Visual Computing

Niclas Kusenbach
LaTeX version: SCHOUTER

Table of Contents

Contents

1 Einführung in Visual Computing

1.1 Grundlagen Visual Computing

Definition

Visual Computing ist die Kombination mehrerer Informatikbereiche, die im Wesentlichen mit Bildern und Modellen arbeiten.

- Umfasst **Computergrafik**, **Computer Vision**, **Mensch-Maschine-Interaktion**, **Mustererkennung** und **Maschinelles Lernen**.
- Enge Verbindung zwischen Computer Vision und Computer Graphics – keine getrennte Betrachtung.
- Zentrale Fragestellungen: Informationsgewinnung aus Daten, effiziente Extraktion relevanter Information.

1.2 Vier exemplarische Themenbereiche

1. **3D Internet:** Erweiterung des Dokumentbegriffs auf 3D-Modelle; Anwendungen in Bildung, Medizin, Kultur.
   - 3D-Web, AR/VR, digitale Kulturgüter, Metaverse.
   - Retro-Digitalisierung vs. Digital Creation (CultLab3D, Flickr-Photogrammetrie).
2. **Skalierbare Objektmodellierung und -erkennung:**
   - Ziel: Erkennung zehntausender Kategorien mittels semantischer Hierarchien.
   - Nutzung von Deep Learning (Convolutional NNs, Transferlernen, 3D-Objektdatenbanken).
3. **Big Data / Visual Analytics:**
   - Kombination von Datenanalyse, ML und Visualisierung zur explorativen Erkenntnisgewinnung.
   - Verarbeitung großer heterogener Datenmengen (z. B. 300 000 Flickr-Bilder von Rom).
4. **Scene Understanding (3D/4D):**
   - Modellierung und Analyse von Szenen in Raum und Zeit.
   - Kombination von Objekterkennung, Tracking, Semantik und Bewegungsanalyse.
   - Anwendungen: Autonomes Fahren, Robotik, Sicherheitsüberwachung, AAL.

1.3 Wichtige Konzepte und Begriffe

- **Deep Learning / Convolutional Networks** – Grundprinzipien, GPU-basierte Berechnungen.
- **Interpretierbarkeit (White-Box vs. Black-Box)** – KI-Erklärbarkeit, Bias und Ethik.
- **Visualisierung und Informationsvisualisierung** – Darstellung komplexer Daten.
- **3D Output und AR/VR Technologien** – z. B. Shapeways, Wikitude.
- **Digitale Kultur und Erhalt von Kulturgütern** – z. B. Project Mosul, 3D-Rekonstruktionen.

1.4 Empfohlene Literatur

Autor / Titel	Relevanz
Szeliski – *Computer Vision: Algorithms and Applications*	Fundamentale Algorithmen zur Bildverarbeitung und -analyse.

Blundell – *An Introduction to Computer Graphics*	Grundlagen der 3D-Darstellung.
Dix et al. – *Human Computer Interaction*	Basis für Mensch-Maschine-Schnittstellen.
Burger & Burke – *Digitale Bildverarbeitung – algorithmische Einführung*	Mathematische Grundlagen für Bildverarbeitung.

1.5 Zusammenfassung

- Visual Computing verbindet Grafik, Vision und Interaktion.
- Vier Kernbereiche (3D Internet, Objekterkennung, Visual Analytics, Scene Understanding) bilden die strukturierende Grundlage der Vorlesung und sind klausurrelevant.
- Verständnis der praktischen Beispiele und Übungen ist essentiell für die Klausur.

2 Wahrnehmung (Perception)

2.1 Motivation und Kognition (Motivation and Cognition)

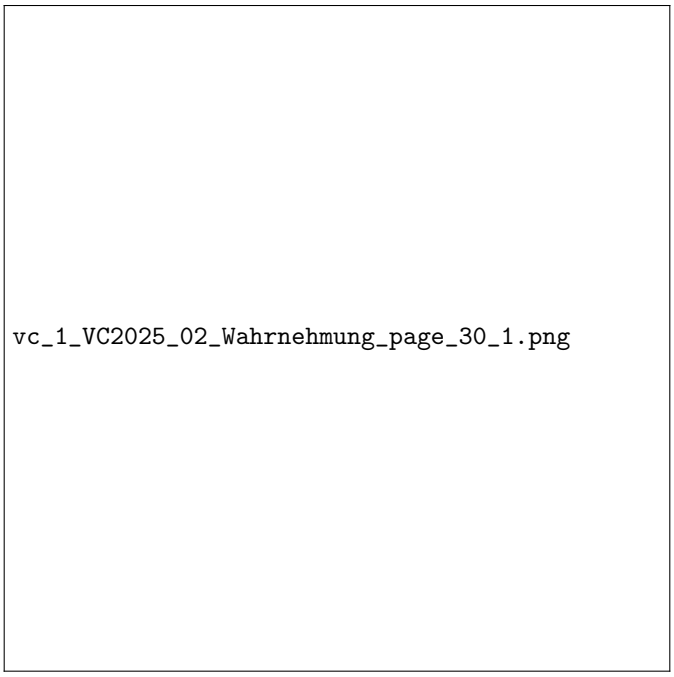
- **Warum VC? (Why VC?)** Die Leistungsfähigkeit von Rechnern wächst exponentiell (**Moore's Law**), aber die Kapazität von Menschen ist (fast) konstant (**Darwin's Law**). VC hilft, diese Lücke zu überbrücken.
- **Kognition (Cognition):** Sammelbegriff für alle Prozesse des Wahrnehmens und Erkennens (Denken, Erinnern, Lernen, etc.).
- **Modell der Informationsverarbeitung (Model of Information Processing):** Ein modulares 3-Stufenmodell:
  1. **Perception** (Wahrnehmung durch Sinne)
  2. **Decision** (Entscheidungsfindung im Gehirn)
  3. **Response** (Reaktion durch Körper)
- **Bearbeitungszeiten (Processing Times):** Jedes Untersystem benötigt Zeit.
  - Wahrnehmung (Perception): ≈ 100 ms
  - Entscheidung (Cognition): ≈ 70 ms
  - Reaktion (Motor): ≈ 70 ms
- **Wahrnehmung vs. Realität:** Was wir wahrnehmen, ist kein direktes Abbild der Realität, sondern eine partielle Hypothese, die auf unvollständiger Information basiert.

2.2 Das Visuelle System (The Visual System)

Visueller Reiz (Visual Stimulus)

Ein äußerer visueller Reiz ist **elektromagnetische Strahlung**. Sichtbares Licht liegt im Wellenlängenbereich von ca. 400nm (violett) bis 700nm (rot). Die Frequenz v und Wellenlänge λ hängen über $v \cdot \lambda = c$ zusammen.

2.2.1 Aufbau des Auges (Structure of the Eye)

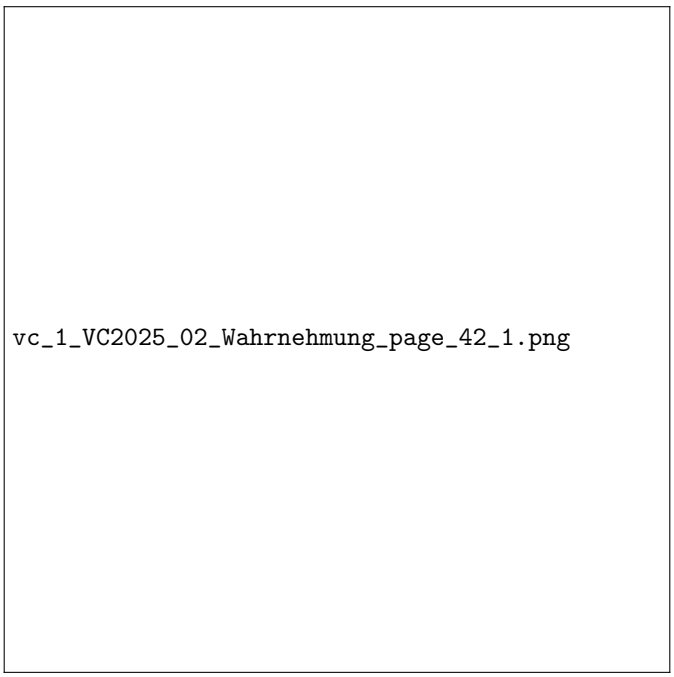


- **Optische Elemente:** Hornhaut (Kornea), Linse, Iris (Blende, 2-8mm), Glaskörper.
- **Linse (Lens):** Akkomodation (Scharfeinstellung).
- **Retina (Netzhaut):** Enthält die Photorezeptoren.
- **Fovea Centralis:** Bereich der höchsten Auflösung (im "gelben Fleck" / Macula lutea).
- **Blinder Fleck (Blind Spot):** Papilla nervi optici; Austrittspunkt des Sehnervs, keine Rezeptoren.

2.2.2 Photorezeptoren (Photoreceptors)

Es gibt zwei Haupttypen von Photorezeptoren auf der Retina:

- **Stäbchen (Rods):**
  - ca. 100-120 Mio.
  - Hauptsächlich außerhalb der Fovea.
  - Für **Nachtsehen (skotopisches Sehen)**.
  - Sehr lichtempfindlich, kein Farbsehen.
  - Empfindlichkeitsmaximum bei 498 nm (grün).
- **Zapfen (Cones):**
  - ca. 7-8 Mio.
  - Hauptsächlich **in der Fovea** (Bereich des schärfsten Sehens).
  - Für **Tagsehen (photopisches Sehen)**.
  - 3 Typen für Farbsehen:
    - **S-Zapfen** (Short): Max. bei 420 nm (Blau).
    - **M-Zapfen** (Medium): Max. bei 534 nm (Grün).
    - **L-Zapfen** (Long): Max. bei 564 nm (Rot).



Bayer-Sensor

Digitale Kamerasensoren nutzen oft ein **Bayer-Muster**. Dies ist ein Farbfilter-Array, meist mit **50% Grün, 25% Rot und 25% Blau**. Grün ist privilegiert, da das menschliche Auge für Grün den größten Beitrag zur **Helligkeits- und Kontrastwahrnehmung** leistet (72% Grünanteil).

2.3 Vorverarbeitung & Helligkeit (Preprocessing & Brightness)

2.3.1 Signalverarbeitung in der Retina

Das Licht trifft (paradoxerweise) erst auf die Ganglien- und Bipolarzellen, bevor es die Stäbchen und Zapfen erreicht.

- **Bipolar-Zellen:** Sammeln, gewichten und leiten Informationen weiter.
- **Horizontal- & Amakrin-Zellen:** Kombinieren Signale mehrerer Rezeptoren (räumlich) bzw. verarbeiten zeitliche änderungen.
- **Ganglien-Zellen:** Integrieren Informationen, z.B. für **Kontrastwahrnehmung** durch Unterschied zwischen Zentrum und Peripherie (Center-Surround-Antagonismus).

2.3.2 Optische Täuschungen (Optical Illusions)

Diese frühe Signalverarbeitung führt zu Täuschungen, die zeigen, dass Wahrnehmung nicht objektiv ist:

- **Hermann-Gitter:** Graue Flecke erscheinen in den Kreuzungen eines weißen Gitters auf schwarzem Grund.
- **Mach-Bänder (Mach Bands):** An Kanten zwischen unterschiedlich hellen, aber homogenen Flächen werden helle/dunkle Bänder wahrgenommen, wo keine sind (eine Art "Uberschwingen" der Wahrnehmung).
- **Simultankontrast (Simultaneous Contrast):** Die wahrgenommene Helligkeit einer Fläche hängt von der Helligkeit ihrer Umgebung ab. Ein identisches Grau erscheint auf schwarzem Grund heller als auf weißem Grund.

2.3.3 Helligkeitswahrnehmung (Brightness Perception)

- Helligkeit (Brightness) ist **keine absolute Größe**, sondern subjektiv.
- Sie ist u.a. abhängig von der Reizstärke (Leuchtdichte), der Adaption an vorherige Leuchtdichten und der Umgebungsleuchtdichte.
- **Weber-Fechnersches Gesetz:** Beschreibt den Zusammenhang zwischen Reizintensität (R) und Hellempfindung (L).
  - **Webersches Gesetz (Schwelle):** $\Delta L = \frac{\Delta R}{R} = const.$ (minimaler Kontrast für Wahrnehmung ca. 0.8%)
  - **Fechnersches Gesetz:** $L = c_1 \times \log R$
  - **Stevensches Gesetz (State-of-the-Art):** $E = c_2 \times R^k$ (für Licht $k = 0.3$)

2.3.4 Auflösung und Kontrast (Resolution and Contrast)

- **Sehschärfe (Visual Acuity):** Die Fähigkeit, kleine Details zu erkennen, ist begrenzt. Z.B. Punktsehschärfe ca. 1 Bogenminute ($1'$).
- **Kontrastempfindlichkeit (Contrast Sensitivity):** Gemessen mit Sinus-Mustern (sinusoidal gratings).
- **Contrast Sensitivity Function (CSF):** Beschreibt die Auflösung im Frequenzraum. Zeigt, dass das Auge für mittlere Ortsfrequenzen (ca. 2-5 Zyklen/Grad) am empfindlichsten ist und die Empfindlichkeit zu sehr hohen (Details) und sehr niedrigen Frequenzen (langsame Übergänge) abfällt.

2.4 Informationsextraktion: Tiefenwahrnehmung (Depth Perception)

Das visuelle System nutzt verschiedene Hinweisreize (**Depth Cues**), um Raumwahrnehmung zu erzeugen.

- **1. Binokulare Cues (Zwei Augen):**

  - **Disparität / Parallaxe:** Der Haupt-Cue. Da die Augen getrennt sind, sehen sie leicht unterschiedliche Bilder. Das Gehirn fusioniert diese.
  - **Positive Parallaxe:** Objekte erscheinen *hinter* der Bildebene.
  - **Negative Parallaxe:** Objekte erscheinen *vor* der Bildebene.
  - **Akkomodation** (Linsenanpassung) und **Konvergenz** (Augenstellung).
- **2. Pictorial Depth Cues (Monokular / Bildlich):**
  - **Linearperspektive:** Parallele Linien konvergieren in der Ferne.
  - **Verdeckung (Occlusion):** Ein Objekt, das ein anderes verdeckt, wird als näher wahrgenommen.
  - **Texturgradient:** Texturen werden mit der Entfernung dichter und feiner.
  - **Atmosphärische Tiefe:** Entfernte Objekte erscheinen blasser und bläulicher.
  - **Schattenwurf (Shadows):** Wichtig für Position und Form; Annahme: Licht kommt von oben.
  - Weitere: Fokus/Blur, Vertraute Größe, Höhe im Gesichtsfeld, etc..
- **3. Dynamische Depth Cues (Bewegung):**
  - **Bewegungsparallaxe (Motion Parallax):** Objekte, die näher sind, bewegen sich bei einer Kopfbewegung scheinbar schneller als entfernte Objekte.
  - **Kinetischer Tiefeneffekt (Kinetic Depth Effect):** 3D-Struktur wird aus der 2D-Projektion einer Bewegung extrahiert (z.B. "Structure from Motion").

Die Depth Cues sind nicht redundant, sondern **additiv** und werden je nach Aufgabe (Task) **flexibel gewichtet**.

2.5 Aufmerksamkeit und Gedächtnis (Attention and Memory)

- **Frühe Wahrnehmung (Preattentive Processing):** Bestimmte Merkmale (Farbe, Größe, Richtung, Schattierung) werden sehr schnell (ca. ¡ 10 ms) und parallel verarbeitet, bevor die bewusste Aufmerksamkeit greift. **Verbindungen** von Merkmalen (z.B. "roter Kreis") erfordern Aufmerksamkeit.
- **Aufmerksamkeit (Attention):** Dient als **Filter** oder "Gateway to Memory".
- **Veränderungsblindheit (Change Blindness):** Unfähigkeit, große änderungen in einer Szene zu bemerken, wenn die Aufmerksamkeit abgelenkt ist (z.B. durch Flimmern). Dies zeigt, dass wir kein vollständiges Bild der Welt im Kopf haben.
- **Arbeitsgedächtnis (Working Memory):**
  - Schneller Zugriff (ca. 70 ms), schneller Verfall (ca. 200 ms).
  - Sehr begrenzte Kapazität: 7 ± 2 **"Chunks"** (Miller, 1956).
  - "Chunks" sind sinnvolle Einheiten (z.B. DA, TU, VC, VL statt DATUVCVL).
- **Langzeitgedächtnis (Long-term Memory):**
  - Nahezu unbegrenzte Kapazität.
  - Langsamerer Zugriff (ca. 100 ms).

3 Erkennung

3.1 Das Problem der Objekterkennung

3.1.1 Herausforderungen der Erkennung

Menschliche Wahrnehmung ist Computern weit überlegen. Die Erkennung ist schwierig aufgrund von:

- **Mehrdeutigkeit & Illusionen:** (z.B. Don Quixote-Gesicht, Mona Lisa-Mosaik, ambivalente Elefantenbeine).
- **Tarnung & Verdeckung:** (z.B. Bev Doolittles Pferde im Schnee).
- **Kontext:** Die Interpretation von Teilen hängt stark vom Gesamtkontext ab (z.B. verschwommene Straßenszene, Punktewolken-Wörter).
- **Fehler im Kontext:** M.C. Escher-Bilder sind *lokal* konsistent, aber *global* unmöglich.

3.1.2 Intuition: Wie funktioniert Erkennung?

Objekterkennung basiert auf zwei Hauptkomponenten:

1. **Lokale Beschreibung / Merkmale:** (z.B. Augen, Nase, Mund bei einem Gesicht).
2. **Globale Anordnung der Merkmale:** (z.B. relative Positionen und Größen der Merkmale zueinander).

Weitere wichtige Aspekte sind Segmentierung und Szenenkontext.

Pictorial Structure (Fischler & Elschlager, 1973) Ein klassisches Modell zur Objekterkennung, das diese Intuition formalisiert.

- **Teile (Parts):** 2D-Bildfragmente (lokale Merkmale).
- **Aufbau (Structure):** Die Anordnung der Teile, oft modelliert durch "Federn", die die relative Position und Deformation beschränken.

Herausforderungen für dieses Modell sind **Deformationen** (Teile ändern ihre Position) und **Durcheinander** (Clutter, irrelevante Merkmale im Hintergrund).

3.2 Bayes Decision Theory (BDT)

Die BDT ist ein mathematisches Framework zur optimalen Entscheidungsfindung unter Unsicherheit. Ihr Ziel ist es, Klassifikationsentscheidungen so zu treffen, dass die **erwartete Fehlklassifikationsrate minimiert** wird. Sie nutzt dazu Wahrscheinlichkeiten als formale Beschreibung von Unsicherheit.

Die drei Kernkonzepte der BDT

1. **A Priori (Prior)** $P(C_k)$**:** Die Wahrscheinlichkeit einer Klasse C_k, *bevor* wir Daten x beobachtet haben. Der Prior kodiert unser Vorwissen oder unsere Erwartungen über die Häufigkeit von Klassen. Wichtig: Priors wirken wie ein "Bias" zugunsten bestimmter Klassen.
2. **Klassenspezifische Dichte (Likelihood)** $p(x|C_k)$**:** Die Wahrscheinlichkeit (genauer: Wahrscheinlichkeitsdichte), die Merkmalsausprägung x zu beobachten, *unter der Annahme*, dass die wahre Klasse C_k ist. Sie beschreibt also, wie typisch ein bestimmtes Datenmuster x für eine Klasse ist. Hinweis: Likelihood ist eine Funktion in x, nicht in der Klasse.
3. **A Posteriori (Posterior)** $P(C_k|x)$**:** Nachbeobachtete Wahrscheinlichkeit einer Klasse, *nachdem* wir x kennen. Dies ist genau die Größe, die wir für die eigentliche Entscheidung benötigen.

Bayes' Theorem (Verbindung der Konzepte)

$$P(C_k|x) = \frac{p(x|C_k)\,P(C_k)}{p(x)}$$

Die Evidenz

$$p(x) = \sum_j p(x|C_j)\,P(C_j)$$

stellt sicher, dass die Posterior-Wahrscheinlichkeiten normiert sind. Wichtig: Für die Entscheidung ist $p(x)$ oft irrelevant, da es für alle Klassen gleich ist.

$$\text{Posterior} = \frac{\textbf{Likelihood} \times \textbf{Prior}}{\text{Evidenz}}$$

3.2.1 Bayes-Entscheidungsregel (Fehlerminimierung)

Wir wählen die Klasse mit der größten posterioren Wahrscheinlichkeit:

$$C^*(x) = \arg\max_k P(C_k|x).$$

Da der Nenner $p(x)$ konstant ist, genügt:

$$C^*(x) = \arg\max_k p(x|C_k)\,P(C_k).$$

Intuition: Die Likelihood misst, wie gut die Daten zu einer Klasse passen; der Prior berücksichtigt, wie plausibel diese Klasse im Voraus ist. Der Schnittpunkt der Produkte $p(x|C_k)P(C_k)$ bildet die **Entscheidungsgrenze**.

3.2.2 Likelihood Ratio Test (Spezialfall 2 Klassen)

Für zwei Klassen C_1 und C_2:

$$\text{Entscheide } C_1 \text{ wenn: } \frac{p(x|C_1)}{p(x|C_2)} > \frac{P(C_2)}{P(C_1)}.$$

Links: datengetriebene Evidenz, Rechts: Schwellwert aus Vorwissen. So sieht man explizit, wie ein starker Prior eine Entscheidung "umkippen" kann.

3.2.3 Prior und Likelihood in der Praxis

Die Trennung zwischen Datenmodell (Likelihood) und Vorwissen (Prior) ermöglicht robuste Modelle:

- **Spracherkennung:** Likelihood = akustisches Modell; Prior = Sprachmodell. Das Sprachmodell zieht uns aus akustischen Mehrdeutigkeiten heraus.
- **Bildverarbeitung:** Likelihood = Appearance Model; Prior = Kontextwissen über sinnvolle Objekte am Ort.

3.3 Naive Bayes Klassifikator

3.3.1 Das Problem hoher Dimensionen

Für d Merkmale:

$$x = (x_1, \ldots, x_d)$$

müssten wir die gemeinsame Dichte $p(x|C_k)$ im d-dimensionalen Raum schätzen. Dies benötigt exponentiell viele Daten: **Fluch der Dimensionalität**. Schon wenige Dutzend Dimensionen machen klassische Dichteschätzung praktisch unmöglich.

Die Naive-Unabhängigkeitsannahme Um das Problem zu umgehen, wird angenommen:

$$p(x|C_k) = p(x_1, \ldots, x_d|C_k) \approx \prod_{i=1}^{d} p(x_i|C_k).$$

Das heißt: Die Merkmale sind *bedingt unabhängig* gegeben die Klasse. Diese Annahme ist oft falsch, aber in vielen Anwendungen erstaunlich effektiv, weil:

- Abhängigkeiten zwischen Merkmalen sich teilweise gegenseitig ausgleichen,
- Die Entscheidung meist nur eine Rangordnung braucht (MAP), nicht exakte Wahrscheinlichkeiten.

3.3.2 Naive Bayes Entscheidungsregel

Durch Einsetzen der Zerlegung:

$$C^*(x) = \arg\max_k P(C_k) \prod_{i=1}^{d} p(x_i|C_k).$$

Lernen bedeutet:

1. Schätzen der Priors $P(C_k)$ (Häufigkeiten).
2. Schätzen der eindimensionalen Dichten $p(x_i|C_k)$ für alle Merkmale und Klassen.

Log-Space (Numerische Stabilität) Da Produkte kleiner Wahrscheinlichkeiten numerisch instabil sind:

$$C^*(x) = \arg\max_k \left[\log P(C_k) + \sum_{i=1}^{d} \log p(x_i|C_k) \right].$$

Logarithmen ersetzen Multiplikationen durch Additionen, ohne die Entscheidung zu verändern.

3.4 Fallstudie: Gesichtsdetektion

Ein Beispiel für **Appearance-Based Methods**: Modelle werden direkt aus (großen) Bilddatensammlungen gelernt.

3.4.1 Sliding Window Ansatz

- Ein Fenster (z.B. 19×19 Pixel) wird über das gesamte Bild geschoben.
- Bei jeder Position wird entschieden: "Gesicht" (C_1) oder "Kein Gesicht" (C_2).
- Das Bild wird skaliert (z.B. Faktor 1.2 verkleinert) und der Vorgang wiederholt sich, um Gesichter aller Größen zu finden.

Fallstudie: Schneiderman & Kanade (1998) Ein sehr erfolgreicher Gesichtdetektor, der auf Naive Bayes basiert.

- **1. Repräsentation (Merkmale x_i): Wavelet-Koeffizienten** an bestimmten Frequenzen, Orientierungen und **Positionen** (f_i, u_i, v_i). Dies kodiert sowohl lokale Merkmale (Kanten) als auch deren globale Anordnung (Position).
- **2. Trainingsdaten:**
  - *Positive Beispiele (C_1):* Tausende von Bildern, die Gesichter enthalten (normalisiert).
  - *Negative Beispiele (C_2):* Tausende von Bildern, die *keine* Gesichter enthalten.
- **3. Klassifikator & Lernen:** Naive Bayes. Das "Lernen" besteht darin, die Wahrscheinlichkeiten $P(x_i|C_{\text{face}})$ und $P(x_i|C_{\text{non-face}})$ für jedes Merkmal x_i zu schätzen. Dies geschieht durch **Zählen (Erstellen von Histogrammen)** in den positiven und negativen Trainingsdatensätzen.
- **Multi-View:** Um Gesichter aus verschiedenen Winkeln zu erkennen, werden separate Detektoren trainiert (z.B. Frontal, Linksprofil, Rechtsprofil) und die Ergebnisse kombiniert.

3.5 Erkennungsarten (Biometrie)

Gesichtserkennung ist ein biometrisches Verfahren. Man muss zwischen verschiedenen Aufgaben unterscheiden:

- **Detektion:** (Face vs. Non-Face) Ist überhaupt ein Objekt (Gesicht) vorhanden?
- **Verifikation (1:1):** "Bin ich das?" (z.B. Smartphone entsperren).
  - Eine Person gibt ihre Identität an (z.B. Nutzer-ID).
  - Das System vergleicht die aktuelle Probe **nur mit dem einen** gespeicherten Template dieser ID.
  - Ausgabe: Ja / Nein.
- **Identifikation (1:n):** "Wer ist das?" (z.B. Überwachung).
  - Eine Person zeigt nur ihr Merkmal (Gesicht).
  - Das System vergleicht die Probe mit **allen** n Templates in der Datenbank.
  - Ausgabe: Eine Kandidatenliste (die m besten Treffer, $m \ll n$).

4 Fouriertheorie

4.1 Motivation

Die Fouriertheorie bietet eine neue Perspektive, um Signale und Funktionen zu analysieren, indem sie vom **Ortsraum** (z.B. Position, Zeit) in den **Frequenzraum** (z.B. Frequenz, Wellenlänge) wechselt.

- **Beispiele:**
  - **Optik/Physik:** Lichtbeugung an einem Spalt. Ein Spalt kann mathematisch als **Rechteckfunktion** ($rect(x)$) beschrieben werden. Das resultierende Beugungsmuster (die Amplitudenverteilung) $B(\varphi)$ hat die Form einer **sinc-Funktion** ($\frac{\sin x}{x}$). Die gemessene Intensität $I(x)$ ist proportional zum Quadrat der Amplitude ($I \propto B^2$), also eine $sinc^2$-Funktion.
  - **Medizintechnik (MRT):** Ein MR-Scanner misst direkt Frequenzmuster im Fourierraum.
  - **Menschliche Wahrnehmung:** Die Kontrastempfindlichkeit des Auges wird im Frequenzraum gemessen, oft mittels sinusförmiger Muster (sinusoidal gratings).
- **Kernidee:** Es besteht ein direkter Zusammenhang zwischen der Gestalt eines Objekts (Ortsraum) und seiner Amplitudenfunktion (Frequenzraum). Dieser Zusammenhang ist die **Fourier-Transformation**.
- **Erstes Fourier-Paar:** $rect(x) \xrightarrow{FT} sinc(u)$.

4.2 Mathematische Grundlagen

- **Vektorraum (R^n):** Ein Raum, in dem Vektoren addiert und skalar multipliziert werden können.
- **Skalarprodukt:** Definiert Längen und Winkel. $\langle \vec{v}, \vec{w} \rangle = \sum_{i=1}^{n} v_i w_i$.
- **Basis:** Ein Satz linear unabhängiger Vektoren (z.B. $\vec{e}_1, \vec{e}_2$), die den Raum aufspannen. Jeder Vektor $\vec{v}$ ist eine Linearkombination der Basis: $\vec{v} = a_1\vec{e}_1 + a_2\vec{e}_2$.
- **Funktionenräume:**
  - Die Elemente sind nun **Funktionen** statt Vektoren.
  - Diese Räume sind i.d.R. **unendlich-dimensional**.
  - **Ziel:** Finde **Basisfunktionen**, um (beliebige) Funktionen als Linearkombination dieser Basen darzustellen. Dies leistet die **Fouriertheorie**.

4.3 Die Fourier-Reihe (Fourier Series)

Grundidee der Fourier-Reihe

Jede **2π-periodische Funktion** $f(x)$, die die **Dirichlet-Bedingungen** erfüllt, kann als unendliche Summe (Überlagerung) von Sinus- und Kosinusfunktionen dargestellt werden.
Dirichlet-Bedingungen :

1. Endliche Anzahl von Unstetigkeiten pro Periode.
2. Endliche Anzahl von Maxima und Minima pro Periode.
3. In jeder Periode integrierbar (d.h. $\int |f(x)|dx < \infty$ pro Periode).

4.3.1 Analogie zum Vektorraum

- **Skalarprodukt für Funktionen:** $\langle f, g \rangle = \int_{-\pi}^{\pi} f(t)g(t)dt$.

- **Orthogonale Basis:** Die Funktionen $u_n(t) = \cos(nt)$ und $v_n(t) = \sin(nt)$ bilden eine **orthogonale Basis** im Funktionenraum H. Das bedeutet, ihr Skalarprodukt ist 0, wenn sie nicht identisch sind (z.B. $\int_{-\pi}^{\pi} \cos(nt)\sin(mt)dt = 0 \ \forall n, m$).

4.3.2 Formeln der Fourier-Reihe

Reelle Fourier-Reihe

$$f(x) = a_0 + \sum_{n=1}^{\infty}(a_n \cos(nx) + b_n \sin(nx))$$

Die **Fourier-Koeffizienten** a_n, b_n werden durch Projektion von $f(x)$ auf die Basisfunktionen berechnet:

- $a_0 = \frac{1}{2\pi}\int_{-\pi}^{\pi} f(x)dx$ (Der "Gleichanteil" oder Mittelwert)
- $a_m = \frac{1}{\pi}\int_{-\pi}^{\pi} f(x)\cos(mx)dx$ (für $m > 0$)
- $b_m = \frac{1}{\pi}\int_{-\pi}^{\pi} f(x)\sin(mx)dx$

- **Symmetrie-Eigenschaften:**
  - **Gerade Funktion** ($f(-t) = f(t)$, z.B. $\cos(x)$): Alle $b_n = 0$.
  - **Ungerade Funktion** ($f(-t) = -f(t)$, z.B. $\sin(x)$): Alle $a_n = 0$ (inkl. a_0).
- **Beispiel (Rechteck-Schwingung):** Dies ist eine ungerade Funktion. Daher sind alle $a_n = 0$. Die b_n Koeffizienten sind $\frac{4k}{n\pi}$ für ungerade n und 0 für gerade n.

$$f(x) = \frac{4k}{\pi}\left(\sin(x) + \frac{1}{3}\sin(3x) + \frac{1}{5}\sin(5x) + \dots\right)$$

(Die Folien 39-43 visualisieren, wie diese Summe die Rechteckfunktion approximiert).

Komplexe Fourier-Reihe

Mit der Euler-Identität ($e^{i\phi} = \cos\phi + i\sin\phi$ und $e^{inx} = \cos(nx) + i\sin(nx)$) lässt sich die Reihe kompakter schreiben :

$$f(x) = \sum_{n=-\infty}^{\infty} c_n e^{inx}$$

4.4 Die Fourier-Transformation (FT)

Motivation der Fourier-Transformation

Erweiterung der Fourier-Reihe auf **nicht-periodische Funktionen**.
Dies geschieht durch einen Grenzübergang: Die Periode L der Funktion wird unendlich groß ($L \to \infty$).
Dabei wird die **diskrete Summe** der Fourier-Reihe (über n) zu einem **kontinuierlichen Integral** (über u).
Das Spektrum ist nicht mehr diskret (Vielfache einer Grundfrequenz), sondern **kontinuierlich**.

Das Fourier-Transformationspaar

- **Fourier-Transformation (FT):** (Analyse: Ortsraum → Frequenzraum)

$$F(u) = \int_{-\infty}^{\infty} f(t) e^{-2\pi i u t} dt$$

- **Inverse Fourier-Transformation (iFT):** (Synthese: Frequenzraum → Ortsraum)

$$f(x) = \int_{-\infty}^{\infty} F(u) e^{+2\pi i u x} du$$

$f(x)$ ist oft reell, aber $F(u)$ ist i.d.R. komplex: $F(u) = \mathrm{Re}(F(u)) + i\ \mathrm{Im}(F(u))$.

4.4.1 Wichtige Fourier-Transformationspaare

- **Rechteck-Funktion ↔ sinc-Funktion:**
  - $f(x) = rect(x)$
  - $F(u) \propto sinc(u)$
- **Dirac-Delta-Distribution ↔ Konstante:**
  - Die **Dirac-Delta-Distribution** $\delta(t)$ ist keine Funktion, sondern eine Distribution, definiert durch ihre **"Sampling-Eigenschaft":** $\int f(t)\delta(t-t')dt = f(t')$.
  - $f(x) = \delta(x) \xrightarrow{FT} F(u) = 1$ (Ein Impuls im Ort enthält alle Frequenzen)
  - $f(x) = K \xrightarrow{FT} F(u) = K \cdot \delta(u)$ (Eine Konstante im Ort ist nur die Frequenz 0)
- **Kosinus ↔ Zwei Delta-Peaks:**
  - $f(x) = \cos(kx)$
  - $F(u) \propto (\delta(u-k) + \delta(u+k))$
  - Eine reine Schwingung im Ortsraum entspricht zwei scharfen Peaks (diskreten Frequenzen) im Frequenzraum.

4.5 Faltung und Filterung

Faltung (Convolution)

Das Faltungsintegral $h(t)$ zweier Funktionen f und g ist definiert als:

$$h(t) = (f \circ g)(t) = \int_{-\infty}^{\infty} f(x) g(t-x) dx$$

Graphisch entspricht dies einer "Spiegelung, Verschiebung, Multiplikation und Integration".
- Bsp: $rect(x) \circ rect(x) = triangle(x)$.

Der Faltungssatz (Convolution Theorem)

Dies ist einer der wichtigsten Sätze der Fouriertheorie:
- Eine **Faltung im Ortsraum** entspricht einer **Multiplikation im Frequenzraum**.
- Eine **Multiplikation im Ortsraum** entspricht einer **Faltung im Frequenzraum**.

$$h(t) = f(t) \circ g(t) \quad \xrightarrow{FT} \quad H(u) = F(u) \cdot G(u)$$

4.5.1 Anwendung: Filterung

Der Faltungssatz ist die Grundlage für effiziente Filterung :

1. Ein Signal $f(x)$ soll gefiltert werden.
2. Anstatt eine aufwändige Faltung im Ortsraum $(f \circ g)$ durchzuführen...
3. ...transformiert man Signal $(f \to F)$ und Filterfunktion $(g \to G)$ in den Frequenzraum.
4. Dort wird eine **einfache Multiplikation** durchgeführt: $H(u) = F(u) \cdot G(u)$.
5. Das Ergebnis $H(u)$ wird zurück in den Ortsraum transformiert $(H \to h)$, um das gefilterte Signal $h(x)$ zu erhalten.

Dies ist oft (via *Fast Fourier Transform*, FFT) viel schneller als die direkte Faltung.

4.6 Abtastung von Signalen (Sampling)

4.6.1 Modell der Abtastung

- **Problem:** Ein kontinuierliches Signal $f(x)$ muss in diskrete Werte $f(n\Delta x)$ für die digitale Verarbeitung umgewandelt werden.
- **Mathematisches Modell:** Abtastung ist eine **Multiplikation** des Signals $f(x)$ mit einer **Kamm-Funktion** (einer Kette von δ-Impulsen im Abstand Δx).

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

4.6.2 Abtastung im Frequenzraum

- Laut Faltungssatz (Multiplikation im Ortsraum → Faltung im Frequenzraum) wird das Spektrum $F(u)$ des Signals mit der FT der Kamm-Funktion gefaltet.
- Die FT einer Kamm-Funktion (Abstand Δx) ist wieder eine Kamm-Funktion (Abstand $1/\Delta x$).
- **Folge:** Das Spektrum $\hat{F}(u)$ des abgetasteten Signals ist eine **periodische Wiederholung** des Originalspektrums $F(u)$, wobei die Kopien den Abstand $1/\Delta x$ haben.

4.6.3 Aliasing und das Abtasttheorem

Angenommen, das Signal ist **bandbegrenzt**, d.h. seine höchste Frequenz ist u_G ($F(u) = 0$ für $|u| > u_G$).

- **Fall 1: Korrekte Abtastung** $(1/\Delta x > 2u_G)$
  - Die Abtastfrequenz $(1/\Delta x)$ ist mehr als doppelt so hoch wie die maximale Signalfrequenz (u_G).
  - Die periodischen Kopien von $F(u)$ im Frequenzraum **überlappen nicht**.
  - Das Originalsignal kann **fehlerfrei rekonstruiert** werden (z.B. durch einen Tiefpassfilter, der nur die zentrale Kopie isoliert).
- **Fall 2: Unterabtastung** $(1/\Delta x < 2u_G)$
  - Die Abtastfrequenz ist zu niedrig.
  - Die Kopien von $F(u)$ **überlappen sich**.
  - In den Überlappungsbereichen addieren sich Frequenzen. Hohe Frequenzen "erscheinen" fälschlicherweise als niedrige Frequenzen.
  - Dieser irreversible Fehler wird **Aliasing** genannt. (z.B. Wagenrad-Effekt bei Filmen , Moiré-Muster).

Abtasttheorem von Whittaker-Shannon

Eine bandbegrenzte Funktion (höchste Frequenz u_G) kann aus ihren Abtastwerten $f(n\Delta x)$ fehlerfrei rekonstruiert werden, wenn die Abtastfrequenz $f_s = 1/\Delta x$ **mindestens doppelt so hoch** wie die höchste Signalfrequenz u_G ist.

$$f_s > 2u_G \quad \textbf{oder} \quad \frac{1}{\Delta x} > 2u_G$$

Die Frequenz $2u_G$ wird als **Nyquist-Frequenz** bezeichnet.

4.7 Zusammenfassung der Konzepte

- **Fourier-Reihe:**
  - **Für:** 2π-periodische Funktionen.
  - **Spektrum:** **Diskret** (Vielfache einer Grundfrequenz).
  - **Formel:** $f(x) = \sum c_n e^{inx}$.
- **Fourier-Transformation:**
  - **Für:** Nicht-periodische Funktionen.
  - **Spektrum:** **Kontinuierlich**.
  - **Formel:** $F(u) = \int f(t)e^{-2\pi iut}dt$.
- **Faltungssatz:**
  - $f(t) \circ g(t) \longleftrightarrow F(u) \cdot G(u)$
  - Ermöglicht effiziente **Filterung** im Frequenzraum.
- **Abtasttheorem:**
  - Abtastung (Multiplikation mit δ-Kamm) im Ortsraum $\rightarrow$ Periodisierung (Faltung mit δ-Kamm) im Frequenzraum.
  - Zur Rekonstruktion muss die Abtastfrequenz f_s größer als die doppelte maximale Signalfrequenz u_G sein ($f_s > 2u_G$), um **Aliasing** zu vermeiden.

5 Bilder

Ein digitales Bild entsteht durch die Diskretisierung einer kontinuierlichen Szene.

- **Lochkameramodell:** Projektion der 3D-Welt auf eine 2D-Ebene.
- **Rasterisierung:** Umwandlung des kontinuierlichen Signals in ein digitales Gitter (Pixel).
- **Repräsentation:** Ein Bild wird als Matrix von Intensitätswerten (Grauwerten oder Farbkanälen) dargestellt.

5.1 Digitale Bildverarbeitung im Ortsraum

Im Ortsraum (Spatial Domain) werden Operationen direkt auf den Pixelwerten des Bildes durchgeführt.

5.1.1 Pixeloperationen

Pixeloperationen manipulieren einen Pixelwert unabhängig von seiner Nachbarschaft (globaler Kontext, aber lokale Anwendung).

Grauwert-Abbildung (Mapping)

Die Transformation eines Pixels an der Stelle (m, n) wird beschrieben durch:

$$g[m, n] = T(f[m, n])$$

wobei f das Eingabebild, g das Ausgabebild und T die Transformationsfunktion ist.

Wichtige Pixeloperationen:

- **Negativ:** $g[m, n] = f_{max} - f[m, n]$ (Invertierung der Helligkeit).
- **Binärisierung (Thresholding):**

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

  Dient zur Trennung von Vorder- und Hintergrund.
- **Grauwertfensterung:** Hervorheben eines bestimmten Intensitätsintervalls (z.B. in der Medizintechnik/CT).
- **Kontrastspreizung:** Abbildung des genutzten Grauwertbereichs auf die volle Skala (z.B. 0-255), um den visuellen Kontrast zu erhöhen.
- **Mittelung:** Unterdrückung von unkorreliertem Rauschen durch Mittelung über mehrere Aufnahmen derselben Szene ($g = \frac{1}{k} \sum f_i$).

5.1.2 Histogramme

Ein Histogramm ist die graphische Darstellung der Häufigkeitsverteilung der Grauwerte in einem Bild.

- **Bildhelligkeit:** Entspricht dem Mittelwert aller Grauwerte.
- **Bildkontrast:** Entspricht der Varianz aller Grauwerte.

Histogrammausgleich (Equalization): Ziel ist es, die Grauwerte so umzuverteilen, dass sie gleichmäßig über den gesamten Bereich verteilt sind (Maximierung des Kontrasts). Dies geschieht über die **Summenwahrscheinlichkeit** (kumulative Verteilungsfunktion).

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

Dies ist eine verlustbehaftete Operation und nicht umkehrbar.

5.1.3 Filterung im Ortsraum (Konvolution)

Hierbei wird der neue Wert eines Pixels unter Berücksichtigung seiner Nachbarschaft berechnet (lokaler Kontext).

Faltung (Convolution)

Die Faltung eines Bildes f mit einer Filtermaske w (Kernel) ist definiert als:

$$(f * w)(m,n) = \sum_{i=-k/2}^{k/2} \sum_{j=-l/2}^{l/2} w(i,j) \cdot f(m+i, n+j)$$

Tiefpass-Filter (Glättung) Dienen der Rauschunterdrückung und Weichzeichnung (Blurring).

- **Eigenschaften:** Koeffizienten sind positiv und summieren sich meist zu 1 (Normalisierung), um die Helligkeit zu erhalten.
- **Box-Filter (Mittelwert):** Alle Koeffizienten sind gleich (z.B. alle $\frac{1}{9}$ bei einer 3×3 Matrix). Nachteil: Erzeugt Artefakte (rechteckige Unschärfe).
- **Gauß-Filter:** Approximation der Gauß-Glocke.

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

  Gewichtet zentrale Pixel stärker als äußere. Besser als Box-Filter, da natürliche Glättung. Kann durch Binomialfilter approximiert werden.

Nicht-Lineare Filter

- **Median-Filter:** Ersetzt den Pixelwert durch den **Median** der Nachbarschaft.
- **Vorteil:** Entfernt "Salt-and-Pepper"-Rauschen (Ausreißer) extrem effektiv und **erhält Kanten** (im Gegensatz zu Tiefpassfiltern, die Kanten verschmieren).

Hochpass-Filter (Kantenextraktion) Dienen dem Finden von Kanten (Hervorhebung von hohen Frequenzen/Änderungen).

- **Eigenschaften:** Koeffizienten sind positiv und negativ, Summe ist meist 0. Ergebnis kann negative Werte enthalten.
- **Gradienten (1. Ableitung):** Reagieren auf Rampen und Stufen (Kanten).
- **Laplace-Filter (2. Ableitung):** Isotrop (richtungsunabhängig).

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

  Reagiert stark auf feine Details und Rauschen ("Double Response" an Kanten: Nulldurchgang).
- **Laplacian of Gaussian (LoG):** Auch "Mexican Hat" oder Sombrero-Filter. Kombination aus Glättung (Gauß) zur Rauschunterdrückung und Laplace zur Kantenfindung.

5.2 Digitale Bildverarbeitung im Frequenzraum

Bilder können durch Transformationen (z.B. Fourier) in Frequenzanteile zerlegt werden.

- **Tiefe Frequenzen:** Grobe Strukturen, Flächen, homogener Hintergrund. (Im Zentrum des Spektrums).
- **Hohe Frequenzen:** Feine Details, Kanten, Rauschen. (Außen im Spektrum).

Faltungssatz

Eine Faltung im Ortsraum entspricht einer Multiplikation im Frequenzraum (und umgekehrt).

$$f * h = F^{-1}(F(f) \cdot F(h))$$

5.2.1 Ablauf der Frequenzfilterung

1. Transformation des Eingabebildes f mittels FFT (Fast Fourier Transform) $\rightarrow F(u, v)$. 2. Multiplikation mit der Filterfunktion $H(u, v)$. 3. Rücktransformation mittels inverser FFT $\rightarrow$ gefiltertes Bild $g(x, y)$.

5.2.2 Filtertypen im Frequenzraum

- **Idealer Tiefpass:** Schneidet alle Frequenzen oberhalb D_0 ab (Rechteckfunktion). Führt im Ortsraum zu **Ringing-Artefakten** (Wellen um Kanten), da die FT einer Rechteckfunktion eine Sinc-Funktion ist.
- **Gauß-Tiefpass:** Weicher Übergang. Kein Ringing, da FT einer Gauß-Funktion wieder eine Gauß-Funktion ist.

$$G(\omega) = e^{-\frac{\sigma^2 \omega^2}{2}}$$

- **Hochpass:** Kehrt das Prinzip um (Zentrum wird auf 0 gesetzt, hohe Frequenzen bleiben).

Aliasing: Tritt auf, wenn Abtastrate zu gering (Unterabtastung) oder im Frequenzraum Frequenzen abgeschnitten werden ("Leakage").

5.3 Bildkompression

Ziel: Reduktion der Datenmenge durch Eliminierung von Redundanzen (räumlich, zeitlich, spektral) und Irrelevanzen (nicht wahrnehmbare Informationen).

5.3.1 Klassifikation

- **Verlustfrei (Lossless):** Original kann exakt wiederhergestellt werden. (z.B. Huffman, RLE, LZW $\rightarrow$ PNG, GIF, TIFF).
- **Verlustbehaftet (Lossy):** Informationen gehen verloren, höhere Kompressionsraten. Nutzt Wahrnehmungsmodelle des Menschen (z.B. JPEG, MPEG).

5.3.2 JPEG (Joint Photographic Experts Group)

Ein Standard für verlustbehaftete Kompression, optimiert für natürliche Fotos (fließende Übergänge).

Die 5 Schritte des JPEG-Baseline-Codecs:

1. **Farbraum-Transformation:** Umwandlung von RGB in YC_bC_r.
   - Y: Luminanz (Helligkeit).
   - C_b, C_r: Chrominanz (Farbdifferenzen Blau/Rot).
2. **Farb-Subsampling (Downsampling):** Da das menschliche Auge Helligkeit besser auflöst als Farbe, werden C_b und C_r räumlich reduziert (z.B. 4:2:0 oder 2x2 Mittelung). Y bleibt voll erhalten.
3. **Block-Aufteilung & DCT (Diskrete Kosinustransformation):** Das Bild wird in 8×8 Blöcke unterteilt. Jeder Block wird mittels DCT in den Frequenzraum transformiert.
   - Ergebnis: 1 DC-Koeffizient (Gleichanteil/Grundhelligkeit) und 63 AC-Koeffizienten (Wechselanteile).
   - Energiekonzentration: Meist sind nur wenige Koeffizienten (niedrige Frequenzen) signifikant groß.

4. **Quantisierung (Der eigentliche Verlust):**

$$F^Q = \text{Round}(F_{ij}/Q_{ij})$$

Die DCT-Koeffizienten werden durch eine Quantisierungsmatrix Q geteilt und gerundet.

- Hohe Frequenzen werden stark quantisiert (durch große Werte in Q), oft zu Null.
- Steuert die Qualität vs. Kompressionsrate.

5. **Kodierung (Entropy Coding):**
   - **Zig-Zag-Scan**: Sortiert die Koeffizienten von niedrigen zu hohen Frequenzen, um lange Ketten von Nullen zu erzeugen.
   - **RLE (Run-Length Encoding):** Kodiert die Nullen-Ketten effizient.
   - **Huffman-Kodierung:** Weist häufigen Werten kurze Bitfolgen zu.

6 Bildrestaurierung und Deblurring

Diese Zusammenfassung behandelt die Methoden zur Wiederherstellung von Bildern, die durch Unschärfe (Blurring) und Rauschen (Noise) degradiert wurden. Wir bewegen uns von einfachen Filtern im Frequenzraum hin zu komplexen iterativen Verfahren mittels partieller Differentialgleichungen.

6.1 Problemstellung und Grundlagen

6.1.1 Das Bildmodell

Ein aufgenommenes Bild ist selten perfekt. Es wird mathematisch oft als eine Faltung des Originalbildes mit einem Kern (Blur-Kernel) plus Rauschen modelliert.

Das Degradationsmodell

Sei f das Originalbild, a der Faltungskern (z.B. eine Gaußglocke für Unschärfe) und n das additive Rauschen. Das beobachtete Bild g ist:

$$g = a(f) + n$$

Im Fourierraum (Faltung wird zur Multiplikation) gilt ohne Rauschen:

$$G = A \cdot F$$

6.1.2 Der naive Ansatz: Inverser Filter

Theoretisch könnte man das Originalbild F einfach durch Division zurückgewinnen:

$$F = \frac{G}{A}$$

Problem: Dieser Ansatz ist in der Praxis instabil (Cheating).

1. Der Kernel A (oft ein Tiefpassfilter wie Gauß) hat bei hohen Frequenzen Werte nahe 0.
2. Division durch sehr kleine Zahlen führt zu riesigen Werten.
3. Da G auch Rauschen enthält ($G = A \cdot F + N$), wird das Rauschen N durch die Division mit kleinen Werten von A massiv verstärkt.
4. Das Ergebnis enthält oft komplexe Zahlen und ist unbrauchbar.

6.1.3 Exkurs: Korrekt gestellte Probleme (Hadamard)

Ein mathematisches Problem ist **korrekt gestellt** (well-posed), wenn:

1. Eine Lösung existiert.
2. Die Lösung eindeutig ist.
3. Die Lösung stetig von den Daten abhängt (Stabilität).

Deblurring ist ein nicht korrekt gestelltes Problem (ill-posed). Kleine Änderungen im Input (Rauschen) führen zu riesigen Änderungen im Output (Ableitungen/Schärfung). Daher ist **Regularisierung** notwendig: Wir müssen zusätzliches Wissen (z.B. Glattheit) einbringen, um eine stabile Lösung zu finden.

6.2 Einschrittverfahren (Frequenzraum)

6.3 Der Wiener Filter

Der Wiener Filter ist die Standardlösung für das Problem des Inversen Filters. Er versucht, den Fehler zwischen dem geschätzten und dem echten Bild statistisch zu minimieren.

Wiener Filter

Anstatt einfach durch A zu teilen, wird der Ausdruck regularisiert:

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

Dabei ist:
- A^*: Die komplex konjugierte Matrix von A.
- $|A|^2 = A^*A$: Das Leistungsdichtespektrum des Kernels.
- R: Ein Parameter, der das Signal-Rausch-Verhältnis (SNR) repräsentiert.

6.3.1 Funktionsweise und Interpretation von R

Der Term R^2 im Nenner verhindert die Division durch Null.

- R **ist groß (viel Rauschen/unsicheres Signal):** Der Filter wirkt wie ein **Tiefpass**. Er glättet das Bild, um das Rauschen zu unterdrücken, verliert aber Details. Der Term $|A|^2$ wird gegenüber R^2 vernachlässigbar.
- R **ist klein (wenig Rauschen):** Der Filter nähert sich dem **inversen Filter** an ($R \approx 0 \Rightarrow \frac{A^*}{|A|^2} = \frac{1}{A}$). Er schärft das Bild stark (Hochpass), verstärkt aber Rauschen, falls vorhanden.
- R **optimal gewählt:** Der Filter agiert als **Bandpass**. Er entfernt Rauschen in Frequenzen, wo das Signal schwach ist, und invertiert den Blur dort, wo das Signal stark ist.

Vorteil: Schnell und einfach zu implementieren (ein globaler Filter).
Nachteil: Ein globaler Parameter R für das ganze Bild. Keine lokale Anpassung an Kanten vs. flache Regionen.

6.4 Mehrschrittverfahren (Iterative Methoden)

Um lokale Verbesserungen zu ermöglichen, nutzen wir iterative Verfahren im Ortsraum (statt Frequenzraum).

6.5 Scale-Space und Diffusionsgleichung

Deblurring kann als Umkehrung von Blurring verstanden werden.

- **Blurring:** Entspricht physikalisch der Wärmeleitungsgleichung (Heat Equation) oder **linearen Diffusion**.

  $$\frac{\partial L}{\partial t} = \Delta L = L_{xx} + L_{yy}$$

  Die Lösung dieser Gleichung über die Zeit t ist die Faltung des Bildes mit einem Gauß-Kern. Das Bild wird unschärfer.

- **Deblurring (Schärfen):** Man kann versuchen, den Laplace-Operator (ΔL) zu subtrahieren (ähnlich „Unsharp Masking“).

  $$L_{neu} = L_{alt} - t \cdot \Delta L$$

  Fügt man höhere Ableitungen (Taylor-Reihe) hinzu, kann man das Bild theoretisch rekonstruieren, aber das Verfahren ist instabil (Rauschen wird verstärkt).

6.6 Nichtlineare Diffusion (Perona-Malik)

Die lineare Diffusion ($\partial_t L = \Delta L$) glättet alles gleichmäßig – Rauschen *und* Kanten. Das Ziel ist aber: **Rauschen glätten, Kanten erhalten (oder schärfen).**

Dazu führen wir einen variablen Diffusionskoeffizienten c ein.

Perona-Malik Gleichung

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

6.6.1 Der Koeffizient c

Die Funktion $c(\cdot)$ steuert die Diffusion abhängig von der lokalen Kantenstärke (Gradientenbetrag $|\nabla L|$).

- **Flache Regionen (kleiner Gradient):** $c \approx 1$. Wir wollen maximale Diffusion zur Rauschunterdrückung.
- **Kanten (großer Gradient):** $c \approx 0$. Die Diffusion wird gestoppt, um die Kante nicht zu verwischen.

Ein typisches Modell für c:

$$c(|\nabla L|^2) = \frac{1}{1 + \frac{|\nabla L|^2}{k^2}} \quad \text{oder} \quad c = e^{-\frac{|\nabla L|^2}{k^2}}$$

6.6.2 Der Parameter k (Contrast Parameter)

k fungiert als Schwellenwert ("Soft-Threshold"):

- Gradienten $< k$: Werden als Rauschen interpretiert $\rightarrow$ **Glättung (Blurring)**.
- Gradienten $> k$: Werden als Kanten interpretiert $\rightarrow$ **Schärfung (Deblurring/Inverse Diffusion)**.

Dies führt zu einer Kantenverstärkung, da an der Kante Material „wegdiffundiert" und sich aufstaut (Schockfilter-Effekt).

6.6.3 Probleme (Perona-Malik Paradox)

Das Modell ist mathematisch „ill-posed“ (inverser Diffusionsanteil ist instabil).

- Es entsteht ein „Staircasing“-Effekt (Bild wirkt blockig).
- Man benötigt eine **Stoppzeit**, da das Bild sonst in triviale Zustände degeneriert.
- Regularisierung ist nötig (z.B. Glättung des Gradienten vor der Berechnung von c).

6.7 Variationsmethoden: Totale Variation (TV)

Anstatt eine PDE (Partielle Differentialgleichung) ad-hoc zu definieren (wie Perona-Malik), definieren wir zuerst eine **Energie**, die minimiert werden soll. Das Minimum dieser Energie ist unser gewünschtes Bild.

Energie-Funktional

Wir suchen ein Bild L, das die Energie $E(L)$ minimiert:

$$E(L) = \underbrace{\int |\nabla L| \, dxdy}_{\text{Regularisierung}} + \underbrace{\frac{\lambda}{2} \int (g - L)^2 \, dxdy}_{\text{Datentreue}}$$

- **Datentreue-Term (Data Fidelity):** $\lambda(g - L)^2$. Zwingt das Ergebnis L dazu, nahe am verrauschten Input-Bild g zu bleiben.
- **Regularisierungs-Term:** $\int |\nabla L|$. Dies ist die **Totale Variation (TV)**.

6.7.1 Warum Totale Variation?

Im Gegensatz zur Tikhonov-Regularisierung ($\int |\nabla L|^2$), die quadratisch bestraft und zu glatten Kanten führt (Blurring), bestraft die TV-Norm ($\int |\nabla L|$) lineare Anstiege weniger stark.

- TV erlaubt **sprunghafte Änderungen** (Kanten) im Bild.
- Es favorisiert stückweise konstante Bilder („Cartoon-Look“).
- Rauschen (kleine Oszillationen) erhöht die TV enorm und wird daher entfernt.
- Kanten (einmaliger Sprung) erhöhen die TV nur moderat und bleiben erhalten.

6.7.2 Anwendungen

- **Denoising:** Entfernt Rauschen extrem effektiv ohne Kanten zu verwischen.
- **Inpainting:** Kann fehlende Bildteile (Löcher, Schriftzüge) rekonstruieren, indem die Kanten in den fehlenden Bereich „hineinpropagiert“ werden (Level-Lines-Fortsetzung).

6.8 Zusammenfassung der Verfahren

Methode	Vorteil	Nachteil
Inverser Filter	Theoretisch exakt ohne Rauschen	Extrem instabil bei Rauschen (nicht nutzbar)
Wiener Filter	Stabil, berücksichtigt SNR, schnell	Globaler Filter, verwischt Kanten oder lässt Rauschen übrig
Lineare Diffusion (Heat Eq.)	Entfernt Rauschen sehr gut	Verwische alle Kanten (Tiefpass)
Perona-Malik	Kantenerhaltend, lokal adaptiv	Instabil (Paradox), benötigt Stoppzeit, Staircasing
Totale Variation	Exzellente Kantenerhaltung, keine Stoppzeit nötig	Bevorzugt stückweise konstante Areale (Cartoon-Effekt)

7 Grafikpipeline

7.1 Computer Vision vs. Computer Grafik

Das Kernproblem kann als Inversion betrachtet werden:

- **Computer Vision (CV)**: Prozess von *Rechts nach Links*. Aus Bildern (Pixelrastern) werden Informationen (Objekte, Tiefe) extrahiert. Ziel: Verstehen, was zu sehen ist ("Inverse Grafik").
- **Computer Grafik (CG)**: Prozess von *Links nach Rechts*. Aus abstrakten Daten (Zahlen, Modellen) werden Bilder erzeugt. Ziel: Realistische oder stilisierte Darstellung von Informationen.

7.2 Computing Paradigmen

Die Entwicklung der Hardware beeinflusst die Interaktion und Darstellung:

1. **Mainframe / Large Scale:** Zentralisierte Rechenleistung, Zugriff über Terminals.
2. **Personal Computing (Desktop):** Eigene Rechenleistung, GUI-basiert (WIMP: Windows, Icons, Menus, Pointer).
3. **Networked Computing:** Vernetzung von Systemen (WAN, LAN, Internet).
4. **Mobile Computing:** Smartphones, Tablets. Einschränkungen bei Leistung und Batterie, aber hohe Portabilität.
5. **Collaborative Computing:** Gemeinsames Arbeiten (z.B. Multi-Touch Tables).
6. **Virtual Reality (VR) & Augmented Reality (AR):**

   Virtual Reality (VR): Immersion in eine vollständig virtuelle Welt (z.B. CAVE, Head Mounted Displays).

   Augmented Reality (AR): Nahtlose Integration virtueller Objekte in die reale Welt zur Erweiterung der Wahrnehmung (z.B. HUDs, Smart Glasses, Smartphone-Kamera). Wichtig: Synchronisation mit der echten Welt.
7. **Ubiquitous / Invisible Computing:** Computer verschwinden in der Umgebung (Ambient Intelligence, Wearables, IoT).

7.3 Virtuelle Charaktere und Wahrnehmung

Bei der Darstellung menschenähnlicher Avatare tritt ein psychologisches Phänomen auf:

Uncanny Valley

Das **Uncanny Valley** (Akzeptanzlücke) beschreibt den Effekt, dass die Akzeptanz eines künstlichen Charakters schlagartig abfällt, wenn dieser sehr menschenähnlich ist, aber nicht perfekt wirkt (z.B. Zombies, Leichen). Stilisierte Charaktere (Comic) werden oft besser akzeptiert als fast-realistische.

7.4 Die 3D-Grafikpipeline

Die Grafikpipeline beschreibt den Weg von der geometrischen Beschreibung einer Szene bis zum fertigen Pixelbild auf dem Monitor. Sie wird typischerweise in vier Hauptstufen unterteilt.

1. **Anwendung (CPU):** Modellierung, Eingabe, Szenegraph.
2. **Geometrieverarbeitung (GPU):** Transformationen, Beleuchtung, Clipping.
3. **Rasterisierung (GPU):** Umwandlung von Primitiven in Fragmente (Pixelkandidaten).

4. **Ausgabe:** Darstellung, Speicherung.

7.4.1 Stufe 1: Anwendung (Application)

Hier finden Berechnungen statt, die nicht fest in der Hardware verdrahtet sind (Software-Seite).

- **Eingabe:** Verarbeitung von Nutzereingaben (Maus, Tastatur, Tracking).
- **Modellierung:** Erstellung von 3D-Modellen (aus CT-Daten, Laserscans oder manuellem Design).
- **Primitive:** Die Grundbausteine der Grafik sind **Punkte**, **Linien** und **Dreiecke** (Polygone). Dreiecke sind bevorzugt, da sie immer planar (eben) sind.

Räumliche Datenstrukturen Um Berechnungen zu beschleunigen (z.B. Kollisionserkennung oder Sichtbarkeitstest), werden Hüllkörper und Raumunterteilungen genutzt.

- **Hüllkörper (Bounding Volumes):** Einfache geometrische Formen, die ein komplexes Objekt umschließen.
  - *Kugel (Sphere):* Einfache Distanzberechnung, oft nicht passgenau.
  - *AABB (Axis Aligned Bounding Box):* Achsenparallel, einfach zu berechnen, rotiert nicht mit dem Objekt.
  - *OBB (Oriented Bounding Box):* Passt sich der Rotation an, teurer in der Berechnung.
- **Raumunterteilung (Space Partitioning):**
  - **Gitter (Grids):** Regelmäßige Unterteilung. Problem: Speicheraufwendig bei leeren Räumen.
  - **Quadtree (2D) / Octree (3D):** Hierarchische Unterteilung. Ein Quadrat/Würfel wird bei Bedarf in 4 (bzw. 8) kleinere Einheiten geteilt. Effizient für ungleichmäßig verteilte Szenen.
  - **BSP-Tree (Binary Space Partitioning):** Der Raum wird durch Ebenen rekursiv in zwei Hälften geteilt. Wichtig für den *Painters Algorithm*, da die Zeichenreihenfolge (Vorn/Hinten) vorberechnet werden kann.

7.4.2 Stufe 2: Geometrieverarbeitung (Geometry Processing)

In dieser Stufe werden die Vertizes (Eckpunkte) der Primitive transformiert und beleuchtet.

Transformationen Objekte müssen vom lokalen Koordinatensystem in das Weltkoordinatensystem und schließlich in das Sichtsystem der Kamera überführt werden. Dies geschieht durch affine Transformationen (Translation, Rotation, Skalierung, Scherung) mittels Matrizenmultiplikation.

Beleuchtung (Illumination) Ziel ist die Simulation der Interaktion von Licht und Material. Das **Phong-Beleuchtungsmodell** ist ein lokales Modell (berücksichtigt nur direkte Lichtquellen, keine Reflexionen zwischen Objekten).

Die Lichtintensität I_{total} an einem Punkt setzt sich zusammen aus:

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

1. **Ambientes Licht (I_{amb}):** Grundhelligkeit der Szene (indirekte Beleuchtung simuliert durch Konstante).

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

2. **Diffuse Reflexion (I_{diff}):** Matte Oberflächen (Lambert-Reflexion). Helligkeit hängt vom Winkel zwischen Lichtvektor L und Oberflächennormale N ab.

$$I_{diff} = k_{diff} \cdot C_{light} \cdot (\vec{N} \cdot \vec{L})$$

3. **Spiegelnde Reflexion (I_{spec}):** Glanzpunkte (Highlights). Hängt vom Winkel zwischen Reflexionsvektor R und Blickvektor V ab. Der Exponent m bestimmt die Rauhigkeit (je höher m, desto kleiner und schärfer der Glanzpunkt).

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

Shading-Verfahren (Interpolation) Wie wird die Beleuchtung auf das gesamte Polygon angewendet?

- **Flat Shading**: Ein Normalenvektor für das ganze Polygon. Eine Farbe pro Polygon. Sieht "kantig" aus.
- **Gouraud Shading**: Beleuchtung wird an den Eckpunkten (Vertizes) berechnet. Die resultierenden *Farbwerte* werden über das Polygon interpoliert. Glatter Verlauf, aber Glanzpunkte können verloren gehen, wenn sie in der Mitte des Polygons liegen.
- **Phong Shading**: Die *Normalenvektoren* werden über das Polygon interpoliert. Die Beleuchtungsgleichung wird für jedes Pixel berechnet. Bestes Ergebnis, aber rechenaufwendigsten.

Clipping & Culling Optimierungsschritte, um nicht sichtbare Geometrie frühzeitig zu verwerfen.

- **Clipping**: Abschneiden von Geometrie, die aus dem Sichtvolumen (Frustum) herausragt.
- **Backface Culling**: Entfernen von Polygonen, die von der Kamera wegzeigen (Rückseiten). *Test:* Skalarprodukt aus Blickrichtung s und Normale n. Wenn $n \cdot s > 0$, ist es eine Rückseite (bei entsprechender Vektor-Definition).

7.4.3 Stufe 3: Rasterisierung

Umwandlung der kontinuierlichen geometrischen Primitive in diskrete Pixel (Fragmente).

Linien-Algorithmen Der **Bresenham-Algorithmus** (1965) ermöglicht das Zeichnen von Linien unter ausschließlicher Verwendung von Integer-Arithmetik (Ganzzahlen).

- *Idee:* Entscheidung, ob das nächste Pixel "rechts" oder "rechts oben" gesetzt wird, basierend auf einem Fehlerterm, der akkumuliert wird.
- Vermeidet langsame Gleitkommaoperationen.

Bresenham-Algorithmus (für 1. Oktant)

Annahme: Linie im ersten Oktanten, d.h. $0 < \Delta y \leq \Delta x$ (Steigung zwischen 0 und 1).
Gegeben: Startpunkt (x_{start}, y_{start}) und Endpunkt (x_{end}, y_{end}).
Vorbereitung:

- Berechne Differenzen: $dx = x_{end} - x_{start}$ und $dy = y_{end} - y_{start}$
- Initialisiere Startpixel: $x = x_{start}$, $y = y_{start}$
- Initialisiere Fehlerterm: $fehler = dx/2$ (Integer-Division)
- Setze erstes Pixel (x, y)

Iteration (Schleife solange $x < x_{end}$):

1. Schritt in die *schnelle Richtung* (hier x):
$$x = x + 1$$
2. Fehlerterm aktualisieren (Idealgerade weicht ab):
$$fehler = fehler - dy$$
3. **Entscheidung:** Ist der Fehler zu groß geworden ($fehler < 0$)?
   - **JA:** Wir müssen auch in die *langsame Richtung* (y) gehen, um der Linie zu folgen.
   $$y = y + 1$$
   $$fehler = fehler + dx$$
   (Fehlerterm korrigieren)
   - **NEIN:** Wir bleiben auf der gleichen y-Höhe.
4. Setze Pixel (x, y)

Polygon-Rasterisierung **Scanline-Algorithmus**: Eine horizontale Linie wandert zeilenweise über das Bild. Schnittpunkte mit Polygonkanten werden berechnet und sortiert. Pixel zwischen Paaren von Schnittpunkten werden gefüllt (Paritätsregel: Umschalten zwischen Malen/Nicht-Malen bei Kantenübertritt).

Verdeckungsrechnung (Visibility) Wie wird bestimmt, welches Objekt vorne liegt?

Z-Buffer Algorithmus

Für jedes Pixel wird neben der Farbe auch ein Tiefenwert (z-Wert) gespeichert.

- Initialisierung: Bildspeicher = Hintergrundfarbe, Z-Buffer = unendlich (max Tiefe).
- Für jedes Pixel eines neuen Polygons: Berechne Tiefe z_{neu}.
- *Test:* Ist $z_{neu} < z_{gespeichert}$?
- *Ja:* Schreibe Farbe in Bildspeicher, aktualisiere $z_{gespeichert} = z_{neu}$.
- *Nein:* Verwerfe Pixel (verdeckt).

Vorteile: Reihenfolge der Objekte egal, einfache Hardware-Implementierung.
Nachteile: Speicherbedarf, Transparenz schwierig, Z-Fighting (Genauigkeitsprobleme).

Alternativ: **Painters Algorithm**. Sortiere Polygone nach Tiefe (weit weg → nah) und zeichne sie übereinander. Problem: Zyklische Überlappungen und komplexe Sortierung ($O(n^2)$).

7.4.4 Stufe 4: Ausgabe

Finales Schreiben in den Framebuffer, Darstellung auf CRT, LCD, Beamer oder Speicherung in Dateien.

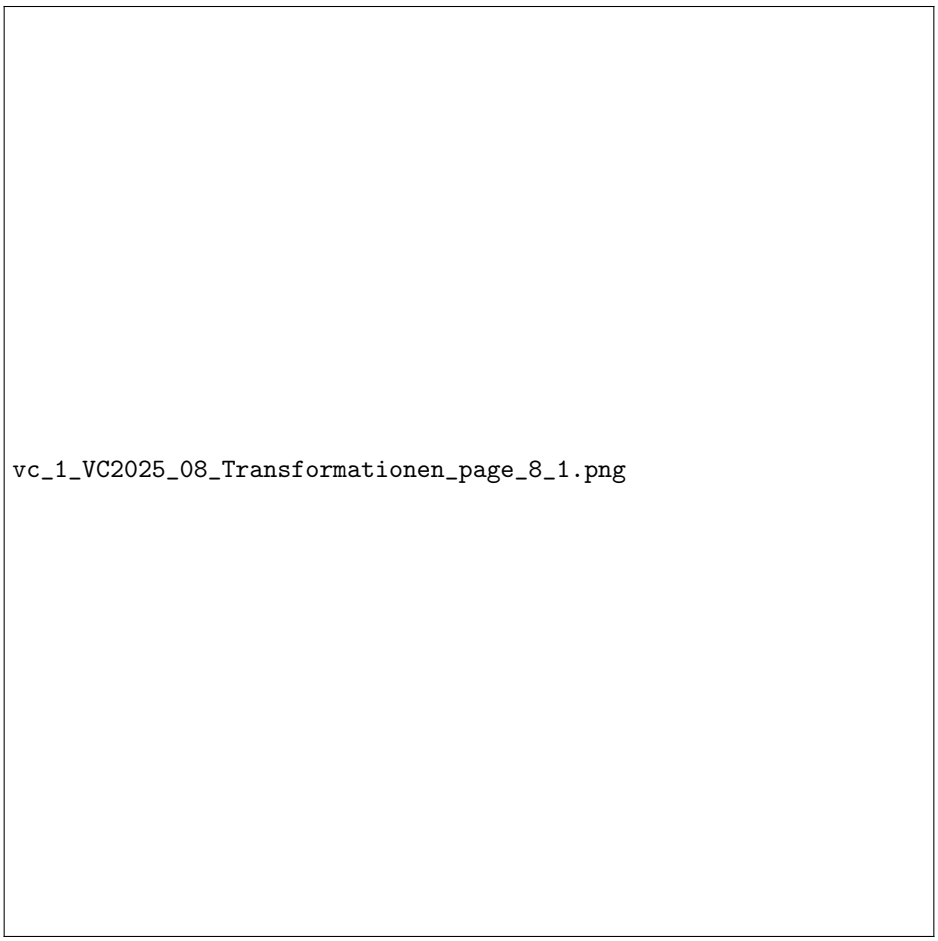
8 Transformationen

In der Computergrafik durchlaufen 3D-Daten verschiedene Stufen (die Grafikpipeline), um schließlich als 2D-Bild auf dem Bildschirm zu erscheinen. Dabei spielen **Koordinatensysteme** und **Transformationen** eine zentrale Rolle.

8.1 Koordinatensysteme in der Pipeline

Um ein Bild zu erzeugen, müssen Objekte durch verschiedene Koordinatenräume transformiert werden. Die Reihenfolge ist essenziell:

1. **Objektkoordinaten (Object Coordinates)**: Lokales Koordinatensystem eines einzelnen 3D-Modells (z.B. Mittelpunkt einer Kugel im Ursprung).
2. **Weltkoordinaten (World Coordinates)**: Beschreiben die gesamte Szene. Alle Objekte werden hier relativ zueinander platziert.
3. **Augenkoordinaten (Eye/Camera Coordinates)**: Die Szene aus der Sicht der Kamera. Die Kamera liegt im Ursprung, Blickrichtung oft entlang der negativen Z-Achse.
4. **Projektionskoordinaten (Clip Coordinates)**: Nach Anwendung der Projektion (perspektivisch oder parallel). Hier findet das *Clipping* statt.
5. **Normierte Gerätekoordinaten (Normalized Device Coordinates – NDC)**: Koordinatenbereich meist von $[-1, 1]$ oder $[0, 1]$.
6. **Bildschirmkoordinaten (Window Coordinates)**: Tatsächliche Pixelpositionen im Ausgabefenster.



8.1.1 Transformationstypen

- **Modelling Transformations:** Anordnung von Objekten in der Welt (Rotation, Translation, Skalierung).
- **Viewing Transformations:** Positionierung und Ausrichtung der Kamera.
- **Projection Transformations:** Definition des Sichtvolumens (Viewing Volume) und Projektion auf 2D.
- **Viewport Transformations:** Mapping auf die Pixelauflösung des Bildschirms.

8.2 Mathematische Grundlagen: Homogene Koordinaten

Um Transformationen effizient zu berechnen, nutzen wir Matrizen. Ein Problem im herkömmlichen euklidischen Raum (R^3) ist, dass lineare Abbildungen (Rotation, Skalierung) den Ursprung fix lassen müssen. Eine **Translation** (Verschiebung) ist jedoch eine affine Abbildung, keine lineare, und kann nicht durch eine 3×3-Matrix dargestellt werden.

Lösung: Homogene Koordinaten

Wir erweitern den n-dimensionalen Raum auf $n+1$ Dimensionen. Ein Punkt $(x, y, z)^T$ im 3D-Raum wird zu $(x, y, z, w)^T$ im homogenen Raum (meist ist $w = 1$).
Die Rückumwandlung erfolgt durch Division durch w:

$$\begin{pmatrix} x \\ y \\ z \\ w \end{pmatrix} \rightarrow \begin{pmatrix} x/w \\ y/w \\ z/w \end{pmatrix}$$

Vorteile:

- Einheitliche Darstellung aller affinen Transformationen (inkl. Translation) als 4×4-Matrizen.
- Hintereinanderausführung von Transformationen entspricht einfacher Matrixmultiplikation.
- Ermöglicht perspektivische Projektionen (durch Manipulation von w).

8.3 Affine Abbildungen

Eine **Affine Abbildung** setzt sich aus einem linearen Teil (Matrix A) und einer Translation (Vektor b) zusammen: $\Phi(v) = A \cdot v + b$.

Eigenschaften affiner Abbildungen

1. Geraden werden auf Geraden abgebildet.
2. Parallelität von Linien und Ebenen bleibt erhalten.
3. Verhältnisse von Längen, Flächen und Volumina bleiben erhalten (relative Abstände).

In homogenen Koordinaten sieht die allgemeine Matrixform so aus:

$$M = \begin{pmatrix} a_{11} & a_{12} & a_{13} & t_x \\ a_{21} & a_{22} & a_{23} & t_y \\ a_{31} & a_{32} & a_{33} & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Der 3×3-Block oben links (a_{ij}) steuert Rotation, Skalierung und Scherung. Die letzte Spalte (t) steuert die Translation.

8.3.1 Translation (Verschiebung)

Verschiebung eines Punktes um den Vektor $d = (x_0, y_0, z_0)^T$. Der lineare Teil ist die Einheitsmatrix.

$$T(d) = \begin{pmatrix} 1 & 0 & 0 & x_0 \\ 0 & 1 & 0 & y_0 \\ 0 & 0 & 1 & z_0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

8.3.2 Skalierung

Skalierung verändert die Größe entlang der Achsen. Sie wird durch eine Diagonalmatrix beschrieben. Mit den Faktoren s_1, s_2, s_3 für die x-, y- und z-Achse:

$$S(s_1, s_2, s_3) = \begin{pmatrix} s_1 & 0 & 0 & 0 \\ 0 & s_2 & 0 & 0 \\ 0 & 0 & s_3 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Ist $s_1 = s_2 = s_3 = s$, spricht man von einer *uniformen Skalierung.*

8.3.3 Scherung (Shearing)

Verzerrt das Objekt, indem Koordinaten in Abhängigkeit von anderen Koordinaten verschoben werden. Beispiel einer Scherung entlang der x-Achse in Abhängigkeit von y:

$$SH_{xy} = \begin{pmatrix} 1 & s & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

8.3.4 Rotation

Wir verwenden ein **Rechtssystem** (Rechte-Hand-Regel). Positive Drehwinkel drehen gegen den Uhrzeigersinn, wenn man entlang der positiven Rotationsachse auf den Ursprung blickt.

Rotation um die Z-Achse (Winkel α):

$$R_z(\alpha) = \begin{pmatrix} \cos\alpha & -\sin\alpha & 0 & 0 \\ \sin\alpha & \cos\alpha & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Rotation um die X-Achse (Winkel α):

$$R_x(\alpha) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha & 0 \\ 0 & \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Rotation um die Y-Achse (Winkel α):

$$R_y(\alpha) = \begin{pmatrix} \cos\alpha & 0 & \sin\alpha & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\alpha & 0 & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Hinweis: Bei der Y-Rotation scheinen die Vorzeichen des Sinus im Vergleich zu X und Z vertauscht. Das liegt an der zyklischen Permutation der Achsen ($x \to y \to z \to x$).

8.4 Komplexe Transformationen & Komposition

8.4.1 Matrix-Multiplikation und Reihenfolge

Die Matrix-Multiplikation ist **nicht kommutativ** ($A \cdot B \neq B \cdot A$). Das bedeutet: **Die Reihenfolge der Transformationen ist wichtig!** Eine Rotation gefolgt von einer Translation ist etwas anderes als eine Translation gefolgt von einer Rotation.

Mathematisch werden Transformationen von rechts nach links auf den Punktvektor p angewendet:

$$p' = M_n \cdot \cdots \cdot M_2 \cdot M_1 \cdot p$$

Hier wird zuerst M_1, dann M_2, usw. ausgeführt.

8.4.2 Rotation um beliebigen Punkt / Achse

Die Standard-Rotationsmatrizen drehen immer um den Ursprung. Um ein Objekt um einen beliebigen Punkt P_f (Fixed Point) zu drehen:

1. Verschiebe das Objekt so, dass P_f im Ursprung liegt ($T(-P_f)$).
2. Rotiere um den Ursprung (R).
3. Verschiebe das Objekt zurück ($T(P_f)$).

Gesamtmatrix: $M = T(P_f) \cdot R \cdot T(-P_f)$.

Rotation um eine beliebige Achse r (durch den Ursprung): Man konstruiert eine orthonormale Basis (r, s, t), wobei r die normierte Rotationsachse ist.

1. Bilde eine Basiswechselmatrix R_{basis}, die die Achse r auf eine Hauptachse (z.B. x-Achse) abbildet. Dies geschieht, indem man die Basisvektoren in die Zeilen der Matrix schreibt.
2. Rotiere um diese Hauptachse (R_α^x).
3. Wende die inverse Basiswechselmatrix an ($R_{basis}^{-1} = R_{basis}^T$).

Dies ist oft effizienter als das Ausrechnen über Euler-Winkel.

8.5 Projektionen

Projektionen bilden den 3D-Raum auf eine 2D-Ebene ab. Dabei gehen Informationen (Tiefe) verloren.

8.5.1 Eigenschaften projektiver Abbildungen

Projektive Abbildungen sind eine allgemeinere Klasse von Transformationen als die affinen Abbildungen. Sie können ebenfalls durch homogene 4×4-Matrizen beschrieben werden, jedoch ist die letzte Zeile der Matrix nicht mehr zwingend $(0, 0, 0, 1)$.

Eigenschaften im Vergleich

Im Gegensatz zu affinen Abbildungen gelten bei projektiven Abbildungen folgende Besonderheiten:

- **Geradentreue:** Geraden werden weiterhin auf Geraden abgebildet.
- **Schnittpunkte:** Schnitte von Geraden bleiben erhalten.
- **Reihenfolge:** Die Reihenfolge von Punkten auf projektiven Geraden bleibt erhalten.
- **Parallelität geht verloren:** Parallele Geraden schneiden sich nach der Transformation oft in sogenannten **Fluchtpunkten**.
- **Winkel und Längen:** Winkel werden verändert und Längenverhältnisse bleiben *nicht* erhalten. Rechtecke werden zu allgemeinen Vierecken verzerrt.

Fluchtpunkte: Bei der perspektivischen Projektion treffen sich alle Geraden, die im 3D-Raum parallel sind (aber nicht parallel zur Bildebene liegen), in einem Punkt auf der Bildebene, dem Fluchtpunkt. Je nach Ausrichtung des Objekts zur Bildebene spricht man von einer 1-, 2- oder 3-Punkt-Perspektive.

8.5.2 Vergleich: Parallel vs. Perspektivisch

Parallelprojektion (Orthografisch)	**Perspektivische Projektion**
Projektionsstrahlen sind parallel.	Projektionsstrahlen treffen sich im Augpunkt (Center of Projection, COP).
Keine perspektivische Verkürzung (Größe unabhängig von Distanz).	Objekte, die weiter weg sind, erscheinen kleiner.
Winkel bleiben meist nicht erhalten, aber parallele Linien bleiben parallel.	Parallele Linien schneiden sich in Fluchtpunkten (außer sie sind parallel zur Bildebene).
Anwendung: Technische Zeichnungen, Medizin, Architektur (Grundriss).	Anwendung: Realistische Darstellung, menschliches Sehen.

8.5.3 Perspektivische Transformation

Mathematisch wird die perspektivische Verkürzung durch den Strahlensatz hergeleitet. Ein Punkt (x, y, z) wird auf die Bildebene im Abstand d projiziert. Die projizierte Koordinate ist $y_p = y \cdot \frac{d}{d+z}$ (wobei hier oft d als Position der Bildebene oder Abstand zum Augpunkt definiert wird).

In homogenen Koordinaten wird dies erreicht, indem die w-Komponente verändert wird. Eine typische Projektionsmatrix schreibt $-z$ (oder einen Faktor davon) in die w-Komponente.

Beispielmatrix für perspektivische Transformation (Zentrum im Ursprung, Bildebene bei $z = -d$):

$$P = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -1/d & 0 \end{pmatrix}$$

Das Ergebnisvektor ist $(x, y, z, -z/d)^T$. Nach der Homogenisierung (Division durch $w = -z/d$) erhalten wir die skalierten Koordinaten.

8.5.4 Kanonisches Sichtvolumen (Canonical View Volume)

In der Pipeline wird die komplexe Pyramide des Sichtvolumens (Frustum) durch die Projektionsmatrix in einen achsenparallelen Einheitswürfel (das kanonische Sichtvolumen) transformiert.

- **Parallelprojektion:** Das Sichtvolumen ist bereits ein Quader. Transformation ist Translation + Skalierung.
- **Perspektive:** Das Sichtvolumen ist ein Pyramidenstumpf. Die Matrix verformt diesen Stumpf in einen Würfel. Dadurch wird das spätere Clipping und die Rasterisierung vereinheitlicht.

8.6 3D-Interaktion

Ein fundamentales Problem bei der Interaktion mit 3D-Szenen ist die Nutzung von **2D-Eingabegeräten** (Maus, Touchscreen).

Das Problem der Mehrdeutigkeit

Eine Bewegung der Maus auf dem 2D-Bildschirm entspricht unendlich vielen möglichen Bewegungen im 3D-Raum (entlang des Sehstrahls).

8.6.1 Lösungsansätze

- **Mehrfachansichten (Quad-View):** Gleichzeitige Darstellung von Oben, Vorne, Seite und Perspektive. Interaktion findet in den 2D-Ansichten statt.
- **Modi-Umschaltung:** Mausbewegung wird je nach Taste/Modus unterschiedlich interpretiert (z. B. x/y Bewegung vs. z Bewegung).
- **3D-Widgets / Manipulatoren:**

– Visuelle Hilfsobjekte, die in die Szene eingeblendet werden (z.B. Pfeile für Achsen, Ringe für Rotation).
– Der Benutzer klickt auf einen bestimmten Teil des Manipulators (Handle).
– Dadurch wird die Bewegungsfreiheit eingeschränkt (Constraint), z. B. “Rotation nur um die X-Achse”.
– **Vorteil:** Intuitiv, reduziert Mauswege, löst Mehrdeutigkeit durch explizite Auswahl der Achse/Ebene.

9 3D-Visualisierung

Dieses Kapitel behandelt die Grundlagen der Gewinnung, Verarbeitung und Visualisierung von 3D-Daten. Der Fokus liegt auf der Unterscheidung zwischen indirekter (Oberflächen-) und direkter (Volumen-) Visualisierung sowie den mathematischen und algorithmischen Grundlagen des Volume Renderings.

9.1 3D-Daten und Datengewinnung

3D-Daten bestehen aus Messwerten, die im dreidimensionalen Raum verteilt sind. Jeder Wert besitzt Koordinaten (x, y, z) und kann skalar (z. B. Dichte, Temperatur) oder höherdimensional (z. B. Vektorfelder wie Windrichtung) sein.

Datenquellen

- **Terrain-Daten**: Höhenmessungen an Positionen (x, y), oft ergänzt durch Satellitenbilder.
- **Laser Scanning**: Aktive Projektion eines Laserstrahls auf eine Oberfläche. Durch Triangulation (bekannter Abstand und Winkel zwischen Laser und Kamera) wird die Tiefe berechnet. Ergebnis: Unstrukturierte Punktwolke.
- **Range Images**: Bilder, bei denen jeder Pixel (u, v) eine Tiefeninformation $r(u, v)$ speichert.
- **Medizinische Bilddaten (CT/MRI)**: Messung physikalischer Eigenschaften (Protonendichte bei MRI, Gewebedichte bei CT). Die Daten liegen meist als Stapel von 2D-Schichten (*Slices*) vor, was ein reguläres 3D-Gitter bildet.
- **Simulation**: Numerische Simulationen (z. B. Wetter, Strömung), definiert auf Gittern.

9.2 Triangulation von Punktwolken

Um aus unstrukturierten Punktwolken (Menge von Punkten $s_i = (x_i, y_i, z_i)$) eine visualisierbare Oberfläche zu erzeugen, ist eine Triangulation notwendig.

9.2.1 Planare Triangulation

Bei einfachen Oberflächen (z. B. Terrain ohne Überhänge) können Punkte auf eine 2D-Ebene projiziert, dort trianguliert und das entstehende Netz anschließend anhand der z-Werte deformiert werden.

9.2.2 Voronoi-Diagramm und Delaunay-Triangulation

Diese Konzepte sind fundamental für die Vernetzung von Punkten.

Voronoi-Diagramm

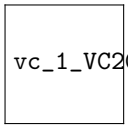
Für eine Menge von Punkten definiert die **Voronoi-Zelle** eines Punktes S_i den Bereich, der näher an S_i liegt als an jedem anderen Punkt der Menge. Die Kanten sind die Orte, die zu zwei Punkten den gleichen Abstand haben; Knoten haben zu mindestens drei Punkten den gleichen Abstand.

Delaunay-Triangulation

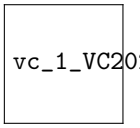
Der duale Graph des Voronoi-Diagramms. Verbindet die Zentren benachbarter Voronoi-Zellen.

- **Leer-Kreis-Eigenschaft**: Der Umkreis eines jeden Dreiecks in einer Delaunay-Triangulation enthält keine anderen Punkte der Punktmenge.
- **Maximierung der kleinen Winkel**: Vermeidet schmale, langgestreckte Dreiecke ("Sliver Polygons"), was für die numerische Stabilität und das Rendering vorteilhaft ist.

Voronoi-Parkettierung (Tesselation) → Voronoi-Diagramm → Delaunay-Triangulation



nicht erlaubt



Korrektur der Delaunay-Triangulation durch Umdrehen der Kanten („Edge Flipping“):

9.3 Indirekte Volumenvisualisierung (Oberflächenrendering)

Bei der indirekten Visualisierung wird aus den Volumendaten eine explizite geometrische Zwischenrepräsentation (meist Polygone) extrahiert.

9.3.1 Isoflächen (Isosurfaces)

Analog zu Konturlinien in 2D (Linien gleicher Höhe) sind Isoflächen in 3D Trennflächen zwischen Strukturen unterschiedlicher Dichte. Eine Isofläche ist definiert durch die implizite Gleichung:

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

wobei $V(x)$ der Voxelwert und τ der Isowert (Schwellenwert) ist.

9.3.2 Marching Cubes Algorithmus

Der Standardalgorithmus zur Extraktion von Isoflächen aus Gitterdaten.

1. Betrachte eine logische Volumenzelle, gebildet durch 8 Voxel-Nachbarn.
2. Klassifiziere jeden Eckpunkt als “innen” ($V \geq \tau$) oder “außen” ($V < \tau$).
3. Dies ergibt $2^8 = 256$ mögliche Konfigurationen.
4. Unter Ausnutzung von Symmetrien (Rotation, Invertierung) Reduktion auf 15 Basisfälle.
5. Generiere Dreiecke basierend auf der Konfiguration (Interpolation der Schnittpunkte auf den Kanten).

Das 2D-Äquivalent hierzu ist der **Marching Squares** Algorithmus (4 Pixel, 16 Fälle).

9.3.3 Optimierung der Rendering-Pipeline

Da Marching Cubes extrem viele Dreiecke erzeugen kann, sind Optimierungen notwendig:

- **Culling**: Entfernen nicht sichtbarer Geometrie.
  - *Backface-Culling*: Rückseiten ignorieren.
  - *View-Frustum-Culling*: Außerhalb des Sichtbereichs ignorieren.
  - *Occlusion-Culling*: Verdeckte Objekte ignorieren.
- **Meshreduktion**: Vereinfachung des Netzes durch Zusammenfassen von Polygonen in flachen Bereichen, um die Anzahl der Primitive zu senken.
- **Mesh-Glättung**: Entfernung von Artefakten und Rauschen (z. B. Laplacesche Glättung), wobei darauf geachtet werden muss, das Volumen nicht zu stark zu verfälschen (“Shrinking”).

9.4 Direkte Volumenvisualisierung (Volume Rendering)

Im Gegensatz zur indirekten Methode wird hier keine Geometrie erzeugt. Das Bild wird direkt aus den Voxeldaten berechnet, indem physikalische Lichtinteraktion simuliert wird.

9.4.1 Physikalisches Modell: Density Emitter Model

Das Volumen wird als halbtransparente Wolke betrachtet, die Licht sowohl emittiert (leuchtet) als auch absorbiert (abschwächt).

9.4.2 Volumen-Rendering-Gleichung

Die Intensität $I(s)$ entlang eines Sichtstrahls berechnet sich aus dem Hintergrundlicht I_{s_0}, das durch das Volumen abgeschwächt wird, und dem kumulierten Licht, das von den Voxeln entlang des Strahls emittiert und wiederum abgeschwächt wird.

Integralform

$$I(s) = I_{s_0} \cdot e^{-\int_{s_0}^{S} \tau(t)dt} + \int_{s_0}^{S} Q(\tilde{s}) \cdot e^{-\int_{\tilde{s}}^{S} \tau(t)dt} ds$$

- $\tau(t)$: Optische Dichte (Absorptionskoeffizient).
- $Q(\tilde{s})$: Emission (Leuchtkraft) an der Stelle $\tilde{s}$.
- Der Exponentialterm beschreibt die Transparenz (Transmission) über eine Strecke.

9.4.3 Diskrete Näherung und Ray Casting Pipeline

Für die computergrafische Umsetzung wird das Integral durch eine Summe approximiert (Ray Casting). Der Strahl wird in Segmente Δs unterteilt.

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

Hierbei ist $t_k = e^{-\tau_k \Delta s}$ die Transparenz des k-ten Segments. Die Pipeline besteht aus vier Schritten:

1. Abtastung (Sampling): Entlang des Sichtstrahls werden Werte in Abständen Δs genommen. Da diese Positionen meist nicht exakt auf Voxelzentren fallen, ist **Interpolation** nötig:

- *Nearest Neighbor*: Blockartige Artefakte, schnell.
- *Trilinear*: Glatter, Standardverfahren (interpoliert in x, y, z Richtung).
- *Shannon-Theorem*: Die Abtastrate muss hoch genug sein (Nyquist-Frequenz), um Aliasing-Artefakte zu vermeiden ($\Delta s < 0.5\times$ Voxelgröße).

2. Klassifikation und Beleuchtung (Shading): Den interpolierten Skalarwerten müssen optische Eigenschaften zugeordnet werden.

- **Transferfunktionen**: Bilden Dichtewerte auf Farbe (Q) und Opazität/Transparenz (t) ab.
  - *Farbtransferfunktion*: Ordnet Farbe zu (z. B. Knochen = weiß, Muskel = rot).
  - *Opazitätstransferfunktion*: Bestimmt Sichtbarkeit (z. B. Luft = transparent).
- **Shading**: Um 3D-Strukturen besser erkennbar zu machen, wird eine Beleuchtung simuliert (z. B. Phong-Shading). Der Normalenvektor für das Shading wird meist aus dem Gradienten der Voxeldaten berechnet.

3. Komposition (Compositing): Das Aufsummieren der Farb- und Opazitätswerte entlang des Strahls.

Back-to-Front vs. Front-to-Back

- **Back-to-Front** (vom Hintergrund zum Auge): Ähnlich dem Painter's Algorithm.

$$I_k = I_{k-1} \cdot t_k + C_k$$

  wobei C_k der emittierte Farbanteil ist.
- **Front-to-Back** (vom Auge zum Hintergrund): Ermöglicht Optimierung. Man muss Intensität I und akkumulierte Transparenz τ tracken.

$$I_{k+1} = I_k + C_{k+1} \cdot \tau_k$$

$$\tau_{k+1} = \tau_k \cdot t_{k+1}$$

- **Early Ray Termination**: Wichtige Optimierung bei Front-to-Back. Wenn die akkumulierte Opazität fast 100% erreicht (Transparenz ≈ 0), kann der Strahl abgebrochen werden, da dahinterliegende Voxel nicht mehr sichtbar sind.

10 Szenengraphen für interaktive 3D-Anwendungen am Beispiel X3DOM

10.1 Einführung und Motivation

Die 3D-Computergrafik ist ein integraler Bestandteil moderner Anwendungen, von Spielen bis hin zu industriellen Visualisierungen. Bei der Entwicklung solcher Systeme stehen Entwickler vor mehreren technischen Herausforderungen:

- **Effizienz:** Handhabung großer Datenvolumina und Echtzeit-Rendering.
- **Portabilität:** Lauffähigkeit auf Desktop-PCs, Tablets und Mobilgeräten.
- **Integration:** Nahtloses Einbetten von 3D-Inhalten in bestehende Anwendungen (z.B. Webseiten).

Ein zentraler Ansatz zur Bewältigung dieser Komplexität ist die Strukturierung der Daten mittels **Szenengraphen**.

10.2 Strukturierung von 3D-Szenendaten

Um eine 3D-Szene rendern zu können, benötigt die Grafikpipeline vielfältige Informationen. Diese lassen sich nicht als unstrukturierte Liste verwalten, da komplexe Beziehungen zwischen den Elementen bestehen.

10.2.1 Notwendige Informationen für das Rendering

- **Objekt-Geometrie:** Form der Objekte (z.B. Vertices, Polygone).
- **Transformationen:** Position, Rotation und Skalierung im Raum.
- **Materialien:** Oberflächenbeschaffenheit, Farben, Texturen.
- **Kameras:** Blickwinkel und Projektionsart.
- **Lichter:** Lichtquellen (Punktlicht, Richtung, Spot) und deren Eigenschaften.
- **Spezialeffekte:** Nebel, Schatten, Skyboxes.

10.2.2 Beziehungen zwischen Daten

In einer Szene treten häufig Redundanzen und Hierarchien auf:

- Mehrere Objekte teilen sich dasselbe Material (z.B. alle Reifen eines Autos sind schwarz).
- Dasselbe geometrische Objekt wird mehrfach instanziiert (z.B. vier identische Räder an verschiedenen Positionen).
- Objekte sind logisch gruppiert und bewegen sich gemeinsam (z.B. bewegt sich das Rad relativ zum Auto, das Auto relativ zur Welt).

10.3 Das Konzept des Szenengraphen

Ein Szenengraph ist die fundamentale Datenstruktur zur Organisation dieser Informationen.

Szenengraph (Scene Graph)

Ein Szenengraph ist ein **gerichteter, azyklischer Graph** (Directed Acyclic Graph, DAG), der die logische und räumliche Struktur einer 3D-Szene beschreibt.

- **Gerichtet:** Kanten haben eine klare Richtung (Eltern → Kind).
- **Azyklisch:** Es gibt keine geschlossenen Wege (Schleifen); eine Endlosschleife beim Rendern wird verhindert.
- **Wurzelknoten:** Der Einstiegspunkt für die Traversierung.

Im Gegensatz zu einem reinen Baum kann ein Knoten im DAG (außer der Wurzel) **mehrere Elternknoten** besitzen. Dies ermöglicht die Wiederverwendung von Ressourcen (Instanziierung).

10.3.1 Knotentypen

Ein Szenengraph besteht üblicherweise aus drei abstrakten Kategorien von Knoten:

1. **G (Gruppierung):** Organisiert Unterobjekte logisch (z.B. "Auto"). Kann Bedingungen prüfen (z.B. Sichtbarkeit).
2. **T (Transformation):** Enthält geometrische Transformationen (Translation, Rotation, Skalierung).
3. **O (Objekt/Shape):** Die eigentlichen Blattknoten, die Geometrie und Material definieren (das, was gezeichnet wird).

vc_1_VC2025_10-X3DOM_page_10_1.png

10.3.2 Traversierung und Rendering

Der Prozess des Zeichnens wird als **Traversierung** bezeichnet. Dabei wird der Graph rekursiv durchlaufen, beginnend bei der Wurzel.**Ablauf der Traversierung:**

- Die Anwendung startet an der Wurzel.
- Jeder Kindknoten wird rekursiv besucht (Tiefensuche).
- Während des Abstiegs wird der globale Zustand aktualisiert, insbesondere die **Current Transformation Matrix (CTM)**.

Operationen während der Traversierung:

- **Bei Gruppierungsknoten:** Prüfe, ob die Gruppe aktiv ist. Falls ja, traversiere alle Kinder.
- **Bei Transformationsknoten:** Multipliziere die aktuelle CTM mit der lokalen Transformationsmatrix M des Knotens:
$$CTM_{neu} = CTM_{alt} \cdot M$$
Diese akkumulierte Matrix wird an die Kinder weitergegeben.
- **Bei Objektknoten:** Zeichne das Objekt unter Verwendung der aktuellen CTM. Dadurch wird das Objekt an der korrekten, akkumulierten Weltposition dargestellt.

10.3.3 Vorteile

- **Wiederverwendbarkeit:** Geometrien (z.B. ein Rad) müssen nur einmal im Speicher liegen und können mehrfach referenziert werden.
- **Semantische Gruppierung:** Einfaches Ein-/Ausblenden ganzer Baugruppen.
- **Transformationshierarchie:** Bewegt man einen Elternknoten (z.B. “Auto”), bewegen sich alle Kindknoten (z.B. “Räder”) automatisch mit.

10.4 X3DOM: Szenengraphen im Web

X3DOM ist eine Technologie, die 3D-Szenengraphen direkt in HTML integriert.

X3DOM

X3DOM ist eine Open-Source-Lösung, die den **X3D**-Standard direkt im HTML-DOM verfügbar macht. Es benötigt keine Plugins, sondern nutzt WebGL via JavaScript (“Polyfill”-Layer), um 3D-Inhalte nativ im Browser zu rendern.

10.4.1 Hintergrund: VRML und X3D

- **VRML (Virtual Reality Modeling Language, 1994/97):** Erster Standard für 3D im Web.
- **X3D (Extensible 3D):** Der XML-basierte Nachfolger von VRML. Er ist modular (Profile) und unterstützt verschiedene Encodings (XML, Binary, VRML-Classic).

10.4.2 Integration in HTML

X3DOM ermöglicht eine **deklarative** Beschreibung der Szene. Das bedeutet, man beschreibt *was* dargestellt werden soll (Struktur), nicht *wie* es gezeichnet wird (Imperative Befehle wie in OpenGL/WebGL direkt).

Grundstruktur eines X3DOM-Dokuments:

```
<!DOCTYPE html><html><head><script src=``x3dom.js''></script><link rel=``stylesheet'' href=``x3dom.css''
```

10.5 Wichtige X3D-Knoten

Der Aufbau einer Szene in X3DOM folgt einer strikten Hierarchie:

10.5.1 Shape (Gestalt)

Der `<Shape>`-Knoten verbindet Geometrie mit Aussehen. Er enthält in der Regel zwei Kindknoten:

1. **Geometry:** Die Form (z.B. `<Box>`, `<Sphere>`, `<Cone>`, `<IndexedTriangleSet>`).
2. **Appearance:** Das Aussehen.

10.5.2 Appearance und Material

Der `<Appearance>`-Knoten beinhaltet den `<Material>`-Knoten, der die optischen Eigenschaften nach dem **Phong-Beleuchtungsmodell** definiert. Das Phong-Modell setzt sich zusammen aus:

$$I_{out} = I_{ambient} + I_{diffuse} + I_{specular}$$

Parameter im `<Material>`-Tag:

- `diffuseColor`: Die eigentliche Farbe des Objekts unter direkter Beleuchtung.
- `specularColor`: Farbe des Glanzlichts (meist weiß).
- `shininess`: Größe des Glanzpunkts (Rauheit der Oberfläche).
- `emissiveColor`: Selbstleuchten (unabhängig von Lichtquellen).
- `transparency`: Durchsichtigkeit.

10.5.3 Komplexe Geometrie: IndexedTriangleSet

Für beliebige Formen, die nicht durch Primitive (Box, Kugel) darstellbar sind, werden Polygonnetze verwendet.

- **Coordinate Point:** Liste aller Eckpunkte (Vertices) im 3D-Raum (x y z).
- **Index:** Definiert, welche Punkte zu einem Dreieck verbunden werden. Dies spart Speicher, da Punkte wiederverwendet werden können.
- **Normal:** Normalenvektoren für die Lichtberechnung.

Hinweis: Große Geometrien sollten aus Performancegründen in externe binäre Dateien ausgelagert werden (ähnlich wie Bilder via `src`).

10.5.4 Transformationen

Der `<Transform>`-Knoten manipuliert das Koordinatensystem für seine Kinder.

- `translation='x y z'`: Verschiebung.
- `rotation='x y z angle'`: Rotation um Achse (x,y,z) mit Winkel (Bogenmaß).
- `scale='x y z'`: Skalierung.

Durch Verschachtelung von Transform-Knoten entstehen kinematische Ketten (z.B. Oberarm → Unterarm → Hand).

10.6 Instanziierung im DOM (DEF/USE)

Ein fundamentales Problem bei der Abbildung von Szenengraphen auf das HTML-DOM ist die Struktur:

- **HTML-DOM:** Ist ein Baum. Jedes Element hat genau *einen* Elternknoten.
- **Szenengraph:** Ist ein DAG. Ein Geometrie-Knoten kann von mehreren Transform-Knoten referenziert werden (Instanziierung).

10.6.1 Lösung: DEF und USE Mechanismus

X3D löst dies durch Referenzierung:

1. **DEF (Definition):** Ein Knoten wird definiert und benannt. Er wird an dieser Stelle normal verarbeitet.`<Group DEF='AutoRad'> ...</Group>`
2. **USE (Verwendung):** An einer anderen Stelle im Baum wird eine Instanz eingefügt, die auf die Definition verweist.`<Group USE='AutoRad'></Group>`

Dies ermöglicht es, komplexe Objekte (z.B. "Suzanne mit Hut") einmal zu definieren (Gruppierung von Affenkopf und Zylinder) und dann an verschiedenen Positionen mittels Transformationen mehrfach darzustellen, ohne die Geometriedaten zu duplizieren.