

---

# Visual Computing

---

Zusammenfassung  
Fabian Damken

---



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT

---

---

## Inhaltsverzeichnis

---

<b>1</b>	<b>Einführung</b>	<b>6</b>
1.1	Visual Computing . . . . .	6
1.1.1	3D-Internet . . . . .	6
1.1.2	Skalierbare Objektmodellierung/-erkennung . . . . .	6
1.1.3	Big Data, Visual Analytics . . . . .	6
1.1.4	Scene Understanding . . . . .	6
1.2	Generalisierte Dokumente . . . . .	6
1.2.1	Retro-Digitalisierung, Digital Creation . . . . .	6
1.2.2	Generative Modeling Language . . . . .	6
<b>2</b>	<b>Wahrnehmung</b>	<b>7</b>
2.1	Human-Computer-Interaction . . . . .	7
2.2	Überblick . . . . .	7
2.2.1	Menschliche Informationsverarbeitung . . . . .	7
2.3	Wahrnehmung . . . . .	9
2.3.1	Das Auge . . . . .	9
2.3.2	Vorverarbeitung visueller Informationen . . . . .	10
2.3.3	Informationsextraktion . . . . .	12
2.4	Aufmerksamkeit . . . . .	14
2.4.1	Limitierung der Wahrnehmung . . . . .	14
2.4.2	Das Gedächtnis und „Gateway to Memory“ . . . . .	14
<b>3</b>	<b>Computer Vision: Objekterkennung und Bayes</b>	<b>16</b>
3.1	Computer Vision . . . . .	16
3.2	Bayesian Decision Theory . . . . .	16
3.2.1	Konzepte und Bayes Theorem . . . . .	16
3.2.2	Problemstellung . . . . .	17
3.2.3	Entscheidungsregel . . . . .	17
3.2.4	Naive Bayes Classifier . . . . .	19
3.3	Probability Density Estimation . . . . .	19
3.4	Gesichtsdetektion . . . . .	19
3.4.1	Sliding Window Ansatz . . . . .	19
3.4.2	Beispiel: Gesichtsdetektion . . . . .	19
3.4.3	Erkennungsarten . . . . .	20
<b>4</b>	<b>Fouriertheorie</b>	<b>21</b>
4.1	Mathematische Grundlagen . . . . .	21
4.1.1	Vektorraum . . . . .	21
4.1.2	Basis eines Vektorraums . . . . .	21
4.1.3	Krummlinige Koordinatensysteme . . . . .	22
4.1.4	Andere Räume . . . . .	22
4.1.5	Komplexe Zahlen . . . . .	22
4.1.6	Gerade/Ungerade Funktionen . . . . .	22
4.2	Fourier-Reihe . . . . .	23
4.2.1	Dirichlet-Bedingungen . . . . .	23

4.2.2	$2\pi$ -periodische Funktion . . . . .	23
4.2.3	Skalarprodukt, Orthogonale Basis . . . . .	23
4.2.4	Berechnung der Koeffizienten $a_m, b_m$ . . . . .	24
4.2.5	Beispiel: Rechteck-Schwingung . . . . .	24
4.3	Fourier-Transformation . . . . .	25
4.3.1	Beispiel: Rechteckimpuls . . . . .	26
4.3.2	Transformationspaare . . . . .	26
4.3.3	2D-Fourier-Transformation . . . . .	27
4.4	Faltung . . . . .	27
4.4.1	Anwendung: Filter . . . . .	27
4.5	Abtastung . . . . .	27
4.5.1	Abtasttheorie . . . . .	28
4.5.2	Abtasttheorem von Whittaker-Shannon . . . . .	28
<b>5</b>	<b>Bilder</b> . . . . .	<b>29</b>
5.1	Bildverbesserung . . . . .	29
5.1.1	Histogramm . . . . .	29
5.1.2	Pixeloperationen . . . . .	29
5.1.3	Kontrastspreizung . . . . .	30
5.1.4	Histogrammausgleich . . . . .	30
5.1.5	Mittelung . . . . .	30
5.2	Bildfilterung . . . . .	30
5.2.1	Ortsraum . . . . .	31
5.2.2	Frequenzraum . . . . .	33
5.2.3	Vergleich: Orts- und Frequenzraum-Filter . . . . .	34
5.3	Bildkompression . . . . .	34
5.3.1	Harmonische Transformation . . . . .	35
<b>6</b>	<b>Bildverarbeitung, Deblurring</b> . . . . .	<b>37</b>
6.1	Korrekt gestellte Probleme . . . . .	37
6.2	Einschrittverfahren . . . . .	37
6.2.1	Wiener Filter . . . . .	37
6.2.2	Mehrkomponentenverfahren . . . . .	38
6.3	Mehrschrittverfahren (Iterative Methoden) . . . . .	39
6.3.1	Energie und Variationsableitung . . . . .	39
6.3.2	Alternativen . . . . .	39
6.3.3	Perona-Malik . . . . .	40
6.3.4	Eingeschränkte Evolution: Totale Variation . . . . .	40
<b>7</b>	<b>Grafikpipeline</b> . . . . .	<b>42</b>
7.1	Hardware . . . . .	42
7.1.1	P1: Large-Scale-Computing . . . . .	42
7.1.2	P2: Personal/Desktop Computing . . . . .	42
7.1.3	P3: Networked Computing . . . . .	42
7.1.4	P4: Mobile Computing . . . . .	42
7.1.5	ZP1: Collaborative Computing . . . . .	42
7.1.6	ZP2: Virtual Reality . . . . .	42
7.1.7	Augmented Reality . . . . .	42
7.1.8	Ambient/Invisible . . . . .	44
7.1.9	Wearable/Ubiquitous . . . . .	44

7.2	Computergrafik . . . . .	44
7.2.1	Geschichte . . . . .	44
7.2.2	Uncanny Valley . . . . .	44
7.3	Grafikpipeline . . . . .	45
7.4	Anwendung . . . . .	45
7.4.1	Eingabe grafischer Daten . . . . .	45
7.4.2	Repräsentation von 3D-Daten . . . . .	45
7.4.3	Räumliche Datenstrukturen . . . . .	46
7.5	Geometrieverarbeitung . . . . .	46
7.5.1	Modelltransformation (Orientierung, Position) . . . . .	46
7.5.2	Simulation der Beleuchtung . . . . .	46
7.5.3	Perspektivische Transformation und Clipping (Abschneiden) . . . . .	47
7.5.4	Culling (Verdeckungsrechnung im Objektraum) . . . . .	47
7.5.5	Projektion . . . . .	48
7.6	Rasterisierung . . . . .	48
7.6.1	Scan-Konvertierung . . . . .	48
7.6.2	Verdeckungsrechnung . . . . .	48
<b>8</b>	<b>Transformationen</b>	<b>50</b>
8.1	Affine Abbildungen . . . . .	50
8.1.1	Eigenschaften . . . . .	50
8.1.2	Homogene Koordinaten . . . . .	51
8.2	Skalierung, Scherung, Rotation . . . . .	51
8.2.1	Skalierung . . . . .	51
8.2.2	Scherung . . . . .	52
8.2.3	Rotation . . . . .	52
8.2.4	Nicht-Kommutativität von Transformationen . . . . .	53
8.2.5	Rechenaufwand . . . . .	53
8.3	Projektion . . . . .	53
8.3.1	Perspektive Projektion . . . . .	54
8.3.2	Parallele Projektion . . . . .	54
8.3.3	Kanonisches Sichtvolumen . . . . .	55
8.4	3D-Interaktion . . . . .	55
8.4.1	Manipulatoren . . . . .	55
<b>9</b>	<b>3D-Visualisierung</b>	<b>56</b>
9.1	(Gewinnung) 3D-Daten . . . . .	56
9.2	Triangulation von Punktwolken . . . . .	56
9.2.1	Ideal Triangulation . . . . .	56
9.2.2	Voronoi-Diagramm . . . . .	57
9.2.3	Delaunay-Triangulation . . . . .	57
9.3	Indirekte Volumenvisualisierung . . . . .	57
9.3.1	3D-Volumen und Nachbarschaft . . . . .	57
9.3.2	2D: Konturlinien . . . . .	57
9.3.3	3D: Isoflächen . . . . .	57
9.3.4	2D: Marching Squares . . . . .	58
9.3.5	3D: Marching Cubes . . . . .	58
9.3.6	Große Polygonmodelle und Performanz . . . . .	58
9.4	Direkte Volumenvisualisierung . . . . .	59
9.4.1	Density Emitter Model . . . . .	59

9.4.2	Volumen-Rendering-Pipeline . . . . .	59
<b>10</b>	<b>Szenengraphen am Beispiel X3DOM</b>	<b>62</b>
10.1	Szenengraph . . . . .	62
10.2	X3DOM . . . . .	63
<b>11</b>	<b>Informationsvisualisierung</b>	<b>64</b>
11.1	Informationsdesign . . . . .	64
11.1.1	Referenzmodell von Card . . . . .	64
11.2	Datentypen . . . . .	64
11.2.1	1D-Daten, Zeitreihen . . . . .	64
11.2.2	2D-Daten . . . . .	64
11.2.3	mD-Daten (multidimensional) . . . . .	64
11.2.4	Hierarchien . . . . .	64
11.2.5	Graphen/Netzwerke . . . . .	64
11.3	Kuchendiagramm (1D) . . . . .	64
11.4	Balkendiagramm (1D) . . . . .	64
11.5	Liniendiagramm (Zeitreihe) . . . . .	64
11.6	Scatterplot (2D, 3D) . . . . .	64
11.7	Scatterplotmatrix (nD) . . . . .	64
11.8	Parallele Koordinaten (3D, nD) . . . . .	64
11.9	Node-Link-Diagramm (Hierarchien, Graphen) . . . . .	65
11.10	Treemap (Hierarchien) . . . . .	65
11.11	Zusammenfassung . . . . .	65
<b>12</b>	<b>Farbe</b>	<b>66</b>
12.1	Dimensionalität . . . . .	66
12.2	Wahrnehmungskorrelate . . . . .	66
12.3	Reproduktion . . . . .	66
12.4	Berechnung von Farbattributen . . . . .	66
12.4.1	Das Auge . . . . .	66
12.4.2	Spektrale Charakterisierung des Auges . . . . .	66
12.4.3	Spektralwertfunktion . . . . .	66
12.4.4	Cone Fundamentals . . . . .	66
12.5	Objektfarben, Lichtmatrix und XIEXYZ-Farbraum . . . . .	66
12.6	Metamerie . . . . .	66
12.7	Gegenfarbtheorie . . . . .	66
12.8	Stevenssche Potenzfunktion . . . . .	66
12.9	CIELAB Farbraum . . . . .	66
12.10	Technische Farbräume . . . . .	66
12.10.1	Geräte RGB . . . . .	66
12.10.2	Geräteunabhängige RGB . . . . .	66
12.10.3	YCbCr . . . . .	66
12.10.4	HSI/HSV/HSL . . . . .	67
12.10.5	CMY/CMYK . . . . .	67
12.11	Komplexität von Farbe . . . . .	67
12.11.1	Chromatische Adaptation . . . . .	67
12.11.2	Farbwahrnehmungsphänomene . . . . .	67
12.11.3	Farbwahrnehmungsmodelle . . . . .	67
12.11.4	Kontrastsensitivität . . . . .	67

---

<b>13 User Interfaces</b>	<b>68</b>
13.1 Interaktion . . . . .	68
13.1.1 Möglichkeiten . . . . .	68
13.1.2 Designprozess . . . . .	68
13.2 GUI: Benutzeroberflächen . . . . .	68
13.2.1 Das WIMP-Interface . . . . .	69
13.2.2 Menübasierte Programme . . . . .	69
13.2.3 GUI-Anwendungen und Event-basiertes Programmieren . . . . .	69
13.3 3D-Interaktion . . . . .	69
<b>14 Multimedia Information Retrieval</b>	<b>70</b>
14.1 Inhaltsbasierte Suche . . . . .	70
14.1.1 Mathematische Beschreibung . . . . .	70
14.1.2 Retrieval Ergebnis . . . . .	70
14.2 Distanzmaße . . . . .	70
14.3 Query-Modalitäten . . . . .	70
14.3.1 Text . . . . .	70
14.3.2 Example . . . . .	70
14.3.3 Example-Bilder . . . . .	70
14.3.4 Sketch . . . . .	70
14.4 Explorative Suche . . . . .	70
14.4.1 Research Data . . . . .	70

---

---

## **1 Einführung**

---

### **1.1 Visual Computing**

---

1.1.1 3D-Internet

---

1.1.2 Skalierbare Objektmodellierung/-erkennung

---

1.1.3 Big Data, Visual Analytics

---

1.1.4 Scene Understanding

---

### **1.2 Generalisierte Dokumente**

---

1.2.1 Retro-Digitalisierung, Digital Creation

---

1.2.2 Generative Modeling Language

---

---

## 2 Wahrnehmung

---

### 2.1 Human-Computer-Interaction

---

Abbildung 2.1 zeigt den klassischen Zyklus der *Human-Computer-Interaction* (HCI), d. h. der Interaktion zwischen Mensch und Maschine. Dabei dient insbesondere die visuelle Interaktion und Kommunikation über das Auge eine große Rolle.

### 2.2 Überblick

---

Der Mensch hat fünf grundlegende Sinne: Sehen, Hören, Fühlen, Schmecken und Riechen, wobei das Sehen, Hören und Fühlen derzeit dominant sind. Der heute sicherlich relevanteste Sinn ist dabei das Sehen und das menschliche Auge. Da die meisten erzeugten Bilder der Kommunikation von und zum Menschen dienen sollen, ist es gut, das menschliche visuelle System zu kennen, um den Informationstransfer optimal zu gestalten (der Monitorausgang ist nicht das Ende des Informationsflusses).

Hören und Fühlen sind dabei relevant für die Informationsaufnahme und Interaktion mit der realen Welt (außerhalb der Mensch-Maschine-Interaktion).

Bei der Gestaltung von Kommunikation gibt es zwei große Probleme:

- Die Wahrnehmung ist nicht objektiv.
- Das visuelle System ist stark nichtlinear (es ist keine einfache Interpolation oder Extrapolation von Versuchsergebnissen möglich).

---

#### 2.2.1 Menschliche Informationsverarbeitung

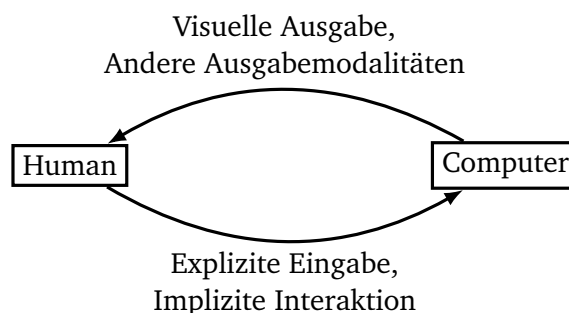
---

Abbildung 2.2 zeigt die drei Stufen der menschlichen Informationsverarbeitung:

- *Wahrnehmung* von Eindrücken durch die Sinne,
- *Entscheidungsfindung* im Gehirn und
- *Reaktion* durch den Körper.

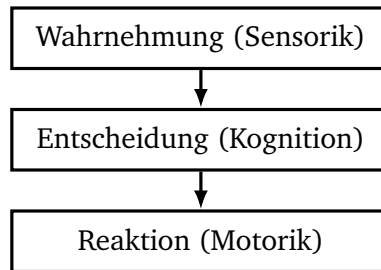
Dabei verhält sich die Ausführungszeit additiv und die Funktionen werden durch neurologisch getrennte Gehirnteile ausgeführt, die „elektronisch“ verbunden sind.

Dabei braucht jede Bearbeitung in den einzelnen Stufen unterschiedlich lange und die benötigten Zeiten können verwendet werden, um die Performanz abzuschätzen, bzw. vorherzusagen (bspw. für die Bildfrequenz von Filmen, die maximale Morserate, ...). Typische Zeiten sind in Tabelle 2.1 abgebildet.



**Abbildung 2.1:** Klassischer Zyklus der Human-Computer-Interaction (HCI).





**Abbildung 2.2:** Modulares Drei-Stufenmodell der menschlichen Informationsverarbeitung.

Untersystem	Durchschnitt	Bereich
Wahrnehmung (Perzeption)	100 ms	50 ms to 200 ms
Entscheidung (Kognition)	70 ms	25 ms to 170 ms
Reaktion (Motorik)	70 ms	30 ms to 100 ms

**Tabelle 2.1:** Typische Bearbeitungszeiten der Untersysteme der menschlichen Informationsverarbeitung.

---

## Eingabe (Wahrnehmung)

---

Die Untersysteme der Wahrnehmung,

- Visuell (Sehen)
- Akustisch (Hören)
- Haptisch (Fühlen)

können dabei (theoretisch) parallel arbeiten.

### Klangwahrnehmung

Die Hauptkomponenten von Klängen sind

- Klangfarbe,
- Tonlage und
- Lautstärke.

Diese werden durch verschiedene Mechanismen wahrgenommen und Informationen (z. B. der Ursprung eines Geräuschs) extrahiert.

### Berührungswahrnehmung

Die Hauptkomponenten der Haptik sind

- Fühl- und Tastsinn (Temperatur, Schmerz, Druck, Oberflächen) und
- Propriozeption (Wahrnehmung der Bewegung und Lage der eigenen Körperteile).

Dabei interagiert die Haptik stark mit Sehen und Hören, was bei sich widersprechenden Informationen Illusionen hervorrufen kann. Ein User-Interface-Designer nutzt Illusionen dabei gezielt aus, um bestimmte Informationen zu vermitteln.

---

## Ausgabe (Reaktion)

---

Die Untersysteme der Reaktion,

- Artikulation (Sprechen)
- Motorisch (Bewegen)

können dabei (theoretisch) parallel arbeiten.

Die motorische Ausgabe kann dabei auf verschiedene Weisen angewandt werden:

- Diskret (Schalter) oder
- Kontinuierlich (Heben).

Sie ist dabei beschränkt durch Geschwindigkeit, Stärke, Koordinationsvermögen, Wendigkeit, ... Neurologisch ist die motorische Ausgabe dabei mit dem haptischen System verbunden (Reflexe).

Das *Muskelgedächtnis* hilft dabei, relevante Positionen im Raum (z. B. die Gangschaltung im Auto) zu lernen.

---

## 2.3 Wahrnehmung

---

---

### 2.3.1 Das Auge

---

---

#### Reiz und Licht

---

Einer äußerer, visueller Reiz (Licht) erzeugt beim Menschen eine physikalische Rezeption des äußeren Reizes (Input). Dies geschieht durch einen Sensor (bspw. das Auge) und die Reizung produziert ein neuro-physiologisches Signal. Dieses wird anschließend verarbeitet und interpretiert.

Physikalisch ist ein solcher Reiz elektromagnetische Strahlung. Dabei wird monochromatisches, d. h. einfarbiges, Licht durch die Angabe der Frequenz  $\nu$ , bzw. der Wellenlänge  $\lambda$ , beschrieben. Diese beiden Größen sind durch die Beziehung

$$\nu\lambda = c, \quad c \approx 3 \times 10^8 \text{ m s}^{-1}$$

miteinander verknüpft, wobei  $c$  die Ausbreitungsgeschwindigkeit des Lichts ist.

Das menschliche Auge kann dabei Frequenzen im Wellenlängenbereich 380 nm to 750 nm wahrnehmen. Kleinere Wellenlängen haben z. B. Ultraviolett-Licht, Röntgen- und  $\gamma$ -Strahlung. Darüber liegende Wellenlängen haben z. B. Infrarot-Licht und Rundfunk-Wellen.

---

#### Das visuelle System

---

Das menschliche Auge ist aufgebaut aus:

- Hornhaut (Kornea)
- Linse (zur Scharfstellung)
- Iris (Blendenmechanismus)
- Retina (Netzhaut)
  - Blinder Fleck: Hier geht der Sehnerv ab.
  - Fovea Centralis (Gelber Fleck): Bereich mit der höchsten Auflösung.

---

## Photorezeptoren

---

Die Photorezeptoren (welche auf der Retina platziert sind), bestehen aus:

- Stäbchen
  - Hauptsächlich außerhalb der Fovea.
  - Das Empfindlichkeitsmaximum liegt bei 498 nm („grün“).
- Zapfen
  - Vor allem in der Fovea platziert.
  - Es gibt drei Zapfentypen für Farbsehen.
  - Das Empfindlichkeitsmaximum dieser Zapfen liegt bei 420 nm („blau“), 534 nm („grün“) und 564 nm („rot“).

---

## Skotopisches und Photopisches Sehen

---

- Nachtsehen (skotopisch): Dominanz der Stäbchen.
- Tagsehen (photopisch): Dominanz der Zapfen.

---

## Zapfenverteilung

---

---

### 2.3.2 Vorverarbeitung visueller Informationen

---

---

## Signalverarbeitung in der Retina

---

Neben den Photorezeptoren gibt es noch weitere Zellen zur Signalverarbeitung in der Retina:

- Horizontale Zellen  
Kombination von mehreren Rezeptoren einer Region.
- Amakrin-Zellen  
Zeitliche Verarbeitung.
- Bipolar-Zellen  
Informationsfilter (Sammeln, Gewichten und Weiterleiten).
- Ganglien-Zellen  
Integration Informationen (z. B. Kontrastwahrnehmung).

---

## Helligkeit

---

- *Helligkeit (brightness)* entspricht der wahrgenommenen Menge an Licht, das von einer selbstleuchtenden Lichtquelle ausgeht.
- *Helligkeit (lightness)* entspricht der wahrgenommenen Menge an Licht, das von einer reflektierenden Oberfläche ausgeht.
  - Dies ist keine absolute Wahrnehmungsgröße und abhängig von

- \* Reizstärke (Leuchtdichte)
  - \* Vorherige Leuchtdichte (Adaption)
  - \* Umgebungsleuchtdichte
  - \* Größe (Fläche) des Reizes
- Somit subjektiv!

- Dies wirft einige nicht so einfach zu beantwortende Fragen auf, z. B.: Was ist weiß? Was ist schwarz? Was ist mittelgrau?
- Der Hell-Dunkel-Kontrast ist dabei eine wichtige Empfindungsgröße zum Form- und Objektsehen. Daher muss der Unterschied groß genug sein (für kleine Details mindestens 3 : 1, besser 10 : 1).

### Kontrast als Reizverhältnis

Für den Kontrast gibt es verschiedene Definitionen, z. B. (dabei ist  $L$  stets die Leuchtdichte):

$$m = k = \frac{L_{\max} - L_{\min}}{L_{\max} + L_{\min}}$$

oder

$$K = \frac{L_R - L_H}{L_H} = \frac{\delta L}{L_H}$$

wobei  $L_R$  die Leuchtdichte des Vordergrunds und  $L_H$  die Leuchtdichte des Hintergrunds darstellt.

---

### Erkennung von Details

---

Die Erkennung kleiner Details ist begrenzt durch

- Optische Eigenschaften des Auges, z. B. Beugungserscheinungen,
- Abtastung durch Rezeptoren und
- nervöse Verarbeitung.

Zwei mögliche Maße zur „Erkennbarkeit“ sind:

- Kontrastempfindlichkeit
- Schwellenkontrast

### Kontrastempfindlichkeit

Die Kontrastempfindlichkeit ist die Auflösung des menschlichen Auges im Frequenzraum. Veränderliche Intensität kann dabei mit Sinus-förmigen Mustern gemessen werden.

---

### Frühe Wahrnehmung

---

Das Auge nimmt einige Veränderungen der Umgebung schneller wahr als andere. Um die Aufmerksamkeit auf etwas zu lenken, können beispielsweise

- Farbe,
- Richtung,

- 
- Bewegung,
  - Größe,
  - Beleuchtung/Schattierung

variiert werden.

---

### 2.3.3 Informationsextraktion

---

Ein reiner Reiz ist noch keine *Wahrnehmung*. Dazu kommen noch andere Faktoren wie Kontext, Erwartungen, Adaption. Das Messen der tatsächlichen Wahrnehmung ist leider sehr schwierig, weshalb häufig nur statistische Aussagen auf Basis von User-Tests getätigt werden können.

Dabei wird erschwert, dass die Wahrnehmung nicht immer der Realität entspricht. Es wird hingegen das Bild durch einen Wahrnehmungsprozess im Gehirn produziert. Dabei wird die menschliche Wahrnehmung adaptiert, bspw. dreht sich das Bild bei einem Kopfstand.

---

### Raumwahrnehmung

---

Die Wahrnehmung des Raums (Raumwahrnehmung) enthält unter anderem

- Tiefenwahrnehmung,
- Entfernungs- und Distanzwahrnehmung und
- Ausrichtung des Körpers im Raum.

Daran sind viele Wahrnehmungssysteme beteiligt:

- Vestibuläres System (im Innenohr)
- Haptisch-somatisches System (Tasten und Berühren)
- Auditives Sehen (Gehört)
- Propriozeptives System (Eigenwahrnehmung)
- Visuelles System

Dabei ist die Raumwahrnehmung auch mit einem Auge (Monokular) möglich (tatsächlich sind 5 % to 10 % aller Menschen stereoblind und 20 % haben eine Stereo-Schwäche).

Tatsächlich ist die Raumwahrnehmung ein sehr komplexer Prozess, der auch heute nur zu Teilen verstanden wird. Dabei fließen noch viele weitere Phänomene ein, z. B. Größenkonstanz, Annahme starrer Körper oder Vektion. Letzteres ist dabei die scheinbare Eigenbewegung bei einem statischen Vordergrund als Referenzrahmen und einem bewegtem Hintergrund.

---

### Depth Cue Theorie

---

Die Annahme der *Depth Cue Theorie* ist, dass die Raumwahrnehmung des visuellen Systems auf Hinweisreizen (sogenannten *Depth Cues*) basiert. Diese werden in drei Kategorien eingeteilt:

1. Binokulare Depth Cues (mit zwei Augen)
  - Disparität/Parallaxe
  - Akkomodation (Krümmung der Augenlinsen)

- 
- Konvergenz (die Augen nach innen drehen)

## 2. Pictorial Depth Cues (mit einem Auge)

- Linearperspektive
- Verdeckung
- Texturgradient
- Fokus und Blur
- Atmosphärische Tiefe
- Vertraute Größe
- Höhe im Gesichtsfeld
- Beleuchtung
- Schattenwurf
- Luminanzänderung
- Transluzenz
- Schattierung

## 3. Dynamische Depth Cues (Animation)

- Bewegungsparallaxe
- Kinetischer Tiefeneffekt
- Interposition
- Bewegung von Highlights

## Stereoskopie

Bei der Stereoskopie nehmen beide Augen ein leicht unterschiedliches Bild wahr, woraus die Entfernung zu einem Objekt berechnet werden kann.

## Pictorial Depth Cues

### Linearperspektive:

**Texturgradient:** Sind als parallel angenommene Linien nicht mehr parallel, so ergibt sich eine scheinbare Tiefe (als wenn kariertes Papier um einen Ball gerollt und von oben betrachtet wird).

**Fokus und Blur:** Das Auge fokussiert einen Punkt und produziert somit eine Tiefenschärfe. Daran kann erahnt werden, welche Objekte im Vorder- oder Hintergrund sind.

**Atmosphärische Tiefe:** Anhand der Atmosphäre (z. B. durch Nebel ausgelöst) wird erkannt, was vermutlich im Hintergrund liegt. So kann zum Beispiel bei einem Foto von einem Berg geschätzt werden, dass der Boden niedriger ist, wenn Wolken über diesem hängen.

**Schattenwurf:** Annahme: Beleuchtung von oben und Vorhandensein einer Grundebene. Dann kann durch den Abstand von Schatten zum Objekt erahnt werden, wie weit dieses vom Boden entfernt ist.

## Dynamische Depth Cues

### Motion Parallax:

**Raumwahrnehmung durch Bewegung:** Wird z. B. eine schaukelnde Vase von oben betrachtet, so bewegt sich die Öffnung charakteristisch, sodass eine Wahrnehmung der Tiefe entsteht.

### Kinetic Depth Effect, Structure from Motion:

---

## Auswertung von Depth Cues

Unterschiedliche Depth Cues haben im Allgemeinen einen unterschiedlichen Informationsgehalt. Dabei sind sie nicht redundant, sondern additiv. Durch ein kompliziertes Zusammenspiel (flexible Gewichtung, Dominanz eines Depth Cue) bildet sich das Gehirn ein Bild. Dabei bildet es sich allerdings kein tatsächliches 3D-Modell, sondern verwendet sie unterschiedlichen Cues für verschiedene Aufgaben. Diese können z. B. sein:

- Einschätzen von Objektgrößen
- Einschätzen von Entfernungen
- Verfolgung von Pfaden
- Navigation
- Einschätzen der Eigenbewegung
- Abschätzung der Kollisionszeit

---

## 2.4 Aufmerksamkeit

---

---

### 2.4.1 Limitierung der Wahrnehmung

---

Die initiale Reizaufnahme hat viele Limitierungen, sodass nur ein Bruchteil des äußeren Reizes zur kognitiven Verarbeitung zur Verfügung steht. Dabei sind Aufmerksamkeit und externe Faktoren wichtige Einflüsse auf die tatsächliche Wahrnehmung. Die Wahrnehmung ist dabei eher eine partielle Hypothese, die auf Basis unvollständiger Informationen generiert wurde. Es wird dabei periodisch aktualisiert aufgrund von Beobachtungen, d. h. die Hypothese wird gegen sensorische Daten getestet. Durch eine dynamische Suche des visuellen Systems wird nach der besten Hypothese/Interpretation/Modell gesucht.

---

### 2.4.2 Das Gedächtnis und „Gateway to Memory“

---

Das Gehirn kann sich auf bestimmte Dinge fokussieren und den Rest ignorieren. Dabei gibt es drei verschiedene Arten der Aufmerksamkeit:

- *Gewählte Aufmerksamkeit* (selective): Zwischen mehreren Möglichkeiten wird eine zu fokussierende Sache aktiv ausgewählt.
  - Das Auge folgt den Objekten von Interesse.
  - Der Kopf folgt den Klängen von Interesse.
  - Es gibt nur einen einzigen „Ort der Aufmerksamkeit“.
- *Geteilte Aufmerksamkeit* (divided): Ein Versuch durch „Multitasking“ mehrere Dinge zu fokussieren.
  - Entweder „gleichzeitig“ oder durch schnelles Umschalten (time multiplexing).
  - Dies wirkt sich negativ auf die Verarbeitung aus, wenn die Aufgaben überfordernd sind.
  - Die Aufgaben beeinträchtigen sich gegenseitig.
- *Erfasste Aufmerksamkeit* (captured): Ein äußerer Reiz zieht alle Aufmerksamkeit auf sich.
  - Im Gegensatz zur gewählten Aufmerksamkeit wird der „Ort“ nicht aktiv ausgewählt.

---

– Dies geschieht z. B. wenn man von einem Tier angefallen wird.

Das menschliche Gedächtnis ist in mehrere „Teilgedächtnisse“ aufgeteilt. Voran steht das *Arbeitsgedächtnis*, auf das ein schneller Zugriff (ca. 70 ms) möglich ist, welches aber einen schnellen Verfall hat (nach ca. 200 ms). Nach wenigen Sekunden wird der Inhalt jedoch an das Langzeitgedächtnis weitergegeben. Es stellt sozusagen das „Schmierblatt“ des Gehirns da.

Das Langzeitgedächtnis ist langsamer (ca. 100 ms), dafür aber auch sehr viel größer (die genaue Größe ist unbekannt). Das Langzeitgedächtnis hat dabei drei Hauptaufgaben:

- Informationen speichern und sich an diese erinnern,
- Informationen abrufen und
- Informationen vergessen.



---

### 3 Computer Vision: Objekterkennung und Bayes

---

Die *Computer Vision* beschäftigt sich mit dem maschinellen Sehen, d. h. der Suche nach einem Modell des menschlichen Sehens. Anwendungsgebiete sind bspw. Autos, die Fußgänger erkennen, medizinische Bildverarbeitung, Überwachung, Unterhaltung, Computergraphik, . . .

---

#### 3.1 Computer Vision

---

Das einfachste Standardmodell einer Lochkamera ist ein Kasten mit einem kleinen Loch. Um ein digitales Bild eines solchen Kameramodells zu erhalten, wird das Bild rasterisiert. Demnach ist ein Graustufenbild eine Matrix an Pixeln mit jeweils einem Wert (die „Grauigkeit“ des Pixels).

Die Computer Vision beschäftigt sich nun damit, aus einem solchen Bild Informationen zu extrahieren. Bei der Objekterkennung ist es wichtig, eine gute lokale Beschreibung/Merkmale zu haben (z. B. Augen, Mund, Nase) und eine globale Anordnung der lokalen Merkmale (z. B. relative Positionen, relative Größen). Es ist aber auch eine schnelle Generierung guter Hypothesen, Segmentierung der Bildbereiche und kennen des Szenenkontextes wichtig.

Nach Fischler und Elschlager hat das Modell eines Bildes zwei Komponenten: Teile (2D Bildfragmente) und den Aufbau (die Anordnung der Teile). Mit diesem abstrakten Modell lassen sich viele Dinge (z. B. ein Gesicht) charakterisieren.

---

#### 3.2 Bayesian Decision Theory

---

Beispiel: Buchstabenerkennung. Es soll ein neu aufgenommener Buchstabe so klassifiziert werden, dass die Wahrscheinlichkeit der Fehlklassifikation minimiert wird.

---

##### 3.2.1 Konzepte und Bayes Theorem

---

#### Vorbemerkung: Wahrscheinlichkeitsdichte und Wahrscheinlichkeit

Ist  $p(x)$  eine Wahrscheinlichkeitsdichte, so ist die Wahrscheinlichkeit, dass  $x$  im Intervall  $(x_0, x_1)$  liegt, gegeben durch:

$$P(x_0 < x < x_1) = \int_{x_0}^{x_1} p(\tau) d\tau$$

Da für die Wahrscheinlichkeit, dass  $x$  im Intervall  $(x, x + \Delta x)$  mit  $\Delta x \rightarrow 0$  gilt:

$$\lim_{\Delta x \rightarrow 0} P(x) = \lim_{\Delta x \rightarrow 0} P(x < t < x + \Delta x) = p(x) \cdot \Delta x$$

kann Wahrscheinlichkeitsdichte und Wahrscheinlichkeit in den meisten Fällen gegeneinander ausgetauscht werden.

#### 1. Konzept: A-Priori Wahrscheinlichkeit (Prior)

Die *a-priori Wahrscheinlichkeit* (Prior) enthält die Information, wie wahrscheinlich eine beliebige Messung der Klasse zugehört (d. h. die „Klassenhäufigkeit“). Ist  $C_k$  eine Klasse, so ist  $P(C_k)$  der Prior bzgl. der Klasse  $C_k$  (analog für  $p(C_k)$ ).

---

## 2. Konzept: Bedingte Wahrscheinlichkeit (Likelihood)

Ist  $\mathbf{x}$  der Merkmalsvektor (Feature), welcher Eigenschaften der Messung beschreibt (Anzahl schwarzer Pixel, Höhe/Breite, ...) und  $C_k$  eine Klasse, so ist  $P(\mathbf{x} | C_k)$  die *Likelihood*, d. h. die Wahrscheinlichkeit, dass  $\mathbf{x}$  für einen Buchstaben der Klasse  $C_k$  gemessen wird (analog für  $p(\mathbf{x} | X_k)$ ).

## 3. Konzept: A-Posteriori Wahrscheinlichkeit (Posterior), Bayes Theorem

Die *a-posteriori Wahrscheinlichkeit* (Posterior) ist die Wahrscheinlichkeit, dass ein Merkmalsvektor  $\mathbf{x}$  einer Klasse  $C_k$  angehört, d. h.  $P(C_k | \mathbf{x})$ . Dieser Posterior kann durch Bayes Theorem gefunden werden:

$$P(C_k | \mathbf{x}) = \frac{P(\mathbf{x} | C_k) \cdot P(C_k)}{P(\mathbf{x})}$$

Oder Namentlich:

$$\text{Posterior} = \frac{\text{Likelihood} \times \text{Prior}}{\text{Normalisierung}}$$

---

### 3.2.2 Problemstellung

---

Abbildung 3.1 zeigt die Likelihood, Prior und den Posterior auf. Die Zielstellung eines Bayesian Classifier ist nun, die Wahrscheinlichkeit der Fehlklassifikation zu minimieren und somit eine Entscheidungsgrenze zu bestimmen. Die Wahrscheinlichkeit eines Fehlers ist gegeben durch:

$$\begin{aligned} P(\text{Fehler}) &= P(x \in R_2, C_1) + P(x \in R_1, C_2) \\ &= P(x \in R_2 | C_1)P(C_1) + P(x \in R_1 | C_2)P(C_2) \\ &= \int_{R_2} p(x \in R_2 | C_1)P(C_1) dx + \int_{R_1} p(x \in R_1 | C_2)P(C_2) dx \end{aligned}$$

Dabei ist  $P(x \in R_i, C_j)$  die Wahrscheinlichkeit, dass  $x$  zu Klasse  $R_i$  gehört, aber als Klasse  $C_j$  klassifiziert wurde (für  $i \neq j$  entspricht dies einer Fehlklassifikation).

---

### 3.2.3 Entscheidungsregel

---

Durch die Minimierung des Erwartungswertes des Fehlers kann die Entscheidungsregel, wann  $x$  in eine Klasse einsortiert wird, hergeleitet werden. Dabei soll  $x$  genau dann in Klasse  $C_1$  sortiert werden, wenn

$$P(C_1 | x) > P(C_2 | x)$$

Da die Posteriors im Allgemeinen nicht bekannt sind, werden die über Bayes Theorem berechnet:

$$\begin{aligned} &P(C_1 | x) > P(C_2 | x) \\ \Leftrightarrow &\frac{P(x | C_1)P(C_1)}{P(x)} > \frac{P(x | C_2)P(C_2)}{P(x)} \\ \Leftrightarrow &P(x | C_1)P(C_1) > P(x | C_2)P(C_2) \\ \Leftrightarrow &\frac{P(x | C_1)}{P(x | C_2)} > \frac{P(C_2)}{P(C_1)} \end{aligned}$$

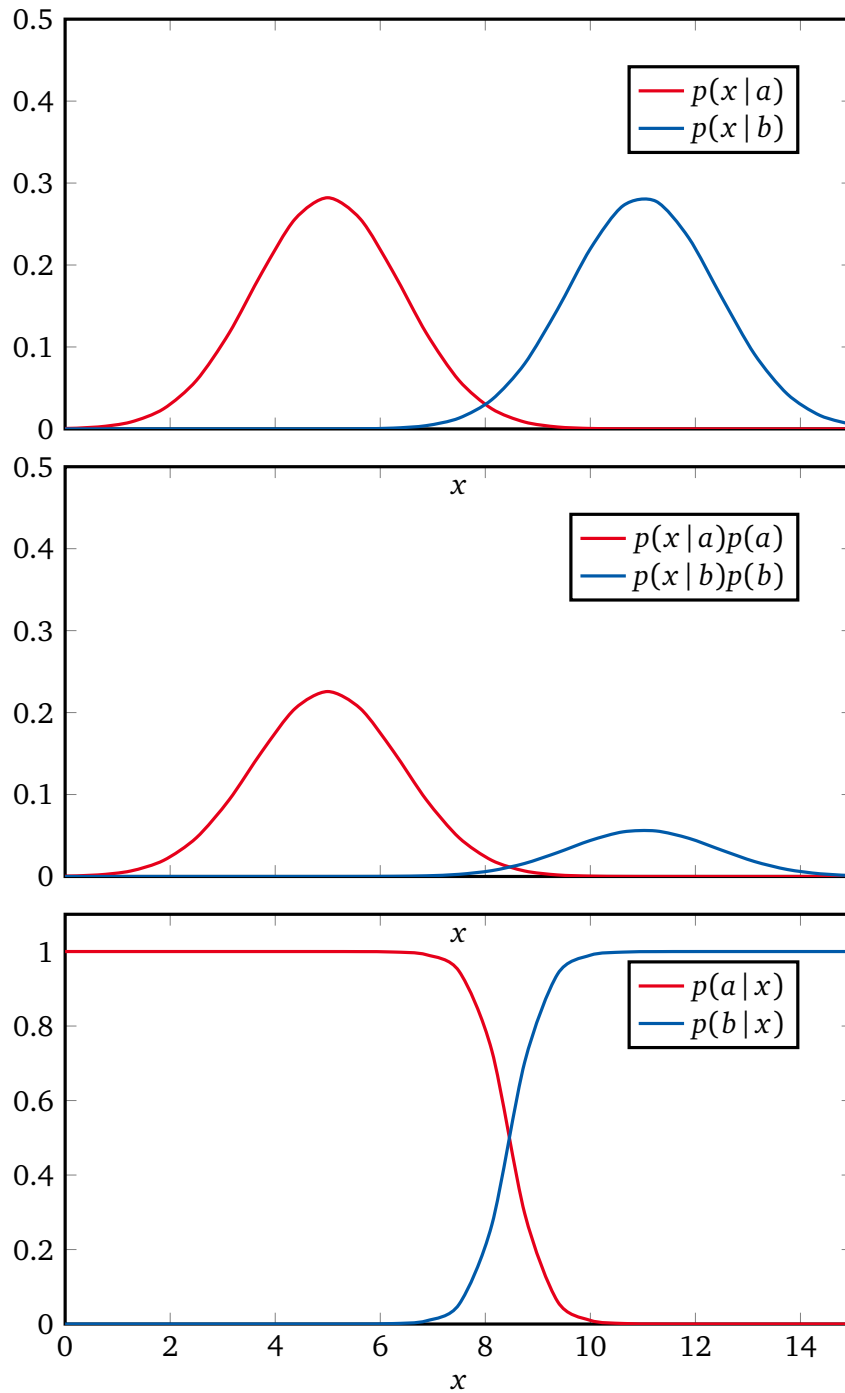
Dies wird auch *Likelihood Ratio Test* genannt.

Dieser Test kann sich für mehr als zwei Klassen verallgemeinern lassen: Wähle Klasse  $k$  genau dann, wenn

$$P(C_k | x) > P(C_j | x) \quad \forall j \neq k$$

gilt. Äquivalent zu dem zwei-Klassen-Fall kann dies in einen Likelihood Ratio Test umgeformt werden:

$$\frac{P(x | C_k)}{P(x | C_j)} > \frac{P(C_j)}{P(C_k)} \quad \forall j \neq k$$



**Abbildung 3.1:** Likelihood, Likelihood  $\times$  Prior und Posterior.

---

### 3.2.4 Naive Bayes Classifier

---

Bei mehr als zwei Merkmalen (z. B. Höhe und Breite) werden  $P(x_1, x_2 | C_k)$  und  $P(x_1, x_2)$  mehrdimensional und eine Schätzung der Dichte ist nicht immer möglich. Daher nimmt ein *Naive Bayes Classifier* an, dass die Merkmale statistisch unabhängig sind. Damit gilt:

$$P(x_1, x_2 | C_k) = P(x_1 | C_k)P(x_2 | C_k)$$
$$P(x_1, x_2) = P(x_1)P(x_2)$$

In der Realität ist diese Annahme oft nicht korrekt, liefert aber häufig gute Ergebnisse und ist somit eine gute Basis zum Vergleich.

---

### 3.3 Probability Density Estimation

---

Bisher wurden die Wahrscheinlichkeiten  $P(x | C_k)$  und  $P(C_k)$  als bekannt vorausgesetzt. In der Realität ist dies oft nicht der Fall, weshalb die Wahrscheinlichkeitsdichte geschätzt werden muss. Siehe hierzu auch Vorlesung Statistical Machine Learning.

---

### 3.4 Gesichtsdetektion

---

Bei *Appearance-Bases Methods* wird ein Erscheinungsmodell aus (üblicherweise) großen Mengen von Bildern gelernt. Dabei wird am häufigsten der Sliding Window Ansatz genutzt (siehe 3.4.1). Dabei sind vor allem drei Aspekte relevant:

1. Repräsentation des Objektes (lokale Merkmale, globale Anordnung)
2. Trainingsdaten (positive und negative Beispiele)
3. Klassifikator und Lernmethode

---

#### 3.4.1 Sliding Window Ansatz

---

Bei dem *Sliding Window Ansatz* wird ein Bild in Ein-Pixel-Schritten horizontal und vertikal gescannt. Nach jedem Durchlauf wird das Bild immer wieder verkleinert, bis das Bild zu klein ist. So können auch mit einem Klassifikator, der nur Bilder einer Größe entgegen nehmen kann, große Bilder durchsucht werden.

---

#### 3.4.2 Beispiel: Gesichtsdetektion

---

1. Repräsentation des Objekts
  - Die Bilder werden in Wavelets zerlegt, d. h. die Gesichtsmarkmale werden mit Frequenzen und deren Ort und Orientierung dargestellt.
  - Lokale Merkmale: Wavelet Koeffizienten (Frequenzen von z. B. Auge und Mund).
  - Globale Merkmale: Absolute Position der Frequenzen im Bild.
2. Trainingsdaten
  - Positive Beispiele
    - Möglichst vielfältig.

- 
- Jedes Bild eines Gesichts wird manuell an den Rändern abgeschnitten und auf eine Größe normalisiert.
  - Zusätzlich werden virtuelle Beispiele erstellt (z. B. durch Spiegelung).
  - Negative Beispiele
    - Beliebige Bilder, die keine Gesichter enthalten.
    - Teilbilder von großen Bildern.

### 3. Klassifikator und Lernmethode

- Naive Bayes Classifier
- Merkmale  $x_i$ : Wavelet Koeffizienten an einer bestimmten Position.
- Zwei-Klasse-Problem:
  - $C_1$ : Gesichter
  - $C_2$ : Alles andere (keine Gesichter)
- Das „Lernen“ entspricht dem Schätzen der Wahrscheinlichkeiten der Wavelet-Koeffizienten.
- Durch Diskretisierung von Koeffizienten und Positionen gibt es eine diskrete und endliche Anzahl von  $x_i$ .
- Schätzen: Zählen, wie häufig jedes  $x_i$  in Bildern mit und ohne Gesichtern vorkommt.
- Dann wird ein Likelihood Ratio Test verwendet.

Um Bilder aus verschiedenen Perspektiven zu erkennen, wird für jede Ansicht ein eigener Detektor verwendet (jeder für eine Ansicht) und diese kombiniert.

---

#### 3.4.3 Erkennungsarten

---

Eine Gesichtserkennung zählt zu den biometrischen Verfahren und werden bspw. in sicherheitstechnischen, kriminalistischen und forensischen Gebieten eingesetzt. Der Zweck ist die Identifikation und Verifikation natürlicher Personen.

- Verifikation: Die Person muss dem System ihren Namen oder User-ID mitteilen und das System entscheidet, ob die Person dazu gehört.
- Identifikation: Die Person offenbart ausschließlich ihre biometrischen Merkmale und das System ermittelt daraus den Namen oder die User-ID.

---

## 4 Fouriertheorie

---

Bei der Beugung an einem einfachen Spalt der Breite  $a$  ergibt sich auf dem Schirm ein Beugungsmuster, welches im Zentrum ein Intensitätsmaximum und nach außen hin immer wieder Intensitätsminima und -maxima hat. Der Spalt kann durch eine Rechteckfunktion

$$\text{Rect}(x) = \begin{cases} 1 & -1 \leq x \leq 1 \\ 0 & \text{sonst} \end{cases}$$

beschrieben werden. Das sich ergebende Beugungsmuster, bzw. die zeitlich gemittelte Intensität  $I$ , hat dann die Form

$$I(\theta) = I_0 \left( \frac{\sin(\theta)}{\theta} \right)^2 = I_0 \cdot \text{sinc}^2(\theta)$$

mit der sinc-Funktion  $\text{sinc}(\theta) = \sin(\theta)/\theta$ . Dabei stellt  $\theta$  den Ausfallwinkel des Lichts aus dem Spalt hinaus dar.

Dieser Zusammenhang zwischen der Gestalt des beugenden Objekts (hier der Spalt) und der Amplitudenfunktion  $I(\theta)$  ist durch eine *Fourier-Transformation* gegeben.

---

### 4.1 Mathematische Grundlagen

---

#### 4.1.1 Vektorraum

---

Ein *Vektorraum* ist eine algebraische Struktur über einen Zahlenbereich mit Operationen wie Addition und Multiplikationen mit einem Skalar. Alle Operationen müssen dabei Elemente des Vektorraums wieder auf selbigen abbilden. Die Elemente eines solchen Raums sind *Vektoren*.

**Beispiel:** Ein Beispiel ist der euklidische Vektorraum über den reellen Zahlen. Dabei repräsentieren Vektoren Verschiebungen und es lassen sich Längen und Winkel messen (rechtwinkliges, kartesisches Koordinatensystem). Es ist außerdem ein Skalarprodukt definiert:

$$\langle \mathbf{v}, \mathbf{w} \rangle = \sum_{i=1}^n v_i w_i \in \mathbb{R}$$

$$\langle \mathbf{v}, \mathbf{w} \rangle = v_1 w_1 + v_2 w_2 = \|\mathbf{v}\| \cdot \|\mathbf{w}\| \cos(\angle(\mathbf{v}, \mathbf{w}))$$

Die letztere Eigenschaft gilt nur für  $n = 2$  (i. A. lassen sich solche Winkel aber auch mit beliebigem  $n$  definieren). In der euklidischen Ebene  $\mathbb{R}^2$  lassen sich Vektoren durch Ortsvektoren (Pfeile) darstellen.

---

#### 4.1.2 Basis eines Vektorraums

---

Jeder Satz (Menge) an linear unabhängigen Vektoren eines Vektorraums kann als Basis verwendet werden. Zwei Vektoren  $\mathbf{v}$ ,  $\mathbf{w}$  sind genau dann linear unabhängig, wenn  $|\langle \mathbf{v}, \mathbf{w} \rangle| < \|\mathbf{v}\| \cdot \|\mathbf{w}\|$  gilt.

**Beispiel:** In der euklidischen Ebene  $\mathbb{R}^2$  ist eine Basis durch

$$\mathbf{e}_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad \mathbf{e}_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

gegeben, wobei  $\mathbf{e}_1$  und  $\mathbf{e}_2$  orthogonal aufeinander stehen ( $\langle \mathbf{v}, \mathbf{w} \rangle = 0$ ) und somit linear unabhängig sind. Alle  $\mathbf{v} \in \mathbb{R}^2$  lassen sich dann als *Linearkombination* der Basisvektoren darstellen (mit geeigneten  $a_1, a_2 \in \mathbb{R}$ ):

$$\mathbf{v} = a_1 \mathbf{e}_1 + a_2 \mathbf{e}_2$$

---

### 4.1.3 Krummlinige Koordinatensysteme

---

Gerade in physikalischen Anwendungen kann es von Vorteil sein, keine kartesischen Koordinaten (mit  $x$ - und  $y$ -Wert) zu nutzen, sondern auf *krummlinige Koordinaten* umzusteigen. Ein typisches krummliniges Koordinatensystem sind z. B. Polarkoordinaten. Dabei wird ein Punkt in der Ebene durch den Abstand  $r$  vom Ursprung und durch den Winkel  $\varphi$  mit der  $x$ -Achse beschrieben. Die Koordinaten lassen sich durch

$$\begin{aligned}x(r, \varphi) &= r \cdot \cos(\varphi) \\y(r, \varphi) &= r \cdot \sin(\varphi)\end{aligned}$$

in kartesische Koordinaten umrechnen.

Weitere krummlinige Koordinatensysteme sind z. B. Kugel- oder Zylinderkoordinaten.

---

### 4.1.4 Andere Räume

---

Es ist auch möglich, dass die Elemente eines Vektorraums Funktionen sind (Funktionenräume). Auch kann ein Raum unendlich-dimensional sein.

Die *Fourier-Theorie* beschäftigt sich mit der Frage, ob es möglich ist, Basisfunktionen zu finden, mit denen sich beliebige Funktionen bzgl. dieser Basen darstellen lassen.

---

### 4.1.5 Komplexe Zahlen

---

Komplexe Zahlen haben zwei Komponenten: Einen Real- und einen Imaginärteil. Dabei können sie als kartesische Koordinaten in einer zwei-dimensionalen Ebene (der komplexen Ebene) aufgefasst werden und entsprechen dargestellt werden (mit der imaginären Zahl  $i$  mit der Eigenschaft  $i^2 = -1$ ):

$$z = a + bi$$

Oder als Polarkoordinaten (in einer zwei-dimensionalen Ebene) mit der Darstellung

$$z = re^{i\varphi}$$

wobei sich kartesische und Polardarstellung wie bei Polarkoordinaten ineinander umrechnen lassen.

Die Äquivalenz der beiden Darstellung geht auf die Euler-Identität

$$e^{i\varphi} = \cos(\varphi) + i \sin(\varphi)$$

zurück, wobei hier  $r = 1$  gilt. Aus dieser folgt (für  $|z| = 1$ ) ebenfalls:

$$a = \cos(\varphi) = \frac{1}{2}(e^{i\varphi} + e^{-i\varphi}) \quad (4.1)$$

$$b = \sin(\varphi) = \frac{1}{2i}(e^{i\varphi} - e^{-i\varphi}) \quad (4.2)$$

---

### 4.1.6 Gerade/Ungerade Funktionen

---

Für eine gerade Funktion gilt

$$f(x) = f(-x)$$

für eine ungerade Funktion gilt

$$f(x) = -f(-x)$$

für jeweils alle  $x$ .

---

## 4.2 Fourier-Reihe

---

### 4.2.1 Dirichlet-Bedingungen

---

Jede Funktion, die die *Dirichlet-Bedingungen* erfüllt:

1. Die Anzahl Unstetigkeiten innerhalb einer Periode ist endlich.
2. Die Anzahl Maxima und Minima innerhalb einer Periode ist endlich.
3. Die Funktion ist in jeder Periode integrierbar (d. h. die Fläche unter dem Betrag der Funktion ist endlich).

Kann durch eine Summe von Kosinus- und Sinusfunktionen dargestellt werden.

---

### 4.2.2 $2\pi$ -periodische Funktion

---

Ist  $f(x)$  eine periodische Funktion mit der Periodenlänge  $2\pi$  (d. h. die wiederholt sich alle  $2\pi$ ), die die Dirichlet-Bedingungen erfüllt, so gilt

$$f(x) = \sum_{n=0}^{\infty} (a_n \cos(nx) + b_n \sin(nx))$$

mit geeigneten *Fourier-Koeffizienten*  $a_n$  und  $b_n$ .

---

### 4.2.3 Skalarprodukt, Orthogonale Basis

---

Sei  $H$  der Raum aller  $2\pi$ -periodischen reellen Funktionen, die die Dirichlet-Bedingungen erfüllen. Dann wird durch

$$\langle f, g \rangle := \int_{-\pi}^{\pi} f(\tau)g(\tau) \, d\tau$$

ein Skalarprodukt definiert.

Die Funktionen

$$u_n(x) = \cos(nx)$$

$$v_n(x) = \sin(nx)$$

bilden dann eine orthogonale Funktionenfolge in  $H$ :

$$\langle u_n, u_m \rangle = \begin{cases} 0 & m \neq n \\ 2\pi & m = n = 0 \\ \pi & m = n > 0 \end{cases}$$

$$\langle v_n, v_m \rangle = \begin{cases} 0 & m \neq n \\ 0 & m = n = 0 \\ \pi & m = n > 0 \end{cases}$$

$$\langle u_n, v_m \rangle = \langle v_m, u_n \rangle = 0$$

Durch diese Darstellung kann die allgemeine Fourier-Reihe mit  $u_n = u_n(x)$  und  $v_n = v_n(x)$  auch geschrieben werden als:

$$f(x) = \sum_{n=0}^{\infty} (a_n u_n + b_n v_n)$$



---

#### 4.2.4 Berechnung der Koeffizienten $a_m, b_m$

---

Um die Koeffizienten  $a_m, m = 1, 2, \dots$  zu bestimmen, wird das Skalarprodukt zwischen  $f$  und  $u_m$  gebildet:

$$\langle f, u_m \rangle = \left\langle \sum_{n=0}^{\infty} (a_n u_n + b_n v_n), u_m \right\rangle = \langle (a_m u_m + b_m v_m), u_m \rangle = \langle a_m u_m, u_m \rangle = a_m \langle u_m, u_m \rangle = a_m \pi$$

Umstellen nach  $a_m$  liefert die Werte der Fourier-Koeffizienten:

$$a_m = \frac{1}{\pi} \langle f, u_m \rangle = \frac{1}{\pi} \int_{-\pi}^{\pi} f(x) \cos(mx) \, dx$$

Analog für  $a_0$  mit  $u_0$ :

$$\langle f, u_0 \rangle = \left\langle \sum_{n=0}^{\infty} (a_n u_n + b_n v_n), u_0 \right\rangle = \langle (a_0 u_0 + b_0 v_0), u_0 \rangle = \langle a_0 u_0, u_0 \rangle = a_0 \langle u_0, u_0 \rangle = a_0 2\pi$$

Umstellen nach  $a_0$ :

$$a_0 = \frac{1}{2\pi} \langle f, u_0 \rangle = \frac{1}{2\pi} \int_{-\pi}^{\pi} f(x) \cos(0x) \, dx = \frac{1}{2\pi} \int_{-\pi}^{\pi} f(x) \, dx$$

Analog für  $b_m, m = 1, 2, \dots$ :

$$\langle f, v_m \rangle = \left\langle \sum_{n=0}^{\infty} (a_n u_n + b_n v_n), v_m \right\rangle = \langle (a_m u_m + b_m v_m), v_m \rangle = \langle b_m v_m, v_m \rangle = b_m \langle v_m, v_m \rangle = b_m \pi$$

Umstellen nach  $b_m$ :

$$b_m = \frac{1}{\pi} \langle f, v_m \rangle = \frac{1}{\pi} \int_{-\pi}^{\pi} f(x) \sin(mx) \, dx$$

Da  $\sin(0x) = \sin(0) = 0$  ist, muss  $b_0$  nicht berechnet werden.

---

#### 4.2.5 Beispiel: Rechteck-Schwingung

---

Sei eine Rechteck-Schwingung

$$f(x) = \begin{cases} -k & -\pi < x < 0 \\ k & 0 < x < \pi \end{cases}, \quad f(x) = f(x + 2\pi)$$

gegeben. Für diese lauten die Fourier-Koeffizienten:

$$a_0 = 0$$

$$a_n = 0$$

$$b_n = \frac{4k}{n\pi} \text{ für ungerade } n$$

Daraus ergibt sich die Fourier-Reihe:

$$f(x) = \frac{4k}{\pi} \sum_{n=0}^{\infty} \frac{1}{2k+1} \sin((2k+1)x)$$

Die Rechteck-Schwingung ist dabei eine ungerade Funktion. Allgemein gilt:

- Für gerade Funktionen sind alle  $b_n = 0$ .
- Für ungerade Funktionen sind all  $a_n = 0$ .

### 4.3 Fourier-Transformation

Mit der Fourier-Transformation wird versucht, eine ähnliche Darstellung wie die Fourier-Reihe für Funktionen zu finden, die nicht  $2\pi$ -periodisch sind.

Durch die Euler-Identität kann die allgemeine Fourier-Reihe umgeformt werden:

$$\begin{aligned}
 f(x) &= a_0 + \sum_{n=1}^{\infty} (a_n \cos(nx) + b_n \sin(nx)) \\
 &= a_0 + \sum_{n=1}^{\infty} \left( a_n \frac{e^{inx} + e^{-inx}}{2} + b_n \frac{e^{inx} - e^{-inx}}{2i} \right) \\
 &= a_0 + \sum_{n=1}^{\infty} \left( a_n \frac{e^{inx} + e^{-inx}}{2} - b_n i \frac{e^{inx} - e^{-inx}}{2} \right) \\
 &= a_0 + \sum_{n=1}^{\infty} \left( \frac{a_n - ib_n}{2} e^{inx} + \frac{a_n + ib_n}{2} e^{-inx} \right) \\
 &= a_0 + \sum_{n=1}^{\infty} \frac{a_n - ib_n}{2} e^{inx} + \sum_{n=1}^{\infty} \frac{a_n + ib_n}{2} e^{-inx} \\
 &= a_0 + \sum_{n=1}^{\infty} \frac{a_n - ib_n}{2} e^{inx} + \sum_{n=-\infty}^{-1} \frac{a_{-n} + ib_{-n}}{2} e^{inx} \\
 &= c_0 + \sum_{n=1}^{\infty} c_n e^{inx} + \sum_{n=-\infty}^{-1} c_n e^{inx} \\
 &= \sum_{n=-\infty}^{\infty} c_n e^{inx}
 \end{aligned}$$

woraus sich eine äquivalente Formulierung der Fourier-Reihe mit den komplexen Koeffizienten

$$c_n = \frac{a_n - ib_n}{2} e^{inx}, \quad n = 1, 2, \dots \qquad c_n = \frac{a_{-n} + ib_{-n}}{2}, \quad n = -1, -2, \dots \qquad c_0 = a_0$$

ergibt. Nun werden zunächst Funktionen  $f_L(x)$  mit einer beliebigen Periode  $2L$  betrachtet:

$$\begin{aligned}
 f_L(x) &= \sum_{n=-\infty}^{\infty} c_n e^{in \frac{2\pi}{2L} x} \\
 &= \sum_{n=-\infty}^{\infty} c_n e^{in \frac{\pi}{L} x}
 \end{aligned}$$

Einsetzen der Koeffizienten  $c_n$ :

$$= \sum_{n=-\infty}^{\infty} \left( \frac{1}{2L} \int_{-L}^L f(\tau) e^{-in \frac{\pi}{L} \tau} d\tau \right) e^{in \frac{\pi}{L} x}$$

Nun wird der Übergang  $L \rightarrow \infty$ , d. h. zu nicht-periodischen Funktionen, betrachtet:

$$\begin{aligned}
 \lim_{L \rightarrow \infty} f(x) &= \lim_{L \rightarrow \infty} \sum_{n=-\infty}^{\infty} \left( \frac{1}{2L} \int_{-L}^L f(\tau) e^{-in\frac{\pi}{L}\tau} d\tau \right) e^{in\frac{\pi}{L}x} \\
 &= \lim_{L \rightarrow \infty} \sum_{n=-\infty}^{\infty} \frac{1}{2L} \int_{-L}^L f(\tau) e^{-in\frac{\pi}{L}(\tau-x)} d\tau \\
 &= \lim_{L \rightarrow \infty} \sum_{n=-\infty}^{\infty} \int_{-L}^L \frac{1}{2L} f(\tau) e^{-in\frac{\pi}{L}(\tau-x)} d\tau \\
 &= \lim_{L \rightarrow \infty} \int_{-L}^L \sum_{n=-\infty}^{\infty} \frac{1}{2L} f(\tau) e^{-in\frac{2\pi}{2L}(\tau-x)} d\tau \\
 &= \lim_{L \rightarrow \infty} \int_{-L}^L f(\tau) \int_{-\infty}^{\infty} e^{-2\pi i u(\tau-x)} du d\tau \\
 &= \int_{-\infty}^{\infty} f(\tau) \int_{-\infty}^{\infty} e^{-2\pi i u(\tau-x)} du d\tau
 \end{aligned}$$

Dieser Übergang lässt sich als „Superposition“ auffassen mit:

$$\begin{aligned}
 f(x) &= \int_{-\infty}^{\infty} F(u) e^{2\pi i u x} du \\
 F(u) &= \int_{-\infty}^{\infty} f(x) e^{-2\pi i u x} dx
 \end{aligned}$$

Dabei heißt der Übergang  $f(x) \rightarrow F(u)$  *Fourier-Transformation* und der Übergang  $F(u) \rightarrow f(x)$  *Inverse Fourier-Transformation*. Dabei ist  $F(u)$  oft komplex und  $f(x)$  ist reell.

### 4.3.1 Beispiel: Rechteckimpuls

Für einen Rechteckimpuls

$$f(x) = \begin{cases} 1 & -1 < x < 1 \\ 0 & \text{sonst} \end{cases}$$

ergibt sich die Fourier-Transformation

$$F(u) = \int_{-1}^1 e^{-2\pi i u \tau} d\tau = \frac{1}{2\pi i u} [e^{-2\pi i u \tau}]_{-1}^1 = \frac{1}{\pi u} \cdot \frac{e^{2\pi i u} - e^{-2\pi i u}}{2i} = 2 \frac{\sin(2\pi u)}{2\pi u} = 2 \operatorname{sinc}(2\pi u)$$

wie erwartet ein Vielfaches der sinc-Funktion.

### 4.3.2 Transformationspaare

Die Fourier-Transformation zerlegt eine Funktion in ihre Frequenzbestandteile! Beispielhafte Fourier-Transformationspaare sind:

- $\cos(0) = 1$ : Delta-Funktion bei  $u = 0$
- $\cos(kx)$ : Delta-Funktion bei  $u = \pm k$
- $\sin(kx)$ : Delta-Funktion bei  $u = \pm ik$

---

### 4.3.3 2D-Fourier-Transformation

---

Für eine zweidimensionale Funktion  $f(x, y)$  lautet die Fourier-Transformation:

$$f(u, v) = \iint_{-\infty}^{\infty} f(x, y) e^{2\pi i(xu+vy)} \, dx \, dy$$
$$F(x, y) = \iint_{-\infty}^{\infty} f(u, v) e^{-2\pi i(ux+vy)} \, du \, dv$$

Die Fourier-Transformierte ist entspricht also zweidimensionalen Funktionen (Real- und Imaginärteil), die als Graustufenbilder visualisiert werden können. Meistens wird dabei aber nur das sogenannte Amplituden-Spektrum betrachtet, welches die Amplituden der Fourier-Transformation visualisiert. Dabei entspricht der Pixelwert an der Stelle  $(u, v)$  der Amplitude, d. h. dem Betrag, der Frequenzen  $|F(u, v)|$ .

---

### 4.4 Faltung

---

Werden zwei Funktionen  $F(u), G(u)$  im Frequenzraum multipliziert:

$$\begin{aligned} F(u) \cdot G(u) &= \int_{-\infty}^{\infty} f(\tau) e^{-2\pi i u \tau} \, d\tau \cdot \int_{-\infty}^{\infty} g(t) e^{-2\pi i u t} \, dt \\ &= \int_{-\infty}^{\infty} f(\tau) e^{-2\pi i u \tau} \int_{-\infty}^{\infty} g(t - \tau) e^{-2\pi i u (t - \tau)} \, dt \, d\tau \\ &= \int_{-\infty}^{\infty} e^{-2\pi i u t} \underbrace{\int_{-\infty}^{\infty} f(\tau) g(t - \tau) \, d\tau}_{h(t) :=} \, dt \\ &= \int_{-\infty}^{\infty} h(t) e^{-2\pi i u t} \, dt \\ &= H(u) \end{aligned}$$

Das Integral  $h(t) = \int_{-\infty}^{\infty} f(\tau) g(t - \tau) \, d\tau =: f(t) * g(t)$  ist das sogenannte *Faltungsintegral* der Funktionen  $f$  und  $g$ . Eine Faltung im Ortsraum entspricht somit einer Multiplikation im Frequenzraum!

Eine Faltung  $f(t) * g(t)$  kann als Mittelwertbildung der Werte von  $f$  mit Gewichten  $g$  verstanden werden. So kann bspw. analytisch ein gleitender Durchschnitt (mit einer Kastenfunktion  $g$ ) erstellt werden.

---

#### 4.4.1 Anwendung: Filter

---

### 4.5 Abtastung

---

Ist eine kontinuierliche Funktion, bzw. ein analoges Signal, gegeben, so muss dieses für eine diskrete Repräsentation *abgetastet* werden, d. h. es müssen Messungen an einzelnen Stellen durchgeführt werden. Eine solche diskrete Abtastung kann durch die Funktion

$$\hat{f}(x) = f(x) \cdot \sum_{n=-\infty}^{\infty} \delta(x - n \cdot \Delta x)$$

d. h. als Produkt einer Funktion  $f(x)$  und einer Kamm-Funktion beschrieben werden. Die Fourier-Transformierte  $\hat{F}(u)$  der abgetasteten Funktion entspricht dann der Fourier-Transformierten  $F(u)$  der nicht abgetasteten Funktion, wird aber periodisch mit der Periode  $1/\Delta x$  wiederholt und mit  $1/\Delta x$  skaliert.

---

---

### 4.5.1 Abtasttheorie

---

Sei die Funktion  $f(x)$  bandbegrenzt durch eine Maximalfrequenz  $u_G$ , d. h.  $F(u) = 0$  für  $|u| > u_G$ .

Gilt nun  $2u_G < 1/\Delta x$ , so überlappen sich die Fouriertransformierten nicht, d. h. die Spektren von  $F(u)$  und  $\hat{F}(u)$  stimmen auf dem Intervall  $[-u_G, u_G]$  (bis auf die Skalierung  $1/\Delta x$ ) überein. Das Frequenzspektrum von  $F(u)$  kann somit vollständig aus dem Abtastsignal und den Abtastwerten berechnet werden.

Gilt nun  $2u_G > 1/\Delta x$ , so überlappen sich die Fouriertransformierten und in den Überschneidungsbereichen bilden sich Summen. Damit ist es unmöglich, das originale Frequenzspektrum von  $F(u)$  zu bestimmen (*Aliasing*).

---

### 4.5.2 Abtasttheorem von Whittaker-Shannon

---

Aus den vorherigen Überlegungen ergibt sich das *Abtasttheorem von Whittaker-Shannon*: Existiert für eine Funktion  $f(x)$  eine Grenzfrequenz  $u_G < \infty$ , sodass  $F(u) = 0$  für  $|u| > u_G$  gilt, dann ist  $f(x)$  fehlerfrei rekonstruierbar, sofern die Abtastfrequenz  $1/\Delta x$  mindestens doppelt so hoch wie  $u_G$  ist:

$$\frac{1}{\Delta x} > 2u_G$$

---

## 5 Bilder

---

### 5.1 Bildverbesserung

---

Bei der Bildverbesserung wird versucht, die Bildinformationen so aufzubereiten, dass die für den Betrachter verbessert sind/wirken. Dafür gibt es (leider) keine allgemeine Theorie, sondern die möglichen Verbesserungen sind sehr Anwendungsspezifisch und abhängig von Bild und Betrachter. Typische Anwendungen sind dabei der Ausgleich von nicht-Linearitäten der Kamera, Anpassung von Helligkeit und Kontrast und Hervorhebung von Bildbereichen.

Es wird unterschieden zwischen Methoden im Ortsraum (die direkt Pixelwerte manipulieren) und Methoden im Frequenzraum, bei denen das Bild zunächst durch eine Fourier-Transformation in seine Frequenzen zerlegt, manipuliert und rücktransformiert wird.

---

#### 5.1.1 Histogramm

---

Das Histogramm eines Bildes ist die graphische Darstellung der Häufigkeitsverteilung von bestimmten Merkmalen (z. B. von bestimmten Grauwerten). Histogramme von Bildern können viele Aussagen treffen, z. B. über:

- Dynamik (Bereich reeller Lichtintensitäten, er auf der Grauwertskala abgebildet wird)
- Kontrast (Bereich der Grauwertskala, der zur Darstellung ausgenutzt wird)
- Helligkeit (Beleuchtungsstärke (der Grauwert))

Dabei entspricht die Helligkeit eines Grauwertbildes dem Mittelwert aller Grauwerte und der Bildkontrast der Varianz aller Grauwerte.

---

#### 5.1.2 Pixeloperationen

---

Bei Pixeloperationen wird ein Pixel unabhängig von seiner Nachbarschaft modifiziert. Beispiele für solche Operationen sind:

- Negativ
- Binärisierung/Thresholding
- Fensterung
- Kontrastspreizung
- Dynamikkompression
- Gammakorrektur (Bildschirm)
- Helligkeit
- Histogrammausgleich
- Differenz
- Mittelung

---

## Bildnegativ

---

Bei einem Bildnegativ wird der Wert eines Pixels von der Maximal möglichen Intensität abgezogen und dieser Wert als neuer Pixelwert verwendet:

$$g[m, n] = f_{\max} - f[m, n]$$

---

## Binärisierung/Thresholding

---

Bei der Binärisierung wird ein Schwellwert  $\tau$  sowie zwei Werte  $f_{\max}$  und  $f_{\min}$  festgelegt und das Bild wie folgt manipuliert:

$$g[m, n] = \begin{cases} f_{\max} & f[m, n] > \tau \\ f_{\min} & f[m, n] \leq \tau \end{cases}$$

Für den Spezialfall der Binärisierung gilt  $f_{\max} = 1$  und  $f_{\min} = 0$ .

---

## Grauwertfensterung

---

Bei der Grauwertfensterung wird ein bestimmtes Intensitätsintervall hervorgehoben (gespreizt) und alles andere wird auf einen fixen Wert gesetzt.

---

### 5.1.3 Kontrastspreizung

---

Bei der Kontrastspreizung wird der Grauwert auf eine neue Grauwertskala anhand einer einwertigen oder monotonen Funktion abgebildet.

---

### 5.1.4 Histogrammausgleich

---

Bei einem Histogrammausgleich wird die Grauwertskala anhand der Kurve der Summenwahrscheinlichkeiten, d. h. anhand der kumulierten Wahrscheinlichkeiten bis zu einem bestimmten Wert, transformiert:

$$p(g) = \max(\text{Intensität}) \cdot \sum_{i=0}^g p(i)$$

Ein solcher Histogrammausgleich ist verlustbehaftet und nicht umkehrbar!

---

### 5.1.5 Mittelung

---

Ein unkorreliertes Rauschen im Bild kann durch Mittelung über  $k$  Aufnahmen des gleichen Motivs unterdrückt werden:

$$g[m, n] = \frac{1}{k} \sum_{i=0}^{k-1} f_i[m, n]$$

---

## 5.2 Bildfilterung

---

Zur Bildfilterung gibt es zwei grundlegende Vorgehensweise:

- Filterung im Ortsraum durch direkte Manipulation der Pixel und
- Filterung im Frequenzraum durch vorherige Fourier-Transformation und Rücktransformation.

---

## 5.2.1 Ortsraum

---

Filter im Ortsraum werden durch *Filtermasken* beschrieben. Dabei wird ein Pixel in Abhängigkeit von seiner Nachbarschaft modifiziert. Eine Filtermaske wird durch eine  $k \times l$ -Matrix (mit  $k, l$  ungerade) beschrieben, die die Gewichtung der umliegenden Pixel beschreibt. Dies entspricht einer linearen Filterung (Faltung) im Ortsraum:

$$(f * w)[m, n] = \sum_{i=-\lfloor k/2 \rfloor}^{\lfloor k/2 \rfloor} \sum_{j=-\lfloor l/2 \rfloor}^{\lfloor l/2 \rfloor} w_{ij} f[m+i, n+j]$$

Dabei entspricht  $w \in \mathbb{R}^{k \times l}$  der Filtermaske.

---

### Tiefpass-Filter

---

- Die Koeffizienten (d. h. die Einträge der Filtermaske) sind allesamt positiv und normalisiert, sodass die Summe 1 ergibt.
- Dadurch werden nur positive Werte produziert.
- Es kommt zu Randeffekten, da am Rand die Nachbarn eines Pixels nicht definiert sind.
- Typische Vertreter dieser Kategorie sind Mittelwert- und Gauß-Filter.

### Mittelwert-Filter

$3 \times 3$ -Mittelwertfilter („Boxfilter“):

$$\frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

$5 \times 5$ -Mittelwertfilter:

$$\frac{1}{25} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Je größer der Filter gewählt wird, desto mehr wird das Bild „verwischt“.

### Gauß-Filter

Bei einem Gauß-Filter werden die umliegenden Pixel durch eine diskrete Approximation der Funktion

$$G(x, y) = \frac{1}{2\pi\sigma^2} \exp\left\{-\frac{x^2 + y^2}{2\sigma^2}\right\}$$

gewichtet. Für  $3 \times 3$  („Binomialfilter“) und  $5 \times 5$  ( $\sigma = 1$ ) ergeben sich folgende Approximationen:

$$\frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \quad \frac{1}{273} \begin{bmatrix} 1 & 4 & 7 & 4 & 1 \\ 4 & 16 & 26 & 16 & 4 \\ 7 & 26 & 41 & 26 & 7 \\ 4 & 16 & 26 & 16 & 4 \\ 1 & 4 & 7 & 4 & 1 \end{bmatrix}$$



---

## Median-Filter

Der Median-Filter ist ein nichtlinearer Filter und kann daher nicht durch eine Faltung ausgedrückt werden. Bei ihm wird jeder Pixel durch den Medianwert seiner Nachbarschaft ersetzt. Dadurch werden keine Grautöne interpoliert, isolierte Punkte und Rauschen wird minimiert und die Schärfe der Kanten bleibt erhalten. Jedoch ist dieser Filter aufgrund der Sortierung sehr rechenintensiv.

---

## Hochpass-Filter

---

- Die Koeffizienten können sowohl negativ und positiv sein und sind normalisiert, sodass die Summe 0 ergibt.
- Dadurch werden positive und negative Werte produziert.
- Typischer Vertreter dieser Kategorie sind Ableitungen und Differenzfilter.

## Diskretisierung von Ableitungen

Eine Ableitung  $\partial f / \partial x$  einer Funktion kann durch Rückwärtsdifferenzen approximiert werden:

$$\frac{\partial f}{\partial x}(x) = \lim_{h \rightarrow 0} \frac{f(x) - f(x-h)}{h} \stackrel{h=1}{\approx} \frac{f(x) - f(x-1)}{1} = f(x) - f(x-1)$$

Dies gilt ebenfalls für die zweite Ableitung  $\partial^2 f / \partial x^2$  (diesmal durch Vorwärtsdifferenzen):

$$\begin{aligned} \frac{\partial^2 f}{\partial x^2}(x) &= \lim_{h \rightarrow 0} \frac{\frac{\partial f}{\partial x}(x+h) - \frac{\partial f}{\partial x}(x)}{h} \\ &\approx \lim_{h \rightarrow 0} \frac{f(x+h) - f(x+h-1) - f(x) + f(x-1)}{h} \\ &\approx \frac{f(x+1) - f(x) - f(x) + f(x-1)}{1} \\ &= f(x+1) - 2f(x) + f(x-1) \end{aligned}$$

## Laplacian-Filter

Durch den Laplacian-Filter wird der Laplace-Operator

$$\Delta f(x, y) = \nabla^2 f(x, y) = \frac{\partial^2 f(x, y)}{\partial x^2} + \frac{\partial^2 f(x, y)}{\partial y^2}$$

approximiert. Für  $3 \times 3$  ergibt sich bspw. die folgende diskrete Approximation:

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

Eine alternative Approximation mit Parametern stellt

$$\frac{1}{\beta} \begin{bmatrix} \alpha & 1-\alpha & \alpha \\ 1-\alpha & -4 & 1-\alpha \\ \alpha & 1-\alpha & \alpha \end{bmatrix}$$

da, wobei

$$\beta = \begin{cases} 4 & 0 \leq \alpha \leq 1 \\ 4(q-\alpha) & -1 \leq \alpha < 0 \end{cases}$$

gilt (der Parameter  $\alpha$  ist frei wählbar).

---

## Laplacian of Gaussian Filter

Oftmals wird zunächst ein Gauß- und danach ein Laplacian-Filter angewandt. Dieser Filter wird „Laplacian of Gaussian Filter“, „Marr-Hildreth-Operator“, „Mexican Hat Filter“ oder „Sombrefilter“ genannt. Dabei wird die Funktion

$$\Delta G(x, y) = \frac{\partial^2 G(x, y)}{\partial x^2} + \frac{\partial^2 G(x, y)}{\partial y^2}, \quad G(x, y) = \frac{1}{2\pi\sigma^2} \exp\left\{-\frac{x^2 + y^2}{2\sigma^2}\right\}$$

approximiert.

---

## Bilaterale Filter

---

Bei der bilateralen Filterung wird versucht, die Bilder weichzuzeichnen, aber scharfe Kanten zu erhalten. Dabei fließen Pixelfarben aus der Nachbarschaft nicht nur in Abhängigkeit von der Entfernung, sondern auch vom Farbabstand in die Berechnung ein.

---

### 5.2.2 Frequenzraum

---

Durch eine Multiplikation jeder Frequenz-Komponente  $F(u, v)$  anhand einer Gewichtungsfunktion (Filter), können bestimmte Komponenten erhöht oder verringert werden. Durch eine inverse Fouriertransformation werden die Veränderungen sichtbar. Eine solche selektive Beseitigung von Frequenz-Komponenten heißt *Fourier-Filterung*. Filter werden bspw. eingesetzt, um den Einfluss von Datenfehlern oder Störsignalen zu verringern, hoch-/niederfrequente Signale zu trennen oder bestimmte Frequenzen hervorzuheben.

Es werden dabei drei grundlegende Filtertypen unterschieden:

- Hochpass-Filter  
Tiefe Frequenzen  $|\omega| < D_0$  werden abgeschnitten und es können nur hohe Frequenzen passieren. Dadurch werden scharfe Übergänge deutlicher.
- Tiefpass-Filter  
Hohe Frequenzen  $|\omega| > D_0$  werden abgeschnitten und es können nur niedrige Frequenzen passieren. Dadurch wird Rauschen eliminiert, das Bild aber etwas unschärfer.
- Bandpass-Filter  
Es können nur Frequenzen aus dem Band  $D_0 < \omega < D_1$  passieren.

---

### Idealer Tiefpass-Filter

---

Bei einem idealen Tiefpass-Filter werden alle Frequenzen jenseits einer Grenzfrequenz  $D_0$  abgeschnitten und der „Kegel“ ist radialsymmetrisch zum Ursprung. Der Filter hat dann die Form

$$H(u, v) = \begin{cases} 1 & D \leq D_0 \\ 0 & D > 0 \end{cases}$$
$$D(u, v) = \sqrt{u^2 + v^2}$$

Dieser Filter ist aber so physikalisch nicht realisierbar (dies liegt an der unendlich langen Impulsantwort, z. B. bei einer Rechteck-Funktion)!

---

## Gaußscher Tiefpass-Filter

---

Stattdessen wird ein Gaußscher Tiefpass-Filter eingesetzt. Da die Fourier-Transformation einer Gauß-Glocke wieder eine Gauß-Glocke ist, ist dieser Filter realisierbar.

---

## Idealer Hochpass-Filter

---

Ein idealer Hochpass-Filter schneidet alle Frequenzen unter einer Grenzfrequenz  $D_0$  ab:

$$H(u, v) = \begin{cases} 0 & D \leq D_0 \\ 1 & D > D_0 \end{cases}$$
$$D(u, v) = \sqrt{u^2 + v^2}$$

Ebenso wie der ideale Tiefpass-Filter ist auch dieser Filter physikalisch nicht realisierbar.

---

### 5.2.3 Vergleich: Orts- und Frequenzraum-Filter

---

- Frequenzraum-Filter können schnell berechnet werden (Fast Fourier-Transform), Ortsraumfilter sind meistens aber noch schneller.
- Einfache Handhabung (Das Filterdesign im Frequenzraum ist intuitiv).
- Ortsraumfilter sind nur eine Approximation der Frequenzraum-Filter (es sind keine unendlich breiten Filter möglich) und Abschneiden führt zu Artefakten.

---

## 5.3 Bildkompression

---

Die Rasterung und Abtastung einer Intensitätsfunktion von Licht erzeugt eine große Menge an Daten, was unpraktisch zur Speicherung und Übertragung ist. Es ist somit eine kompaktere Darstellung gewünscht (ohne oder mit zumutbarem Qualitätsverlust).

Bildkompression versucht dabei die Menge an Daten zur Repräsentation zu reduzieren:

- Eliminierung redundanter Daten
- Kodierungen
- Nachbarschaftsbeziehungen (räumlich, zeitlich)
- Psychovisuelle Effekte (Wahrnehmung des Menschen, Farbauflösung des Auges)

Die Kompressionsverfahren werden dabei in zwei Klassen eingeteilt:

- Verlustfreie Kompression, z. B.:
  - Variable-Length-Coding (Huffman Code, Arithmesischer Code)
  - Bit-Plane Coding (Bit-Plane Slicing, Run-Length Coding)
  - Predictive Coding
  - Lempel-Ziv-Welch-Algorithmus (LZW; GIF, TIFF, Kombination von Variable-Length und Run-Length Coding)
- Verlustbehaftete Kompression

- Die Bildinformationen werden so komprimiert, dass nicht alle Eigenschaften berücksichtigt werden und eine exakte Rekonstruktion ggf. nicht mehr möglich ist.
- Viele Verfahren erlauben dem Anwender das Qualitäts-Kompressions-Verhältnis einzustellen (z. B. JPEG oder PNG).
- Häufig werden Modelle der menschlichen Wahrnehmung verwendet (zur Identifizierung von für den Betrachter irrelevanten Bildeigenschaften, die nicht kodiert werden müssen).

Beispiele für Kompressionsverfahren:

- Audio
  - Unkomprimiert: AIFF, WAV, ...
  - Verlustlos: MPEG-4-ALC, Apple Lossless (ALAC), WMA Lossless, ...
  - Verlustbehaftet: MP3, Ogg Vorbis, MPEG-Audio, AAC (iTunes), WMA, ...
- Bilder
  - Unkomprimiert: BMP, RAW, ...
  - Verlustlos: TIFF, GIF, PNG, ...
  - Verlustbehaftet: JPEG, JPEG2000, ...
- Video
  - Unkomprimiert: Nicht praktikabel.
  - Verlustlos: Nicht praktikabel.
  - Verlustbehaftet: H.264 (DivX, QuickTime), MPEG-4 part 2 (Xvid, DivX), WMV, ...

---

### 5.3.1 Harmonische Transformation

---

Bei einer Kompression durch harmonische Transformation werden die Daten in verschiedene Frequenzanteile zerlegt, z. B. durch Fourier-Transformation oder Wavelet-Transformation. Ein typischer Vertreter ist die das JPEG-Kompressionsverfahren.

---

#### JPEG

---

JPEG ist eine Familie von Algorithmen zur Kompression in Echtfarbqualität (dabei gibt es verlustfreie und verlustbehaftete Verfahren). Die verlustbehafteten Prozesse sind für fotografische Aufnahmen mit fließenden Farbübergängen optimiert und daher nicht so gut für Text oder ähnlichen Bilddaten mit harten Kontrasten geeignet.

Durch JPEG sind Kompressionsraten bis zu 1 : 20 bis 1 : 35 erreichbar, wobei diese in den Hauptanwendungsgebieten verlustbehaftet sind. Dabei basiert JPEG auf einer diskreten Kosinustransformation.

#### Schritt 1: Umwandlung in den YCbCr-Farbraum

Im ersten Schritt werden die Farben als

- Y Helligkeitswert
- $C_B$  Abweichung von Grau in Richtung Blau
- $C_R$  Abweichung von Grau in Richtung Rot

kodiert:

$$\begin{bmatrix} Y \\ C_B \\ C_R \end{bmatrix} \approx \begin{bmatrix} 0 \\ 128 \\ 128 \end{bmatrix} + \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.168736 & -0.331264 & 0.5 \\ 0.5 & -0.418688 & -0.081312 \end{bmatrix} \cdot \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

---

## Schritt 2: Farb-Subsampling

Die Farben werden verlustbehaftet komprimiert (dies ist aufgrund der höheren Genauigkeit des Auges im grünen Bereich möglich). Dabei wird für ein kleines Gebiet (üblicherweise  $2 \times 2$  Pixel) die Farbdifferenzwerte  $C_R$  und  $C_B$  gemittelt und für das gesamte Gebiet zusammen angegeben.

## Schritt 3: Diskrete Kosinustransformation

In diesem Schritt werden die Bildinformationen in den Frequenzbereich zerlegt. Dazu wird zunächst jede Komponente ( $Y, C_B, C_R$ ) in  $8 \cdot 8 = 64$  Bildblöcke gerastert und diese anschließend einer diskreten Fourier-Transformation unterzogen. Dabei wird nur der Kosinus-Teil berechnet, da dadurch die Berechnung einfacher wird.

Das Ziel ist, die Informationen in eine Darstellung zu überführen, die besser für die folgenden Schritte geeignet ist.

Vorteil: Wenn sich benachbarte Bildpunkte kaum unterscheiden, d. h. das Bild keine scharfen Kanten hat, sind die meisten Koeffizienten gleich Null.

## Schritt 4: Quantisierung

Bei der Quantisierung werden die Informationsanteile beseitigt, die das Auge nicht oder nur schlecht wahrnimmt.

## Schritt 5: Kodierung der Koeffizienten

Aus den entstehenden Blöcken wird ein sequentieller Bitstrom erzeugt und die Koeffizienten werden als Differenzen zum vorhergehenden Koeffizienten kodiert (durch die Kohärenz ergeben sich hier kleine Werte). Die Koeffizienten werden dabei entlang einer Zick-Zack-Kurve kodiert (ähnliche wie bei Cantors Diagonalargument). Da hohe Frequenzen oft sehr klein sind, entsteht so eine für die Kompression günstige Reihenfolge.

Bisher wurde noch nichts wirklich komprimiert, sondern nur grob transformiert. Der entstehende Bitstrom kann nun aber durch typische Kompressionstechniken (Huffman-Algorithmus, Arithmetisches Kodieren) komprimiert werden.

---

## 6 Bildverarbeitung, Deblurring

---

Beim Deblurring wird versucht, eine vorhandene Verwischung (Blurring) eines Bildes zu entfernen.

Sei bekannt, dass das Bild  $g$  die mit einer Faltung  $a$  verwischte Version (Blurring) eines Bildes  $f$  ist, d. h.  $g = a(f)$  (oftmals ist  $a$  eine Gauß-Glocke). Im Fourier-Raum ergibt sich dann  $G = A \cdot F$  und die Rekonstruktion des Bildes  $f$ , bzw. der Frequenzen  $F$ , scheint mit  $F = A^{-1} \cdot G$  sehr einfach. Bei dieser Rekonstruktion treten jedoch mehrere Probleme auf:

1. Der Blurring-Kernel  $A$  kann unendlich klein werden, sodass es beinahe zu einer Division durch Null kommt. Dadurch werden Rauschen und numerische Fehler verstärkt.
2. Es gibt immer Rauschen  $n$ :  $g = a(f) + n$

Für das erste Problem kann verwendet werden, dass  $A$  i. A. komplex ist. Es gilt dann mit der komplex konjugierten Matrix  $A^*$ :

$$G = A \cdot F \quad \Longrightarrow \quad A^* \cdot A \cdot F = |A|^2 \cdot F$$

Und mit  $|A|^2 > 0$  kann die Rekonstruktion umgeformt werden:

$$F = \frac{1}{A} G = \frac{A^*}{A^* A} G = \frac{A^*}{|A|^2} G$$

Dadurch hat das rekonstruierte Bild nun keine reellen Zahlen mehr.

---

### 6.1 Korrekt gestellte Probleme

---

Nach Jacques Hadamard ist ein mathematisches Modell *korrekt gestellt*, wenn:

- Eine Lösung existiert,
- diese eindeutig ist und
- die Lösung in einer vernünftigen Topologie kontinuierlich von den Daten abhängt.

Ansonsten ist das Problem nicht korrekt gestellt.

Als Konsequenz daraus folgt, dass Blurring korrekt gestellt, Deblurring aber nicht korrekt gestellt ist. Daher ist eine Regularisierung notwendig, d. h. es werden zusätzliche Annahmen (Glätte, Informationen zum Rauschen) hinzugenommen.

---

### 6.2 Einschrittverfahren

---

---

#### 6.2.1 Wiener Filter

---

Das zweite Problem von Deblurring (Rauschen) kann durch eine Regularisierung des Filter im Fourier-Raum:

$$F = \frac{A^*}{|A|^2 + R^2} G$$

reduziert werden (dies wird als *Wieder Filter* bezeichnet). Dabei ist  $R$  das Verhältnis von Rauschen zum Signal. Der Parameter entscheidet dabei, das verstärkt wird.

- Ist  $R$  zu groß, so verhält sich der Filter wie ein Tiefpass-Filter, d. h.:
  - Grobe Struktur bleibt erhalten.
  - Kanten werden verwischt.
  - Rauschen wird entfernt.
- Ist  $R$  zu klein, so verhält sich der Filter wie ein Hochpass-Filter, d. h.:
  - Grobe Strukturen werden entfernt.
  - Kanten werden entfernt.
  - Rauschen wird verstärkt.
- Ist  $R$  optimal, so verhält sich der Filter wie ein Bandpass-Filter, d. h.:
  - Grobe Struktur bleibt erhalten.
  - Kanten werden verstärkt.
  - Rauschen wird entfernt.

**Vorteile:**

- Schnell
- Häufig verwendet
- Beliebt (dadurch viel Know-How vorhanden)
- Leicht zu implementieren

**Nachteile:**

- Nur ein Filter für das gesamte Bild
- Keine lokalen, spezifischen Verbesserungen
- Nur ein Wert für  $R$

Der Wiener-Filter kann z. B. durch lokale Verfeinerungen (Mehrkomponentenverfahren) oder iterative Verfeinerungen (Mehrschrittverfahren) verbessert werden.

## 6.2.2 Mehrkomponentenverfahren

### Scale-Space-Ansatz

Zum Schärfen wird der Laplace-Operator, multipliziert mit einer unabhängigen Konstante  $t$ , vom Bild abgezogen:

$$L_{\text{schärfer}} = L_0 - t \underbrace{(L_{xx} + L_{yy})}_{=\Delta L} = L_0 - t \cdot \Delta L$$

Durch Hinzufügen von zusätzlichen Termen (mit Ableitungen höherer Ordnung) kann das Ergebnis weiter verfeinert werden:

$$L_{\text{schärfer}} = L_0 - t(L_{xx} + L_{yy}) + \frac{1}{2}t^2(L_{xxxx} + 2L_{xxyy} + L_{yyyy}) - \frac{1}{6}t^3(L_{xxxxx} + 3L_{xxxxy} + 3L_{xxyyy} + L_{yyyyy})$$

Diese Sequenz ist eine Taylorreihe der partiellen Differentialgleichung

$$\frac{\partial L}{\partial t} = \frac{\partial^2 L}{\partial x^2} + \frac{\partial^2 L}{\partial y^2}$$

in  $-t$ . Die Veränderung in einem Bild über eine gewisse Zeit ist also durch eine partielle Differentialgleichung zweiter Ordnung definiert.

---

## 6.3 Mehrschrittverfahren (Iterative Methoden)

---

### 6.3.1 Energie und Variationsableitung

---

Sei  $E$  die Energie eines Bildes  $L$ :

$$E(L) = \frac{1}{2} \iint_{x,y} L^2 \, dx \, dy$$

Diese „Energie“ sagt aus, wie viel Intensität in den Pixeln vorhanden ist. Eine Minimierung der Intensität führt dann zum optimalen Bild bzgl. der definierten Energie. Dieses Minimum kann durch Variationsrechnung oder iterative Prozesse gefunden werden.

Die Variationsableitung  $\delta E(L)$  ist eine Verallgemeinerung der normalen Ableitung, wobei dieses Prinzip hier nicht näher beleuchtet werden soll. Für die Energie

$$E(L) = \frac{1}{2} \iint_{x,y} L^2 \, dx \, dy$$

gilt  $\delta E(L) = L$  und das Minimum liegt bei  $\delta E(L) \stackrel{!}{=} 0$ . In diesem Fall liegt das Minimum also bei  $L = 0$ , d. h.  $E(L) = 0$ .

Für die Energie

$$E(L) = \frac{1}{2} \iint_{x,y} L_x^2 + L_y^2 \, dx \, dy$$

folgt  $\delta E(L) = -(L_{xx} + L_{yy}) = -\Delta L$ . Hier ist es weniger trivial das Minimum zu finden, weshalb das System in eine partielle Differentialgleichung

$$L_t = -\delta E(L)$$

überführt wird und das Minimum iterativ gesucht wird.

---

### 6.3.2 Alternativen

---

Da mit den bisherigen Energien keine guten interessanten Bilder generiert werden können, gibt es noch andere Energien, z. B.:

- Perona-Malik
  - Rauschen wird verwischt
  - Kanten werden verstärkt
  - Smart Energy Term und Stoppzeit
- Totale Variation
  - Rauschen wird verwischt
  - Kanten werden verstärkt
  - Smart Energy Term und Distance Penalty
- uvm.



---

### 6.3.3 Perona-Malik

---

Die Heat-Equation  $L_t = L_{xx} + L_{yy} = \Delta L$  wird modifiziert zu

$$\partial_t L = \nabla \circ (c \cdot \nabla L)$$

wobei  $c$  der *Conductivity Coefficient* ist, durch den die Diffusion an lokale Bildstrukturen anpassbar ist (d. h.  $c = c(L, L_x, L_x x, \dots)$ ). Mit  $c = 1$  fällt die Methode zurück zum gaußschen Scale-Space.

---

#### Die Perona-Malik-Gleichung

---

Nach Perona und Malik ist  $c(\cdot)$  eine Funktion der Gradientenstärke, welche die Diffusion dort reduziert, wo Kanten sind ( $c$  nahe Null) und in flachen Bereichen erhöht. Dies eingesetzt ergibt:

$$\partial_t L = \nabla \circ (c(|\nabla L|^2) \cdot \nabla L)$$

Für  $c$  gibt es im Grunde zwei Möglichkeiten, die beide mehr oder weniger das gleiche Verhalten erzeugen:

$$c_1 = \exp \left\{ -\frac{|\nabla L|^2}{k^2} \right\} \quad c_2 = \left( 1 + \frac{|\nabla L|^2}{k^2} \right)^{-1}$$

Dabei bestimmt  $k$  den Einfluss der Kantenstärke. Bei einem großen  $k$  bleiben nur die größten Gradienten (starke Kanten) übrig, bei einem kleinen  $k$  bleiben (fast) alle Gradienten (Kanten, Rauschen) übrig.

---

#### Implementierung

---

Für die Lösung der Perona-Malik-Gleichung gibt es keine analytischen Methoden, weshalb eine iterative Methode eingesetzt wird:

$$L^{(t+1)} = L^{(t)} + \Delta t \cdot \left( \nabla \circ (c(|\nabla L|^2) \cdot \nabla L) \right)$$

dabei ist:

- $L^{(0)}$  das originale Bild und
- $\Delta t$  ein kleiner Zeitschritt.

„Irgendwann“ muss die Iteration beendet werden, z. B. nach  $n$  Schritten oder wenn  $L^{(t+1)}$  „gut aussieht“.

---

#### Stopzeit

---

Während der Iteration steige das Signal-Rausch-Verhältnis i. A. an und fällt danach wieder ab (das Bild konvergiert gegen eine gleichmäßig graue Fläche). Das heißt die Iteration stoppt nicht bei der optimalen Lösung und es wird eine *Stopzeit* benötigt.

---

### 6.3.4 Eingeschränkte Evolution: Totale Variation

---

Wird sichergestellt, dass die Iterationen gegen die optimale Lösung konvergieren, so ist keine benutzerdefinierte Stopzeit nötig. Daher wird versucht, eine klug gewählte Energie zu minimieren und ein *Distance Penalty* hinzuzufügen.

---

## Distance Penalty

---

Bei einem Distance Penalty wird zusätzlich zum Modell des Bildes ein Rauschmodell angenommen (z. B. gaußsches Zufallsrauschen), sodass es zusätzliche Bedingungen an die Lösung  $L$  gibt:

$$\begin{aligned}\iint_{x,y} (g - a(L)) \, dx \, dy &= 0 \\ \iint_{x,y} (g - a(L))^2 \, dx \, dy &= \sigma^2\end{aligned}$$

Diese Beschränkungen können zur Energie hinzugefügt werden, um das Konvergenzverhalten zu verbessern.

---

## Totale Variation

---

Bei der Methode der totalen Variation wird die Energie

$$E(L) = \iint_{x,y} (|\nabla L| + \lambda(g - a(L))^2) \, dx \, dy$$

minimiert, wobei  $\lambda$  ein vom Rauschen abhängiger Parameter ist. Da die totale Variation gegen die optimale Lösung konvergiert, wird keine Stoppzeit benötigt.

---

## Erweiterungen

---

- Schwierigere Funktionen/Energien (Statistik, Niveaumengen für Segmentierung, ...)
- Andere Rauschstatistiken (z. B. Multiplikatives Rauschen)
- Andere Steuerungsmechanismen für den Conductive Coefficient  $c$  (z. B. Kantenverstärkende Diffusion, Kohärenzverstärkende Diffusion, ...)

---

## 7 Grafikpipeline

---

### 7.1 Hardware

---

Abbildung 7.1 zeigt einen Überblick über Computer-Paradigmen. Die Überschneidungen sind dabei Bereiche, in denen mehrere Paradigmen zusammenlaufen.

#### 7.1.1 P1: Large-Scale-Computing

---

Die ursprünglichen Mainframe-Computer waren sehr große Rechenmaschinen, die als Hosts angesteuert wurden. Der Zugriff auf diese fand über externe, mit Tastaturen ausgerüstete, alphanumerische, Terminals statt (Host- und Terminal-System).

#### 7.1.2 P2: Personal/Desktop Computing

---

Computer stehen nun Zuhause und sind sehr viel kleiner.

#### 7.1.3 P3: Networked Computing

---

Computer kommunizieren untereinander.

#### 7.1.4 P4: Mobile Computing

---

Es gibt viele verschiedene Arten von mobilen Geräten (Laptops, Tablets, Smartphones, ...).

#### 7.1.5 ZP1: Collaborative Computing

---

Zum Beispiel Multi-Touch-Tables.

#### 7.1.6 ZP2: Virtual Reality

---

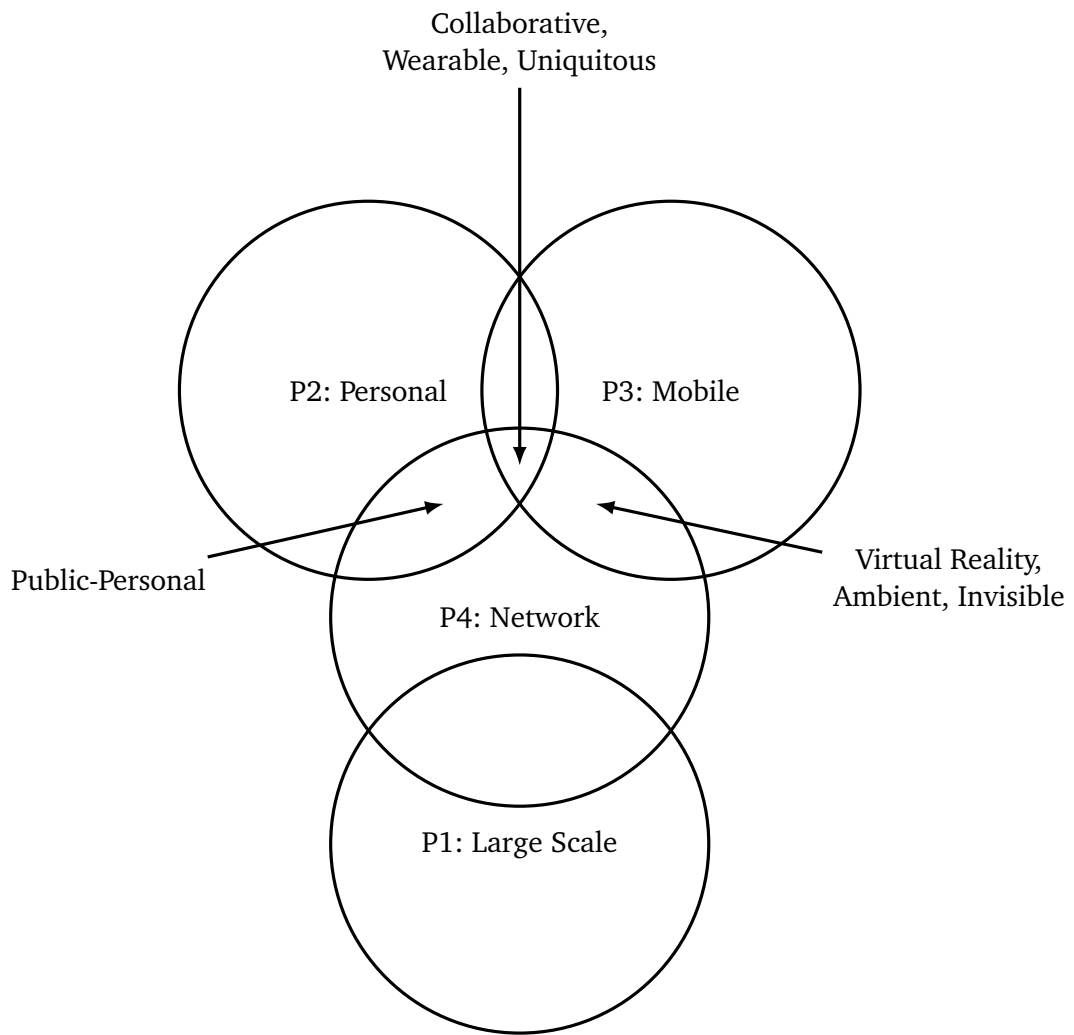
Virtuelle Realität lässt den Menschen in virtuelle Welten eintauchen. Dabei gibt es zwei Arten:

- Nicht-immersive Umgebungen (Bildschirm- und Zeigerbasiert, 3D-Anzeige, mglw. haptisches Feedback).
- Immersive Umgebungen (es wird tatsächlich der Eindruck erweckt, in einer Welt aus virtuellen Objekten zu sein).

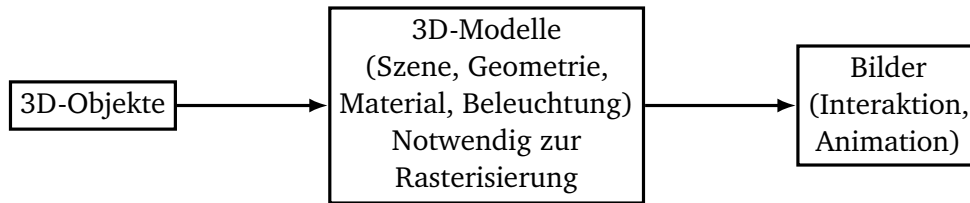
#### 7.1.7 Augmented Reality

---

Augmented Reality beschreibt die Integration von virtuellen und realen Objekten, wobei die Wahrnehmung des Benutzers erweitert und verbessert werden soll. Die Ein- und Ausgabegeräte sind z. B. Heads-Up-Displays (HUDs) oder Head-Mounted-Displays (HMDs).



**Abbildung 7.1:** Hardware und Paradigmen



**Abbildung 7.2:** Einfache Pipeline der Computergrafik.

---

### 7.1.8 Ambient/Invisible

---

Zum Beispiel Freiraum-Gestenerkennung.

---

### 7.1.9 Wearable/Ubiquitous

---

Die Geräte müssen nicht mehr explizit gehalten werden, sondern sie sind „anziehbar“, z. B. eine Smart-watch.

---

## 7.2 Computergrafik

---

Abbildung 7.2 zeigt eine stark vereinfachte Pipeline der Computergrafik.

---

### 7.2.1 Geschichte

---

---

#### Die Anfänge

---

---

60er Jahre

---

---

70er Jahre

---

---

80er Jahre

---

---

90er Jahre

---

---

2000 bis 2005

---

---

2006 bis 2020

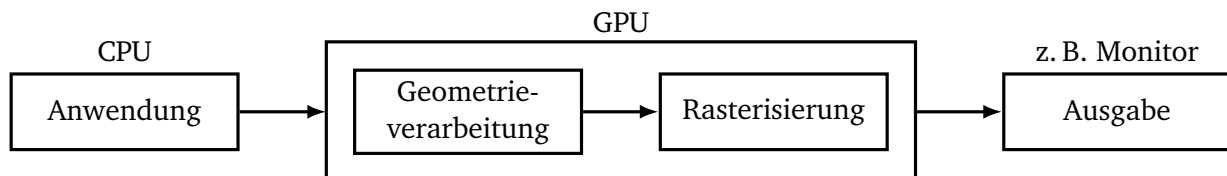
---

---

### 7.2.2 Uncanny Valley

---

Das *Uncanny Valley* beschreibt eine Beziehung zwischen der Ähnlichkeit eines Objekts zum Menschen und der emotionalen Reaktion darauf. Dabei tritt kurz vor einer absoluten Ähnlichkeit ein Tief auf, welches als Uncanny Valleys bezeichnet wird. In diesem Bereich werden Objekte besonders gruselig wahrgenommen (siehe z. B. hier für eine Demonstration dieses Effekts).



**Abbildung 7.3:** Typische Grafikpipeline

---

## 7.3 Grafikpipeline

---

Abbildung 7.3 zeigt die typische Grafikpipeline von der Anwendung zur Ausgabe. Dabei geschehen folgende Schritte:

- Anwendung  
Dieser Schritt produziert eine Modellierung der Daten.
- Geometrieverarbeitung  
Dieser Schritt produziert 2D-Koordinaten und zusätzliche Daten (z-Buffer Werte, Farbwerte pro Knoten/Primitiv).
- Rasterisierung  
Dieser Schritt produziert ein Rasterbild (Bildspeicher).
- Ausgabe  
Speichern/Anzeigen des Bilds.

---

## 7.4 Anwendung

---

Die Anwendung beschäftigt sich mit der Eingabe von grafischen Daten sowie deren Repräsentation.

---

### 7.4.1 Eingabe grafischer Daten

---

Grafische Daten können bspw. durch die Generierung von 3D-Modellen oder der Abtastung realer Objekte eingegeben werden.

---

### 7.4.2 Repräsentation von 3D-Daten

---

---

#### Grafische Primitive

---

Grafische Primitive, aus denen ein größeres Modell zusammengesetzt werden kann, sind z. B.:

- Punkte
- Linien
- Dreiecke

---

#### Transformationen

---

Siehe Kapitel 8.

---

### 7.4.3 Räumliche Datenstrukturen

---

Räumliche Datenstrukturen werden bspw. zum View Frustum Culling (feststellen der Sichtbarkeit eines Objekts im Sichtvolumen), Occlusion Culling/z-Buffer (Verdeckung) oder Kollisionserkennung genutzt. Sie werden in folgende Klassen eingeteilt:

- Hüllkörperhierarchie (bspw. Bounding Sphere Hierarchy)
- Raumunterteilung (Gitter, Hierarchisch (k-d Tree, Quadtree, Octree, Binary Space Partition))

---

#### Hüllkörper/-hierarchien

---

Ein *Hüllkörper* ist eine einfache Form um den eigentlichen Körper herum, sodass sich Schnitttests mit anderen Primitiven einfach durchführen lassen. Typische Primitive sind z. B. Kugeln, Rechtecke oder rotierte Rechtecke. Eine *Hüllkörperhierarchie* wird dann aus mehreren solchen Hüllkörpern zusammengesetzt.

---

#### Raumunterteilung

---

##### **Achsenparallele Gitter (Grids)**

Der Raum wird in ein Gitter eingeteilt, wobei Objekte in mehreren Zellen enthalten sein können (Redundanz). Die Aufteilung kann sich demnach nicht der Geometrie anpassen und ist sehr speicherintensiv. Dennoch ist sie sehr effizient traversierbar und es ist ein schneller Zugriff auf Nachbarn möglich.

##### **Quadtree/Octree**

Bei einem *Quadtree* wird der Raum in ein Gitter aufgeteilt und aus diesem ein Baum berechnet (siehe GdR Zusammenfassung, Abschnitt „Approximative Zellzerlegung“).

Ein *Octree* entspricht dann einem Quadtree im dreidimensionalen.

##### **Binary Space Partition**

Bei einer *Binary Space Partition* (BSP) wird der Raum binär unterteilt, wobei immer an den durch Polygone induzierten Ebenen geteilt wird. Dann entspricht jeder Knoten einer Unterteilungsebene, die den Raum in zwei Halbräume unterteilt.

---

## 7.5 Geometrieverarbeitung

---

---

### 7.5.1 Modelltransformation (Orientierung, Position)

---

---

### 7.5.2 Simulation der Beleuchtung

---

Um die Beleuchtung eines Objekts zu simulieren, müssen zunächst die Leuchtdichten eines Primitivs bestimmt werden (in der Praxis findet die Bestimmung der Leuchtdichte pro Pixel erst während der Rasterisierung statt). Zur Bestimmung der Beleuchtung gibt es unterschiedliche Ansätze:

- Flat Shading
  - Die Normale des Primitivs ergibt eine einheitliche Helligkeit.
- Gouraud Shading
  - Die Normalen in den Eckpunkten ergeben die Helligkeitswerte für die Eckpunkte.

- Die Helligkeitswerte der Eckpunkte werden linear interpoliert.
- Phong Shading
  - Die Normalen in den Eckpunkten werden für jeden Punkt linear interpoliert und normiert.
  - Der Helligkeitswert ergibt sich aus der interpolierten Normalen.

---

### Phong-Beleuchtungsmodell

---

Im Phong-Beleuchtungsmodell setzt sich die Gesamtbeleuchtung

$$I_{total} = I_{amb} + I_{diff} + I_{spec}$$

aus

- Ambienter Reflexion  $I_{amb}$  (global modelliert),
- Diffuser Reflexion  $I_{diff}$  (lokal modelliert) und
- Spiegelnder Reflexion  $I_{spec}$  (lokal modelliert).

zusammen. Die ambiente Komponente ist richtungsunabhängig und abhängig von dem Umgebungslicht  $C$  sowie der Materialkonstanten  $k$ :

$$I_{amb} = k_{amb} C_{amb}$$

Die diffuse Reflexion ist abhängig von der Richtung des Lichts, der Oberflächennormalen  $\mathbf{N}$  und der Richtung zum Licht  $\mathbf{L}$ :

$$I_{diff} = k_{diff} C_{light} (\mathbf{N} \circ \mathbf{L})$$

Die spiegelnde Reflexion ist abhängig von der Richtung der Reflexion  $\mathbf{R}$ , der Betrachtungsrichtung  $\mathbf{V}$  und der „Rauheit“  $m$  des Materials:

$$I_{spec} = k_{spec} C_{light} (\mathbf{R} \circ \mathbf{V})^m$$

---

### 7.5.3 Perspektivische Transformation und Clipping (Abschneiden)

---

Liegen mehrere Objekte voreinander, so ist der dem Auge nächste Punkt sichtbar (es sei denn, das Objekt ist transparent, dann wird auch der dahinterliegende Punkt sichtbar, usw.). *Clipping* bezeichnet nun das Abschneiden von Objekten am Rand des gewünschten Bildausschnitts.

---

#### Painters Algorithmus

---

Bei diesem Algorithmus werden die Primitive „wie von einem Maler“ gezeichnet: Es wird mit dem tiefsten z-Wert begonnen und die Objekte mit aufsteigendem z-Wert darüber gezeichnet. So verdecken sich hintereinander liegende Polygone „automatisch“. Jedoch sind transparente Objekte sowie „im Kreis überdeckende“ Objekte nicht korrekt darstellbar.

---

### 7.5.4 Culling (Verdeckungsrechnung im Objektraum)

---

Üblicherweise machen die Rückseiten von Objekten ca. die Hälfte der vorkommenden Flächen aus, können aber nicht gesehen werden. Durch *Culling* werden die Rückseiten berechnet und beim Rendering explizit ausgeschlossen, um Rechenleistung zu sparen.

Eine Fläche ist immer dann eine Rückseite, wenn das Skalarprodukt von Sehstrahl  $\mathbf{s}$  und Normale  $\mathbf{n}$  positiv ist:  $\mathbf{n} \circ \mathbf{s} > 0$ .



---

## 7.5.5 Projektion

---

Siehe 8.3.

---

## 7.6 Rasterisierung

---

Bei der *Rasterisierung* werden die Primitive (Linien, Polygone) in Pixel zerlegt und zusätzlich pro Pixel eine Verdeckungsrechnung und Shading durchgeführt.

---

### 7.6.1 Scan-Konvertierung

---

---

#### Rasterisierung von Linien (Bresenham-Algorithmus)

---

Der Bresenham-Algorithmus 1 ist ein Algorithmus zum Zeichnen von Linien von Anfangspunkt  $(x_1, y_1)$  mit Endpunkt  $(x_2, y_2)$ . Mit  $\Delta x := x_2 - x_1 \geq 0$  und  $\Delta y := y_2 - y_1 \geq 0$  (die Bedingung  $\geq 0$  kann durch geschickte Vertauschung immer eingehalten werden), muss der Algorithmus genau  $\max\{\Delta x, \Delta y\} + 1$  Pixel zeichnen.

---

**Algorithmus 1:** Bresenham-Algorithmus zum Rastern einer Linie.

---

**Input:** Startpunkt  $(x_1, y_1)$ , Endpunkt  $(x_2, y_2)$

```
1  $\delta x \leftarrow x_2 - x_1$ 
2  $\delta y \leftarrow y_2 - y_1$ 
3  $x \leftarrow x_1$ 
4  $y \leftarrow y_1$ 
5 Setze Pixel  $(x, y)$ 
6  $\xi \leftarrow \delta x / 2$ 
7 while  $x < x_2$  do
8    $x \leftarrow x + 1$ 
9    $\xi \leftarrow \xi - \delta y$ 
10  if  $\xi < 0$  then
11     $y \leftarrow y + 1$ 
12     $\xi \leftarrow \xi + \delta x$ 
13  Setze Pixel  $(x, y)$ ;
```

---

---

#### Rasterisierung von Polygonen (Scanline Algorithmus)

---

Polygone können bspw. mit dem *Scanline Algorithmus* gerastert werden. Dieser scannt die Pixelebene von oben nach unten mit einer „Scan Line“ durch und findet alle Schnittpunkte mit den Kanten des Polygons. Anschließend werden die Schnittpunkt nach  $x$ -Koordinaten sortiert und die Pixel zwischen Paaren aufeinanderfolgender Schnittpunkte gefüllt. Dabei wird eine Parität mitgeführt, die in jedem Schritt um Eins erhöht wird (beginnend von Null). Ist die Parität ungerade, wird der Pixel gesetzt, sonst nicht.

---

### 7.6.2 Verdeckungsrechnung

---

---

## z-Buffer-Algorithmus

---

- Zu jedem Bildpunkt wird noch ein z-Wert gespeichert.
- Initialisierung: Der Bildspeicher wird auf die Hintergrundfarbe gesetzt, der z-Speicher auf den maximalen Wert.
- Anschließend werden alle Objekte der Szene nacheinander gerastert, wobei keine besondere Reihenfolge notwendig ist:  
Für jeden Punkt  $(x, y)$  eines Polygons wird  $z(x, y)$  berechnet. Aufgrund der perspektivischen Transformation ist eine lineare Interpolation nicht mehr einfach möglich! Ist  $z(x, y)$  kleiner als der bereits gespeicherte Wert, so wird  $z(x, y)$  gespeichert und der zugehörige Farbwert in den Bildspeicher geschrieben.
- Nach der Behandlung aller Objekte steht im Objektspeicher das Bild der gewünschten (Teil-) Flächen.

### Vorteile:

- Jede Szene mit jeder Art von Objekten kann behandelt werden.
- Die Komplexität ist unabhängig von der Tiefenkomplexität.
- In eine fertige Szene können nachträglich Objekte eingefügt werden.
- Spezielle Objekte (z. B. ein 3D-Cursor) können in der Szene mit korrekter Verdeckung dargestellt werden.
- Leicht in Hardware zu realisieren.

### Nachteile:

- Für jeden Bildpunkt wird nur ein Objekt gespeichert (dies führt zu Abtastfehlern).
- Transparenz ist prinzipiell nicht realisierbar.
- Die Genauigkeit des z-Buffers ist beschränkt (getrennte Objekte erhalten den selben z-Wert, die Farbe wird dann von der Objektreihenfolge bei der Rasterung bestimmt).

---

## 8 Transformationen

---

Aus Sicht der Grafikkipeline gibt es viele unterschiedliche Koordinaten:

- Objektkoordinaten (Festlegung der lokalen Lage von 3D-Objekten)
- Weltkoordinaten (Beschreibung der gesamten Szene in 3D)
- Projektionskoordinaten (nach der Anwendung der Projektionstransformation)
- Bildschirmkoordinaten (Darstellung der Szene in einem Fenster mit gewählter Position und Größe)

Abbildung 8.1 zeigt die Umwandlung der Koordinaten innerhalb der Grafikkipeline.

---

### 8.1 Affine Abbildungen

---

---

#### 8.1.1 Eigenschaften

---

*Affine Abbildungen* (Translation, Rotation, Skalierung, Scherung) haben folgende Eigenschaften:

- Geraden werden auf Gerade abgebildet.
- Beschränkte Objekte bleiben beschränkt.
- Verhältnisse von Längen, Flächen, Volumen bleiben erhalten.
- Parallele Objekte bleiben parallel.

Mathematischer formuliert: Eine Abbildung  $\Phi : \mathbb{R}^n \rightarrow \mathbb{R}^n$  heißt *affin*, gdw.  $\Phi$  in der Form

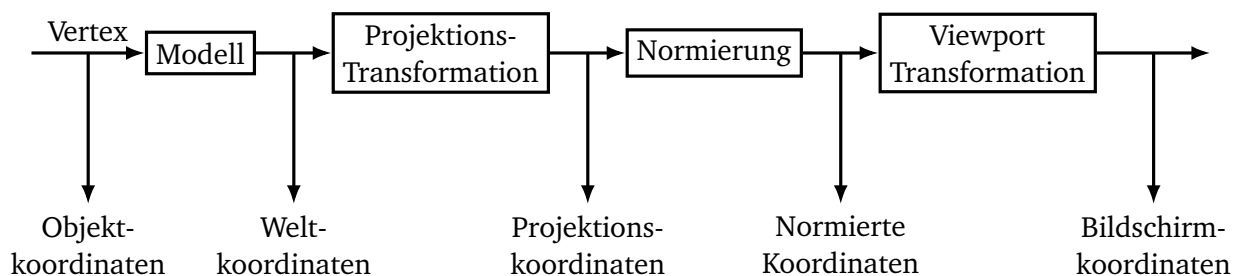
$$\Phi(\mathbf{v}) = A(\mathbf{v}) + I(\mathbf{b})$$

mit einer linearen Abbildung  $A : \mathbb{R}^n \rightarrow \mathbb{R}^n$ , der Identitätsabbildung  $I : \mathbb{R}^n \rightarrow \mathbb{R}^n$  und  $\mathbf{v}, \mathbf{b} \in \mathbb{R}^n$  darstellbar ist. Eine affine Abbildung setzt sich also aus einer linearen Abbildung und einer Translation zusammen.

Eine Abbildung  $A : \mathbb{R}^n \rightarrow \mathbb{R}^n$  heißt *linear*, gdw.

$$A(\lambda \mathbf{u} + \mu \mathbf{v}) = \lambda A(\mathbf{u}) + \mu A(\mathbf{v})$$

für alle  $\mathbf{u}, \mathbf{v} \in \mathbb{R}^n$  und  $\lambda, \mu \in \mathbb{R}$  gilt.



**Abbildung 8.1:** Änderung der Koordinaten innerhalb der Grafikkipeline.

---

## 8.1.2 Homogene Koordinaten

---

Anstelle der aufwendigen Notation  $Av + \mathbf{b}$  für affine Abbildungen können homogene Koordinaten verwendet werden (sei dazu im folgenden  $n = 3$ ). Dann wird eine Äquivalenzklasse nach  $\mathbb{R}^4$  definiert durch:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} \rightarrow \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \quad \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} \rightarrow \begin{bmatrix} x/w \\ y/w \\ z/w \\ 1 \end{bmatrix}$$

Wobei  $w \neq 0$  einen Skalierungsfaktor darstellt (dieser ist meistens 1).

Eine reine Translation kann dann in homogenen Koordinaten ausgedrückt werden als:

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & x_0 \\ 0 & 1 & 0 & y_0 \\ 0 & 0 & 0 & z_0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x + x_0 \\ y + y_0 \\ z + z_0 \\ 1 \end{bmatrix}$$

Allgemein ergibt sich für homogene Koordinaten die Transformationsmatrix

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & x_0 \\ a_{21} & a_{22} & a_{23} & y_0 \\ a_{31} & a_{32} & a_{33} & z_0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

was einer Anwendung der Matrix  $A$  und eine Translation um den Vektor  $[x_0 \ y_0 \ z_0]^T$  entspricht (dies ist leicht nachzurechnen).

Diese einheitliche und kompakte Darstellung von Rotationen/Skalierungen/... und Translationen als eine Matrixmultiplikation erlaubt eine einfache Implementierung und eine Hintereinanderausführung mehrerer Operationen entspricht einer reinen Multiplikation von Matrizen.

---

## 8.2 Skalierung, Scherung, Rotation

---

Die affinen Abbildung der Skalierung, Scherung und Rotation lassen den Ursprung invariant, d. h. der Vektor  $[0 \ 0 \ 0]^T$  wird nicht verschoben. Hierfür während theoretisch normale  $(3 \times 3)$ -Matrizen ausreichend.

---

### 8.2.1 Skalierung

---

Eine Skalierung wird durch eine Diagonalmatrix

$$\begin{bmatrix} s_1 & 0 & 0 \\ 0 & s_2 & 0 \\ 0 & 0 & s_3 \end{bmatrix} \rightarrow \underbrace{\begin{bmatrix} s_1 & 0 & 0 & 0 \\ 0 & s_2 & 0 & 0 \\ 0 & 0 & s_3 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{\text{Homogene Koordinaten}}$$

beschrieben. Dabei wird die  $x$ -Achse um  $s_1$  skaliert, die  $y$ -Achse um  $s_2$  und die  $z$ -Achse um  $s_3$ . Gilt  $s_1 = s_2 = s_3$ , so werden alle Koordinaten gleichermaßen skaliert.

---

## 8.2.2 Scherung

---

Eine Scherung wird durch eine Matrix

$$\begin{bmatrix} 1 & s_2 & s_5 \\ s_1 & 1 & s_6 \\ s_3 & s_4 & 1 \end{bmatrix} \rightarrow \underbrace{\begin{bmatrix} 1 & s_2 & s_5 & 0 \\ s_1 & 1 & s_6 & 0 \\ s_3 & s_4 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{\text{Homogene Koordinaten}}$$

beschrieben.

---

## 8.2.3 Rotation

---

Eine Basisrotation (um eine der Koordinatenachsen) wird durch folgende drei Matrizen beschrieben (von oben nach unten jeweils um die  $x$ -,  $y$ - und  $z$ -Achse):

$$\begin{aligned} \mathbf{R}(x; \alpha) &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\alpha) & -\sin(\alpha) \\ 0 & \sin(\alpha) & \cos(\alpha) \end{bmatrix} \\ \mathbf{R}(y; \alpha) &= \begin{bmatrix} \cos(\alpha) & 0 & \sin(\alpha) \\ 0 & 1 & 0 \\ -\sin(\alpha) & 0 & \cos(\alpha) \end{bmatrix} \\ \mathbf{R}(z; \alpha) &= \begin{bmatrix} \cos(\alpha) & -\sin(\alpha) & 0 \\ \sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 1 \end{bmatrix} \end{aligned}$$

Der Kompaktheit halber wurden hier die homogenen Formulierungen weg gelassen, sie werden aber analog zu denen der Skalierung und Scherung gebildet.

---

### Rotation um beliebige Achse

---

Zur Rotation um einen beliebigen, normierten Vektor  $\mathbf{r}$  um den Winkel  $\alpha$  muss zunächst das körperfeste Koordinatensystem des zu rotierenden Körpers koinzident zu den Ursprungsachsen gedreht werden (durch eine Rotationsmatrix  $\mathbf{R}$ ) und anschließend um eine der Achsen (z. B. der  $x$ -Achse) rotiert werden und anschließend wieder zurück in das körperfeste Koordinatensystem gedreht werden:

$$\mathbf{R}_r(\alpha) = \mathbf{R}^{-1} \cdot \mathbf{R}(x; \alpha) \cdot \mathbf{R}$$

Dazu muss zunächst eine orthonormale Basis  $(\mathbf{r}, \mathbf{s}, \mathbf{t})$  bestimmt werden. Der erste Basisvektor ist die Drehachse  $\mathbf{r}$ , die anderen beiden Vektoren werden wie folgt berechnet:

$$\mathbf{s} = \begin{cases} \frac{\mathbf{r} \times \mathbf{e}_y}{\|\mathbf{r} \times \mathbf{e}_y\|} & \text{falls } r \parallel \mathbf{e}_x \\ \frac{\mathbf{r} \times \mathbf{e}_x}{\|\mathbf{r} \times \mathbf{e}_x\|} & \text{sonst} \end{cases}$$

$$\mathbf{t} = \mathbf{r} \times \mathbf{s}$$

Daraus ergibt sich mit  $\mathbf{R} = [\mathbf{r} \ \mathbf{s} \ \mathbf{t}]$  die benötigte Transformationsmatrix.

---

## Rotation um beliebigen Punkt

---

Soll ein Objekt um einen anderen Punkt als den Ursprung gedreht werden, so muss das Rotationszentrum zunächst in den Ursprung verschoben, dann die Rotation durchgeführt und anschließend das Rotationszentrum wieder zurückgeschoben werden.

**Beispiel:** Es soll eine Transformationsmatrix erstellt werden, die eine Rotation um die  $i$ -Achse ( $i \in \{x, y, z\}$ ) im Punkt beschrieben durch  $\mathbf{r}$  durchführt. Die Transformationsmatrix lautet dann allgemein:

$$T = \text{Trans}(\mathbf{r}) \cdot \text{Rot}(i; \alpha) \cdot \text{Trans}(-\mathbf{r})$$

---

### 8.2.4 Nicht-Kommutativität von Transformationen

---

Die Reihenfolge von Transformationen darf i. A. nicht vertauscht werden (insbesondere ist die Matrixmultiplikation nicht kommutativ).

---

### 8.2.5 Rechenaufwand

---

Bei vielen nacheinander ausgeführten Transformationen ist es sinnvoller, einmal die gesamte Transformationsmatrix zu berechnen, statt oftmals Matrix-Vektor-Multiplikationen durchzuführen.

---

## 8.3 Projektion

---

Zur Projektion von 3D-Elementen gibt es viele unterschiedliche Möglichkeiten:

- Aufriss (Frontansicht)
- Kabinett-/Kavallierperspektive
- Allgemeine Parallelprojektion
- Isometrische Perspektive
- Zentralperspektive
- Vogelperspektive

Solche projektiven Abbildungen können durch homogene Transformationen beschrieben werden, wobei jedoch die Winkel verändert werden! Außerdem geht die Parallelität von Linien oft verloren (d. h. Parallelen schneiden sich in Fluchtpunkten). Dies ist der allgemeine Unterschied zwischen perspektivischer und paralleler Projektion:

- Bei einer perspektivischen Projektion treffen sich die Strahlen im Augenpunkt (Projektionszentrum) und Winkel werden verändert.
- Bei der parallelen Projektionen sind die Projektionsstrahlen parallel und die Winkel bleiben erhalten.

---

### 8.3.1 Perspektive Projektion

---

- Vergleichbar mit einem fotografischen System.
- Entspricht der natürlichen Wahrnehmung.
- Der Abstand zwischen Objekten und Projektionsebene geht ein.
- Längenverhältnisse ändern sich.
- Winkel ändern sich.
- Parallele Geraden bleiben nicht parallel.

Das heißt perspektivische Projektionen sind im allgemeinen keine affinen Abbildungen! Insbesondere werden vom Blickpunkt weiter entfernte Objekte kleiner dargestellt.

Ist der zu projizierende Punkt  $P = (x, y)$  und der Augpunkt  $A = (-x_0, 0)$  gegeben, so gilt für den Bildpunkt  $B = (0, y_0)$ :

$$\frac{y_0}{y} = \frac{x_0}{x + x_0}$$

Allgemein lautet mit  $y_0 = yx_0/(x + x_0)$  die Abbildung wie folgt:

$$\begin{bmatrix} x \\ y \end{bmatrix} \mapsto \begin{bmatrix} 0 \\ \frac{yx_0}{x_0+x} \end{bmatrix}$$

Daraus ergibt sich die homogene ( $3 \times 3$ )-Matrix:

$$\begin{bmatrix} 0 \\ \frac{yx_0}{x_0+x} \\ 1 \end{bmatrix} \approx \begin{bmatrix} 0 \\ yx_0 \\ x_0 + x \end{bmatrix}$$

Und somit die perspektivische Transformation:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & x_0 & 0 \\ 1 & 0 & x_0 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \approx \frac{1}{x_0} \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 1/x_0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

---

### Allgemeine perspektivische Transformation

---

Mit dem Fluchtpunkt in  $(x_0, y_0, z_0)$  wird eine allgemeine perspektivische Transformation beschrieben durch:

$$\begin{bmatrix} x' \\ y' \\ z' \\ w' \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1/x_0 & 1/y_0 & 1/z_0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}$$

---

### 8.3.2 Parallele Projektion

---

- Ist weniger realistisch.
- Winkel ändern sich i. A. nicht.
- Parallele Geraden bleiben parallel.

---

### 8.3.3 Kanonisches Sichtvolumen

---

Die perspektivische Projektion wird in zwei Abbildungen zerlegt:

- Die perspektivische Transformation und
- eine anschließende Parallelprojektion.

Nach der perspektivischen Transformation ist das Sichtvolumen ein Würfel und durch eine Rotation in den Augpunkt wird erreicht, dass der Würfel achsenparallel wird:

$$\underbrace{\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 1/x_0 & 0 & 1 \end{bmatrix}}_{\text{Perspektivische Projektion}} = \underbrace{\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{\text{Parallelprojektion}} \cdot \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1/x_0 & 0 & 1 \end{bmatrix}}_{\text{Perspektivische Transformation}}$$

Ist die Kamera im Unendlichen (Parallelprojektion), wird das Sichtvolumen ein Einheitswürfel. Bei einer perspektivischen Projektion ist das Sichtvolumen eine Pyramide, nach der perspektivischen Transformation hingegen wieder ein Würfel!

---

## 8.4 3D-Interaktion

---

Bei der 3D-Interaktion ist nicht immer klar, welche Bewegung der Nutzer ausgeführt haben möchte. Ansätze hierzu sind

- Desktop,
- Multi-Window (Mehrfachauswahl),
- Direktes 2D-Maus-Mapping oder
- Manipulatoren.

---

### 8.4.1 Manipulatoren

---

Im zweidimensionalen werden Manipulatoren häufig eingesetzt (Manipulatoren sind dabei zum Beispiel kleine „Rädchen“ um ein Objekt, mit welchen dieses gedreht werden kann). Auch werden z. B. Kästchen eingesetzt, mit denen skaliert und verschoben werden kann (Drag-and-Drop). Dabei sind sie im zweidimensionalen sehr einfach zu implementieren, da eine Eins-zu-Eins Abbildung der Mauszeigerposition zum Manipulator erstellt werden kann. Außerdem gibt es keine Probleme bei der perspektivischen Abbildung.

Immer häufiger werden Manipulatoren auch im dreidimensionalen, z. B. zur Navigation der Kamera eingesetzt. Hier ist die Implementierung jedoch sehr viel schwieriger, da es unendliche viele Möglichkeiten gibt, eine Cursorposition auf einer geraden Linie im 3D-Raum abzubilden. Dennoch stellen Manipulatoren eines der besten momentan verfügbaren Werkzeuge zur 3D-Interaktion dar.



---

## 9 3D-Visualisierung

---

### 9.1 (Gewinnung) 3D-Daten

---

Bei 3D-Daten werden die Messwerte dreidimensional im Raum verteilt und jeder Wert hat drei Koordinaten  $(x, y, z)$ . Dabei können die Werte gleichmäßig oder unterschiedlich verteilt sein und ein Wert kann skalar oder höherdimensional sein (z. B. ein Vektor in einem Strömungsfeld).

#### Terrain

Zum Scannen eines Terrains wird an beliebigen Positionen  $(x, y)$  die Höhe  $z$  gemessen, wodurch sich eine 3D-Position ergibt. Oberflächeninformationen (z. B. Vegetation) können durch Satellitenbilder gewonnen werden.

#### Laser Scanning

Beim Laser Scanning wird ein Laserstrahl auf eine Oberfläche projiziert und das rückstrahlende Licht gemessen. Aus dem Abstand zwischen Laser und Kamera kann dann die Distanz durch Triangulation berechnet werden. Dies ergibt eine unstrukturierte Punktwolke.

#### Range Images

Aus einem Range Image  $r(u, v)$  ergibt sich die Pixelinformation als 3D-Punkt  $(u, v, r(u, v))$ .

#### Medizinische Bilddaten

In der Medizin werden viele bildgebende Geräte verwendet, um physikalische Eigenschaften zu messen (MRI, CT, Ultraschall, ...). Diese produzieren einen „Stapel“ von parallelen Scheiben (Slices), die jeweils einem regulären 2D-Gitter entsprechen. Ein solcher Scan produziert jedoch riesige Datenmengen!

#### Wetter

Die Parameter des Wetters (z. B. Temperatur, Druck, Niederschlag, Windrichtung, ...) sind für bestimmte Regionen auf verschiedenen Höhen unterschiedlich und sowohl vektoriell als auch skalar. Zur Messung dieser Daten wird die Erde in viele Zellen einer bestimmten Größe (meist einige Kilometer) aufgeteilt.

---

### 9.2 Triangulation von Punktwolken

---

Eine unstrukturierten Punktmenge  $s_i = (x_i, y_i, z_i)$  auf einer Oberflächen  $S$  wird als *Punktwolke* bezeichnet. Für einfache Oberflächen (ohne Falten) können Punkte auf eine Ebene projiziert und in 2D trianguliert werden (planare Triangulation). Dieses 2D-Netz wird dann entsprechend der  $z_i$ -Werte deformiert.

---

#### 9.2.1 Ideal Triangulation

---

Bei einer idealen Triangulation haben alle Dreiecke die Innenwinkel (60 deg, 60 deg, 60 deg), d. h. die Dreiecke sind gleichschenkelig. Dies führt zu einer numerischen Stabilität und vereinfacht das Post-Processing.

---

## 9.2.2 Voronoi-Diagramm

---

Statt einer idealen Triangulation, die schwer zu berechnen ist, kann ein Voronoi-Diagramm eingesetzt werden. Dabei wird für jeden (in 2D projizierten) Punkt  $\bar{s}_i$  eine *Voronoi-Zelle* definiert, die alle Punkte enthält, die näher an  $\bar{s}_i$  als an allen anderen Orten (andere  $\bar{s}_j$ ) liegt. Die Kante einer Voronoi-Zelle liegt dann auf den Punkten mit dem gleichen Abstand zu den zwei nächsten Orten und der Knoten einer Voronoi-Zelle auf einem Punkt, der den gleichen Abstand zu drei anderen Orten hat. Dadurch wird die 2D-Fläche „parkettiert“.

---

## 9.2.3 Delaunay-Triangulation

---

Durch Betrachtung des dualen Graph zu einem Voronoi-Diagramm wird eine *Delaunay-Triangulation* beschrieben (es sind jedoch mglw. Korrekturen nötig). Dabei ist ein Dreiecksnetz nur dann eine Delaunay-Triangulation, wenn alle Umkreise von allen Dreiecken leer sind, d. h. es liegt kein Ort in ihnen. Dies kann durch das „Umdrehen“ einer Kante korrigiert werden.

---

## 9.3 Indirekte Volumenvisualisierung

---

Eine Menge von Volumendaten enthält viele Informationen, was das Rendering verlangsamt, obwohl vieles (z. B. verdeckte) Elemente gar nicht angezeigt wird. Deshalb wird i. A. nicht das gesamte Volumen, sondern nur ein Teil angezeigt.

---

### 9.3.1 3D-Volumen und Nachbarschaft

---

In einem normalen 3D-Raster wird ein Volumenelement (Würfel) als *Voxel* bezeichnet. Die Dicke eines Slices (d. h. die Breite eines Würfels) ist dabei oftmals größer als die Pixelabstände (anisotropische Volumen). Die Eine Rasterposition wird dabei durch einen Index  $(i, j, k)$  beschrieben.

Ein Voxel ist dabei adjazent zu einem Referenzvoxel:

- Im zweidimensionalen sind dies über 4 Kanten und 4 Ecken insgesamt 8 Nachbarvoxel.
- Im dreidimensional sind dies über 6 Flächen, 12 Kanten und 8 Ecken insgesamt 24 Nachbarvoxel.

---

### 9.3.2 2D: Konturlinien

---

Eine *Konturlinie* ist eine Linie entlang der selben Höhe, d. h. der Wert ist entlang einer Konturlinie konstant (d. h. ein Kontur-Diagramm entspricht einer Höhenkarte). Dabei ist die Ausrichtung des Gefälles orthogonal zu den Konturlinien.

---

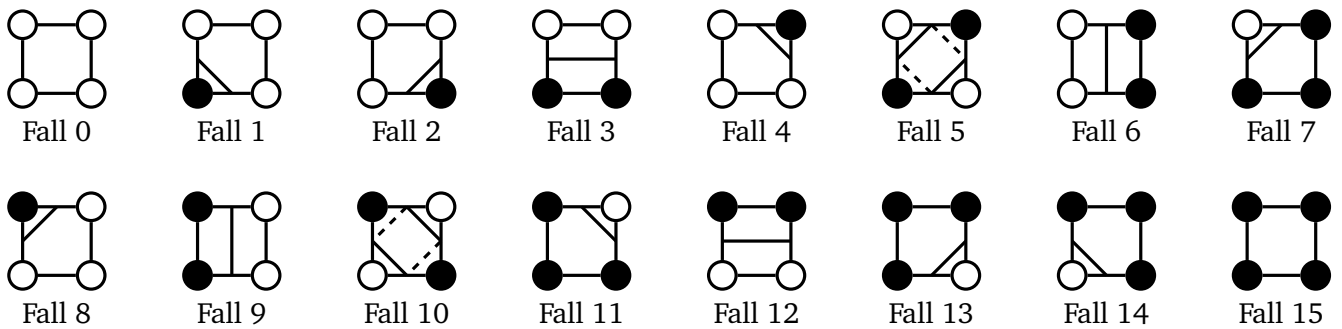
### 9.3.3 3D: Isoflächen

---

Eine Trennung zwischen verschiedenen Strukturen führt zu einer Eingrenzung von Strukturen, wodurch ebendiese Strukturen erkannt werden könne. Die Voxel an einer solchen Eingrenzung bilden, wenn die die gleiche Intensität haben, sogenannte *Isoflächen*. Eine Isofläche ist dabei eine implizite Fläche

$$i(x) = V(x) - \tau = 0$$

wobei  $V(x)$  der Voxelwert und  $\tau = \text{const}$  ein festgesetzter Isowert ist. Die Datenmenge wird dann aufgeteilt in innen ( $i(x) > 0$ ) und außen ( $i(x) < 0$ ) liegende Flächen. Die Definition eines solchen Isowerts entspricht also einem Thresholding der Daten.



**Abbildung 9.1:** Möglichkeiten zur Isolinien-Legung in den Bildzellen beim Marching Squares Algorithmus. Dabei bedeutet ein schwarzer Pixel, dass der Pixel im Zustand „Set Pixel“ ist, weiß bedeutet dementsprechend „Reset Pixel“. In den Fällen 5 und 11 ist die Legung der Linie nicht eindeutig und es kann wahlweise die gestrichelte Version oder die durchgezogene Version gewählt werden.

### 9.3.4 2D: Marching Squares

Der *Marching Squares Algorithmus* versucht, gegeben einem Fixen Isowert  $\tau$  die Isolinie  $s(x) = 0$  zu finden, um so die Fläche in innen und außen zu unterteilen. Dabei sei jede Bildzelle durch ihre vier umgebenden Pixel definiert. Danach wird jede Bildzelle abgelaufen, um sie einem der in Abbildung 9.1 aufgezeigten Fälle zuzuordnen und die Isolinie entsprechend zu ziehen. Ein Pixel hat dabei den „Set Pixel“-Zustand, wenn dieser größer oder gleich  $\tau$  ist.

Werden Rotationen und Symmetrien beachtet, so reduzieren sich die Fälle auf fünf Fälle.

### 9.3.5 3D: Marching Cubes

Der *Marching Cubes Algorithmus* erweitert die Idee von Marching Squares auf drei Dimensionen, wobei eine Volumenzelle durch ihre acht umgebenden Voxel definiert ist. Es gibt, ohne Beachtung von Symmetrien, 256 verschiedene Kombination von „Set Pixel“ und „Reset Pixel“ Zuständen. Unter Einbeziehung von Symmetrien werden diese auf 15 Klassen reduziert.

### 9.3.6 Große Polygonmodelle und Performanz

Der Marching Cubes Algorithmus erzeugt sehr viele Millionen Dreiecke, was zu einem hohen Berechnungsaufwand führt. Daher muss das entstehende Mesh vor dem Rendern noch reduziert werden.

#### Culling von Geometrie

Eine Möglichkeit stellt das Culling von Geometrie dar, bei dem unsichtbare Polygone aus der Rendering-Pipeline entfernt werden:

- Backface-Culling: Rückseiten werden nicht gezeichnet.
- View-Frustum-Culling: Polygone, die sich ganz oder teilweise außerhalb des View-Frustums befinden, werden nicht (oder nur teilweise) gezeichnet.
- Occlusion-Culling: Polygone werden nach der tiefe sortiert und nur gerendert, wenn sie nicht vollständig verdeckt sind (Transparenz muss beachtet werden!).
- uvm.

---

## Meshreduktion

---

Bei der Meshreduktion wird die Anzahl der Polygone verringert, wobei die „Größe“ der Vereinfachung stark vom Szenario abhängt (Genauigkeit vs. Zeitbegrenzung).

---

## Mesh-Glättung

---

Das Ziel der Mesh-Glättung ist die Bereitstellung einer guten Visualisierung sowie der Artefakt-Reduzierung und Entfernung von „Löchern“. Die Herausforderung ist hierbei, dass Volumen zu erhalten. Laplacian Glättung:

- Es wird eine „Regenschirm“-Region betrachtet (d. h. ausgehend von einem Vertex alle durch eine Kante verbundenen Vertexe sowie deren Verbindungen).
- Anschließend werden hochfrequente Oberflächeninformationen reduziert.
- Dies sorgt für eine Reduktion von Krümmungen.

---

## 9.4 Direkte Volumenvisualisierung

---

Im Gegensatz zur indirekten Volumenvisualisierung, bei der zunächst eine Zwischendarstellung generiert wird und die Komplexität von der Anzahl an Polygonen abhängig ist, wird bei der direkten Volumenvisualisierung die Voxel direkt ohne Zwischendarstellung visualisiert. Dabei ist die Komplexität von der Anzahl der Voxel und der Auflösung der Anzeigefläche abhängig.

---

### 9.4.1 Density Emitter Model

---

- Es werden nur Emission und Absorption betrachtet.
- Jeder Voxel in der Datenmenge ist eine kleine Lichtquelle.
- Das Licht wird schwächer, wenn es durch die Volumendatenmenge wandert.
- Das Medium ist eine homogene Dichtewolke.

Dadurch wird die Bestrahlungsstärke  $I(s)$  eines Voxels  $s$  beschrieben durch:

$$I(s) = I_{s_0} \cdot \exp \left\{ - \int_{s_0}^s \tau(t) dt \right\} + \int_{s_0}^s Q(\tilde{s}) \cdot \exp \left\{ - \int_{\tilde{s}}^s \tau(t) dt \right\} d\tilde{s} \quad (9.1)$$

wobei  $I_{s_0}$  die Beleuchtungsstärke des Hintergrunds und  $Q(\tilde{s})$  die aktive Emission des Voxels  $\tilde{s}$  ist. Mit  $t_i := \exp \left\{ - \tau(i \cdot \Delta t) \cdot \Delta t \right\}$  kann die Volumen-Rendering-Gleichung 9.1 diskretisiert werden:

$$I(s) = I_0 \prod_{k=0}^{n-1} t_k + \sum_{k=0}^{n-1} \left( Q(k \cdot \Delta s) \cdot \Delta s \prod_{j=k+1}^{n-1} t_j \right) \quad (9.2)$$

---

### 9.4.2 Volumen-Rendering-Pipeline

---

In der Volumen-Rendering-Pipeline gibt es drei grundlegende Schritte:

1. Abtastung (Sampling)
2. Klassifizierung und Beleuchtung
3. Komposition

Die nacheinander durchlaufen werden

---

## Abtastung

Es werden Voxelwerte an bestimmten Orten angesammelt, wobei die Position dieser Orte durch die Abtastdistanz  $\Delta s$  festgelegt ist (diese sollte kleiner als die Hälfte der Rasterauflösung sein, Shannons Abtasttheorem). Dabei befinden sich die Abtastpositionen meistens zwischen den Rasterpositionen.

Anschließend werden die Werte interpoliert, bspw. mit Nearest Neighbor, Trilinear oder B-Spline Modellen.

## Klassifikation und Beleuchtung

Für jeden Abtastpunkt wird nun der Anteil (Farbwert)  $Q_k = Q(k \cdot \Delta s)$  sowie der Abschwächungsfaktor (Transparenz)  $t_i$  bei jedem Abtastpunkt berechnet. Dadurch wird der beleuchtete Anteil berechnet, wobei Volumenabtastungen als gerichtete Lichtquellen betrachtet werden (Shading).

## Komposition

Die abgetasteten, klassifizierten und beleuchteten Objekte werden nun zusammengeführt (akkumuliert), wobei die Volumen-Rendering-Gleichung numerisch approximiert wird (Gleichung 9.2). Diese Akkumulation wird in zwei Unterschritte geteilt:

1. Back-to-Front-Komposition
2. Front-to-Back-Komposition

**Back-to-Front-Komposition:** Bei der Back-to-Front-Komposition wird mit der Abtastposition am Ende des Volumens mit der Zusammensetzung begonnen und in die Richtung des Sichtpunktes fortgefahren. Dabei wird iterativ die Farbe  $C_k$  sowie die Intensität  $T_k$  an der aktuellen Abtastposition  $k$  berechnet:

$$C_k = Q(k \cdot \Delta s) \cdot \Delta s$$
$$I_k = I_{k-1} \cdot t_k + C_k$$

**Front-to-Back-Komposition:** Es wird mit der Abtastposition am Anfang des Volumens begonnen und in Richtung des Endes fortgefahren. Dabei werden iterativ Intensität  $I_{k-1}$ , Transparenz  $\tau_{k-1}$  berechnet:

$$I_{n-1} = C_{n-1}$$
$$\tau_{n-1} = t_{n-1}$$
$$I_{k-1} = T_k + C_k \cdot \tau_k$$
$$\tau_{k-1} = t_k \cdot \tau_k$$

Es müssen immer zwei akkumulierte Werte (Intensität und Transparenz) berechnet werden, um z. B. Nebel darzustellen (bei diesem sind Lichtquellen ab einem bestimmten Punkt irrelevant).

Die Zusammensetzung wird gestoppt, sobald die akkumulierte Transparenz zu klein wird (Early Ray Termination). Dadurch müssen nicht alle Positionen entlang des Strahls betrachtet werden und die Rendering-Geschwindigkeit wird erhöht. Üblicherweise ist dies durch ein Transparenz-Threshold definiert.

---

## Transferfunktion

---

Die *Transferfunktion* bildet gemessene und abgetastete Werte auf optische Eigenschaften ab (der Farbwert  $Q$  durch eine Farbtransferfunktion und die Transparenz  $t$  durch eine Opazitätstransferfunktion). Dabei bilden die Voxelwerte den Definitionsbereich und die optischen Eigenschaften den Bildbereich einer eindimensionalen Transferfunktion:

$$tf_i : V \rightarrow O_i$$

---

Bei Grauwertabbildungen werden nur Grauwerte betrachtet und die Intensität wird gemäß den Abtastwerten zugewiesen (niedrigster Wert schwarz, höchster Wert weiß, die Werte dazwischen werden linear interpoliert). Dunklen Werten wird dabei eine geringe Opazität (Transparenz) zugewiesen.

Spezifikation:

- Es kann z. B. ein Histogramm verwendet werden.
- Die Referenzeinstellungen für Opazität und Farbe werden für eine begrenzte Anzahl von Werten im Definitionsbereich spezifiziert.
- Zwischen den Werten wird dann linear interpoliert.
- Oftmals werden Höchstwerte gewählt, um die Opazität zu erhöhen und die Farbe zu ändern.
- Meistens sind fotorealistische Renderings das Ziel.
- Den verschiedenen Strukturen werden demnach „echte“ Farben zugewiesen.
- Problem: Zwei Strukturen mit dem gleichen Grauwert haben mglw. verschiedene echte Farben.

---

## 10 Szenengraphen am Beispiel X3DOM

---

Zur Beschreibung einer 3D-Szene werden viele Informationen benötigt:

- Objekt-Geometrie (Säule, Ball, ...)
- Transformationen (Positionierung einer Säule, Rotation, ...)
- Materialien (Farbe, Textur, ...)
- Kameras (Ansichten, Kontrolle der Kamera, ...)
- Lichter (Art der Lichtquelle, Farben, ...)
- Spezialeffekte (Nebel, Schatten, Skyboxes, ...)
- uvm.

Dabei haben die Daten untereinander komplexe Beziehungen, z. B. verwendet verschiedene Objekte das gleiche Material, das gleiche Objekt ist an mehreren Orten instantiiert, Objekte werden Gruppirt, ... Ein *Szenengraphen* strukturieren diese Informationen.

---

### 10.1 Szenengraph

---

Ein Szenengraph ist ein gerichteter, azyklischer Graph mit einem Wurzelknoten, der die gesamte Szene „zusammenhält“. Zum Rendering wird der Graph durchlaufen (traversiert):

- Das Durchlaufen startet an der Wurzel und jeder Kindknoten wird rekursiv abgearbeitet.
- Wurde der gesamte Graph traversiert, ist das Bild fertig (da keine Zyklen erlaubt sind, geschieht dies immer).
- Die konkreten Operationen hängen vom Typ des jeweiligen Knotens ab:
  - Gruppierungen: Ist die aktuelle Gruppe eingeschaltet, traversiere die Kindknoten. Wenn nicht, tue nichts.
  - Transformationen: Anwendung der hinterlegten Transformationsmatrix auf alle Kindknoten.
  - Objektdaten: Ein tatsächliches Objekt, welches gezeichnet wird (unter Anwendung aller vorherigen Transformationen).
- Während der Traversierung werden die verschiedenen zustände jeweils geupdated.

Dieses Konzept hat viele Vorteile:

- Bereits definierte Objekte können wiederverwendet werden.
- Objektdaten sind semantisch Gruppierbar.
- Die Transformationshierarchie ermöglicht die Transformation kompletter Gruppen, ohne diese explizit ändern zu müssen.

---

## 10.2 X3DOM

---

X3DOM ist eine deklarative Szenengraph-API auf Basis von X3D im DOM:

- Deklarativ: Der Szenengraph wird durch ein strukturiertes Textformat (z. B. XML) beschrieben.
- X3D: Szenengraph-Standard, XML-basiert, Nachfolger von VRML; Benötigt traditionell sogenannte X3D-Player.
- DOM: HTML Document Object Modell; Die Dokumenten-Baumstruktur und API in HTML/JavaScript.

X3DOM erlaubt dabei die Verwendung von X3D im DOM durch eine reine Implementierung in JavaScript (dadurch ist kein gesonderter X3D-Player nötig), wobei als Rendering-Backend z. B. WebGL verwendet wird. Bisher gibt es jedoch noch keine native Implementierung in den Browsern.



---

---

## **11 Informationsvisualisierung**

---

### **11.1 Informationsdesign**

---

#### 11.1.1 Referenzmodell von Card

---

### **11.2 Datentypen**

---

#### 11.2.1 1D-Daten, Zeitreihen

---

#### 11.2.2 2D-Daten

---

#### 11.2.3 mD-Daten (multidimensional)

---

#### 11.2.4 Hierarchien

---

#### 11.2.5 Graphen/Netzwerke

---

### **11.3 Kuchendiagramm (1D)**

---

### **11.4 Balkendiagramm (1D)**

---

### **11.5 Liniendiagramm (Zeitreihe)**

---

**Problem: Viele Zeitreihen**

**Problem: Länge**

---

### **11.6 Scatterplot (2D, 3D)**

---

**Problem: Overplotting**

---

### **11.7 Scatterplotmatrix (nD)**

---

### **11.8 Parallele Koordinaten (3D, nD)**

---

**Problem: Overplotting**

**Problem: Viele Dimensionen**

---

---

## **11.9 Node-Link-Diagramm (Hierarchien, Graphen)**

---

**Problem: Layout**

**Problem: Viele Knoten**

---

## **11.10 Treemap (Hierarchien)**

---

**Problem: Überlappung**

**Problem: Größendarstellung**

---

## **11.11 Zusammenfassung**

---

---

---

## **12 Farbe**

---

### **12.1 Dimensionalität**

---

### **12.2 Wahrnehmungskorrelate**

---

### **12.3 Reproduktion**

---

### **12.4 Berechnung von Farbattributen**

---

#### 12.4.1 Das Auge

---

#### 12.4.2 Spektrale Charakterisierung des Auges

---

#### 12.4.3 Spektralwertfunktion

---

#### 12.4.4 Cone Fundamentals

---

### **12.5 Objektfarben, Lichtmatrix und XIEXYZ-Farbraum**

---

### **12.6 Metamerie**

---

### **12.7 Gegenfarbtheorie**

---

### **12.8 Stevenssche Potenzfunktion**

---

### **12.9 CIELAB Farbraum**

---

### **12.10 Technische Farbräume**

---

#### 12.10.1 Geräte RGB

---

#### 12.10.2 Geräteunabhängige RGB

---

#### 12.10.3 YCbCr

---

---

---

12.10.4 HSI/HSV/HSL

---

12.10.5 CMY/CMYK

---

## **12.11 Komplexität von Farbe**

---

12.11.1 Chromatische Adaptation

---

Modellbildung

---

12.11.2 Farbwahrnehmungsphänomene

---

Simultankontrast

---

Crispening Effekt

---

Stevens Effekt

---

Hunt Effekt

---

12.11.3 Farbwahrnehmungsmodelle

---

CIECAM02

---

12.11.4 Kontrastsensitivität

---

S-CIELAB (Spacial-CIELAB)

---

iCAM (Image Color Appearance Model)

---

---

---

## **13 User Interfaces**

---

### **13.1 Interaktion**

---

#### 13.1.1 Möglichkeiten

---

Kommandozeile

---

Menüs

---

Formulare

---

Fragen und Antworten

---

Direkte Manipulation

---

3D-Umgebungen

---

Natürliche Sprache

---

Gesten

---

#### 13.1.2 Designprozess

---

Wasserfallmodell

---

Spiralmodell

---

V-Modell

---

Dynamic Systems Development Method (DSDM)

---

Design Process Model

---

### **13.2 GUI: Benutzeroberflächen**

---

---

---

### 13.2.1 Das WIMP-Interface

---

Fenster-Komponenten

---

**Multiple Document Interface (MDI)**

**Single Document Interface (SDI)**

**Tabbed Document Interface**

---

Dialogboxen

---

**Checkboxen**

**Radio Buttons**

**Listboxen**

**Comboboxen**

**Spinner**

**Slider**

**Weiteres**

---

### 13.2.2 Menübasierte Programme

---

Untermenüs

---

Auswahl (if-then-else-Struktur)

---

**Verschachtelte Entscheidungsstrukturen**

---

Die case-Struktur

---

Modularisierung

---

### 13.2.3 GUI-Anwendungen und Event-basiertes Programmieren

---

Graphical User Interfaces (GUIs)

---

Event-Handler

---

Nutzerinteraktionen

---

Das Delegationsmodell

---

## 13.3 3D-Interaktion

---

---

---

## **14 Multimedia Information Retrieval**

---

### **14.1 Inhaltsbasierte Suche**

---

#### 14.1.1 Mathematische Beschreibung

---

#### 14.1.2 Retrieval Ergebnis

---

### **14.2 Distanzmaße**

---

### **14.3 Query-Modalitäten**

---

#### 14.3.1 Text

---

#### 14.3.2 Example

---

#### 14.3.3 Example-Bilder

---

#### 14.3.4 Sketch

---

### **14.4 Explorative Suche**

---

#### 14.4.1 Research Data

---