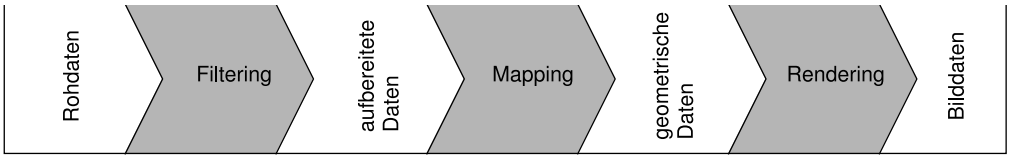


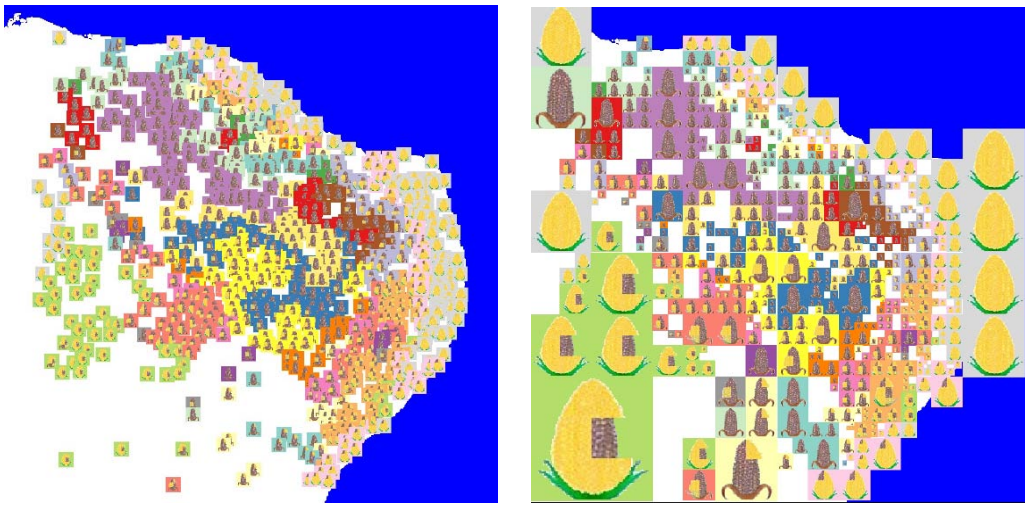

Abbildung 4.1. Visualisierungspipeline

Als ein Beispiel für die Anwendung nicht-photoprealistischer Rendering-Verfahren in der Visualisierung dient die Arbeit „Icon-Based Visualization Using Mosaic Metaphors“ [NSS05]. Die zu visualisierenden Daten bestanden aus Klimawerten und Ernteerträgen in Brasilien. Ziel war es, den Zusammenhang zwischen dem Niederschlagsverlauf (insbesondere während einer aufgetretenen schweren Dürrezeit 1983) und den Ernteerträgen darzustellen. Für jede Meßstation lag ein siebendimensionaler Merkmalsvektor vor, in dem ein Merkmal die Zugehörigkeit zu einem Cluster beschreibt. Daher wurde sich für eine Icon-basierte Visualisierungstechnik entschieden, bei der in jedem Icon die sechs anderen Merkmale kodiert sind. Die Clusterzugehörigeit sollte durch das Layout der Icons auf einer Karte dargestellt werden. Angelehnt an Techniken zur Erzeugung von Photomosaiken, wie sie aus der nichtphotorealistischen Computergraphik bekannt sind (siehe hier [FR98]), wurden verschiedene Layout-Varianten untersucht. Abbildung 4.2 zeigt zwei dieser Varianten, eine gestreute Anordnung der Icons und eine Anordnung, bei der sich die Größe der Icons an die Clusterung der Daten anpaßt. Größere Cluster, die bei der Variante aus Abbildung 4.2(a) durch mehrere Icons oder durch größere Abstände zwischen den Icons repräsentiert wurden, werden nun durch größere Icons dargestellt. Die neuen Layouts für Icon-basierte Visualisierungen erlauben die Exploration von Daten auf unterschiedlichen Detailstufen und insbesondere die Anpassung der Darstellung und eine mögliche Clusterung der Daten, wie sie in Abbildung 4.2(b) gezeigt ist.

→ Anhang F
Icon-Based Visualization Using Mosaic Metaphors

Typische Anwendungen der Informationsvisualisierung erstellen im Mapping-Schritt allerdings zweidimensionale Modelle, die dann in diagrammähnliche Darstellungen überführt werden. Ein Beispiel hierfür wird in der Arbeit „Connecting Time-Oriented Data and Information to a Coherent Interactive Visualization“ untersucht [BSM04]. Hier werden zeitabhängige Daten im Rahmen eines interaktiven Systems dargestellt. Bei den Daten handelt es sich um Behandlungspläne eines Patienten, die mit aktuellen Meßwerten für Vitalfunktionen verbunden werden sollen. Eine von *LifeLines* [PMR+96] abgeleitete Technik bestimmt zunächst das Layout der Behandlungsabschnitte und der zugeordneten Meßwerte in einer graphischen Übersicht über den gesamten Zeitraum. Hier werden unterschiedliche Repräsentationen verwendet, wie beispielsweise Icons für Ereignisse oder Graphen für Meßreihen. Diese Meßreihen werden wiederum in zweidimensionalen Diagrammen dargestellt, die sich einerseits dem jeweiligen Detailgrad (und damit der Größe des Diagramms im Behandlungsplan) anpassen und die andererseits zusätzliche aus den Daten berech-

→ Anhang G
Connecting Time-Oriented Data and Information to a Coherent Interactive Visualization



(a) Scattered Layout zur Visualisierung der Daten einzelner Meßstationen

(b) QuadTree-basiertes Layout zur Visualisierung größerer zusammenhängender Cluster.

Abbildung 4.2. Mosaic-basierte Visualisierung multivariater Daten am Beispiel eines Datensatzes mit Klima- und Landwirtschaftsdaten in Brasilien.

nete Informationen (statistische Informationen, Schwellwerte, Aggregationen) durch unterschiedliche Methoden (Farbe, Vereinfachungen, usw.) kodieren. Verschiedene Focus+Context-Techniken bzw. Zoom-Methoden komplettieren das vorgestellte System, das in Abbildung 4.3 zu sehen ist.

Der in Kapitel 2 aufgestellte Katalog mit Qualitätsmerkmalen wurde bisher in der Literatur hauptsächlich im Kontext von (Information-)visualisierung verwendet. Die Diskussion der Qualitätsmerkmale ähnelt hier der zur nicht-photorealistischen Computergraphik, da ähnliche Ziele und Aufgaben im Mittelpunkt stehen.

Expressivität: Auch in der Informationsvisualisierung erschwert die Vielzahl der zur Verfügung stehenden Techniken die Erstellung expressiver Darstellungen. Expressivität erfordert, daß *nur* die in der Datenmenge enthaltenen Daten visualisiert werden. Dies kann insbesondere durch eine entsprechend zielgerichtete Auswahl der Darstellungstechniken und der zu verwendenden visuellen Variablen im Mapping-Schritt der Visualisierungspipeline erfolgen.

Effektivität: Die intuitive Wahrnehmbarkeit einer Visualisierung ist nur dann gegeben, wenn die wahrnehmungspsychologischen Prinzipien beispielsweise hinsichtlich Anordnung der Elemente oder Farbgebung eingehalten werden. Da in der Informationsvisualisierung abstrakte Daten dargestellt werden, spielt dies eine noch größere Rolle als bei der Darstellung geometrischer Daten, die bereits inhärent über ihre Form erkannt werden können.

Angemessenheit: Der mentale Aufwand zum Dekodieren einer Visualisierung ist höher als bei der Darstellung rein geometrischer Daten, da die im Mapping-Schritt getroffenen Zuordnungen der Eigenschaften der Daten zu visuellen Va-

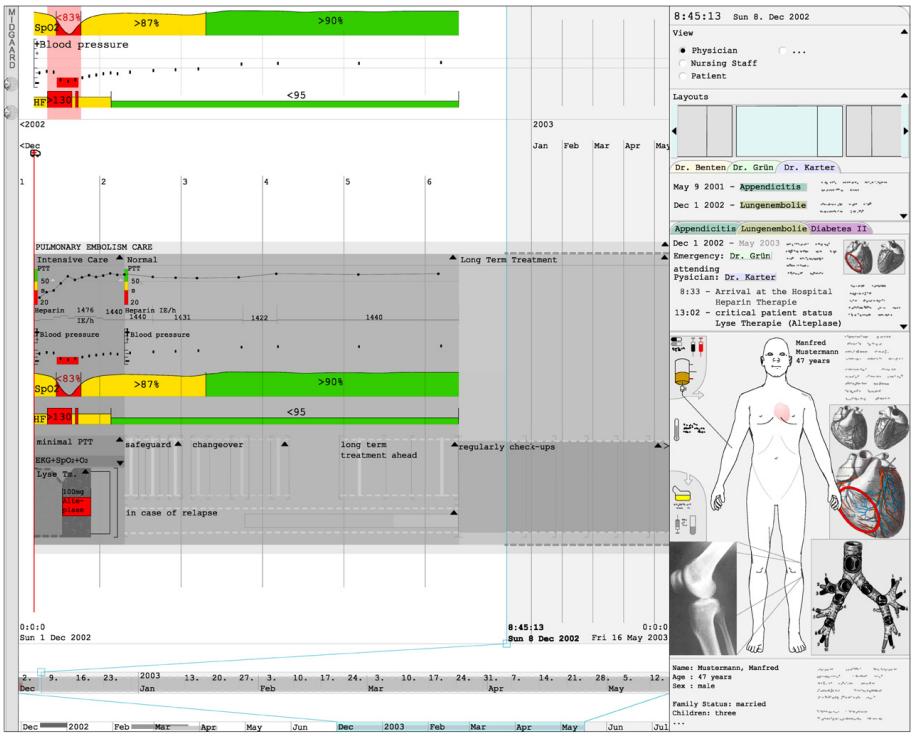



Abbildung 4.3. Screenshot des Systems Midgaard zur kohärenten Visualisierung von Behandlungsaplänen und Patientendaten.

riablen erkannt werden müssen. Die technischen Verfahren zur Erzeugung von Visualisierungen hängen in ihrer Komplexität auch vom Umfang und der Art der zu visualisierenden Daten ab. Die Vorbereitung der Daten im Filtering-Schritt der Visualisierungspipeline kann sich bei hochdimensionalen oder sehr stark strukturierten Daten z. B. als sehr aufwendig herausstellen.

Im folgenden wird der Einsatz der Informationsvisualisierung im Umfeld digitaler Dokumente näher beleuchtet und anhand dreier Beispiele die Informationsdarstellung in diesem Kontext betrachtet.

## 4.2 Informationen in und über digitale Dokumente

Digitale Dokumente spielen im Alltag eine immer größere Rolle und versuchen an vielen Stellen, die Papierdokumente zu ersetzen. Dies kann nur dann erfolgreich geschehen, wenn digitale Dokumente einen deutlichen Mehrwert gegenüber herkömmlichen Papierdokumenten aufweisen. Dazu muß einerseits der Umgang mit ihnen vereinfacht werden, andererseits können die zusätzlichen Informationen ausgenutzt werden, die dadurch vorhanden sind, daß digitale Dokumente ebenfalls „Daten“ im Sinne der Informatik sind.

Digitale Dokumente bestehen nicht nur aus dem eigentlichen textuellen Inhalt (und

eventuell vorhandene weitere multimediale Inhalte wie Bilder, Video oder Audio), sondern beinhalten auch Informationen über eine Strukturierung dieser Inhalte oder weitere Zusammenhänge. Diese Informationen können ausgenutzt werden, um verschiedene Aufgaben, wie z. B. Suchanfragen oder die Navigation im Dokument zu unterstützen. Eine solche Unterstützung ist Gegenstand der Abschnitte 4.2.2 und 4.2.3. Zunächst wird allerdings eine Einordnung dieser Techniken in das Gebiet vorgenommen und verwandte Arbeiten vorgestellt.

### 4.2.1 *Verwandte Arbeiten und Einordnung*

Arbeiten, die sich mit dem Umfeld der Visualisierung von Textdokumenten und den in ihnen enthaltenen (zusätzlichen) Informationen beschäftigen, lassen sich grob in vier Bereiche einteilen:

- Visualisierung des Dokumentes and sich bzw. der Textstruktur mit überlagerten zusätzlichen Informationen,
- Visualisierung von mehreren Dokumenten und Beziehungen zwischen ihnen,
- Visualisierung hierarchischer Strukturen und
- Visualisierung von relationalen Informationen.

Eines der Hauptprobleme bei der Visualisierung von Dokumenten besteht im großen Datenumfang. Hier müssen Techniken gefunden werden, die diese Datenmenge zumindest visuell reduzieren. Jerding und Stasko legten mit *Information Murals* [JS98] einen Grundstein für solche Techniken. Ein *Information Mural* ist eine zweidimensionale, verkleinerte Repräsentation eines gesamten Informationsraumes. Dabei werden visuelle Attribute wie Farbe und Intensität zusammen mit Kompressionstechniken, die dem Antialiasing ähneln, verwendet, um Informationsattribute und deren Dichte abzubilden. Eick verwendete diese Technik zunächst in seinem System *SeeSoft*, um den Quellcode größerer Softwaresysteme zu analysieren [ESS92], später dann auch zur Textanalyse [Eic94]. Nachteilig ist hier die Orientierung am Seiten- und Textlayout, da die Murals auf der Basis der einzelnen (Quell-)Textzeilen berechnet werden. Eine Unterteilung des Textes in einzelne strukturelle Abschnitte (Absatz, Abschnitt, Kapitel) liegt den *TileBars* von Hearst zugrunde [Hea95]. Eine Icon-ähnliche Darstellung dieser Abschnitte ermöglicht eine Erweiterung der Schlüsselwortsuche dadurch, daß Fundstellen farbig hervorgehoben werden und so eine übersichtliche Darstellung der Fundstellen im Kontext des Dokumentes gegeben ist. Allerdings wird hier die hierarchische Struktur des Dokumentes nicht beachtet.

Werden mehrere Dokumente in eine Visualisierung einbezogen, ist es oft von Interesse, Beziehungen zwischen diesen darzustellen. Die *Perspective Wall* von Mackinlay et al. [MRC91] stellt zeitliche Bezüge durch die Positionierung der einzelnen Dokumente (bzw. Dokumentdateien) in einer zweidimensionalen Anordnung her. Während die horizontale Achse die Zeit repräsentiert, können auf der vertikalen Achse verschiedene Kategorisierungen visualisiert werden. Eine besondere perspektivische

Betrachtungstransformation ergibt eine Focus+Context-Ansicht, die zentrale Dokumente hervorhebt. Große Anstrengungen werden ebenfalls unternommen, thematische Bezüge zwischen Dokumenten zu visualisieren. Dazu ist allerdings eine tiefergehende Analyse der Dokumente notwendig. Ein Clustering der Analyse-Ergebnisse faßt thematisch zueinandergehörige Dokumente zusammen. Diese Cluster können dann verschiedentlich visualisiert werden. Zwei Beispiele für eine solche Visualisierung werden von Wise et al. in [WTP+95] vorgestellt. Eine Möglichkeit besteht in der Anordnung der Dokumente als Sterne, so daß die Cluster Galaxien – *Galaxies* – bilden. Die zweite Technik besteht in der Visualisierung der geclusterten Dokumente als Höhenkarte – *ThemeScape* – so daß höhere „Berge" umfangreichere Cluster repräsentieren. Diese Techniken sind eher für digitale Bibliotheken interessant, bei denen eine große Menge an Dokumenten und ihre Beziehungen zueinander dargestellt werden müssen.

Bei der Betrachtung mehrerer Dokumente spielen neben dem thematischen Zusammenhang auch relationale Beziehungen eine bedeutende Rolle. Insbesondere Zietierungen und Ko-Autorschaften werden häufig bei der Analyse eines Fachgebietes bzw. bei der Literaturrecherche herangezogen, um weitere Literaturquellen zu finden. Digitale Bibliotheken zeigen hier noch wenig innovative Lösungen, diese Daten in die Informationssuche mit einzubeziehen. Einige Forschungsarbeiten stellen allerdings interessante Ansätze für diese Fragestellung vor. Während sich die Systeme *BibRelEx* von Bruggemann-Klein et al. [BKKL99] und *DBL-Browser* von Klink et al. [KLR+04] Standard-Techniken zur Visualisierung der Relationen bedienen, stellen einige Autoren sehr innovative Lösungen vor. *Monkellipse* von Hsu et al [HIMS04] nutz eine ellipsenförmige Visualisierung, um die einzelnen Dokumente, die auf der Ellipsenlinie dargestellt sind mit Themenbereichen im Inneren der Ellipse zu verbinden. Wong et al. [WHP+04] verwenden ebenfalls die Landkarten- oder Galaxie-Metapher in ihrem *InSpire*-System. Sließlich wird mit *WilmaScope* von Ahmed et al. [ADM+04] ein System vorgestellt, bei dem sehr vielfältige Relationen wie Zitierungen, Ko-Autorschaften oder die Zugehörigkeit zu Themenbereichen in einem komplexen Netzwerk modelliert und dieses Netzwerk dann visualisiert wird. Die Darstellungen können hier allerdings nicht interaktiv exploriert werden. Die Nutzung solcher Visualisierungen zur Exploration von digitalen Bibliotheken birgt ein großes Potential zur Vereinfachung des Zugriffs auf umfangreiche Literaturbestände.

Die Analyse von Zitierungen und Ko-Zitierungen sowie der Einordnung von Dokumenten in verschiedene Themenbereiche kann auch dazu verwendet werden, wissenschaftliche Trends zu erkennen bzw. die wichtigsten Arbeiten auf einem Gebiet zu finden. Chen stellt mit *Citespace* ein solches System vor [Che04, Che06]. Basierend auf dem Zitiernetzwerk sowie auf einer Textanalyse, die Begriffe aus den Titeln und Abstracts der Dokumente extrahiert, wird eine Visualisierung aufgebaut, in der durch Clusterung wissenschaftliche Trends sichtbar werden. Herausragende Arbeiten sind durch ihre Größe und Farbgebung gekennzeichnet. Die visuelle Inspektion dieses aus Relationen zwischen Dokumenten aufgebauten Graphen bringt dabei neue Informationen.

Ein einzelnes Dokument ist üblicherweise hierarchisch in Kapitel, Abschnitte, usw. unterteilt. Die Visualisierung hierarchischer Strukturen ist eines der Kernthemen in der Informationsvisualisierung und daher auch mit sehr vielen Techniken in der Literatur vertreten. Hierarchische Strukturen lassen sich generell duch Bäume modellieren, zwischen deren Knoten eine Vater-Sohn-Beziehung z. B. der Art „ist enthalten in“ besteht. Allgemein lassen sich zwei Klassen von Visualisierungstechniken unterscheiden. Bei den *expliziten* Techniken werden die Relationen zwischen den Informationsobjekten durch Kanten explizit dargestellt, während *implizite* Techniken spezielle Anordnungen bzw. Verschachtelungen nutzen, um diese Relationen zu veranschaulichen. Zu den expliziten Techniken zählen damit alle GraphDrawing-Methoden, die Knoten-Kanten-Diagramme nutzen. Je nach verwendetem Algorithmus unterscheiden sich die entstehenden Layouts, generell besteht aber das Problem, daß sehr große Hierarchien nur unzureichend dargestellt werden können. Hier besteht die Möglichkeit, dreidimensionale Darstellungen zu nutzen, wie dies bei *ConeTrees* und *CamTrees* geschieht [RMC91]. Diese dreidimensionale Darstellung benötigt allerdings eine spezielle Navigation und kann durch Verdeckungen von Teilen der Hierarchie zu Problemen führen. Zweidimensionale explizite Darstellungen für große Hierarchien nutzen beispielsweise die hyperbolische Geometrie, um das Baumlayout zu berechnen [LR94, LRP95, Mun97]. Dabei entsteht eine natürliche Focus+Context-Ansicht, bei der auf gleicher Fläche weit größere Baumstruklturen dargestellt werden können als mit herkömmlichen GraphDrawing-Algorithmen. Eine ähnliche Herangehensweise wird in [KS99] verfolgt. Hier wird das Baum-Layout zunächst auf einer Halbkugel berechnet und dann durch eine Projektion im Zweidimensionalen betrachtet.

Eine bessere Platzausnutzung kann durch die Verwendung impliziter Techniken erzielt werden. Der klassischen Ansatz der *Treemaps* von Shneiderman [Shn92] verschachtelt Hierarchieebenen ineinander, so daß niedrigere Ebenen immer komplett in höheren Ebenen enthalten sind. Jeder Knoten der Hierarchie wird als Rechteck dargestellt. Zusammen mit speziellen Layout-Berechnungen ergeben sich flächenfüllende Darstellungen in denen die hierarchische Struktur intuitiv erkennbar ist. Erweiterungen dieser grundlegenden Technik verändern das Layout der Knoten-Rechtecke, um ästhetisch ansprechendere Visualisierungen zu erhalten (*Squarified Treemaps* [BHvW00]) oder nutzen Shading, um die hierarchische Struktur weiter hervorzuheben (*Cushion Treemaps* [vWvdW99]). Neuere Ansätze verzichten auf eine rechteckige Darstellung der Knoten und verwenden andere Formen, wie beispielsweise Balzer et al. [BDL05], deren Treemaps auf Voronoi-Diagrammen basieren.

Sind zusätzlich zu einer Hierarchie weitere Relationen zwischen den Knoten vorhanden ist es relativ schwer, diese mit einer Hierarchievisualisierung zu verbinden. Eine Möglichkeit besteht wiederum darin, ein generelles GraphDrawing-Problem zu lösen, was allerdings ebenfalls mit den oben bereits genannten Problemem behaftet ist. Fekete et al. nutzen Treemaps, um die Hierarchie zu visualisieren und überlagern diese Darstellung mit einem Knoten-Kanten-Diagramm der zusätzlichen Relationen [FWDP03]. Diese Darstellungen integrieren zwar beide Arten von Relationen, werden aber sehr schnell unübersichtlich.

Relationale Informationen in linearen Daten, wie sie beispielsweise auch in digitalen Dokumenten vorkommen, können mit Hilfe von *Arc Diagrams* visualisiert werden. Diese Technik wurde ursprünglich von Wattenberg entwickelt, um Wiederholungen in langen Zeichenketten zu visualisieren [Wato2b]. Kerr stellte mit *Thread Arcs* eine Anwendug dieser Technik vor, mit der Relationen zwischen eMail-Messages visualisiert werden [Kero3].

### 4.2.2 *Schlüsselwortsuche*

Schlüsselwortsuche ist eine der häufigsten Aktionen in Verbindung mit der Arbeit mit digitalen Dokumenten und Bibliotheken. Außer einer (farbigen) Markierung der Fundstellen im Dokument wird heute aber kaum eine weitergehende Unterstützung zur Auswertung der Ergebnisse angeboten. In [SSWS04a] und [SSWS04b] wird mit *SearchTreemaps* eine Technik vorgestellt, die eine Einordnung der Ergebnisse in die Struktur des Dokuments vornimmt und somit eine schnelle Evaluierung der Güte der Fundstellen erlaubt.

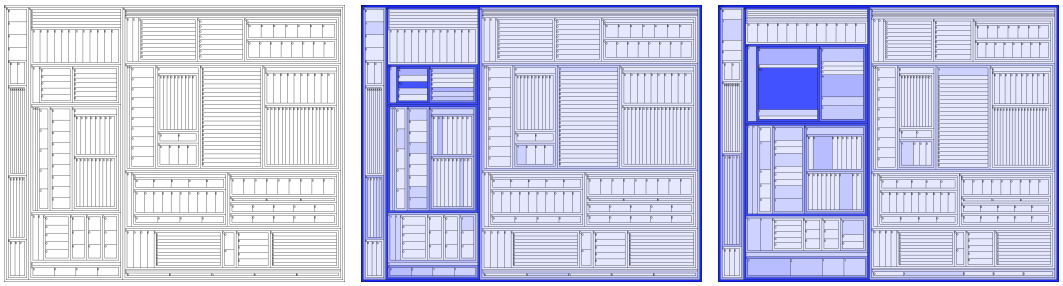
→ Anhang H
Interactive Treemaps With Detail on Demand to Support Information Search in Documents

Bei der Suche nach Schlüsselwörtern in einem digitalen Dokument stehen zwei Ziele im Vordergrund:

1. das schnelle Auffinden der Fundstellen und
2. eine einfache Einschätzung, inwieweit ein Dokumentteil (Abschnitt, Kapitel, etc.) Ergebnisse zur gegebenen Suchanfrage enthält.

Zum Erreichen des zweiten Zieles ist eine Verknüpfung der Informationen über die Struktur eines Dokumentes mit den Suchergebnissen erforderlich. Da sich die meisten Dokumente hierarchisch strukturieren, wird eine Visualisierung vorgestellt, die strukturelle Informationen als Treemap darstellt. Die Suchergebnisse werden durch die Attribute der Treemap-Regionen kodiert. Treemaps wurden ausgewählt, da sie intuitiv erfaßbar sind und eine gute Platzausnutzung aufweisen. Außerdem können auch Focus+Context-Techniken angewandt werden, um unterschiedlich detaillierte Einsichten zu erhalten.

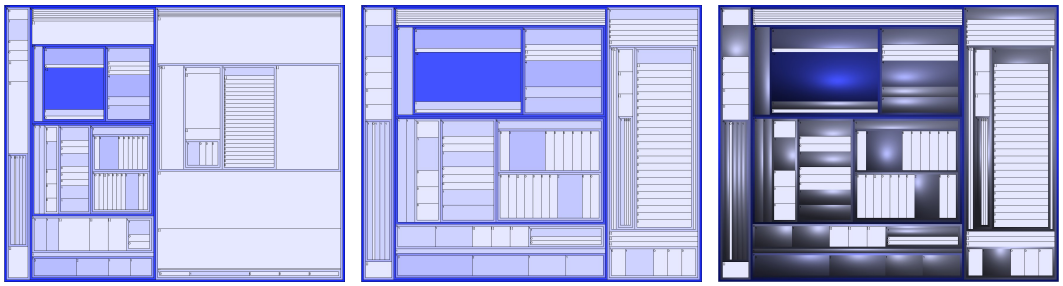
In Abbildung 4.4(a) ist die Struktur eines gegebenen Dokumentes als Treemap dargestellt. Für eine zusätzliche Kodierung der Fundstellen bietet sich an, die Anzahl der gefundenen Suchbegriffe pro Struktureinheit entweder über Farben oder über die Größe der Treemap-Region darzustellen. Abbildung 4.4(b) zeigt die Farb- bzw. Helligkeitskodierung. Es wird von einer Grundfarbe ausgegangen, deren Helligkeit anhand der Suchergebnisse angepaßt wird. Regionen mit mehr Fundstellen erscheinen dunkler. Dabei werden die Fundstellen in der Hierarchie aggregiert, so daß für alle Knoten in der Hierarchie die Gesamtergebnisse visualisiert werden. Ändert man die Metrik, die die Größe der Treemap-Regionen bestimmt, kann auch eine Größenkodierung erfolgen, wie sie in Abbildung 4.4(c) dargestellt ist. Durch diese Kombination ist eine schnelle und zielgerichtete Lokalisation der besten Fundstellen auf niedrigeren Ebenen der Struktur hauptsächlich durch die Größe der Regionen und auf höheren Ebenen hauptsächlich durch die Färbung möglich.



(a) Struktur des Dokuments (b) Suchergebnisse durch Farbe (c) Suchergebnisse durch Farbe und Größe

Abbildung 4.4. Prinzip der SearchTreemaps: Ausgehend von einer Visualisierung der Struktur des Dokumentes (a) können die Suchergebnisse auf verschiedene Arten in die Darstellung integriert werden.

Gerade bei großen Dokumenten kann die Treemap-Darstellung der Struktur schnell unübersichtlich werden. Hinzu kommt, daß typischerweise nur wenige Knoten der Hierarchie die Suchbegriffe auch enthalten. Hier werden Filtermechanismen eingesetzt, die die Anzeige von solchen Substrukturen in den Knoten unterdrückt, die keine Suchergebnisse enthalten und daher als weniger wichtig eingeschätzt werden (Abbildung 4.5(a)). Diese Filterung kann erweitert werden, indem zusätzlich die Größe dieser weniger wichtigen Regionen verringert wird (Abbildung 4.5(b)) oder indem andere Darstellungsarten – wie Cushion Treemaps – zur weiteren Hervorhebung der Fundstellen eingesetzt werden (Abbildung 4.5(c)).



(a) Ausblenden der inneren Struktur bei Knoten ohne Suchergebnisse (b) Anpassen der Größe bei ausgeblendeten Knoten (c) Hervorhebung durch Cushion Treemaps

Abbildung 4.5. Filtermechanismen zur Fokussierung auf Knoten mit Suchergebnissen.

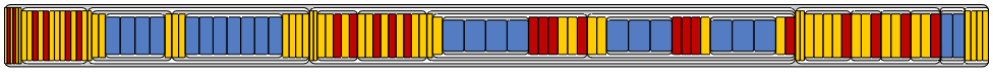
Der Einsatz solcher Visualisierungen ermöglicht ein schnelles Auffinden der Struktureinheiten eines Dokumentes, in denen die Suchbegriffe enthalten sind. Dies stellt einen Fortschrit gegenüber herkömmlichen Methoden dar, die Fundstellen nur farbig im Dokument hervorzuheben. Werden die Treemaps mit interaktiven Techniken kombiniert, die es erlauben, weitere Informationen über die entsprechenden Struktureinheiten einzuholen bzw. die eine direkte Navigation im Dokument ermöglichen, stellen sie ein effektives Werkzeug zur Arbeit mit digitalen Dokumenten dar.

### 4.2.3 *Relationen in Dokumenten*

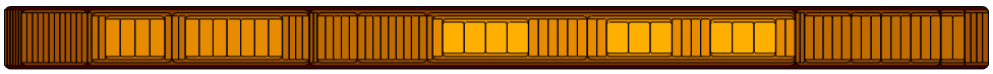
Neben der Struktur spielen in digitalen Dokumenten auch weitere Relationen eine wichtige Rolle, die entweder durch den Autor z. B. in Form von Verweisen eingebracht werden, implizit durch die thematische Struktur gegeben sind (z. B. benötigtes Vorwissen) oder aber durch den Leser beim Verarbeiten des Dokumentes eingebracht werden (in Form von Annotationen). Insbesondere für die Navigation innerhalb des Dokumentes sind solche Beziehungen wichtig. In [NSC05] wird mit *ArcTrees* eine interaktive Visualisierung vorgestellt, die strukturelle und relationale Informationen in digitalen Dokumenten verbindet und zur Navigation nutzt.

→ Anhang I
ArcTrees: Visualizing Relations in Hierarchical Data

Zur Strukturdarstellung wurde eine Visualisierung entwickelt, die sich an Treemaps anlehnt, aber wesentlich kompakter ist. Die Unterteilung in Regionen erfolgt hier immer vertikal, so daß alle Knoten in einer „Zeile“ angeordnet sind. Auch hier können weitere Informationen über Farbe kodiert werden. In Abbildung 4.6(a) ist der Typ der Dokumentstruktur (Textabschnitt, Bild, Liste, etc.) farblich kodiert, Abbildung 4.6(b) veranschaulicht die Hiererchieebenen durch unterschiedliche Helligkeiten. Da bei dieser Darstellung der Platzbedarf noch schneller zum begrenzenden Faktor wird, werden Interaktions- und Focus+Context-Techniken eingesetzt, um die Struktur interaktiv zu explorieren. Teilbäume können interaktiv ein- oder ausgeblendet werden, wie dies am Beispiel in Abbildung 4.6(c) zu sehen ist.



(a) Farbkodierung des Typs der Struktureinheit in einer Dokumentstruktur



(b) Helligkeitskodierung der Hierarchieebenen in einer Dokumentstruktur

(c) Darstellung der Dokumentstruktur mit ausgeblendeten Knoten (durch Schaltflächen visualisiert)

Abbildung 4.6. Verschiedene Inhaltskodierungen in der Strukturdarstellung für ArcTrees

Die Darstellung der Relationen zwischen den einzelnen Knoten erfolgt durch direkte Kantenverbindungen unter Beachtung verschiedener Faktoren. Die Anzahl der Relationen zwischen zwei Knoten wird dabei durch die Breite der Kanten dargestellt. Durch Transparenz wird die Sichtbarkeit der Verbindungen auch an Kreuzungspunkten gewährleistet. Relationen zwischen momentan nicht sichtbaren Knoten (Knoten auf einer tieferen Hierarchiestufe, die durch Interaktion ausgeblendet wurden) werden durch eine noch weiter erhöhte Transparenz gekennzeichnet (Abbildung 4.7).

Die Integration dieser beiden Darstellungen im Zusammenhang mit Focus+Context-Techniken und Metriken zur Visualisierung auch der an der Interaktion beteiligten Knoten ergibt ein mächtiges Werkzeug, um in digitalen Dokumenten anhand der

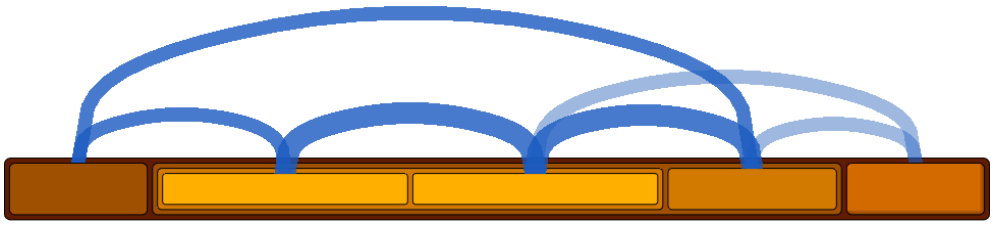


Abbildung 4.7. Darstellung der Relationen bei ArcTrees. Transparente Kanten verbinden nicht direkt sichtbare Knoten miteinander.

Struktur und der internen Relationen zu navigieren. Weitere Hilfsmittel wie Tooltips und Beschriftungen ermöglichen eine gezielte Informationssuche. Abbildung 4.8 zeigt ein komplexeres Beispiel eines Buches, in dem eine Themenauswahl insbesondere aufgrund der Relation „erfordert Vorwissen von“ erfolgt bzw. unterstützt wird.

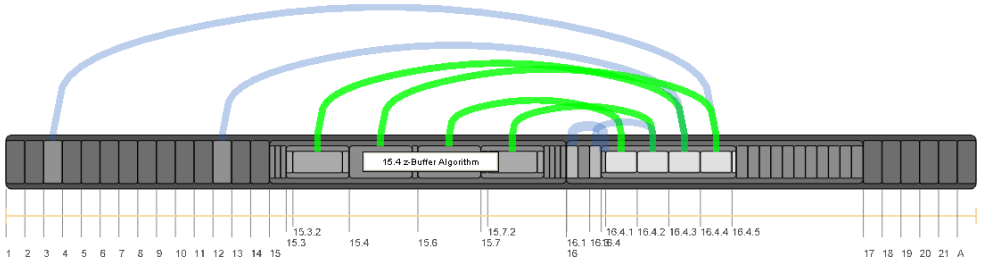



Abbildung 4.8. Anwendungsbeispiel für ArcTrees. Dargestellt ist die Struktur eines Buches. Die Bögen geben die Relation „erfordert Vorwissen von“ wieder. Kapitelnummern als Labels und Tooltips erleichtern die Navigation.

### 4.2.4 *Visualisierung von relationalen Beziehungen zwischen mehreren Dokumenten*

Wie bereits in Abschnitt 4.2.1 dargelegt wurde, bietet die Visualiserung von Beziehungen zwischen mehreren Dokumenten eine Möglichkeit, um den Zugriff auf große Literaturbestände zu vereinfachen und die Navigation in digitalen Bibliotheken zu erleichtern. In [SBG06] werden Visualisierungen und Interaktionstechniken vorgestellt, die verschiedene Relationen zur interaktiven Exploration größerer Literaturdatenbasen verwenden.

→ Anhang J
Interactively Exploring Bibliographical Data for Literature Analysis

Diese Daten werden analysiert und es werden drei Informationsmengen gebildet

- die Menge der Autoren $\mathcal{A}$,
- die Menge der Dokumente $\mathcal{D}$ und
- die Menge der Themenbereiche $\mathcal{T}$.

Zwischen diesen Informationsmengen bestehen vielfältige Relationen, die sich in zwei große Gruppen einteilen lassen:

- *interne Relationen*, die Elmente einer Informationmenge miteinander in Beziehung setzen. Hierzu gehören beispielsweise Ko-Autorenschaft $\mathcal{A} \rightarrow \mathcal{A}^n$ oder Zitierungen $\mathcal{D} \rightarrow \mathcal{D}^n$.
- *externe Relationen*, die Elemente zweier Informationmengen miteinander in Beziehung setzen. Hierzu gehört beispielsweise die Autorschaft $\mathcal{A} \rightarrow \mathcal{D}^n$ bzw. $\mathcal{D} \rightarrow \mathcal{A}^n$.

Beide Arten von Relationen werden in einem elliptischen Grundlayout visualisiert. Bei externen Relationen wird dabei eine Informationsmenge außen auf der Ellipse, die andere im Inneren der Ellipse dargestellt und die Relationen durch direkte Verbindungen angedeutet (siehe Abbildung 4.9(a)). Interne Relationen können mit einer zweifachen Darstellung der betreffenden Informationsmenge ebenso dargestellt werden, oder die Relationen werden direkt zwischen den Elementen auf der Ellipse visualisiert (Abbildung 4.9(b)).


(a) Relationen innerhalb einer Informationsmenge – hier Zitierungen zwischen Dokumenten

(b) Relationen zwischen zwei Informationsmengen – hier alle Dokumente (außen) eines Autors (innen).

Abbildung 4.9. Visualisierung von Beziehungen zwischen bibliographischen Daten.

Diese Visualisierung bildet die Grundlage für interaktive Techniken, die eine Navigation basierend auf den Relationen erlauben. Die Selektion eines Eintrags ermöglicht es, weitere Beziehungen zu diesem Eintrag auszuwählen und damit eine Filterung auf dem Datenbestand durchzuführen. Dies führt einerseits zu einer Neuanordnung der Datenelemente, andererseits auch zu Einschränkungen der angezeigten Menge an Daten. Relationenketten, die über mehrere Ebenen verlaufen, können ebenfalls durch eine Erweiterung der Darstellung in drei Dimensionen (durch „Stapeln“ mehrerer Ellipsen) visualisiert werden.

Im Vergleich zu bisherigen Formular-basierten Schnittstellen zu digitalen Bibliotheken erleichtert eine solche relationenbezogene Exploration die Suche nach weiteren Dokumenten. Suchanfragen können aus der momentan visualisierten Situation entwickelt und müssen nicht länger mental miteinander verknüpft werden.

## 4.3 ZUSAMMENFASSUNG

Die (Informations-)Visualisierung ist das klassische Anwendungsgebiet der Informationsdarstellung durch Bilder. Aufgrund der Vielzahl an möglichen verschiedenen Eingabedaten und der möglichen Techniken entsteht hier das gleiche Problem wie

bei nicht-photorealistischen Computergraphiken. Es lassen sich expressive graphische Darstellungen erzeugen; dies geht aber unter Umständen zu Lasten der Effektivität und der Angemessenheit. Aus diesem Grund stehen gerade in der Informationsvisualisierung immer mehr interaktive Methoden im Mittelpunkt der Forschung. Eine graphische Darstellung wird nicht mehr rein als Visualisierung der unterliegenden Daten verstanden, sondern als *Benutzungsschnittstelle* zur Navigation im Datenbestand bzw. für Operationen auf den Daten. Die in diesem Kapitel angeführten Beispiele haben dies gezeigt. Diese Beispiele sind davon ausgegangen, daß ein vollständiger und strukturierter Informationsraum vorliegt. Ist dies nicht der Fall, kann die Visualisierung auch genutzt werden, Daten zu ergänzen oder zu verändern. Einerseits dient die graphische Darstellung hier wie oben bereits genannt als Benutzungsschnittstelle zur Navigation, andererseits auch zur direkt-manipulativen Eingabe neuer Daten.

# 5 ZUSAMMENFASSUNG UND AUSBLICK

Die in dieser Arbeit angegebenen Beispiele haben gezeigt, daß die Informationsdarstellung mit Bidlern in vielen Anwendungsbereichen sehr effektiv eingesetzt werden kann. Die verwendeten Techniken aus den Bereichen der Computergraphik und der Visualisierung ermöglichen dabei die Erzeugung eines breiten Spektrums an Bildern. Für deren Einsatz sind allerdings die Prinzipen der Expressivität, Effektivität und Angemessenheit zu beachten, um im momentanen Dialogkontext die gewünschte Aussage durch das Bild zu vermitteln.

Die Entwicklung in der Computergraphik an sich ist in letzter Zeit sehr stark technologiegetrieben. Der technische Fortschritt bei der Graphikhardware und die damit verbundenen Möglichkeiten, graphische Operationen auf dafür entworfenenen Spezialprozessoren auszuführen, wirken sich insbesondere positiv auf die Rendering-Geschwindigkeit aus. Daher stehen mehr und mehr echtzeitfähige Techniken im Mittelpunkt der Forschung sowohl der Computergraphik als auch der Visualisierung. Damit einher geht eine noch stärkere Verbindung der Computergraphik und der Visualisierung. Der Einsatz von zunächst in der Computergraphik entwickelten Algorithmen und Techniken – insbesondere auch solcher aus dem Gebiet des nicht-photorealistischen Renderings – wird sich weiter verstärken. Ein Beispiel hierfür ist die Verwendung nicht-photorealistischer Techniken zur Hervorhebung in Bildern und einer damit verbundenen Lenkung der Auferksamkeit.

Andererseits nimmt die Komplexität der darzustellenden Modelle ständig zu. Ebenfalls technologisch bedingt können sehr detailgetreue Geometriemodelle in den unterschiedlichsten Verfahren relativ einfach hergestellt werden. Datenmodelle bzw. Datensätze, die zu visualisieren sind, nehmen an Umfang und Strukturierung zu. In vielen Fällen ist eine simple Darstellung der Geometrie bzw. der Daten nicht mehr möglich. Dem Adressaten der Visualisierung muß die Möglichkeit gegeben werden, die Darstellung und damit die Daten interaktiv zu explorieren. Getreu dem „Information Visualization Mantra“ von Shneiderman [Shn96]: *Overview first, zoom and filter, then details-on-demand*, ist Benutzerinteraktion gerade dann wichtig, wenn Details erkannt werden sollen.

Benutzerinteraktion ist allerdings nicht auf das Explorieren der Darstellung und das Extrahieren von Detailsichten im Bild beschränkt. Gerade bei einer Vielzahl verschiedener Techniken, mit denen ein Modell dargestellt werden kann, ist eine interaktive Auswahl der entsprechend anzuwendenden Technik sinnvoll. Halper et al. haben in [HSS02] dazu erste Schritte vorgestellt. Werden solche Systeme mit direktmanipulativen Methoden der Bilderzeugung gekoppelt, wie sie beispielsweise mit *WYSIWYG NPR* von Kalnins et al. [KMM+02] gegeben sind, können interessante Anwendungen entstehen, die es auch relativ computerunkundigen Benuztzern er-

möglichen, expressive Darstellungen auf einfache Weise zu erzeugen, ohne daß diese die Denkweise ihres Fachgebietes zu weit verlassen müssen.

Ein weiteres wichtiges Einsatzgebiet interaktiver Techniken ist die Manipulation der Datenmenge an sich. Dazu zählt, wie bereits in Abschnitt 4.3 angemerkt, sowohl die Dateneingabe als auch die Navigation im Datenraum. Hinzu kommt eine interaktive Strukturierung der Daten, die z. B. Bestandteil des Filtering-Schrittes in der Visualisierungs-Pipeline sein kann. Für das in dieser Arbeit auch betrachtete Anwendungsgebiet der digitalen Dokumente bringen interaktive, durch graphische Darstellungen unterstützte Techniken zur Anreicherung der Dokumente mit Beziehungen und Daten einen deutlichen Mehrwert. Erste Schritte auf diesem Gebiet wurden von Götze et al. insbesondere in drei Richtungen unternommen:

- die Eingabe zusätzlicher Informationen als Annotationen in digitalen Dokumenten durch Anwendung einer Papier-und-Bleistift-Metapher in [GSS02a, GSS02b],
- die zielgerichtete Anzeige dieser und weiterer Informationen durch spezielle Filtering-Techniken in [GS03] und
- das Ermöglichen von Diskussionen und gemeinschaftlicher Arbeit an Dokumenten auf Basis von WWW-Seiten in [GSS03].

Diese und ähnliche Techniken, verbunden mit den in Abschnitt 4.2 vorgestellten Visualisierungsmethoden können in Zukunft die Akzeptanz und den Einsatz digitaler Dokumente weiter erhöhen.

Schließlich werden auch neue Anwendungsgebiete insbesondere für die Informationsvisualisierung erschlossen. Die Entwicklung sogenanter „Social Software“ bzw. von „Social User Interfaces“ ist hier ein Beispiel. Grundlegender Gedanke hierbei ist es, die sozialen Beziehungen zwischen Benutzern abzubilden, in der Benutzungsschnittstelle von Anwendungen offenzulegen und damit eine noch stärkere Vernetzung zwischen den Benutzern zu ermöglichen. Die hier zu Grunde liegenden Datenmengen sind extrem umfangreich und die einzelnen Datenelemente (z. B. Benutzer) stark durch Relationen miteinander verbunden (siehe auch [MS04]). Die entstehenden Netzwerke bzw. Graphen können somit nicht mehr mit herkömmlichen Mitteln visualisiert bzw. statisch präsentiert werden. Momentane Trends im Hinblick auf die Nutzung des World Wide Web zeigen, daß solche Anwendungen in Zukunft Kernstück des täglichen Umgangs mit dem Computer sein werden.

[ADM+04] Adel Ahmed, Tim Dwyer, Colin Murray, Le Song und Ying Xin Wu: „WilmaScope Graph Visualisation“. In: *Proceedings of INFOVIS'04*. IEEE Computer Society, Washington, 2004. S. 216.4.

[Bad02] Ragnar Bade: *Methoden zur Visualisierung von und Interaktion in zeitbasierten Patientendaten und Behandlungsplänen*. Diplomarbeit, Institut für Simulation und Graphik der Otto-von-Guericke-Universität Magdeburg, 2002.

[BBT+06] Pascal Barla, Simon Breslav, Joëlle Thollot, François Sillion und Lee Markosian: „Stroke Pattern Analysis and Synthesis“. *Computer Graphics Forum (Proceedings of Eurographics 2006, Vienna, Austria, September 4–8, 2006)*, 25(3):(2006), S. 663–671.

[BDL05] Michael Balzer, Oliver Deussen und Claus Lewerentz: „Voronoi Treemaps for the Visualization of Software Metrics“. In: *Proceedings of the 2005 ACM Symposium on Software Visualization*. ACM, ACM Press, New York, 2005. S. 165–172.

[BE99] Fabien Benichou und Gershon Elber: „Output Sensitive Extraction of Silhouettes from Polygonal Geometry“. In: *Proceedings of the 7th Pacific Graphics Conference*. IEEE Computer Society Press, Los Alamitos, CA, 1999. S. 60–69.

[BHvW00] Mark Bruls, Kees Huizing und Jarke van Wijk: „Squarified Treemaps“. In: *Proceedings of the Joint Eurographics and IEEE TCVG Symposium on Visualization (TCVG 2000)*. IEEE Computer Society Press, Los Alamitos, 2000. S. 33–42.

[BKKL99] Anne Brüggemann-Klein, Rolf Klein und Britta Landgraf: „BibRelEx: Exploring Bibliographic Databases by Visualization of Annotated Contents-Based Relations“. *D-Lib Magazine*, 5(11).

[BS00] John W. Buchanan und Mario Costa Sousa: „The Edge Buffer: A Data Structure for Easy Silhouette Rendering“. In: *Proceedings of First International Symposium on Non Photorealistic Animation and Rendering (NPAR 2000, Annecy, France, June 5–7, 2000)*. ACM Press, New York, 2000. S. 39–42.

[BSM04] Ragnar Bade, Stefan Schlechtweg, und Silvia Miksch: „Connecting Time-Oriented Data and Information to a Coherent Interactive Visualization“. In: Elisabeth Dykstra-Erickson und Manfred Tscheligi (Hrsg.), *Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI04)*. ACM Press, New York, 2004. S. 105–112.

[Büd05] Stefan Büder: *Visualisierung von Zusammenhängen in Literaturdaten*. Diplomarbeit, Institut für Simulation und Graphik der Otto-von-Guericke-Universität Magdeburg, 2005.

[Che04] Chaomei Chen: „Searching for Intellectual Turning Points: Progressive Knowledge Domain Visualization". *Proceedings of the National Academy of Sciences of the United States of America (PNAS)*, 101:(2004), S. 5303–5310.

[Che06] Chaomei Chen: „CiteSpace II: Detecting and Visualizing Emerging Trends and Transient Patterns in Scientific Literature". *Journal of the American Society for Information Science and Technology*, 57(3):(2006), S. 359–377.

[CM02] Drew Card und Jason L. Mitchell: „Non-Photorealistic Rendering with Pixel and Vertex Shaders". In: Wolfgang Engel (Hrsg.), *Direct3D ShaderX: Vertex and Pixel Shader Programming Tips and Tricks*. Wordware, 2002. S. 319–333.

[DHvS00] Oliver Deussen, Stefan Hiller, Cornelius W. A. M van Overveld und Thomas Strothotte: „Floating Points: A Method for Computing Stipple Drawings". *Computer Graphics Forum (Proceedings of Eurographics 2000, Interlaken, Switzerland, Aug. 2000)*, 19(3):(2000), S. 40–51.

[Eic94] Stephen G. Eick: „Graphically Displaying Text". *Journal of Computational and Graphical Statistics*, 3(2):(1994), S. 127–142.

[ESS92] Stephen G. Eick, Joseph L. Steffen und Eric E. Sumner Jr.: „Seesoft—A Tool For Visualizing Line Oriented Software Statistics". *IEEE Transactions on Software Engineering*, 18(11):(1992), S. 957–968.

[FR98] Adam Finkelstein und Marisa Range: „Image Mosaics". In: Roger D. Hersch, Jacques André und Heather Brown (Hrsg.), *Electronic Publishing, Artistic Imaging and Digital Typography, Proceedings of the EP'98 and RIDT'98 Conferences*, Bd. 1375 von *Lecture Notes in Computer Science*. Springer-Verlag, Berlin · Heidelberg · New York, 1998. S. 11–22.

[FvDFH90] James D. Foley, Andries van Dam, Steven K. Feiner und John F. Hughes: *Computer Graphics Principles and Practice*. Addison-Wesley Publishing Company, Reading, 1990.

[FWDP03] Jean-Daniel Fekete, David Wang, Niem Dang und Catherine Plaisant: *Overlaying Graph Links on Treemaps*. Technischer Bericht, Human-Computer Interaction Lab / University of Maryland, 2003.

[Ger04] Tobias Germer: *RenderBots: Multiagentensysteme für NPR-Graphiken*. Diplom thesis at the Department of Simulation and Graphics, Otto-von-Guericke University Magdeburg, Germany, 2004.

[GG01] Bruce Gooch und Amy A. Gooch: *Non-Photorealistic Rendering*. A K Peters, Ltd., Natick, 2001.

[Gof82] Erving Goffman: *Das Individuum im öffentlichen Austausch*. Suhrkamp, Frankfurt am Main, 1982.

[GS03] Marcel Götze und Stefan Schlechtweg: „Magic Pages—Providing Added Value to Electronic Documents“. In: Constantine Stephanidis und Julie Jacko (Hrsg.), *Proceedings of HCI International 2003 (Crete, Greece, June 22-27, 2003)*, Bd. 2. Lawrence Erlbaum Associates, London, 2003. S. 651–655.

[GS05] Tobias Germer und Stefan Schlechtweg: „RenderBots—Multi-Agent Systems for NPR“. *NORSIGD Info, medlemsblad for NORSIGD*, 2005(1):(2005), S. 4–5.

[GSG+99] Bruce Gooch, Peter-Pike J. Sloan, Amy A. Gooch, Peter Shirley und Richard Riesenfeld: „Interactive Technical Illustration“. In: Stephen N. Spencer (Hrsg.), *Proceedings of the 1999 Symposium on Interactive 3D Graphics*. ACM Press, New York, 1999. S. 31–38.

[GSS02a] Marcel Götze, Stefan Schlechtweg und Thomas Strothotte: „The Intelligent Pen – Towards a Uniform Treatment of Electronic Documents“. In: Andreas Butz, Antonio Krüger, Patrick Olivier, Stefan Schlechtweg und Michelle Zhou (Hrsg.), *Proceedings of the 2nd International Symposium on Smart Graphics 2002*. ACM Press, New York, 2002. S. 129–135.

[GSS02b] Marcel Götze, Stefan Schlechtweg und Thomas Strothotte: „ViDio – Virtual Digital Annotations“. In: Michael Herczeg, Wolfgang Prinz und Horst Oberquelle (Hrsg.), *Mensch & Computer 2002: Vom interaktiven Werkzeug zu kooperativen Arbeits- und Lernwelten*. German Chapter of the Association for Computing Machinery (ACM), B.G. Teubner Verlag, Stuttgart, Leipzig, Wiesbaden, 2002. S. 195–204.

[GSS03] Marcel Götze, Stefan Schlechtweg und Thomas Strothotte: „Web-Seiten als Kommunikationsplattform zur Unterstützung des Online-Lesens“. In: Klaus P. Jantke, Wolfgang S. Wittig und Jörg Herrmann (Hrsg.), *Von e-Learning bis e-Payment 2003. Tagungsband LIT'03, Leipziger Informatiktage*. Akademische Verlagsgesellschaft Aka GmbH, Berlin, 2003. S. 67–75.

[Hae90] Paul Haeberli: „Paint By Numbers: Abstract Image Representations“. *ACM SIGGRAPH Computer Graphics, Proceedings of ACM SIGGRAPH 90 (Dallas, TX, August 6–10, 1990)*, 24(3):(1990), S. 207–214.

[Hau01] Alejo Hausner: „Simulating Decorative Mosaics“. In: Eugene Fiume (Hrsg.), *Proceedings of ACM SIGGRAPH 2001 (Los Angeles, CA, August 12–17, 2001)*, Computer Graphics Proceedings, Annual Conference Series. ACM SIGGRAPH, New York, 2001. S. 573–580.

[Hea95] Marti A. Hearst: „TileBars: Visualization of Term Distribution Information in Full Text Information Access“. In: *Proceedings of CHI'95 Confe-*

*rence on Human Factors in Computing Systems*. ACM SIGCHI, New York, 1995. S. 59–66.

[Her98] Aaron Hertzmann: „Painterly Rendering with Curved Brush Strokes of Multiple Sizes“. In: Michael Cohen (Hrsg.), *Proceedings of ACM SIGGRAPH 98 (Orlando, FL, July 19–24, 1998)*, Computer Graphics Proceedings, Annual Conference Series. ACM Press/ACM SIGGRAPH, New York, 1998. S. 453–460.

[Her01] Aaron Hertzmann: „Paint By Relaxation“. In: Horace H. S. Ip, Nadia Magnenat-Thalmann und Tat-Seng Chua Rynson W. H. Lau (Hrsg.), *Proceedings of Computer Graphics International 2001 (Hong Kong, Jul 2001)*. IEEE Computer Society Press, Los Alamitos, 2001. S. 47–54.

[Her03] Aaron Hertzmann: „A Survey of Stroke-Based Rendering“. *IEEE Computer Graphics and Applications*, 23(4):(2003), S. 70–81.

[HHD03] Stefan Hiller, Heino Hellwig und Oliver Deussen: „Beyond Stippling – Methods for Distributing Objects on the Plane“. *Computer Graphics Forum (Proceedings of Eurographics 2003, Granada, Spain, September 1–6, 2003)*, 22(3):(2003), S. 515–522.

[HIMS04] Tzu-Wei Hsu, Lee Inman, Dave McColgin und Kevin Stamper: „MonkEllipse: Visualizing the History of Information Visualization“. In: *Proceedings of INFOVIS'04*. IEEE Computer Society, Washington, 2004. S. 216.9.

[HIR+03] Nick Halper, Tobias Isenberg, Felix Ritter, Bert Freudenberg, Oscar E. Meruvia Pastor, Stefan Schlechtweg und Thomas Strothotte: „OpenNPAR: A System for Developing, Programming, and Designing Non-Photorealistic Animation and Rendering“. In: Jon Rokne, Reinhard Klein und Wenping Wang (Hrsg.), *Proceedings of Pacific Graphics 2003*. IEEE Computer Society, IEEE, Los Alamitos, CA, 2003. S. 424–428.

[Hof06] Marc Hofmann: *Interaktion und Visualisierung zur Kommunikation über wissenschaftliche Literatur*. Diplomarbeit, Institut für Simulation und Graphik der Otto-von-Guericke-Universität Magdeburg, 2006.

[HSS02] Nick Halper, Stefan Schlechtweg und Thomas Strothotte: „Creating Non-Photorealistic Images the Designer's Way“. In: *Proceedings of Second International Symposium on Non Photorealistic Animation and Rendering (NPAR 2002, Annecy, France, June 3–5, 2002)*. ACM Press, New York, 2002. S. 97–104.

[HZ00] Aaron Hertzmann und Denis Zorin: „Illustrating Smooth Surfaces“. In: Kurt Akeley (Hrsg.), *Proceedings of ACM SIGGRAPH 2000 (New Orleans, LA, July 23–28, 2000)*, Computer Graphics Proceedings, Annual Conference Series. ACM SIGGRAPH, New York, 2000. S. 517–526.

[IBSC05] Tobias Isenberg, Angela Brennecke, Mario Costa Sousa und Sheelagh

Carpendale: *Beyond Pixels: Illustration with Vector Graphics*. Technischer Bericht 2005-804-35, Department of Computer Science, University of Calgary, Canada, 2005.

[IFH+03] Tobias Isenberg, Bert Freudenberg, Nick Halper, Stefan Schlechtweg und Thomas Strothotte: „A Developer's Guide to Silhouette Algorithms for Polygonal Models". *IEEE Computer Graphics and Applications*, 23(4):(2003), S. 28–37.

[JL97] Bruno Jobard und Wilfrid Lefer: „Creating Evenly-Spaced Streamlines of Arbitrary Density". In: *Proceedings of the 8th Eurographics Workshop on Visualization in Scientific Computing*. 1997. S. 45–55.

[JS98] Dean F. Jerding und John T. Stasko: „The Information Mural: A Technique for Displaying and Navigating Large Information Spaces". *IEEE Transactions on Visualization and Computer Graphics*, 4(3):(1998), S. 257–271.

[Ker03] Bernard Kerr: „THREAD ARCS: An Email Thread Visualization". In: *Proceedinsg of the IEEE Symposium on Information Visualization 2003 (InfoVis'03)*. IEEE Press, Los Alamitos, CA, USA, 2003. S. 211–218.

[KLR+04] Stefan Klink, Michael Ley, Emma Rabbidge, Patrick Reuther, Bernd Walter und Alexander Weber: „Browsing and Visualizing Digital Bibliographic Data". In: Oliver Deussen, Charles Hansen, Daniel A. Keim und Dietmar Saupe (Hrsg.), *Data Visualization. Proceedings of the 2004 Eurographics/IEEE TVCG Workshop on Visualization*. Eurographics Association, 2004. S. 237–242.

[KMM+02] Robert D. Kalnins, Lee Markosian, Barbara J. Meier, Michael A. Kowalski, Joseph C. Lee, Philip L. Davidson, Matthew Webb, John F. Hughes und Adam Finkelstein: „WYSIWYG NPR: Drawing Strokes Directly on 3D Models". *ACM Transactions on Graphics, Proceedings of ACM SIGGRAPH 2002 (San Antonio, TX, July 21–26, 2002)*, 21(3):(2002), S. 755–762.

[KS99] Matthias Kreuseler und Heidrun Schumann: „Information visualization using a new Focus+Context Technique in combination with dynamic clustering of information space". In: *Proceedings of NPIV'99 (New Paradigms in Information Visualization and Manipulation)*. 1999. S. 1–5.

[Lit97] Peter Litwinowicz: „Processing Images and Video for an Impressionist Effect". In: Turner Whitted (Hrsg.), *Proceedings of ACM SIGGRAPH 97 (Los Angeles, CA, August 3–8, 1997)*, Computer Graphics Proceedings, Annual Conference Series. ACM Press/ACM SIGGRAPH, New York, 1997. S. 407–414.

[Lov02] Jörn Loviscach: „Rendering Artistic Line Drawings Using Off-the-Shelf 3-D Software". In: Isabel Navazo Alvaro und Philipp Slusallek

(Hrsg.), *Proceedings of Eurographics 2002, Short Presentations*. Eurographics Association, Blackwell Publishers, Oxford, UK, 2002. S. 125–130.

[LR94] John Lamping und Ramana Rao: „Laying Out and Visualizing Large Trees Using a Hyperbolic Space". In: *ACM Symposium on User Interface Software and Technology*. 1994. S. 13–14.

[LRP95] John Lamping, Ramana Rao und Peter Pirolli: „A Focus+Context Technique Based on Hyperbolic Geometry for Visualizing Large Hierarchies". In: *Proc. ACM Conf. Human Factors in Computing Systems, CHI*. 1995. S. 401–408.

[Mac86] Jock Mackinlay: „Automating the Design of Graphical Presentations of Relational Information". *ACM Transactions on Graphics*, 5(2):(1986), S. 110–141.

[MBC02] Jason L. Mitchell, Chris Brennan und Drew Card: „Real-Time Image-Space Outlining for Non-Photorealistic Rendering". In: *ACM SIGGRAPH 2002 Conference Abstracts and Applications*. ACM SIGGRAPH, New York, 2002. S. 239.

[Mei96] Barbara J. Meier: „Painterly Rendering for Animation". In: Holly Rushmeier (Hrsg.), *Proceedings of ACM SIGGRAPH 96 (New Orleans, LA, August 4–9, 1996)*, Computer Graphics Proceedings, Annual Conference Series. ACM Press/ACM SIGGRAPH, New York, 1996. S. 477–484.

[MPFS03] Oscar E. Meruvia Pastor, Bert Freudenberg und Thomas Strothotte: „Real-Time, Animated Stippling". *IEEE Computer Graphics and Applications*, 23(4):(2003), S. 62–68.

[MRC91] Jock D. Mackinlay, George G. Robertson und Stuart K. Card: „The Perspective Wall: Detail and Context Smoothly Integrated". In: *Proceedings of the 1991 SIGCHI conference on Human factors in computing systems: Reaching through technology*. ACM Press, New York, 1991. S. 173–176.

[MS04] Winfried Marotzki und Stefan Schlechtweg: „Identitätspräsentationen in virtuellen Communities". In: Hans-Uwe Otto und Nadia Kutscher (Hrsg.), *Informelle Bildung Online*. Juventa Verlag, Weinheim, 2004. S. 41–53.

[Mun97] Tamara Munzner: „H3: Laying Out Large Directed Graphs in 3D Hyperbolic Space". In: *Proceedings of the 1997 IEEE Symposium on Information Visualization (Phoenix, Oct 1997)*. 1997. S. 2–10.

[Neu04] Petra Neumann: *Focus+Context Visualisation of Relations in Hierarchical Data.* Diplomarbeit, Institut für Simulation und Graphik der Otto-von-Guericke-Universität Magdeburg, 2004.

[Nie77] Gerhard Niemeyer: *Kybernetische System- und Modelltheorie, System Dynamics*. Franz Vahlen, München, 1977.

[NSC05] Petra Neumann, Stefan Schlechtweg und Sheelagh Carpendale: „ArcTrees: Visualizing Relations in Hierarchical Data“. In: Ken W. Brodlie, David J. Duke und Ken I. Joy (Hrsg.), *Data Visualization 2005, Eurographics/IEEE VGTC Symposium on Visualization Symposium Proceedings*. Eurographics Association, Aire-la-Ville, 2005. S. 53–61.

[NSS05] Thomas Nocke, Stefan Schlechtweg und Heidrun Schumann: „Icon-Based Visualization Using Mosaic Metaphors“. In: *Proceedings of the Ninth International Conference on Information Visualisation (IV'05)*. IEEE Computer Society Press, Los Alamitos, CA, 2005. S. 103–109.

[PMR+96] Catherine Plaisant, Brett Milash, Anne Rose, Seth Widoff und Ben Shneiderman: „LifeLines: Visualizing Personal Histories“. In: Michael J. Tauber (Hrsg.), *Proceedings of the SIGCHI conference on Human factors in computing systems: common ground*. ACM Press, New York, 1996. S. 221–227.

[RC99] Ramesh Raskar und Michael Cohen: „Image Precision Silhouette Edges“. In: Stephen N. Spencer (Hrsg.), *Proceedings of the Conference on the 1999 Symposium on interactive 3D Graphics*. ACM Press, New York, 1999. S. 135–140.

[Rey87] Craig W. Reynolds: „Flocks, Herds, and Schools: A Distributed Behavioral Model“. *Computer Graphics (Proceedings of ACM SIGGRAPH 83)*, 21(4):(1987), S. 25–34.

[RMC91] George G. Robertson, Jock D. Mackinlay und Stuart K. Card: „Cone Trees: Animated 3D Visualizations of Hierarchical Information“. In: Scott P. Robertson, Gary M. Olson und Judith S. Olson (Hrsg.), *Proceedings of the ACM SIGCHI conference on Human Factors in Computing Systems '91*. ACM Press, New York, 1991. S. 89–194.

[Rus89] Pramod Rustagi: „Silhouette Line Display From Shaded Models“. *IRIS Universe*, 1989(9):(1989), S. 42–44.

[RvE92] Jarek R. Rossignac und Maarten van Emmerik: „Hidden Contours on a Frame-Buffer“. In: *Proceedings of the 7th Eurographics Workshop on Computer Graphics Hardware*. 1992. S. 188–204.

[S+98] Thomas Strothotte et al.: *Computational Visualization: Graphics, Abstraction, and Interactivity*. Springer Verlag, Berlin · Heidelberg · New York, 1998.

[SABS94] Michael P. Salisbury, Sean E. Anderson, Ronen Barzel und David H. Salesin: „Interactive Pen-and-Ink Illustration“. In: Andrew Glassner (Hrsg.), *Proceedings of ACM SIGGRAPH 94 (Orlando, FL, July 24–29, 1994)*, Computer Graphics Proceedings, Annual Conference Series. New York, 1994. S. 101–108.

[SALS96] Michael P. Salisbury, Corin Anderson, Dani Lischinski und David H.

Salesin: „Scale-Dependent Reproduction of Pen-and-Ink Illustration“. In: Holly Rushmeier (Hrsg.), *Proceedings of ACM SIGGRAPH 96 (New Orleans, LA, August 4–9, 1996)*, Computer Graphics Proceedings, Annual Conference Series. ACM Press/ACM SIGGRAPH, New York, 1996. S. 461–468.

[SBG06] Stefan Schlechtweg, Stefan Büder und Marcel Götze: „Interactively Exploring Bibliographical Data for Literature Analysis“. In: Andreas M. Heinecke und Hansjürgen Paul (Hrsg.), *Mensch & Computer 2006: Mensch und Computer im Strukturwandel*. Oldenbourg Verlag, München, 2006. S. 293–302.

[Sch05a] Stefan Schlechtweg: „Computergrafik“. In: Kai Bruns und Klaus Meyer-Wegener (Hrsg.), *Taschenbuch der Medieninformatik*. Fachbuchverlag, Leipzig, 2005. S. 220–240.

[Sch05b] Stefan Schlechtweg: „Stile in der Computergraphik, oder: Können Rechner malen?“ In: Klaus Sachs-Hombach (Hrsg.), *Bildwissenschaft zwischen Reflexion und Anwendung*. Herbert von Halem Verlag, Köln, 2005. S. 546–560.

[Sec02] Adrian Secord: „Weighted Voronoi Stippling“. In: *Proceedings of Second International Symposium on Non Photorealistic Animation and Rendering (NPAR 2002, Annecy, France, June 3–5, 2002)*. ACM Press, New York, 2002. S. 37–44.

[SGS05] Stefan Schlechtweg, Tobias Germer und Thomas Strothotte: „RenderBots—Multi Agent Systems for Direct Image Generation“. *Computer Graphics Forum*, 24(2):(2005), S. 137–148.

[SH05] Peter Stahlknecht und Ulrich Hasenkamp: *Einführung in die Wirtschaftsinformatik*. Springer-Verlag, Berlin · Heidelberg · New York, 11. Aufl., 2005.

[Shn92] Ben Shneiderman: „Tree visualization with treemaps: a 2-d space-filling approach“. *ACM Transactions on Graphics*, 11(1):(1992), S. 92–99.

[Shn96] Ben Shneiderman: „The Eyes Have it: A Task by Data Type Taxonomy for Information Visualization“. In: *Proc. VL'96 Symposium on Visual Languages*. IEEE Press, 1996. S. 336–343.

[SM00] Heidrun Schumann und Wolfgang Müller: *Visualisierung. Grundlagen und allgemeine Methoden*. Springer-Verlag, Berlin · Heidelberg · New York, 2000.

[SS97] Christine Strothotte und Thomas Strothotte: *Seeing Between the Pixels*. Springer-Verlag, Berlin · Heidelberg · New York, 1997.

[SS02] Thomas Strothotte und Stefan Schlechtweg: *Non-Photorealistic Computer Graphics. Modeling, Animation, and Rendering*. Morgan Kaufmann Publishers, San Francisco, 2002.

[SSWS04a] Stefan Schlechtweg, Petra Schulze-Wollgast und Heidrun Schumann: „Interactive Treemaps With Detail on Demand to Support Information Search in Documents“. In: Oliver Deussen, Charles Hansen, Daniel A. Keim und Dietmar Saupe (Hrsg.), *Data Visualization 2004. Eurographics/IEEE TCVG Visualization Symposium Proceedings*. Eurographics Association, Aire-la-Ville, 2004. S. 121–128.

[SSWS04b] Stefan Schlechtweg, Petra Schulze-Wollgast und Heidrun Schumann: „Visual Support for Keyword Search in Electronic Documents“. In: Thomas Schulze, Stefan Schlechtweg und Volkmar Hinz (Hrsg.), *Proceedings Simulation und Visualisierung 2004*. SCS Europe, Erlangen · San Diego, 2004. S. 215–225.

[ST90] Takafumi Saito und Tokiichiro Takahashi: „Comprehensible Rendering of 3-D Shapes“. *ACM SIGGRAPH Computer Graphics, Proceedings of ACM SIGGRAPH 90 (Dallas, TX, August 6–10, 1990)*, 24(3):(1990), S. 197–206.

[SW69] Claude E. Shannon und Warren Weaver: *The mathematical theory of communication*. University of Illinois Press, Urbana and Chicago, 1969.

[SWHS97] Michael P. Salisbury, Michael T. Wong, John F. Hughes und David H. Salesin: „Orientable Textures for Image-Based Pen-and-Ink Illustration“. In: Turner Whitted (Hrsg.), *Proceedings of ACM SIGGRAPH 97 (Los Angeles, CA, August 3–8, 1997)*, Computer Graphics Proceedings, Annual Conference Series. ACM Press/ACM SIGGRAPH, New York, 1997. S. 401–406.

[vWvdW99] Jarke J. van Wijk und Huub van de Wetering: „Cushion Treemaps: Visualization of Hierarchical Information“. In: *Proceedings of the IEEE Symposium on Information Visualization 1999 (INFOVIS'99)*. 1999. S. 73–78.

[Wat02a] Alan Watt: *3D-Computergrafik*. Pearson Education, München, 2002.

[Wat02b] Martin Wattenberg: „Arc Diagrams: Visualizing Structure in Strings“. In: *Proceedinsg of the IEEE Symposium on Information Visualization 2002 (InfoVis'02)*. IEEE Press, Los Alamitos, CA, USA, 2002. S. 110–116.

[WHP+04] Pak Chung Wong, Beth Hetzler, Christian Posse, Mark Whiting, Susan Havre, Nick Cramer, Anuj Shah, Mudita Singhal, Alan Turner und Jim Thomas: „IN-SPIRE InfoVis 2004 Contest Entry“. In: *Proceedings of INFOVIS'04*. IEEE Computer Society, Washington, 2004. S. 216.2.

[WP98] Alan Watt und Fabio Policarpo: *The Computer Image*. Addison Wesley Longman Ltd., Harlow, 1998.

[WS94] Georges A. Winkenbach und David H. Salesin: „Computer-Generated Pen-and-Ink Illustration“. In: Andrew Glassner (Hrsg.), *Proceedings of*

*ACM SIGGRAPH 94 (Orlando, FL, July 24–29, 1994)*, Computer Graphics Proceedings, Annual Conference Series. New York, 1994. S. 91–100.

[WS96] Georges A. Winkenbach und David H. Salesin: „Rendering Parametric Surfaces in Pen and Ink“. In: Holly Rushmeier (Hrsg.), *Proceedings of ACM SIGGRAPH 96 (New Orleans, LA, August 4–9, 1996)*, Computer Graphics Proceedings, Annual Conference Series. ACM Press/ACM SIGGRAPH, New York, 1996. S. 469–476.

[WTP+95] James A. Wise, James J. Thomas, Kelly Pennock, David Lantrip, Marc Pottier, Anne Schur und Vern Crow: „Visualizing the Non-Visual: Spatial Analysis and Interaction with Information for Text Documents“. In: *Proceedings of InfoVIS'95*. IEEE Press, 1995. S. 51–58.

[ZG99] Philip G. Zimbardo und Richard J. Gerrig: *Psychologie*. Springer-Verlag, Berlin · Heidelberg · New York, 7. Aufl., 1999.

[ZISS04a] Johannes Zander, Tobias Isenberg, Stefan Schlechtweg und Thomas Strothotte: *Creating High Quality Hatching Illustrations*. Technischer Bericht 12/2004, Fakultät für Informatik der Otto-von-Guericke-Universität Magdeburg, Germany, 2004.

[ZISS04b] Johannes Zander, Tobias Isenberg, Stefan Schlechtweg und Thomas Strothotte: „High Quality Hatching“. *Computer Graphics Forum (Proceedings of Eurographics 2004, Grenoble, France, August 30–September 3, 2004)*, 23(3):(2004), S. 421–430.

# A

## NON-PHOTOREALISTIC COMPUTER GRAPHICS. MODELLING, RENDERING, AND ANIMATION

Bibliographische Angaben:

Thomas Strothotte und Stefan Schlechtweg: *Non-Photorealistic Computer Graphics. Modeling, Animation, and Rendering*. Morgan Kaufmann Publishers, San Francisco, 2002.

# B

## STILE IN DER COMPUTERGRAPHIK, ODER: KÖNNEN RECHNER MALEN?

Bibliographische Angaben:

Stefan Schlechtweg: „Stile in der Computergraphik, oder: Können Rechner malen?“ In: Klaus Sachs-Hombach (Hrsg.), *Bildwissenschaft zwischen Reflexion und Anwendung*. Herbert von Halem Verlag, Köln, 2005. S. 546–560.

Abstract:

Von Anfang an war es das Ziel der Computergraphik, so realistische Bilder wie möglich zu erzeugen. Mit dem Bereitstehen der entsprechenden technischen Voraussetzungen entwickelte sich daher das Gebiet des Photorealismus in der Computergraphik, das dieses Ziel bereits im Namen trägt. Bis heute ist ein Grad an Realismus erreicht, des es schwer macht, eine computergenerierte Graphik von einer Photographie zu unterscheiden. Dieser Beitrag stellt dar, wie andere, eben nicht-photorealistische Graphiken, erzeugt werden können und in wie weit hier – ähnlich wie bei photorealistischen Graphiken – automatische Methoden angewendet werden können.

Im folgenden wird zunächst ein kurzer Überblick über photorealistische Bilderzeugungsverfahren gegeben, der auch aufzeigen soll, daß neue Zielstellungen in der Computergraphik durchaus benötigt werden, um computergenerierte Graphiken in noch größerer Breite einsetzen zu können. Diese neuen Zielstellungen – nämlich Bilder zu generieren, die nicht einer Photographie ähneln – führten zu einem neuen Zweig der Computergraphik, dem nicht-photorealistischen Rendering. Dieses Teilgebiet wird im Anschluß vorgestellt und grundlegende Techniken, die zum Erzeugen von unterschiedlichen Bildern genutzt werden, aufgezeigt. Im letzten Teil des Beitrages steht die Benutzerinteraktion im Vordergrund, da die neuen Rendering-Verfahren eine viel stärkere Einbeziehung des Benutzers und neuartige Konzepte für Benutzungsschnittstellen erfordern.

Stefan Schlechtweg
Institut für Simulation und Graphik
Otto-von-Guericke-Universität Magdeburg
Universitätsplatz 2, 39106 Magdeburg
eMail: stefans@isg.cs.uni-magdeburg.de

# Stile in der Computergraphik, oder: Können Rechner malen?

Von Anfang an war es das Ziel der Computergraphik, so realistische Bilder wie möglich zu erzeugen. Mit dem Bereitstehen der entsprechenden technischen Voraussetzungen entwickelte sich daher das Gebiet des *Photorealismus* in der Computergraphik, das dieses Ziel bereits im Namen trägt. Bis heute ist ein Grad an Realismus erreicht, des es schwer macht, eine computergenerierte Graphik von einer Photographie zu unterscheiden. Dieser Beitrag stellt dar, wie andere, eben *nicht-photorealistische Graphiken* erzeugt werden können und in wie weit hier – ähnlich wie bei photorealistischen Graphiken automatische Methoden angewendet werden können.

Im folgenden wird zunächst ein kurzer Überblick über photorealistische Bilderzeugungsverfahren gegeben, der auch aufzeigen soll, daß neue Zielstellungen in der Computergraphik durchaus benötigt werden, um computergenerierte Graphiken in noch größerer Breite einsetzen zu können. Diese neuen Zielstellungen – nämlich Bilder zu generieren, die nicht einer Photographie ähneln – führten zu einem neuen Zweig der Computergraphik, dem nicht-photorealistischen Rendering. Dieses Teilgebiet wird im Anschluß vorgestellt und grundlegende Techniken, die zum Erzeugen von unterschiedlichen Bildern genutzt werden, aufgezeigt. Im letzten Teil des Beitrages steht die Benutzerinteraktion im Vordergrund, da die neuen Rendering-Verfahren eine viel stärkere Einbeziehung des Benutzers und neuartige Konzepte für Benutzungsschnittstellen erfordern.

## Neue Ziele in der Computergraphik

Die Computergraphik beschäftigt sich mit der Synthese von Bildern. Hier geht es im Besonderen um die Synthese von Bildern aus dreidimensionalen Modellen, also um die Abbildung von Gegenständen. Schon früh wurde das Ziel spezifiziert, Bilder erzeugen zu können, die von der Realität – oder genauer: von einer Photographie der Realität – nicht zu unterscheiden sind. Die Forschungsrichtung in der Computergraphik wurde Photorealismus genannt in Anlehnung an einen Kunststil, der insbesondere in Nordamerika in den 1960er Jahren sehr beliebt war (Meisel 1989). Lange Zeit wurden in verschiedenen Richtungen Techniken und Algorithmen entwickelt, die versuchen,

(a) den Vorgang der Aufnahme eines photographischen Bildes zu imitieren,
(b) die Physik bei der Bildentstehung zu nutzen bzw. nachzubilden und/oder
(c) psychologische und physiologische Aspekte der menschlichen Wahrnehmung in die Bildgenerierung einfließen zu lassen.

Diese drei Aspekte führten zu Software, die heute allgemein verfügbar ist und weithin verwendet wird. Als Beispiele sei hier die Filmindustrie genannt, die heute fast selbstverständlich Computergraphik einsetzt, um einerseits Spezialeffekte zu erzeugen (Explosionen, etc.), die real nur schwer zu filmen wären, die aber andererseits auch ganze computergenerierte Filme produziert (wie z.B. Final Fantasy).

Eine der Möglichkeiten bzw. ein Teilaspekt, algorithmisch photorealistische Bilder zu erzeugen besteht darin, die Arbeitsweise einer photographischen Kamera nachzubilden. Die wichtigsten Bestandteile einer Kamera, die hier betrachtet werden müssen sind das Objektiv und der Film. An die Stelle des Filmes tritt die Bildebene, in der eine Bitmap das entstehende Bild pixelweise speichert. Das einfachste Modell eines Objektives findet sich in einer Lochkamera, die Berechnung des Strahlengangs reduziert sich dann auf die Anwendung des Strahlensatzes (vgl. Foley 1990). Natürlich können auch komplexe, aus mehreren Linsen zusammengesetzte Objektive nachgebildet werden, was die Berechnungen entsprechend verkompliziert (Kolb 1995).
Die Nachbildung von Objektiv und Film ermöglicht die Bestimmung der Orte auf der Bildebene, an denen die Originalpunkte der Modelle abgebildet werden. Die im Bild zu setzende Farbe wird durch ein Beleuchtungsmodell bestimmt. Hier finden meistens lokale Modelle Anwendung, die die Helligkeit an einem Punkt im Modell lokal aus der Lage des Punktes zu den Lichtquellen und dem Betrachterstandpunkt berechnen. Eines der bekanntesten Modelle hierzu wurde bereits 1975 von Phong entwickelt und ist heute Standard in der photorealistischen Computergraphik (Phong 1975). Diese Approximation des realen – globalen – Ablaufs der Beleuchtung liefert zufriedenstellende Ergebnisse. Trotzdem führt eine weitergehende Einbeziehung der physikalischen Grundlagen zu Verbesserungen.
Die Beleuchtung eines Punktes ist ein globales Phänomen, d.h. nicht nur abhängig von der Lage des Punktes zur Lichtquelle. Vielmehr spielen Reflexion des Lichtes in der Umgebung eine Rolle, Brechungs- und Beugungsvorgänge lassen spezielle Effekte entstehen. Solche Simulationen der globalen Vorgänge beruhen auf einer Gesetzmäßigkeit, die Kajia in der Rendering-Gleichung formulierte (Kajia 1986). Demnach setzt sich das von einem Punkt ausgestrahlte Licht zusammen aus dem Licht, das der Punkt selbst emittiert und dem an der Stelle reflektierten Licht. Verschiedene Arten und Weisen der Auswertung dieser Rendering-Gleichung führen zu verschiedenen Simulationsverfahren. Die wichtigsten beiden Verfahren hier sind

(a) Raytracing (Strahlenverfolgung), bei dem Lichtstrahlen auf ihrem Weg von der Lichtquelle zum Betrachter verfolgt werden und der Anteil eines jeden Lichtstrahls an der Helligkeit des Bildes bestimmt wird und
(b) Radiosity, bei dem Licht als Energie betrachtet und ein Energiegleichgewicht in der Szene berechnet wird. Aus der Energieverteilung kann dann auf die Helligkeit bzw. die Beleuchtung geschlossen werden.

Diese Art der Nachbildung von Kamera und Beleuchtungsvorgängen wird gemeinhin als *Rendering* bezeichnet und führt zu realistischen Darstellungen die heute wirklich kaum noch von Photographien zu unterscheiden sind, wie verschiedene Versuche gezeigt haben.
Trotzdem bleiben viele Fragen offen. Ein Ziel scheint erreicht, es wurde ein Verfahren entwickelt, um Photographien nachzubilden, was ist aber mit all den anderen bildlichen Darstellungen, die sich im Laufe der Geschichte entwickelt haben und die ihre ganz konkreten Anwendungsfelder haben. Interessanterweise in einem Lehrbuch zur photorealistischen Bilderzeugung (Foley 1990) findet man folgendes Zitat: *„If the ultimate goal of a picture is to convey information, then a picture that is free of all the complications of shadows and reflections may well be more successful than a tour de force of photographic realism.“* Demnach sind Photographien und photorealistische Computergraphiken weniger geeignet wenn es darum geht, Informationen zu vermitteln. Das ist verständlich, denn der absolute, perfekte Blick, der durch solche Bilder vermittelt wird, regt weniger zum Nachdenken an, fokussiert weniger und stellt alle Aspekte der Szene gleich detailliert dar (Strothotte/Strothotte 1997; Schumann 1996). Das heißt andererseits aber auch, daß in der Computergraphik neue Ziele gesteckt wurden. Diese betreffen die Erzeugung illustrativer Graphiken, die zur Vermittlung von Ideen, Konzepten und Informationen eingesetzt werden können, die weit über rein geometrische Angaben (beispielsweise in einem 3D-Modell)

hinausgehen. Es begann sich das Teilgebiet des *nicht-photorealistischen Renderings (NPR)* herauszubilden, in dem Techniken und Algorithmen entstehen, die genau solche alternativen Darstellungen zum Photorealismus zum Ziel haben.

# Nicht-photorealistisches Rendering

Seit dem Beginn der 90er Jahre entwickelte sich NPR als eigenständiges Gebiet der Computergraphik (vgl. Strothotte/Schlechtweg 2002). Die entworfenen Techniken nutzen dabei eine erstaunliche Breite an Erkenntnissen aus anderen Wissenschaften und anderen Teilgebieten der Informatik. Allgemeines Ziel ist es, die Expressivität der computergraphischen Techniken dadurch zu erweitern, daß Graphiken in verschiedenen Stilen erzeugt werden können. Den hier verwendeten Stilbegriff kann man eher damit übersetzen, daß er eine künstlerische Technik beschreibt, die mit dem entsprechenden Algorithmus nachgeahmt werden soll. Tatsächlich stellen viele der Techniken, die in der nicht-photorealistischen Computergraphik behandelt werden eine Nachahmung künstlerischer Techniken dar – oder korrekter: eine Nachahmung der Effekte, die mit künstlerischen Techniken erzielt werden. Im folgenden soll auf drei große Gebiete eingegangen werden:

- Liniengraphiken: Hier stehen das Zeichnen einer einzelnen Linie und die Kombination mehrerer Linien zu einer Graphik im Mittelpunkt.
- Gemäldeartige Graphiken: Hier entstehen Bilder aus der Kombination mehrerer „Pinselstriche“, die jeder für sich eine entsprechende Charakteristik haben.
- Flächenhafte Graphiken. Hier werden Flächen im Bild komplett gefüllt, meist unter Ausnutzung alternativer Beleuchtungsmodelle.

Diese drei Gruppen stellen nur einen Teil der Vielfalt von NPR-Graphiken dar. Diese sind aber typisch für die Art und Weise, in der Techniken entwickelt werden. Auch ist die oben getroffene Einteilung nicht als Strukturierung des Gebietes an sich zu verstehen. Die folgenden Ausführungen sollen dazu dienen, die verwendeten Techniken näher zu beschreiben und daraus die Frage zu beantworten, wie weit eine automatische Erzeugung solcher nicht-photorealistischer Graphiken möglich und sinnvoll ist.

## *Liniengraphiken*

Bei der Erstellung von Liniengraphiken steht zunächst das Aussehen einer einzelnen Linie im Zentrum der Betrachtungen. Linien, die mit Hilfe von Computeralgorithmen generiert werden, beschreiben üblicherweise den geometrischen Ort aller Punkte, die eine bestimmte – meist mathematisch formulierte Bedingung erfüllen. Geraden beispielsweise werden durch eine einfache lineare Beziehung beschrieben. Komplexere Linien erfordern auch komplexere Beschreibungen, wie Splines und andere Freiformkurven zeigen (vgl. Farin 1994). Durch diese mathematische Grundlage wirken computergenerierte Linien eher „steril“ und wenig lebhaft – im Gegensatz zu handgezeichneten Linien. Studien, wie die von Schumann et al. (Schumann 1996) haben gezeigt, daß gerade durch diese Lebhaftigkeit Handzeichnungen beispielsweise eher zu Diskussionen anregen und deshalb auch häufig für die Präsentation von Designentwürfen eingesetzt werden.
Das Hinzufügen von Unregelmäßigkeiten bei der Erzeugung von computergenerierten Linien bedeutet, daß diese Unregelmäßigkeiten mathematisch erfaßt werden müssen. Dies gelingt durch die Einbeziehung von Zufallsfunktionen. Die so entstehenden Linien sind „zufällig unregelmäßig“, ein Charakteristikum, daß handgezeichnete Linien nur zu einem gewissen Teil aufweisen. Daher ist die Beschreibung von Linien unter Zuhilfenahme von Zufallsfunktionen nur eine Teillösung. Generell werden für NPR-Linien zwei Wege beschritten. Einerseits können Linien durch Überlagerung zweier Kurven erzeugt werden,

andererseits können auch charakteristische Muster entlang einer gegebenen Kurve verzerrt werden. Gemeinsam ist beiden Ansätzen, daß der grobe Verlauf einer Linie als „Pfad“ vorgegeben wird. Im ersten Ansatz werden dann die Abweichungen von diesem Pfad (die bei Handzeichnungen durch Unregelmäßigkeiten des Materials auftreten oder durch unregelmäßige, unbewußte Handbewegungen) ebenfalls als Kurven beschrieben und diese auf den Pfad aufaddiert. (Schlechtweg 1998; Finkelstein 1994) Dadurch können insbesondere geometrische Stile erzeugt werden. Im zweiten Fall werden die Abweichungen durch Muster kodiert, die dann als Bild aufgefaßt so transformiert werden, daß die Hauptachse des Bildes mit dem Pfad übereinstimmt. Hier können interessante visuelle Effekte entstehen, aber auch reale Zeichenmaterialien nachgebildet werden. (Hsu 1994).
Das Erstellen einer Liniengraphik erfordert schließlich die Kombination mehrerer Linien. Erst durch diese Kombination ist es möglich, sowohl Textur als auch Helligkeit im Bild zu vermitteln. Die Kombination erfolgt dabei üblicherweise regelbasiert, um Texturen zu erzeugen. Diese Regeln beschreiben, in welcher Reihenfolge die einzelnen Linien zu zeichnen sind um eine bestimmte Textur zu generieren (Winkenbach 1994; Salisbury 1994). Beispielsweise werden für eine Ziegelsteintextur zuerst die Umrisse der einzelnen Ziegel gezeichnet, dann Schraffuren in eine Richtung hinzugefügt, diese dann durch Schraffuren in weitere Richtungen ergänzt. Die Helligkeit wird durch eine bestimmte Anzahl an Linien erreicht. Bei ständigem Vergleich mir einem Referenzbild werden so lange Linien hinzugefügt, bis ein entsprechender Helligkeitswert erreicht ist.

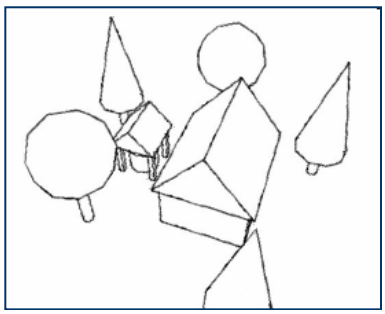


***Abbildung 1: Bleistiftähnliche Linien***

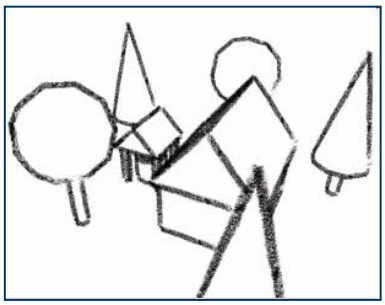


***Abbildung 2: Zeichenkohleartiger Linienstil***

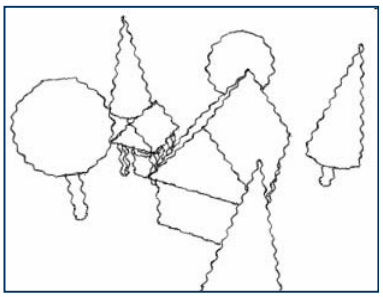


***Abbildung 3: starke geometrische Variation entlang der Linien***

Diese Herangehensweise ermöglicht es, weit mehr Parameter und damit Gestaltungsmöglichkeiten zu identifizieren und zu nutzen, als das bei photorealistischen Graphiken der gleichen Szene der Fall wäre. Jede einzelne Linie ist in ihrem Aussehen parametrisierbar, wie die Abbildungen 1–3 zeigen. Weiterhin ist die Kombination der Linien auch an die Gegebenheiten anpaßbar und liefert unterschiedliche Graphiken aus ein und demselben Modell. Abbildung 4 zeigt eine komplexere Graphik, die sowohl Umrißlinien als auch Schraffuren enthält. Farbe kann auch eingesetzt werden, um die Wahrnehmung der räumlichen Gegebenheiten der Szene weiter zu unterstützen oder als künstlerisches Mittel.

## *Gemäldeartige Graphiken*

Für computergenerierte Graphiken, die die visuellen Effekte von Gemälden nachbilden, können zunächst die gleichen Techniken verwendet werden, wie sie bei Liniengraphiken schon angesprochen werden. Einzelne „Pinselstriche“ – sogenannte Strokes werden nach bestimmten Regeln plaziert. Der Ansatz, über die Helligkeiten zu entscheiden, wie viele Pinselstriche gesetzt werden, muß dann allerdings erweitert werden um eine Entscheidung über die Farbe der Strokes (Kowalski 1999). Jeder einzelne Stroke wiederum hat ein bestimmtes Aussehen, so daß unterschiedliche Pinsel nachgebildet werden können.

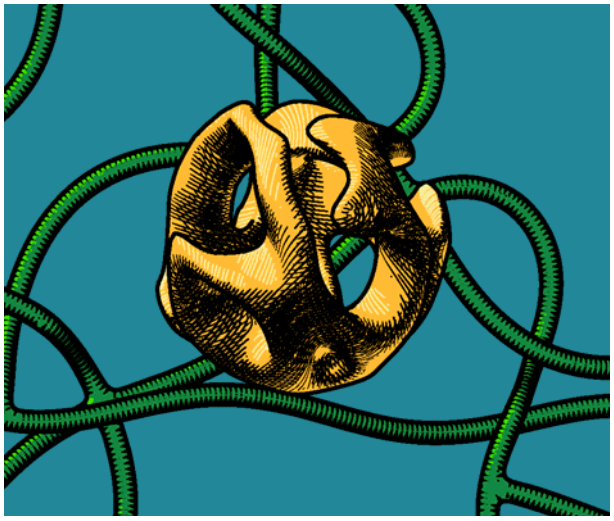


***Abbildung 4: Kombination aus Umriß- und Schraffurlinien in verschiedenen Farben***

Abbildungen 5 und 6 zeigen ein auf diese Art entstandenes Bild. Auch hier ergeben sich sehr viele Gestaltungsmöglichkeiten durch die Parametrisierung der einzelnen Strokes und des Platzierungsalgorithmus.
Neben diesen approximativen Methoden, gemäldeähnliche Graphiken zu erzeugen, existieren auch physikalisch basierte Techniken. Diese streben eine mehr oder weniger umfassende Simulation der beim Malen stattfindenden physikalischen Prozesse an. Zum einen wird dabei die Verteilung der Farbe auf dem Papier mittels flüssigkeitsdynamischer Gesetzmäßigkeiten simuliert (Curtis 1997; Zhang 1999). Solche Simulationsalgorithmen berücksichtigen z.B. die Verdunstungs- oder Absorptionsrate der Farbe und ermitteln daraus, wie die Farbe auf dem Papier verläuft. Dies führt bis hin zu Simulationen des Trocknungs- und Alterungsprozesses von Gemälden. Zum anderen werden die genauen optischen Eigenschaften der Farbpigmente analysiert und in Modellen nachgebildet, so daß die entstehenden Bilder auch hier die Effekte realer Gemälde nachbilden. Die rechnerisch sehr aufwendigen Simulationen führen zu erstaunlich realistischen Ergebnissen (vgl. Curtis 1997). Diesen Simulationen sind allerdings auch Grenzen gesetzt. So wirken Ölgemälde beispielsweise sehr stark durch die Dreidimensionalität des Farbauftrages und die dadurch entstehenden Reflexionen und Schatten, die aber auf zweidimensionalen Medien (wie das bei Computerbildschirmen der Fall ist) sehr schwer nachzubilden sind.

## *Flächige Graphiken*

Graphiken, bei denen keine Pinselstriche oder Linien auf der Zeichenfläche verteilt werden

***Abbildung 5: 3D-Szene mit computergenerierten Pinselstrichen***

***Abbildung 6: unterschiedliche Parametrisierung der Pinselstriche führt zu unterschiedlichem Aussehen***

wodurch der zu erzielende Effekt entsteht, sondern bei denen für jeden Pixel eine Helligkeit bzw. Farbe bestimmt wird, sollen im folgenden flächige Graphiken genannt werden. Die gesamte Bildfläche wird dabei entsprechend einem Algorithmus „gefüllt“. Die Algorithmen zum Bestimmen der Farbwerte der Pixel orientieren sich an den Beleuchtungsmodellen, die in der photorealistischen Computergraphik verwendet werden, legen aber keine oder nur grundlegende physikalische Gesetzmäßigkeiten zu Grunde. Hier werden eher die Herangehensweisen von Künstlern nachgebildet, die beispielsweise über Farbtemperaturen bestimmte Stimmungen hervorrufen oder Objekte gruppieren können, oder die ein vereinfachtes – nur aus zwei oder drei Helligkeitsstufen bestehendes – Beleuchtungsmodell verwenden, wie es in Comics üblich ist.
Die in der Computergraphik verwendeten Beleuchtungsmodelle sind nur Approximationen der realen physikalischen Vorgänge. Maler und Zeichner weichen häufig von der physikalischen Beleuchtung komplett ab und verwenden eigens definierte Farbbereiche, in die sie dann die einzelnen Helligkeiten abbilden. Sehr oft wird dazu die Farbtemperatur als Grundlage verwendet und es findet eine Interpolation zwischen einer kalten und einer warmen Farbe statt. Kombiniert man diesen Übergang mit einer Interpolation zwischen hell und dunkel, erzeugt man Untertöne, die gewisse geometrische Eigenschaften deutlicher hervortreten lassen oder die auf sonstige Weise dem Bild Charakter verleihen. Algorithmisch kann dies dadurch erfolgen, daß die Parameter, die in physikalisch basierten Beleuchtungsmodellen (insbesondere Winkel zwischen Oberfläche und Beleuchtungsrichtung) verwendet werden hier zur Interpolation zwischen den zusätzlichen Farben genutzt werden (Gooch 1998). Zur Umsetzung einer Comic-artigen Beleuchtung werden die (photorealistisch) berechneten Helligkeitswerte quantisiert und diese Helligkeitsstufen dann in unterschiedlichen Farben dargestellt (Lake 2000). Eine Quantisierung anhand eines Schwellwertes führt dann zu den bekannten Graphiken mit einem dunkleren und einem helleren Ton. Weitere Techniken existieren hier, um Highlights in den aus Comics üblichen stilisierten Formen zu generieren (Anjyo 2003)
Obige Aufzählung stellt nur einen kleinen Ausschnitt aus der Vielfalt der Möglichkeiten dar, wie nicht-photorealistische Graphiken erzeugt werden können. Gemeinsam ist allen, daß eine Menge zusätzlicher Parameter den Bilderzeugungsprozeß steuern, die vom Benutzer möglichst intuitiv verstanden und genutzt werden müssen.

## Interaktion als Teil der Bilderzeugung

Photorealistische Bilderzeugung liefert eine Darstellung der modellierten Szene, die sich auf physikalische Gesetzmäßigkeiten begründet. Die Gestaltungsparameter sind denen der Realität ähnlich: Plazierung und Parameter der Lichtquellen, Plazierung und Materialien der Objekte, Auswahl des Kamerastandpunktes und der Kameraparameter. Nach der Modellierung hat der Benutzer also relativ wenig Einfluß auf die Gestaltung des Bildes (siehe auch Hoppe 1998). Dies ist bei nicht-photorealistischen Graphiken anders. Die (stilistische) Gestaltung des Bildes ist hier relativ weit losgelöst von der eigentlichen Modellierung, so daß aus einem Modell viele verschiedene Darstellungen erzeugt werden. Auch aufgrund der Vielzahl der einstellbaren Parameter ist eine automatische Erzeugung nicht-photorealistischer Graphiken zwar möglich aber wenig zweckmäßig.
Ein weiterer Punkt spielt hier eine große Rolle: NPR-Graphiken sollen eine Kommunikationsziel vermitteln. Daher steht hinter einem solchen Bild immer auch eine Intention bzw. eine Idee des Benutzers, der das Bild erzeugt. Diese muß natürlich dann in den Prozeß der Bildgenerierung einfließen. Benutzerinteraktion wird damit zum integralen Bestandteil der Bilderzeugung (Halper 2002). Betrachtet man heute übliche Benutzungsschnittstellen, erkennt man schnell deren Beschränkungen in Bezug auf die Integration in den Bilderzeugungsprozeß. Typische WIMP-Schnittstellen (Windows, Icons, Menus and a Pointing device) sind mit Dialogboxen, Menüs und anderen Interface-Elementen

eigentlich prädestiniert für die Eingabe numerischer und anderer Parameter, erfüllen hier Ihren Zweck aber nicht aus zwei Gründen

(a) Der Benutzer wird durch die Benutzung der Interface-Elemente (die nach entsprechenden Regeln auf dem Bildschirm angeordnet sind) von der eigentlichen Tätigkeit abgelenkt
(b) Die (mentale) Abbildung von erwünschten Effekten auf numerische Werte ist schwierig.

Das Erzeugen von NPR-Bildern ist mit einem Designprozeß vergleichbar, der auch immer Kreativität und kreative Aktionen beinhaltet. Desweiteren ist dies ein iterativer Vorgang, bei dem Varianten akzeptiert und verworfen werden, bei dem schrittweise eine begonnene Darstellung verfeinert wird und bei dem auch Ideen entstehen, die nicht vom Programmierer einer entsprechenden Anwendung vorhergesehen werden können. Diese auf „versuch und Irrtum“ beruhende Herangehensweise muß demnach auch von einer entsprechenden Anwendung und ebenso von ihrer Benutzungsschnittstelle unterstütz werden.
Halper et al. stellen hierzu das System OpenNPAR vor, bei dessen Entwicklung die oben genannten Punkte im Mittelpunkt standen (Halper 2002; Halper 2003). Insbesondere die Benutzungsschnittstelle ist innovativ und unterstützt den kreativen Prozeß. Abbildung 7 zeigt einen Screenshot einer Sitzung mit OpenNPAR, der im folgenden näher erläutert wird.

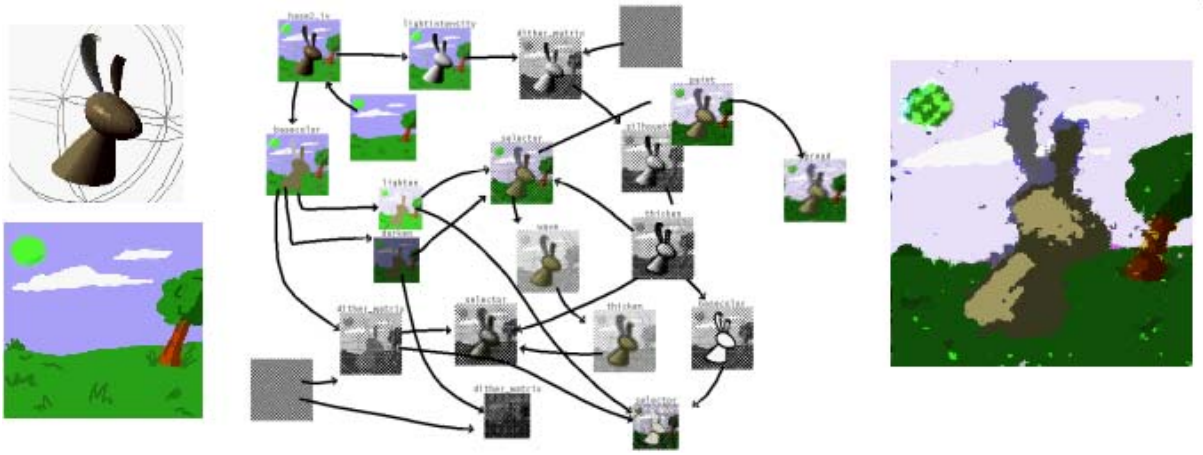


**Abbildung 7: Benutzungsschnittstelle von OpenNPAR**

Als Ausgangspunkt für die Bilderzeugung dient das (2D-)Bild einer Landschaft und ein 3D-Modell eines Hasen, wie sie in Abbildung 7 links zu sehen sind. Beide werden integriert und bilden den Ausgangspunkt für die Gestaltung der Graphik (Mittelteil der Abbildung links oben). Der Benutzer hat nun die Möglichkeit, verschiedene Operatoren auf diese Graphik anzuwenden. Dabei wird nicht – wie bisher üblich – nach der Schrittfolge „Bereich auswählen“, „Operator auswählen“, „Operator anwenden“ verfahren, sondern die im momentanen Kontext möglichen Operatoren werden dem Benutzer angeboten. Der Benutzer entscheidet sich für eine Ausgangssituation und zieht mit der Maus (oder dem Stift) eine Verbindung zu einer freien Stelle auf dem Hintergrund. Dort kann er auch durch einfaches Zeichnen eines Rechtecks die neue Größe des Bildes festlegen. Dann bekommt er in einem Menü alle möglichen Operatoren angeboten, nach der entsprechenden Auswahl wird dieser dann auf das Bild angewendet und das neue Bild dargestellt. Verfolgt man die obere Reihe im Mittelteil von Abbildung 7, wurde hier zunächst die Materialinformation aus dem Hasen entfernt und dann das Bild mit einem speziellen Halftoning-Verfahren in ein Schwarz-Weiß-Bild umgewandelt. Die entsprechenden Operatoren wirken nur dort, wo sie definiert sind. Das Entfernen des Materials ist beispielsweise eine Operation auf dem 3D-Modell, das Halftoning-Verfahren eine 2D-Bildoperation. Auf diese Art und Weise entstehen immer neue Darstellungen. Zwischenergebnisse können als Ausgangspunkt für neue Ideen verwendet werden, verschiedene Zweige der „Evolution“ der Darstellung können miteinander

kombiniert werden bis letztenendes eine Graphik entsteht, die den Zielvorstellungen des Benutzers entspricht (Abbildung 7, rechts).
Mögen die entstehenden „Graphen“ der Evolutionsgeschichte eines Bildes auf den ersten Blick auch unübersichtlich erscheinen, so kommen sie dem „Versuch und Irrtum“-Ansatz wesentlich näher als eine Auflistung von Aktionen in einer Liste – wie es für die „Undo“- und „Redo“-Funktionalität herkömmlicher Anwendungen Standard ist.
Das hinter der Benutzungsschnittstelle arbeitende System OpenNPAR ist ein modulares System mit frei konfigurierbaren Rendering-Pipelines, das auf OpenInventor basiert. Einige der im ersten Teil des Beitrages besprochenen Rendering-Stile sind hier implementiert; das System ist aber dahingehend einfach erweiterbar. Die unterliegende Technik der Verwaltung und Konfiguration der Rendering-Pielines bleibt dem Benutzer verborgen. Es werden nur Effekte angeboten, die angewendet werden können. Dadurch und durch die Verlagerung der meisten Elmente der Benutzungsschnittstelle an die Stelle der Arbeitsoberfläche, an der momentan gearbeitet wird, ist eine Konzentration auf die kreative Tätigkeit möglich. Zwischenergebnisse bleiben erhalten und können für weitere Experimente genutzt werden. Auch das unterstützt kreatives Arbeiten. Schließlich ist die Anordnung der Elemente auf dem Bildschirm nicht vorgegeben, so daß jeder Benutzer den Arbeitsplatz entsprechend seiner eigenen Arbeitsweise gestalten und anordnen kann.

## Zusammenfassung

In diesem Betrag standen zwei Schwerpunkte in Mittelpunkt. Einerseits eine Betrachtung, wie die Expressivität der computergraphischen Methoden gesteigert werden kann und andererseits wie der Benutzer besser mit den dadurch entstehenden neuen Techniken und Methoden umgehen kann. Die nicht-photorealistische Computergraphik erweitert das, was bisher als Computergraphik bekannt war um Stile, also um alternative Darstellungsmöglichkeiten. Dadurch entwickeln sich natürlich auch neue Anwendungsgebiete bzw. werden die Anwendungen der Computergraphik weiter ausgebaut. Einige Möglichkeiten der Anwendung solcher alternativen Darstellungen seien hier kurz genannt:

- *(Informations-)Visualisierung:*
  Die Visualisierung von realen physikalischen Daten oder auch von abstrakten Daten wird durch die Bereitstellung weiterer sogenannter Präsentationsvariablen stark erweitert.
- *Unterstützung von Planung und Design:*
  Zusätzliche „ungenaue“ Stile machen computergenerierte Graphiken zu einem Werkzeug für die Vermittlung von Ideen auch in sehr frühen Phasen des Designprozesses.
- *Künstlerische Darstellungen:*
  Mit Hilfe der Verfahren des nicht-photorealistischen Renderings können auch künstlerische Graphiken erzeugt werden.

Die Techniken zur Bilderzeugung werden seit den 1990er Jahren erforscht und untersucht. Zum gegenwärtigen Zeitpunkt stehen Arbeiten im Mittelpunkt, die neue Anwendungsfelder für diese Techniken im obigen Sinne erschließen. Neuartige Interaktionstechniken, die die Anwendung nicht-photorealistischer Verfahren unterstützen und die – wie oben argumentiert – zu einem integralen Bestandteil der Bilderzeugung werden müssen, standen bisher eher weniger im Mittelpunkt. Hier ist eine interdisziplinäre Zusammenarbeit zwischen Informatikern, die die Algorithmen entwickeln, Designern, die sich mit der Unterstützung der Benutzer beschäftigen und Wissenschaftlern aus den Anwendungsgebieten gefordert. In diesem Sinne ist dieser Beitrag auch als Plädoyer für eine solche interdisziplinäre Zusammenarbeit im Rahmen der Bildwissenschaft zu verstehen.

# Literatur


Anjyo, K.; Hiramitsu,. K.: Stylized Highlights for Cartoon Rendering and Animation. In: *IEEE Computer Graphics and Applications*, 4, 2003, S. 54–61

Curtis, C. J.; Anderson, S. E.; Seims, J. E.; Fleischer, K. W.; Salesin, D. H.: Computer-GeneratedWatercolor. In: Whitted, T. (Hrsg.): *Proceedings of ACM SIGGRAPH 1997*. New York. ACM SIGGRAPH. New York [ACM Press/ACM SIGGRAPH] 1997, S. 421–430.

Farin, G.: *Kurven und Flächen im Computer Aided Geometric Design*, Braunschweig · Wiesbaden [Vieweg] 1994.

Finkelstein, A.; Salesin, D. H. : Multiresolution Curves. In: Glassner, A. (Hrsg.): *Proceedings of ACM SIGGRAPH 1994.* New York [ACM Press/ACM SIGGRAPH] 1994, S. 261–268.

Foley, J. D.; van Dam, A.; Feiner, S. K.; Hughes, J. F.: *Computer Graphics. Principle and Practice*. Reading [Addison Wesley] 1990.

Gooch, A. A.; Gooch, B.; Shirley, P.; Cohen, E.: A Non-Photorealistic Lighting Model for Automatic Technical Illustration. In: Cohen, M. (Hrsg.): *Proceedings of ACM SIGGRAPH 1998.* New York [ACM Press/ACM SIGGRAPH] 1998, S. 447–452.

Halper, N.; Schlechtweg, St.; Strothotte, T.: Creating Non-Photorealistic Images the Designer's Way. In: *Proceedings of NPAR 2000*. New York [ACM Press] 2000, S. 97–104

Halper, N.; Isenberg, T.; Ritter, F.; Freudenberg, B.; Meruvia, O.; Schlechtweg, St.; Strothotte, T. OpenNPAR: A System for Developing, Programming, and Designing Non-Photorealistic Animation and Rendering. In: *Proceedings of Pacific Graphics 2003*. Los Alamitos [IEEE Computer Society] 2003, S. 424-428

Hoppe, A.: *Validierung und Nachbearbeitung von gerenderten Bildern*. Dissertation Otto-von-Guericke-Universität Magdeburg, Magdeburg, 1998.

Hsu, S. C.; Lee, I. H. H.: Drawing and Animation Using Skeletal Strokes. . In: Glassner, A. (Hrsg.): *Proceedings of ACM SIGGRAPH 1994.* New York [ACM Press/ACM SIGGRAPH] 1994, S. 109–118.

Kajiya, J.T.: The Rendering Equation. In: *Computer Graphics*, 4, 1986, S. 143–150

Kolb, C., Mitchell, D., Hanrahan, P.: A Realistic Camera Model for Computer Graphics. In: Cook, R; (Hrsg.): *Proceedings of ACM SIGGRAPH 1995*. New York [ACM Press/ACM SIGGRAPH] 1995, S. 317–324

Kowalski, M. A.; Markosian, L.; Northrup, J. D.; Bourdev, L.; Barzel, R.; Holden, L. S.; Hughes, J. F.: Art-Based Rendering of Fur, Grass, and Trees. In: Rockwood, A. (Hrsg.): *Proceedings of ACM SIGGRAPH 1999.* New York [ACM Press/ACM SIGGRAPH] 1999, S. 433–438.

Lake, A.; Marshall, C.; Harris, M.; Blackstein, M. Stylized Rendering Techniques for Scalable Real-time 3D Animation. In: *Proceedings of NPAR 2000*. New York [ACM Press] 2000, S. 13–20

Meisel, L. K.: *Photorealism*. New York [Harry N. Abrams Inc.] 1989.

Phong, B.-T.: Illumination for Computer Generated Pictures. *Communications of the ACM*, 6, 1975, S. 311–317

Salisbury, M. P.; Anderson, S. E.; Barzel, R.; Salesin, D. H.: Interactive Pen-and-Ink Illustration. In: Glassner, A. (Hrsg.): *Proceedings of ACM SIGGRAPH 1994.* New York [ACM Press/ACM SIGGRAPH] 1994, S. 101–108.

Schlechtweg, St.; Schönwälder, B.; Schumann, L.; Strothotte, T.: Surfaces to Lines: Rendering Rich Line Drawings. In: Skala, V. (Hrsg.): *Proceedings of WSCG'98, The 6th International Conference in Central Europe on Computer Graphics and Visualization*, 1998, S. 354–361.

Schumann, J.; Strothotte, T.; Raab, A.; Laser, S.: Assessing the Effect of Non-photorealistic Rendered Images in CAD. In: *Proceedings of ACM CHI'96 Conference on Human Factors in Computing Systems.* New York [ACM SIGCHI] 1996, S. 35–42

Strothotte, C.; Strothotte, T.: *Seeing Between the Pixels. Pictures in Interactive Computer Systems*. Berlin · Heidelberg · New York [Springer] 1997.

Strothotte, T.; Schlechtweg, St.: *Non-Photorealistic Computer Graphics: Modeling, Rendering, and Animation.* San Francisco [Morgan Kaufmann] 2002.

Winkenbach, G.; Salesin, D. H.: (1994). Computer-Generated Pen-and-Ink Illustration. In: Glassner, A. (Hrsg.): *Proceedings of ACM SIGGRAPH 1994.* New York [ACM Press/ACM SIGGRAPH] 1994, S. 91–100.

Zhang, Q.; Sato, Y.; ya Takahashi, J.; Muraoka, K.; Chiba, N.: Simple Cellular Automaton-based Simulation of Ink Behaviour and Its Application to Suibokugalike 3D Rendering of Trees. In: *The Journal of Visualization and Computer Animation*, 1, 1999, S. 27–37.

# C

## A DEVELOPER'S GUIDE TO SILHOUETTE ALGORITHMS FOR POLYGONAL MODELS

Bibliographische Angaben:

Tobias Isenberg, Bert Freudenberg, Nick Halper, Stefan Schlechtweg und Thomas Strothotte: „A Developer's Guide to Silhouette Algorithms for Polygonal Models". *IEEE Computer Graphics and Applications*, 23(4):(2003), S. 28–37.

Abstract:

Generating object silhouettes lies at the heart of nonphotorealism. Algorithms for computing polygonal model silhouettes are surveyed to help find the optimal approach for developers specific needs.

# A Developer's Guide to Silhouette Algorithms for Polygonal Models

**Tobias Isenberg, Bert Freudenberg, Nick Halper, Stefan Schlechtweg, and Thomas Strothotte**
*Otto-von-Guericke University of Magdeburg, Germany*

**Generating object silhouettes lies at the heart of nonphotorealism. Algorithms for computing polygonal model silhouettes are surveyed to help find the optimal approach for developers' specific needs.**

For a long time, line drawings have been a part of artistic expression (for example, any pencil or pen-and-ink drawing), scientific illustrations (medical or technical), or entertainment graphics (such as in comics). Hence, computer graphics researchers have extensively studied the automatic generation of such lines. In particular, the area of nonphotorealistic rendering has focused on two main directions of research in this respect: the generation of hatching that conveys illumination as well as texture in an image and the computation of outlines and silhouettes.

Silhouettes play an important role in shape recognition because they provide one of the main cues for figure-to-ground distinction. However, since silhouettes are view dependent, they need to be determined for every frame of an animation. Finding an efficient way to accomplish this is nontrivial. Indeed, a variety of different algorithms exist that compute silhouettes for geometric objects. This article provides a guideline for developers who need to choose between one of these algorithms for his or her application.

Here, we restrict ourselves to discussing only those algorithms that apply to polygonal models, because these are the most commonly used object representations in modern computer graphics. (For an algorithm to compute the silhouette for free-form surfaces see, for example, Elber and Cohen.[1]) Thus, we can use all algorithms discussed here to take a polygonal mesh as input and compute the visible part of the silhouette as output. Some algorithms, however, can also help compute the silhouette only, without additional visibility culling. The silhouette's representation might vary depending on the algorithm class—that is, the silhouette might take the form of a pixel matrix or a set of analytic stroke descriptions.

## Definitions and terminology

The silhouette $S$ of a free-form object is typically defined as the set of points on the object's surface where the surface normal is perpendicular to the vector from the viewpoint.[2] Mathematically, this means that the dot product of the normal $\mathbf{n_i}$ with the view vector at a surface vertex $P$'s position $p_i$ is zero: $S = \{P : 0 = \mathbf{n_i} \cdot (\mathbf{p_i} - \mathbf{c})\}$, with $c$ being the center of projection. In case of orthographic projection $(\mathbf{p_i} - \mathbf{c})$ is exchanged with the view direction vector $\mathbf{v}$.

Unfortunately, for polygonal objects this definition can't be applied because normals are only well defined for polygons and not for arbitrary points on the surface. Hence, typically no points exist on the surface where the normal is perpendicular to the viewing direction. However, we can find silhouettes along those edges in a polygonal model that lie on the border between changes of surface visibility. Thus, we define the silhouette edges of a polygonal model as edges in the mesh that share a front- and a back-facing polygon.

Some authors refer to the contour rather than the silhouette. We define a contour as the subset of the silhouette that separates the object from the background. Also, the subset of the silhouette, which in 2D divides one portion of the object from another one of the same object, is not part of the contour. Those lines are sometimes called internal silhouettes.

Other lines significant in the context of silhouettes are creases, borders, and self-intersections. Creases are edges that should always be drawn, for instance the edges of a cube. Typically, we can identify a crease by comparing the angle between its two adjacent polygons with a certain threshold. Border lines only appear in models where the mesh is not closed and are those edges with only one adjacent polygon. Lastly, self-intersection lines are where two polygons of a model intersect. They aren't necessarily part of the model's edges but are important for shape recognition. In the literature, these three additional line categories together with the silhouette lines are often called feature lines. Some silhouette detection methods make no algorithmic distinction between these types of lines. This is because you must determine

 Published by the IEEE Computer Society 

the visibility for all of these line types to generate an image. Hence, an algorithm's visibility aspect usually treats the lines in the same way. The visibility test yields visible segments of silhouette edges and potentially also feature lines. Sometimes visible segments are joined together into visible strokes.

## Classification

Every silhouette algorithm must solve two major problems. First, we must detect the set of silhouette edges. Next, we determine the visible subset thereof (visibility culling). For the purpose of this article, we classify the approaches according to how they solve each of these problems. With respect to solving the edge detection problem, we distinguish between image space algorithms that only operate on image buffers, hybrid algorithms that perform manipulations in object space but yield the silhouette in an image buffer, and object space algorithms that perform all calculations in object space with the resulting silhouette represented by an analytic description of silhouette edges. The second problem—visibility culling—is inherently solved within the algorithm for both image space and hybrid approaches of silhouette detection. However, for object space methods, we must approach this problem separately. We can categorize the algorithms that perform this visibility culling into image-space, object-space, and hybrid approaches. Besides belonging to one of these categories, the algorithms differ from each other in other aspects more relevant in practice, such as:

- whether they solve the edge detection and edge visibility problem in one step or separately,
- how they represent results (that is, an image with the lines of a specific color or an analytic set of edges),
- how precise the results are (image, subpixel, or exact precision),
- how complete their results are (finding all silhouette edges or only a subset of them),
- how much computation time the algorithm takes (real-time, interactive rates, or offline rendering),
- how much memory the algorithms needs,
- whether they allow animation of the model, and
- whether they are easy to implement.

According to these measures, we give recommendations for which algorithm to use in a certain situation in Table 1.

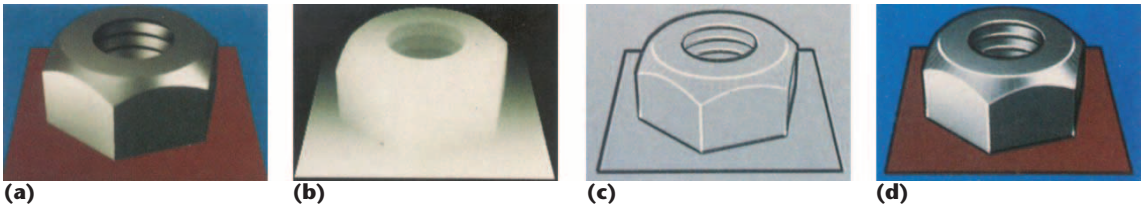
**Table 1. Choosing the best silhouette detection algorithm.**

| Requirements | Recommendations |
|---|---|
| Real-time or interactive frame rates. | Image space and hybrid algorithms.<br>Object space methods that use precomputation for silhouette detection and an image space or hybrid method for visibility culling.<br>Brute-force object space silhouette detection or silhouette visibility test not recommended. |
| Silhouettes in an analytic description. | Object space silhouette algorithms with an object space or a hybrid visibility test. |
| Exact results; pixel or subpixel accuracy is not sufficient. | Object space silhouette algorithms with an object space visibility test. (This usually slows down the program significantly.) |
| Guarantee that all silhouette edges can be found. | An object space algorithm for all edges sharing a front- and a back-facing polygon; also image space and hybrid algorithms for all silhouette edges that contribute visually to the silhouette.<br>A stochastic silhouette edge detection approach is not appropriate. |
| Animate model in real time beyond a mere flight through the scene. | Real-time technique.<br>Object space silhouette detection method that uses preprocessing is not appropriate. |
| Stylized silhouettes. | Object space silhouette detection algorithm with an object space or a hybrid visibility test.<br>Hybrid silhouette detection algorithms can also produce somewhat stylized silhouettes. |
| Cope with all model types, even those with errors. | Image space or a hybrid silhouette detection algorithm typically don't need connectivity information (polygon soups can be handled) and can deal with errors. |
| Silhouettes for a huge data set (with millions of polygons). | Image space or a hybrid silhouette detection algorithm for interactive or real-time applications.<br>All other algorithms, but these probably won't scale linearly in their computation times. |
| Least amount of memory. | Image space or hybrid algorithm (these usually don't require an additional data structure besides the geometry). |
| Compute just the contour. | Rustagi's hybrid contour algorithm.[8]<br>Object space silhouette detection algorithm with an adapted hybrid silhouette visibility test. |

## Image space algorithms

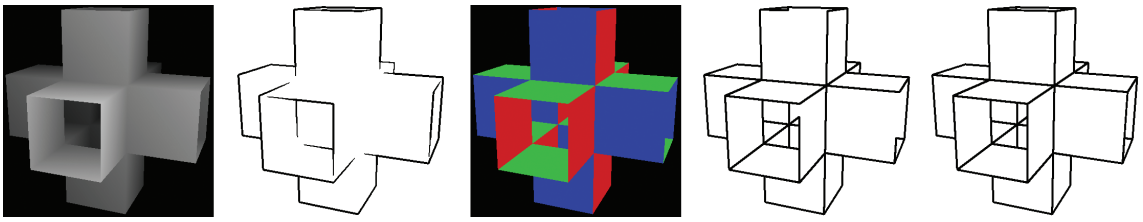
Image space algorithms exploit discontinuities in the image buffer(s) that result from conventional rendering and extract them using image-processing methods. These methods provide a silhouette represented as features in a pixel matrix.

The easiest way to find significant lines would be to detect edges in the color buffer. This, however, doesn't necessarily detect silhouettes since changes in shading and texturing can cause erroneous edge detection. Saito and Takahashi suggest using the $z$-buffer instead and applying an edge detector such as the Sobel operator.[3] This has the advantage of only finding object-relevant edges such as silhouette lines including contours, because at most of the places where silhouette lines are in the image there is a ($C^0$) discontinuity in the $z$-buffer



(a) (b) (c) (d)

**1** Image space silhouette detection based on edge detection operators on the *z*-buffer. (a) Original image, (b) *z*-buffer, (c) detected edges, and (d) composed image. (Courtesy of Saito and Takahshi[3] © 1990 ACM. Reprinted with permission.)



(a) (b) (c) (d) (e)

**2** Image space silhouette detection using edge detection operators on the *z*-buffer and the normal buffer. (a) *z*-buffer, (b) edges extracted from the *z*-buffer, (c) normal buffer, (d) edges extracted from the normal buffer, and (e) combination of both edge buffers. (Courtesy of Aaron Hertzmann[4] © 1999 ACM. Reprinted with permission. )

(see Figure 1). Hertzmann extends this method by using a normal buffer instead.[4] This can also detect $C^1$ discontinuities. Combining both approaches yields a pleasing result (see Figure 2).

Deussen and Strothotte use a simplified version of Hertzmann's algorithm to compute pen-and-ink illustrations of trees.[5] They render the tree's foliage as primitives (such as oriented disks) only to the *z*-buffer and look for discontinuities larger than a specified threshold. Depending on the primitives' size and the depth threshold this achieves the special look of pen and ink.

The advantage of image space algorithms is that we can use existing graphics hardware and rendering packages to generate the buffers on which the algorithms operate. This makes these algorithms relatively simple to implement. The computational complexity of silhouette detection in this manner depends on the number of pixels that comprise the image, rather than the number of polygons in the model, and thus is constant provided that image resolution remains constant. (Although silhouettes are not computed from polygons, these polygons must be rendered first into the required buffers.) Thus, these algorithms are usually fast, and appropriate graphics hardware can accelerate them even more. Mitchell suggests using the pixel shader technique on newer graphics cards for hardware acceleration of image space silhouette detection.[6] Finally, an image space algorithm deals with silhouette edges in the same manner as it does with feature lines. In contrast to other methods, these algorithms can automatically find self-intersections. They also share similarities with *z*-buffer rendering; the algorithms are generic and robust to errors in the models.

The main disadvantage of image space algorithms is that the user has little control over the resulting lines' attributes. The only way to directly influence the resulting lines is by choosing an edge detection operator and a source buffer on which to apply the operator. A second disadvantage is that the silhouette is not available in the form of an analytic line description. Hence, silhouettes can typically not be stylized or used for further processing. Thus, we can't readily apply many techniques simulating traditional drawing or painting utensils because they usually require this analytic information. On the other hand, some approaches extract curves from the silhouette pixels such as presented by Loviscach, who fits Bezier curves to the pixels.[7] Loviscach's approach allows for subsequent stylization of the resulting curves. However, the process of fitting curves to the pixel silhouettes might introduce new artifacts and inaccuracies. In addition, this approach is too slow for interactive or real-time applications.

Inherent in the use of image processing operators is that the resulting silhouettes don't have distinct borders. On the contrary, the intensity—the gray value—of a silhouette pixel usually depends on the derivative and thus on the intensity of features detected in the original buffer. However, this also means that the resulting images tend to not have significant aliasing problems. Another characteristic of these algorithms is that they are limited to pixel precision. This means that some fine details of less-than-pixel size might be hidden. However, for visual appearance this usually makes no difference.

## Hybrid algorithms

Hybrid algorithms are characterized by operations in object space that are followed by rendering the modified polygons in a more or less ordinary way using the *z*-buffer. This usually requires two or more rendering passes. The result is similar to image space algorithms in that the silhouette is represented in a pixel matrix.

Rustagi presents a simple algorithm using the stencil

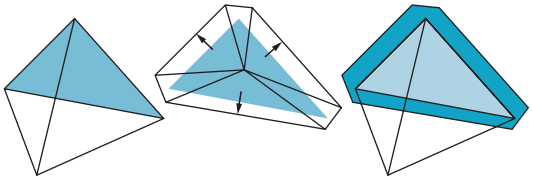
buffer that delivers the contour—not the complete silhouette—of an object.[8] The algorithm renders the object's mesh four times, each time translating the objects by one pixel in screen coordinates in the positive or negative $x$ or $y$ directions. During each pass it increments the stencil buffer where the object fills the viewport. After these four passes, the object's interior pixels will have a stencil value of four, the perimeter pixels will have values of two or three, and the exterior will have values of zero or one. Finally, setting the stencil function to pass if the stencil value is two or three and rendering a primitive larger than the object will result in only the outline pixels being changed.

Rossignac and Emmerik present a method based on $z$-buffer rendering that yields silhouettes and not only contours.[9] They show four algorithms that differ slightly from each other and that render polygonal objects either in wireframe or silhouette mode with the hidden lines either removed or dashed. For generating an image with visible silhouette and feature lines only, they first draw the object's faces into the $z$-buffer. Then they translate the mesh away from the viewer by a small distance and render a wireframe representation using wide lines. This ensures that only silhouette edges are visible since the previously rendered faces in the $z$-buffer will occlude the other lines. Finally, they translate the object forward again by twice the former distance and render the feature lines of the objects in regular line width.

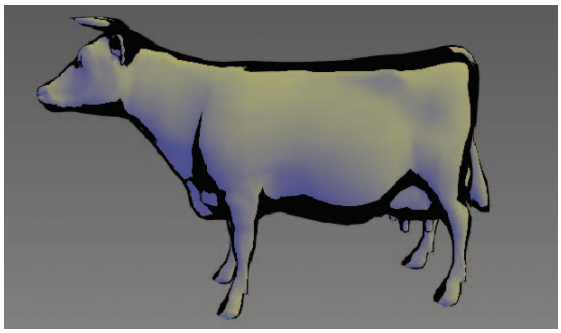
Raskar and Cohen generalize this approach in their work.[10] For automatically determining front- and back-facing polygons that define the silhouette they use traditional $z$-buffering along with back- or front-face culling, respectively. Similar to Rossignac and Emmerik, by first rendering all polygons in white on a white background with back-face culling enabled, they fill the $z$-buffer with the depth data of front-facing polygons. Afterwards, they render all polygons again, but this time in the desired silhouette color and using front-face culling. Hence, only the back-facing polygons affect the frame buffer during this second pass. By employing the *equal to* depth function, Raskar and Cohen's approach draws only the edges where the two groups of polygons meet and thus yield the model's silhouette. Rendering the wireframe representation in the second pass achieves similar results. This allows silhouette rendering with a certain line width by using thicker lines for the wireframe rendering.

Performing additional transformations before rendering the back-facing polygons can improve this general technique. A translation of the back-facing polygons toward the viewer yields thicker silhouette lines in the resulting image. However, these lines don't have a constant line width. For silhouettes with adjustable but constant width, the back-facing polygons are enlarged depending on the distance from the viewer and the angle with the viewing direction (see Figure 3).

Gooch et al. present a similar technique that also uses hardware-accelerated rendering.[11] Instead of rendering directly to a color buffer, their method draws lines into a stencil buffer. This stencil buffer acts as a mask for drawing the back-facing polygons in a second pass. Also, it can render creases in a different color than silhouettes.



**3** Enlargement of a back-facing polygon to achieve wide silhouette lines. (Courtesy of Ramesh Raskar and Michael Cohen[10] © 1999 ACM. Reprinted with permission.)



**4** Silhouette generated with a hybrid algorithm. (Courtesy of Bruce Gooch,[11] University of Utah © 1999 ACM. Reprinted with permission.)

Raskar proposes a one-pass hardware implementation that basically adds borders around each triangle.[12] This method arranges the borders in such a way that they disappear between two neighboring front-facing polygons during rendering. It also inherently includes generating crease lines where the angle between the faces exceeds a given threshold.

Another technique introduced by Gooch et al. uses environment mapping in addition to regular shading.[11] By using a partially darkened environment map, this method assigns dark values to vertices with normals that are almost perpendicular to the viewing direction, whereas other vertices remain unchanged. This method achieves a stylistic effect (see Figure 4).

The advantage of hybrid over image space algorithms is the typically higher degree of control over the outcome. The choice of algorithm and parameters such as translation or enlargement factors that control line width provide this control. The visual appearance of the generated images tends to be a more stylistic one. Also, in contrast to image space algorithms, the silhouettes inherently have distinct borders, which might be a desired trait. On the other hand, the distinct borders can cause aliasing artifacts in the image. However, we can avoid these artifacts by employing well-known antialiasing techniques.

Computation times for hybrid algorithms generally don't differ much from those for image space methods and run at interactive to real-time frame rates. Some algorithms need two or more render passes but, in return, don't require additional manipulation of the generated buffers. Object space manipulations needed for some of the algorithms might add computation time in addition to the number of rendering passes. However, the algorithms can easily make use of commonly available graphics hardware to speedup the rendering process.

The drawbacks of a pixel matrix representation of the

detected silhouette lines—as mentioned for image-space algorithms—apply to hybrid algorithms also. Similarly, the silhouette lines have pixel resolution and don't facilitate further stylization. In addition, the algorithms might have some numerical problems due to limited $z$-buffer resolution.

## Object space algorithms

To apply further stylization to the lines an analytic representation of the silhouette is needed. This is not achievable with the previously discussed algorithms because of the disadvantage of a pixel matrix representation of the computed silhouette. A good way to overcome this problem is to employ an object space algorithm. The computation of silhouettes in this group of algorithms, as the name suggests, takes place entirely in object space. In contrast to hybrid and image space algorithms, object space algorithms deal with the problems of edge detection and edge visibility in separate stages. Thus, we discuss these algorithms in two parts: methods for silhouette edge detection and, because parts of the silhouette edges might be occluded, finding the visible subset of those edges.

### *Silhouette edge detection*

A straightforward way to determine a model's silhouette edges follows directly from the definition of a silhouette. Approaches that speed up the process use some precomputed data structures, while other algorithms achieve faster execution by employing stochastic methods.

**Trivial method.** The trivial algorithm is based on the definition of a silhouette edge. The algorithm consists of two basic steps. First, it classifies all the mesh's polygons as front or back facing, as seen from the camera. Next, the algorithm examines all model edges and selects only those that share exactly one front- and one back-facing polygon. The algorithm must complete these two steps for every frame.

Buchanan and Sousa suggest using a data structure called an *edge buffer* to support this process.[13] (The edge buffer optimizes platforms such as game consoles that have certain hardware restrictions.) In this data structure they store two additional bits per edge, F and B for front and back facing. When going through the polygons and determining whether they face front or back, they XOR the respective bits of the polygon's adjacent edges. If an edge is adjacent to one front- and one back-facing polygon the FB bits are 11 after going through all polygons and we can use them for silhouette rendering. Buchanan and Sousa extend this idea to support border edges and other feature edges.

This simple algorithm—with or without using the edge buffer data structure—works for both perspective and orthographic projections. Also, it's guaranteed to find all silhouette edges in the model, be easy to implement, and be well suited for applications that only use small models. However, it's computationally expensive for common hardware-based rendering systems and the model sizes typically used with them. (For software-based rendering systems, however, the silhouettes computed with the brute-force approach come at little extra cost.) Even if we suppose an effective data structure that delivers local connectivity information in constant time, an algorithm must look at every face, determine face orientation (using one dot product per face; for perspective projection it must recompute the view vector for every face), and look at every edge. This is a linear method in terms of the number of edges and faces but too slow for interactive rendering of reasonably sized models. When we also consider that only a small number of edges typically exist that are in fact silhouette edges, testing each one also seems unnecessary.

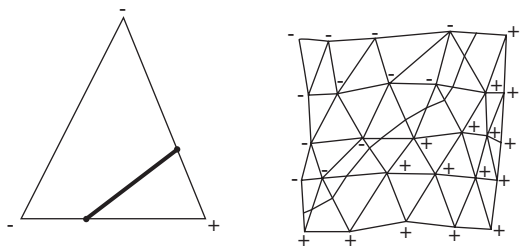
To speed up the brute-force algorithm, Card and Mitchell suggest employing user-programmable vertex processing hardware.[14] For every potential silhouette edge in the model a separate quad is generated along the edge. In addition, each vertex of such an edge stores the normals of both faces adjacent to the edge. When rendering the quads, a vertex program tests whether the normals are front or back facing. Only in cases where the result is different for both normals is the quad actually drawn.

**Subpolygon silhouettes.** Because a polygonal mesh is usually only an approximation of a free-form object, silhouettes of polygonal meshes typically have some artifacts (for example, zigzags or silhouette edge clusters). Hence, the expected silhouette of the real object can differ significantly from the silhouette that the trivial algorithm yields. Therefore, besides the quest for a fast algorithm, there are approaches that try to find a more exact silhouette, similar to that of the real object.

Hertzmann and Zorin consider the silhouette of a free-form surface approximated by a polygonal mesh.[2] To find this silhouette, they recompute the normal vectors of the approximated free-form surface in the vertices of the polygonal mesh. Using this normal, they compute its dot product with the respective viewing direction. Then, for every edge where the sign of the dot product is different at both vertices, they use linear interpolation along this edge to find the point where it is zero. These points connect to yield a piecewise linear subpolygon silhouette line (see Figure 5). The resulting silhouette line is likely to have far fewer artifacts. Also, the result is much closer to the real silhouette than the result of a traditional polygonal silhouette extraction method. Hence, it's well suited for later stylization of the lines. Figure 6 shows an example of this method.

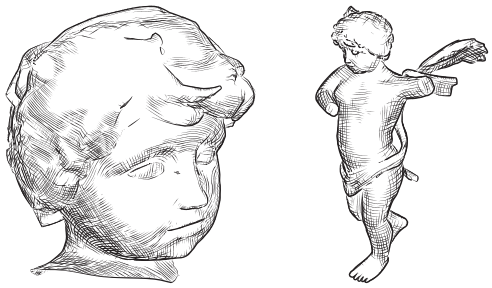
**Precomputation methods.** To speed up the process of silhouette edge determination, various authors have developed methods that accomplish some type of preprocessing. The preprocessing stage sets up data structures used to find silhouette edges more quickly.

Gooch et al.[11] and Benichou and Elber[15] present a preprocessing procedure based on projecting face normals onto a Gaussian sphere. Here, every mesh edge corresponds to an arc on the Gaussian sphere, which connects the normal's projections of its two adjacent polygons (see Figure 7). For orthographic projection, a view of the scene is equivalent to a plane through the origin of the Gaussian sphere. They further observe that every

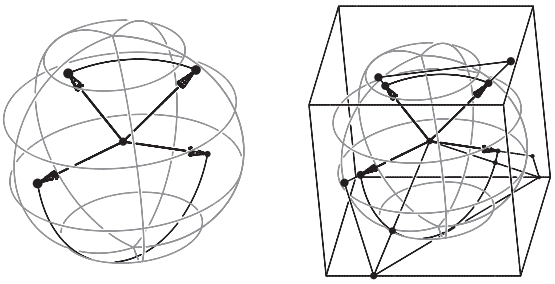


(a) (b)

**5** Computation of a subpolygon silhouette for (a) a single triangle and (b) a mesh. The dot product between normal vector and view direction is positive at vertices with plus signs and negative at vertices with minus signs. Between a positive and a negative vertex linear interpolation is used to find the silhouette. (Courtesy of Aaron Hertzmann[4] © 1999 ACM. Reprinted with permission.)

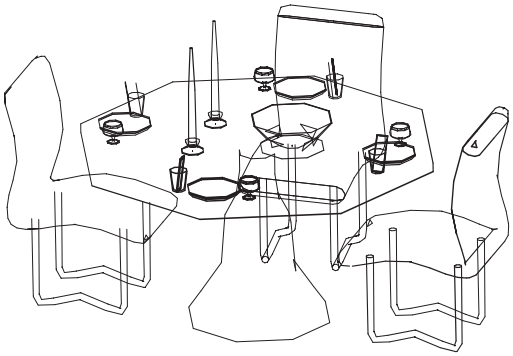


**6** Two images generated with Hertzmann and Zorin's subpolygon method. The silhouettes are combined with a hatching technique. (Courtesy of Aaron Hertzmann and Denis Zorin[2] © 2000 ACM. Reprinted with permission.)



(a) (b)

**7** Preprocessing using projection of the normals of two faces adjacent to an edge onto (a) the Gaussian sphere and (b) its surrounding cube. (Courtesy of Gershon Elber and Fabien Benichou[15] © 1999 IEEE.)



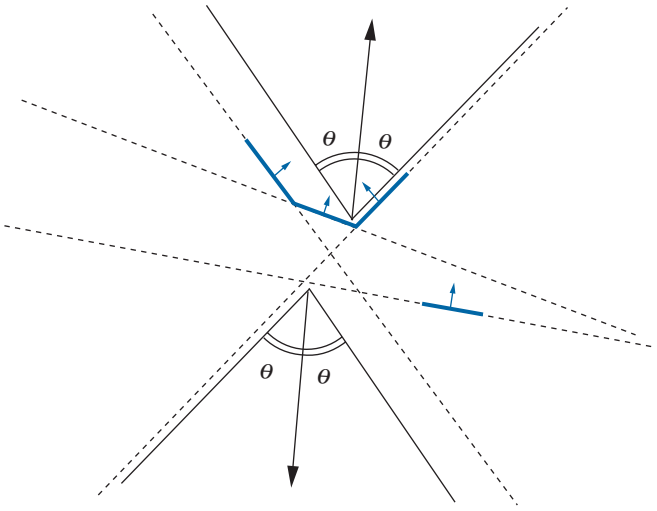
**8** Example scene rendered with the Gaussian sphere preprocessing algorithm. No visibility test was used. (Courtesy of Gershon Elber and Fabien Benichou.)

arc intersected by this plane is a silhouette edge in the corresponding view. Applying this observation to silhouette edge extraction removes the need to check for each frame if every face is front or back facing. The arcs are computed in a preprocessing step and at runtime only the intersections with the view plane are tested. Figure 8 shows an example rendering.

To further limit the number of arcs tested, Gooch et al. use a hierarchical decomposition of the sphere. They start with a platonic solid and consecutively apply subdivision to solid's sides.[11] The arcs are stored in the lowest possible level. Benichou and Elber, on the other hand, map the Gaussian sphere and the arcs to a cube surrounding the sphere.[15] The arcs are equivalent to line segments on the cube. This method maps the view plane's great arc onto the cube, resulting in a set of line segments, which simplifies the intersection test. To reduce the number of arcs tested, this approach decomposes the sides of the cube into a grid and only tests for intersection those edge line segments in grid cells that touch a viewplane line segment.

Hertzmann and Zorin[2] present a method that uses a data structure also based on a dual representation, which is, in fact, similar in principle to the one by Benichou and Elber. This approach, however, constructs a dual representation of the mesh in 4D space based on the position and tangent planes of every vertex. The viewpoint's dual (a plane in 4D) intersects with the mesh triangles' dual. Beforehand, the approach normalizes the dual vertices using the $l_\infty$ norm so that the vertices end up on one of the unit hypercube's sides. (The normalization does not make a difference because the viewpoint's dual plane goes though the origin.) This reduces the problem to intersecting the triangles on a hypercube's sides with the viewpoint's dual plane. This means you need to intersect triangles in eight 3D unit cubes (the eight hypercube sides) with a plane. Here, again, speedup occurs by employing space partitioning, namely by using an octree for each hypercube sides. At runtime, the approach only computes the viewpoint's dual plane and then intersects it with each hypercube side, resulting in edges that intersect the silhouette. The major advantage of this approach over methods discussed earlier in this section is that it works for orthographic as well as perspective projections. For orthographic projection you simply place the viewpoint at infinity.

Pop et al. present another algorithm based on a dual representation.[16] Similar to the previous approach but this time in 3D, this method constructs a dual space with




**9** Preprocessing for faster silhouette detection through arrangement of face clusters in anchored cones (visualized in 2D) by Sander et al.[17] (© 2000 ACM. Reprinted with permission.)

vertices having planes as duals and vice versa. This reduces the silhouette problem to finding intersections of the viewpoint's dual plane with the duals of the mesh edges. This, unfortunately, is still expensive to compute for every frame. These authors note, however, that you only need to find silhouette changes in two consecutive frames. To accomplish this they identify the dual edges with one vertex inside and the other vertex outside the wedge formed by the dual planes of two consecutive viewpoints. Similar to Hertzmann and Zorin's method, to speed up this process they use an octree data structure. The advantage of this approach is that it works in 3D only. Hence, it performs the search for intersections only once per frame as opposed to eight times. Also, the approach only detects silhouette changes. However, although Hertzmann and Zorin's approach requires eight octrees instead of just one, each of the eight octrees only contains one-eighth of the faces. Also, both methods should have an expected extraction time linear to the number of found silhouette edges.

Sander et al. use a different method for silhouette edge detection.[17] This approach constructs a hierarchical search tree that stores the mesh's edges. All the faces attached to edges that are stored in a node or its associated subtree make up a face cluster. At runtime, the method searches the tree recursively to find face clusters that are entirely front or back facing. All edges of such a face cluster can be discarded and the search for the associated subtree stopped. To effectively test whether a face cluster is entirely front or back facing in constant time, Sander et al. store two anchored cones in every node (see Figure 9). One cone represents those positions where the viewpoint can be located for the entire face cluster to be front facing, the other cone for the face cluster to be back facing.

Summarizing, all precomputation methods presented here reduce the number of triangles or edges checked at runtime, speeding up the silhouette detection process without trading accuracy. They accomplish this by establishing an efficient data structure during preprocessing. All authors claim to achieve at least interactive frame rates for reasonably sized models. However, these methods make animation of the models inefficient since the precomputed data structure stores information about the visibility of polygons to quickly identify silhouettes for the moving viewpoint. If a model were animated beyond a flight through the scene, this precomputed data structure would become invalid and result in new precomputation steps for every frame. On current hardware, this would reduce the frame rate to below interactive rates. In addition, precomputation algorithms need a separate elaborate data structure besides the actual geometry for achieving their speedups.

**Stochastic method.** In contrast to precomputation, Markosian et al. suggest a stochastic algorithm to gain faster runtime execution of silhouette detection.[18] They observe that only a few edges in a polygonal model are actually silhouette edges. (Typically $O(\sqrt{n})$ edges of the $n$ polygons according to Sander et al.[17].) In the hope of finding a good initial set of candidates for front- and back-face culling, they randomly select a small fraction of the edges and exploit spatial coherence. Once they detect a silhouette edge, they recursively test adjacent edges until they reach the end of the silhouette line. In addition, they also exploit spatial coherence, as the silhouette in one frame is typically not far from the (visually) similar silhouette in the next frame.

The combination of these two parts of the algorithm yields most of the silhouette edges in one image. By exploiting spatial and temporal coherence, Markosian et al. achieve fast runtime execution for interactive or real-time applications. Also, the method is not restricted to static objects, so animation does not pose a problem. However, in contrast to the precomputation algorithms we discussed previously, the algorithm can't guarantee finding the entire set of silhouette edges for a certain view on the scene.

### *Line visibility determination*

In most cases the process of computing object space silhouettes produces the problem of visibility culling of the silhouette edges. This problem is the classic computer graphics problem of hidden line removal. Similar to the problem of silhouette detection, three general ways exist to attack this problem. A fast way to solve it is an image space approach. A highly precise solution employs an object space method that yields visible silhouette segments in an analytic description. Finally, by combining both approaches you can use a hybrid method that's faster than an object space method but still yields analytic descriptions of the visible silhouette lines.

**Image space.** A trivial and fast way to determine the visibility of silhouette edges in image space is to use the $z$-buffer. The simplest method is to render the sil-

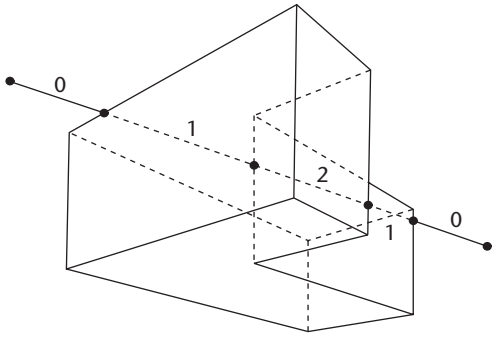
houette edges and let the $z$-buffer remove the hidden lines. The result is comparable to Raskar and Cohen's hybrid method using wireframe rendering in the second pass. A more advanced algorithm renders the silhouettes in a certain line width and perpendicular to the viewing direction. Unfortunately, this generates images with thick internal silhouettes and thin contour lines because the object itself partly occludes contour lines but not internal silhouettes.

Besides the limitation to pixel accuracy, the main disadvantage of this image space approach is that when you apply style variations to the silhouette the line might be partially occluded.

**Object space.** Researchers have proposed many algorithms for hidden line removal in object space (see, for example, Sutherland et al.[19]). Most of these also solve the problem of hidden surface removal. It would go beyond the scope of this article to address all of these algorithms, but you could use many of them to easily determine silhouette visibility. The visible line algorithm presented by Appel,[20] however, deserves further attention since it's frequently used in the context of silhouette algorithms. The algorithm is based on the notion of quantitative invisibility (QI)—the number of front-facing polygons between a point on the edge to be rendered and the viewer—which is determined for all edge segments (see Figure 10). All edge segments with a QI value of zero are visible; all others are invisible. The fact that the QI value only changes when the edge to be rendered intersects a silhouette edge in the 2D projection allows for propagation along connected sets of edges. (However, it can also change at a vertex if the vertex lies on a silhouette edge, which causes some complications for the computation.) Therefore, for every connected set of edges tested the QI value is initially established for one point using, for example, ray tracing. The QI value is then propagated along the edges to be rendered determining whether it increases or decreases each time the edge passes behind a silhouette edge.

Markosian et al.[18] use a modified version of Appel's algorithm to improve computation time. They redefine the QI value as the total number of faces between a point and the viewer. This simplifies Appel's algorithm in that the QI value can now only change at a vertex if that vertex is a so-called cusp vertex. In addition, they avoid many of the required ray tests by first finding out how the QI value changes by establishing relative QI values when traversing a connected set of edges before executing the ray test. Sometimes, this marks as invisible the whole connected set of edges, making it unnecessary to establish initial QI values. In the remaining cases, they avoid even more tests by inferring from QI values of one set of connected edges to those of others by using the relative QI values determined previously.

Hertzmann and Zorin apply this approach to their algorithm for subpolygon silhouettes.[2] First, they divide their silhouette curves into segments at points where the visibility can possibly change and determine visibility individually for each of these segments. Then, in addition to their subpolygon silhouette edges, they also determine regular (non-subpolygon) silhouette edges




**10 Example for Appel's notion of quantitative invisibility for a line passing behind the object. The dots denote the positions where the QI value changes. Adapted from Appel.[20] (© 1967 ACM. Reprinted with permission.)**

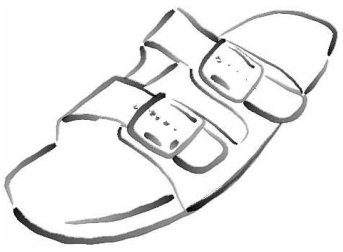
that are a subset of the input mesh. They determine edge visibility using the adapted Appel's algorithm described previously. Finally, they use the visibility of the majority of edges adjacent to a subpolygon silhouette edge segment to infer the segment's visibility. For finely tessellated objects this yields a sufficiently correct visible silhouette.

The advantage of analytic algorithms is they typically yield highly precise visibility information. However, since their computational complexity is not constant, they usually take longer to compute than image space approaches.

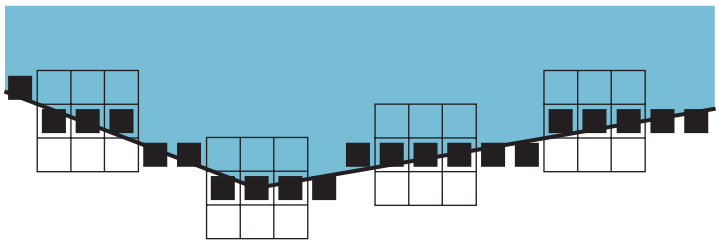
**Hybrid.** As we just showed, because they precisely solve the hidden line problem, object space silhouette visibility tests are usually expensive. A fast but less accurate way of determining silhouette edge visibility is an image space approach. However, in many applications only pixel accuracy is necessary. Thus, combining object space and image space approaches in a hybrid algorithm can achieve significant speedup for the visibility test.

In addition to regular rendering with a $z$-buffer, Northrup and Markosian use an ID buffer to determine silhouette edge visibility.[21] A unique color identifies each triangle and silhouette edge in this ID buffer. For each frame, the ID buffer is read from the graphics hardware and all reference image pixels are examined to extract all silhouette edges represented by at least one pixel. The approach then scan converts and checks for visibility the remaining silhouette edges according to whether a pixel with the edge's unique color exists in the ID buffer. Figure 11 shows an example rendering.

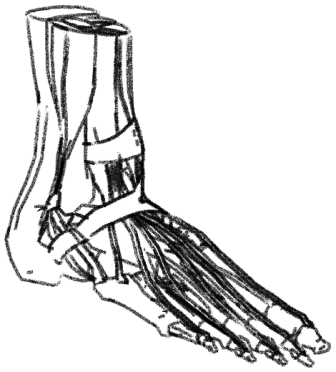
Isenberg et al. use a similar approach, in principal.[22]



**11 Image generated using Northrup and Markosian's hybrid visibility test. Line styles are used for rendering silhouette strokes. (Courtesy of J.D. Northrup[21] © 2000 ACM. Reprinted with permission.)**



**12** Testing visibility by looking at the *z*-buffer values of the pixel and its 8-pixel neighborhood. Not every pixel is tested (in this case every fifth pixel is examined). (Courtesy of Tobias Isenberg, Nick Halper, and Thomas Strothotte, *Computer Graphics Forum*, published by Blackwell Publishing.[22] Reprinted with permission.)



**13** Image generated by applying the hybrid *z*-buffer visibility test. (Courtesy of Tobias Isenberg, Nick Halper, and Thomas Strothotte, *Computer Graphics Forum*, published by Blackwell Publishing.[22] Reprinted with permission.)

However, they base their visibility test on the *z*-buffer instead of an additional ID buffer. This saves time for the second render pass otherwise required for the ID buffer. In addition, they note that simply scan converting silhouette edges and comparing the computed pixels with values in the *z*-buffer is somewhat numerically unstable. Thus, they suggest not only looking at the pixel's exact position but also at its 8-pixel neighborhood for pixels that are further away than the tested pixel (see Figure 12). This significantly reduces the numerical problems. To further speed up the process they only look at every *n*th pixel. This introduces a trade-off between accuracy and speed.

The result of either of these techniques is—as with object space visibility algorithms—a set of visible silhouette lines. We can now use them, for example, for stylized silhouette rendering. After concatenating the visible silhouettes to form paths, we can apply line styles and render them without an additional visibility test (Figure 13 shows an example). In contrast to image space visibility-tested silhouettes, style features that significantly distort the path are now completely visible.

## Future work

The various algorithms presented here can guide developers who need to find a polygonal model's silhouette. Goals for future research could include, for example, making object animation compatible with certain precomputation algorithms for object space silhouette detection—that is, to find ways to efficiently update the data structure. This would open the category of object space silhouette detection to general applications that require large models and animation. Also, recent advances in computer graphics hardware could speed the process of silhouette visibility detection in hybrid methods or even support the computation of silhouettes directly. ■

## Acknowledgments

We thank Stefan Schirra for many interesting discussions on the topic and Bert Vehmeier for his help with the images.

## References

1. G. Elber and E. Cohen, "Hidden Curve Removal for Free Form Surfaces," *Proc. Siggraph 90*, *Computer Graphics* (Proc. Ann. Conf. Series), vol. 24, ACM Press, 1990, pp. 95-104.
2. A. Hertzmann and D. Zorin, "Illustrating Smooth Surfaces," *Proc. Siggraph 00*, *Computer Graphics* (Proc. Ann. Conf. Series), S.N. Spencer, ed., ACM Press, 2000, pp. 517-526.
3. T. Saito and T. Takahashi, "Comprehensible Rendering of 3-D Shapes," *Proc. Siggraph 90*, *Computer Graphics* (Proc. Ann. Conf. Series), F. Baskett, ed., ACM Press, 1990, pp. 197-206.
4. A. Hertzmann, "Introduction to 3D Non-Photorealistic Rendering: Silhouettes and Outlines," *Non-Photorealistic Rendering* (Siggraph 99 Course Notes), S. Green, ed., ACM Press, 1999.
5. O. Deussen and T. Strothotte, "Computer-Generated Pen-and-Ink Illustration of Trees," *Proc. Siggraph 2000*, *Computer Graphics* (Proc. Ann. Conf. Series), vol. 34, ACM Press, 2000, pp. 13-18.
6. J.L. Mitchell, C. Brennan, and D. Card, "Real-Time Image-Space Outlining for Non-Photorealistic Rendering," *Siggraph 02 Conf. Abstracts and Applications*, ACM Press, 2002, p. 239.
7. J. Loviscach, "Rendering Artistic Line Drawings Using Off-the-Shelf 3-D Software," *Proc. Eurographics: Short Presentations*, I.N. Alvaro and P. Slusallek, eds., Blackwell Publishers, 2002, pp. 125-130.
8. P. Rustagi, "Silhouette Line Display from Shaded Models," *Iris Universe*, Fall 1989, pp. 42-44.
9. J.R. Rossignac and M. van Emmerik, "Hidden Contours on a Frame-Buffer," *Proc. 7th Eurographics Workshop Computer Graphics Hardware*, Eurographics, 1992, pp. 188-204.
10. R. Raskar and M. Cohen, "Image Precision Silhouette Edges," *Proc. 1999 ACM Symp. Interactive 3D Graphics*, S.N. Spencer, ed., ACM Press, 1999, pp. 135-140.
11. B. Gooch et al., "Interactive Technical Illustration," *Proc. 1999 ACM Symp. Interactive 3D Graphics*, ACM Press, 1999, pp. 31-38.
12. R. Raskar, "Hardware Support for Non-Photorealistic Rendering," *Proc. 2001 Siggraph/Eurographics Workshop on Graphics Hardware*, ACM Press, 2001, pp. 41-46.
13. J.W. Buchanan and M.C. Sousa, "The Edge Buffer: A Data Structure for Easy Silhouette Rendering," *Proc. 1st Int'l Symp. Non-Photorealistic Animation and Rendering*, ACM Press, 2000, pp. 39-42.
14. D. Card and J.L. Mitchell, "Non-Photorealistic Rendering with Pixel and Vertex Shaders," *Vertex and Pixel Shaders Tips and Tricks*, W. Engel, ed., Wordware, 2002.

15. F. Benichou and G. Elber, "Output Sensitive Extraction of Silhouettes from Polygonal Geometry," *Proc. 7th Pacific Graphics Conf.*, IEEE CS Press, 1999, pp. 60-69.
16. M. Pop et al., "Efficient Perspective-Accurate Silhouette Computation," *Proc. 17th Ann. ACM Symp. Computational Geometry*, ACM Press, 2001, pp. 60-68.
17. P.V. Sander et al., "Silhouette Clipping," *Proc. Siggraph 2000, Computer Graphics* (Proc. Ann. Conf. Series), S.N. Spencer, ed., ACM Press, 2000, pp. 327-334.
18. L. Markosian et al., "Real-Time Nonphotorealistic Rendering," *Proc. Siggraph 97, Computer Graphics* (Proc. Ann. Conf. Series), T. Whitted, ed., Addison Wesley, 1997, pp. 415-420.
19. I.E. Sutherland, R.F. Sproull, and R.A. Schumacker, "A Characterization of Ten Hidden-Surface Algorithms," *Computing Surveys*, vol. 6, no. 1, 1974, pp. 1-55.
20. A. Appel, "The Notion of Quantitative Invisibility and the Machine Rendering of Solids," *Proc. ACM National Conf.*, Thompson Books, 1967, pp. 387-393.
21. J.D. Northrup and L. Markosian, "Artistic Silhouettes: A Hybrid Approach," *Proc. 1st Int'l Symp. Non-Photorealistic Animation and Rendering*, J.-D. Fekete and D.H. Salesin, eds., ACM Press, 2000, pp. 31-37.
22. T. Isenberg, N. Halper, and T. Strothotte, "Stylizing Silhouettes at Interactive Rates: From Silhouette Edges to Silhouette Strokes," *Computer Graphics Forum* (Proc. Eurographics 2002), vol. 21, no. 3, 2002, pp. 249-258.

***Tobias Isenberg*** *is a PhD candidate in the Department of Simulation and Graphics at the Otto-von-Guericke University of Magdeburg, Germany. His research interests include feature detection on 3D shapes and nonphotorealistic rendering with an emphasis on silhouette algorithms, line stylization, and hybrid rendering. Isenberg received a BSc from the University of Wisconsin, Stevens Point, and a Diplom from the Otto-von-Guericke University of Magdeburg.*

***Bert Freudenberg*** *is a PhD candidate in the Department of Simulation and Graphics at the Otto-von-Guericke University of Magdeburg. His research interests include real-time rendering, nonphotorealistic computer graphics, and interactive educational environments. Freudenberg received a Diplom from the Otto-von-Guericke University of Magdeburg.*

***Nick Halper*** *is a PhD candidate in the Department of Simulation and Graphics at the Otto-von-Guericke University of Magdeburg. His research interests include nonphotorealistic rendering (modular systems, user interfaces, evaluating the influence of nonphotorealistic rendering), camera presentation (declarative specification, real-time techniques), and computer games (action summarization, nonphotorealistic rendering in games). Halper received an MEng in computer systems and software engineering from the University of York, England.*

***Stefan Schlechtweg*** *is an assistant professor in the Department of Simulation and Graphics at the Otto-von-Guericke University of Magdeburg. His research interests include nonphotorealistic rendering, the application of nonphotorealistic techniques in interactive systems, and visualization. Schlechtweg received a PhD in computer science from the Otto-von-Guericke University of Magdeburg.*

***Thomas Strothotte*** *is a full professor of computer science in the Department of Simulation and Graphics at the Otto-von-Guericke University of Magdeburg, where he is the Chair for Graphics and Interactive Systems. His research interests include nonphotorealistic rendering, image-text coherence in interactive systems, and techniques for rendering illustrations. Strothotte received a BSc and MSc from Simon Fraser University, a PhD in computer science from McGill University, as well as a Habilitation degree from the University of Stuttgart.*

*Readers may contact Tobias Isenberg at the Otto-von-Guericke University of Magdeburg, Dept. of Simulation and Graphics, Universitatsplatz 2, 39106 Magdeburg, Germany; isenberg@isg.cs.uni-magdeburg.de.*

## HIGH QUALITY HATCHING

D

Bibliographische Angaben:

Johannes Zander, Tobias Isenberg, Stefan Schlechtweg und Thomas Strothotte: „High Quality Hatching“. *Computer Graphics Forum (Proceedings of Eurographics 2004, Grenoble, France, August 30–September 3, 2004)*, 23(3):(2004), S. 421–430.

Abstract:

Hatching lines are often used in line illustrations to convey tone and texture of a surface. In this paper we present methods to generate hatching lines from polygonal meshes and render them in high quality either at interactive rates for on-screen display or for reproduction in print. Our approach is based on local curvature information that is integrated to form streamlines on the surface of the mesh. We use a new algorithm that provides an even distribution of these lines. A special processing of these streamlines ensures high quality line rendering for both intended output media later on. While the streamlines are generated in a preprocessing stage, hatching lines are rendered either for vector-based printer output or on-screen display, the latter allowing for interaction in terms of changing the view parameters or manipulating the entire line shading model at run-time using a virtual machine.

Weitere Veröffentlichung:

Johannes Zander, Tobias Isenberg, Stefan Schlechtweg und Thomas Strothotte: *Creating High Quality Hatching Illustrations*. Technischer Bericht 12/2004, Fakultät für Informatik, Otto-von-Guericke-Universität Magdeburg, 2004.

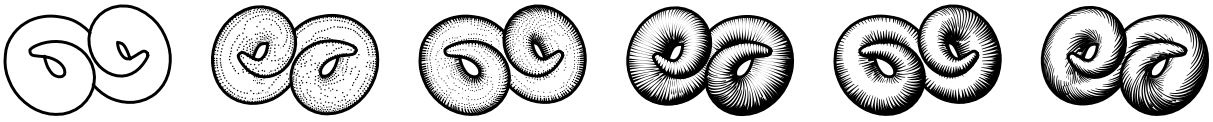
EUROGRAPHICS 2004 / M.-P. Cani and M. Slater
(Guest Editors)

*Volume 23 (2004), Number 3*

# High Quality Hatching

Johannes Zander, Tobias Isenberg, Stefan Schlechtweg, and Thomas Strothotte

Department of Simulation and Graphics
Otto-von-Guericke University of Magdeburg, Germany
jzander@cs.uni-magdeburg.de, {isenberg|stefans|tstr}@isg.cs.uni-magdeburg.de



**Abstract**

*Hatching lines are often used in line illustrations to convey tone and texture of a surface. In this paper we present methods to generate hatching lines from polygonal meshes and render them in high quality either at interactive rates for on-screen display or for reproduction in print. Our approach is based on local curvature information that is integrated to form streamlines on the surface of the mesh. We use a new algorithm that provides an even distribution of these lines. A special processing of these streamlines ensures high quality line rendering for both intended output media later on. While the streamlines are generated in a preprocessing stage, hatching lines are rendered either for vector-based printer output or on-screen display, the latter allowing for interaction in terms of changing the view parameters or manipulating the entire line shading model at run-time using a virtual machine.*

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Picture/Image Generation—Display algorithms; I.3.3 [Computer Graphics]: Picture/Image Generation—Line and curve generation

**Keywords:** non-photorealistic rendering, high quality hatching, line rendering, line shading

## 1. Introduction

The area of non-photorealistic rendering (NPR) has become a rapidly growing field in computer graphics over the last two decades. The main goal behind NPR is to enrich the expressiveness of computer graphics techniques by generating synthetic images that embody qualities of hand-drawn imagery. One of the techniques receiving high interest is the creation of line drawings. Here, two classes of lines need to be generated and, thus, distinguished. First, there is the outline or silhouette that closes an object and segregates it from the surrounding. Second, there are hatching lines that collectively convey tone as well as texture of an object's surface.

In this paper we present a way to generate and render hatching lines based on a three-dimensional polygonal mesh (in particular, we use triangle meshes). Most approaches for line rendering today aim for a fast generation and rendering of lines possibly exploiting capabilities of graphics hardware. These methods compute the lines on the surface of the model in order to avoid artifacts such as incoherence and shower-door-effect but keep the speed. However, the lines are finally output in pixel images yielding sampling artifacts and reducing the quality of the images. Other techniques generate the lines after the model has been projected into 2D which resembles the traditional way of generating line drawings, for example, in engravings, copper plates, or pen-and-ink drawings.

Our method aims for the generation of vector oriented hatching in order to yield line renditions with a higher quality. Since we compute the lines directly in 3D, additional procedures are needed to achieve high quality, rendering speed, and esthetic appeal of the resulting images. Our approach not only allows us to reproduce images in an appropriate quality for printing. We also give the designer of the drawings the opportunity to interactively work with the rendition when creating it. This comprises being able to manipulate the view on the model as well as adapting parameters of the hatching process including the shading model.

Our method does therefore not achieve the frame-rates of the aforementioned real-time hardware shading techniques but does, nonetheless, allow for interactive rendering. At the same time we also have the possibility to create versions of the rendition adapted to the desired reproduction quality. We use OpenGL lines (or similar technologies) for rendering on screen but change to a vector based description for high quality output.

The main contributions of this paper are therefore:

- object-space generation of streamlines to allow for generation of hatching illustrations with interactive frame-rates as well as off-line,
- a new algorithm to achieve an even distribution of the object-space streamlines computed from a polygonal mesh,
- the use of a virtual machine to replace the formulas of the line shading model at run-time allowing for changes that go beyond a mere parameter adjustment, and
- techniques to render hatching lines in high quality by processing the streamlines specifically with respect to print media, in particular to requirements resulting from using monochrome ink.

The paper is organized as follows. Section 2 describes related work in the field of computer generated hatching lines. In Section 3 we give an outline of our algorithm to create the lines and describe the main steps in detail. Section 4 is devoted to the actual rendering, i. e., the adaptive visualization of the lines. We discuss a number of examples in Section 5 and finally conclude the paper with some ideas for future work.

## 2. Related Work

Traditional line renditions that employ hatching are quite commonly used in arts and for illustration purposes. Hatching lines fill an area of an image by collectively conveying texture and tone. While these lines may vary in their length, they typically follow some geometric features of the object being depicted and they may be layered to produce cross-hatching. Considering hand-made drawings, hatching is not a drawing style or technique in itself, instead it is used and achieved with several different techniques. Therefore, a wealth of methods have been developed also in computer graphics to achieve the same or similar effects.

In the first of a series of papers, SALISBURY et al. show how to use *stroke textures* to convey a certain darkness for shading with pen-and-ink lines [SABS94]. When interactively drawing an illustration, the algorithms selects strokes from the stroke texture until a desired shading has been achieved. WINKENBACH and SALESIN introduce the concept of *prioritized stroke textures* [WS94]. This allows the resolution dependent placement of pre-recorded strokes to achieve the same perceived grey value. If the resolution changes, their method places more or fewer strokes until the desired tone is achieved yielding a rendition that is appropriate for the specific resolution. SALISBURY et al. further discuss the scale-dependence of pen-and-ink drawings in terms of perceived darkness of the hatched areas [SALS96]. In particular, they demonstrate how to maintain sharp discontinuities of the textures across various resolution levels.

Being the first to create hatching renditions from 3D scenes, LEISTER bases his approach on modified ray tracing [Lei94]. He uses an additional direction that is defined for each object's surface and a parameter to determine the distance between two hatches, similar to the *u*-*v* parameter field defined for texturing. Being a ray tracing adaption, his method can emulate reflections and refractions. The algorithms produces image-space results which means that they cannot easily be processed any further using stroke manipulations. In addition, the appearance of the created images is rather clean and artificial.

PNUELI and BRUCKSTEIN present their **Dig$^{\mathbf{i}}$***Dürer* system that uses greyscale images as input and creates a halftoned output image that resembles the style of engravings [PB94]. It computes level contours of a potential field and is based on a curve evolution algorithm that controls the density of line elements.

WINKENBACH and SALESIN introduce a technique for generating hatching renditions from parametric surfaces by employing isoparametric curves [WS96]. They use prioritized stroke textures and align the strokes according to these curves. They achieve a quite natural look by using long and short strokes as well as adding small alterations to the strokes. Additionally they used randomized dots on lines to stipple an area with a desired tone. SALISBURY et al. use a 2D greyscale image as input and require the user to specify a direction field as well as example strokes [SWHS97]. Their system then generates hatch line textures that reproduce the shading of the original image while conveying the impression to be attached to the surface of the object and following its features.

In a completely image-based approach, OSTROMOUKHOV presents an algorithm that uses a 2D source image and so called engraving layers—basic dither screens that are combined to form hatching specific dither patterns [Ost99]. He requires user-interaction in order to specify how these layers have to be deformed by image warping in order to follow certain features of the image. A screening process computes the final rendition, that possesses a very clean and artificial appearance.

In an object-space approach, DEUSSEN et al. use internal skeletons created from triangle meshes using progressive meshes [DHR*99]. The skeletons are used to determine a direction perpendicular to which the object is sliced. The slice curves are then used as hatching lines for the objects again producing a very clean and artificial appearance. By adding line styles and thus changing the appearance of the hatch-

ing lines based on shading or other geometric information, a more natural look can be achieved.

The technique presented by RÖSSL and KOBBELT works in image-space and also uses triangle meshes [RK00]. They first compute an approximation of curvature directions and normals for each vertex and do a linear interpolation for the values on the faces. They then render G-buffers for both normal and curvature direction vectors. Afterwards, they use streamlines for following the hatching lines in 2D. Interaction is required for specifying homogeneous parts of curvature directions as well as reference lines in the projection. Although the shading they used in the given examples could be improved, they achieve a fairly natural look of the images.

HERTZMANN and ZORIN also work in image-space and base their method on smooth surfaces given by a polygonal control mesh [HZ00]. Similar to the previously discussed approach, they also use approximated principal curvature lines in 3D that are projected into image-space. They add some preprocessing of the direction field in order to avoid artifacts in the hatching lines and also create a fairly natural look of the renditions.

There are a number of approaches that generate lines on 3D shapes for rendering based on some features of the model. For example, in [ARS79], [Elb95b], [Elb95a], and [Elb98] the generation of isoparametric lines on freeform surfaces is discussed and it is demonstrated how to enhance them with, e. g., line haloes. ELBER [Elb99] and RÖSSL et al. [RKS00] discuss how to use lines on the surface of objects to visualize vector or curvature direction fields.

INTERRANTE uses principal curvature directions for visualizations of volumetric data using lines on the surface [Int97]. In a subsequent paper [GIHL00], GIRSHICK et al. discuss the use of principal curvature directions for 3D line drawings in general. They state that there are, for example, psychological reasons for employing principal curvature lines in order to enhance shape recognition where, e. g., silhouette lines are not enough. However, the example renditions they present, both generated from volumetric and polygonal data, do not resemble what is traditionally considered to be hatching style. Also using principal curvatures for line orientation, DONG et al. apply the generation of hatching lines that are computed for volumetric data to the area of medical illustration [DCLK03]. By taking the local characteristics of the volume data into account they are able to improve the quality of the rendition.

Summarizing these findings, we come to the conclusion that we can successfully use principal directions for the generation of hatching lines. Principal directions are defined everywhere on a surface except in isolated singularities, they are not dependent on a parameterization of the surface and they are known to convey the form of an object to the viewer. Our observations have also shown that ray-tracing, semi-automatic methods with manually fitted parametric curves, and skeleton approaches produce very sterile lines. The use of principal curvatures gives the possibility to create a more hand-crafted appearance if we compare the results of the aforementioned techniques to hand-made engravings. However, a post processing of the direction field obtained from principal direction vectors is necessary in order to avoid too many distracting details and to generate a more homogeneous field.

## 3. Algorithm Overview

Our algorithm to produce high quality hatching uses triangular meshes as input. The whole process comprises two stages: a preprocessing phase where lines are computed in 3D and a rendering phase where these lines are visualized. In the first stage, we start with the generation of a direction field based on curvature information. We then process the curvature field in order to enhance its quality before 3D streamlines are generated. These streamlines are the input to the second stage where they are rendered according to the desired output device. In the following we will describe each of these steps in more detail.

### 3.1. Generation of a Curvature Field

Hatching lines, as already stated above, follow some geometric feature of an object. To achieve this, the first step in the preprocessing phase is to establish direction information for the lines. We therefore generate a direction field, consisting of a unit direction vector for each vertex of the model, laying in the appropriate tangent plane. As has been argued before, e. g., by INTERRANTE [Int97], GIRSHICK et al. [GIHL00], and HERTZMANN and ZORIN [HZ00], principal curvature values are well suited. Indeed, it has been found that in traditional illustrations hatching lines are frequently used to emphasize curvature. Therefore, we approximate the principal curvature directions using a method introduced by RÖSSL AND KOBBELT [RK99] and store these values in the direction field. For each vertex, there are two possible vectors following the maximal and minimal curvature direction $\kappa_1$ and $\kappa_2$ as can be seen in Figure 1. The curvature $\kappa_i$ with the higher absolute value (indicated by the sign of the mean curvature $\frac{1}{2}(\kappa_1 + \kappa_2)$) determines the more curved direction which will then be used. In most cases this is a good heuristic to emphasis cylindrical structures.

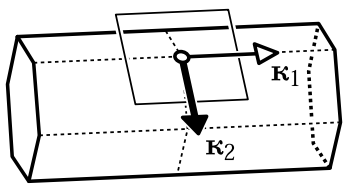



**Figure 1:** *Approximated principal curvature directions for a vertex of the polygonal model. In this case the direction of $\kappa_2$ is chosen in order to hatch around the cylinder's circumference.*

### 3.2. Processing of the Curvature Field

The quality of the resulting direction field directly depends on the underlying algorithm for curvature computation (see GOLDFEATHER [Gol01]) and on the properties of the mesh. Noise and high levels of detail easily introduce unwanted artifacts. To avoid these, a simplification algorithm can be applied to the mesh (see PRAUN et al. [PHWF01]) or the mesh can be smoothed before curvature is computed. However, no reliable curvature information can be extracted from flat or spherical surfaces. Therefore, too smooth surfaces tend to result in poorly aligned direction vectors.

Since the resulting direction field relies solely on local curvature information, is not very homogeneous. To improve this, we have gone a similar route as HERTZMANN and ZORIN [HZ00]: an energy term is defined that measures the deviation of a direction vector to the ones located on its incident edges. In contrast to HERTZMANN and ZORIN [HZ00] this is done for $180^\circ$ and not $90^\circ$ symmetries. Our algorithm only wants to wrap up the surface with evenly distributed parallel lines. Cross-hatching is then achieved by repeating the process with rotated lines (see Figure 10 for an example). This allows to generate cross-hatchings with arbitrary angles whereas Hertzmann et al. only generate orthogonally crossing lines. To achieve almost parallel lines, directions within a homogeneous neighborhood are used as a basis for fitting the other directions using a global non-linear optimization technique (cf. [HZ00]), which is applied on all regions which do not satisfy a user selectable level of homogeneity.

### 3.3. Generation of 3D Streamlines

After a smooth direction field has been computed, streamlines are generated by integrating the direction vector field on the surface of the model. In contrast to the technique suggested in [ACSD*03] that solves the problem in 2D, we employ a version of the original algorithm presented in [JL97] and adapt it for the third dimension. Therefor it is necessary to determine direction vectors not only at the individual vertices, but for every point on the surface. This is done by a spherical-linear interpolation of the respective direction vectors using the *barycentric coordinates* of a position as weights. The seeding strategy was also modified from the original paper. Since it is not always possible to reach all parts of a model from a single initial seed point, each face centroid is used as a possible seeding point. Starting from there, the algorithm tries to reach as much area as possible. While a streamline is growing through the direction vector field, new possible seed points are generated alongside. Only if none of these can be used to create a new streamline, the next face centroid is used as a new possible seeding point.

To prevent line crossings, streamlines are only integrated until they meet each other. Instead of the grid-based streamline-proximity scheme, cylinders are used to compute streamline distances and to end a streamline if it closes in on another one (see Figure 2). The cylinders are generated along the growing directions of the streamlines and are stored in all the faces they intersect. This helps to prevent streamlines from influencing each other on opposite sides of thin regions in the mesh.

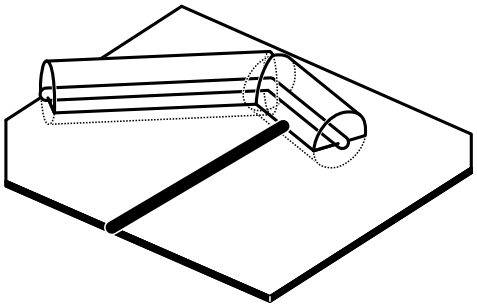


**Figure 2:** *A new streamline (black) is terminated at the cylinder surrounding a previously computed streamline (white). The endpoint of the new streamline is keept on a distance which equals the cylinder's radius.*

In order to find the distance, a lookup is done for all cylinders that are stored in the according face and the nearest intersection in the growing direction of a streamline is computed. As long as the size of the faces is less or roughly equal to the intended distance between two streamlines, this is very fast because, in average, there is only one cylinder per face. The computational overhead only grows for large faces. Using cylinders has another advantage. Streamlines are allowed to come very close with their tips, removing wide gaps that otherwise tend to occur (see Figure 3).

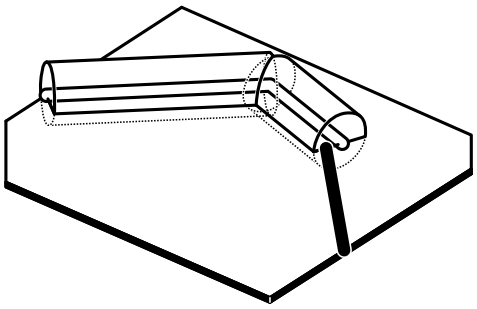


**Figure 3:** *The use of cylinders allows a new stramline (black) to come closer to an existing streamline (white) than the cylinder's radius, if it approaches its tip.*

A property that is needed later is that the streamlines have to follow the surface of the mesh closely. So while integrating the direction field, new points are inserted each time an edge is crossed, moving from one face to the next. This ensures that even with a wide step size, streamlines will not poke through the surface. The quality of the integrator also needs further observation. First, we used a very simplistic

Euler integrator, but changing to a fourth order Runge-Kutta allowed a wider step-size. This drops the number of segments that are needed to represent the streamlines. Nonetheless, the quality continues to be very high, resulting in decreased time needed for the preprocessing stage.

### 3.4. Line Tapering

In the final phase of the preprocessing stage data for line tapering is gathered. The process differs slightly from [JL97] since we use a different distance metric for streamline collisions. In our implementation it is possible that streamlines can get very close with their tips. This effect should not be impaired by line tapering. So closeness of nearby streamlines is not measured omni-directional, but parallel to the direction of the streamline (see Figure 4). This aligns with the streamline generation process using cylinders as described above. For each point on a streamline, first the local average direction is calculated. This is then used in combination with the normal of the face below the point to construct a plane orthogonal to that direction. To measure the distance of nearby streamlines the intersection points of all nearby streamlines with this plane are computed and the minimal distance is stored. Finally, this minimal distance is employed to calculate the tapering value that is needed in the rendering stage as a multiplicative modulator for the line width.

**Figure 4:** *Computing streamline distances omni-directionally results in gaps between the tips of nearby streamlines as can be seen on the left. Using our approach these gaps are minimized resulting in the image on the right.*

## 4. Rendering in High Quality

After the preprocessing stage has been completed, the second phase begins—visualizing the streamlines as hatching lines. The major goal in this step is to render the streamlines with high quality in order to be able to use the generated images, for example, for reproduction in print media. Therefore, three goals have to be accomplished:

- a fast and effective line processing including hidden line removal (HLR) that removes the silhouette strokes and streamlines not visible from the given viewpoint while still offering interactive frame-rates,
- an appropriate NPR line shading for the hatching strokes in order to produce a smooth transition between fully drawn and not drawn lines according to the specific lighting condition, and
- a high quality line output that produces vector-oriented data for reproduction including, for example, the transformation of a shaded line into a monochrome representation.

### 4.1. HLR and Line Processing

The generated streamlines are inserted into a stroke rendering pipeline that first uses a hybrid hidden line removal algorithm (we employ the $z$-buffer based method presented by ISENBERG et al. [IHS02]) in order to remove the occluded parts. The mentioned HLR algorithm was originally conceived to clip object silhouettes, but it works very well under these new requirements. There are only minor artifacts at the object's silhouette due to $z$-buffer imprecisions as shown in Figure 5. This can be resolved by simply removing all strokes on backfaces. A welcome side-effect of this procedure was a noticeable frame-rate improvement because the backfacing test is cheaper than feeding segments through the HLR algorithm.

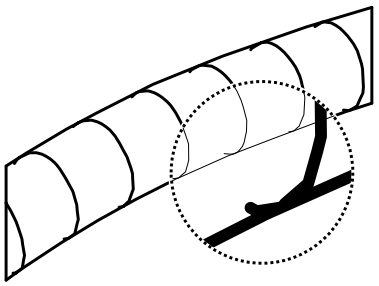


**Figure 5:** *If only the z-buffer based HLR scheme is used, small tickmarks will appear near the silhouette. These have to be removed.*

Using a hybrid HLR scheme also enables us to achieve interactive frame-rates when rendering the images. This results from computing the streamlines in the preprocessing stage off-line and only performing the HLR test at run-time. Being able to render the images at interactive rates is very important for illustration designers to directly see the effects of changes they made.

In addition, we do not use advanced line stylization with the generated hatching and silhouette lines. In general, for producing hatching renditions it is not necessary to follow a style based approach as discussed by various authors (see, e. g., NORTHRUP and MARKOSIAN [NM00] or ISENBERG et al. [IHS02]). In contrast, when examining cross-hatched images created by artists one finds that they only consist of monochrome lines. Thus, advanced stylization that introduces structure to the lines (as, e. g., by using textures or

applying path variations) is not necessary in this case. This has many advantages, for example, faster rendering and less artifacts. In particular, we achieve frame-coherency since the hatching lines are attached to the surface of the objects. In addition, there are no artifacts created by the stylization when it comes to generating high quality output.

### 4.2. Line Shading

In order to limit the number of lines that are drawn and to convey a specific illumination condition it is now necessary to apply line shading. We opted for a rich NPR centric shading model that features effects like rim shading or curvature lighting (see, e. g., STROTHOTTE and SCHLECHTWEG [SS02]). To provide more flexibility to the user for adapting this shading model and to allow easier experiments with different effects, it was not implemented as a "hardwired" formula. Instead, it is generated by a virtual machine (bytecode interpreter). This is a very elegant way of allowing the user to make changes to the expressions and by that to control the two main parameters: stroke width and stroke density. These changes are possible at run-time and avoid tedious recompiles. At run-time, the formulas are being compiled into byte code that is executed by a small stack-based interpreter in order to obtain the shading values. This is achieved by parsing the expression and breaking it down into constants, symbols, and operators. The whole expression is converted into *Reverse Polish Notation* in order to simplify the evaluation of the term because in this representation brackets and operator precedence are of no importance. Later, the term is evaluated for each vertex of a streamline and symbols are substituted with their associated value which changes depending on view direction or local surface orientation. The speed impact of the interpretation is negligible.

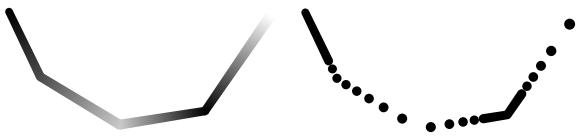
### 4.3. Line Output

When closely examining real hatched drawings, we found that monochrome lines are used with the two properties: variable width and the possibility to dissolve solid lines into dots. Therefore, we have two parameters—line width and line density. Because speed is not an issue when creating output for high quality reproduction, we made sure that line caps and bends are round. Another noteworthy feature is the possibility to use negative values for the line width. This helps to seamlessly blend out strokes because visible line tips can now appear in the middle of a stroke segment. This decouples the visual occurrence of a line end from the positions of the individual stroke vertices. This means that a line may now also terminate in the middle of a stroke segment (see lower two examples in Figure 6).

The use of dotted lines in hand-made hatching is caused by the limitation to monochrome primitives. In addition, it is also in part caused by restrictions of reproduction techniques that can only handle monochrome ink. When trying

**Figure 6:** *Decreasing the width of the line first results in tapering and later also in shortening the visible portion when the width gets negative.*

to reproduce grey values, these would be dithered into black and white pixels which would destroy the appearance of a hatching (see Figure 7(a)). In order to avoid this to occur for shaded hatching lines, we also represent line density by a form of one-dimensional stippling (see Figure 7(b)). This one-dimensional line stippling is, therefore, not intended or used to simulate regular two-dimensional stippling as done in previous approaches. Instead, we use it as an alternative to change the tone of a line, which does not rely on line width. To accomplish this the line is broken down into a string of dots that are placed to locally approximate the required density of the specific stroke. The distance of two dots is chosen in a way to match the ratio of black and white space inside the quadrilateral spanned through their centers to the average density value for that region. This can be further controlled by specifying a minimum distance between dots. In order to draw segments in full color where the computed distance would lead to distances smaller than desired, the segment is subdivided at places where exactly this minimum distance is reached.



(a) Desired density of the stroke.

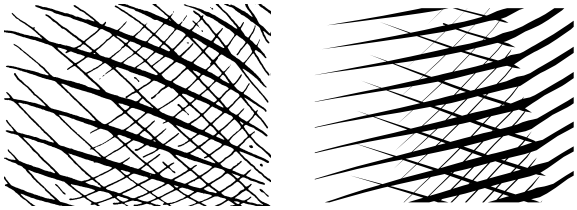
(b) Approximation of the line density using dots.

**Figure 7:** *Achieving a desired line density using line halftoning using one-dimensional stippling along the stroke.*

We implemented two specific renderers with respect to the two different goals for the images to be created. The first is able to output lines and arcs to create the final rendition as PDF files for high resolution and high quality reproduction. The second is for WYSIWYG-purposes and supports designers that want to interactively create hatching illustrations. This second renderer triangulates the strokes into triangle strips that can afterwards be rendered at interactive frame-rates using OpenGL.

In addition to producing single hatched images, it is also possible to combine multiple rendering passes. This is frequently used in hand-made images as shown in Figure 8(a).

It is easily possible to generate cross-hatching effects by rotating the whole direction field around arbitrary angles and compositing several layers of differently shaded strokes (see Figure 8(b)).

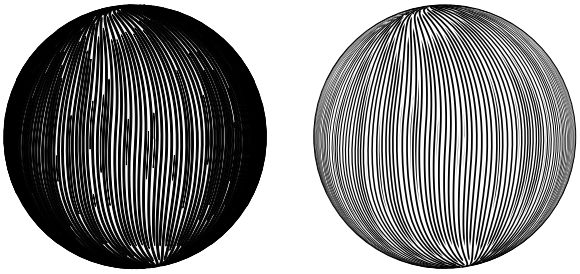


(a) Closeup of a real artistic rendition (from [Hod89]).

(b) Three layers of strokes resulting from a rotated direction field.

**Figure 8:** *Comparison of hand-made hatching with a magnified section of a computer-generated image (Figure 8(b) is a detail from Figure 10(b)).*

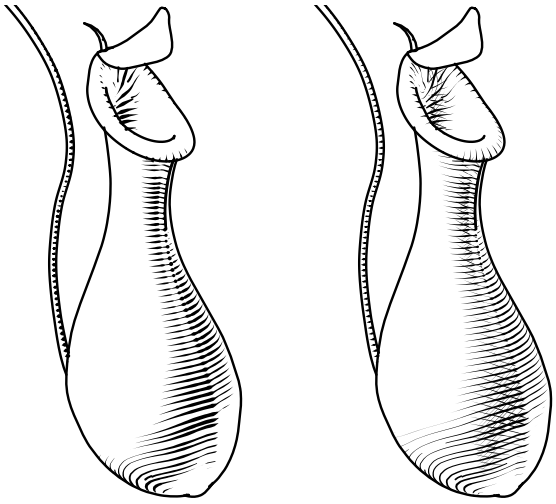
As we have outlined above, the hatching lines are generated on the object's surface and, thus, in object-space. A disadvantage of working in object-space that has to be considered for high-quality rendering is that shading not only depends on the width of the lines but also on their distance to each other after the projection to screen space. If the lines get closer to each other, the perceived brightness of these regions will be darker (see Figure 9(a)). This is particularly noticeable close to the silhouette or if the scene is scaled. Solutions have been thoroughly discussed in, for example, [SABS94], [WS94], and [SALS96]. We determine a correction factor for each vertex of a stroke. Specifically, a normalized vector perpendicular to the line direction and the surface normal is computed using the cross product. This is a local estimate of the distance to imaginary nearby hatching lines. The length of the projection of this vector onto the viewplane is measured and used as the correction factor for the line width. It reduces the unwanted effect of differences in the perceived brightness of strokes with different distances to each other (see Figure 9(b)). Therefore, the line shading can now be determined depending entirely on the parameterized NPR shading model.



(a) Without a correction factor.

(b) With the correction factor.

**Figure 9:** *Use of a correction factor that compensates differences in the perceived brightness. Unfortunately, the printing process may weaken the correction due to, for example, a minimum possible line width or effects related to dot gain.*



(a) Single hatching.

(b) Triple hatching.

**Figure 10:** *Single hatching and triple hatching of the model of a tropical pitcher plant's trap. Note that the hatching is able to effectively convey the small ridge that runs at the front side of the pitcher. This is done by varying line width or by employing triple hatching.*

## 5. Examples

Now we will give a number of specific examples along with descriptions of how the effects in the images were achieved.

A first example compares single and multiply hatched illustrations of the same object. Figure 10(a) shows the illustration of a tropical pitcher plant's trap with only single hatching applied. Note that small details such as the small ridge that runs at the front side of the pitcher are still clearly visible because they are emphasized through the line shading model. In the second version of the same object in Figure 10(b), three layers of hatching have been combined and emphasize the details even more.

Instead of just using black and white, background and foreground colors can be changed in order to create a composed rendition. Figure 11 shows an example where several layers of hatching in different colors have been composited along with certain background colors. This method can easily be applied to create illustrations for use in colored printing. If spot colors are used, no unwanted dithering artifacts are created when the rendition is reproduced and printing cost can often be lowered in contrast to the usual four color printing process.

Figure 12 demonstrates that line stippling alone can be used to convey shading information. Therefore, the line

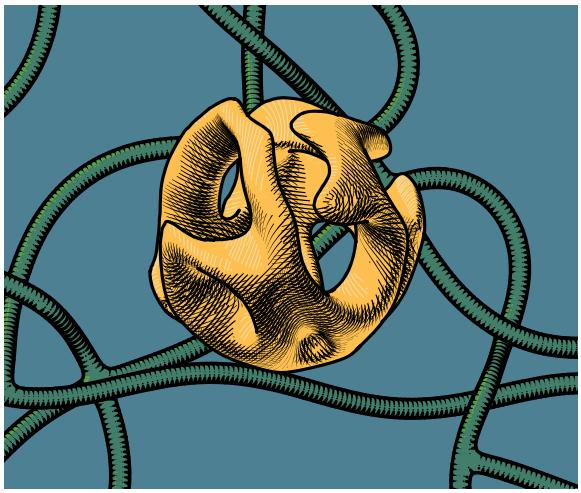


**Figure 11:** *Compositing a number of hatching layers in different colors and background colors in order to create a multi-color illustration (five colors and black).*

width has been set to constant in the example. This way not only shading information is transported but also the form of the individual leaves is accentuated, which would be not as clear with pure two-dimensional stippling.

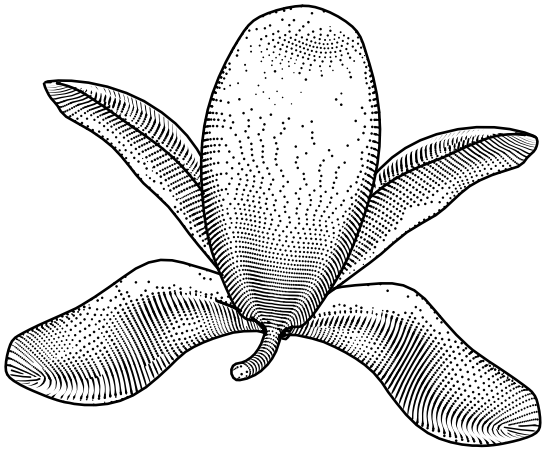


**Figure 12:** *Orchid-backside with quasi constant line width. Shading has been achieved by varying line density.*

Figure 13 demonstrates the effect of varying line widths. In Figure 13(a), the line width is used to convey shading information and the line ends are rounded without using separate dots. In contrast, Figure 13(b) shows the use of sharp line tips while the individual lines are placed closer to each other. Making use of cross-hatching, the statue in Figure 13(c) is rendered using lines that are dissolved into individual dots in order to show the shading effects. Finally, Figure 13(d) makes use of fairly wide lines that overlap each other. This produces a very sharp contrast between shaded and lit areas.

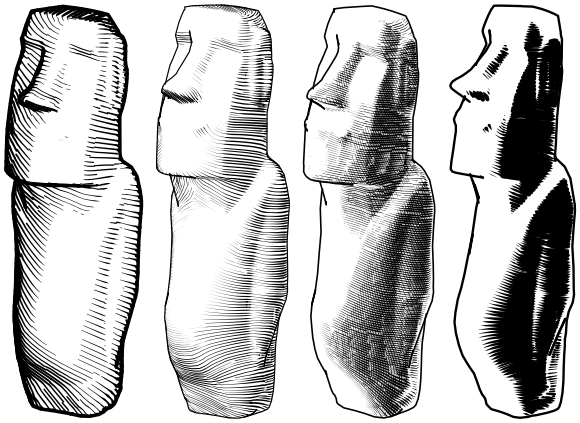


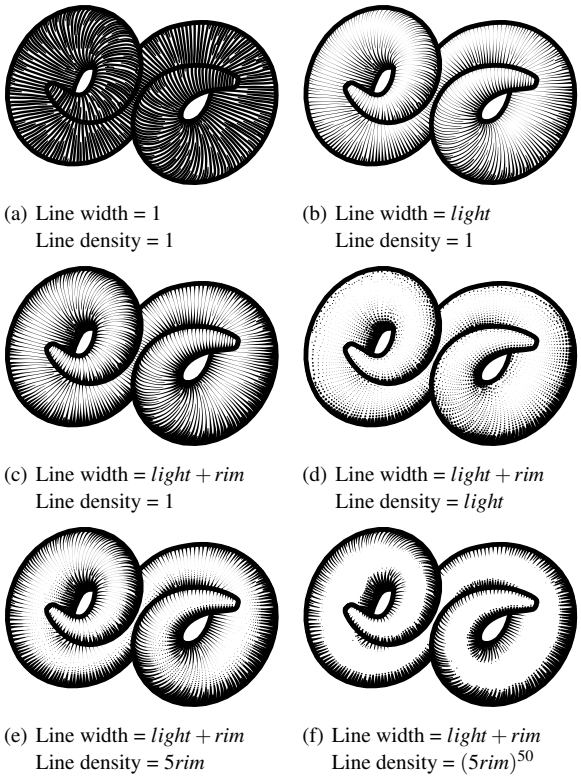

**Figure 13:** *Different effects that can be achieved by varying the number and the distance between neighboring hatching lines.*

Figure 14 illustrates how the appearance of an object changes while the user adapts the two shading formulas at run-time. In Figure 14(a) the raw strokes can be seen with constant terms for line width and density. This results in a quasi-constant tone. The use of a single light source as a modulator for the linewidth adds depth as seen in Figure 14(b). To enhance the edges, rim shadow lighting has been added in Figure 14(c). Using the light to also modulate the line density as seen in Figure 14(d) creates a fairly contrastless halftoning-like effect. If the density term is further changed this can be constrained to a rather small band (Figure 14(e)). The addition increase of the contrast may be used to eliminate lines and as a means to place highlights (Figure 14(f)).

## 6. Conclusion

In this paper we have presented a hatching method for generating high quality vector-oriented images mainly aimed at reproduction in print media. Our technique computes the hatching lines in object-space in a pre-processing step which gets rid of frame-incoherences and the shower-door effect as well as allows for fast rendering of the lines at run-time. Therefore, we can offer interactive frame-rates for designing the hatching illustration using a hybrid HLR technique. We have shown that the illumination model can be modified at run-time giving the designer a great freedom of expression. Using the examples shown in the paper, we demonstrated that this allows a wide range of different effects that can be achieved by parameterizing this model.

We have demonstrated how to solve a variety of issues that may occur when attempting high-quality reproduction



(a) Line width = 1
Line density = 1

(b) Line width = *light*
Line density = 1

(c) Line width = $light + rim$
Line density = 1

(d) Line width = $light + rim$
Line density = *light*

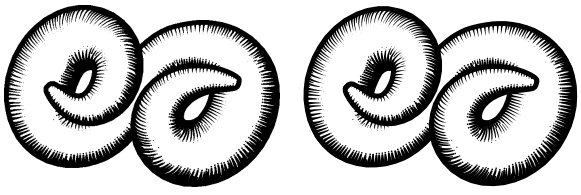
(e) Line width = $light + rim$
Line density = $5rim$

(f) Line width = $light + rim$
Line density = $(5rim)^{50}$

**Figure 14:** *The process of altering the two line shading formulas until a desirable result is achieved.*

in print. Those are, in particular, the possibility to use negative line widths to allow for a more flexible line tapering, the use of dotted lines in order to avoid the dithering effects from line shading, the removal of small artifacts that result from HLR, and employing a correction factor for achieving an approximately equal perceived grey value across the object to account for different viewing angles on the surface.

In addition, we provide two renderers for these two goals for the presentation of the hatching illustration. The first is designed for interactive on-screen display (see Figure 15(a)) while the second aims at generating the high-quality renditions for reproduction in print media (see Figure 15(b)).



(a) On-screen rendition.

(b) Vector graphic output.

**Figure 15:** *The two output media used compared on the basis of Figure 14(f).*

For future work we plan to further improve the computation of the streamlines in order to create a more esthetic line placement. An hierarchic construction-order should get rid of some of the more striking placement artifacts and at the same time keep the uniformity of line distances after projecting to screen-space. In addition, a downside of the on-screen renderer is the lack of anti-aliasing, which is a serious disadvantage because line art needs good anti-aliasing for lower resolutions. However, on newer graphics hardware it is possible to use full scene anti-aliasing but the resulting frame-rate drop is severe in our implementation. This will be improved in future versions of the software so that the on-screen version better resembles the printed version of the line drawing. We also envision an easier and more intuitive interface to adjust the settings of the illumination model so that non-technical people (e. g., designers) can easier adjust the parameters that influence the appearance of the rendering. Improvements of the interface will also include possibilities for creating composed drawings in one step rather than compositing them afterwards. Finally, we want to integrate ways that let the system suggest additional cross-hatching directions automatically rather than specifying them manually.

**Acknowledgments**

We would like to thank Petra Neumann for the model of the tropical pitcher plant's trap in Figure 10. Specifically, the model represents the upper trap of a *Nepenthes alata* plant.

**References**

[ACSD*03] ALLIEZ P., COHEN-STEINER D., DEVILLERS O., LÉVY B., DESBRUN M.: Anisotropic Polygonal Remeshing. *ACM Transactions on Graphics 22*, 3 (July 2003), 485–493.

[ARS79] APPEL A., ROHLF F. J., STEIN A. J.: The Haloed Line Effect for Hidden Line Elimination. In *Proceedings of SIGGRAPH'79* (New York, 1979), ACM Press, pp. 151–157.

[DCLK03] DONG F., CLAPWORTHY G. J., LIN H., KROKOS M. A.: Nonphotorealistic Rendering of Medical Volume Data. *IEEE Computer Graphics and Applications 23*, 4 (July/Aug. 2003), 44–52.

[DHR*99] DEUSSEN O., HAMEL J., RAAB A., SCHLECHTWEG S., STROTHOTTE T.: An Illustration Technique Using Hardware-Based Intersections and Skeletons. In *Proceedings of Graphics Interface'99* (1999), Morgan Kaufmann Publishers Inc., pp. 175–182.

[Elb95a] ELBER G.: Line Art Rendering via a Coverage of Isoparametric Curves. *IEEE Transactions on Visualization and Computer Graphics 1*, 3 (Sept. 1995), 231–239.

[Elb95b] ELBER G.: Line Illustrations ∈ Computer Graphics. *The Visual Computer 11*, 6 (June 1995), 290–296.

[Elb98] ELBER G.: Line Art Illustrations of Parametric and Implicit Forms. *IEEE Transactions on Visualization and Computer Graphics 4*, 1 (Jan. 1998), 71–81.

[Elb99] ELBER G.: Interactive Line Art Rendering of Freeform Surfaces. *Computer Graphics Forum 18*, 3 (Sept. 1999), 1–12.

[GIHL00] GIRSHICK A., INTERRANTE V., HAKER S., LEMOINE T.: Line Direction Matters: An Argument for the Use of Principal Directions in 3D Line Drawings. In *Proceedings of NPAR 2000* (New York, 2000), ACM Press, pp. 43–52.

[Gol01] GOLDFEATHER J.: *Understanding Errors in Approximating Principal Direction Vectors*. Tech. Rep. 01-006, University of Minnesota – Computer Science and Engineering, 2001.

[Hod89] HODGES E. R. S. (Ed.): *The Guild Handbook of Scientific Illustration*. Van Nostrand Reinhold, New York, 1989.

[HZ00] HERTZMANN A., ZORIN D.: Illustrating Smooth Surfaces. In *Proceedings of SIGGRAPH 2000* (New York, 2000), ACM Press, pp. 517–526.

[IHS02] ISENBERG T., HALPER N., STROTHOTTE T.: Stylizing Silhouettes at Interactive Rates: From Silhouette Edges to Silhouette Strokes. *Computer Graphics Forum 21*, 3 (Sept. 2002), 249–258.

[Int97] INTERRANTE V.: Illustrating Surface Shape in Volume Data Via Principal Direction-Driven 3D Line Integral Convolution. In *Proceedings of SIGGRAPH'97* (New York, 1997), ACM Press, pp. 109–116.

[JL97] JOBARD B., LEFER W.: Creating Evenly-Spaced Streamlines of Arbitrary Density. In *Proceedings of the 8th Eurographics Workshop on Visualization in Scientific Computing* (1997), pp. 45–55.

[Lei94] LEISTER W.: Computer Generated Copper Plates. *Computer Graphics Forum 13*, 1 (Mar. 1994), 69–77.

[NM00] NORTHRUP J. D., MARKOSIAN L.: Artistic Silhouettes: A Hybrid Approach. In *Proceedings of NPAR 2000* (New York, 2000), ACM Press, pp. 31–37.

[Ost99] OSTROMOUKHOV V.: Digital Facial Engraving. In *Proceedings of SIGGRAPH'99* (New York, 1999), ACM Press, pp. 417–424.

[PB94] PNUELI Y., BRUCKSTEIN A. M.: **Dig$^{\mathbf{i}}$D**ürer – A Digital Engraving System. *The Visual Computer 10*, 5 (Apr. 1994), 277–292.

[PHWF01] PRAUN E., HOPPE H., WEBB M., FINKELSTEIN A.: Real-Time Hatching. In *Proceedings of SIGGRAPH 2001* (New York, 2001), ACM Press, pp. 581–586.

[RK99] RÖSSL C., KOBBELT L.: Approximation and Visualization of Discrete Curvature on Triangulated Surfaces. In *Vision, Modeling, and Visualization (VMV) '99 Proceedings* (St. Augustin, Germany, 1999), infix, pp. 339–346.

[RK00] RÖSSL C., KOBBELT L.: Line-Art Rendering of 3D-Models. In *Proceedings of Pacific Graphics 2000* (Los Alamitos, CA, 2000), IEEE Computer Society Press, pp. 87–96.

[RKS00] RÖSSL C., KOBBELT L., SEIDEL H.-P.: Line Art Rendering of Triangulated Surfaces Using Discrete Lines of Curvature. In *Proceedings of WSCG'2000* (2000), The University of West Bohemia, Plzeň, Czech Republic, pp. 168–175.

[SABS94] SALISBURY M. P., ANDERSON S. E., BARZEL R., SALESIN D. H.: Interactive Pen-and-Ink Illustration. In *Proceedings of SIGGRAPH'94* (New York, 1994), ACM Press, pp. 101–108.

[SALS96] SALISBURY M. P., ANDERSON C., LISCHINSKI D., SALISIN D. H.: Scale-Dependent Reproduction of Pen-and-Ink Illustration. In *Proceedings of SIGGRAPH'96* (New York, 1996), ACM Press, pp. 461–468.

[SS02] STROTHOTTE T., SCHLECHTWEG S.: *Non-Photorealistic Computer Graphics: Modeling, Rendering, and Animation*. Morgan Kaufmann Publishers, San Francisco, 2002.

[SWHS97] SALISBURY M. P., WONG M. T., HUGHES J. F., SALESIN D. H.: Orientable Textures for Image-Based Pen-and-Ink Illustration. In *Proceedings of SIGGRAPH'97* (New York, 1997), ACM Press, pp. 401–406.

[WS94] WINKENBACH G., SALESIN D. H.: Computer-Generated Pen-and-Ink Illustration. In *Proceedings of SIGGRAPH'94* (New York, 1994), ACM Press, pp. 91–100.

[WS96] WINKENBACH G., SALESIN D. H.: Rendering Parametric Surfaces in Pen and Ink. In *Proceedings of SIGGRAPH'96* (New York, 1996), ACM Press, pp. 469–476.

# E

## RENDERBOTS – MULTI AGENT SYSTEMS FOR DIRECT IMAGE GENERATION

Bibliographische Angaben:

Stefan Schlechtweg, Tobias Germer und Thomas Strothotte: „RenderBots—Multi Agent Systems for Direct Image Generation“. *Computer Graphics Forum*, 24(2):(2005), S. 137–148.

Abstract:

The term stroke-based rendering collectively describes techniques where images are generated from elements that are usually larger than a pixel. These techniques lend themselves well for rendering artistic styles such as stippling and hatching. This paper presents a novel approach for stroke-based rendering that exploits multi-agent systems. RenderBots are individual agents each of which in general represents one stroke. They form a multi-agent system and undergo a simulation to distribute themselves in the environment. The environment consists of a source image and possibly additional G-buffers. The final image is created when the simulation is finished by having each RenderBot execute its painting function. RenderBot classes differ in their physical behavior as well as their way of painting so that different styles can be created in a very flexible way.

Betreute Diplomarbeit:

Tobias Germer: *RenderBots: Multiagentensysteme für NPR-Graphiken*. Diplomarbeit am Institut für Simulation and Graphik der Otto-von-Guericke-Universität Magdeburg, 2004.

Weitere Veröffentlichung: [GS05]

Tobias Germer und Stefan Schlechtweg: „RenderBots—Multi-Agent Systems for NPR“. *NORSIGD Info, medlemsblad for NORSIGD*, 2005(1):(2005), S. 4–5.

Volume 24 (2005), number 2 pp. 137–148 COMPUTER GRAPHICS *forum*

# RenderBots—Multi-Agent Systems for Direct Image Generation

Stefan Schlechtweg, Tobias Germer and Thomas Strothotte

Department of Simulation and Graphics, Otto-von-Guericke University of Magdeburg, Universitätsplatz 2, D-39106 Magdeburg, Germany
stefans@isg.cs.uni-magdeburg.de, germer@web.de, tstr@isg.cs.uni-magdeburg.de

**Abstract**

*The term* stroke-based rendering *collectively describes techniques where images are generated from elements that are usually larger than a pixel. These techniques lend themselves well for rendering artistic styles such as stippling and hatching. This paper presents a novel approach for stroke-based rendering that exploits multi-agent systems. RenderBots are individual agents each of which in general represents one stroke. They form a multi-agent system and undergo a simulation to distribute themselves in the environment. The environment consists of a source image and possibly additional G-buffers. The final image is created when the simulation is finished by having each RenderBot execute its painting function. RenderBot classes differ in their physical behavior as well as their way of painting so that different styles can be created in a very flexible way.*

**Keywords:** non-photorealistic rendering, multi-agent systems, stroke-based rendering, behavioral simulation.

**ACM CCS:** I.3.3 [Computer Graphics]: Line and curve generation, I.2.11 [Distributed Artificial Intelligence]: Multiagent systems.

## 1. Introduction

The development of non-photorealistic rendering has brought up a wealth of new image generation methods. Rendering is no longer only the process of projecting a geometry into two dimensions and calculating the light intensity for each pixel position. Many techniques, for example, take a source image and re-render it using picture elements that are larger than a pixel (strokes). These so called *stroke-based rendering* (SBR) algorithms are well suited to create images that embody qualities of hand-made images, be this their look or their use for communication purposes.

The main problem to be solved in stroke-based rendering is the distribution of the strokes according to the properties of the source image while at the same time adhering to rules that characterize the style that should be mimicked. In this paper, we present an approach for stroke-based rendering that uses multi-agent systems to distribute strokes of various kinds over an image and to adjust strokes locally. Relatively simple autonomous agents, so called *RenderBots* are simulated within an artificial environment, which consists of the source image and possibly additional information in the form of G-buffers [1]. Different RenderBots are responsible for different rendering styles, so that on the one hand, images can be created in one distinctive style but, on the other hand, styles can also be mixed where appropriate.

The contribution of this paper is a unified approach for SBR. Different styles such as stippling, hatching, painterly rendering and mosaics can be created using the same framework. This is achieved by providing a specific class of RenderBots for each style. The environment and main control algorithms for the actual image generation remain the same. This way, our approach can easily be extended to new styles simply by implementing a respective RenderBot class. Also, existing styles, i.e. existing RenderBot classes can easily be extended and parameterized. Moreover, various styles can also be mixed by having several classes of RenderBots in the environment at the same time. The user can interactively control the image generation by changing the environment, the parameters of the RenderBots or the number of agents being used.



Submitted July 2004
Revised December 2004
Accepted January 2005

The remainder of this paper is organized as follows. Section 2, deals with related work in the context of stroke-based rendering and gives an overview of multi-agent systems. In Section 3, we introduce the general framework by describing the components used in our approach. This is followed by a closer explanation of some kinds of RenderBots in Section 4. To illustrate the results, Section 5 gives some examples before we conclude the paper.

## 2. Related Work

This paper introduces a new stroke-based rendering approach that uses multi-agent systems. Therefore, both areas are presented in the following. We start with stroke-based rendering to set the stage and we will then provide a short overview of multi-agent systems.

### 2.1. Stroke-Based Rendering

Many non-photorealistic image generation techniques rely on the distribution of artifacts over the image area. These artifacts are discrete elements and collectively convey tone, color and texture. In contrast to photorealistic rendering techniques, their size is usually larger than a pixel. Depending on the style to achieve, such artifacts may be stipples, hatching strokes, paintbrush strokes, etc. Hertzmann [2] summarizes all such techniques that create images by automatically placing discrete elements as SBR. Very often, SBR algorithms are optimization algorithms that try to find the optimal stroke placement by minimizing an objective function.

For hatching or paintbrush strokes, the most simple approach uses a *difference image algorithm* as introduced by Salisbury *et al.* [3]. The stroke is placed at an appropriate position that is derived as the position with the maximum value in a difference image where the value at each pixel is the difference between the source image and a blurred version of the illustration. A blurred version of the stroke is then subtracted from the difference image to adjust the difference values. This algorithm continues until the difference image is almost empty, i.e. there is almost no difference between the source and the target image. This algorithm is used for various image generation techniques: hatching as in [3, 4], graftals as in [5].

Instead of creating and evaluating a difference image, some methods work on a local level. For each stroke, a proposal is made. If the proposed changes do not yield an improvement, then they are discarded, otherwise the changes are performed in the image. This procedure is applied by Haeberli in his seminal work on SBR [6]. Hertzmann [7] derived a method for painterly rendering as well as Litwinowicz [8] for achieving impressionist effects in images and video.

Stippling as another technique in the realm of SBR is characterized by the placement of single dots in a random but uniform way where the density of the dots creates the tone in the final image. The optimization problem here consists of two parts: a tonal constraint that matches the density of the stipples with the tone of the source image and geometric constraints that achieve a uniform distribution of stipples while at the same time preventing stipples from overlapping each other. The most often used approach here exploits (Centroidal) Voronoi diagrams (computed using Lloyd's method [9] or variations thereof) as in [10, 11]. Due to the properties of Voronoi diagrams, they lend themselves well for the distribution of point-like strokes, even though Hiller *et al.* [12] have developed a method to distribute arbitrary shapes.

Voronoi diagrams are also used to render traditional mosaics as in [13]. Hausner starts with a Voronoi diagram from a set of points in the image. Each Voronoi region represents one mosaic tile. Then, a centroidal Voronoi diagram (CVD) is computed because it ensures that the regions cover the image plane fairly. Since square tiles are desired, the CVD is computed using a Manhattan distance metric. Within the process of computing the CVD some additional constraints, such as the orientation of the tiles, are evaluated. Other approaches to create mosaics, such as in [14, 15] use Voronoi diagrams as intermediate steps or construction aids.

Besides these approaches, there are also other ways of creating stroke-based illustrations that will not be covered in detail here. Among these are methods that place the stroke particles in 3D, apply the viewing transform and place strokes at the resulting image positions. This has been used by Meier [16] to create painterly animations, or by Meruvia *et al.* [17] to produce stippled animations. One advantage of this approach is its usability for animation since frame-to-frame coherence is automatically achieved. Furthermore, there are interactive methods for stroke placement that actually turn the computer into a painting and illustration tool as presented, for example, by Kalnins *et al.* [18].

In conclusion, for automatic SBR, many algorithms exist that differ largely from each other. On the one hand, there are very specific algorithms for certain types of images. On the other hand, energy optimization methods seem to be a unifying approach. Nevertheless, no single technique exists so far that is able to create different stroke-based styles using the same basic algorithm. In this paper, we introduce such a technique that is built on multi-agent systems and that is capable to generate a wide variety of styles.

### 2.2. Multi-Agent Systems

Our idea for SBR is to use a self-organizing system that distributes the strokes over the image area. Each stroke has the ability to change its position and state autonomously depending on the environment conditions it comes across and also depending on neighboring strokes. Thus, an optimization is performed on a local level without an explicit knowledge of the global state of the system.

The basis for our work can be found in particle systems or, more specifically, multi-agent systems. Particle systems have been introduced by Reeves [19] to model smoke, fire, explosions and other 'fuzzy' objects. Since then they have become a standard for simulation approaches in computer graphics and especially animation. They have also been used in the area of non-photorealistic rendering, for example to simulate animated paintings [16], or to render loose and sketchy line drawings [20]. Particle systems are an example of stochastic procedural modeling, however, they have their limitations when used for image generation in the sense of SBR. The dynamics of the whole system has to be described as a set of rules, as, for instance, force fields or any similar means. To gain more flexibility in stroke placement, the strokes should be self-organizing elements—agents.

The concept of agents comes from the field of artificial intelligence. An agent is an autonomously acting entity 'living' in an artificial world and having its own behavior [21]. Agents are independent from each other and are capable of acting on their own. The following properties of agents make them useful for our approach [22]. An agent is able to

- act autonomously,
- interact and communicate with other agents,
- sense its environment and react to this sensations,
- act without any sensory input,
- adapt to its environment,
- change its position in the artificial world.

The power of such autonomous entities—or agents—is revealed when many of them act together in so called *multi-agent systems* (MAS). An MAS consists of a set of discrete entities (agents) that share an environment. In contrast to particle systems, the entities here are capable of acting autonomously. The properties of agents lead to the development of the behavior of the complete MAS. This behavior may become very complex and will possibly exceed the sum of the behaviors of its component agents. In this sense, MAS show a form of emergence.

In computer graphics and animation, the use of MAS is similar to particle systems for modeling and animating large collections of objects. In order to model the complex structure of swarms of birds or herds of animals, Reynolds [23] used agents (so called *boids*) to model the individuals. Based on real world observations, each agent is designed to emulate the behavior of one animal. Based on its position and velocity and the sensation of the positions and velocities of nearby agents, each agent computes a new velocity vector and moves accordingly. Therefore, swarming behavior is achieved.

Liu describes several aspects of autonomous agents using examples from image processing, as there are edge detection and image segmentation [24]. These agents move on a pixel grid and are capable of 'sensing' brightness, contrast and other information. While moving over the image, marking positions, cloning and dying, image segmentation can be performed. Adding memory to the agents enables them to follow edges by remembering which pixels have been visited. Liu shows that reproduction and inheritance are concepts that improve the performance of an MAS. Other application areas for MAS include the simulation of virtual life, for example in the form of realistic fish swarms [25] or crowds of humans [26].

## 3. General Framework

Our goal is to use a MAS for direct image generation. The agents—so called *RenderBots*—are responsible for creating strokes in the sense of SBR, i.e. artifacts larger than a pixel that together convey color, tone and texture. Depending on the style of the rendition that should be achieved, different kinds of strokes need to be used and, therefore, different kinds of agents are needed. The source and target images form the environment in which the agents 'live'. Additional buffers, either derived from the source image, or G-buffers (derived within a conventional rendering process, cf. [1]) can be present and allow for a richer set of 'sensations' for the agents. In the following, we shall describe the general framework before we go into details for different types of agents and, hence, different kinds of strokes.

As already stated, we concentrate on two-dimensional environments, i.e. images. Additional, three-dimensional information can be supplied using G-buffers. The *universe* therefore consists of a set of image layers, each of which is accessible by a unique ID (see Figure 1). We are working with pixel images so that the triple *(ID, x, y)* unambiguously identifies a pixel in one of the image layers. The main layer which is always part of the universe is the source image, i.e. a photograph or a rendered image that should be reproduced as a stroke-based rendition (*A* in Figure 1). Additional layers give pre-computed data either from 2D or 3D. These can be, for example, object IDs, depth information (z-buffer), edge information, gradient information, etc. These additional layers need to be computed in a pre-processing step. Since we propose an image-space technique, the original three-dimensional information which is represented in these layers is not available to the RenderBots at the time of image generation. However, edge information could be computed by the RenderBots using the approach presented, for example, by Liu [24]. The problem here is that if we rely on edges from the rendered image, these edges do not represent geometric discontinuities but shading discontinuities which is not what we primarily need. Nonetheless, these additional G-buffer layers are not essential for most RenderBot classes. They enrich the possible behavior and, therefore, the expressive capabilities of the system.

While the additional layers (*C* through *H* in Figure 1) may or may not be present, the target buffer (*B*) is the second

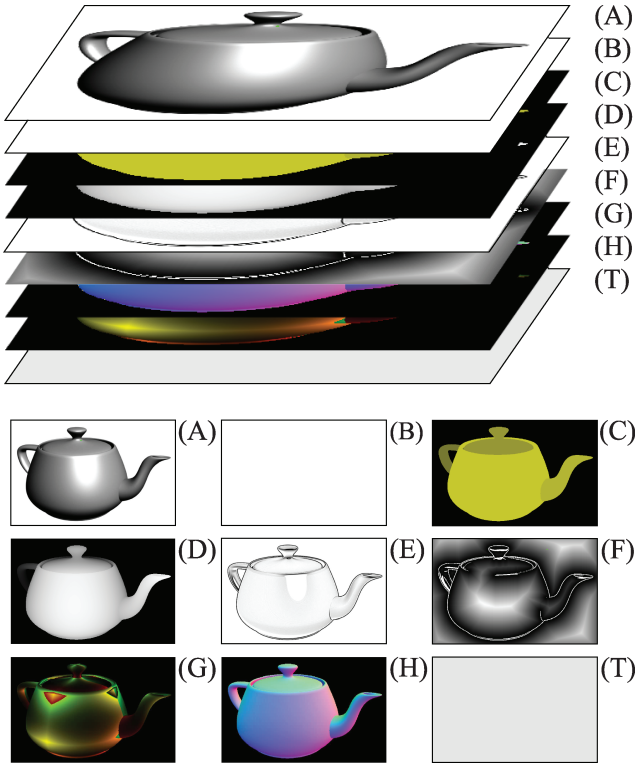



**Figure 1:** *The universe as a set of layers containing different kinds of pixel-based information. The input image (A), the result buffer (B) and the temporary buffer (T) are always present while the other buffers are optional and contain additional data, as there are, for example, object ID (C), depth (D), pre-computed edges (E), edge distances (F), texture coordinates (G) or normal vectors (H).*

required component of the universe. Here, the RenderBots build the final image and, therefore, always have write access. In addition, a temporary buffer ($T$) might be needed for storing temporary data in the form of markers.

The second component of the framework are the *RenderBots* themselves. They are relatively simple autonomous agents. We concentrate more on the behavior and status of the agents and less on their 'artificial intelligence'. Most noticeable, RenderBots do not have any sophisticated learning capabilities since we have realized that for our purpose this is not needed. As we shall see later, for each rendering style, specific RenderBots need to be developed. However, they are based on the same general structure. First of all, a RenderBot knows its position in the image plane as an $(x, y)$-coordinate pair. To change this position, velocity and acceleration values are stored and continuously updated during the simulation. Specific state variables complete the data portion of a RenderBot. Among these are also physical properties (like, for example, the mass $m$ of the bot) that are needed for the physical motion simulation.

The behavior of the RenderBots in our approach is mainly based on [23] and described via procedures. The three main actions are

1. *Simulate*: This procedure controls the bot's physical behavior, i.e. it uses information obtained ('sensed') from the environment in order to compute new direction and velocity values and possibly change the internal state of the RenderBot. To perform the changes, the RenderBot is able to gain information about its environment. This includes information from the universe as well as information about other RenderBots in a certain neighborhood. From this information forces $F_i$ are derived that act on the bot. The direction of the resulting force determines the direction of movement. A new acceleration value is computed as $a = \sum F_i/m$. Taking $\Delta t$ as the time difference between two time steps in the simulation, the new velocity computes as $v_{\text{new}} = v_{\text{old}} + a\Delta t$. This approach using a mass-force system was chosen since it offers an intuitive way of specifying the behavior in physical terms. Also, it introduces some kind of damping so that changes in velocity and direction are smooth and controllable.
2. *Move*: This procedure performs the actual movement of the bot by computing a new position $p_{\text{new}} = p_{\text{old}} + v\Delta t$.
3. *Paint*: RenderBots are capable of direct image generation, so this procedure actually renders the resulting image. Depending on the rendering style to achieve, i.e. the RenderBot class, two ways of painting are possible. In the first case, the RenderBot leaves behind a trace in the output buffer (e.g. for edge tracing or hatching). In the second case, the RenderBot itself represents the stroke to be drawn. Here, at the end of the simulation, each bot paints a certain 'footprint' into the output buffer (e.g. for stippling or mosaics).

Another important property of the RenderBots is their ability to clone themselves or to die. Cloning a RenderBot means to create a copy of the bot with identical parameter values and include this copy in the simulation. Dying simply means to remove the bot from the simulation. Cloning and dying depend on the environment conditions that are found.

In summary, each RenderBot is characterized by a specific set of data values that are needed for physical simulation as well as painting, and a set of procedures to perform simulation, movement and painting. To create different rendition styles, different classes of RenderBots perform different tasks. In Section 4, we describe some classes in more detail.

The complete image generation process can be described as follows:

1. setup the environment
2. create RenderBot swarms

3. while the image is not finished

   (3.1) simulate each bot

   (3.2) move each bot

   (3.3) paint each bot.

To setup the environment, a number of RenderBots of a specific class are created and distributed in the environment. Given the number of bots to be created, they will be distributed automatically in a random way with possible constraints being added so that, for example, they are only placed on certain image regions (e.g. on certain objects represented in an object ID buffer). The initial placement of the RenderBots can also be done interactively, i.e. the user 'paints' the bots using an interactive tool. A third way is that only a few RenderBots are initially created and, by cloning, they themselves cover a specific region or the whole image. In this manner several RenderBot 'swarms' can be created that might be of different classes or that are differently parameterized so that rendering styles can be mixed.

The actual control of the RenderBots is performed within the simulation main loop. After setting up the environment and creating the RenderBots, the following steps are executed until the user interrupts the image generation. Within the simulation step, each bot first gathers information from the environment and surrounding bots, then possibly manipulates the environment and finally manipulates its own state by computing new velocity and acceleration vectors and changing state variables. In the movement step, the position of each bot is updated according to the new velocity parameters. The painting step finally updates the target buffer.

As said above, the simulation ends on user interaction. The use of a MAS as it is done in our approach does not guarantee a convergence of the algorithm, i.e. there is no 'optimal' image that will be reached. First of all, the setup of the RenderBots relies on randomness, then, some of the control mechanisms also include random components so that no two runs are actually the same. The image is displayed while it is being created so that the user can interrupt the simulation whenever he or she is satisfied with the current state.

## 4. RenderBot Classes

There are various styles of SBR which mainly differ in the kind of strokes that are being used. Therefore, we provide different RenderBot classes, each of which corresponds to a certain kind of stroke. These bot classes mainly differ in the simulation process since the arrangement of the strokes also depends on the kind of strokes being used. Also, each bot class has its own painting procedure to realize the actual look of the represented strokes. The starting point for each simulation in general is any distribution of a number of the respective bots over the image plane. If only a single style is needed, then all these bots are of the same kind. As we will see in Section 5, RenderBot classes can also coexist in an environment to mix styles in one image.

### 4.1. Drawing Edges

Edges are an important tool for communicating the shape of an object. Especially, silhouettes aid the figure ground distinction. Looking at hand-drawn imagery, it is possible to reveal the structure and form of an object with just a few lines. Such lines are generally drawn along discontinuities of certain properties. This may be shading discontinuities or discontinuities in object space (object ID, depth, normal directions). The main idea for EdgeBots, i.e. RenderBots that follow and draw edges, is that these bots travel over the image and use information from an edge buffer to find edges. Once on an edge the bots move along the edge and draw.

To implement this behavior, an EdgeBot may take two different states: 'searching' and 'drawing'. In the searching state, the EdgeBot uses an edge buffer to calculate a force directed toward the nearest edge with a magnitude proportional to the distance to this edge. This moves the bot toward edges. If an EdgeBot has been searching for a given number of time steps without hitting an edge, it starts a random movement for a few simulation steps and then restarts the search for edges. Once an edge is found, the state of the bot switches to 'drawing' and it starts leaving behind a trace in the output buffer and markers in the temporary buffer. The markers are static agents that are inserted into the environment to show other RenderBots that this area has already been processed. They are set in a certain distance from each other along the trace left behind in the output. Now, the force that determines the movement is directed to follow the edge, i.e. toward a minimal difference in the distance from the edge. Figure 2 illustrates this. Once the end of an edge (no direction can be computed) or an area that has been drawn already (there are markers in the temporary buffer) is reached, the bot's state is again switched to 'searching'.

For an efficient implementation, some additions can be made. Searching EdgeBots are not only attracted by edges but also to some degree by nearby drawing EdgeBots. Hence, they follow their colleagues that already found an edge until they themselves have found an edge and, therefore, start drawing or until the EdgeBot being followed stops drawing. Then, the search behavior is started again without the attraction of drawing EdgeBots. An image generated this way using discontinuities in the input buffers as edges can be seen in Figure 3.

Compared to image-processing methods for edge detection, RenderBots offer the possibility to draw stylized edges by choosing appropriate parameters. Due to the chosen approach, however, an edge is not necessarily covered by one

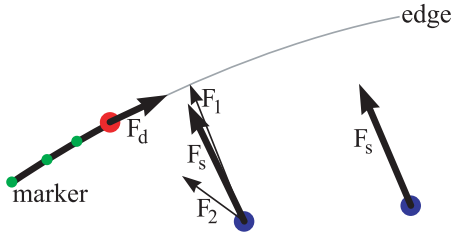



**Figure 2:** *The force affecting a drawing EdgeBot is always directed along the edge ($F_d$). $F_s$ as the force determining the movement of a searching EdgeBot results from a force $F_1$ directed toward the edge and possibly a force $F_2$ directed toward a nearby drawing EdgeBot. A searching EdgeBot further away from an edge just moves toward the edge.*

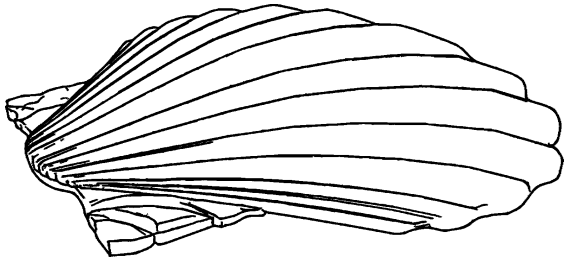


**Figure 3:** *Edges based on the principle outlined in Section 4.1.*

single stroke. This is not a limitation for the intended application area of rendering expressive images. In rare cases, RenderBots may need a long time to find an edge since they rely on local information and randomness.

### 4.2. Hatching

Hatching is characterized by the coverage of an area using short or long lines. These lines may follow certain image or object properties (e.g. curvature) and are usually almost parallel to each other. Several layers of hatching lines with different directions yield cross hatching. The general idea for HatchingBots is that they move along a certain direction and leave a trace behind if the environment conditions are appropriate.

The force acting on a HatchingBot is derived from the main direction of movement. This direction is either set within the initialization phase (e.g. the bots move horizontally over the image) or derived from the environment (e.g. the bot moves along the texture coordinates). At each time within the simulation a search radius is computed from the gray value at the current position. If there are optimal conditions within the search area, then the bot leaves a trace in the result buffer and sets markers in the temporary buffer, otherwise it just continues traveling without drawing. Optimal conditions means that there must not be more than a certain number of

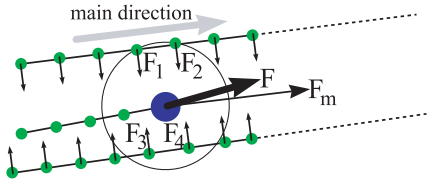



**Figure 4:** *Computation of the force affecting a HatchingBot. The main component $F_m$ is aligned to the main direction of movement. Neighboring markers apply repelling forces $F_1$ through $F_4$ that may change the direction slightly.*

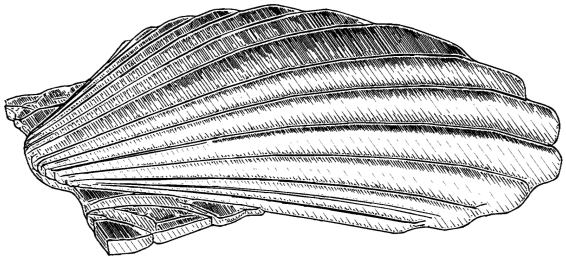


**Figure 5:** *Hatching based on the principle outlined in Section 4.2.*

markers within the search area. Otherwise the image would get too dark if a new line was added. If there are way too few markers within the search area, the HatchingBot clones itself to create more bots that darken the area. Two more conditions yield a more correct behavior of the HatchingBots. First, they try to maintain a maximal distance to lines that are already dawn. Second, they try to maintain a maximal distance to neighboring bots. To realize this, all markers and all neighboring bots in the search area apply a force onto the bot that is directed orthogonally to the main direction of movement. As a result, the direction of movement may change away from markers or neighboring bots. However, the weighting of these forces in relation to the main force only allows for small direction changes and, thus, keeps the general direction of movement (see Figure 4 for a principle illustration).

Also here, additional parameters are used for fine tuning. For example, the line width is configurable as well as the creation of halos around edges. For the latter, a HatchingBot consults the edge distance buffer and stops drawing if it comes too close to an edge. Figure 5 shows an example image with edges included for clarity.

In comparison to the use of stroke textures (cf. [27]) the limitation of our approach is that RenderBots cannot create complex texture patterns to represent, for instance, brick walls or stone pebbles. On the other hand, hatching and cross hatching is created in a unique style with the possibility of stylizing each stroke. Due to interaction with other RenderBot classes (e.g. EdgeBots), rather complex visual effects

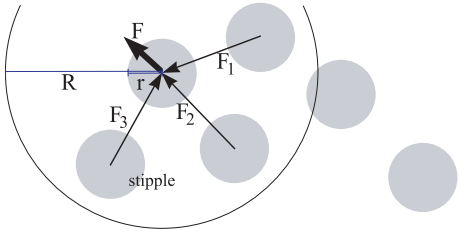



**Figure 6:** *Computation of the force affecting a StippleBot. Here, r is the actual dot radius, R the search radius. $(r/R)^2$ equals the intensity at the current position. $F_i$ are forces applied by nearby bots resulting in the force F that determines the movement.*

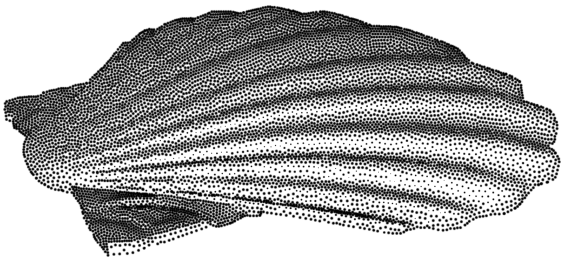


**Figure 7:** *Stippling based on the principle outlined in Section 4.3.*

(e.g. edge halos) can be achieved as an intrinsic feature of the technique.

### 4.3. Stippling

With stippling, tone as well as texture are created by the placement of small dots (stipples) where the dots are equally, but randomly, distributed. Each StippleBot, therefore, represents one dot of a certain radius $r$. The general idea for the simulation is that StippleBots require a certain amount of free space around them, which is determined from the underlying intensity of the source image. Thus, they move to keep a certain minimum distance in all directions to neighboring bots. This distance is smaller in darker areas and grows when the source image becomes brighter.

At each time step in the simulation, each StippleBot obtains the gray value from the source image at its current position and computes a search radius such that the gray value corresponds to the ratio between the area of the actual dot and the area of a circle with the search radius. All neighboring StippleBots within the search radius apply a repelling force to the current bot. The resulting force then determines the bot's movement. This basic principle is illustrated in Figure 6. This behavior to keep a certain distance to neighboring bots is equivalent to the collision avoidance described by Reynolds [23].

For a more complete treatment of stippling, some additional techniques are needed. If initially, there are too few StippleBots to achieve a decent coverage of the area, a StippleBot clones itself. Conversely, if there are too many StippleBots within the search radius, the bot itself dies. A StippleBot resting on a completely white area would cover an infinitely large search radius. We constrain this to a maximum value. Also, starting from a specified gray value threshold, the radius of the dot to be drawn may change. This is an additional tool to cover bright areas. Here, the dots not only become more sparse but also smaller. Finally, we have included a specific behavior of the StippleBots if they come close to an edge. The bots are attracted to edges but they must not cross edges. Hence, a chain of StippleBots settles close to an edge while their number and mutual distance is still controlled by the source gray value and the number of nearby bots. An image rendered this way can be seen in Figure 7.

Interestingly enough, the placement of the dots and, thus, the artifacts that result from our technique and from the method proposed by Deussen *et al.* [10] are equal. Both result from the same geometric principle of locally relaxing a point distribution to achieve an almost equal spacing between the dots while the average dot distance depends on the gray value to be represented. To circumvent the formation of these characteristic relaxation patterns, minimal random position changes can be introduced. This, however, violates the homogeneous stipple distribution to a certain degree.

### 4.4. Mosaics

Mosaics are traditional designs and pictures formed from the juxtaposition of small tiles of stones, terracotta or glass. They are one of the earliest form of artistic imagery. The general idea to generate mosaics is to have each MosaicBot represent one tile. The bots and, thus, the tiles have a rectangular form and orient themselves according to a given vector field (e.g. direction vectors perpendicular to edges in the image). To achieve the mosaic effect, tiles are prohibited to overlap themselves.

Each MosaicBot covers a certain area of a certain rectangular form as can be seen in Figure 8. This is also the search area where other bots influence the position. To calculate the affecting forces, each neighboring bot applies a repelling force that is proportional to the area the two bots overlap. These forces result in a movement that shuffles the tiles in place. In addition, MosaicBots rotate themselves to align with the given vector field. This means, a rotation component of the force is generated from the alignment difference between the MosaicBot and the given direction field. This rotation aligns nearby bots mostly in a similar way so that the chance for overlapping decreases. If edge information is present, an additional constraint can be modeled. Markers at edges placed by EdgeBots (see Section 4.1) also

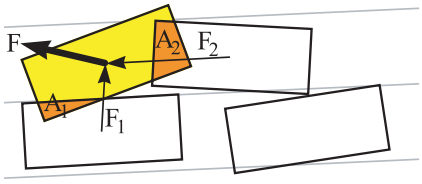



**Figure 8:** *Computation of the force affecting a MosaicBot as the resulting force of $F_1 \sim A_1$ and $F_2 \sim A_2$. In addition, the bots rotate to align with the gray lines that illustrate the direction field.*

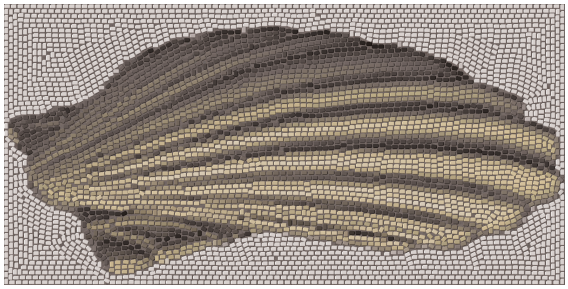


**Figure 9:** *Mosaic based on the principle outlined in Section 4.4.*

apply repelling forces to the MosaicBots so that they do not cross edges in the image.

For rendering the mosaic, the MosaicBot's area is then filled to represent the tile. The color is taken from the source buffer at the current position or averaged over the pixels in the source buffer covered by the bot's area. For an even more realistic look, the shapes that are drawn in the output buffer might further be modified. Figure 9 shows an example. The start configuration fills the whole image with a few randomly places tiles. MosaicBots are capable of cloning themselves so that in the beginning the image will be covered almost completely with MosaicBots (i.e. tiles) which are then simulated as described above.

### 4.5. Paintings

Painterly rendering as another style of SBR tries to cover an area with brush strokes that make the final image resemble traditional paintings. In paintings, strokes almost always cover areas of like color, so that the main idea for the PaintBots is to draw a stroke until the color difference between the starting point of the stroke and the current point is above a certain threshold. A PaintBot starts at an empty point in the target buffer and picks the color from the source buffer as the current color. It then moves in the direction of the minimum color difference between the source buffer color and the starting point color. The direction of movement is restricted to deviate no more than a certain threshold from the direction in the previous simulation step in order to prevent wildly bent

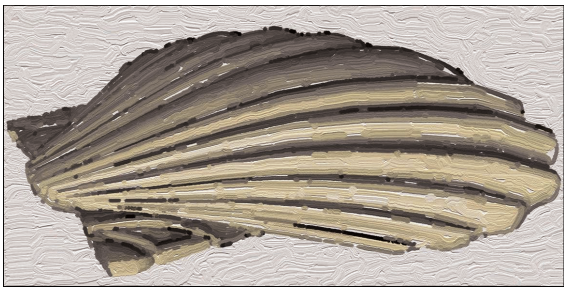


**Figure 10:** *Painterly rendering based on the principle outlined in Section 4.5.*

strokes. A stroke ends if either the color difference gets too large or the PaintBot arrives at a different stroke.

The painting is created by the following procedure. In each step, a quadrilateral is computed from the current position $p_i$ to the position $p_{i-1}$ from the previous step which is centered on the line between $p_i$ and $p_{i-1}$. The width of the quadrilateral is determined by a parameter holding the width of the brush. These quadrilaterals are then connected to form a strip which is filled using the color from the starting point. An alpha texture gives the characteristic appearance of a brush stroke.

Also this basic principle can be extended in various ways. The length of the strokes can and should be constrained to a maximum value so that not too long strokes appear as a result. Also, we use a bump map to create a three-dimensional look as can be seen in Figure 10.

## 5. Examples

In the following, we shall provide some examples in order to show how the various RenderBot classes can be mixed to create stylistic images. Also, we will show some additional details that are left out in the presentation of the general approach in Sections 3 and 4.

Figure 11 shows a mosaic that was created using seven different kinds of MosaicBots. A photograph served as input. This photograph has been segmented by hand and the segmentation result has been stored as object ID buffer. The MosaicBot swarms differ from each other in the form and size of the tiles. The 3D effect on the tiles is created in the final painting phase by treating each tile as a 3D surface that has different normal vectors for each side plane and thus makes the shading of the tiles dependent on the lighting conditions when rendered using OpenGL.

Besides mixing of bots of the same class but with different initial parameters, also different bot classes can be mixed. The top image in Figure 12 shows an example where EdgeBots work on the complete image and StippleBots were constrained to the sole of the sandal. The HatchingBots for the lower part of the straps move along the $u$ texture direction,

**Figure 11:** *Mosaic created with seven swarms of MosaicBots, each of which has a different size and form of the tiles.*

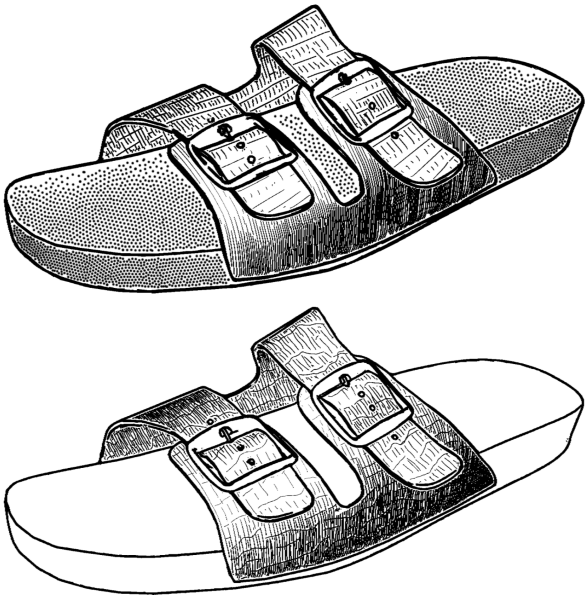


**Figure 12:** *Mixing various styles by combining approximately 200 EdgeBots, up to 2,000 HatchingBots, and approximately 4,500 StippleBots, some of them constrained to object IDs (top). Loose hatching for the straps, again approximately 200 EdgeBots, and up to 2,000 HatchingBots (bottom).*

while for the upper part two swarms of HatchingBots were used that create the impression of crosshatching by following the $u$ and $v$ texture coordinates, respectively. Changing the physical behavior of the bots allows for different stylistic variations of the lines. Overshooting, for example, is easily accomplished by giving the bots a higher mass and acceleration and, hence, a higher inertia. A more loose style, as can be seen in the bottom image in Figure 12, is created by adding a random sideways force acting on the bots and making them deviate from the main direction of movement. Figure 13 finally shows another example using various RenderBot classes in one image.

The examples show the flexibility of our approach which is mainly gained through the many possible options to setup the RenderBots (and the environment). Connecting the behavior of the bots with the painting capabilities is the key for creating the desired images. Once the setup is complete, the simulation can handle up to 10,000 RenderBots at interactive rates. Since the user decides when to end the simulation, i.e. when the desired image quality is reached, he or she determines the simulation time to a large extent. Also, the initial distribution of the bots has a large influence so that in complex cases an interactive initial placement is preferable over an automatic random distribution. The simulation time obviously depends also on the number of bots that are present in the environment. In typical cases, as they can be seen in this section, each simulation run needs a few seconds up to 1 or 2 minutes (on a consumer class PC) from setting up the environment until the image is satisfying. To find the optimal parameter settings, some runs are needed—typically 3–10.

## 6. Conclusion

We have presented a novel way to create stroke-based renditions using MAS. Our technique works on a given image and re-renders this image using various kinds of strokes. This approach uses a MAS where different stroke types are represented by different agent classes, so called RenderBots.

Our approach is capable of rendering various styles using the same basic principle. The placement of the strokes is mainly determined by a physical simulation of the distribution and movement of the RenderBots while the graphical properties, i.e. the look of the final image, comes from painting algorithms that differ for the RenderBot classes. Due to the randomized initial distribution of the RenderBots, our approach is non-deterministic in that no two same images are created. However, the presented technique is highly flexible and easily to extend to new styles. A vast amount of parameters allow to control the behavior of the RenderBots and, therefore, to achieve a large number of effects. Once the parameter setup is complete, we have found that even if a large number of RenderBots is simulated, the time needed to render an image is within feasible limits. In this sense, the

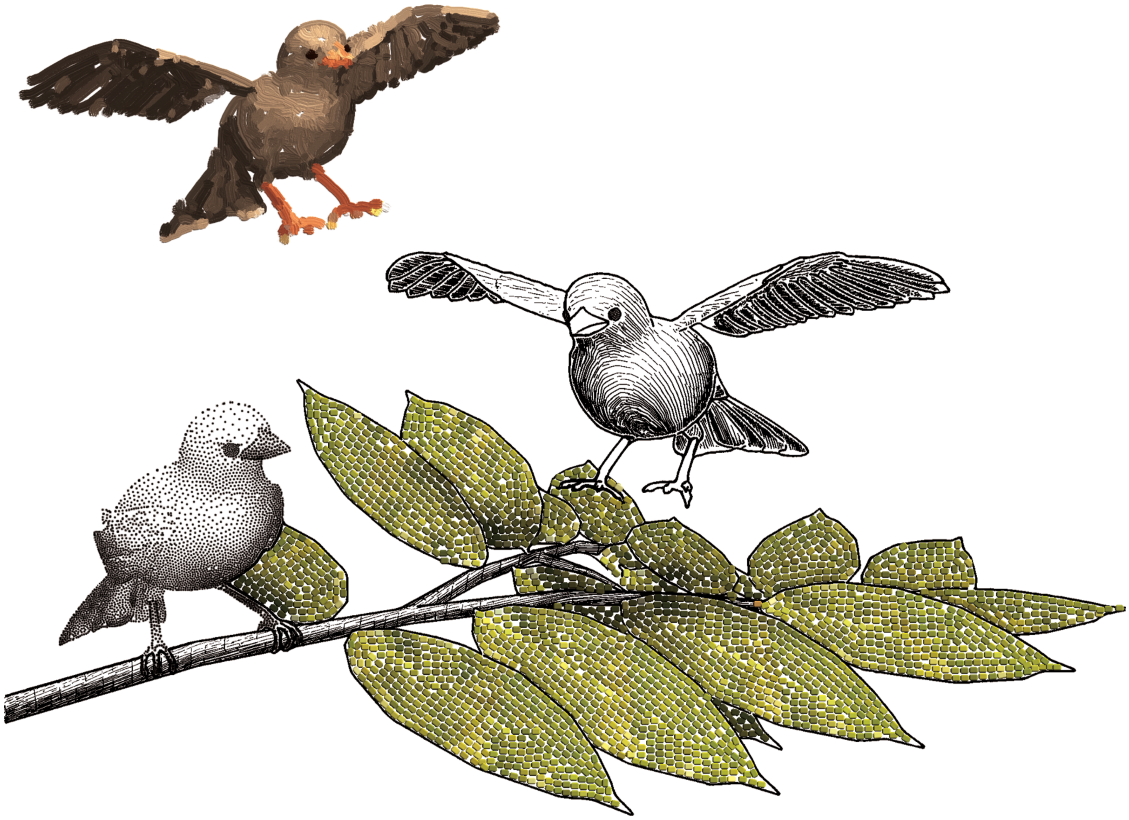


**Figure 13:** *Another example combining different styles in one image. To create this image, approx. 300 EdgeBots, 3,000 MosaicBots, 4,000 StippleBots, 100 PaintBots, and up to 1,000 HatchingBots are used.*

use of MAS provides a unified treatment of SBR. Since SBR is concerned with distributing strokes (i.e. image elements) over the surface of an image, our approach perfectly matches this goal. Having RenderBots work in object space would certainly open up new possibilities but would also need some conceptual changes since now RenderBots do no longer represent image features or elements.

The presented approach has some limitations, however. MAS, as they are used here, do not guarantee that all features of the image are found that should be re-rendered. This results from the approach that RenderBots can only access information locally in a certain area. Therefore, in some cases, blank spots may occur. On the other hand, they give the resulting images a rather loose look and can be prevented by placing additional RenderBots by hand. Even though the system runs at interactive frame rates, the images are not created in real time, so that it is not possible to generate real time animations. Nonetheless, the approach is in some cases more efficient than the known techniques presented in earlier papers. Also, the (local) behavior of the bots can intuitively be formulated and, therefore, new behaviors can easily be developed. Finally, to circumvent highly complex simulation and painting procedures, it is sometimes necessary to find a compromise between the RenderBot's complexity and efficiency. So, for example, drawing a line twice in certain areas is sometimes less a problem than coding a complex behavior that keeps a RenderBot from doing so.

For future work, we agree on Hertzmann [2], who calls for an expansion of the range of styles that can be created using SBR. We believe that our approach is capable of producing far more styles than presented in this paper. A thorough analysis of hand-drawn imagery will reveal the principles behind an adequate simulation model. Extending the capabilities of the RenderBots over and above the relatively simple physical simulation will open the way for more complex styles. Making the RenderBots more 'intelligent', i.e. provide more memory, learning capabilities and extend the cloning by mutation and other modifications, will give a wider range of possibilities to experiment with and to develop new pictorial styles. Due to the flexibility gained by the use of a MAS, such new styles can be incorporated easily. Furthermore, we are not restricted to pixel-based output. The painting functions can easily be changed to produce high-quality vector-based output.

A second direction that needs attention is the search for application areas. First tests have shown that RenderBots are also applicable in the area of visualization, comparable to Spray Rendering introduced in [28]. Having the bots react in a certain way to certain properties of a 2D vector field,

for example, provides an intuitive and flexible tool for the exploration and visualization of flowing phenomena. Here, new visualization techniques will arise from the use of SBR methods. More general, RenderBots can be extended to the visualization of 2D oriented data, i.e. data that is given over a 2D surface and hence need to be visualized as 2D images.

Finally, as with many rather technically oriented rendering methods, the quest for good user interfaces is a big problem. We are looking for intuitive ways of making the (physical) behavior of the RenderBots understandable and controllable by the user. This includes the choice of parameters for a RenderBot swarm as well as an interface for the development of new RenderBot classes. Nonetheless, the agent metaphor we have used here offers an exploratory way to create stylistic renditions which is often called for by users.

## References


1. T. Saito, and T. Takahashi. Comprehensible rendering of 3-D shapes. *Computer Graphics (Proceedings of ACM SIGGRAPH 90)*, 24(4): 197–206, 1990.

2. A. Hertzmann. A survey of stroke-based rendering. *IEEE Computer Graphics and Applications*, 23(4): 70–81, 2003.

3. M. P. Salisbury, S. E. Anderson, R. Barzel and D. H. Salesin. Interactive pen-and-ink illustration. In *Proceedings of ACM SIGGRAPH 94*, New York, A. Glassner, (Ed.), *Computer Graphics Proceedings, Annual Conference Series*, ACM Press/ACM SIGGRAPH, pp. 101–108, 1994.

4. M. P. Salisbury, M. T. Wong, J. F. Hughes and D. H. Salesin. Orientable textures for image-based pen-and-ink illustration. In *Proceedings of ACM SIGGRAPH 97*, New York, T. Whitted (Ed.), *Computer Graphics Proceedings, Annual Conference Series*, ACM, pp. 401–406, 1997.

5. M. A. Kowalski, L. Markosian, J. D. Northrup, L. Bourdev, R. Barzel, L. S. Holden and J. F. Hughes. Art-based rendering of fur, grass, and trees. In *Proceedings of ACM SIGGRAPH 99*, New York, *Computer Graphics Proceedings, Annual Conference Series*, ACM, pp. 433–438, 1999.

6. P. Haeberli. Paint by numbers: Abstract image representations. *Computer Graphics (Proceedings of ACM SIGGRAPH 90)*, 24(4): 207–214, 1990.

7. A. Hertzmann. Paint by relaxation. In *Proceedings of Computer Graphics International 2001, Hong Kong, July 2001,* Los Alamitos, IEEE Computer Society Press, pp. 47–54, 2001.

8. P. Litwinowicz. Processing images and video for an impressionist effect. In *Proceedings of ACM SIGGRAPH 97*, New York, T. Whitted (Ed.), *Computer Graphics Proceedings, Annual Conference Series*, *ACM*, ACM Press/ACM SIGGRAPH, pp. 407–414, 1997.

9. S. P. Lloyd. Least square quantization in PCM. *IEEE Transactions on Information Theory*, 28(2): 129–137, 1982.

10. O. Deussen, S. Hiller, C. W. A. M. van Overveld and T. Strothotte. Floating points: A method for computing stipple drawings. *Computer Graphics Forum (Proceedings of EuroGraphics 2000)*, 19(3): 40–51, 2000.

11. A. Secord. Weighted voronoi stippling. In *Proceedings of NPAR 2002, International Symposium on Non Photorealistic Animation and Rendering*, New York, ACM Press, pp. 37–44, 2002.

12. S. Hiller, H. Hellwig and O. Deussen. Beyond stippling—Methods for distributing objects on the plane. *Computer Graphics Forum (Proceedings of Eurographics 2003)*, 23(3): 515–522, 2003.

13. A. Hausner. Simulating decorative mosaics. In *Proceedings of ACM SIGGRAPH 2001*, New York, E. Fiume (Ed.), *Computer Graphics Proceedings, Annual Conference Series*, ACM, pp. 573–580, 2001.

14. G. Elber and G. Wolberg. Rendering traditional mosaics. *The Visual Computer*, 19(1): 67–78, 2003.

15. J. Kim and F. Pellacini. Jigsaw image mosaics. *ACM Transactions on Graphics*, 21(3): 657–664, 2002.

16. B. J. Meier. Painterly rendering for animation. In *Proceedings of ACM SIGGRAPH 96*, New York, H. Rushmeier (Ed.), *Computer Graphics Proceedings, Annual Conference Series*, ACM, ACM Press/ACM SIGGRAPH, pp. 477–484, 1996.

17. O. Meruvia Pastor, B. Freudenberg and T. Strothotte. Real-time, animated stippling. *IEEE Computer Graphics and Applications*, 23(4): 62–68, 2003.

18. R. D. Kalnins, L. Markosian, A. Michael, B. J. M. Kowalski, J. C. Lee, P. L. Davidson, M. Webb, J. F. Hughes and A. Finkelstein. WYSIWYG NPR: Drawing strokes directly on 3D models. *ACM Transactions on Graphics*, 21(3): 755–762, 2002.

19. W. T. Reeves. Particle systems — A technique for modeling a class of fuzzy objects. *Computer Graphics (Proceedings of ACM SIGGRAPH 83)*, 17(3): 359–376, 1983.

20. C. Curtis. Loose and sketchy animation. In *SIGGRAPH 98 Conference Abstracts and Applications*, New York, *ACM SIGGRAPH*, p. 317, 1998.

21. G. Weiss. *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*. MIT Press/AAAI Press, Cambridge, MA, USA, 1999.

22. C. Tessier, L. Chaudron and H. J. Müller. *Conflicting Agents: Conflict Management in Multi-Agent Systems*. Kluwer Academic Publishers, Boston, 2002.

23. C. W. Reynolds. Flocks, herds, and schools: A distributed behavioral model. *Computer Graphics (Proceedings of ACM SIGGRAPH 83)*, 21(4): 25–34, 1987.

24. J. Liu. *Autonomous agents and Multi-Agent Systems: Explorations in Learning, Self-Organization, and Adaptive Computation*. World Scientific Publishing, 2001.

25. X. Tu and D. Terzopoulos. Artificial fishes: Physics, locomotion, perception, behavior. In *Proceedings of SIGGRAPH 94*, New York, A. Glassner (Ed.), *Computer Graphics Proceedings, Annual Conference Series*, ACM, pp. 43–50, 1998.

26. S. R. Musse, C. Babski, T. Capin and D. Thalmann. Crowd modelling in collaborative virtual environments. In *Proceedings of the ACM Symposium on Virtual Reality Software and Technology 1998*, New York, ACM Press, pp. 115–123, 1998.

27. G. A. Winkenbach and D. H. Salesin. Computer-generated pen-and-ink illustration. In *Proceedings of ACM SIGGRAPH 94*, New York, A. Glassner (Ed.), *Computer Graphics Proceedings, Annual Conference Series*, ACM, pp. 91–100, 1994.

28. A. Pang and K. Smith. Spray rendering: visualization using smart particles. In *Proceedings of IEEE Visualization'93*, Los Alamitos, IEEE, IEEE Computer Society Press, pp. 283–290, 1993.

# ICON-BASED VISUALIZATION USING MOSAIC METAPHORS

F

Bibliographische Angaben:

Thomas Nocke, Stefan Schlechtweg und Heidrun Schumann: „Icon-Based Visualization Using Mosaic Metaphors“. In: *Proceedings of the Ninth International Conference on Information Visualisation (IV'05, London, England)*. IEEE Computer Society Press, Los Alamitos, CA, 2005. S. 103–109.

Abstract:

This paper introduces a new approach to extend icon-based visualization methods by using a mosaic-based paradigm. We discuss, how image metaphors closely related to the application domain can be applied for icon-based representations. Therefore, we enhance visualizations by well-known Image Mosaic techniques, such as image layouts, image selection and color adaption. Furthermore, we present the results of our approach by discussing an example of a clustered real-world climate data set.

# Icon-based Visualization using Mosaic Metaphors

Thomas Nocke[1], Stefan Schlechtweg[2], Heidrun Schumann[1]
[1]University of Rostock, {nocke, schumann}@informatik.uni-rostock.de
[2]University of Magdeburg, stefans@isg.cs.uni-magdeburg.de
Germany

**Abstract**

*This paper introduces a new approach to extend icon-based visualization methods by using a mosaic-based paradigm. We discuss, how image metaphors closely related to the application domain can be applied for icon-based representations. Therefore, we enhance visualizations by well-known Image Mosaic techniques, such as image layouts, image selection and color adaption. Furthermore, we present the results of our approach by discussing an example of a clustered real-world climate data set.*

**Keywords** Icon-based Visualization, Mosaic-based Rendering, Climate Impact Research, Visualization Metaphors

## 1. Introduction

The visualization of large multi-variate data sets is still a challenging task, especially when we consider the exploration of a variety of attributes in one representation. Many techniques have been developed to visualize such multi-variate data sets (cf. [5] for an overview):

- pixel-oriented techniques represent each data value as a single pixel on the screen,
- geometric projection techniques map the $n$-dimensional information space to "interesting" subspaces,
- icon-based techniques display small icons representing the attribute values of the data records, and
- hierarchical and graph-based techniques display the information space as a hierarchy or a graph.

In this paper we will focus on presenting multi-variate data on maps. In this context icons are a frequently used approach. One advantage of such icons is that they combine all data values of a given data set in one primitive and, thus, enable users to recognize multi-variate correlations. However, the perceptibility of such icons quickly decreases with the number of variables being displayed. To increase the intuitive recognition of such icons, easy perceptible metaphors such as Chernoff faces [2] and Infobugs [3] can be applied.

Techniques visualizing information using intuitive metaphors have been developed for different fields. The InfoCanvas system [11] enables users to interactively select and place metaphors on an image. In [10] the simple and expressive image metaphor of "money trees" for stock data has been introduced, to give just two examples.

Especially in the domain of cartography metaphors or symbols are widely used to support the effective recognition of general information. The question we want to answer is whether metaphors can be used to communicate numerical, multi-variate data as well. In other words: Can we replace abstract icons by metaphor-based icons?

When placing such metaphor-based icons on maps, several problems similar to the generation of Image Mosaics have to be solved. In this paper we discuss how algorithms that create such Image Mosaics can be applied to enhance icon-based visualizations on maps.

The paper is organized as follows. In Section 2 we describe the background. Afterwards, we discuss our approach to mosaic-based visualization (Sec. 3). Furthermore, we describe and discuss the results of our approach on a data set from the climate impact research background (Sec. 4). Finally, we conclude with a short discussion and outline future work (Sec. 5).

## 2. Background

In this section, we first outline basics of icon-based visualization techniques, afterwards discuss the background of Image Mosaics and, finally, describe the background of the data set to which we have applied our approach.

### 2.1. Icon-based visualization techniques

Icon-based visualization techniques have been proven to be effective for the visualization of multi-variate data. They map data variables onto geometric (e.g., shape, size, orientation) and non-geometric (e.g., color and texture) visual attributes (cf. [14]). There are two general *modi operandi*:

1. techniques generating individually recognizable icons, and

2. icon-based techniques generating textures.

Well-known examples for the first class are StarGlyphs [13] and Autoglyphs [1]. Both enable users to identify and compare certain data values. With a rising number of concurrently displayed icons, texture-based techniques from the second class are more expressive (e.g., Stick-Figure-Icons [9] and Color-Icons [6]). They support the identification of general trends and clusters in the data.

The design of suitable icons is a sensitive process. Special editors have been developed to interactively construct such icons. However, metaphor-based icons to visualize multi-variate data are – with the exception of Chernoff faces [2] – rather uncommon.

## 2.2. Image Mosaics

Image Mosaics have become very popular in the last few years as a form of art, which is used for advertisements, posters, etc. An Image Mosaic is created from other smaller images which together portray a larger subject. An input image is to be replaced by an output image which is constructed from several "tiles" of a size larger than a pixel and which match the tonal values of the input image (cf. [4, 7]). The creation of an Image Mosaic requires the following four major steps:

1. choose images which are to be used as mosaic tiles,
2. choose a tiling grid,
3. find an arrangement for the mosaic tiles in the grid,
4. possibly perform a color correction on the tiles to match the target image

The most interesting feature of Image Mosaics that will be used in the context of this paper is that they include two different levels of detail in one image. First, you get a global view on the portrayed subject by looking at the mosaic as a whole. Second, the individual tiles show small details depending on the selection of the tile. Typically, the tiles are selected in such a way that they portray something that is connected to the subject which is shown in the mosaic.

If we compare this to icon-based visualization techniques, the icons representing the data can be regarded as the image tiles and the display of the whole data set as the mosaic itself. The challenges are now similar to the four steps described above:

1. find iconic representations of the individual data items,
2. find a layout for the icons depending on the underlaying data set or chose a layout,
3. place the icons according to the chosen layout,
4. possibly perform a color correction to fulfill the goals regarding a global overview of the data.

We want to discuss this strategy considering as example a concrete application scenario.

## 2.3. Application: a maize harvest data set

A challenge for climate researchers is to develop methods that support the evaluation of climate model results and to reproduce extreme conditions. Important instruments in this context are cluster analysis and visualization techniques (see [8]). Here especially, the visual exploration of the multi-variate features of spatially-scattered clusters can benefit from Image Mosaic algorithms.

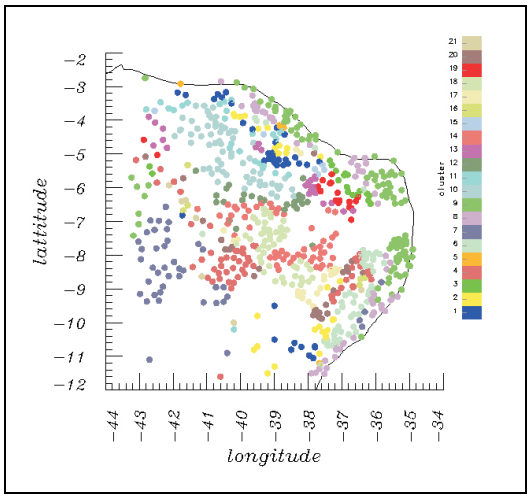



**Figure 1. Visualization of clusters representing the risk of a drought for maize cultivation during the year 1983 in the semi-arid Northeast of Brazil based on regional climate model results.**

The data set that we will be using as example throughout the paper represents a severe drought that occurred during the year 1983 in the semi-arid northeast of Brazil. To generate the data set, idealized criteria, identified by climate researchers, were assumed based on certain total precipitation thresholds. Doing so, it is possible to characterize the risk of potential total yield loss for maize as one of the major agricultural crops of this region. These criteria have been used to derive six parameters expressing the positive differences between the individual precipitation thresholds and the actual rainfall. For these, a cluster analysis on the 865 measurement stations has been performed (see Fig. 1).

In this figure, similar observation stations – referring to the mentioned parameters – are aggregated into the same cluster, whereas dissimilar stations belong to different clusters. Unfortunately, such an image of circles representing the cluster ID of each observation station does not allow to get an overall overview about cluster features as well as to

explore all the parameters constituting the cluster in one image. To achieve a better expressiveness and effectiveness of the image, the following tasks have to be supported:

1. the cognition of general cluster information, e.g., the identification of extreme cultivation conditions,
2. the exploration of individual parameters and of parameter dependencies, and
3. the identification and localization of measurement stations.

This can be achieved using metaphor-based icons.

## 3. An approach for mosaic image visualization

In the following we want to discuss, how the tasks from Section 2.3 can be supported using the Image Mosaic approach outlined in Section 2.2. The first step is to find an adequate iconic metaphor for the individual measurement stations.

### 3.1. Designing the mosaic pattern

The data set consists of six parameters and a cluster ID at each 2D position, altogether seven values. Since the cluster ID is of more general character and of higher importance for a general overview, we need to encode it separately. Considering the meaning of the parameters in an agricultural context, a metaphor based on a simplified maize cob is very straightforward and intuitive (see Fig. 2). Low risks for the maize harvest are represented by a thick, yellow cob (Fig. 2 left), whereas high risks are represented by a thin, brown cob. Other image metaphors, for instance, displaying clouds, plants and the sun are imaginable. We have decided against it, since the maize cob is well-perceptible even in an image with a high number of measurement stations.

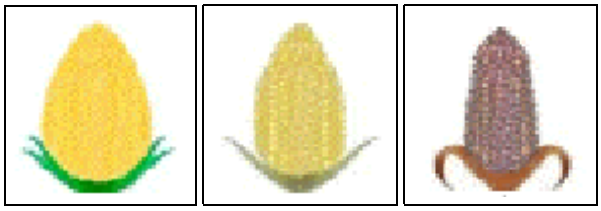


**Figure 2. The three base icons displaying maize conditions: good (left), middle (center) and bad (right) conditions**

The metaphoric images from Fig. 2 can be arranged and chosen analogous to mosaic images, yellow cobs in “good” regions and brown cobs for “bad” regions. However, keeping in mind that we want to analyze six parameters, we need to extend this simple metaphor. A straightforward approach is to draw six of these maize cobs (one for each parameter). Unfortunately, due to the requirement that the small images should remain identifiable to retain their metaphoric meaning, the resulting rectangular icon gets quite large, producing overlapping problems in regions with high point density. Thus, we subdivide the maize cob image into six regions, and construct the resulting image for a parameter combination based on six image parts of Fig. 2. For instance, high values of parameter $p_1$ lead to a thick, yellow head on the left cob side. Fig. 3 illustrates this procedure. Doing so, we avoid to enlarge the image and the user gets all information at a glance.

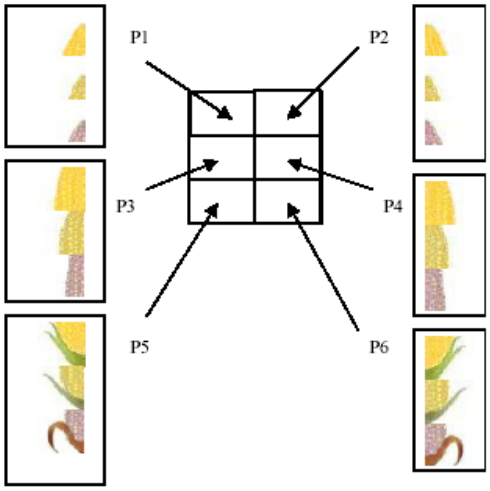



**Figure 3. Construction of a metaphor-based icon, representing six parameters**

Furthermore, using a certain background color, these icons simply allow to encode the cluster membership as well (see Fig. 4). Doing so, the resulting mosaic image gets a well-perceptible “coat of paint”, that encodes this important information.

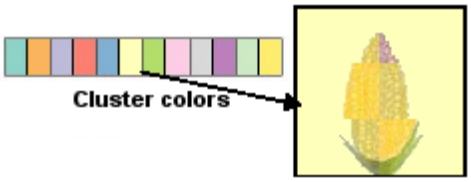



**Figure 4. Metaphor-based icon representing six parameters with the background color identifying the cluster**

### 3.2. The layout problem

After the mosaic tiles have been designed we have to find a layout for the tiles and place them according to the chosen layout (steps 2 and 3 from Section 2.2). Three basic grid layout types have been introduced [12]:

- scattered layouts,
- regular layouts and
- multi-resolution layouts.

Fig. 5 illustrates these layout types. Each of them has certain advantages and disadvantages for the visualization of our data set.

The *scattered layout* for Image Mosaics is a random process, in general used as an artistic feature. This procedure is – in general – not suitable for a visualization of numerical data, since the tile images (or icons) in visual representations – in contrast to mosaic images – can not be placed to arbitrary positions. In the visualization context we need to keep the important relation between the visual representation of the measured data and the position of the measurement stations. Based on these considerations, there are two possible scattered layout mechanisms:

- using a straightforward layout placing icons of constant size at the position of the measurement stations, and
- placing icons into given or calculated (e.g., by a Voronoi tesselation) regions, adjusting their position and size according to the region's extent.

We use the first mechanism, placing images at the measurement station positions only (see Fig. 5 left). One advantage of this layout is that it directly represents the underlying distribution of 2D locations and avoids possibly misleading interpretations. A disadvantage, however, is the possible overlapping of icons in regions with high station density.

To overcome this drawback, regular and multi-resolution layouts can be applied. *Regular layouts* (Fig. 5 center) show in each grid cell an aggregated information over all measurement stations (see Sec. 3.3). Three kinds of regular mosaic layouts are commonly used:

- rectangular grid (standard regular grid),
- hexagonal grid, and
- brick wall layout.

For visualization purposes rectangular as well as hexagonal grids are suitable for different icon shapes. However, a brick wall layout is a more artistic instrument and seems to bring no new benefit for the visualization. We use the rectangular grid, which is especially suited for the rectangular icon we have designed. Moreover, we have applied a region merging of neighboring grid cells with the same (mean) cluster to larger blocks ($2 \times 2$, $3 \times 3$, $\ldots$), and display these regions with a larger icon. This allows to identify large homogeneous regions quickly.

A regular grid layout enables users to get a quick overview of the data set and to explore the overall distribution of values and clusters. A drawback of such layouts is that the original parameter and cluster values are aggregated and, hence, no longer explicitly visible. As a solution to this problem, the grid resolution can be changed in a detail-on-demand manner. A low grid resolution supports a fast overview of distributions in the data with large mosaic icons, and a high grid resolution allows to get details of certain regions if required with smaller icons. Furthermore, to keep an overview of the point distribution in a certain region, small dots representing the measurement stations can be faded in (see Fig. 6).

As a third layout technique, a *multi-resolution layout* (see Fig. 5 right) can be applied. Here, icons are placed in a region, if there is exactly one cluster in this region. If not, the region is subdivided. Then, parameter values of similar stations only from the same cluster are aggregated and, thus, we avoid to merge regions with inhomogeneous properties. We used a quadtree-based multi-resolution algorithm.

This layout allows to gain a deeper insight into local cluster distributions, keeping all clusters visible. Now the question is how to visualize the metaphor-based icons in grid cells with varying sizes. On the one hand, the icons can be visualized in constant size independently of the grid cell size. On the other hand, the icon size can be scaled to the available space of the grid cell. The decision between these possibilities is background dependent. For instance, if large homogeneous regions should be perceived quickly, the second technique is appropriate (see Fig. 5 right).

A quadtree bases on a successive subdivision of rectangles and, by doing so, approximates the underlying regions. Actually, region boundaries have irregular shape. Thus, the quadtree subdivision can cause the extension of icons into "undefined" regions. To avoid drawing mosaic tiles into these regions (in this case to the Atlantic ocean (Fig. 5 right)), clipping has to be integrated into the quadtree traversal algorithm. Altogether, a quadtree-based multi-resolution layout strongly adulterates the region boundaries (Fig. 5 right), whereas a scattered layout only minimally changes the region shape representation (Fig. 5 left).

### 3.3. Icon placement

After the layout step has been performed we have to decide which icons to place on the map (step 3 from Section 2.2). This decision is strongly layout-dependent, especially owing to different region merging strategies (see Sec. 3.2).

In the case of a scattered layout, we simply construct the icons based on the individual station parameter values. If,

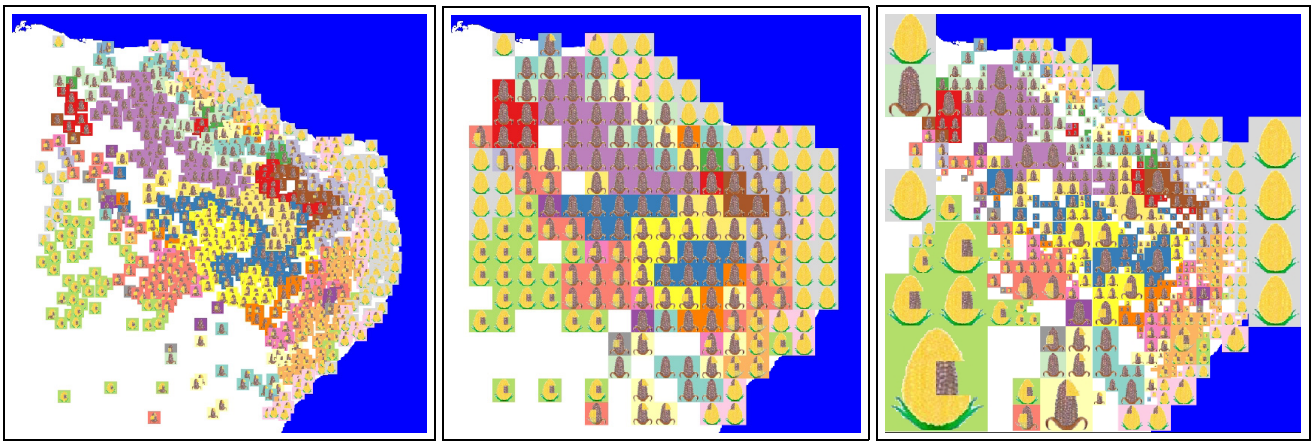


**Figure 5. Three layout types of Image Mosaics: scattered(left), regular (center) and quadtree-based multi-resolution (right) layout**

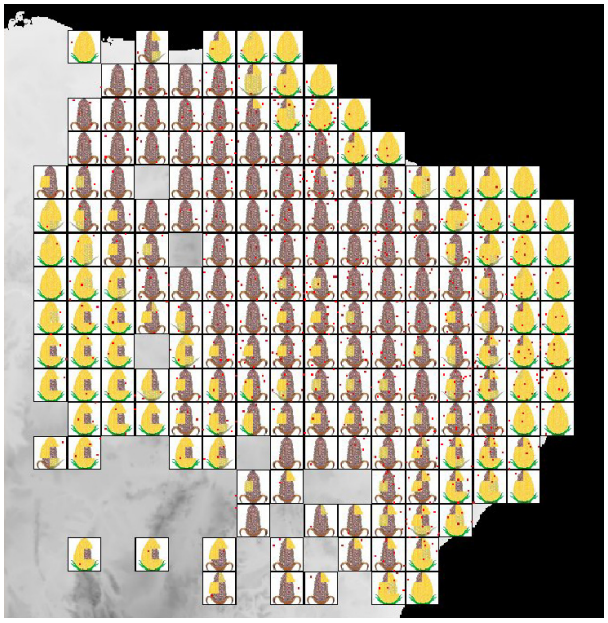


**Figure 6. Mosaic image in regular layout with stations faded in (red dots)**

on the other hand, the icon should represent a region possibly consisting of more than one station, the user can decide which icon to display:

- a “typical” station in the region,
- an icon based on the mean values for all the stations in the region,
- a cluster representative of the
  - most common cluster (for a regular layout), or
  - only cluster (for a multi-resolution layout)

  in the region.

### 3.4. Color correction

Finally, we have to parameterize the metaphoric icons, first of all applying a color correction (step 4 from Section 2.2).

First of all, a general consideration: when designing the icons (see Sec. 3.1) we have chosen colors for the parameter categories for each of the six icon parts. This process can be seen as a kind of color correction with only three colors. To generalize this process, any number of categories and referring image parts can be produced using a color interpolation of the “key images” (in the sense of color correction). Since we – in addition to the colors – use different shapes, we have to interpolate the shapes as well. Further research needs to be done to evaluate if – in dependency of the applied metaphors – interpolation of colors and shapes is a suitable procedure.

A further kind of color correction for the metaphoric icon is the adaption of the background color representing the cluster membership. For the maize icon, two modi support different exploration tasks. To explore spatial cluster distributions, the background of the icon can be completely filled out. On the other hand, the cluster membership can be mapped to the surrounding rectangle border colors only, keeping the rest of the background transparent. This allows to correlate the parameter values with other features on the map, e.g. a height field or other geographic information.

## 4. Implementation and results

The presented techniques have been implemented in a prototypical system with a variety of interaction possibilities. The modular system can be easily adapted to other 2D clustered multi-variate data sets and other image metaphors. Fig. 7 shows the GUI of the system.

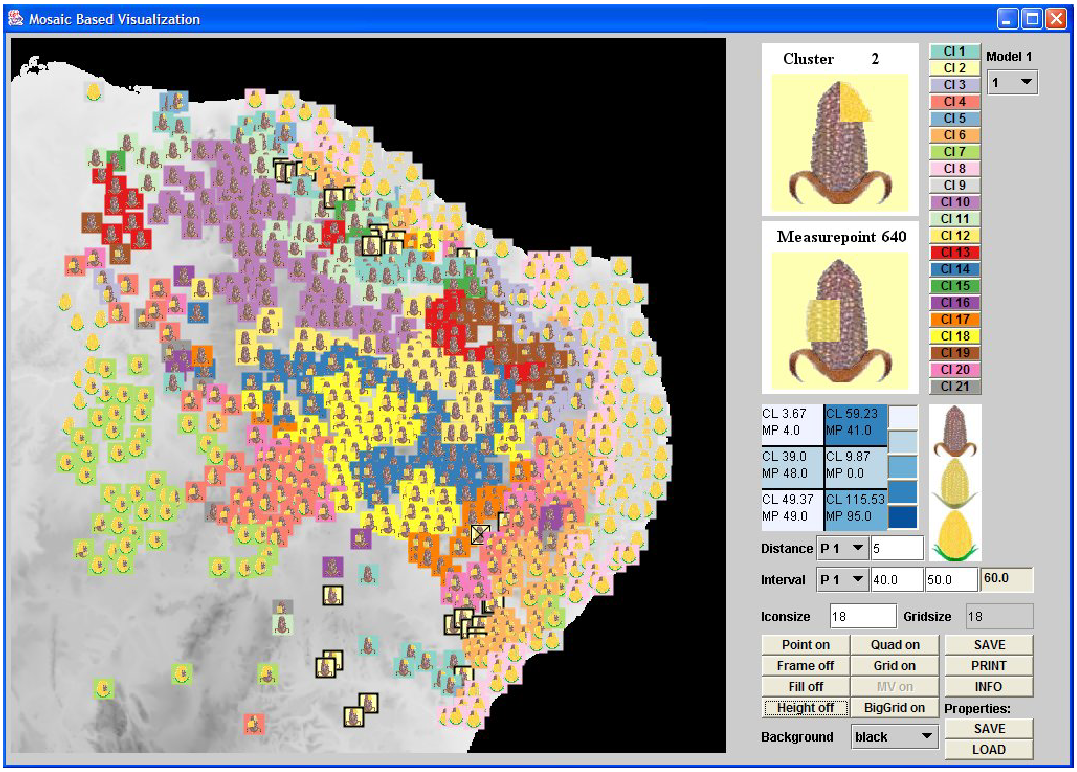



**Figure 7. The mosaic framework: Image Mosaic map (left) and parameter control and legends (right); a certain measurement station and all the stations of the same cluster have been selected**

The user interface consists of a visualization window (Fig. 7 left), and a GUI area showing legends and parameters (Fig. 7 right). To obtain detailed information about clusters and their properties, a zoomed icon displays the currently selected cluster and its mean values in the top left part of the GUI area. To gain details about measurement stations, the zoomed icon for the currently selected measurement station is drawn below. Alongside to these two icons, the legend for the cluster colors is presented. On the left hand side of this legend the user can change the cluster "model" and, thus, can compare various clusterings calculated for the data set. This allows a statistical evaluation of the deployed proximity measures and cluster algorithms.

Below the icon for the currently selected station, the numerical values of the selected cluster's mean parameters and the selected station parameters are represented. Additionally – to enable users to evaluate the homogeneity of a certain cluster – the differences are displayed using white-blue color coding.

The image icon was designed on the base of thresholds distinguishing between good and bad maize harvest conditions. These thresholds can be interactively adapted and, thus, trigger the mosaic tiles on the map. The base image icons for good and bad conditions (Fig. 2) are depicted on the right hand side of the numerical values of the currently selected measurement stations and the currently selected cluster and their differences, below the cluster color legend.

Moreover, a wide variety of other parameters can be set. This includes grid size, icon size, background color, layout style, icon calculation and others. Furthermore, to reproduce successful parameterizations, a certain parameter combination can be stored and reloaded.

Altogether, we developed a system for metaphor-based Image Mosaics that supports a variety of exploration tasks (see tasks from 2.3). First, general cluster information can be easily perceived. Especially the multi-resolution layout gives a fast overview of local cluster distributions, not merging clusters together (see Fig. 5 right). Extreme conditions and the referring clusters can be easily recognized. Second, individual parameters, their spatial distributions and their dependencies can be easily explored. In this context, the regular layout (see Fig. 5 center and Fig. 6) is suited to get a fast overview of the spatial distribution of the parameter values. In the costal regions and in the south eastern parts, very good conditions for the maize harvest are predicted. On the other hand, in a large area from the north-west to the south-east, the risk for a potential total yield loss is high. Third, to identify and explore individual measurement stations together with their clusters, the scattered layout is especially

suited (see Fig. 5 left and Fig. 7). The user can interactively focus on a certain measurement station, study its parameter values and compare it with the cluster mean values and with other measurement stations of this cluster.

In first tests, climate researchers received new insights into the data set that could not be achieved by earlier techniques (see Fig. 1). For instance, the mentioned spatial distribution of regions with high risk of a drought for maize cultivation can be separated very well from areas with a low risk, and the single risk causes (represented by the parameters) can be identified and compared directly. Thus, climate researchers feel that our approach has high potential for the estimation of extreme climate conditions in general. A further achievement is that the approach supports to acquire information at all levels of detail, from an overview of the whole data set (using a regular layout), via regional distributions (e.g., using multi-resolution layouts) to detailed information about station parameters and cluster properties. "Typical" stations, regions and clusters as well as outliers can be examined.

## 5. Conclusions and future work

In this paper, we have investigated the applicability of mosaic-based methods to enhance icon-based visualizations. Based on the example of a practical agricultural data set, we introduced a new image metaphor and discussed how methods well-known from Image Mosaics, such as image layouts and color adaption, can be used for visualization purposes. Based on these considerations, we described our modular mosaic-based visualization system and presented first results for the application background.

Finally, further research has to be done to explore the application, interpretation and reusability of our approach. First tests with users gave promising insights, however, we have to further test the system with other data sets and suitable metaphors. Especially, this includes to compare the usability of metaphors with non-metaphoric icon techniques. Moreover, we want to extend spatial cluster exploration. This includes to enable users to compare two clusterings in one image and to visualize cluster variances in addition to the currently displayed mean values. Furthermore, we plan to apply focus+context techniques in our framework and to evaluate their use to metaphoric icons on maps.

## Acknowledgements

The authors thank Mario Baalcke for implementing the Mosaic Visualization System. Furthermore, we thank our partners Uwe Böhm and Michael Flechsig from the Potsdam Institute of Climate Impact Research for providing the data set and for helpful discussions regarding its mosaic-based visualization.

## References

[1] J. Beddow. Shape Coding of Multidimensional Data on a Microcomputer Display. In *Proceedings of the IEEE Visualization conference*, pages 238–246, 1990.

[2] H. Chernoff. The Use of Faces to Represent Points in k-Dimensional Space Graphically. *Journal of American Statistical Association*, 68:361–368, 1973.

[3] M. C. Chuah and S. G. Eick. Information Rich Glyphs for Software Management Data. *IEEE Computer Graphics and Applications*, 18(4):24–29, July/August 1998.

[4] Adam Finkelstein and Marisa Range. Image mosaics. Technical Report TR-574-98, Princeton University, Computer Science Department, March 1998.

[5] D. A. Keim and H.-P. Kriegel. Visualization Techniques for Mining Large Databases: A Comparison. *IEEE Transactions on Knowledge and Data Engineering*, 8(6):923–938, December 1996.

[6] H. Levkowitz. Color Icons: Merging Color and Texture Perception for Integrated Visualization of Multiple Parameters. In *Proceedings of the IEEE Visualization conference*, pages 164–170. IEEE Press, 1991.

[7] T. McKenna and G. R. Arce. New image mosaic structures. Technical report, Department of Electrical and Computer Engineering, University of Delaware, 2000.

[8] T. Nocke, H. Schumann, and U. Boehm. Methods for the Visualization of Clustered Climate Data. *Computational Statistics*, 19(1):75–94, 2004.

[9] R.M. Pickett and G. G. Grinstein. Iconographics Displays for Visualizing Multidimensional Data. In *Proceedings of the IEEE Conference on Systems, Man, and Cybernetics*, pages 514–519, 1988.

[10] X. Shen and P. Eades. Using MoneyTree to Represent Financial Data. In *Proceedings of IEEE Information Visualization, IV'04, London*, pages 285–289, 2004.

[11] J. Stasko, T. Miller, C. Plaue, Z. Pousman, and O. Ullah. Personalized Peripheral Information Awareness through Information Art. In *Proceedings of the International Conference on Ubiquitous Computing*, pages 18–35, September 2004.

[12] T. Strothotte and S. Schlechtweg. *Non-Photorealistic Computer Graphics: Modeling, Rendering, and Animation*. Morgan Kaufmann, 2002.

[13] M.O. Ward. XmdvTool: Integrating Multiple Methods for Visualizing Multivariate Data. In *Proceedings of IEEE Visualization conference*, pages 326–333, 1994.

[14] M.O. Ward. A taxonomy of glyph placement strategies for multidimensional data visualization. *Palgrave Information Visualization*, 1:194–210, 2002.

# G

## CONNECTING TIME-ORIENTED DATA AND INFORMATION TO A COHERENT INTERACTIVE VISUALIZATION

Bibliographische Angaben:

Ragnar Bade, Stefan Schlechtweg, und Silvia Miksch: „Connecting Time-Oriented Data and Information to a Coherent Interactive Visualization“. In: Elisabeth Dykstra-Erickson und Manfred Tscheligi (Hrsg.), *Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI04)*. ACM Press, New York, 2004. S. 105–112.

Abstract:

Abstract In modern intensive care units (ICUs), the medical staff has to monitor a huge amount of high-dimensional and timeoriented data, which needs to be visualized user- and taskspecifically to ease diagnosis and treatment planning. Available visual representations, like diagrams or charts neglect the implicit information as well as a-priory or associated knowledge about the data and its meaning (for example, 38.5°C (101.3°F) is moderate fever and 41°C (105.8°F) is critical fever). Another challenge is to provide appropriate interaction techniques to explore and navigate the data and its temporal dimensions. In this context one major challenge is to connect time-oriented data and information to a coherent interactive visualization. In this paper we present different interactive visualization techniques which enable the users to reveal the data at several levels of detail and abstraction, ranging from a broad overview to the fine structure. We will also introduce a time visualization and navigation technique that connects overview+detail, pan+zoom, and focus+context features to one powerful time-browser.

Betreute Diplomarbeit:

Ragnar Bade: *Methoden zur Visualisierung von und Interaktion in zeitbasierten Patientendaten und Behandlungsplänen*. Diplomarbeit am Institut für Simulation and Graphik der Otto-von-Guericke-Universität Magdeburg, 2002.

# Connecting Time-Oriented Data and Information to a Coherent Interactive Visualization

**Ragnar Bade and Stefan Schlechtweg**
Otto-von-Guericke University of Magdeburg
Department of Simulation and Graphics
Universitätsplatz 2, D-39106 Magdeburg
{rbade|stefans}@isg.cs.uni-magdeburg.de

**Silvia Miksch**
Vienna University of Technology
Institute of Software Technology and Interactive Systems
Favoritenstrasse 9 - 11 / 188, A-1040 Vienna
silvia@ifs.tuwien.ac.at

## Abstract
In modern intensive care units (ICUs), the medical staff has to monitor a huge amount of high-dimensional and time-oriented data, which needs to be visualized user- and task-specifically to ease diagnosis and treatment planning. Available visual representations, like diagrams or charts neglect the implicit information as well as a-priory or associated knowledge about the data and its meaning (for example, 38.5°C (101.3°F) is moderate fever and 41°C (105.8°F) is critical fever). Another challenge is to provide appropriate interaction techniques to explore and navigate the data and its temporal dimensions. In this context one major challenge is to connect time-oriented data and information to a coherent interactive visualization. In this paper we present different interactive visualization techniques which enable the users to reveal the data at several levels of detail and abstraction, ranging from a broad overview to the fine structure. We will also introduce a time visualization and navigation technique that connects *overview+detail*, *pan+zoom*, and *focus+context* features to one powerful time-browser.

**Categories & Subject Descriptors:** H.5.m [Information Interfaces and Presentation]: Miscellaneous.

**General Terms:** Design, Human Factors.

**Keywords:** Visualization, health care, user interface design, information visualization, temporal data modeling and abstraction, medical application: intensive care units.

## INTRODUCTION
In medical domain, e.g., for treating patients in intensive care units (ICUs), the medical staff has to interpret and respond rapidly to a large number of clinical parameters and has to select an appropriate treatment for the patient among many different options and alternatives. According to the temporal dimension in medical applications, each data point is defined by its value, occurrence time, valid time, measurement deviation tolerance and trustability. As one consequence, the almost overwhelming amount and the high-dimensional, time-oriented structure of this data need to be visualized user- and task-specifically to ease diagnosis and treatment planning. Challenges are the visualization of this n-dimensional data, in particular of its temporal-dimensions, since time and the temporal dimensions need to be treated differently than other features. For example, time cannot be reversed, data is linear as well as cyclical over time and the data is collected and expressed in different granularities of time (e.g., in days, hours, minutes, seconds and milliseconds).

Abstraction of n-dimensional data into 2- or 3-dimensional visualizations is widely used but including time needs special treatment to capture the temporal properties. Consequently, new solutions are necessary to capture n-dimensional data and treating the temporal dimensions differently.


*CHI 2004,* April 24–29, 2004, Vienna, Austria.


## STATE OF THE ART
During the last years various approaches have been developed to visualize time-oriented data in general. Also in the medical field, different approaches were introduced to capture the complex representation of time-varying patients' data. Here several dimensions need to be visualized according to the medical problems, diagnoses, or treatment actions.

One technique to visualize personal histories and other temporal data is the *LifeLines* approach [9,10]. Starting from an overview of the whole chronicle, zooming provides more details. One fundamental component of visualizing temporal data using *LifeLines* is the use of timelines [14], where colored horizontal bars are plotted over the required period of time of an assigned action or event. The length of those bars is derived from the duration time and the width/height from the significance of that action or event. In contrast discrete events are represented by icons. Color is used to visualize specific sections or relationships. In comparison to tabular representations *LifeLines* lead to faster response times, better first impressions of the data and higher recall rates [1]. However, there is a lack of visualizing continuous data over time (e.g., a fever curve).

The *Graphical Summary of Patient Status* by Powsner and Tufte [11] visualizes a lot of "fever curves" in small repeated graphs with identical formats. The data in these small multiples is scaled to five graphically same sized qualitative ranges (critically reduced, reduced, normal, elevated and critically elevated). Using scaled values simplifies rating and interpretation since big vertical displacement corresponds to clinical significance. Unfortunately, this visualization scales every qualitative level by a different amount. As a result the visual distance between two values in different quality levels is not proportional to the numerical distance. In addition to the qualitative scales used, the temporal *focus+context* visualization of the *Graphical Summary of Patient Status* is very convincing. Due to a non-linear time-scale, a graphical summary is achieved, compressing two years of data into a context for recent trends. Additional ideas concerning data summaries are presented by Shahar and Cheng [12]. They utilize derived temporal data abstractions to communicate complex data dependencies.

Such *focus+context* techniques are visualization and interaction methods adapted to the human visual perception and cognition. According to the human foveal vision and context-oriented cognition [15,4] rich detailed foci are displayed in the context of bigger coarse parts. Comparative evaluations of *focus+context*, *pan+zoom* and *overview+detail* techniques attest *focus+context* techniques a leading position [3,7,8] but also address disadvantages in respect of *information scent* [7,8].

## OUR VISUALIZATION APPROACH

In the next sections we will introduce our solutions for the visualization of high-dimensional, time-oriented data as well as our time visualization and navigation techniques.

### Visualizing Time-Oriented Data

The alternative visual representations of time-oriented data are closely connected to the available display space and the required information content. The information space and display space are tied by the complexity of the data to be visually explored and the tasks the users want to accomplish. Therefore, we introduce different visualizations ordered by the required display space and automatically ordered by the attainable information content. Afterwards we explain the cooperation and interaction of these visualization techniques.

#### *Visualizing Qualitative Scales*

Two techniques are useful to visualize qualitative data or respectively to visualize qualitatively abstracted data. The first one "*color-coded timelines*" is similarly used in the *LifeLines* approach and the second one "*height-coded timelines*" has been developed as a descendant of the first.

In the original *LifeLines* approach, colored horizontal bars are plotted over the required period of time of an assigned action or event. In that approach, color is used to visualize specific fractions or relationships. In our case of visualizing data, color can be used to represent regions of different qualitative characteristics. As an example, critical fever values can be colored red, moderate fever yellow and normal temperature green. In Figure 1 critical fever periods can be easily located. Such kind of *color-coded timelines* provides a fair amount of information by using only a minimum of display space. They can be minimized to a minimum of only one pixel of screen height by carrying the same amount of information. However, appropriate colors have to be found for the different qualitative levels in the data. To carry on with the fever example, now the question is how reduced and critically reduced temperature can intuitively be colored.

Consequently, we developed *height-coded timelines* as a descendant of *color-coded timelines*. They can be used without assigning color to the qualitative levels of the data. *Height-coded timelines* represent different qualitative levels like bar-charts by using different heights of the *timelines*. Figure 2 shows the same fever data as Figure 1. As a result of using *height-coded timelines* the ordinal scale of the data is visualized. As you can see in Figure 3 this visual encoding allows to represent intuitively the full range from critically reduced temperature up to critical fever. To enhance *height-coded timelines* they can also be colored additionally. Thus, the alarming nature of red regions can be retained (see Figure 4).

By using height, the amount of information content has been increased, but at the same time the required display space has also been increased slightly. To sum up *height-coded timelines* can be minimized to a minimum of *n* pixels (where *n* is the number of different qualitative levels) and they provide additional information about the ordinal scale of the data.


**Figure 1. *Color-coded timeline* representation of a fever curve.**

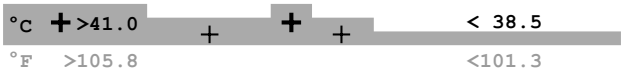



**Figure 2. *Height-coded timeline* representation of the same fever curve as in Figure 1.**


**Figure 3. *Height-coded timeline* of critically elevated, elevated, normal, reduced and critically reduced qualitative levels.**

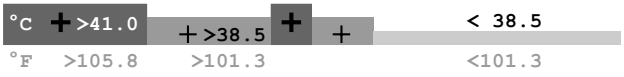



**Figure 4. Colored version of the *height-coded timeline* in Figure 2.**

### *Qualitative - Quantitative Hybrids*

Hybrid techniques can combine intuitive and easily to interpret qualitative representations with more detailed quantitative representations. A quantitative data stream enhanced with color-coded qualitative regions is visualized in Figure 5. This visual representation eases the recognition of critical periods as well as of concrete fluctuations in the data. Especially flat data streams become more obvious since the colored regions aid the perception of the otherwise small variations in the data.

The data stream can also be split in to its qualitative regions without using color, as illustrated in Figure 6. In this case, vertical lines divide the different parts from each other. In contrast to common verticall markings (like, using strokes starting at the y-axis to split different regions in a chart) the horizontal splitting used (by applying color regions or vertical lines) is more appropriate for visualizing flat data streams.

Another hybrid approach, used in the *Graphical Summary of Patient Status* [11] was described above. As mentioned there, the data is scaled to graphically same sized qualitative ranges. Unfortunately, due to this scaling, the visual distance between two values in different quality levels is not proportional to the numerical distance. Our visualizations overcome these disadvantages.

### *Visualizing Quantitative Scales*

Quantitative scales offer the ability to read off exact values and relations from the data. Unfortunately, they do not automatically include the knowledge that is included in qualitative scales (e.g., is a particular value critical or not).

A possible solution is to color the y-axis to visualize information about qualitative attributes. Additional horizontal colored lines mark points, where the data values leave one qualitative level and enter another one. The used color is determined by the color of the leaving and entering level as you can see in Figure 7. Direct coloring the data trail as well as marking regions by coloring the background of the visualization is also used.

Another challenge when visualizing quantitative data is the amount of attributes or dimensions assigned to each data point. Among them are the value, the occurrence time, the valid time, measurement deviation tolerances, and trustability.

### *Visualizing Data Points and its Dimensions*

The occurrence time of a data point is marked at the corresponding point in time. According to the granularity of the specified point it is represented by a small mark or by an extended mark capturing the uncertainties as shown in Figure 8.

The valid time of a data point is represented by a horizontal line indicating a valid value over the specified period of time (Figure 8). The missing of a valid value is indicated by filling the whole region of time with a gray box (Figure 11).

If data values suffer from measurement deviations (e.g. ± 1%) then the corresponding range of values is marked as illustrated in Figure 9.

The trustability of a data value is represented by the fill level of a green bar. This vertical bar is plotted right above the data value itself as can be seen in Figure 10.

It is not always the case that a data point possesses all these attributes. Nevertheless, the developed techniques can depict them all at the same time in the same visualization (Figure 11 bottom).

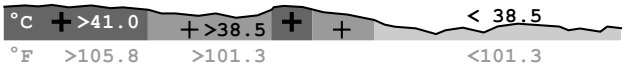



**Figure 5. Flat quantitative data display with colored qualitative regions.**

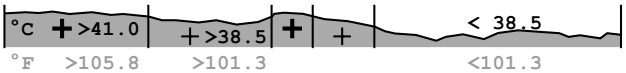



**Figure 6. Flat quantitative data display split into qualitative regions without using color.**

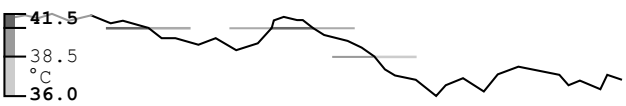



**Figure 7. Fever curve with a qualitative colored y-axis and marked qualitative level crossings.**

**Figure 8. Representation of a data point with a more coarse (left) and a more precise (right) occurrence time than the actual timeline scale. Additionally a horizontal line indicates the valid time of the data point.**

**Figure 9. Visualization of measurement deviation by extending the representations as shown in Figure 8.**

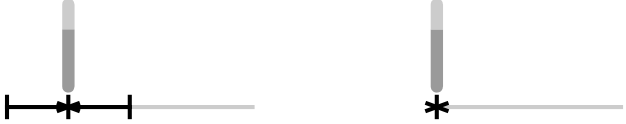


**Figure 10. Trustability of data points represented by a green filled bar. (extending Figure 8).**

*Visualizing High-Frequency Data*

Data points can be connected by straight lines (as shown in Figure 11 bottom) or by interpolation to visualize a contiguous data curve (e.g. a fever curve). Unfortunately, those visualizations reach their limits when it comes to displaying high-frequency data. When displaying high-frequency data many data values have to be laid down on one and the same small display region (e.g. a pixel or a row of pixels). In that case, appropriate visualization techniques are necessary that abstract the data and nevertheless generate an expressive representation.

The common way of visualizing high-frequency data is to calculate and display statistical measures, like average value, median or standard deviation. Thereby information on minima, maxima and critical values, etc. are lost.

To minimize the amount of information loss and to expressively represent the data *Information Murals* [5] can be used (e.g., see Figure 12). Unfortunately, an *Information Mural* represents the distribution of the data very well but neglects information on minima and maxima as well. Additionally, this information and other statistical values can be integrated. We have enhanced the *Information Mural* visualization by displaying minima, maxima, the median and the 25% and 75% percentile as you can see in Figure 13. Thus the data stream can be traced closely, the stray area of 50% of the data (interquartile distance, between the 25% and 75% percentile) as well as minima, maxima and the median can be read off.

Even though the information content is high, the resulting visualizations are rather complex and, hence, hard to interpret. Therefore, to get an even more abstract, but nevertheless expressive representation of high-frequency data, we use a redesign (inspired by Tufte [14]) of *Tukey box-plots* (former known from statistical visualizations). In this redesign (see Figure 14) the space between the minimum and the maximum is filled light-colored at first. Then the space between the 25% and 75% percentile is filled a little bit darker and the median is marked black. Now, a lot of these small *box-plot-bars* can be connected to a full data stream representation (Figure 15).

In combination with the enhanced *Information Murals* and the common way of displaying the median only, the user is provided with multiple views on to the data and, doing so, new insights are granted.

### Interacting with Data and Time

In the sections above the visualization techniques have been described separately. In this section we show how they cooperate and how the user can interact with them.

The major tasks when exploring the data will be searching/browsing in the data (e.g. by filtering and zooming) and in time (e.g. by panning and zooming).

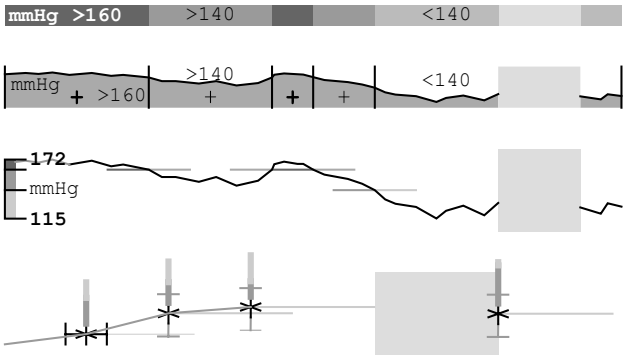



**Figure 11. Gray regions indicate missing of valid data values in any representation.**

**Figure 12. *Information Mural* representation of a high-frequency data stream.**

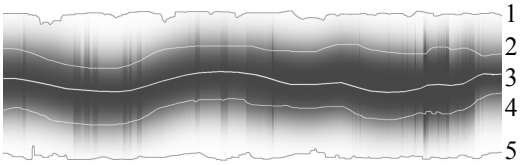



**Figure 13. Extended *Information Mural* representation with (1) maxima, (2) 75% percentile, (3) median, (4) 25% percentile and (5) minima.**

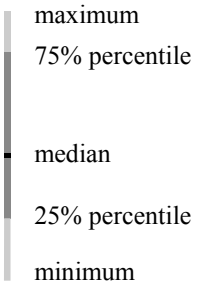



**Figure 14. Very slim redesign of a *Tukey box-plot* inspired by Tufte [14].**

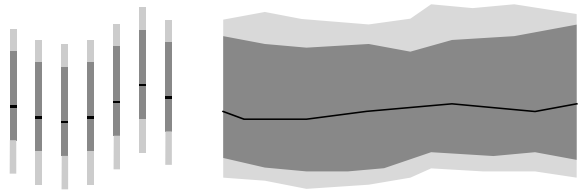


**Figure 15. Visualization of a data stream using the redesigned box-plot of Figure 14. Connecting redesigned box-plots with an other (left) generates a data stream representation as shown right.**

### *Browsing the Data*

According to the limited display space and human cognition and perception capabilities, the visualization techniques mentioned above can be connected to a coherent interactive data browser.

Resizing the data vertically zooms and browses intuitively through the different visualization techniques and abstraction levels of the data. Figure 16 shows the changing visual representation of the data, from a broad overview to the fine structure. By this means, not only is the display space efficiently used but also new exploration and examination methods are provided. Additionally the different visualization techniques automatically explain one another.

The scaling/zooming described is useful on the one hand to provide the right amount of detail for each task and on the other hand to enable the user to explore the data step by step adding one detail after another. Adding more detail only step by step aids to keep the cognitive context when changing the visual representation. Furthermore, the user is able to quickly assign a qualitative level to a quantitative visualization and vice versa, due to smooth zooming between both.

If the user is not familiar with one visual representation then he or she can keep track of changes from a familiar one to the unknown in order to understand the mapping from properties between one and the other.

To read off exact values in every visualization the mouse cursor provides additional information of date and value under the tip of the cursor (Figure 17).

According to the available horizontal display space, the data points are visualized individually or by the above mentioned techniques for high-frequency data. The next section describes how to navigate and browse in time.

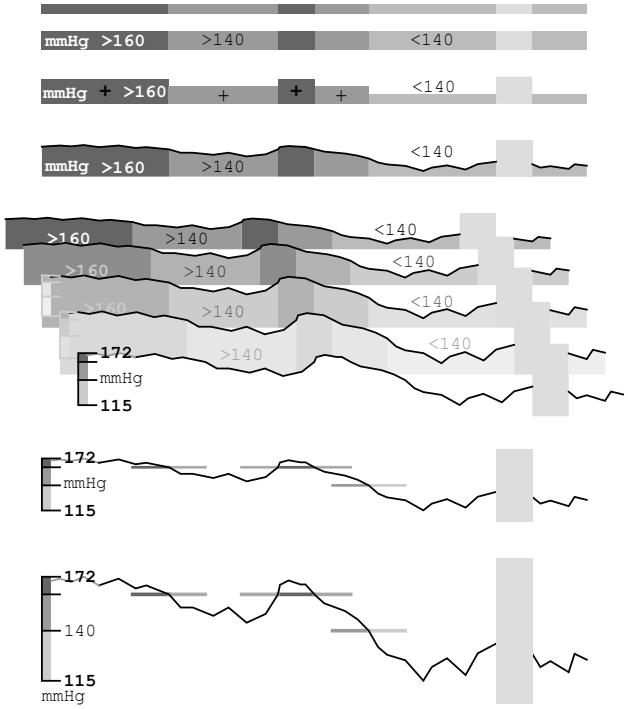



**Figure 16. Steps of resizing/zooming the representation of a data stream from a broad overview to the fine structure.**

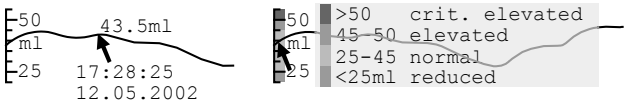



**Figure 17. The mouse cursor provides additional information about exact values, date (left) and about used colors and scales (right).**

### *Browsing the Time*

The time visualization is spread over three connected timelines (see Figure 18). The first (bottom) one provides a fixed overview of the underlying data and its full temporal range. Selecting a sub-range in the first timeline defines the temporal bounds for the second (middle) and third (top) timeline. By interacting with that sub-range you can easily *pan+zoom* in time (Figure 18).

The second (middle) and third (top) timeline provide the ability to add distortion borders in time. These borders can be moved interactively on one or the other timeline to distort the linearity of time visualization (Figure 19). This provides the ability to define *focus+context* regions in an intuitive and interactive way. Achievable distortions can vary from simple *focus+context* regions (Figure 19 bottom-right) to *portals* [6] (Figure 19 bottom-left). Highlighted sharp borders of different time distortions minimize interpretation uncertainties.

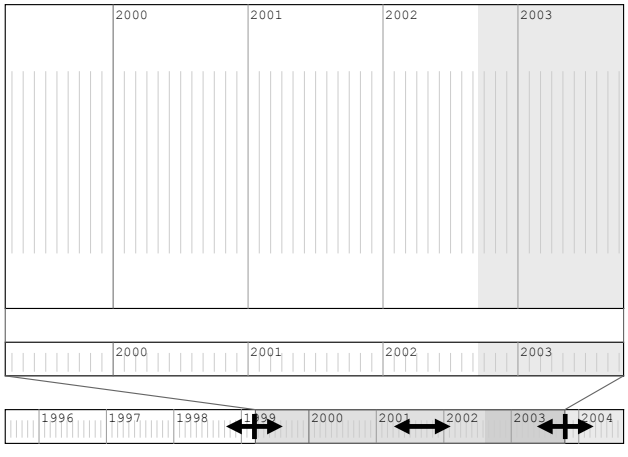



**Figure 18. Timeline interaction: The selected sub-range at the bottom timeline can be moved and rescaled to *pan+zoom* the time range shown in the middle and top timeline.**

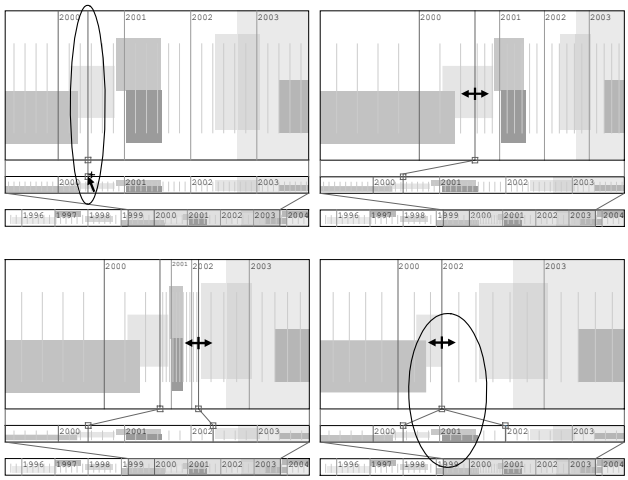


**Figure 19. Interactive timeline distortion: (top-left) added distortion border, (top-right) distort timeline by simple dragging, (bottom-left) distort timeline by two distortion borders, (bottom-right) *portal* distortion.**

The whole temporal range of the data can be browsed by using both or only one technique. Additional *overview+detail* is provided by displaying the same data at all three timelines at different abstraction levels according to the available display space.

A menu with selectable predefined layouts eases the use of and change between several common and preferred distortion configurations.

## SCENARIO: PULMONARY EMBOLISM CARE

This section illustrates the discussed techniques in context of an intensive care unit application and a pulmonary embolism case. Figure 20 shows a screenshot of the implemented prototype for intensive care patient treatment. In addition to the visualization of temporal data, treatment plans have also been included in the image.

Horizontal resizing of a treatment plan results in resizing/zooming of its data content. In doing so the used data visualization techniques change according to the available horizontal display size (as described above in "*Browsing the Data*")

In the screenshot a selected temporal range, from December 2002 to the middle of May 2003, can be seen at the bottom timeline. The patient was not at the ICU the whole period, but only after the admission to the hospital for one day. The first timeline (bottom) shows the whole period of the patient's case history, the second time line (middle) illustrates the period the patient is dealing with pulmonary embolism, and the third timeline (top) depicts the first week of therapeutic interventions in focus, and the remaining time as context. At the second timeline (middle) exact dates can already be seen from the selected region. At the third timeline (top) the first week is scaled to three quarters of the display size and the remaining time of the selected temporal region is compressed to the remaining quarter.

Different visualization techniques are used to represent the temporal patient data. For example a colored hybrid technique is used for the saturation of oxygen in blood (SpO2). *Height-color-coded timelines* as well as quantitative visualizations are also used in the screenshot.

Critical regions at the first day and a smooth stabilizing process over the rest of the week are easy to recognize. Due to the use of *color-coding* it is recognizable (even without medical knowledge) that the saturation of oxygen in blood (SpO2) and the partial thromboplastin time (PTT) were to low and the heart frequency (HF) were to high on the first day.

At the right side of the screenshot the actual patient status as well as information about treatment, involved physicians, etc. are displayed.

## CONCLUSION

In this paper we have presented a novel way of visualizing time-oriented data. We have developed several visualization techniques which enhance the understanding of qualitative and quantitative characteristics of a given data stream. The challenges in that circumstance were to support the exploration tasks of users with different tasks in mind and to capture as much as possible temporal information on a limited display space without loss of overview and detail. In this context Shneidermann proposed the Visual Information Seeking Mantra: "Overview first, zoom and filter, then details-on-demand" [13]. The proposed solution enables those features by particular visualization and navigation techniques, which takes the temporal dimensions of the data into account.

We have shown, how these techniques can be used in the medical domain, in particular we have shown how to explore patients with pulmonary embolism. However, our proposed techniques are not only applicable in the medical field. The techniques are applicable to any applications, which meet the same problem characteristics (e.g. complexity of data, tasks, users). We have applied the techniques to the field of stock-exchange too [2].

## ACKNOWLEDGEMENTS

We want to acknowledge and thank Arno Krüger for hunting harts in this paper as well as all other colleges for their contributions.

## REFERENCES

[1] Alonso, Diane L.; Rose, Anne; Plaisant, Catherine.; Norman, Kent L.: Viewing Personal History Records: A Comparison of Tabular Format and Graphical Presentation Using LifeLines. In *Behaviour and Information Technology*, 17(5), pp. 249-262, 1998.

[2] Bade, Ragnar: Methoden zur Visualisierung von und Interaktion in zeitbasierten Patientendaten und Behandlungsplänen. Master thesis, Department of Simulation and Graphics, Faculty of Computer Science, University of Magdeburg, Germany, 2002.

[3] Baudisch, Patrick; Good, Nathaniel; Bellotti, Victoria; Schraedley, Pamela: Keeping Things in Context: A Comparative Evaluation of Focus Plus Context Screens, Overviews, and Zooming. In *Proceedings of CHI 2002, ACM Conference on Human Factors in Computing Systems*, ACM Press, pp. 259-266, 2002.

[4] Furnas, George W.: Generalized fisheye views. In *Proceedings of CHI'86, ACM Conference on Human Factors in Computing Systems*, SIGCHI Bulletin, pp. 16-23, 1986.

[5] Jerding, Dean F.; Stasko, John T.: The Information Mural: A Technique for Displaying and Navigating Large Information Spaces. In *IEEE Transactions on Visualization and Computer Graphics*, 4(3), pp. 257-271, 1998.

[6] Perlin, Ken; Fox, David: Pad: An alternative approach to the computer interface. In *Proceedings of SIGGRAPH'93, ACM Conference on Computer Graphics and Interactive Techniques*, pp. 57-64, 1993.

[7] Pirolli, Peter; Card, Stuart K.; Van Der Wege, Mija M.: The Effect of Information Scent on Searching Information Visualizations of Large Tree Structures. In *Proceedings of AVI 2000, ACM Conference on Advanced Visual Interfaces*, ACM Press, pp. 161-172, 2000.

[8] Pirolli, Peter; Card, Stuart K.; Van Der Wege, Mija M.: Visual information foraging in a Focus+Context visualization. In *Proceedings of CHI 2001, ACM Conference on Human Factors in Computing Systems*, 2001.

[9] Plaisant, Catherine; Milash, Brett; Rose, Anne; Widoff, Seth and Shneiderman, Ben: LifeLines: Visualizing Personal Histories. In *Proceedings of CHI 96, ACM Conference on Human Factors in Computing Systems*, volume 1 of PAPERS: Interactive Information Retrieval, ACM Press, pp. 221-227, 1996.

[10] Plaisant, Catherine, Mushlin, Richard, Snyder, Aaron, Li, Jia, Heller, Dan, Shneiderman, Ben: LifeLines: Using Visualization to Enhance Navigation and Analysis of Patient Records. Revised version in *American Medical Informatic Association Annual Fall Symposium AMIA*, pp. 76-80, 1998.

[11] Powsner, Seth M; Tufte, Edward R.: Graphical Summary of Patient Status. In *The Lancet*, 344, pp. 386-389, 1994.

[12] Shahar, Yuval; Cheng, Cleve: Model-Based Visualization of Temporal Abstractions. In *Computational Intelligence*, 16(2), pp. 279-306, 2000.

[13] Shneiderman, Ben: The Eyes Have It: A Task by Data Type Taxonomy for Information Visualizations. University of Maryland, Department of Computer Science, Human-Computer Interaction Laboratory and Institute for Systems Research, Maryland, USA, Technical Report, CS-TR-3665, ISR-TR-96-66, 1996.

[14] Tufte, Edward R.: The Visual Display of Quantitative Information. Graphics Press, Cheshire, CT, 1983.

[15] Wandmacher, Jens: Softwareergonomie - Mensch-Computer-Interaktion Grundwissen. Walter de Gruyter Verlag, Berlin, 1993.

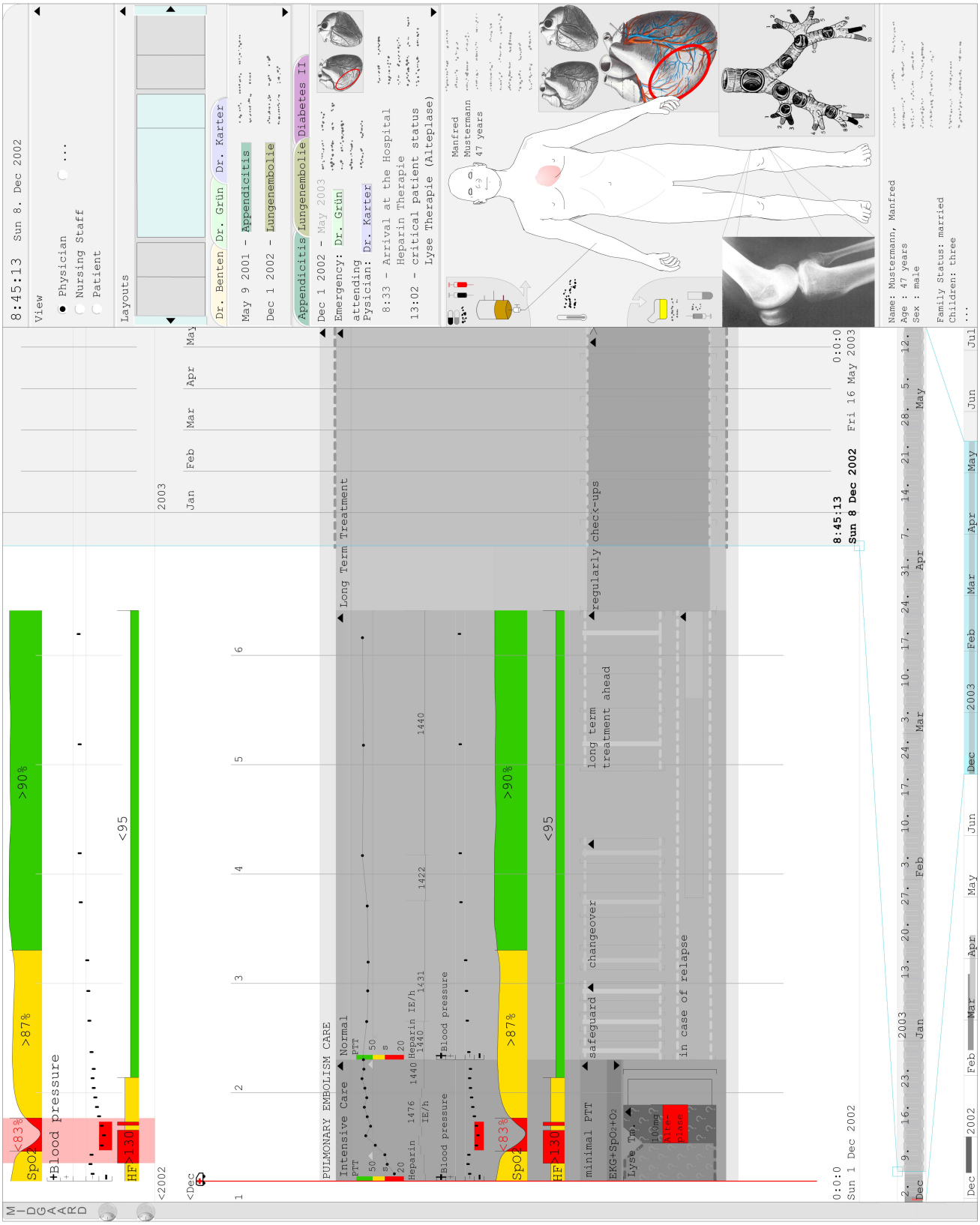



**Figure 20. Screenshot of the implemented prototype for intensive care and long term patient treatment.**

# H

## INTERACTIVE TREEMAPS WITH DETAIL ON DEMAND TO SUPPORT INFORMATION SEARCH IN DOCUMENTS

Bibliographische Angaben:

Stefan Schlechtweg, Petra Schulze-Wollgast und Heidrun Schumann: „Interactive Treemaps With Detail on Demand to Support Information Search in Documents“. In: Oliver Deussen, Charles Hansen, Daniel A. Keim und Dietmar Saupe (Hrsg.), *Data Visualization 2004. Eurographics/IEEE TCVG Visualization Symposium Proceedings*. Eurographics Association, Aire-la-Ville, 2005. S. 121–128.

Abstract:

This paper addresses the issue of how information visualization techniques can be used to assist full-text search in electronic documents. Our approach supports multiple term queries with interactive treemaps. We use a treemap to visualize the basic structure of the document and exploit color coding to show the distribution of query terms on various levels of the hierarchy. Furthermore, we include filtering techniques to concentrate on those parts of the structure that actually contain the requested information, and, finally provide interactive tools to give access to detailed information whenever the user wishes.

Weitere Veröffentlichung:

Stefan Schlechtweg, Petra Schulze-Wollgast und Heidrun Schumann: „Visual Support for Keyword Search in Electronic Documents“. In: Thomas Schulze, Stefan Schlechtweg und Volkmar Hinz (Hrsg.), *Proceedings Simulation und Visualisierung 2004*. SCS Europe, Erlangen · San Diego, 2004. S. 215–225.

Joint EUROGRAPHICS - IEEE TCVG Symposium on Visualization (2004)
O. Deussen, C. Hansen, D.A. Keim, D. Saupe (Editors)

# Interactive Treemaps With Detail on Demand to Support Information Search in Documents

Stefan Schlechtweg[1], Petra Schulze-Wollgast[2] and Heidrun Schumann [2]

[1] Department of Simulation and Graphics, Otto-von-Guericke University of Magdeburg, Universitätsplatz 2, D-39106 Magdeburg, Germany
[2] Institute of Computer Science, University of Rostock, Albert-Einstein-Straße 21, D-18055 Rostock, Germany

**Abstract**

*This paper addresses the issue of how information visualization techniques can be used to assist full-text search in electronic documents. Our approach supports multiple term queries with interactive treemaps. We use a treemap to visualize the basic structure of the document and exploit color coding to show the distribution of query terms on various levels of the hierarchy. Furthermore, we include filtering techniques to concentrate on those parts of the structure that actually contain the requested information, and, finally provide interactive tools to give access to detailed information whenever the user wishes.*

Categories and Subject Descriptors (according to ACM CCS): H.5.2 [Information Interfaces and Presentation]: User Interfaces – Graphical user interfaces (GUI) I.3.6 [Computer Graphics]: Methodology and Techniques – Interaction techniques

**Keywords:** information visualization, hierarchy visualization, treemaps, full-text search, filtering, detail-on-demand

## 1. Introduction

When working with electronic documents, full-text search is one of the most often performed tasks. In fact, it is an important advantage of electronic texts over their paper counterparts. The problem of full-text search can be more formally stated as follows. Given a text and a set of query terms, find all occurrences of the terms in the text and give the reader the possibility to locate the respective positions in the text. Connected herewith is the assessment whether the context of the located finding fulfills the information need of the reader and to which extend. While the first part of the problem – finding the occurrences of given terms in a text – has been solved, there are still open questions connected with the second part.

In our work we concentrate on searches in well structured texts. In contrast to almost unstructured texts, like novels, there are other requirements. Such structured texts (like textbooks, conference proceedings, scientific works, etc.) are usually not read cover-to cover but instead certain chapters are treated in an order determined by the reader's interest. To speed up full-text searches in this case, we need to tell the user, which structural units contain relevant information. Therefore, it is not sufficient to simply show these locations of query terms and let the user jump back and forth between them. The reader is more concerned with identifying larger structural units of the document that describe the topic characterized by the given term collection.

Given this intention, the following issues are important:

- quickly finding the "best" matches for the issued query,
- easily assessing to which extent a unit of text matches the query (i.e., which terms are contained how often in a unit of text) by connecting structural information of the text with the query result.

In this paper we propose to use Information Visualization techniques to incorporate the above mentioned issues in the process of full-text search. More specifically, we use treemaps to give the user access to structural information and enrich these treemaps with interactive techniques to relate the resulting location to the context on different structural levels of the document and to provide detail on demand.

The remainder of the paper is organized as follows. In Section 2 we give an overview of related material before we state our approach in Section 3. Starting with the main prob-

lem. Section 4 shows how treemaps can be used to integrate structural information and full-text search results. In order to enhance the visualization, alternative presentations are introduced in Section 5. Further user support is given by providing detail-on-demand techniques in Section 6. We then conclude the paper and give some ideas for future work.

## 2. Related Work

We are looking for visualizations of term distribution in highly structured scientific texts. Therefore, the research presented in this paper is especially connected to two areas:

1. visualization of hierarchical structures and
2. visualization of query results in documents.

Several visualization methods exist that emphasize hierarchical relations (see Keim et al. for an overview [KMS02]). We distinguish between explicit methods, i.e., drawing a graph of the hierarchy with edges and nodes, and implicit space-filling methods, that show hierarchical relations by spatial arrangements of nodes.

For visualizing a document hierarchy, the child-parent relations between structural units can be regarded as containment relations, i.e., the document *contains* several chapters, one of these *contains* several sections, and so on. Therefore, implicit hierarchy visualization methods with a nested arrangement like treemaps [Shn92] are usable. Here, the containment relations can be deducted from the spatial arrangement of the elements of the visualization. However, hierarchy visualization methods focus on presenting the structure itself, especially dealing with large structures consisting of a huge amount of nodes (e.g., as shown by Fékete and Plaisant [FP02]). Our main focus is on the visualization of certain complex features and attributes (regarding the query term distribution) in connection with such a structure visualization.

A standard way of visualizing the distribution of query terms in a document as it is exploited by various application programms (as, for instance, internet search engines) highlights the terms using different text attributes within a visualization of the document. The user may then browse through the text or use a "find next/previous" function to jump between the locations in a sequential manner. In some cases, this is sufficient if the reader is interested in finding all locations of a given (set of) term(s).

Besides emphasizing the query terms in a textual representation of the document, there are approaches that graphically visualize information about the frequency of occurrence of terms after a search in a document collection. These approaches first try to find documents that contain specific terms. Spence describes two general approaches for graphically presenting the results of that search [Spe01]: a map-like fashion as Themescapes, e.g., [WTP*95], or a point cloud version as Galaxies, e.g., [Ren94]. Both approaches are based on a special arrangement of the documents either on a map or in a multidimensional feature space (given by a set of keywords). Beside these general strategies several techniques were developed to address special problems. For example, changes in the frequency of terms (and, hence, the change of topic) over time in a large document collection can be visualized as ThemeRivers [HHWN02]. On the other hand, Tilebars [Hea95] were developed to show the term distribution in large text documents. The document is analyzed for topic changes which impose a "tiling" – a sequential partitioning – on the document. A special visualization technique is then used to show the distribution of query terms in this text. The user interface allows the specification of the query terms and the search in multiple documents.

While the techniques mentioned so far were primarily for document collections, in our case, the locations of query terms within a probably long single text have to be presented. A possible solution for this problem are so called Information Murals as introduced in [JS98]. An Information Mural is a two-dimensional reduced representation of the entire document that fits entirely in a display region. Jerding and Stasko have shown the use of such representations for visualizing the distribution of query terms, however, without further information about the context the user is not able to assess whether a location of a term fits his or her information needs.

## 3. Approach and Challenge

Our approach supports multiple term queries with interactive treemaps. We have chosen to use treemaps since they provide an optimal representation of the containment relations that form the document structure. The chapter structure can be extracted while parsing the text. We have further subdivided the text up to the level of paragraphs. Special parts, such as images, tables, equations, and inset text boxes are treated as paragraphs. As an example throughout this paper serves a textbook on computer graphics (cf. [SS02]).

Our approach can be described as follows:

1. using a treemap to visualize the basic document structure,
2. exploiting color coding to show the distribution of query terms on various levels of the hierarchy,
3. including filtering techniques to concentrate on those parts of the structure that actually contain the requested information, and, finally,
4. providing interactive tools to give access to detailed information whenever the user wishes.

Treemaps have been introduced by Shneiderman [Shn92] and the use of treemaps for different tasks and problems is well-known. Treemaps are introduced as a 2D space-filling representation for arbitrary trees where child nodes are represented as rectangles that are contained within the rectangle representing the father node. Several extensions to the original treemap algorithm were published mainly addressing layout problems. Considering the original "slice-and-dice"

treemap algorithm it presents an intuitive layout where the structure of the hierarchy can be seen easily. Also, the ordering of the underlying data is preserved in the visualization. Bruls et al. present "Squarified Treemaps", an algorithm that generates layouts in which the rectangles approximate squares [BHvW00]. However, the ordering is no longer preserved. Shneiderman and Wattenberg present "Ordered treemap layouts" that adhere to the ordering in the underlying data even in the squarified form [SW01].

Our aim is to establish a highly interactive treemap that fulfills the "Information Seeking Mantra" *Overview first, zoom and filter, then details-on-demand* stated by Shneiderman [Shn96]. Hence, we allow different views with different levels of detail for information representation on the basis of a treemap. We use the "slice-and-dice" treemap layout since it provides a sufficient base for our work. Moreover, we integrate alternative presentation styles to highlight some regions (e.g. by using cushion treemaps [vWvdW99] for special nodes of interest), or to simplify other regions that do not contain relevant terms. Doing so, we provide a rich functionality for an intuitive exploration of structured documents and for a well directed search for information. There exist various possible application scenarios, as for instance an intuitive search interface for concept indexes, or for full-text queries of documents in a digital library. In the following we will outline our approach in more detail.

## 4. Combining Structure and Search Results – Overview

Given an electronic document containing a well structured text, the structure of the text can be extracted and forms a tree that is to be visualized. Using standard treemaps, the text structure becomes clearly visible. The nested box character of the document structure can even be emphasized by using a framed version of the standard treemaps. As can be seen in Figure 1, paragraphs of text (the leaf nodes of the hierarchy) are displayed as rectangles and all inner nodes (chapters, sections, etc.) frame these so that the nesting becomes clear. The use of treemaps yields the possibility to encode various other information at the same time as, for example, a characterization of the contents of a node as can be seen in Figure 2. Here, each leaf node is characterized as to whether it is a paragraph of text (orange), an image (magenta), an equation (yellow), a list (blue for bullet lists and red for numbered lists), a table (green), or an algorithm (purple).

There are basically two possibilities to include additional attributes in a treemap visualization: changing the size or the color of the rectangles according to a given attribute value. In the above examples and in the following, we compute a weighting function for each node within the hierarchy that then determines the size or color of the rectangle to be drawn for each node. To visualize the structure of the document alone this function should be designed in such a way that larger parts of the document also cover larger rectangles in the treemap so that the distribution of the material within the

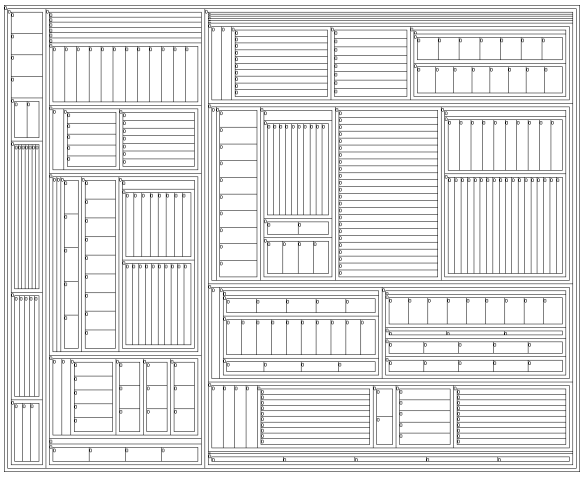


**Figure 1:** *Using treemaps to visualize the structure of the document without showing any additional information.*

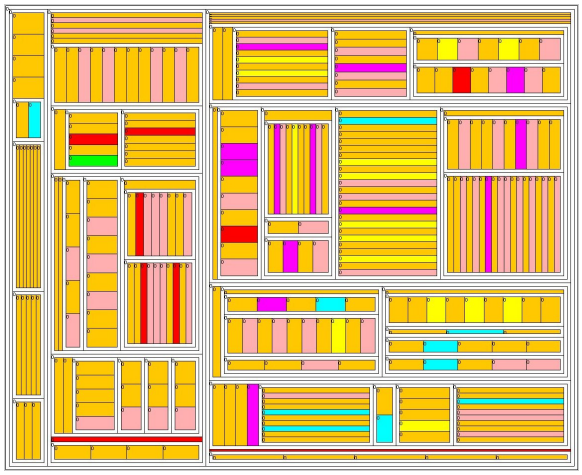


**Figure 2:** *Color coding the type of leaf nodes in addition to a structure visualization.*

document becomes clear. The most simple solution counts the number of leaves in a subtree and uses this measure as weight for the root node of a subtree. We compute such a measure – $\omega_{\text{leaf}}(n)$ – recursively bottom-up starting at the leaf nodes:

$$\omega_{\text{leaf}}(n) = \begin{cases} 1 & n \text{ is leaf node} \\ \sum_{i=1}^{m} \omega_{\text{leaf}}(n_i) & n \text{ is inner node} \end{cases}$$

In the above equation $n_i$ are the child nodes to node $n$. Given $A_n$ as the area available to draw an inner node $n$, i.e., to draw the subtree rooted at $n$, the area covered by the $i$-th child of $n$, denoted as $n_i$, is then computed as

$$A_{n_i} = \frac{\omega_{\text{leaf}}(n_i)}{\sum_{j=1}^{m} \omega_{\text{leaf}}(n_j)} A_n$$

The area of node $n$ is subdivided according to these fractions. Since we are using the standard treemap algorithm, the areas are alternately subdivided horizontally and vertically. Setting $\omega_{\text{leaf}}(n) = 1$ for leaf nodes yields the same size for all sibling leaves. Figures 1 to 3 are generated using this approach. There are other possible weighting functions that support the visualization of the document structure, for example, using the length of the text forming one paragraph or similar measures. Also, Strahler numbers [Str52] could be used as weights since these values give a quantitative information about the complexity of a (sub)tree (see also Herman et al. [HMD98]). In the following, we use $\omega_{\text{leaf}}$ as basis for further derived weighting functions with the goal to present not only the structure but also the results of a full-text query, i.e., to visualize the frequency of occurrence of the query terms.

Performing a query with a single query term results in assigning an integer value – the number of occurrences of the query term – to each leaf node of the tree (the paragraphs) and summing up the values of the child nodes for each inner node. The root node finally contains the sum of the occurrences for the complete document. Therefore, we introduce a new weighting function $\omega_{\text{occ}}(n)$ as follows:

$$\omega_{\text{occ}}(n) = \begin{cases} 1 + t(n) & n \text{ is leaf node} \\ \sum_{i=1}^{m} \omega_{\text{occ}}(n_i) & n \text{ is inner node} \end{cases}$$

with $t(n)$ being the number of occurrences of a query term in a paragraph (leaf) node. This function combines the structure of the tree (counting the number of leaf nodes) and the number of occurrences of the query term. We can map these integers onto the color of the treemap regions as in Figure 3. Here, we have chosen to use the same color (hue) for each node and only change the brightness, so that the color gets darker with an increasing number of occurrence of the query term, i.e., the brightness for each node is a function $b_n = f(\omega_{\text{occ}}(n))$.

It becomes apparent that color coding enhances the recognition of the search result on higher levels of the hierarchy since here the number of occurrence is significantly higher resulting in a significant darkening. Up to now, the *size* of the treemap regions only represent the document structure as it is introduced with the weighting function $\omega_{\text{leaf}}(n)$. We can emphasize the recognition of query term locations by also using $\omega_{\text{occ}}(n)$ to determine the size of the treemap regions. Hence, by combining both size and color coding of query results, we achieve an interesting overview of the term distribution compared to the document structure as shown in Figure 4. The process of changing size and color is visible to the user so that he or she gets another indicator of where search results are found.

Full-text search operations are usually performed with more than one query term. This yields the need to (1) find a way to merge the search results for all terms in one integer value that is then visualized and (2) find a way to inform the user which term distribution exists at a specific location.

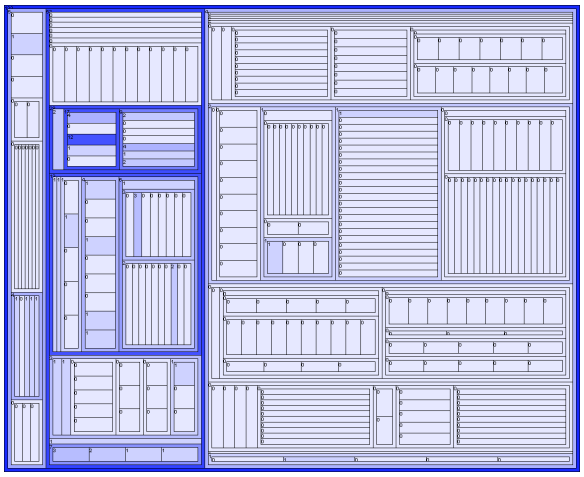


**Figure 3:** *Encoding the number of occurrences of a single query term in the color of the rectangles.*

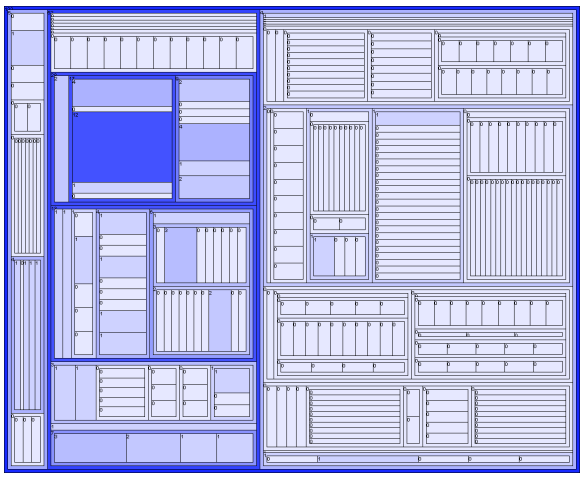


**Figure 4:** *Combining size and color coding in one view results in an overview of the term distribution related to the document's structure.*

The most simple solution to the first part of the problem is to sum up the values for the query terms and present this sum as the overall weight for a paragraph (or higher level node). This is sufficient in most cases since the user is most probably interested in places where any of the query term is found, i.e., the query terms are combined using a logical "OR". Given the document and a set of $k$ query terms that occur $t_1(n), \ldots, t_k(n)$ times in a paragraph, we derive the weighting function $\omega_{\text{comb}}(n)$ as

$$\omega_{\text{comb}}(n) = \begin{cases} 1 + \sum_{j=1}^{k} t_j(n) & n \text{ is leaf node} \\ \sum_{i=1}^{m} \omega_{\text{comb}}(n_i) & n \text{ is inner node} \end{cases}$$

Here, the combination with the document structure (as it was

already done for $\omega_{occ}$) becomes especially valuable. Finding a section or chapter containing the query terms better satisfies the information need at hand than finding just a series of sentences. In Section 6 we will show how to solve the problem of visualizing the term distribution.

## 5. Alternative Presentations – Filtering

As the examples have already shown, the query results are easily recognizable. However, if the document structure becomes larger or more complicated, the presented view might contain too much information. We need filtering techniques to even further emphasize where query terms are found and to deemphasize the less relevant parts. Also here, the user directly sees the process of the changes so that an additional indication is given where the filtering occurs.

In the following we built upon the result that a combination of size and color coding the query results gives the best view of the data. First, we can omit drawing the internal structure for those nodes where no search terms have been found but keeping the size of the rectangles as if the substructure was present (cf. Figure 5). Here, the size of the rectangles is computed as above using $\omega_{comb}(n)$ as weighting function to reflect the number of occurrences of all query terms. However, when drawing the treemap, we stop subdividing a rectangle if there are no occurrences in the subtree rootet at the respective node $n$, i.e., if $\omega_{comb}(n) = \omega_{leaf}(n)$. The problem here is that the size of a rectangle in this case represents the size of the respective subtree while on the other hand the size of a rectangle for a non-empty node represents also the number of occurrences of query terms. This might be confusing and misleading.

To prevent this, such nodes where a substructure is hidden are also treated as leaf nodes when drawing the treemap resulting in a size of the rectangle that is similar to a paragraph node at the same level of the hierarchy. An example is given in Figure 6. For leaf nodes we compute the weight as described in Section 4 while for inner nodes the computation is changed as follows:

$$\omega_{filt}(n) = \begin{cases} 1 + \sum_{j=1}^{k} t_j(n) & n \text{ is leaf node} \\ 1 & \text{if } \sum_{i=1}^{m} \omega_{filt}(n_i) - \omega_{leaf}(n) = 0 \\ \sum_{i=1}^{m} \omega_{filt}(n_i) & \text{otherwise} \end{cases}$$

This yields a size as for an empty leaf node at the respective hierarchy level if no search term was found.

Still, there is another source of misunderstanding. Two very different kinds of nodes are now displayed in one view without visually distinguishing them: *real* leaf nodes that do not contain the query term and roots of subtrees that do not contain query terms and hide a complete substructure. However, nodes not containing query terms are irrelevant for the current search so that it is more important to emphasize these parts of the structure that do actually match the query. We therefore combine cushion treemaps [vWvdW99] with regular treemaps. All those nodes that represent findings of query terms are displayed as cushion treemap nodes while the rest remains flat. In addition, the label of such rectangles that hide substructures with no query terms is changed. The result is shown in Figure 7.

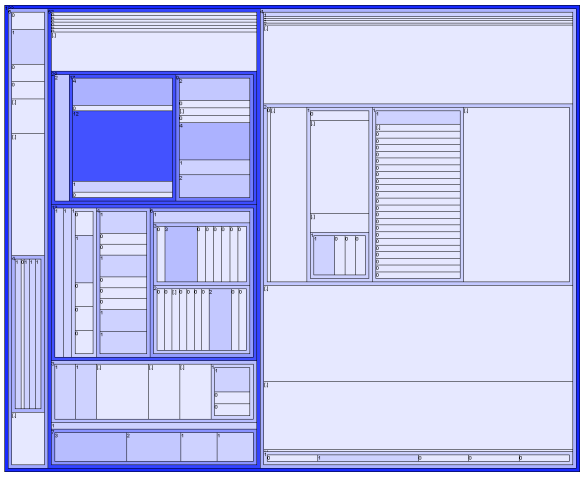


**Figure 5:** *Omitting the structure but keeping the size as that of the whole subtree*

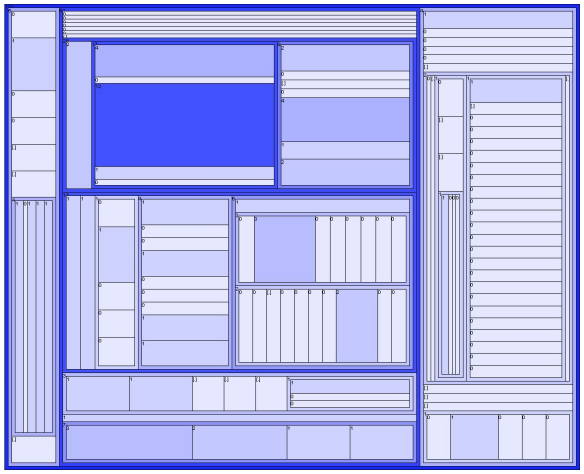


**Figure 6:** *Omitting the structure and adjusting the size*

The resulting visualizations simplify the structure by not showing irrelevant parts (i.e., hiding all subtrees that do not contain any query terms) but still make the structure visible. If at least one query term was found, then the second level of the hierarchy (in our case, the chapters of the document) is always displayed since the root node of the complete document is not empty. Also, the areas of the document that match the query are clearly recognizable due to the alternative presentation. In order to even more raise the effectivity of the visualization, additional information should be given to the viewer on request.

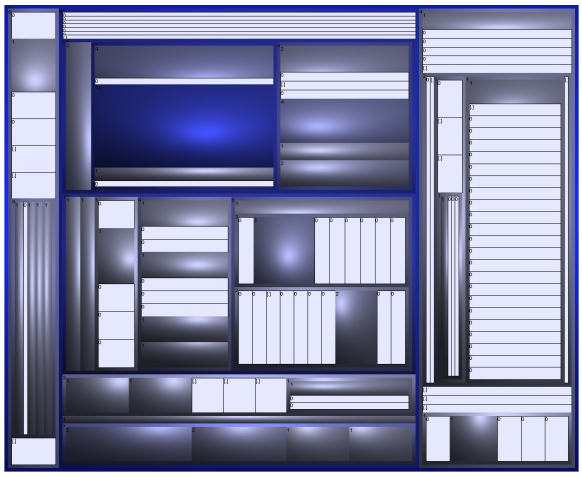
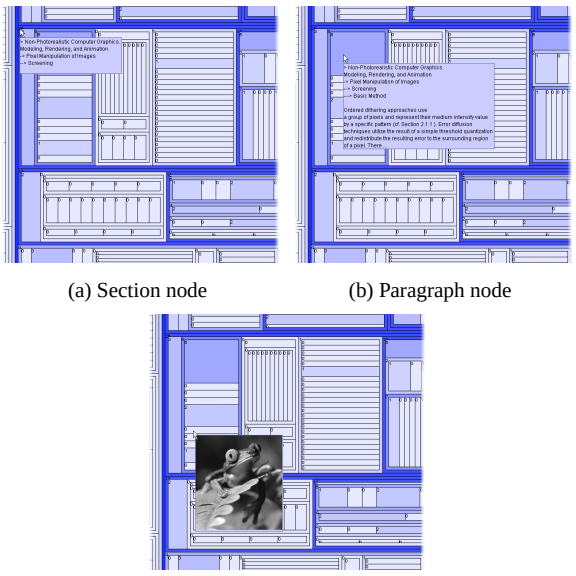


**Figure 7:** *Using cushion treemaps in areas where query terms have been found and leaving all other areas flat.*



(a) Section node (b) Paragraph node

(c) Image node

**Figure 8:** *Integrating context information as tooltips. The images show just a part of the treemap*

## 6. Interactive Tools – Details on Demand

While we have concentrated so far on integrating a visualization of query results and a visualization of the document structure, we still need to provide tools to assess the context of a location. Again, the document structure gives information of how well a larger unit of text (section or chapter) might fit. We provide local and more global information via tooltips as one technique to provide detail on demand. An example that uses such techniques in another application domain is the "Map of the Market" on `www.smartmomey.com` where tooltips are used to present specific data and a selection of further information sources.

In our scenario, moving the mouse in an area of the treemap causes a box to appear containing information that depends on the type of the node under the mouse. Grouping nodes, such as sections and chapters, are well characterized by their title and their relation to higher order nodes. So we show all section titles that are found on the direct path from the root to the respective node (see Figure 8(a)). Paragraph nodes (leaf nodes) usually do not have a title so that here either the first part of the paragraph can be displayed or, preferrably, the part of the paragraph that contains the query terms. Here, also, the position of the paragraph in the document hierarchy is shown by the titles of all higher order nodes (Figure 8(b)). Finally, if other document parts – such as images – are also contained in the hierarchy, they can be displayed and help to estimate the context (Figure 8(c)).

Such tooltips can also be used to visualize the distribution of query terms from a multi term query. When presenting the query results simply by giving the overall sum of matches, there is no way of assessing whether the respective location fits the user's information needs or not. Imagine a set of query terms (a, b, c) being found with the following quantities. Paragraph A contains 20 findings with the distribution (6, 7, 7) while paragraph B also contains 20 findings but this time distributed as (0, 20, 0). Visualizing the number 20 alone (as it is done with the techniques described in Section 4) gives no chance of distinguishing between these two situations. Therefore, the user has no possibility to reject paragraph B and concentrate on paragraph A. We have decided, nevertheless, to use the above mentioned technique to visualize the overall numbers in a treemap and include an additional visualization that shows the term distribution when interacting. We can show a histogram of the term distribution as tooltip when moving the pointing device over the treemap. This gives immediate access to all needed data but prevents, on the other hand, the display of the information about the context of a term location. Giving priority to the latter kind of information, we therefore show the histogram alongside the treemap in a different area of the query interface as can be seen in Figure 9.

Finally, the search interface is tightly coupled with a standard presentation of the underlying text, for example, in a browser or a special reader. Selecting a node in the treemap moves the selection to the specified location in the text. This functionality is comparable to the user interface used for TileBars [Hea95] where mouse interaction within the TileBar visualization leads to a selection of the text in the document. While TileBars work on a sequential text partitioning, our approach also supports navigation based on hierarchy information.

Putting it all together, the process of full-text search in a

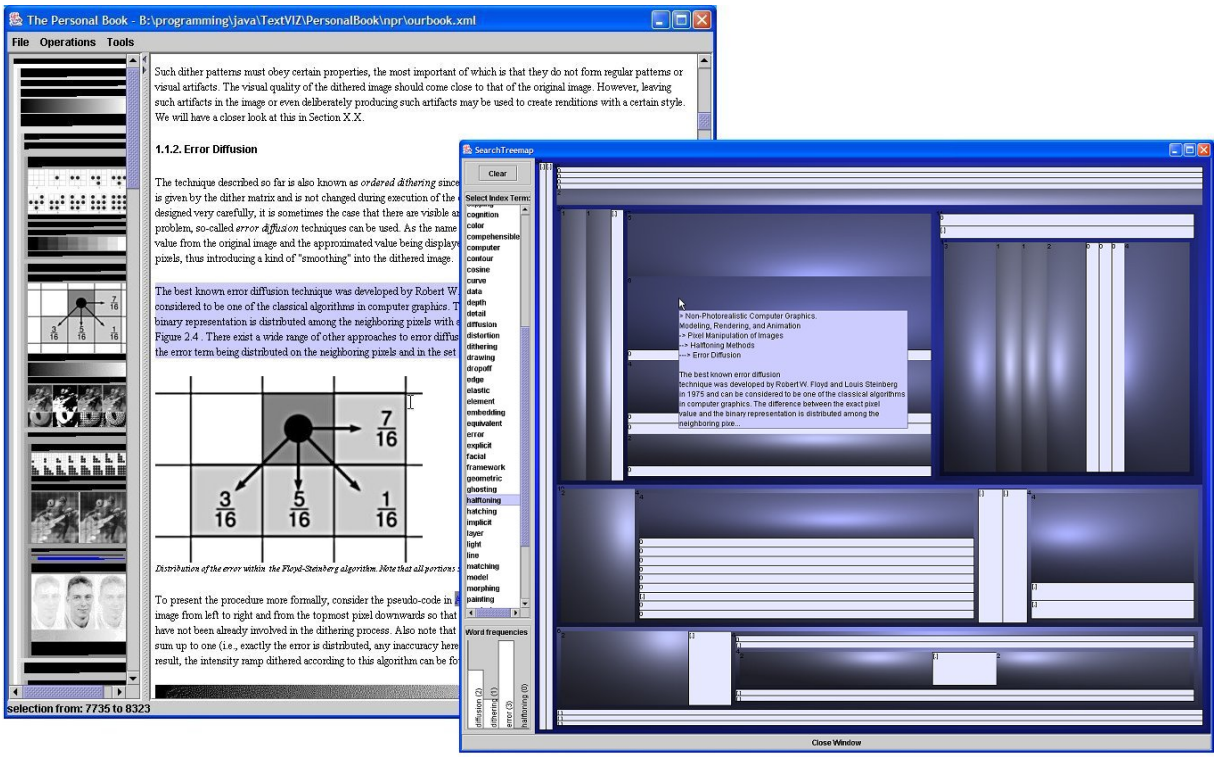



**Figure 9:** *The complete search interface. In the right window, the query terms are selected from the list and the treemap is built. The term distribution in the currently pointed at node is shown in the histogram in the bottom left corner of that window. Finally, the user selected this specific paragraph and the document browser (left window) shows the respective position.*

document now consists of two steps. After issuing the query, a treemap visualization of the document is built that helps to find the part of the document that best fits the user's information needs. From the treemap he or she can then directly jump to the desired location in the text (cf. Figure 9).

## 7. Conclusion

In this paper we have addressed the problem of full-text search in documents. The problem of assessing to which extend a finding matches the intention of the query needs a close connection of the search itself and the placement of the location in the document structure. Given a query built from one or multiple query terms, we build a treemap based visualization of the document structure and integrate quantitative information about the query term distribution by encoding the number of occurrences via color and size of the treemap regions. This yields an overview visualization to help locating interesting parts of the document. To further enhance this visualization filtering techniques hide substructures that do not match the query. We use alternative graphical representations to communicate the result of this filtering to the user. Finally, for an assessment of the local surrounding of a location, interactive tools are provided to give context details on demand. The visualization is closely coupled with a standard display of the document so that it can be used for navigation.

Further questions that arise from this research are mainly connected with the alternative presentations introduced in connection with the filtering techniques. Since the transition between the areas displayed as cushion treemaps and using standard treemaps is a rather abrupt one, we are searching for other techniques to deemphasize irrelevant parts of the structure. Semantic Depth of Field [KMH01] as one possible solution is only adequate if many adjacent regions are to be blurred since otherwise there are only uniquely colored rectangles and the effect can be neglected. One approach would be to use different textures on the treemap regions or to include schematic representations, such as TileBars or Information Murals, of the respective paragraph directly in the treemap.

Another question arises when the documents get larger. Then, the treemap regions become too small to be recognizable. Therefore, adaptive techniques seem to be promising where the detail of the structure visualization is set according to the number of occurrences of query terms. Since it is important to provide local detail in the surrounding of query term locations, simply visualizing the document

structure only up to a certain level is insufficient. Here overview + detail techniques should prove useful. Adding a dimension to the visualization and use 3D layouts like BeamTrees [vHvW02] or Information Cubes [RG93] can be another possibility to handle large documents.

Finally, an even closer coupling between the search visualization and the rendered text would further enhance the usability. One example would be to present the paragraph text using text attributes and images in the tooltips since humans tend to remember such graphical information better than the actual contents of a paragraph. In general, a user study should compare our approach with standard tools known from WWW browsers or eBooks. We hypothesize that the task completion time for finding a specific section in a document based on a roughly stated idea could be lower using our approach than with standard tools. Therefore, such tools as presented here provide even more added value for electronic documents.

**References**


[BHvW00] BRULS M., HUIZING K., VAN WIJK J.: Squarified Treemaps. In *Proc. TCVG 2000, the Joint Eurographics and IEEE TCVG Symposium on Visualization* (2000), IEEE Computer Society, pp. 33–42.

[FP02] FEKETE J.-D., PLAISANT C.: Interactive Information Visualization of a Million Items. In *Proc. InfoVis 2002* (2002), IEEE Computer Science Press, pp. 117–127.

[Hea95] HEARST M. A.: TileBars: Visualization of Term Distribution Information in Full Text Information Access. In *Proc. CHI'95* (1995), ACM SIGCHI, pp. 59–66.

[HHWN02] HAVRE S., HETZLER E., WHITNEY P., NOWELL L.: ThemeRiver: Visualizing Thematic Changes in Large Document Collections. *IEEE Trans. on Visualization and Computer Graphics 8*, 1 (Jan. 2002), 9–20.

[HMD98] HERMAN I., MELANÇON G., DELEST M.: Tree Visualisation and Navigation Clues for Information Visualisation. *Computer Graphics Forum 17*, 2 (1998), 153–165.

[JS98] JERDING D. F., STASKO J. T.: The Information Mural: A Technique for Displaying and Navigating Large Information Spaces. *IEEE Trans. on Visualization and Computer Graphics 4*, 3 (Sept. 1998), 257–271.

[KMH01] KOSARA R., MIKSCH S., HAUSER H.: Semantic Depth of Field. In *Proc. InfoVis 2001* (2001), IEEE Computer Society, pp. 97–104.

[KMS02] KEIM D., MÜLLER W., SCHUMANN H.: Visual Data Mining. In *State of the Art Reports – Eurographics'2002* (2002), Eurographics Association.

[Ren94] RENNISON E.: Galaxy of News. An Approach to Visualising and Understanding Expansive News Landscapes. In *Proc. UIST'94* (1994), ACM Press, pp. 3–12.

[RG93] REKIMOTO J., GREEN M.: The Information Cube: Using Transparency in 3D Information Visualization. In *Proc. Workshop on Information Technologies & Systems (WITS'93)* (1993).

[Shn92] SHNEIDERMAN B.: Tree Visualization With Treemaps: a 2-d Space-Filling Approach. *ACM Transactions on Graphics 11*, 1 (1992), 92–99.

[Shn96] SHNEIDERMAN B.: The Eyes Have it: A Task by Data Type Taxonomy for Information Visualization. In *Proc. VL'96 Symposium on Visual Languages* (1996), IEEE Press, pp. 336–343.

[Spe01] SPENCE R.: *Information Visualization*. ACM Press, New York, 2001.

[SS02] STROTHOTTE T., SCHLECHTWEG S.: *Non-Photorealistic Computer Graphics. Modelling, Animation, and Rendering*. Morgan Kaufmann Publishers, San Francisco, 2002.

[Str52] STRAHLER A.: Hypsomic analysis of erosional topography. *Bulletin Geological Society of America* (1952).

[SW01] SHNEIDERMAN B., WATTENBERG M.: Ordered Treemap Layouts. In *Proc. InfoVis 2001* (2001), IEEE Computer Society, pp. 73–79.

[vHvW02] VAN HAM F., VAN WIJK J. J.: Beamtrees: Compact Visualization of Large Hierarchies. In *Proc. InfoVis 2002* (2002), IEEE Computer Science Press, pp. 93–100.

[vWvdW99] VAN WIJK J. J., VAN DE WETERING H.: Cushion Treemaps: Visualization of Hierarchical Information. In *Proc. InfoVis'99* (1999), IEEE Computer Society, pp. 73–78.

[WTP*95] WISE J., THOMA J., PENNOCK K., LANTRIP D., POTTIER M., SCHUR A., CROW V.: Visualising the Non-Visual: Spatial Analysis and Interaction With Information from Text Documents. In *Proc. InfoVis'95* (1995), IEEE Press, pp. 51–58.

# I

## ARCTREES: VISUALIZING RELATIONS IN HIERARCHICAL DATA

Bibliographische Angaben:

Petra Neumann, Stefan Schlechtweg und Sheelagh Carpendale: „ArcTrees: Visualizing Relations in Hierarchical Data“. In: Ken W. Brodlie, David J. Duke und Ken I. Joy (Hrsg.), *Data Visualization 2005, Eurographics/IEEE VGTC Symposium on Visualization Symposium Proceedings*. Eurographics Association, Aire-la-Ville, 2005. S. 53–61.

Abstract:

In this paper we present ArcTrees, a novel way of visualizing hierarchical and non-hierarchical relations within one interactive visualization. Such a visualization is challenging because it must display hierarchical information in a way that the user can keep his or her mental map of the data set and include relational information without causing misinterpretation. We propose a hierarchical view derived from traditional Treemaps and augment this view with an arc diagram to depict relations. In addition, we present interaction methods that allow the exploration of the data set using Focus+Context techniques for navigation. The development was motivated by a need for understanding relations in structured documents but it is also useful in many other application domains such as project management and calendars.

Betreute Diplomarbeit:

Petra Neumann: *Focus+Context Visualisation of Relations in Hierarchical Data.* Diplomarbeit, Institut für Simulation und Graphik der Otto-von-Guericke-Universität Magdeburg, 2004.

EUROGRAPHICS - IEEE VGTC Symposium on Visualization (2005)
K. W. Brodlie, D. J. Duke, K. I. Joy (Editors)

# ArcTrees: Visualizing Relations in Hierarchical Data

Petra Neumann[1], Stefan Schlechtweg[2], and Sheelagh Carpendale[1]

[1] Department of Computer Science, University of Calgary, 2500 University Drive NW, Calgary, AB, Canada T2N 1N4
[2] Department of Simulation and Graphics, Otto-von-Guericke University of Magdeburg, Universitätsplatz 2, D-39106 Magdeburg, Germany

**Abstract**

*In this paper we present,* ARCTREES*, a novel way of visualizing hierarchical and non-hierarchical relations within one interactive visualization. Such a visualization is challenging because it must display hierarchical information in a way that the user can keep his or her mental map of the data set and include relational information without causing misinterpretation. We propose a hierarchical view derived from traditional Treemaps and augment this view with an arc diagram to depict relations. In addition, we present interaction methods that allow the exploration of the data set using Focus+Context techniques for navigation. The development was motivated by a need for understanding relations in structured documents but it is also useful in many other application domains such as project management and calendars.*

Categories and Subject Descriptors (according to ACM CCS): H.5.2 [Information Interfaces and Presentation]: User Interfaces – Graphical user interfaces (GUI) I.3.6 [Computer Graphics]: Methodology and Techniques – Interaction techniques

**Keywords:** Information visualization, tree structures, relations, arc diagrams, user interface, Focus+Context

## 1. Introduction

Many types of data are either naturally hierarchical or amenable to a recursive grouping that can establish a hierarchy. However, the relations indicated by this natural or imposed hierarchical or tree structure are usually not the only types of relations of interest. While there are an increasing number of methods for drawing trees, almost no methods have been designed specifically to display the additional relations and reveal their connections to the hierarchy without resorting to general graph layout strategies. We present ARCTREES, a visualization method that integrates the display of additional relations with the hierarchical structure.

To illustrate this point, we mention a few of the many possible examples of information with a primarily hierarchical structure that also incorporate additional relations. For example, consider a text book. It has an explicit hierarchical structure composed of the book, its chapters, their sections, and the paragraphs and images that comprise those sections. There are also additional relations of interest. These include indicated forward and backward references, the sequence of readings for a course of instruction, and exactly which previous sections in the text are needed to understand a particular passage (cf. Figure 1). Another example can be taken from examining sports tournament data. During elimination rounds players advance through initial games and play-off rounds to quarter and semi finals with eventually the winner ending at the root of the hierarchy. Here, additional relations might include tracing players of a particular nationality, or discovering which players are playing at which facilities. As a third example, we may look at calendars. With a common display of a calendar it can be difficult to discover such things as start and end dates of recurring events or exceptions within an otherwise regularly occurring event.

ARCTREES can be used for any general graph by displaying a spanning tree as the hierarchical structure and the remaining edges as additional relations. However, it has been primarily designed as an informative interactive tool for viewing digital media, as, for example, digital or on-line documents. ARCTREES were developed to provide insight about relations within data through visual and interactive means. Therefore, the general problem is to combine two very different kinds of relational information in one visualization. These are the hierarchical parent-child relations, and vari-

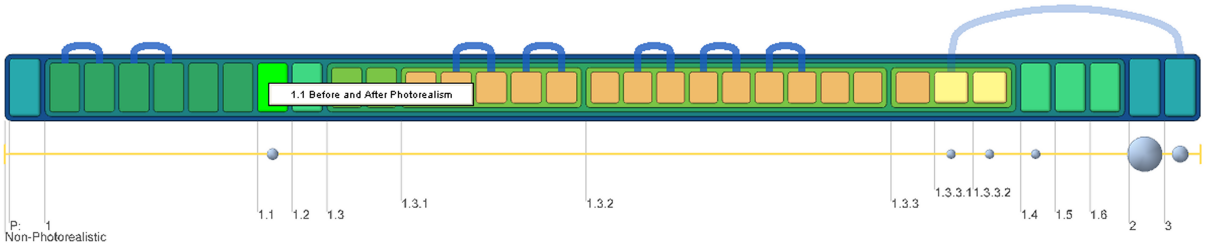



**Figure 1:** *An* ARCTREES *visualization of a scientific book. Arcs indicate references between images and paragraphs in the book. The circular glyphs indicate currently not visible relations.*

ous other kinds of relations that may provide links between nodes on different levels of the hierarchy. The goal of such a visualization is to provide a visual tool that enables a better understanding of the structure of the data and that supports interactive exploration of the data. As such it is important to keep the following constraints in mind:

- use as little screen space as possible because the majority of the space will be needed for the media itself, e.g., the document,
- reveal both the hierarchy and the additional relations,
- make explicit how the additional relations are linked to the hierarchy, and
- include navigational tools and methods that can reveal the navigation actions taken within the visualization and provide methods for reversing these actions.

ARCTREES address three main points: the visualization of the hierarchical structure of the data, the visualization of additional relations, and interaction techniques that allow the exploration of the data.

The remainder of the paper is organized as follows. After a discussion of related research in Section 2 we explain the three main issues of our approach: the visualization of the hierarchical structure in Section 3.1, the visualization of additional relations in Section 3.2, and interaction techniques in Section 4. In Section 5 we illustrate the use of ARCTREES with practical examples and conclude the paper in Section 6.

## 2. Related Work

Visualizing relations between data items is an essential information visualization task. Relations can represent an inherent attribute of the data such as a linear structure of time or another metric, or a hierarchical parent-child relation. Other types of relations might be given through additional parameters or attributes of data items like their size, date, or type. Typical encoding mechanisms for relations are color, shape, orientation, proximity, position, and connection. Position is often used to encode relations in the visualization of linearly structured data sets. Chronological and line-oriented textual data items are usually presented in a linear order implying before and after-relations. If data items are related in some other way they are often encoded with similar color, shape, or texture. A visualization of linear data that breaks away from this pattern is Arc Diagrams, a recent visualization technique for repetition patterns in strings [Wat02]. Direct connections with arcs visualize relations between repeating parts in the string. This visualization has inspired Thread Arcs [Ker03], a technique that shows relations in email conversations that have a direct connection.

The visualization of relations through direct connections with lines or arrows, for example, is a well-known technique. Given a set of data items and a set of relations between these items, a common approach is to use a node-link diagram in the form of a graph. Here the items are represented as nodes and the relations become the links or edges between the nodes. There are many possible graph drawing techniques yielding a variety of different node link diagrams (cf. [HMM00, BETT99]). The parent-child relations of hierarchical data have been depicted with connections in node link diagrams (cf. [HMM00]), with containment in Treemaps [Shn92], and with position and proximity in a variety of configurations including Sunburst [SZ00], Info Slices [AH98], InterRing [JORP01], or Icicle Plots [KL83]. While the latter two kinds of tree visualizations are superior to the traditional node-link diagram in the usage of screen-real-estate, it is typically more difficult to understand their hierarchical relations. However, Barlow and Neville have shown in a study [BN01] that Icicle Plots compared favorably to other visualization techniques with respect to communicating topology and comparison of node size. Furthermore, Icicle Plots of large but not very deep hierarchies result in layouts with a rational use of screen space.

Visualizations specifically tailored to the combination of both hierarchical and non-hierarchical relations in one view are rare. Similarity relations between nodes that are related to other nodes are sometimes encoded through similar color, shape, or texture. Fekete et al. [FWD*03] presented a technique where a Treemap provides the hierarchy visualization and additional relations were overlaid as Bézier arcs connecting two Treemap regions. While this visualization does offer an example that combines a hierarchy with additional relations, it suffers from many edge crossings which makes it

hard to read and uses a considerable amount of screen space. Balzer et al. recognize this problem within their visualization of large software systems [BNDL04]. They state that very unclear representations would be the result of representing multiple relations as direct connections in 2D. Therefore, they suggest building a three-dimensional visualization which can be explored interactively by the user. Since the view parameters can be changed freely, an exploration of the relations is possible by selecting parameters that keep occlusion and overlapping at a minimum.

## 3. The ARCTREES Visualization

The purpose of ARCTREES is to graphically represent hierarchical data together with additional relations in order to trigger discoveries necessary for insight or decision making. Therefore, the hierarchical structure must be clear and the explicit details of the external relations apparent.

### 3.1. Visualizing the Hierarchy

Since ARCTREES are intended to be used as a modular components of an information display such as adjacent to a related text display, minimum and consistent use of screen space is important. We chose to represent the hierarchical structure via containment since this offers effective use of screen space that does not expand during interaction. Also, to honor the fact that the leaf nodes of text information often have significant linear ordering, such as chapters in a book, we combined the ideas of Treemaps [Shn92] and Icicle Plots [KL83] to develop a "one-dimensional Treemap". Each node $n$ is represented by a rectangle on the screen. Leaf nodes are aligned linearly and sequentially. The width of the root node is set to the available display space and the height of the root rectangle is influenced by the depth of the tree. The height is calculated such that the rectangles for child nodes are contained within their parent's rectangle. Some offset is reserved for visual clarity and interaction while ensuring that the deepest leaf node is still visible. Within a level the width of a node is determined by a node weight $w(n)$ while the height depends on the level of $n$.

The hierarchy layout algorithm partitions the rectangular display space for the root into required parts according to the number of root's children. For each child, its horizontal display space is then again divided according to the number of children at the next level. This algorithm establishes an ordered tree layout and provides a spatially constrained display. The visual metaphor involves viewing a stack of nodes, similar to an upside-down Icicle Plot, from the top (Figure 2).

Node weights are influenced by a given node metric $f(n)$. A node metric measures or quantifies certain abstract features associated with a node. Node weights are used to determine the display size of a node in relation to its siblings and ancestor nodes. In Figure 3, several examples are given

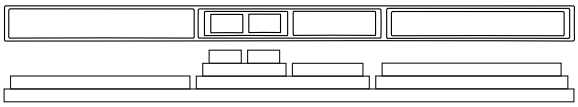


**Figure 2:** ARCTREES *layout (top) and metaphor (bottom).*

including the most simple metric that assigns the same constant value to each node so that on each level a uniform subdivision is achieved. More meaningful structural metrics calculate the size of a rectangle according to the number of direct children or the size of the subtree rooted at a particular node. We can also work with semantic metrics where the size depends on the data contents of the node or some value provided via interaction (degree of interest). Node weights can be interactively adjusted (see Section 4).

Color coding is used to portray structural information and highlight certain aspects of the visualized hierarchy. Figure 4 shows two examples. To allow for a better understanding of the hierarchy itself, a color scale is used to assign equal colors to nodes on an equal level in the hierarchy (top image in Figure 4). The bottom image in Figure 4 shows a semantic coding where the leaf nodes are colored according to their type. The group nodes on higher levels of the hierarchy are alternately colored white and gray so that the level information becomes immediately apparent.

### 3.2. Visualizing Relations

Relations are given as pairs of nodes and each relation may also be defined with a type parameter so that different kinds of relations can be visualized. Inspired by Arc Diagrams (cf. [Wat02, Ker03]), for each relation we draw an arc between the horizontal centers of the two nodes. Each arc is drawn above the tree visualization with its two endpoints extending down to the connected nodes using straight lines. Using semicircles as arcs results in a visualization with a sub-optimal use of screen space as can be seen in Figure 5 (top). For instance, a semi-circle relation spanning the complete length of the hierarchy would result in an arc height of half the length of the rectangle. Therefore, we use Chaikin's Algorithm [Cha74], to draw curves which can be controlled in height as can be seen in Figure 5 (bottom). Drawing half-ellipses would be another alternative but the resulting curves are less flexible in shape.

Additional information can be encoded using the opacity and width of the arcs. All arcs are drawn somewhat transparent to make the crossings clear and understandable (see Figure 6). An even more decreased opacity encodes connections between collapsed nodes (see Section 4). The width of an arc is determined by a metric that specifically calculates a weight value for each relation. Similarly to a node metric the relational metric also depends on structural or semantic information. Furthermore, if more than one relation connects

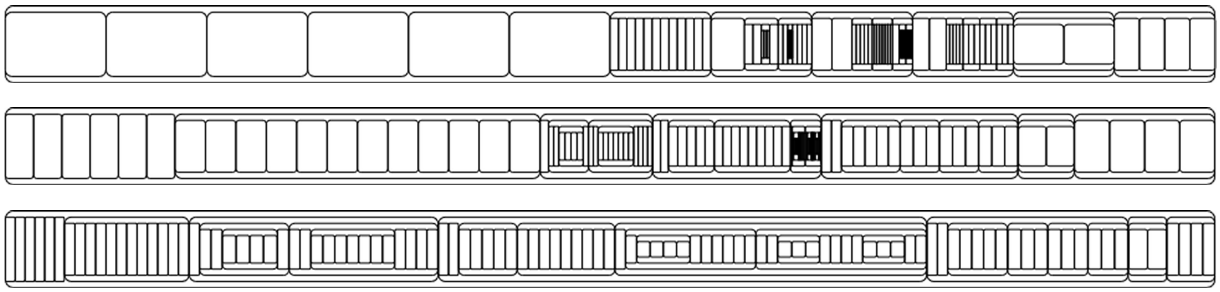


**Figure 3:** *Different tree layouts depending on the node metric. All leaf nodes having the same width (top), node width depending on the number of direct children (middle), node width depending on the size of the subtree (bottom)*

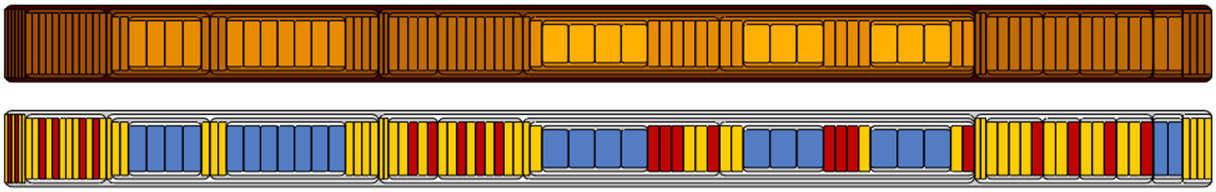


**Figure 4:** *Color coding according to the hierarchy levels (top) and based on semantic information (bottom). Here, an XML file is visualized using different colors for different tag types in the leaf nodes. Inner nodes all represent the same kind of grouping tag so that level information can be given by alternating dark and bright rectangles.*

two nodes, only one arc is drawn and the weights of all relations are added to calculate the arc's width. This resembles the approach taken by Balzer et al. in [BNDL04], where the width of an arc represents the number of relations that are present. Color could also be used to encode additional information. However, since the choice of color needs to be coordinated with the colors that are in use within the hierarchy visualization, currently color is not being used as an encoding technique for additional information about relations.

Figure 6 shows an example that uses this encoding scheme. The transparent arcs on the right hand side connect a collapsed node (far right) with other nodes. The different widths of the more opaque arcs show different weight values that have been computed from the number of relations that are present between the particular nodes.

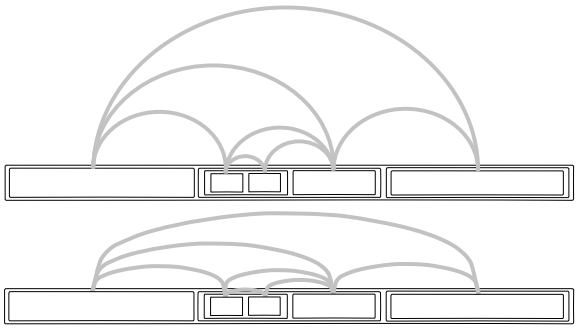


**Figure 5:** *Drawing circular arcs (top) or Chaikin curves (bottom) to visualize relations.*

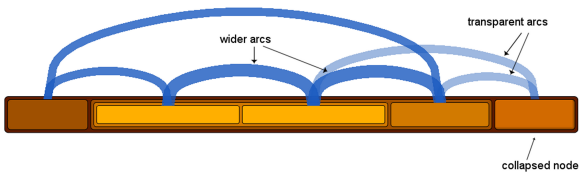



**Figure 6:** *Encoding techniques used for relations.*

## 4. Interaction

To enable the exploration of large hierarchical data structures, interaction techniques are needed to augment the spatial layout. Appropriate interaction tools are criticial because the number of leaf nodes that can be displayed with the proposed tree layout is bounded by the display resolution. As with all layouts, as the number of nodes gets larger the visual clutter increases and individual nodes may get too small to be discernible (Figure 4). To address this problem we provide a variety of interaction techniques including zoom and filter techniques and Focus+Context techniques for both the hierarchy and the relations. Also, a Focus+Context navigation can be combined with zoom and filter operations that depend on user-specified nodes of interest. This allows for larger trees and relations to be explored in detail at subtree

resolution while providing a contextual overview for the remainder of the tree.

### 4.1. Zoom and Filter Interactions

We implemented a simple zoom and filter mechanism by expanding and collapsing subtrees based on user interaction, as can be seen in Figure 7. A node that can be further expanded is represented as a button with shading to indicate that it can be pushed. A fully expanded or leaf node is drawn as flat. A subtree that has been collapsed will be represented as a button indicating that this operation can be reversed.

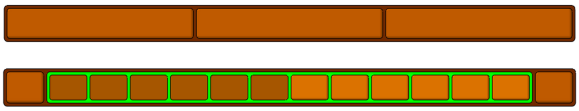


**Figure 7:** *A tree containing three subtrees (top). All three nodes are collapsed. After interaction, the middle subtree is expanded one level (bottom).*

### 4.2. Focus+Context: Interactions with the Hierarchy

Subtrees may also be expanded or collapsed according to their assigned degree of interest. The degree of interest values for each node $n$ are computed based on

1. the distance $d(n,r)$ of the node $n$ from the root node $r$ as an "a priori importance" according to Furnas' terminology (cf. [Fur86])
2. the distance $d(n,f)$ of the node $n$ to the focus node $f$ as the "a posteriori importance"

Both are combined as $\mathrm{DOI}(n) = -(d(n,f)+d(n,r))$. Note that the distance in both the above cases is defined as the length of the path between the two nodes in question. After having selected a node as focus node, thresholding on the DOI determines those nodes that need to be collapsed (i.e., the context) and those nodes that need to be expanded (i.e., the focus area). The DOI values can also be used to obtain a node metric which then determines the size of the rectangles to be drawn (cf. Section 3.1), thus, providing a full Focus+Context technique. Figure 8 shows the same tree as in Figure 3, this time using a DOI based metric.

### 4.3. Focus+Context interactions for relations

Providing a Focus+Context visualization for the relations requires additional care. Three issues are of importance:

1. the arc layout may have to change during interaction,
2. relations between two nodes should contribute to the DOI value, and
3. relations might be used to initiate the interaction.

When interacting with the tree visualization, nodes that are connected via relations may become invisible (when a parent node is collapsed) or nodes may become visible (when a parent node is expanded) that need to be directly connected via relations. To avoid information loss during interaction a visualization is needed that encodes relations connecting to hidden child nodes of a collapsed node. When a node connected by a relation becomes hidden during interaction, its connecting arc is drawn to the collapsed parent. If at least one of the nodes directly connected by a relation is hidden inside a collapsed node, the arc representing this relation is drawn using a lower degree of opacity (see Figure 6). A relayout is also necessary when an *indirect connection* (leading to a collapsed node) becomes a *direct connection* (leading to visible nodes). In this case, arcs need to be connected to the respective visible nodes and the opacity needs to be increased. The case in which a relation is completely hidden inside a collapsed subtree is indicated by a circular glyph below the collapsed node to indicate that hidden relations exist (cf. Figure 9).

To explore the data based on relational information, the relation arcs may be used directly for interaction. Selecting a relation will cause an increase in the DOI value of the connected nodes, i.e., both connected nodes are regarded as focus nodes. Therefore, they are (a) expanded and (b) enlarged in size. To achieve this, external relations are considered when calculating the DOI. An initial DOI is calculated as described in Section 4.2. Then, the following four steps adjust these DOI values to consider relational connections:

1. Collect all relations currently connected to a focus node directly or indirectly.
2. Follow these relations recursively and adjust DOI values if needed.
3. If a new DOI value was set for a node adjust its ancestors if needed.
4. Repeat the previous steps for all nodes in the focus area.

This procedure enlarges the DOI values for all nodes that are connected via external relations directly or indirectly to the focus node. Due to the recursive application, the distance along these relations is automatically taken into account. The recursion stops when reaching a node without further relations or when returning to the focus node in question (circular relational references are possible). The adjustment of the ancestor nodes is needed since no parent node can have a lower DOI than a child node (cf. [Fur86]).

Figure 9 shows an example for layout changes when interacting with relations. The top image shows the start situation. The user interacts with the left arc that connects a visible and a collapsed node. After interaction the relations need to be updated since the collapsed node has expanded due to DOI changes (it becomes part of the focus since it is connected to the selected relation). Note that the selected relation splits up into two relations to different child nodes of the formerly collapsed node (highlighted in green). Other relations appear

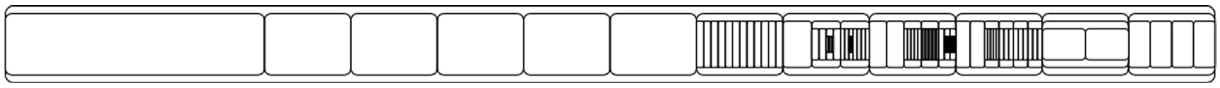


**Figure 8:** *DOI based node metric. The focus is on the first node on the left. Note that no Focus+Context techniques have been applied here, this is just a visualization of the node metric.*

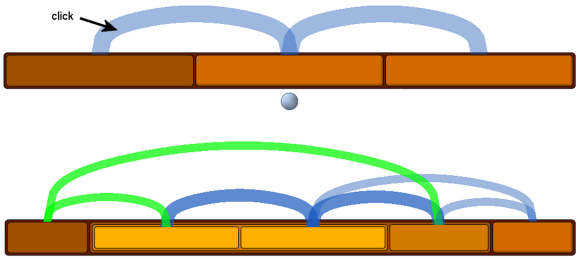



**Figure 9:** *Direct interaction with relations.*

within the middle node as well as from the middle one to the right node. Since the middle node is expanded, the right relation arc also needs to be split to display the now visible relations to the child nodes. All changes finally result in the bottom image.

Hidden relations are encoded as circular glyphs below the tree visualization. The diameter of the circular glyph stands for the number of hidden relations it encodes. Glyphs are drawn with shading to provide push-able affordance indicating interaction possibilities. Interaction with the circular glyph buttons expands the respective tree nodes until at least one relation becomes visible. In addition, our system provides tooltips to display detail on demand for relations and nodes as can be seen in Figure 10.

## 5. Examples

In this section we use an example of a scientific text to illustrate the intended use of ARCTREES. Scientific books are seldomly read from cover to cover, instead, they are consulted to learn about a specific topic. Reading a chapter that deals with this particular topic may require background information from other chapters. Assume an electronic book where such information in the form of relations named "requires knowledge of" or "is prerequisite for" is present. As an example we have chosen a book on Computer Graphics (cf. [FvDFH90]) and modeled the structure and the above mentioned relations. Figure 11(a) shows an overview of the 21 chapters that build this book. (Note that we have not modeled all chapters, therefore, only a few of them are expandable.)

Now, the reader looks up shadow algorithms and the table of contents tells him or her that Section 16.4 deals with this topic. Having no previous knowledge about shadow algorithms he or she might not be familiar with all the background information needed to understand the algorithms covered in this section. Figure 11(b) shows additional information describing "requires knowledge of" relations for Section 16.4. The circular button below Chapter 16 tells the reader that some sections in the same chapter are required reading. The reader will also notice a relatively wide arc between Chapter 15 and 16 which represents more than one connection between these chapters. He or she might then first request additional information on Chapter 15 and find out through the tooltip that Chapter 15 is concerned with visible surface determination which seems to be an important aspect of shadow algorithms. A click on the arc connecting to Chapter 15 reveals four relations between both chapters. In a Focus+Context view eight focus nodes are placed in the visualization. This leads to the eight connected nodes being displayed larger in comparison to their siblings giving a clearer picture of the connected nodes in Figure 11(c) Through detail on demand the reader will notice that the knowledge of four algorithms is needed for implementing shadows: Appel's Algorithm, a z-Buffer Algorithm, Scan-Line Algorithms, and the Weiler-Atherton Algorithm. Each of these algorithms is connected to one different shadow implementation covered in Section 16.4. Based on this information the reader can now decide to concentrate on a shadow algorithm for which he or she already has the required background, to concentrate on a specific one, or to go through all algorithms after learning the required background information.

## 6. Discussion, Directions and Conclusions

In this paper we have presented ARCTREES, an interactive visualization tool that integrates hierarchical and relational information in one view. Since ARCTREES were designed to be used as an accompanying tool with another information display, the space usage has been successfully limited to a narrow strip approximately one tenth the size of a normal desktop display. ARCTREES:

- require only a small amount of screen space,
- integrate additional relations with a hierarchical representation without obscuring the structure of the hierarchy,
- use 3D button affordances to indicate available interactions and hidden information,
- provide Focus+Context interactions that consider the importance of the structure in the hierarchy and the structure in the relations, and

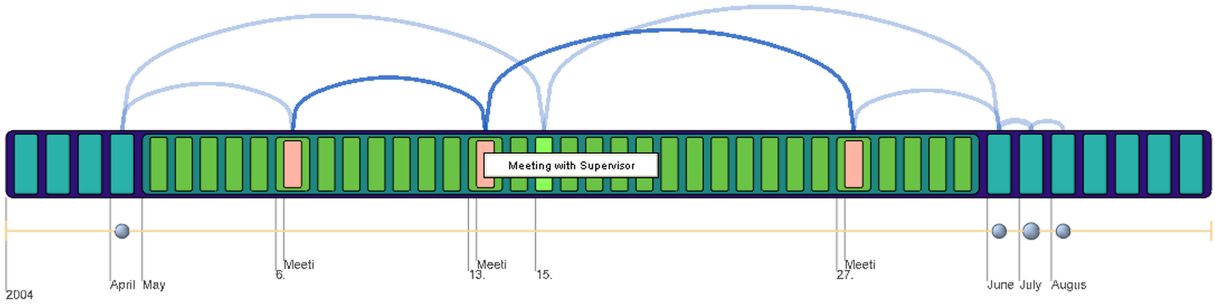



**Figure 10:** *The complete interactive visualization with all additional tools present. This example shows the exploration of a person's calendar. Relations represent recurring events.*

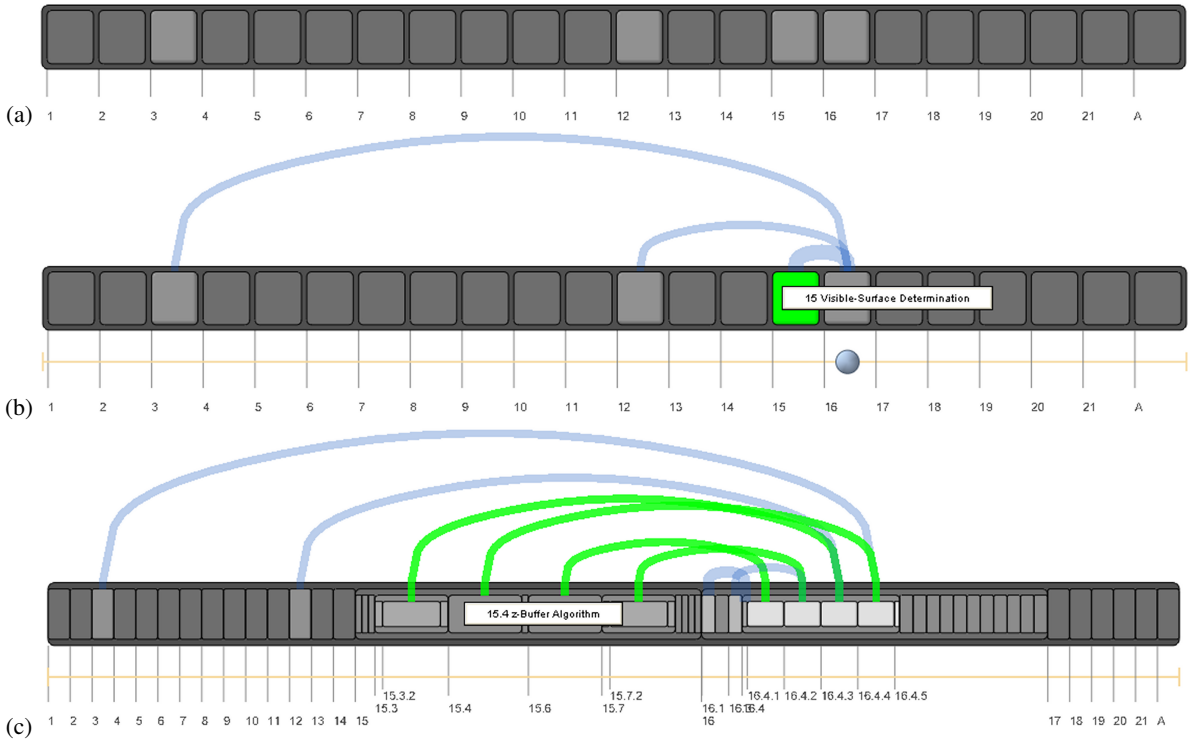



**Figure 11:** *Using* ARCTREES *to explore a scientific book. (a) shows the general structure of the book being divided into 21 chapters and an appendix. (b) shows that the topic that is treated in Chapter 16 builds on information from Chapters 3, 12, and 15. Selecting the wide arc finally results in (c), which allows a detailed exploration of the connections between both chapters.*

- augment these interactions with users requested zoom and filter, labels, tooltips and additional text.

Both classes of relations, i.e., hierarchical and non-hierarchical, are often encountered in information spaces. Electronic documents are only a prominent example. Here, the document structure defines the hierarchical relations while a wealth of other relations may connect parts of the text. For navigation in and exploration of such an information space, the understandable depiction of both types of relations is important. ARCTREES provide an explicit visualization of the hierarchy and how the additional relations are connected to it.

The effectiveness of ARCTREES as a visualization tool depends on several aspects. One is whether the structural relations are understandable. Our goal was to provide a visualization that used much less space than Treemaps and maintained the same degree of readability. Initial comparative pilot studies suggest that the layout of ARCTREES can

be read significantly faster at equal error rate when compared in several tasks to the traditional slice-and-dice Treemap layout while using approximately 1/8 of the space of Treemaps for large trees.

We are also interested in enriching the visualization of relations. For instance, color schemes to indicate relation type could be integrated with the color schemes in use for the hierarchy. Also some relations are directional. Visual indications of direction could be included. One possible method would be to make arcs become gradually darker in the direction of the relation.

Significant layout changes occur when expanding and contracting nodes—especially if many hidden relations exist. While animation of these changes is important, it is possible that other techniques might be developed that better communicate the changes to the user.

ARCTREES might, for example, be used as a modular component of an electronic reading environment. Given the space constraints that were observed in the design of this visualization, it is well imaginable that such a visualization might be part of, for example, Acrobat Reader as the thumbnail overview is today.

### Acknowledgments

We gratefully thank our funding providers: Natural Sciences and Engineering Research Council (NSERC), Alberta's Informatics Circle of Research Excellence (iCORE), and Alberta Ingenuity. We would also like to thank the researchers from the Department of Simulation and Graphics at the University of Magdeburg and the Interactions Lab at the University of Calgary for their insightful comments on this work.

### References

[AH98] ANDREWS K., HEIDEGGER H.: Information Slices: Visualising and Exploring Large Hierarchies using Cascading, Semi-Circular Discs. In *Proc. InfoVis '98, Late Breaking Hot Topics* (1998), IEEE Press, pp. 9–12.

[BETT99] BATTISTA G. D., EADES P., TAMASSIA R., TOLLIS I. G.: *Graph Drawing: Algorithms for the Visualization of Graphs*. Prentice Hall, Upper Saddle River, NJ , USA, 1999.

[BN01] BARLOW T., NEVILLE P.: A Comparison of 2-D Visualizations of Hierarchies. In *Proc. InfoVis'01* (2001), IEEE Press, pp. 131–138.

[BNDL04] BALZER M., NOACK A., DEUSSEN O., LEWERENTZ C.: Software Landscapes: Visualizing the Structure of Large Software Systems. In *Eurographics/IEEE TCVG Visualization Symposium Proceedings* (2004), Eurographics Association, pp. 261–266.

[Cha74] CHAIKIN G. M.: An Algorithm for High Speed Curve Generation. *Computer Graphics and Image Processing 3*, 12 (1974), 346–349.

[Fur86] FURNAS G. W.: Generalized Fisheye Views. In *Proceedings of CHI'86* (New York, 1986), ACM SIGCHI, pp. 16–23.

[FvDFH90] FOLEY J. D., VAN DAM A., FEINER S. K., HUGHES J. F.: *Computer Graphics. Principle and Practice*, 2. ed. Addison Wesley Publishing Company, Reading, MA, 1990.

[FWD*03] FEKETE J.-D., WANG D., DANG N., ARIS A., PLAISANT C.: Overlaying Graph Links on Treemaps. In *Proc. InfoVis'03, Poster Compendium* (Aug. 2003), IEEE Press, pp. 82–83.

[HMM00] HERMAN I., MELANÇON G., MARSHALL M. S.: Graph Visualization and Navigation in Information Visualization: A Survey. *IEEE Transactions on Visualization and Computer Graphics 6*, 1 (2000), 24–43.

[JORP01] J.YANG, O.WARD M., RUNDENSTEINER E. A., PATRO A.: InterRing: A Visual Interface for Navigating and Manipulating Hierarchies. In *Proc. InfoVis'01* (2001), IEEE Press, pp. 16–30.

[Ker03] KERR B. J.: *THREAD ARCS: An Email Thread Visualization*. Tech. Rep. RC22850, IBM Research, Cambridge, MA, USA, 2003.

[KL83] KRUSKAL J. B., LANDWEHR J. M.: Icicle Plots: Better Displays for Hierarchical Clustering. *The American Statistician 37*, 2 (May 1983), 162–168.

[Shn92] SHNEIDERMAN B.: Tree Visualization With Treemaps: a 2-d Space-Filling Approach. *ACM Transactions on Graphics 11*, 1 (1992), 92–99.

[SZ00] STASKO J. T., ZHANG E.: Focus+Context Display and Navigation Techniques for Enhancing Radial, Space-Filling Hierarchy Visualizations. In *Proc. InfoVis'00* (Oct. 2000), IEEE Press, pp. 57–65.

[Wat02] WATTENBERG M.: Arc Diagrams: Visualizing Structure in Strings. In *Proc. InfoVis'02* (2002), IEEE Press, pp. 110–116.

# J

## INTERACTIVELY EXPLORING BIBLIOGRAPHICAL DATA FOR LITERATURE ANALYSIS

Bibliographische Angaben:

Stefan Schlechtweg, Stefan Büder und Marcel Götze: „Interactively Exploring Bibliographical Data for Literature Analysis“. In: Andreas M. Heinecke und Hansjürgen Paul (Hrsg.), *Mensch & Computer 2006: Mensch und Computer im Strukturwandel*. Oldenbourg Verlag, München, 2006. S. 293–302.

Abstract:

This paper introduces techniques for interactive navigation within large sets of bibliographic data. The main conceptual idea is to use various relations between the entries to navigate within the information space. A visualization that supports both the display of the data itself and the relations is introduced. Interaction techniques offer possibilities to follow relations and, thus, create new views onto the data. The proposed visualization and interaction techniques improve literature analysis tasks as occur when writing a scientific document or when reviewing and exploring a scientific topic based on literature.

Betreute Diplomarbeiten:

Stefan Büder: *Visualisierung von Zusammenhängen in Literaturdaten*. Diplomarbeit, Institut für Simulation und Graphik der Otto-von-Guericke-Universität Magdeburg, 2005.

Marc Hofmann: *Interaktion und Visualisierung zur Kommunikation über wissenschaftliche Literatur*. Diplomarbeit, Institut für Simulation und Graphik der Otto-von-Guericke-Universität Magdeburg, 2006.

A. M. Heinecke, H. Paul (Hrsg.): Mensch & Computer 2006: Mensch und Computer im Strukturwandel.
München, Oldenbourg Verlag, 2006

# Interactively Exploring Bibliographical Data for Literature Analysis

Stefan Schlechtweg, Stefan Büder, Marcel Götze

Otto-von-Guericke-Universität Magdeburg, Institut für Simulation und Graphik

**Abstract**

This paper introduces techniques for interactive navigation within large sets of bibliographic data. The main conceptual idea is to use various relations between the entries to navigate within the information space. A visualization that supports both the display of the data itself and the relations is introduced. Interaction techniques offer possibilities to follow relations and, thus, create new views onto the data. The proposed visualization and interaction techniques improve literature analysis tasks as occur when writing a scientific document or when reviewing and exploring a scientific topic based on literature.

## 1 Introduction

Writing a scientific paper, a thesis, or any other work in the scientific community requires a thorough understanding and analysis of other people's written work. Literature review and analysis is therefore an integral part of successful scientific work. Working with literature not only includes to know about the contents of papers and books but also to know relations between certain works, topics and authors who work in an area and how these are interrelated. Especially these relations are a valuable basis to derive new ideas.

Keeping track of the literature in a field is rarely supported by visual tools for exploring the above mentioned relations. In most cases, a literature database containing bibliographic information, possibly personal annotations, and a reference to an electronic version of the article is the only source for a literature survey or analysis. Relations are established and maintained mainly mentally by the person doing the survey, or, for example, taken down in (handdrawn) diagrams.

We propose techniques to support the literature analysis by means of visualization and interaction of and with such relations. Based on an extensive database containing bibliograpic and additional information about the literature of a certain field, several different kinds of data are visualized. This includes information for single papers, a whole body of papers, and rela-

tions between papers and/or between authors. We further propose interactive tools to explore the information space, and possibly to add new information.

# 2 Related Work

There exist a wealth of projects that deal with a certain visualization of bibliographic data. Most interfaces to digital libraries, however, do not use the full power that can be gained when exploiting relational connections among the documents in the library. A standard search engine interface is not enough if it comes to analyzing a field of research or following a scientific topic. The problem here is that the queries to the digital library have to be created manually from the contents of the read paper or the information presented within the digital library. The exploitation of relational information would help to create such queries automatically and, thus, provide an easier way to navigate within the library.

Almost all digital libraries today use a form based interface to their search engine and provide the query results in list form on a web page. Different information about the documents are given ranging from author, title and abstract to a list of citations or a list of documents that are related in some other way. Navigation is almost exclusively based on user formulated queries. A recent example that uses relations for navigation in a digital library is CITEULIKE, (www.citeulike.org). Here, a user can collect documents and relate them to each other via the assignment of "tags". CITEULIKE offers a node-link-diagram to show which documents are related to each other and to navigate through the library. The relations are solely computed based on the assigned tags, no citation or authoring relations are considered.

Hsu et al. (2004) present with MONKELLIPSE an interactive visualization that builds on an elliptical design. All documents are chronologically laid out in an elliptical shape, where additional indicators show a grouping by years. Research topics are shown in the inner area of the ellipse with each area in the relative center of the documents that belong to it. Selecting a document leads to an emphasis of all cited documents as well as of the respective research area. A selection of the research area emphasises all comprising documents. While the pleasing layout, a good use of screen space and the elaborate interaction concept make MONKELLIPSE a nice tool to explore a body of literature, many features that are needed for a deeper literature analysis are missing. Also, the unchangeable layout poses some problems, especially if other criteria than the year of publication are needed to sort the documents.

A different approach was taken by Wong et al. (2004) with their IN-SPIRE system. Here, the most important point is the distribution of documents among various topic areas. A set of documents is visualized as topic map or galaxy view showing clusters of documents belonging together. Additional tools allow, for instance, filtering by time of publication. The outlier tool allows further filtering and a more detailed examination of the document set. Both the topic map and the galaxy view are not connected to each other. This makes the exploration of the information space rather difficult. Also, the visualizations can not be adopted to other information needs, as for instance, authors or relationships between documents and authors.

With a focus on visualization and analysis of large complex networks, WILMASCOPE was presented by Ahmed et al. (2004). To visualize bibliographic data, relations between authors, documents, and other relations are extracted from the data and a network is built. This is then visualized in various ways, allowing to see, for example, the (co-)citations, authors and their relations, or relations within and inbetween research topics. Considering all such relations yields a complex network which can be visualized differently. The used layout techniques make the most prominent nodes in the network stand out. Also, the layered layout offers the advantage of less edge crossings as with normal 2D graph layouts. WILMASCOPE leads to very comprehensive visualizations which are, however, not interactively explorable.

Most applications for handling bibliographical data restrict themselves to browsing the database and showing statistical data. BIBRELEX (Brueggemann, 1999) uses standard graph drawing techniques to reveal and show document relationships, e.g. the citation network. It is, however, restricted to these kinds of relations and the chosen spring mass based graph layout changes if new nodes were added, possibly leading to a complete layout change. The DBL-BROWSER (Klink, 2004) is primarily an interface for browsing in an online bibliographic database. Some visualization tools are added to explore various aspects. The timeline graph gives an overview of time-oriented aspects, e.g., the distribution of published papers of an author over a certain time period. Other more network based visualizations show relations between authors (co-authorship) or between documents (citations). While these visualizations are very useful, a global overview of the complete data set is missing.

Citation and Co-Citation analysis is another area to be considered. Often citation chains and co-citations lead to different documents that are of use for the task at hand. Garfield's HISTCITE (Garfield, 2002) uses, for example, node link diagrams to visualize citation and co-citation graphs. While such visualizations help to get an overview, they might become rather complex and contain too much information for an actual literature analysis task. This can be seen in the work of Chen on the CITESPACE system (Chen, 2004; Chen, 2006). Based on co-citation networks, Chen builds a visualization that supports the identification of intellectually significant articles based on a visual inspection of the graph. Even though the goal behind Chen's work is somewhat different from ours, it shows that a visualization of relational data helps to get new information from a set of bibliographic data.

# 3 Requirements

For an overview of the complete data set, a comprehensive visualization of the *complete* data should be provided. Almost all of the mentioned systems offer such a feature. Searching as well as filtering needs to be provided to support navigation and to reduce the amount of data with which a user is working. WILMASCOPE does not offer searching while even more of the previously mentioned systems, namely WILMASCOPE, MONKELLIPSE, and BIBRELEX, do not contain filtering. As far as the navigation is concerned, a smooth transition between the views is required in order to keep the context. Detail-on-demand techniques can be used to get specific information about one selected entity, narrow a selection, or working as initiator

for a new filtering. An example is to get information about the authors of a publication and then use the co-authorship relation to get to a different set of documents. To enable this filtering, relational information have to be extracted from the initial data and tools have to be provided for navigating with these relations. Astonishingly enough, only WILMASCOPE makes extensive use of such relations. User defined relations are a further information source and even more support navigation. Such relations need to be defined for or computed from the given data. Also, user defined annotations and possibly changes of the data (addenda, corrections) will help in supporting literature analysis in an interactive and visual way.

# 4 The Data

The original data which were used to form the information space consist of an extended BiBTeX database of approximately 600 documents from the area of computer graphics written by about 800 authors. We have added a field to each entry that holds the keys of all documents which are cited by the respective document, a field for topic keywords and a field containing an abstract of the document. Each document in the database is characterized by a set of attributes, which are derived from the given BiBTeX fields. In order to provide the necessary data for a rich visualization and interaction, we derive three sets of data entities, a set of documents: **D**, a set of authors: **A**, and a set of research topics: **T**.

There exist various relations between these three sets which can be calculated from the data base. The following are the most important of them:

- $\mathbf{D} \rightarrow \mathbf{A}^{\mathbf{n}}$: all authors of a document
- $\mathbf{A} \rightarrow \mathbf{D}^{\mathbf{n}}$: all documents written by an author
- $\mathbf{A} \rightarrow \mathbf{A}^{\mathbf{n}}$: all co-authors of an author
- $\mathbf{D} \rightarrow \mathbf{D}^{\mathbf{n}}$: all documents being cited by a document or all documents citing a document
- $\mathbf{D} \rightarrow \mathbf{T}^{\mathbf{n}}$: all topic areas of a document
- $\mathbf{T} \rightarrow \mathbf{D}^{\mathbf{n}}$: all documents in a topic area

If we generalize this with respect to the visualization task at hand, the following general visualization tasks need to be realized:

- visualizing all members of a dataset (overview)
- visualizing the attributes of a single document (attribute relations)
- visualizing relations *within one* dataset (internal relations)
- visualizing relations *between two different* datasets (external relation)

# 5 Visualization and Interactive Literature Analysis

To visualize the literature data and relations, we borrow the elliptic design from TEXTARC (Paley, 2002) and MONKELLIPSE (Hsu, 2004). TEXTARC is an alternative way of displaying a continuous text which is arranged line by line in an elliptical form. In the inner area of the ellipse relevant words are placed according to their position in the text. Relations between words and the whole text are displayed via lines drawn when moving the pointer over a certain word. We build on this basic idea and generalize it for certain types of relations and data sets. The main power of our approach lies in the combination of those visualizations via interaction. Thus, the presented visualization together with interactive techniques allows a target-oriented navigation within the information space. In the following we will present tools for navigation using actual screenshots from the implemented system.

The elliptic design offers two primary advantages. First, due to its aspect ratio, screen space is better used in comparison to a circular layout. Second, an elliptical visualization can be devided into an inner (the ellipse's surface) and an outer part (the ellipse's line and the surface outside of the ellipse), so two different data sets can be easily combined visually.

## 5.1 Visualization of Bibliographical Data

The most general visualization, tool is the *browse ellipse* which displays all entities of a certain set. Figure 1 shows an implementation of this concept. All entries are arranged around the circumference of the ellipse where a single entry is displayed as label.


*Figure 1: Concept of the browse ellipse, showing all documents of one author*

Such a visualization gives a first overview of the information space and offers various navigation tools to gain a deeper insight. The ellipse itself in subdivided into segments, one for each label, which are colored alternatively to make the items more distinguishable. If a very large number of items is present, the labels are not readable. Therefore, Focus+Context techniques according to the *Information Seeking Mantra* (Shneiderman, 1996) should be provided. In a sense, this layout with labels made unreadable to fit the whole information space in the visualization, resembles the Information Murals as presented by Jerding and Stasko (Jerding & Stasko, 1998). Moving the pointer over the labels is used to enlarge the focused label, a direct selection of one item triggers the attribute relation and reveals detailed information in the inner area of the ellipse. Selecting more than one item gives the possibility to group these or open a new browse ellipse with just these items. Offering a search functionality yields a selection of entries that match the search and which are then displayed.

The browse ellipse is also the basis for the visualization of relations. Figure 1 has already shown the display of attribute relations: selecting exactly one entry activates the attribute relation and the entry's attributes are displayed in the inner surface of the ellipse. If an internal relation like $\mathbf{A} \rightarrow \mathbf{A^n}$ or $\mathbf{D} \rightarrow \mathbf{D^n}$ is to be displayed, the originator of the relation is selected and all items being related will automatically become selected. Between both, lines or arrows, depending on the kind of relation, are drawn as Figure 2 shows. To enhance the visibility of the lines, especially when they reach to nearby items, they start from markers which are drawn inward from the position of the respective labels. All selected labels are enlarged and made readable.

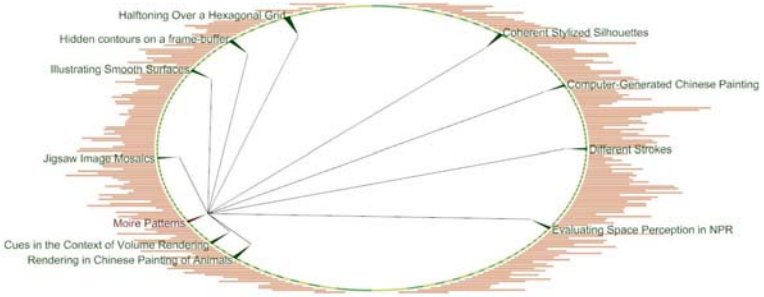



*Figure 2: Showing internal relations as arrows between the selected origin and all related entries*

External relations, i.e., relations between data items from two different sets use the inner area of the ellipse to display the second data set while the first is laid out as shown for the browse ellipse. This layout is especially useful for relations between authors and documents, i.e., $\mathbf{D} \rightarrow \mathbf{A^n}$ or $\mathbf{A} \rightarrow \mathbf{D^n}$. A single data item from the second set is placed in the inner area while lines emanate from there to the related items from the first data set on the circumference of the ellipse. The position of the inner data item is either central or can be computed using a force based algorithm that positions the inner item with respect to the positions of the related items in their relative center.

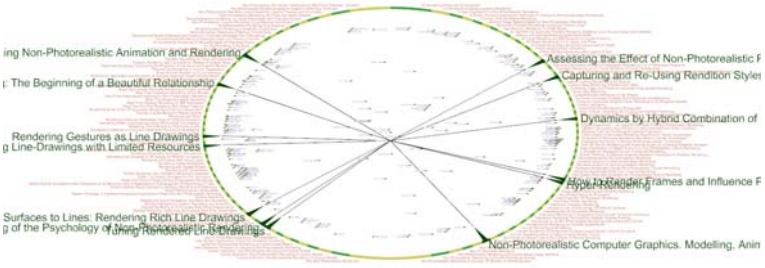



*Figure 3: A single item is related to some data items from another data set.*

Such a force based model is also needed if external relations between several data items from both sets are to be displayed. The items from the first set are, again, distributed on the ellipse's circumference. All items from the second set are laid out following the force based approach in the inner area. In this way, items having similar relations to items from the first set come close together. A direct link via lines and arrows is established when moving the pointer over the respective item (cf. Figure 3). This visualization seems especially valuable

for relations like $\mathbf{D} \rightarrow \mathbf{A}^{\mathbf{n}}$ or $\mathbf{A} \rightarrow \mathbf{D}^{\mathbf{n}}$. However, both data sets can also be identical, so that this concept can also be used to show internal relations like $\mathbf{A} \rightarrow \mathbf{A}^{\mathbf{n}}$ or $\mathbf{D} \rightarrow \mathbf{D}^{\mathbf{n}}$.

To visualize groupings, the browse ellipse is segmented according to the number of groups and these groups are visually emphasized by drawing wedges in the inner area of the ellipse. Each wedge can then hold certain information if space permits. The most simple kind of information being displayed is a label as in Figure 4. Moreover, all group members can be displayed as points within the area of the wedge belonging to the group so that a display of relations becomes possible. When the pointer is moved over a data item, all relations to other items from other groups are shown as direct lines between the respective points.

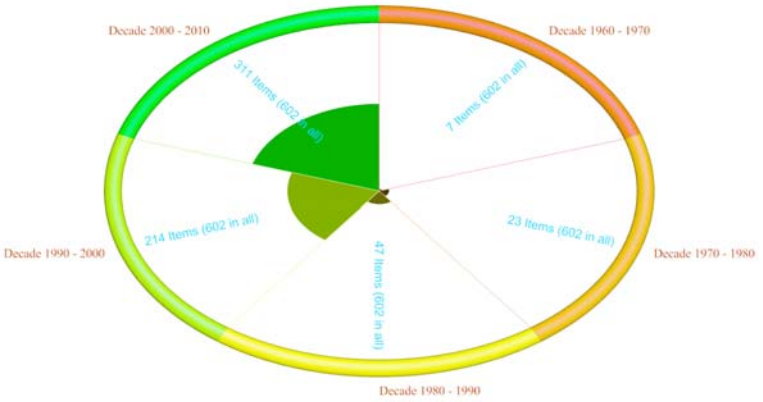



*Figure 4: Grouping of entries with general information about the groups shown as labels. Here, overview of the data grouped by decade of publication*

Some of the relations which are of particularly interest in literature analysis lead to relation chains where they reach over more than just one level. The most prominent example here is the citation relation which yields citation networks. A standard way of visualizing such citation networks as graphs soon becomes problematic since the resulting graphs are rather complex. For a literature analysis in connection with a writing task, citation networks are often needed level by level, or, alternatively, single paths need to be followed, so that an interactive exploration of such networks is more appropriate. Starting with a browse ellipse, all citation relations (for example) can be treated like external relations, so that the ellipse itself with the selected item is moved into the center of a second ellipse which visualizes the same data set. All relations are drawn as direct links. For the next level of the network, a third ellipse is created in the same manner. Moving those ellipses in 3D on separate levels even enhances the recognition of the links due to the possibility to explore the network from various different angles (see Figure 5).

The visualization designs so far concentrate mainly on one particular relation. When working with bibliographical data, there is often the need to follow different relations in order to gain deeper insight into the topic or to establish connections between different documents that are not obvious on a first glance. Therefore, interaction techniques are needed to explore the information space in various ways.

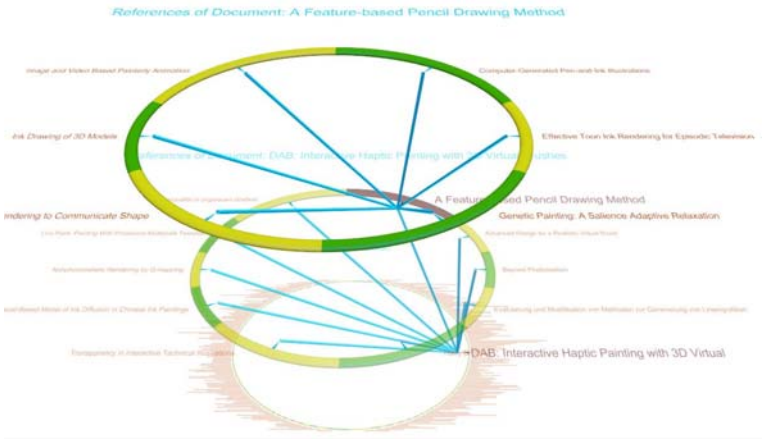



*Figure 5: Three levels of the citation network shown in three ellipses laid out in 3D*

## 5.2 Interactive Literature Analysis

The usual starting point for a literature analysis is a search in the database for either a specific author or a keyword that reveals a set of papers which contain that keyword in the title or even the document text. It would also be possible to start with an overview visualization of the complete set of documents (e.g., as shown in Figure 1). This is a good idea if someone who starts an analysis within a relatively unknown area and wishes to actually search for a paper or author. In our example, we shall start with a different kind of overview visualization that gives the temporal distribution and grouping of the papers related to computer graphics (our example data set). Figure 4 shows this overview where all papers are grouped by decade of publication. This overview is augmented by a histogram like indicator in the ellipse's area to quickly get a comparison of the number of documents in each decade.

We then select the biggest chunk of data, the decade starting 2000 and call a browse ellipse for all documents within this decade. Now the user can browse through these data by moving the mouse over the titles. Note that due to the number of elements the titles are reduced in size so that they are not readable at the moment. However, the title currently under the mouse will be enlarged and highlighted. Moreover, for each selected document, detail information is shown in the middle of the ellipse, as can be seen in Figure 1.

After the user has found a paper “show references” was selected yielding Figure 2 which shows all cited documents as internal relation of the kind $\mathbf{D} \rightarrow \mathbf{D}^{\mathbf{n}}$. From this relation mainly alternative or previous research approaches can be derived. Taking this further and selecting one of the cited documents and again calling up all references leads to the second level of the citation network. Going another step yields Figure 5 where three levels of the citation network are shown. Note that only the subset of the documents that is actually cited is shown on higher levels to avoid too much overlapping and visual clutter.

After working with the citation network, we return to the document overview in Figure 1 and go a different way which is also often gone when analyzing scientific literature. Assuming

that the author of a specific paper is working in a specific topic area, it is a good idea to find other papers by the same author. This will yield an overview of the author's research. There are two possible ways to get this information. First, a new browse ellipse can be opened which shows just the documents by the respective author (see Figure 1). Second, the external relation $\mathbf{A} \rightarrow \mathbf{D}^{\mathbf{n}}$ can be used to relate author and documents visually to each other as can be seen in Figure 3. In contrast to the first option, here the displayed information space is not limited and can be explored further. However, the visualization is also more complex. The inner visualization is made up of the authors which are placed in the relative center of their documents. Selecting one author highlights the documents he or she has written.

Co-authors of an author are likely to work on the same or similar research topics, so that the co-authorship relation can be used to get a more focussed overview of a research area. Figure 2 is an example of a visualization of the co-author relation $\mathbf{A} \rightarrow \mathbf{A}^{\mathbf{n}}$, which is an internal relation on the set of authors.

All visualizations as shown in this section are interlinked with each other and reachable via simple interactions, i.e., selecting an item and determining the new visualization via popup menu selection. The transition between one visualization to the other is smooth and an undo functionality helps in going back to earlier stages in the exploration process.

# 6 Conclusion and Future Work

The proposed visualization and interaction techniques offer a way to browse bibliographic data while being focussed on relational connections between documents, authors and their attributes. In comparison to standard form based interfaces to digital libraries, such a relation based exploration eases the way to find new documents or to mentally connect information. Form based interfaces to digital libraries are perfectly suited for specific queries if the user exactly knows what he or she is looking for. In comparison, queries that may accure from reading a document are rather unspecific, as for example “all documents form an author and his co-authors”, or do not (yet) belong to standard queries for digital libraries.

We argue that for such queries an exploratory approach using relational information and directly visualizing relations is better suited as a standard form based input. Furthermore, the time to find matching results and the mental load is supposed to be smaller. The user is no longer required to build the query outside of the interface and, possibly to mentally integrate several earlier query results. First informal tests have supported this hypothesis, however, a detailed user study will be necessary to evaluate the amount of improvement. Such a study will, nonetheless, require a reimplementation of the techniques so that it could be integrated in existing digital libraries.

The core argument of this paper is that an explicit visualization of relational information within bibliographical data and the use of such visualizations in navigating bibliographical information helps in performing literature analysis tasks. So far we have only considered those relations that have a strong bibliographical background. Adding user-centered relations

as it is sometimes done in digital libraries ("users who have read A also read B") or even user defined relations will open new possibilities and directions. One application area that is interesting to investigate is the use of our techniques in thematic communities related to scientific literature. One such community, CITEULIKE has already been mentioned in Section 2. The discussion of literature will also be enhanced by offering navigation aids through the steadily increasing amount of publications.

Aside from bibliographical data, the presented techniques can also be applied to other areas where relational information between data items play an important role. It might be worthwhile to investigate the use of our techniques in the context of online communities and social networks where members are connected by a wealth of different relations.

## References

AHMED A., DWYER T., MURRAY C., SONG L., WU Y. X. (2004): Wilmascope graph visualisation. In Proceedings of INFOVIS'04, IEEE Computer Society, p. 216.4.

BRÜGGEMANN-KLEIN A., KLEIN R., LANDGRAF B. (1999): Bibrelex: Exploring bibliographic databases by visualization of annotated contents-based relations. D-Lib Magazine 5(11):(1999).

CHEN C. (2004): Searching for intellectual turning points: Progressive knowledge domain visualization. In Proceedings of the National Academy of Sciences of the United States of America (Washington, 2004), vol. 101, National Academy of Sciences, pp. 5303–5310.

CHEN C. (2006): CiteSpace II: Detecting and Visualizing Emerging Trends and Transient Patterns in Scientific Literature. Journal of the American Society for Information Science and Technology.

GARFIELD E., PUDOVKIN A. I., ISTOMIN V. S. (2002): Algorithmic Citation-Linked Historiography – Mapping the Literature of Science. In Proceedings of the 65th Annual Meeting of the American Society for Information Science & Technology, vol. 39, pp. 14–24.

HSU T.-W., INMAN L., MCCOLGIN D., STAMPER K. (2004): MonkEllipse: Visualizing the History of Information Visualization. In Proceedings of INFOVIS'04 (Washington, 2004), IEEE Computer Society, p. 216.9.

JERDING D. F., STASKO J. T. (1998): The Information Mural: A Technique for Displaying and Navigating Large Information Spaces. IEEE Transactions on Visualization and Computer Graphics 4(3):(1998), pp. 257–271.

KLINK S., LEY M., RABBIDGE E., REUTHER P., WALTER B., WEBER A. (2004): Browsing and visualizing digital bibliographic data. In: Proceedings of the 2004 Eurographics/IEEE TVCG Workshop on Visualization., Eurographics Association, pp. 237–242.

PALEY W. B. (2002): TextArc: Revealing Word Associations, Distributions and Frequency. Interactive Poster at the IEEE INFOVIS'02,.

SHNEIDERMAN B. (1996): The Eyes Have it: A Task by Data Type Taxonomy for Information Visualization. In Proceedings of VL'96 Symposium on Visual Languages, IEEE Press, pp. 336–343.

WONG P. C., HETZLER B., POSSE C., WHITING M., HAVRE S., CRAMER N., SHAH A., SINGHAL M., TURNER A., THOMAS J. (2004): In-spire infovis 2004 contest entry. In Proceedings of INFOVIS'04, IEEE Computer Society, p. 216.2.