

1. EVC Test - Computervision

Vorwort

Diese Stoffsammlung/Zusammenfassung enthält den Stoff, der in der EVC Vorlesung der TU Wien im Sommersemester 2025 vorgetragen wurde, der auch in den jeweiligen Skripten und Slides zu finden ist. Die Struktur dieser Zusammenfassung basiert demnach auch der des Skriptums.

Disclaimer

Vieles der Zusammenfassung wurde mit AI generiert, basiert allerdings nur auf Inhalten der Unterlagen. Die Stellen die mit AI generiert wurden, wurden von mir überprüft und mit den Unterlagen verglichen, aber auch ich kann Fehler machen.

Demnach, falls sich irgendwo Fehler befinden oder es Verbesserungsvorschläge gibt, bitte an [@xmozz](#) auf Discord wenden.

Inhalt

- [1. Einführung in Computer Vision](#)
- [2. Bildaufnahme](#)
- [3. Bildcodierung und Kompression](#)
- [4. Punktoperationen](#)
- [5. Lokale Operationen](#)
- [6. Kantenfilterung](#)
- [7. Globale Operationen](#)
- [8. Bildmerkmale - Interest Points](#)

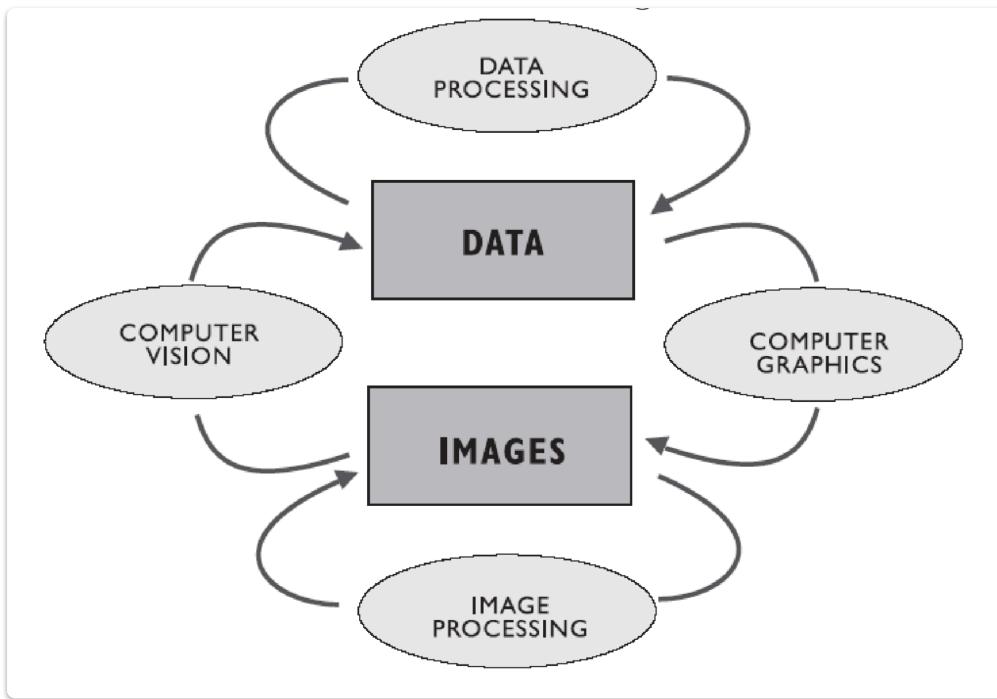
1. Einführung in Computer Vision

Quellen:

- [EVC_Skriptum_CV, p.4](#)
- [EVC_Skriptum_CV, p.5](#)
- [EVC_Skriptum_CV, p.6](#)
- [EVC_Skriptum_CV, p.7](#)

Die drei Teilbereiche

- **Computergrafik:**
 - Erzeugt Bilder aus Daten (z.B. Diagramme, 3D-Modelle)
 - Ziel: Realistische Bildgenerierung.
- **Computer Vision (Maschinelles Sehen):**
 - Extrahiert Semantik aus Bildern (also genau solche Diagramme / Modelle).
 - Ziel: Nachbildung des Sehvorgangs, Erkennung von Objekten, Bewegung, etc.
 - Teilbereich: Mustererkennung (z.B. optische Zeichenerkennung).
- **Bildverarbeitung:**
 - Verbessert Bilder für bestimmte Aufgaben (z.B. Rauschunterdrückung, Qualitätsverbesserung).
 - Ergebnis: Ein verändertes Bild.
 - Unterschied zur Bildbearbeitung: Bildverarbeitung ist algorithmisch, Bildbearbeitung ist interaktiv.



Definition Computer Vision

- Automatische Ableitung von Struktur und Eigenschaften einer 3D-Welt aus 2D-Bildern.
- Ziel: Computer sollen visuelle Daten wahrnehmen, verarbeiten und verstehen wie Menschen.
- Anwendungsbereiche:
 - Industrielle Bildverarbeitung (z.B. Flaschenerkennung).
 - Künstliche Intelligenz (z.B. autonome Roboter).
 - Gesichtserkennung.
- Computer Vision vs. Machine Vision: Computer Vision ist die Kerntechnologie zur Bildanalyse, Machine Vision kombiniert die Bildanalyse mit anderen Technologien.
- Teilgebiete: Szenenrekonstruktion, Objekterkennung, Video-Tracking, etc.
- Beispiel Gesichtserkennung:



Abbildung 2: Gesichtserkennung

3. Menschliches Sehen vs. Computer Vision

- Computer Vision versucht, menschliches Sehen zu reproduzieren, aber:
 - Menschliches Sehen ist Ergebnis von Millionen Jahren Evolution
 - Menschliches Sehen ist fehlerhaft (optische Täuschungen).
 - Wenig Wissen über die genauen Prozesse des menschlichen Sehens.
 - Die Frage besteht, ob es erstrebenswert ist, dass Maschinen genauso sehen wie Menschen.
- Das menschliche Sehen ist nicht unfehlbar. Optische Täuschungen und Mehrdeutigkeiten beweisen dies.
- In CV Herleitung durch geometrischen Eigenschaften, Materialeigenschaften und Beleuchtung
- **Begrenztes Wissen:**
 - Unser Verständnis der komplexen Prozesse im Gehirn, die nach der Reizaufnahme durch das Auge ablaufen, ist begrenzt.
- **Fehlerhaftigkeit des menschlichen Sehens:**
 - Optische Täuschungen, Mehrdeutigkeiten und Inkonsistenzen zeigen, dass unser Sehen nicht perfekt ist.
- **Subjektivität der Wahrnehmung:**

- Die Drehung eines Bildes kann unsere Wahrnehmung komplett verändern.
- **Helmholtz' Theorie:**
 - Wir sehen, was wir erwarten oder was unter normalen Bedingungen zu erwarten wäre.
 - Unser Gehirn konstruiert aktiv unsere visuelle Realität.
- **Toleranzunterschiede:**
 - Wir akzeptieren die Unvollkommenheit des menschlichen Sehens, sind aber bei Maschinen weniger tolerant.
- **Die Frage:**
 - Wollen wir, dass Maschinen genauso "sehen" wie wir, mit all den damit verbundenen Fehlern und Verzerrungen?

2. Optische Täuschungen und ihre Bedeutung

- **Einblicke in die Natur des Sehens:**
 - Optische Täuschungen geben uns wertvolle Einblicke in die Funktionsweise unseres visuellen Systems.
- **Kontrollierte Halluzination?:**
 - Die Frage, ob unser Sehen eher einer "kontrollierten Halluzination" gleicht, wird aufgeworfen.
- **Zuverlässigkeit der Wahrnehmung:**
 - Die Täuschungen verdeutlichen, dass unser Vertrauen in unsere visuellen Fähigkeiten nicht immer gerechtfertigt ist.

3. Konsequenzen für die Computer Vision

- **Nachbildung des menschlichen Sehens:**
 - Die Schwierigkeiten bei der Nachbildung des menschlichen Sehens werden deutlich.
- **Erwartungen an Maschinen:**
 - Wir müssen uns fragen, welche Art von "Sehen" wir von Maschinen erwarten.
- **Grundlagenforschung:**
 - Die Erforschung optischer Täuschungen hilft, die Grundlagen des menschlichen Sehens besser zu verstehen, was wiederum in der Entwicklung von Computer Vision Systemen hilfreich ist.

4. Weitere Informationen:

- Weitere Infos zu wie wir sehen findet man hier:
 - [2. Bildaufnahme](#)
 - [4. Farbe](#)

4. Plenoptische Funktion

- Beschreibt die Menge aller Lichtstrahlen in einem Raum.
- Parameter zum Beschreiben der Beleuchtung:
 - Position (V_x, V_y, V_z).
 - Einfallswinkel (θ, ϕ).
 - Wellenlänge (λ).
 - Zeit (t).

Wir wollen nun die Parameter zur Beschreibung beleuchteter Umgebungen betrachten. Im ersten Schritt nehmen wir eine Schwarz-Weißfotografie einer Lochkamera an, die uns darüber Auskunft gibt, wie groß die Intensität des Lichtes von einem einzigen Blickpunkt aus zu einem bestimmten Zeitpunkt, gemittelt über die Wellenlängen des Spektrums des sichtbaren Lichtes, ist. Diese Lichtintensitätsverteilung P kann beim Durchgang der Strahlenbündel durch die Linse gemessen und anhand von Kugelkoordinaten $P(\theta, \phi)$ oder kartesischen Koordinaten $P(x, y)$ parametrisiert werden.

Ein Farbbild enthält zusätzliche Informationen, welche die Veränderung der Lichtintensität bezüglich der Wellenlänge λ berücksichtigt (daher $P(\theta, \phi, \lambda)$). Ein Farbvideo oder Film erweitert die Informationen um die Dimension der Zeit t : $P(\theta, \phi, \lambda, t)$. Schlussendlich zeigen farbige holografische Filme die komplette wahrnehmbare Lichtintensität von jeder Betrachtungsposition V_x, V_y und V_z an: $P(\theta, \phi, \lambda, t, V_x, V_y, V_z)$. Solch eine vollständige Repräsentation impliziert die Beschreibung aller möglichen Bilder, die von einem bestimmten Raum-Zeit Stück der Welt aufgenommen werden können (unter Vernachlässigung der Polarisation und momentanen Phase des einfallenden Lichtes). Es ist zu beachten, dass die plenoptische Funktion keine zusätzlichen Parameter zur Spezifizierung der "Blickrichtung" des Auges benötigt. Die Veränderung der Blickrichtung ohne Veränderung der Position des Auges hat keine Auswirkung auf die Verteilung des Lichtes eines Strahlenbündels beim Auftreffen auf die Pupille. Nur die relative

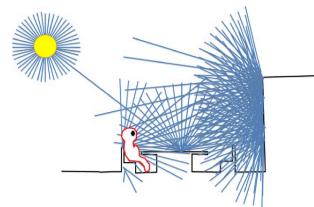


Abbildung 4: Plenoptische Funktion

Einfallposition des Lichtes auf der Netzhaut wird dadurch verändert. Die Messung der plenoptischen Funktion erfolgt durch die imaginäre Platzierung eines idealen Auges an jeder möglichen (V_x, V_y, V_z) Position und beinhaltet die Messung der Intensität der Lichtstrahlen, für jeden möglichen Einfallswinkel (θ, ϕ) , für jede Wellenlänge λ , zu jedem Zeitpunkt t , die durch das Zentrum der Pupille gehen. Die Winkel (θ, ϕ) können immer relativ zu einer optischen Achse, die parallel zur V_z -Achse liegt, beschrieben werden. Die resultierende Funktion hat die Form: $p = P(\theta, \phi, \lambda, t, V_x, V_y, V_z)$.

- Die plenoptische Funktion beschreibt alle möglichen Bilder, die von einem bestimmten Raum-Zeit-Abschnitt der Welt aufgenommen werden können.

5. Camera Obscura / Lochkamera

- Einfachstes Kameraprinzip.
- war die erste Spycam
- Funktionsweise: Lichtstrahlen werden durch eine kleine Öffnung auf eine Bildebene projiziert.
- Ergebnis: Verkleinertes, seitenverkehrtes Abbild.
- Diente als Grundlage für die Entwicklung der Fotografie.

Die Lochkamera besteht aus einer geschlossenen Box mit einer winzigen Öffnung an der Vorderseite ("Pinhole") und der Bildebene an der gegenüberliegenden Rückseite (siehe Abbildung 5). Lichtstrahlen, die von einem Objektpunkt vor der Kamera ausgehend durch die Öffnung einfallen, werden geradlinig auf die Bildebene projiziert, wodurch ein verkleinertes und seitenvorkehrtes Abbild der sichtbaren Szene entsteht. Durch den Einsatz von Spiegeln kann das projizierte Bild in die richtige Position gedreht werden. Eine portable Version der Lochkamera stellte eine Kiste mit einem angewinkelten Spiegel dar, mit denen ein Bild in richtiger Ausrichtung auf ein durchscheinendes Papier, welches auf einer Glasoberfläche liegt, projiziert wird. Die Lochkamera kann als einfache Kamera ohne Linse und einer einzigen kleinen Apertur (Öffnung) angesehen werden.

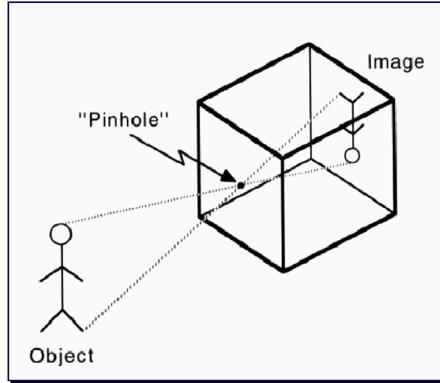


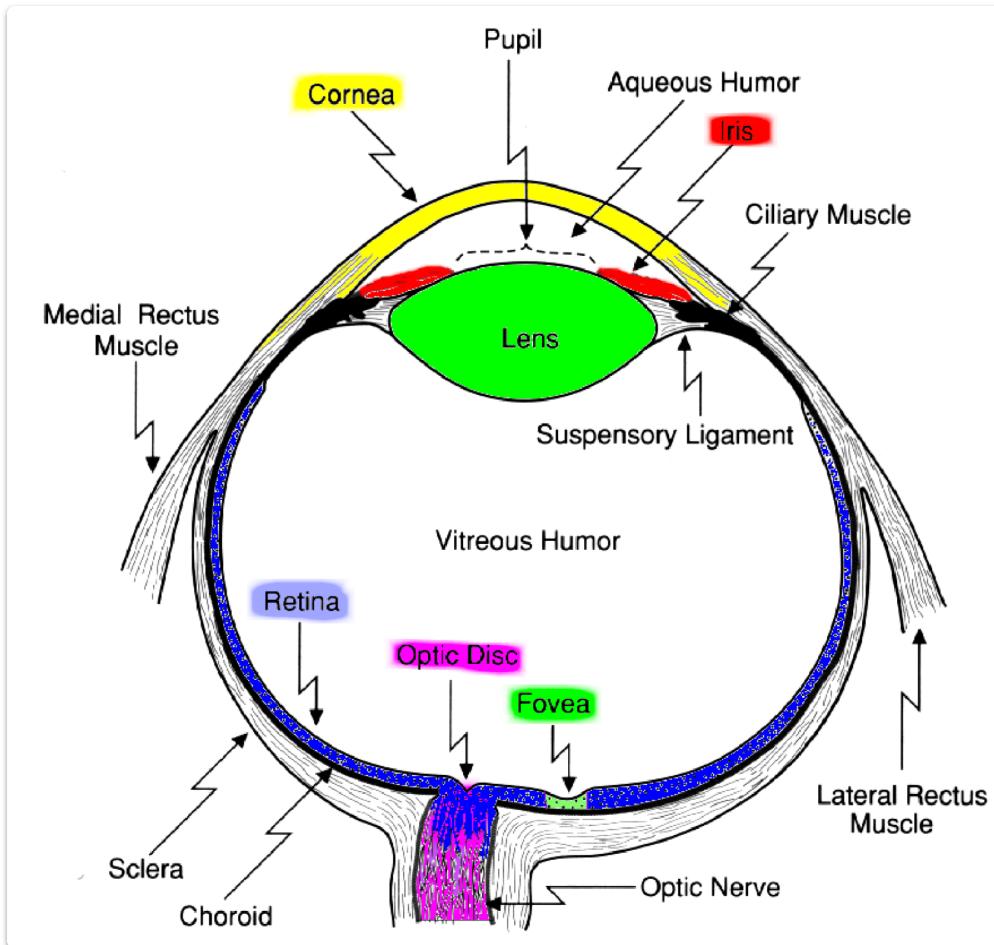
Abbildung 5: Modell der *pinhole camera*

2. Bilddatenaufnahme

Quellen:

- [EVC_Skriptum_CV, p.8 bis EVC_Skriptum_CV, p.14](#)

Das Menschliche Auge:



- Linse des Auges fokussiert Licht aus der Umgebung
- Bild wird **seitenverkehrt & auf dem Kopf stehend** auf die Retina projiziert
- **Retina (Netzhaut)** = lichtempfindliche Membran auf der Rückseite des Auges

Fotorezeptoren

- Spezialzellen, die Lichtreize in **neuronale Impulse** umwandeln
- Zwei Typen:
 - **Stäbchen**
 - Verantwortlich für **Schwarz-Weiß-Sehen**
 - Hohe Lichtempfindlichkeit → wichtig für das **Dämmerungssehen**
 - **Zäpfchen**
 - Zuständig für **Farbsehen**
 - Funktionieren bei **Tageslicht**

- Drei Typen für unterschiedliche **Wellenlängenbereiche** (rot, grün, blau)

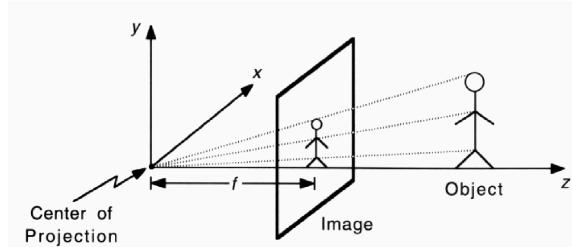
Signalverarbeitung

- Reaktion der Fotorezeptoren: **chemisch-elektrisch**
- Entstehung eines **neuronalen Impulses**
- Weiterleitung über mehrere **Neuronenschichten**
- Schichten sind durch **Synapsen** miteinander verbunden

Besondere Strukturen

- **Fovea centralis** (im Zentrum des gelben Flecks / *Macula lutea*):
 - Ort des **schärfsten Sehens**
 - Dichteste Anordnung von Zäpfchen
- **Sehnerv (Nervus opticus)**:
 - Leitet visuelle Impulse ans Gehirn
 - **50 % der Fasern**: Informationen aus der **Fovea**
 - **50 % der Fasern**: Informationen aus der **restlichen Retina**

Perspektivische Projektion



Grundprinzip

- Einfachstes Modell der Bildaufnahme ohne Linse, nur mit punktförmiger Blende
- Beschreibt perspektivische Projektion eines 3D-Punktes auf eine 2D-Bildecke
- Idealisiert: keine geometrischen Verzerrungen, keine Unschärfe durch Linsen

Annahmen (zur Vereinfachung der Herleitung)

- Zentrum der Projektion (Punkt O) = Ursprung des Kamerakoordinatensystems
- Optische Achse (Kamaraachse) ist entlang der z-Achse der Kamera ausgerichtet
- Bildecke liegt vor dem Zentrum der Projektion (→ keine Bildumkehrung)

Aufbau des Modells

- O = Zentrum der Projektion (Punkt, durch den alle Lichtstrahlen verlaufen)
- Bildecke = Fläche, auf die das 3D-Bild projiziert wird

- f = Brennweite (*Abstand von O zur Bildebene*) (*Bei Kameras: "Abstand zwischen Linse und Chip/Film"*)
- Optische Achse = Gerade durch O , senkrecht zur Bildebene

Mathematische Beziehung (perspektivische Projektion)

- Gegeben: 3D-Punkt $P = (X, Y, Z)$
- Projektionsgleichungen:
 - $x = \frac{f}{Z} \cdot X$
 - $y = \frac{f}{Z} \cdot Y$
- x und y : Position des projizierten Punktes auf der Bildebene

Eigenschaften der Projektion

- Nicht-linear: Bildgröße $\propto 1/Z \rightarrow$ je näher ein Objekt, desto größer erscheint es
- 1:n-Abbildung: mehrere 3D-Punkte können auf denselben 2D-Punkt projiziert werden
- 3D-Linien \rightarrow 2D-Linien (außer wenn parallel zur optischen Achse)
- Winkel und Abstände bleiben nicht erhalten
- Parallele Linien bleiben nur erhalten, wenn sie parallel zur Bildebene liegen
- Informationsverlust: 3D \rightarrow 2D nicht umkehrbar; Tiefeinformation geht verloren

Praktische Anwendung

- Vereinfachtes Modell zur Beschreibung von Bildaufnahmeprozessen
- Grundlage in Computervision, Grafik und Kamerakalibrierung
- Idealisierung realer Kameras zur mathematischen Beschreibung der Abbildungsgeometrie

Linsen

Motivation

- Die einfache Lochkamera hat **praktisch keine Bedeutung** in der realen Bildaufnahme
- Für eine **scharfe Projektion** benötigt man ein sehr kleines Loch \rightarrow lässt kaum Licht durch
- Folge: **lange Belichtungszeiten**, ungeeignet für bewegte Szenen oder praktische Anwendungen

Lösung: optische Linsen

- Statt Lochblende werden **Linsen oder Linsensysteme** verwendet
- Diese bieten bessere Lichtbündelung und schärfere Abbildung
- Abbildungsverhalten ist komplexer, aber wesentlich **praxisnäher**

Modell der dünnen Linse

- Vereinfachtes, idealisiertes Modell
- Ersetzt die Lochblende durch eine **symmetrische, unendlich dünne Linse**
- **Lichtstrahlen werden an einer virtuellen Ebene** (in der Linsenmitte) gebrochen
- Abbildungsgeometrie bleibt gleich wie bei der Lochkamera → perspektivische Projektion

Brennpunktverhalten

- **Parallele Strahlen** zur optischen Achse werden im **Brennpunkt** (Abstand f zur Linse) gebündelt
- Die **Brennweite f** ist ein zentraler Parameter der Abbildung
- Fokuspunkt entsteht auf der Bildebene, wenn das Objekt korrekt positioniert ist

Linsengleichung

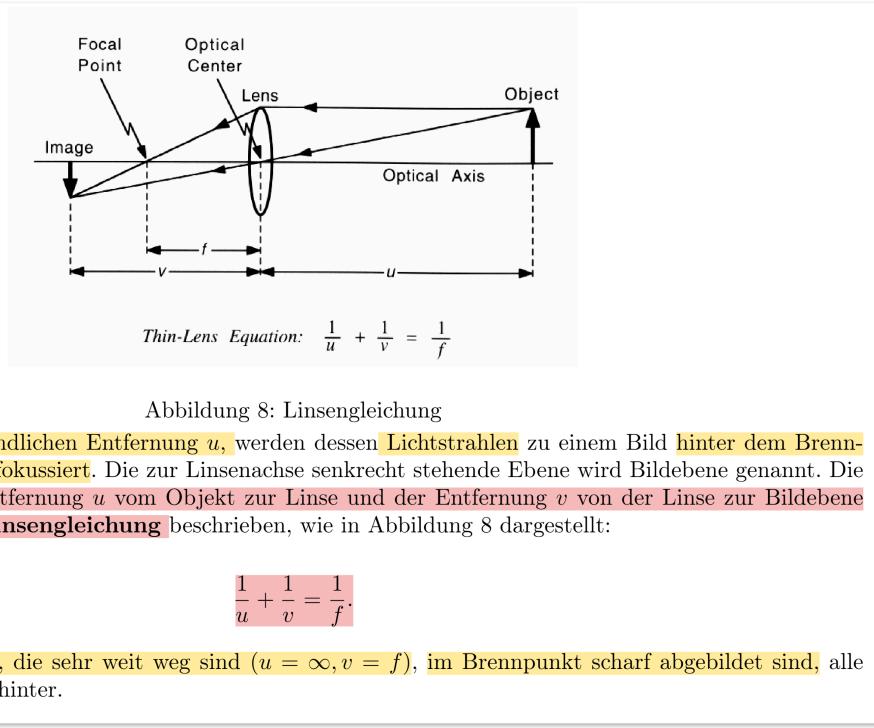


Abbildung 8: Linsengleichung

Liegt ein Objekt in einer endlichen Entfernung u , werden dessen **Lichtstrahlen** zu einem Bild **hinter dem Brennpunkt** in der Entfernung v fokussiert. Die zur Linsenachse senkrecht stehende Ebene wird **Bildebene** genannt. Die Beziehung zwischen der **Entfernung u vom Objekt zur Linse** und der **Entfernung v von der Linse zur Bildebene** wird durch die **einfache Linsengleichung** beschrieben, wie in Abbildung 8 dargestellt:

$$\frac{1}{u} + \frac{1}{v} = \frac{1}{f}.$$

Diese besagt, dass **Objekte, die sehr weit weg sind ($u = \infty, v = f$)**, im Brennpunkt scharf abgebildet sind, alle Objekte, die näher sind, dahinter.

Tiefenschärfebereich (DOF (Depth of Field))

- Wegen einfacher Linsengleichung immer nur Objekte mit bestimmter Entfernung zur Kamera scharf
- alles andere unscharf
- In Praxis Objekte mit gewissen DOF scharf abgebildet.
- DOF = Entfernung zwischen dem nächsten und weitesten entfernten Objekt einer Szene, welches scharf dargestellt wird

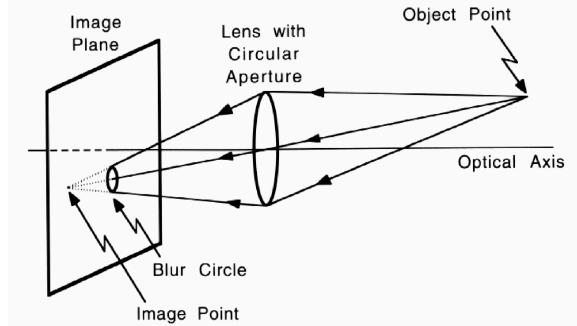
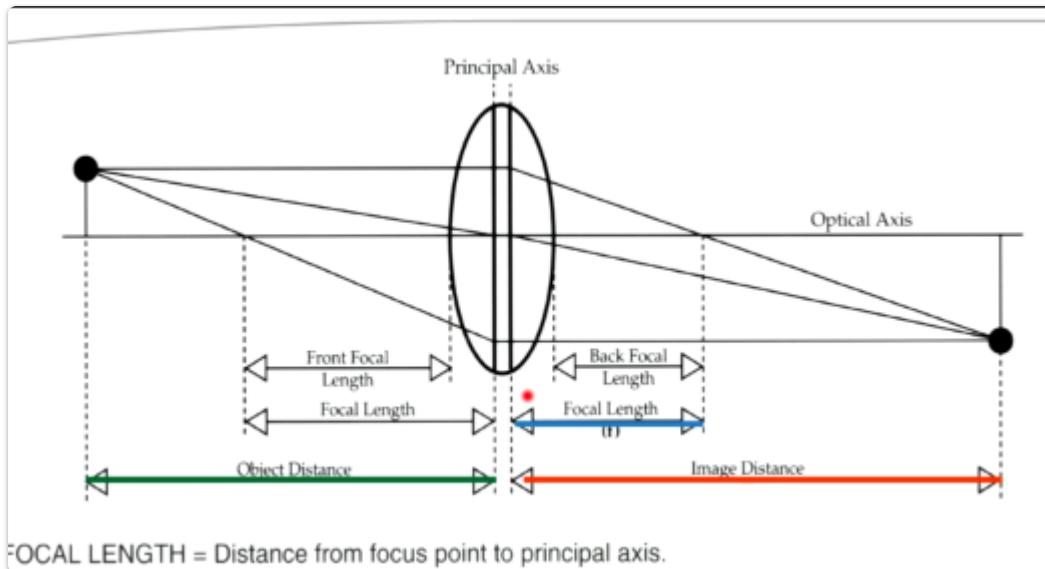


Abbildung 9: Tiefenschärfenbereich

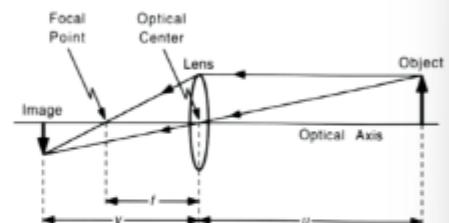


Test-ähnliches Beispiel:

Beispiel



- Eine Person steht 5m vor einer Kamera. Die fokale Länge der Linse beträgt 200mm. Wie muss der Bildabstand gewählt werden, damit die Person scharf auf der Bildebene erscheint?



$$\begin{aligned}
 & \bullet u = 5000 \text{ mm} \quad \frac{1}{u} + \frac{1}{v} = \frac{1}{f} \Rightarrow \frac{1}{5000} + \frac{1}{v} = \frac{1}{200} \Rightarrow \frac{1}{v} = \frac{1}{200} - \frac{1}{5000} \Rightarrow \\
 & \bullet f = 200 \text{ mm} \\
 & \bullet v = ? \\
 & \frac{1}{v} = \frac{24}{5000} \Rightarrow v = \frac{5000}{24} \Rightarrow v = 208,333
 \end{aligned}$$

Einflussfaktoren

- Sensorauflösung
- Größe der Linse / Blendenöffnung
- Brennweite der Linse

Blende und Lichtbündel

- Blende = ringförmige Öffnung vor der Linse
- Bestimmt den **Kegelwinkel** des Strahlenbündels, das auf die Bildebene trifft
- **Kleine Blende**:
 - Weniger Strahlen erreichen die Bildebene
 - Geringere Lichtmenge → dunkleres Bild
 - **Größerer DOF** (mehr vom Bild erscheint scharf)
- **Große Blende**:
 - Mehr Strahlen pro Objektpunkt → höhere Lichtausbeute
 - **Kleinerer DOF** (weniger Tiefenbereich, Bild weniger scharf)

Unschärfekreise (Blur Circles)

- Wenn Objekt nicht im richtigen Abstand v zur Linse steht → Strahlen bündeln sich nicht exakt
- Statt eines Punktes entsteht ein **Unschärfekreis** auf der Bildebene
- Benachbarte Objektpunkte ergeben überlappende Kreise → **Bild wird unscharf**

Zusammenhang: Blende – Blur – DOF

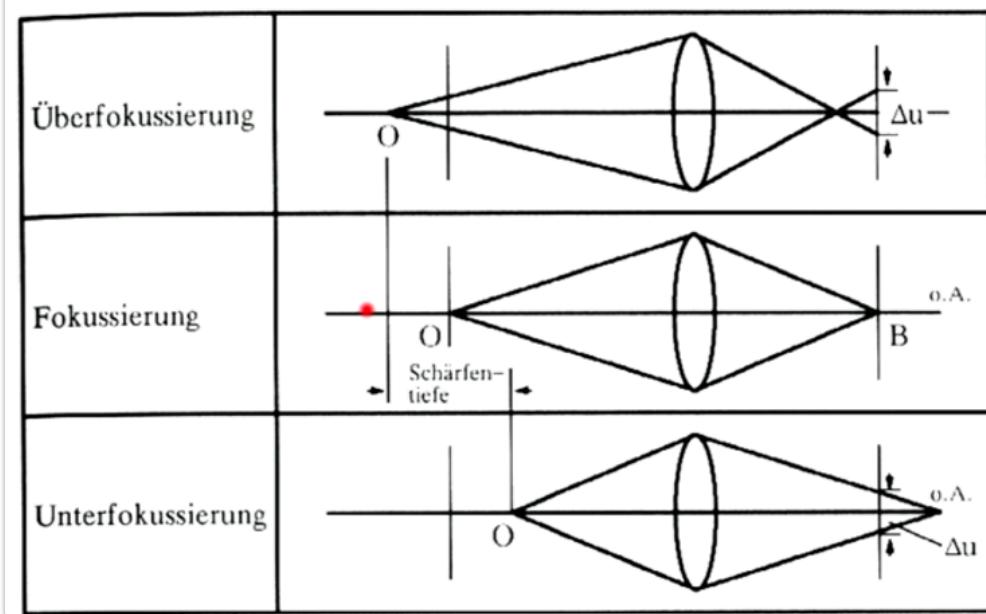
- **Größere Linse** = größere Unschärfekreise = **kleinerer DOF**
- **Kleinere Linse** = kleinere Unschärfekreise = **größerer DOF**

Einfluss der Sensorauflösung

- **Höhere Auflösung**: Unschärfekreise können besser abgebildet werden → **kleinerer DOF**
- Geringere Auflösung: Unschärfen weniger deutlich → scheinbar größerer DOF

Einfluss der Brennweite

- **Längere Brennweite (z. B. Teleobjektiv)** → **kleinerer DOF**
- **Kürzere Brennweite (z. B. Weitwinkelobjektiv)** → **größerer DOF**



Radiometrie

Grundlagen

- Radiometrie = **Messung elektromagnetischer Strahlung**, inkl. sichtbarem Licht
- Ziel: Beschreibung der Lichtmenge, die von der realen Welt auf ein Bild projiziert wird

Strahldichte (Radiance)

- Beschreibt die **Menge des reflektierten Lichtes** eines Oberflächenpunktes
- Richtungsabhängige Größe
- Gibt an, wie viel Licht in eine bestimmte Richtung von einem Punkt abgestrahlt wird
- Entspricht der **Lichtmenge, die durch ein kleines Raumwinkel-Element ausgesendet wird**

Bestrahlungsstärke (Irradiance)

- Gibt an, **wie viel Licht** von einer Oberfläche **empfangen** wird
- Entspricht der Helligkeit eines Bildpunktes
- Wird aus der Strahldichte berechnet, indem über alle Richtungen integriert wird

Spektrum

- Radiometrische Größen beziehen sich auf **Strahlung aller Frequenzen**
- Betrachtet man **einzelne Frequenzbereiche**, spricht man vom **Spektrum**
- Wichtige spektrale Größe:
 - **Spektrale Bestrahlungsstärke (spectral irradiance)**
 - Gibt an, **wie viel Strahlungsenergie pro Wellenlänge** auf eine Fläche trifft

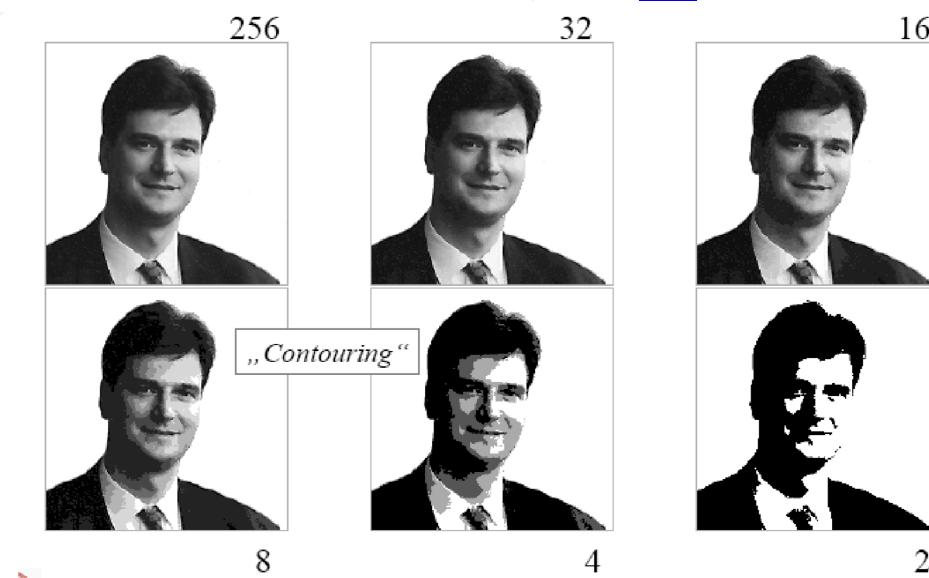
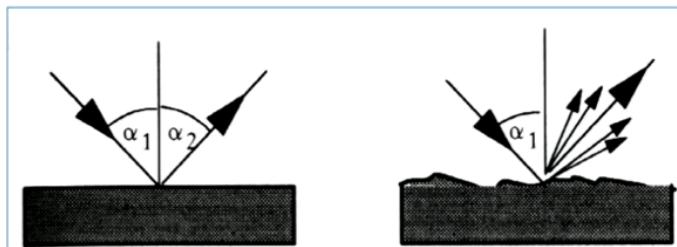


Abbildung 10: Einfluss der Bitanzahl auf die Bildqualität

The radiometric relation between the world and its projection is formed by:

- Amount of light that is reflected by a surface point = **Radiance**
- Amount of light that is projected from this point onto the image = **Irradiance**
- measured in watts per square meter (W/m^2),



Sampling

Bildauflösung – Überblick

- **Bildauflösung** = Maß für den **Detailreichtum** eines Bildes
- Abhängig von:
 - **Radiometrischer Auflösung**
 - **Sensorauflösung**
 - **Räumlicher (geometrischer) Auflösung**
 - **Zeitlicher Auflösung**

Radiometrische Auflösung

- Gibt an, wie fein ein System **Helligkeitsunterschiede unterscheiden** kann
- Beschrieben durch **Anzahl der Bits** (z. B. 8 Bit = 256 Graustufen)

- Mensch kann ~120 Grauwerte unterscheiden → 256 Graustufen genügen

$2^1 = 2$ levels (0,1)



$2^2 = 4$ levels (0,1,2,3)



$2^8 = 256$ levels (0-255)



$2^{12} = 4096$ levels (0-4095)

- Weniger Graustufen → geringerer Speicherbedarf, aber:

- Qualitätsverlust

- Contouring-Effekte (sichtbare Tonwertsprünge)

(siehe hier:

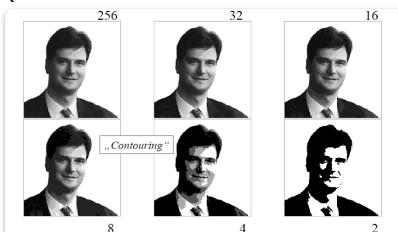


Abbildung 10: Einfluss der Bitanzahl auf die Bildqualität

)

Sensorauflösung

- Anzahl der **Pixel (Bildpunkte)** eines digitalen Bildes
 - z.B. $640 \times 480 = 307.200$ Pixel = ~0.3 MP
- Häufige Angabe: **Megapixel**
- Alternativ: **dpi / ppi** (dots/pixels per inch) – sinnvoll nur bei Scannern
- Für Kameras ungeeignet, da Objekt-Sensor-Abstand variabel

Räumliche (geometrische) Auflösung

- Fähigkeit, **benachbarte Objektstrukturen getrennt** zu erfassen
- Maßeinheiten:
 - Linien pro Millimeter (L/mm)** in Optik/Photographie
 - Definiert durch:
 - minimale Entfernung** zwischen zwei erkennbaren Merkmalen
 - minimale Größe** eines erkennbaren Merkmals

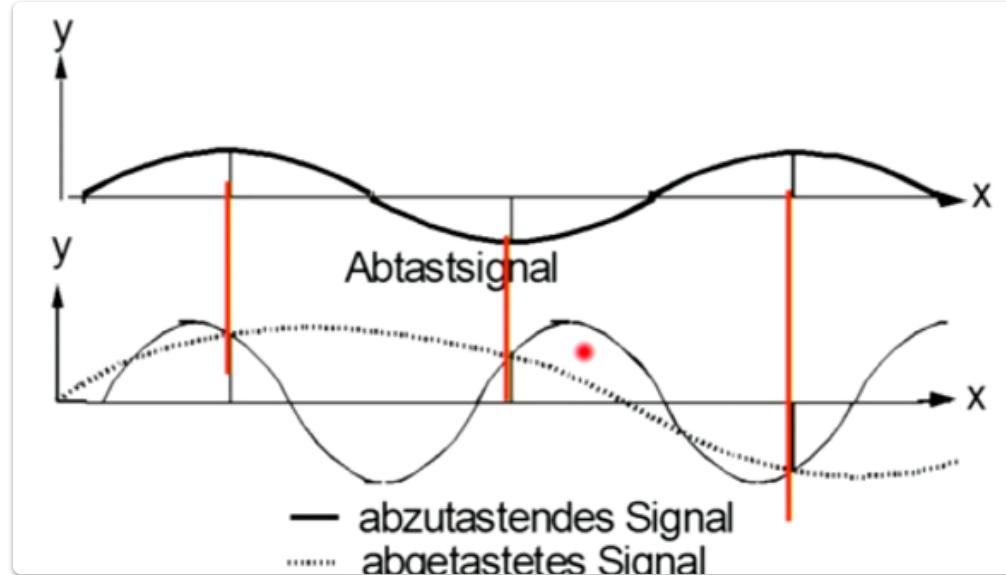
Zeitliche Auflösung

- Anzahl der aufgenommenen Bilder **pro Sekunde**
 - z.B. 25 fps (frames per second)
- Wichtig bei Videoaufnahmen oder Bewegungsanalysen

Sampling (Abtastung)

- Umwandlung** eines kontinuierlichen Signals (z.B. in Raum/Zeit) in eine **diskrete Folge**

- Zentrale Grundlage: **Nyquist–Shannon Sampling Theorem**
 - Ein bandbegrenztes Signal kann exakt rekonstruiert werden, wenn:
 - **Samplingrate > $2 \cdot B$** (B = höchste Frequenz im Signal)
 - d.h. $f_{abtast} > 2 \cdot f_{maxSignal}$



- Anwendung bei **analoger/digitaler Wandlung** in digitalen Kameras:
 - Raum (Pixelraster)
 - Zeit (fps)
 - Spektrum (z.B. Farbbandsampling)
- mehr Beispiele zu Nyquist: [7. Clipping und Antialiasing](#)

Bildsensoren

Allgemeines

- Ein **Bildsensor** wandelt ein **optisches Bild** in ein **elektronisches Signal** um
- Es gibt zwei Arten digitaler Pixelsensoren:
 - **CCD (Charge-Coupled Device)**
 - **CMOS (Complementary Metal-Oxide-Semiconductor)**

CCD-Sensoren

- Bestehen aus einer **Matrix lichtempfindlicher Fotodioden**
 - Zusätzlich enthalten: **MOS-Kondensatoren, Feldeffekttransistoren, Steuerleitungen, Leitungspfade**
- **Belichtungszeit**: Licht erzeugt elektrische Ladung in der Fotodiode
- Ladung wird im Kondensator gespeichert (parallel zur Fotodiode geschaltet)
- Auslesung erfolgt **zeilenweise**
- Spannung wird in der Kamera oder extern in **digitale Information** umgewandelt
- Arbeitet **linear**, wie analoge Kameras → basiert auf **Flächenintegral-Prinzip**

CMOS-Sensoren

- Besitzen **aktive Pixelsensoren**, jedes Pixel ist individuell auslesbar
- Spannung, die der **Helligkeit** entspricht, liegt **ständig an**
- Output ist **nicht-linear (logarithmisch)**
 - Ähnelt dem Prinzip der **menschlichen Bildwahrnehmung**
- Vorteile gegenüber CCD:
 - **Niedriger Energieverbrauch**
 - **Schnellere Lesegeschwindigkeit**
 - **Günstigere Herstellung**

Farbe auf CCD-Sensoren

Allgemeines

- **Bildsensoren** erfassen **nur Helligkeit**, nicht direkt Farbe
- Um **Farbinformation** zu erhalten, werden **Farbfilter** eingesetzt
- Drei gängige Methoden zur Farberfassung:
 - **Field Sequential Technik**
 - **3-Chip-Technik**
 - **Color Filter Array (CFA)**

Field Sequential Technik

- Ursprünglich 1900er Jahre, von **Produkin Gorski** entwickelt
- Drei **Schwarz-Weiß-Bilder** nacheinander aufgenommen – je mit **Rot-, Grün- oder Blaufilter**
- Darstellung durch **Lichtprojektion** (Laterna Magica)
- Nachteil: **Geisterbilder** bei Bewegung, da jede Farbe zu einem **anderen Zeitpunkt** aufgenommen wird
- Heute noch bei **Mikroskopen** in Verwendung

Sequential Color Three-Pass CCD Imaging System

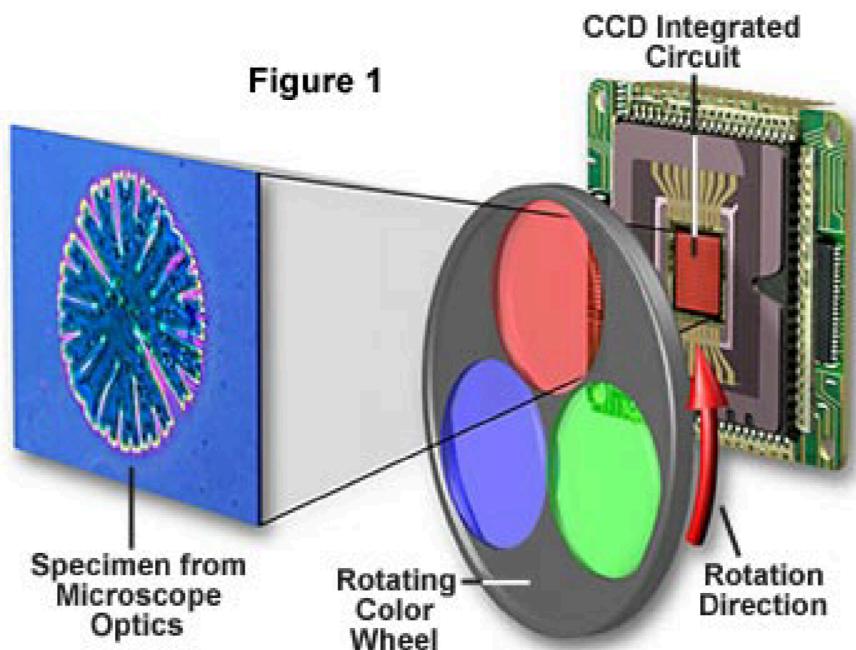


Abbildung 11: Farbaufnahme mittels Color Wheel

3-Chip-Technik

- Prisma trennt Licht in Rot, Grün, Blau (RGB)
- Jeder Farbkanal wird von eigenem CCD-Sensor aufgenommen
- Exakte Ausrichtung nötig (Der Lichtstrahl muss bei allen 3 Filtern den selben Pixel treffen), um Farbtreue zu garantieren
- nur eingesetzt bei Videokameras und Camcordern

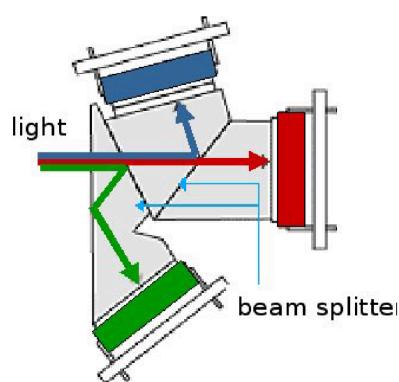


Abbildung 12: Farbtrennung durch Prisma

Color Filter Array (CFA)

- Häufigste Methode bei einzelnen CCD- oder CMOS-Sensoren
- Über jedem Pixel liegt ein Farbfilter (Mosaikstruktur)
- Bayer-Filter ist Standard:

- 50% Grün
- 25% Rot
- 25% Blau
- Anpassung an **menschliche Farbwahrnehmung** (grünempfindlichster Bereich)

Bayer-Muster und Demosaicing

- Output eines Bayer-Sensors = **Rohdatenbild** (Bayer-Muster)
- Jedes Pixel enthält **nur eine Farbkomponente**
- **Demosaicing**: Interpolation der fehlenden Farben
 - Nutzt Annahme: **Geringe Unterschiede** zwischen benachbarten Farbwerten
 - Einfache Verfahren: **Durchschnittswerte** benachbarter Pixel gleicher Farbe
 - Erweiterte Verfahren: Interpolation **entlang von Kanten**
 - Farbverhältnisse (z.B. Rot-Grün) bleiben konstant → erst Grün interpolieren, dann Rot und Blau

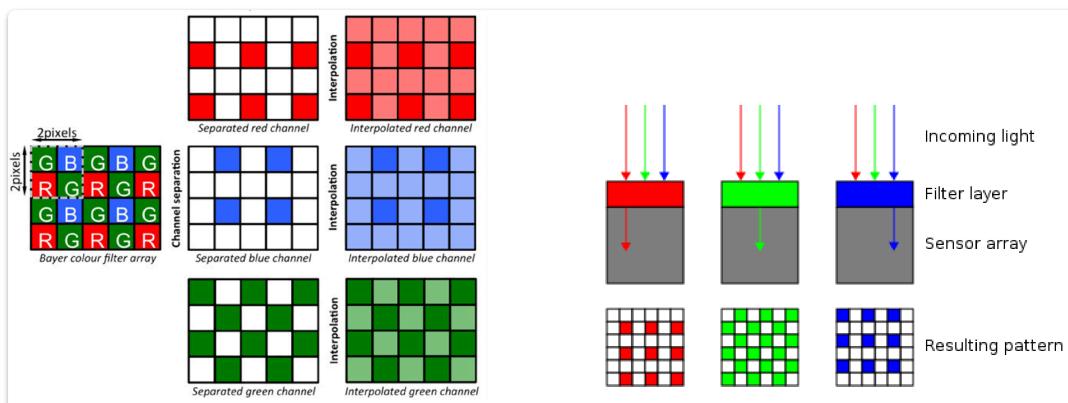


Abbildung 13: Prinzip des Bayer-Filters (links) und Ausführung mittels Filter pro Pixelsensor (rechts).

Auflösung und Verarbeitung

- Effektive **geometrische Auflösung** des Sensors beträgt **nur ein Drittel**
- Demosaicing kann erfolgen:
 - **In der Kamera** (z.B. JPEG, TIFF-Export)
 - **Extern** durch Programme wie **nomacs** (www.nomacs.org)

Foveon X3 Sensor

- Der **Foveon X3 Sensor** kombiniert die **3CCD-** und **CFA-Technik**
- Er nutzt ein **Array** von Fotodioden, wobei jede **vertikal gestapelt** ist und in einem **zweidimensionalen Raster** angeordnet wird
- Jede Fotodiode ist auf **verschiedene Wellenlängen** des Lichts empfindlich, was bedeutet, dass jede Diode eine **eigene spektrale Sensitivitätskurve** hat
- Das Signal der drei Fotodioden wird verarbeitet, um **drei additive Primärfarben** (Rot, Grün, Blau) zu erzeugen und so ein **fehlerfreies Farbbild** darzustellen

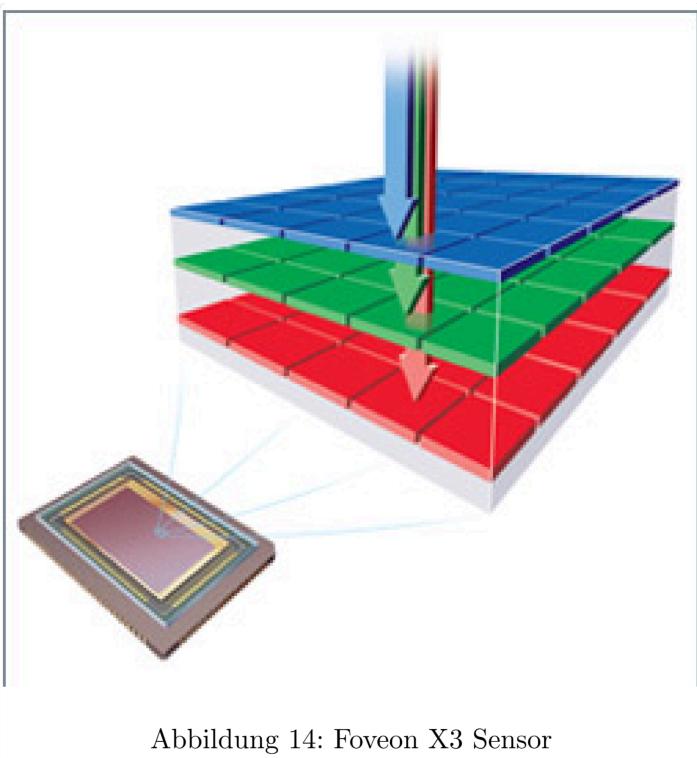


Abbildung 14: Foveon X3 Sensor

Multi-Shot-Technologie

- Eine **weitere Möglichkeit** zur Erzeugung von **Echtfarbbildern** ist der Einsatz von **beweglichen Sensoren**
- Bei der **Multi-Shot-Technologie** wird der Sensor mithilfe eines **Piezomotors** präzise in **Einpixel-Schritten** bewegt
- Diese Technik ermöglicht es der Kamera, **mehr Farbinformationen** und **Daten** zu erfassen, als bei einer Einzelbilddurchnahme
- Im **4-Shot-Modus** werden **vier Aufnahmen** gemacht:
 - **Sensorbewegung** erfolgt in vier Schritten (1 Pixel horizontal, 1 Pixel vertikal, erneut 1 Pixel horizontal und 1 Pixel vertikal)
 - Dieser Zyklus sorgt dafür, dass für jedes Pixel **echte RGB-Daten** erfasst werden
- Nach Beendigung des Zyklus kehrt der Sensor in seine **Ausgangsstellung** zurück

Ideale vs. Reale Kamera

Reale Kameras weichen von idealen Kameramodellen ab, da sie Linsen mit optischen Phänomenen und nicht-ideale Bildsensoren mit Chipartefakten verwenden. Einige dieser Störungen sind:

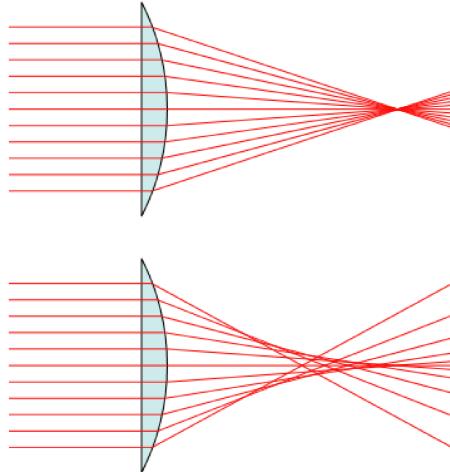


Abbildung 15: Optische Aberration

- Die **optische Aberration** tritt auf, wenn Licht von einem Punkt auf einem Objekt nicht in einem einzigen Punkt nach der Transmission durch die Linse zusammenfällt.
- Die **Chromatische Aberration** ist eine Art von Störung, bei der die Linse nicht im Stande ist, alle Farben im gleichen Konvergenzpunkt zu fokussieren. Diese Aberration tritt auf, da Linsen unterschiedliche Brechungsindizes für verschiedene Wellenlängen des Lichtes aufweisen. Chromatische Aberration manifestiert sich durch "Farbränder/-säume entlang der Abgrenzung zwischen dunklen und hellen Bildteilen."

- Die **sphärische Aberration** tritt auf, da Kugeloberflächen nicht die ideale Form für Linsen sind, aber beim Schleifen von Glas die einfachste Form darstellen. Diese Aberration führt dazu, dass Strahlen, die weiter am Rand der Linse auftreffen in einem etwas anderen Punkt fokussiert werden, als Strahlen die nahe der Achse liegen.
- Die **Linsenverzerrung** ist eine Abweichung einer geradlinigen Projektion, gerade Linien in einer Szene bleiben nicht gerade im Bild, diese können nach außen (tonnenförmig) oder nach innen (kissenförmig) verzerrt sein.

Weißabgleich

- Jede Kamera** geht bei der Aufnahme eines Fotos zunächst von einem **Durchschnittsbild** aus
- Die Kamera versucht, ein Bild zu erstellen, das dem **Durchschnitt** in Bezug auf Belichtung, Farbigkeit und mehr entspricht – was nicht immer korrekt ist
- Weißabgleich** hilft, die **Farbstimmung** oder **Farbtemperatur** anzupassen
- Der Weißabgleich sagt der Kamera, welche Farbe als **weiß** bzw. als **18%-iges neutrales Grau** angesehen werden soll
- Farbtemperatur** wird in **Kelvin** gemessen, z.B. hat **Licht von Leuchstoffröhren** ca. 4000 Kelvin, **normale Glühbirnen** etwa 3200 Kelvin und im **Schatten** um 7000 Kelvin
- Automatischer Weißabgleich:**
 - Die Kamera sucht eine neutrale Fläche und passt alle Farben an diese an
 - Dies funktioniert nicht immer perfekt, daher gibt es **verschiedene Anpassungsmöglichkeiten**

Optionen für den Weißabgleich

- Der **Weißabgleich** ist meist auf **Automatik** eingestellt

- **Voreingestellte Lichtsituationen** wie z.B. **Kunstlicht, Sonne, Schnee** oder **Schatten** stehen zur Auswahl
- **Manueller Weißabgleich:**
 - Ein weißes Blatt oder eine **18% Graukarte** wird fotografiert
 - Dieses Bild wird als **Referenz** verwendet, um den Weißabgleich korrekt zu erzeugen
- **Software:** Der Weißabgleich kann auch **nachträglich** angepasst werden
 - Hierbei wird ein **weißer Bereich** im Bild ausgewählt, und die Software berechnet den Weißabgleich
 - Beim **RAW-Format** funktioniert dies besonders gut, da der in der Kamera vorgenommene Weißabgleich nur als Vorschlag gespeichert wird und nachträglich geändert werden kann

No white balance



White balance



3. Bildcodierung und Kompression

Quellen:

- [EVC_Skriptum_CV, p.15](#) bis [EVC_Skriptum_CV, p.19](#)

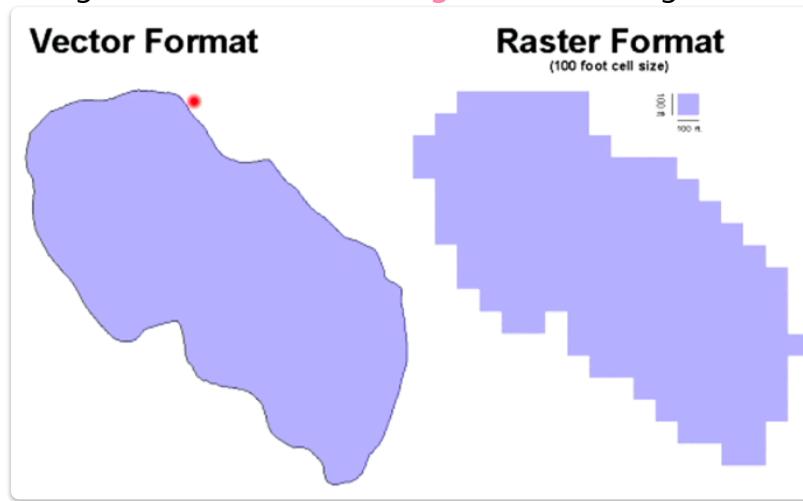
Digitales Bild-Dateienformat

Kontinuierliche vs. digitale Daten

- Wenn eine Zahl unendlich viele **mögliche Werte** annehmen kann, spricht man von **kontinuierlichen** oder **analogen** Daten
- **Computer** können mit analogen Daten nicht direkt arbeiten
- Daher müssen diese Daten **digitalisiert** werden, um sie für den Computer bearbeitbar zu machen
- Dieser Umwandlungsprozess erfolgt z.B. beim **Scannen von Fotos** oder beim Fotografieren mit **digitalen Kameras**

Digitale Bilder

- Ein digitales Bild ist eine **numerische Repräsentation** eines zweidimensionalen Bildes
- Das Bild kann entweder aus **Vektorbeschreibungen** oder einem **Raster** bestehen
- **Rasterbild:** Ein Raster von **diskreten Werten** (Pixel), bei dem jeder **Bildpunkt** mit seinem **Helligkeitswert** oder **Farbwert** gespeichert wird
- **Vektorbild:** Der Bildinhalt wird in **geometrischen Objekten** dargestellt und die Rasterung erfolgt erst bei der **Darstellung** auf einem Endgerät (z.B. **Display** oder **Drucker**)



Speicherung von Bilddaten

- Digitale Bildinformationen müssen in einem **Bilddatenformat** abgespeichert werden
- In der Frühzeit der digitalen Bildverarbeitung (bis etwa 1985) gab es eine große Anzahl unterschiedlicher **Dateiformate**, was zu vielen notwendigen **Konvertierungsprogrammen** führte

- Heute gibt es standardisierte **Dateiformate**, die den **Austausch** von Bilddaten erleichtern und die **langfristige Lesbarkeit** fördern

Speichergröße eines Rasterbildes

- Ein Rasterbild enthält ein **Pixelraster**, das für jede Rasterzelle eine bestimmte Anzahl an **Bits** zur Farbgebung bereitstellt – dies entspricht der **Farbtiefe**
- Ein Bild mit **LxN** (L Zeilen und N Spalten), **2B** (B = Anzahl der Bits pro Rasterzelle), **c** Farbkomponenten kann unkomprimiert als:
 - **L x N x B x c** gespeichert werden
 - Beispiel: Bei einer Bildgröße von **1024x768**, **3 Farbkomponenten (RGB)** und **256 (28) Graustufen** ergibt sich:
 - $1024 \times 768 \times 3 \times 8 = 18,87 \text{ MBit} \rightarrow 2,36 \text{ MByte}$
- **Bilddatengröße** korreliert positiv mit der Anzahl der **Pixel** und der **Farbtiefe** (Bits pro Pixel)
- **Prüfungs-ähnliches Beispiel** dazu:
 - Wie viel Speicherplatz benötigt man für die Speicherung des Bildinhaltes bei einem **RGB Farbbild** der Größe **1.024x768**, wenn pro Farbkanal **4.096** verschiedene Werte kodiert werden sollen?
 - $1.024 \times 768 = 786.432$ Pixel
 - $4.096 = 2^{12} \Rightarrow 12$ Bit/Pixel
 - RGB = **3** Farbkanäle
 - ⇒ **786.432 · 12 · 3 = 28.311.552 Bit**
 - ⇒ **$28.311.552 : 8 = 3.538.944 \text{ Byte}$**
 - ⇒ **$3.538.944 : 1.024 = 3.456 \text{ KiloByte (KB)}$**

$$\text{Size} = L \times N \times B \times c$$



Raster-Bildformate

Raw-Bildformat

- **Raw-Bildformat** ermöglicht es, auf die tatsächlich von der Kamera aufgenommenen Bilddaten zuzugreifen
- Speichert für **jeden Pixel** den entsprechenden **Farbwert** ohne **Nachbearbeitung** (bei 1-Chip-Kameras werden Rot-, Grün- und Blauwerte entsprechend dem **CFA-Muster** gespeichert)
- **RAW-Format** stellt kein „richtiges“ Farbbild dar, da **2/3 der Farbinformation** interpoliert werden müssen
- Muss zur **farbigen Anzeige** umgewandelt werden

Konventionelle Raster-Bildformate

- Beispiele für konventionelle Raster-Bildformate:
 - **Bitmap (BMP)**

- **Portable Network Graphics (PNG)**
- Ein modernes Raster-Bildformat ist in der Lage, **zweidimensionale digitale Bilder** beliebiger **Breite, Höhe** und **Auflösung** abzuspeichern

Struktur von Rasterbilddateien

- Eine Rasterbilddatei besteht aus **Strukturen fixer Größe (Header)** und **variabler Größe** (bildabhängig)
- Die Strukturen erscheinen in einer vordefinierten Sequenz
 - Beispiel: **BMP**: Der **Bitmap File Header** speichert allgemeine Informationen über die Bitmap-Datei (14 Bytes)
- **Metainformationen** werden in jedem individuellen Dateiformat gespeichert

Vektor-Bildformate

- **Vektor-Bildformat** beinhaltet eine **geometrische Beschreibung**, die problemlos für jede gewünschte **Anzeigegröße** gerendert werden kann
- **Rasterisierung**: An einem bestimmten Punkt müssen **alle Vektorgrafiken** rasterisiert werden, um auf einem digitalen Bildschirm angezeigt werden zu können
 - siehe: [5. Rasterisierung](#)

Plotter und Vektordaten

- **Plotter** sind Drucker, die Vektordaten zum **Zeichnen von Grafiken** verwenden

Computer Graphics Metafile (CGM)

- **CGM** ist ein freier und offener internationaler Standard für die Speicherung von **2D-Vektorzeichnungen, Rasterbildern und Text**
- Der Standard wird in Bereichen wie **technische Illustration, Kartografie, Visualisierung** und **elektronische Publikationen** verwendet
- Alle grafischen Elemente werden in **Quelltextdateien** spezifiziert, die anschließend zu einer **Binärdatei** oder **Textdarstellung** kompiliert werden
- **CGM** stellt Instrumente für den Austausch von Grafikdaten bei der Darstellung zweidimensionaler grafischer Informationen zur Verfügung, unabhängig von einer bestimmten **Anwendung, Plattform, System** oder **Gerät**

Windows-Metafile (WMF)

- **WMF** wurde 1990 entwickelt
- **WMF** ermöglicht den **Datenaustausch** zwischen Anwendungen und beinhaltet sowohl **Vektorgrafiken** als auch **Bitmap-Komponenten**

Bildkompression

- **Ziel der Bildkompression:** Reduzierung irrelevanter und redundanter Bildinformationen, um die Daten effizient zu speichern oder zu übertragen.
- **Arten der Kompression:**
 - **Verlustfrei (lossless):** Keine Daten gehen verloren, Bildqualität bleibt erhalten. Wird oft für medizinische Bilder, technische Zeichnungen oder Comics verwendet.
 - **Verlustbehaftet (lossy):** Daten gehen verloren, jedoch oft unmerklich. Wird für natürliche Bilder wie Fotografien verwendet, da es die Dateigröße stark reduziert.
- **Verlustbehaftete Kompression:**
 - Produziert **kompressionsartefakte** bei niedriger Bitrate.
 - In vielen Fällen als **visuell verlustfrei** bezeichnet, wenn der Verlust für den menschlichen Betrachter nicht wahrnehmbar ist.

Verlustfreie Datenkompression

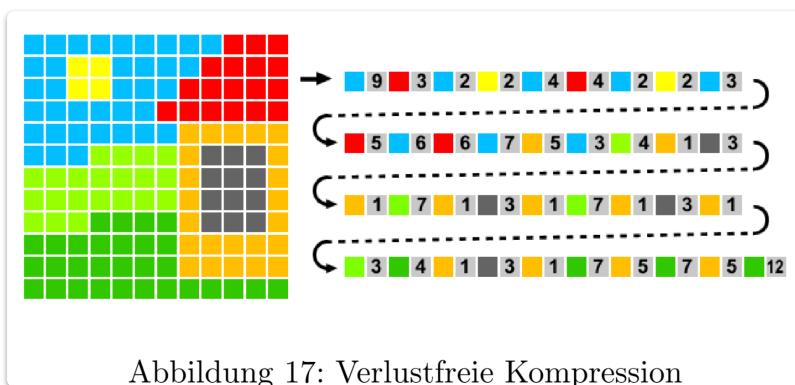


Abbildung 17: Verlustfreie Kompression

- Erlaubt eine exakte Rekonstruktion der Originaldaten. Wird in Bereichen genutzt, in denen die vollständige Datenintegrität wichtig ist (z. B. bei medizinischen Daten oder technischen Zeichnungen).

Prozess der Kompression

1. **Erstellung eines statistischen Modells** der Eingabedaten.
2. **Abbildung der Daten auf eine Bitreihe**, wobei häufig vorkommende Daten kürzere Bitfolgen erzeugen als seltene.

Run Length Encoding (RLE)

- Ein grundlegender Kompressionsalgorithmus, der **Datenwiederholungen** speichert.
- Beispiel: Eine lange Sequenz gleicher Farben (z. B. bei Icons, Linienzeichnungen).
- **Nachteil:** Für natürliche Bilder, die keine langen Wiederholungssequenzen haben, kann es zu einer **Vergrößerung der Dateigröße** führen.
- **Entropiecodierung:** Sie erstellt **kurze Codes für häufige Symbole** und **lange Codes für seltene Symbole**. Dies reduziert die durchschnittliche Länge der Codes und verbessert die Kompression.

Huffman-Codierung

- **Binärbaum:** Wird erstellt, bei dem die **Blätter** die Symbole und deren **Wahrscheinlichkeit (Häufigkeit)** enthalten. Die **Knoten** sind entweder Blätter oder interne Knoten.
- **Codeerstellung:**
 1. Beginnt mit den **Blättern**, die die Häufigkeit jedes Symbols enthalten.
 2. Zwei Knoten/Blätter mit den **geringsten Wahrscheinlichkeiten** werden zu einem neuen Knoten zusammengeführt. Der neue Knoten erhält eine Wahrscheinlichkeit, die der Summe der beiden Kinder entspricht.
 3. Der Vorgang wird wiederholt, bis nur noch ein Knoten übrig bleibt – der **Huffman Tree**.

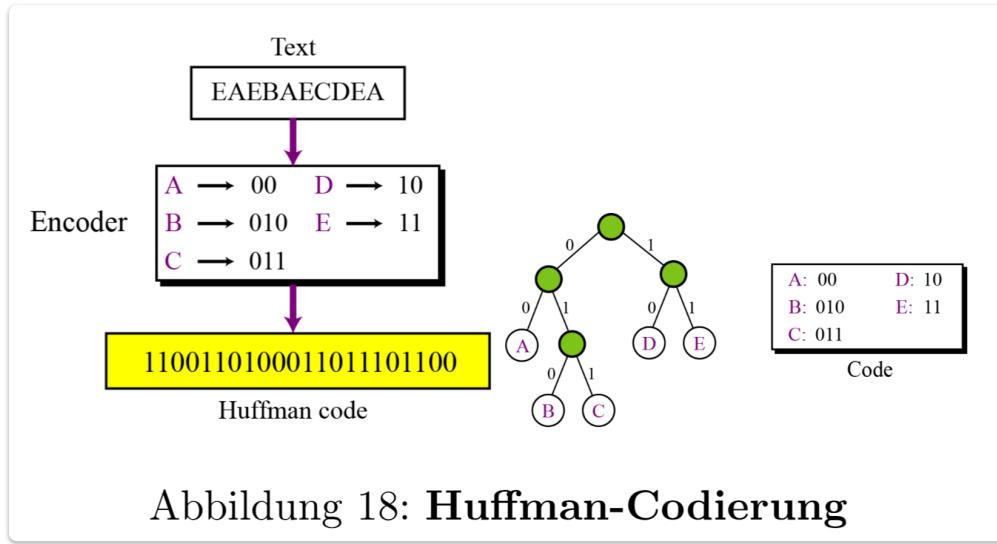


Abbildung 18: **Huffman-Codierung**

- **Ziel:** Der **Huffman Tree** ermöglicht die effiziente Zuordnung von **binären Codes** zu Symbolen, wobei häufige Symbole kürzere Codes erhalten und seltene Symbole längere.

Lempel-Ziv (LZ) Kompressionsverfahren

Das **Lempel-Ziv (LZ) Kompressionsverfahren** basiert auf der **Wiederholung** von Daten und speichert diese Wiederholungen als Referenzen in einer **Tabelle**.

- **Codierung:**
 - 8-Bit Datensequenzen werden als 12-Bit Code komprimiert.
 - Codes 0-255 repräsentieren **einzelne Zeichen** (1-Zeichen-Sequenzen).
 - Codes 256-4059 repräsentieren **Sequenzen**, die in einer Tabelle gespeichert sind.
- **Kompressionsprozess:**
 1. **Tabelle initialisieren:** Alle **1-Zeichen-Strings** werden zu Beginn als Einträge in der Tabelle gespeichert.
 2. **Längster String finden:** Der längste String **W**, der mit den aktuellen Eingabedaten übereinstimmt, wird identifiziert.
 3. **Tabellenindex ausgeben:** Der **Index für W** wird ausgegeben, und der String **W** wird vom Input entfernt.
 4. **Neuen String hinzufügen:** Der **neue String W + das nächste Symbol** wird zur Tabelle hinzugefügt.

5. **Wiederholen**: Der Prozess geht weiter, bis das gesamte Eingabedaten verarbeitet sind.

- **Ziel**: Der Algorithmus **reduziert die Daten**, indem häufig wiederholte Sequenzen als Referenzen gespeichert werden, anstatt sie mehrfach abzulegen.

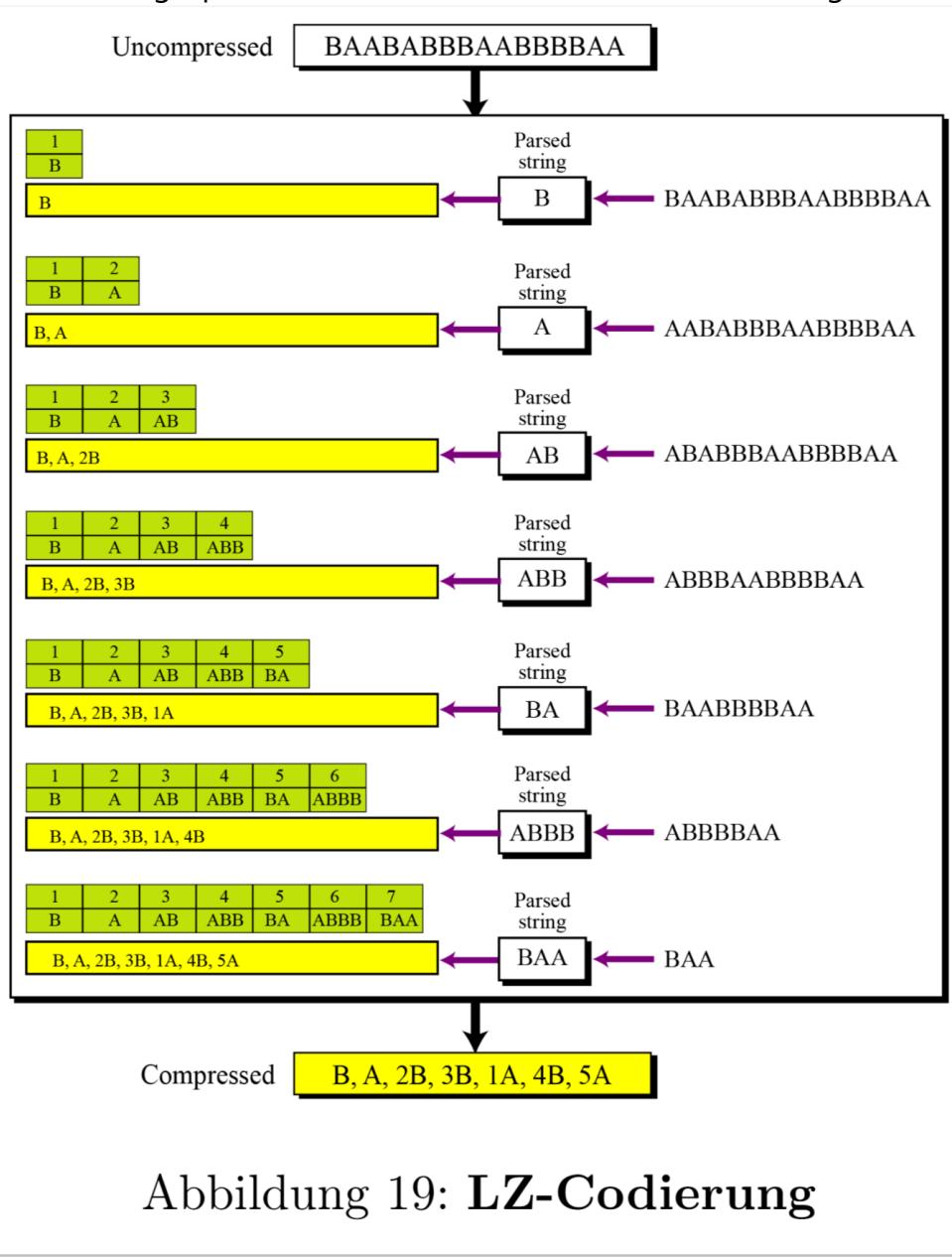


Abbildung 19: **LZ-Codierung**

Bildformate die Verlustfreie Kompression verwenden:

GIF (Graphics Interchange Format):

- **Einführung**: 1987 von CompuServe.
- **Farbe**: Unterstützt bis zu **256 Farben** (8 Bit pro Pixel) aus dem 24-Bit RGB Farbraum.
- **Verwendung**: Besonders geeignet für **kleine Bilder** (z. B. Icons) und **Animationen**.
- **Einschränkungen**: Weniger geeignet für **Fotografien** aufgrund der begrenzten Farbpalette.

PNG (Portable Network Graphics):

- **Entwicklung**: Entwickelt als Verbesserung und Ersatz für GIF.

- **Farbe:** Unterstützt **24-Bit RGB** oder **32-Bit RGBA** (mit Transparenz) sowie **Graustufen**.
- **Kompression:** Verlustfreie Kompression mittels **PKZIP**.
- **Verwendung:** Besonders für den **Webbereich** geeignet, aber nicht für **hochqualitative Druckgrafiken**.
- **Farträume:** Unterstützt nur **RGB**, keine CMYK.

TIFF (Tagged Image File Format):

- **Verwendung:** Beliebt bei **Grafikern, Fotografen, Verlagen und Wissenschaftlern**.
 - **Unterstützung:** Kann **Graustufenbilder, Indexbilder** und **Vollfarbenbilder** speichern.
 - **Besonderheit:** Kann mehrere **Bilder** mit unterschiedlichen Eigenschaften in einer Datei speichern.
 - **Kompression:** Unterstützt **verschiedene Kompressionsverfahren** (z. B. **LZW, ZIP, JPEG, CCITT**).
 - **Verwendung:** Wird häufig für **Dokumentenarchivierung, wissenschaftliche Anwendungen** und in der **Digitalfotografie** verwendet.
-

Verlustbehaftete Datenkompression

- **Prinzip:** Entfernung von Datenteilen, die für das menschliche Wahrnehmungssystem nicht oder nur schwer erkennbar sind.
- **Ziel:** Die Daten werden so komprimiert, dass möglichst wenig Qualität verloren geht, aber die Dateigröße erheblich reduziert wird.
- **Transformation:** Daten werden in eine neue Domäne umgewandelt, die die relevanten Informationen effizienter darstellt.

JPEG-Kompression:

- **Entwickelt:** Von der **Joint Photographic Experts Group (JPEG)**, 1990 als **ISO-Standard** etabliert.
- **Ziel:** Durchschnittliche Datenreduktion von **1:16**, ideal für **fotografische Bilder**.

Codierungsprozess in JPEG:

1. Farbraumkonversion und Downsampling:

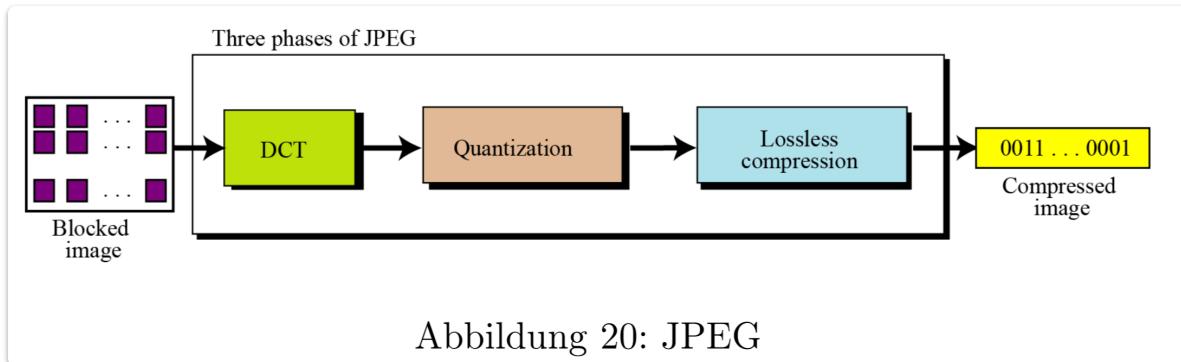
- Das Bild wird von **RGB** (Rot, Grün, Blau) in den **YCbCr-Farbraum** umgewandelt, wobei Y die Helligkeit (Luminanz) und Cb sowie Cr die Farbinformationen (Chrominanz) darstellen.
- **Downsampling** der Chrominanzkanäle (Cb und Cr) erfolgt, da das menschliche Auge weniger empfindlich auf Farbdetails als auf Helligkeitsdetails reagiert.

2. Kosinustransformation und Quantisierung:

- Das Bild wird in **8x8 Blöcke** unterteilt, und für jeden Block wird eine **diskrete Kosinustransformation (DCT)** durchgeführt, um die Frequenzen des Bildes zu berechnen.
- Die resultierenden **Spektralkoeffizienten** werden **quantisiert**, wobei hohe Frequenzen stärker reduziert werden, da diese weniger zur Wahrnehmung der Bildschärfe beitragen.

3. Verlustfreie Kompression:

- Nach der Quantisierung wird der Datenstrom mittels **verlustfreier Kompression** (z.B. **Lauflängenkodierung** oder **Huffman-Kodierung**) weiter komprimiert, um die Dateigröße zu minimieren, ohne zusätzliche Informationen zu verlieren.



Das JPEG-Verfahren nutzt also **wahrnehmungspsychologische Erkenntnisse** zur Reduktion von Bilddaten und ermöglicht eine hohe Kompressionsrate bei gleichzeitig guter Bildqualität.

Diskrete Cosinus Transformation (DCT)

DCT (Diskrete Kosinustransformation):

- Die DCT ist eine Variante der **Fouriertransformation**, die Signale (hier Bildpixel) in **Cosinuswellen** unterschiedlicher **Frequenz** und **Amplitude** zerlegt.
- Ziel: **Frequenz- und Amplitudenverteilung** der Bildpixel sichtbar zu machen, um zu verstehen, welche Bildteile hohe oder niedrige Frequenzen enthalten.

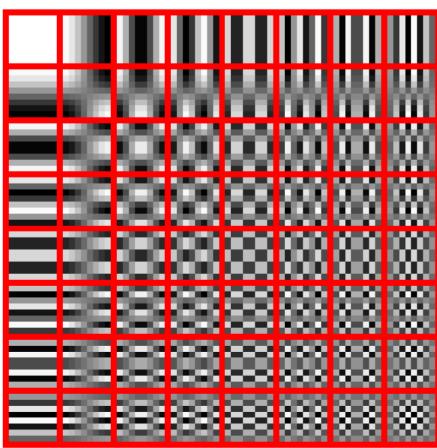


Abbildung 21: Die DCT ist eine Variation der Fouriertransformation.

- Formel für die 2D-DCT (für einen 8x8 Block):

$$F(u, v) = \alpha(u) \cdot \alpha(v) \sum_{x=0}^7 \sum_{y=0}^7 f(x, y) \cos\left(\frac{(2x+1)u\pi}{16}\right) \cos\left(\frac{(2y+1)v\pi}{16}\right)$$

- u und v : **Horizontale** bzw. **vertikale Ortsfrequenz** ($0 \leq u, v < 8$).
- $f(x, y)$: Pixelwert am Punkt (x, y) .
- $F(u, v)$: DCT-Koeffizient, der das Signal in Frequenzkomponenten zerlegt.
- $\alpha(u)$ und $\alpha(v)$: **Skalierungsfaktoren** zur Wahrung der Orthonormalität.

JPEG-Kompression:

- **Quantisierung**: Die zentrale Methode der **verlustbehafteten** Kompression bei JPEG. Jeder DCT-Koeffizient wird durch einen vordefinierten Quantisierungswert geteilt und auf den nächsten Integer gerundet.
 - **Häufige Quantisierungswerte**: Niedrige Frequenzen erhalten **kleinere Quantisierungswerte** (präziser), während **hohe Frequenzen** größere Werte erhalten (weniger präzise), da das menschliche Auge weniger empfindlich gegenüber hohen Frequenzen ist.
 - Das führt dazu, dass **hochfrequente Komponenten auf 0 gerundet werden** und **niedrigere Frequenzen eine hohe Genauigkeit behalten**.
- **Ergebnis**: Die Quantisierung reduziert die Bildgröße, indem sie **unnötige Bilddetails entfernt**, insbesondere bei den hohen Frequenzen.
 - **Visuell kann die Kompression das Bild auf ein Fünftel seiner Originalgröße reduzieren**, ohne dass große visuelle Qualitätseinbußen sichtbar werden.

Kompressionsartefakte:

- Bei zu starker **Kompression** (zu hoher Quantisierung) können **Blockartefakte** auftreten, da das Bild in 8x8-Blöcke unterteilt wird und hohe Frequenzen unzureichend dargestellt

werden.

- **Schwächen:**

- JPEG zeigt Schwächen bei **abrupten Übergängen** (wie Kanten oder Text).
- **Blockbildung:** Bei sehr starker Kompression können die 8x8 Blöcke sichtbar werden, was das Bild unnatürlich erscheinen lässt.

Optimierung für natürliche Bilder:

- JPEG wurde speziell für **natürliche fotografische Bilder** entwickelt und ist nicht ideal für **Computergrafiken** oder **Bilder mit scharfen Kanten**, bei denen die Blockbildung besonders auffällt.

Video Kompression

- Ziel: **Reduzierung der Redundanz** in Videodaten, um die **Datenmenge** zu verringern und effizienter zu speichern oder zu übertragen.
- Kombination aus **räumlicher Bildkompression** (ähnlich wie bei Bildern) und **zeitlicher Bewegungskompensation** (bezieht sich auf die Bewegung zwischen den Frames).

Techniken:

1. Verlustbehaftete Kompression:

- Entfernt große Mengen an Daten, aber der visuelle Unterschied ist oft **kaum erkennbar**.
- Es gibt einen **Kompromiss** zwischen Videoqualität, Kompressionsaufwand und den Systemanforderungen.

2. Räumliche Bildkompression:

- Komprimiert das Bild innerhalb eines einzelnen Frames (ähnlich wie JPEG).

3. Zeitliche Bewegungskompensation:

- Verwendet **Makroblöcke** (quadratische Bildausschnitte), die Unterschiede zwischen Frames messen.
- Bei viel Bewegung im Video müssen mehr Daten codiert werden, da mehr Pixel sich zwischen den Frames ändern.

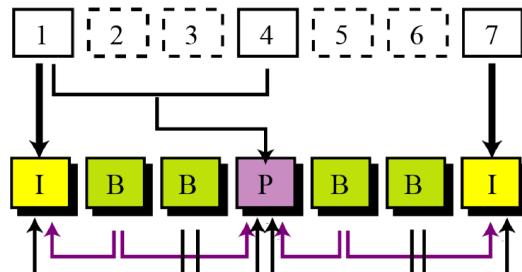
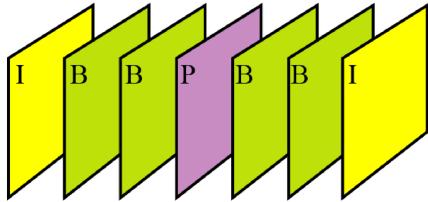
Interframe- und Intraframe-Kompression:

- **Interframe-Kompression:**

- Verwendet **Frames davor und danach** (B/P-Frames) zur Kompression.
- Beispiel: Ein Frame wird nur dann gespeichert, wenn sich etwas verändert hat, ansonsten wird er durch ein Referenzbild ersetzt.

- **Intraframe-Kompression:**

- Komprimiert nur den aktuellen Frame (ähnlich wie Bildkompression, z.B. JPEG).



Video-Kompressionstechniken:

- Veränderungen innerhalb eines Frames:** Wenn sich ganze Makroblöcke eines Frames verändern, kann der Kompressor Anweisungen wie **Verschieben, Rotieren oder Aufhellen** an den Dekompressor senden, um die Veränderung zu rekonstruieren.
- Interframe-Kompression:** Komprimiert Bereiche, die sich nicht verändert haben, durch **einfachen Verweis auf den vorherigen Frame**.

MPEG-Kompression:

- MPEG** (Moving Picture Experts Group) ist eine weit verbreitete Technik zur Video- und Audiokompression.
- Asymmetrisch**: Codierung ist algorithmisch komplexer als Dekodierung (Vorteil im Broadcasting, da viele billige Dekodierer und wenige teure Codierer benötigt werden).

MPEG-Standards:

- MPEG-1:**
 - Entwickelt für **Video CDs, SVCDs und DVDs** mit niedriger Videoqualität.
 - Ziel war es, Film und Ton auf die Bitrate einer **Compact Disc** zu kodieren.
- MPEG-2:**
 - Unterstützt **Zeilensprungverfahren** (Interlaced) und **High Definition** Auflösung.
 - Wichtig für **digitales Fernsehen, Kabelsignale und DVDs**.
- MPEG-4:**
 - Bietet **effizientere Codierung** und eignet sich auch für **Computergrafik-Applikationen**.
 - Wird neben MPEG-2 auch für **Blu-ray Discs** verwendet.

4. Punktoperationen

Quellen:

- [EVC_Skriptum_CV, p.20](#) bis [EVC_Skriptum_CV, p.23](#)

Was sind Punktoperationen?

- **Definition:** Punktoperationen betreffen nur die Werte der einzelnen Bildelemente und verändern nicht die Größe, Geometrie oder Struktur des Bildes. Der neue Pixelwert hängt nur vom Wert des ursprünglichen Pixels an derselben Position ab.
- **Formel:** Der neue Wert $I'(u, v)$ wird durch eine Funktion f des Originalwertes $I(u, v)$ bestimmt:

$$I'(u, v) = f(I(u, v))$$

Der neue Wert hängt also nur von den ursprünglichen Pixelwerten und eventuell von konstanten Parametern ab.

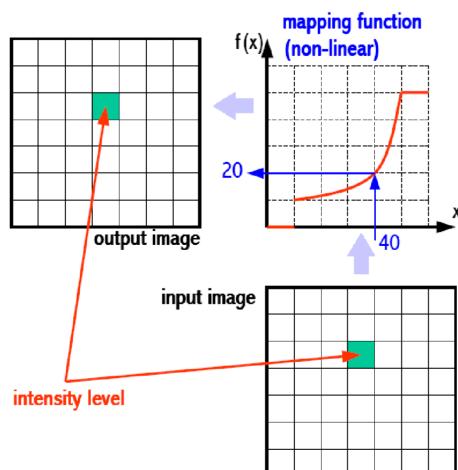


Abbildung 23: Beispiel Punktoperati-
on

[Unterklassen der Punktoperatoren:](#)

1. Homogene Punktoperatoren:

- Die Funktion f ist unabhängig von den Bildkoordinaten, d.h., die Operation ist für alle Bildpositionen gleich.
- Beispiele:
 - [Helligkeits- und Kontraständerungen](#)
 - [Bildinvertierung](#)
 - [Quantisierung der Bildhelligkeit](#)
 - [Schwellwertbildung](#)
 - [Gammakorrektur](#)

- **Farbtransformationen**

- Diese Operationen verändern die Pixelwerte, aber nicht deren Position im Bild.

2. Inhomogene Punktoperationen:

- Diese Art der Punktoperation berücksichtigt zusätzlich die Bildkoordinaten (u, v) .
- Sie können beispielsweise bei der Verarbeitung von Bildteilen oder bei der Anwendung von Filtern mit Bildpositionen variieren.

Affine Punktoperatoren:

- Diese sind eine spezielle Unterklasse der homogenen Punktoperatoren.
- Sie lassen sich durch eine lineare Gleichung beschreiben:

$$I'(u, v) = a \cdot I(u, v) + b$$

- a und b sind Konstanten. Je nach Wahl von a und b kann man unterschiedliche Bildveränderungen erzielen:
 - **Helligkeitsveränderung:** $a = 1, b \neq 0$
 - **Kontrastveränderung:** $a \neq 1$

Abbildungsfunktion (Mapping Function):

- Die Abbildungsfunktion ordnet jedem Pixelwert im Eingangsbild einen neuen Helligkeitswert im Ausgangsbild zu.
- Sie kann verschiedene Formen annehmen:
 - **Linear**
 - **Stufenweise linear**
 - **Nichtlinear** (z.B. Gammakorrektur, welche typischerweise eine nichtlineare Funktion darstellt).

Beispiel einer nichtlinearen Abbildungsfunktion:

- In einem Beispiel könnte die Abbildungsfunktion dem Eingangswert 40 den Ausgangswert 20 zuordnen. Dies zeigt, dass die Zuordnung der Ausgangswerte nicht immer direkt proportional oder linear zum Eingangswert ist.

Identitätsfunktion und Invertierung

Die einfachste Abbildungsfunktion ist die **Identitätsfunktion**, alle Werte behalten denselben Wert, die **Kennlinie verläuft von links unten nach rechts oben**. Die **Invertierung** ist eine einfache affine Punktoperation, die einerseits die Ordnung der Pixelwerte (durch **Multiplikation mit -1**) umkehrt und andererseits durch Addition eines konstanten Intensitätswerts dafür sorgt, dass das Ergebnis innerhalb des erlaubten **Wertebereichs bleibt**. Für ein Bild $I(u, v)$ mit dem maximalen Wertebereich $[0, q]$ ist die zugehörige Operation daher: $\tilde{I}(u, v) \rightarrow -I(u, v) + q = q - I(u, v)$.



Abbildung 24: Identitätsfunktion und Invertierung

Schwellwertoperation

Eine Schwellwertoperation (engl. *thresholding*) ist eine spezielle Form der Quantisierung, bei der die Bildwerte in zwei Klassen p_0 und p_1 getrennt werden, abhängig von einem vorgegebenen Schwellwert (engl. *threshold value*) p_{th} :

$$I'(u, v) \leftarrow f_{th}(I(u, v)) = \begin{cases} p_0 & \text{for } I(u, v) < p_{th} \\ p_1 & \text{for } I(u, v) \geq p_{th} \end{cases}$$



Abbildung 25: Schwellwertoperation

wobei $0 < p_{th} \leq q$. Eine häufige Anwendung ist die Binarisierung von Grauwertbildern mit $p_0 = 0$ und $p_1 = 1$.

Kontrast und Helligkeit

Um den Kontrast in einem Bild um 50% (d.h. um den Faktor 1,5) zu erhöhen, wird dies durch eine affine Punktoperation ausgedrückt:

$$I'(u, v) = I(u, v) \cdot 1.5$$

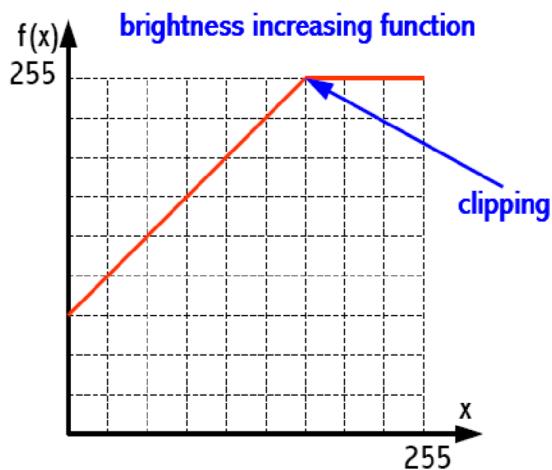
Hierbei wird der Pixelwert $I(u, v)$ des ursprünglichen Bildes mit dem Kontrastfaktor 1.5 multipliziert, was den Kontrast im Bild entsprechend anhebt.

Wichtiger Hinweis: Bei der Anwendung solcher Operationen muss der vorgegebene Wertebereich der Bildpixel beachtet werden. Für **8-Bit-Grauwertbilder** liegt der zulässige Pixelwertbereich zwischen **0** und **255**.

Problematik:

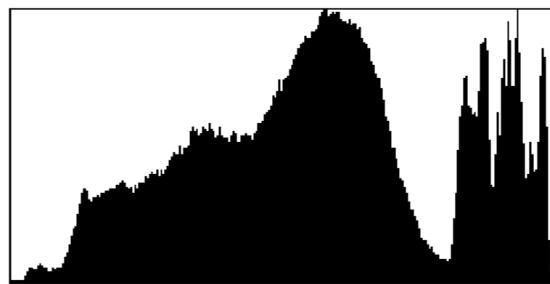
- **Clipping:** Wenn nach der Kontrastanhebung der berechnete Pixelwert **größer als 255** ist, wird dieser Wert auf den Maximalwert **255** begrenzt.
- Ebenso wird, wenn der Pixelwert **kleiner als 0** wird, dieser Wert auf den **Minimalwert 0** begrenzt, um negative Werte zu vermeiden.

Das Clipping verhindert, dass der Wertebereich überschritten wird, wodurch die Berechnungen im zulässigen Bereich bleiben. Diese Begrenzung (Clipping) ist notwendig, um eine korrekte Darstellung und Bildverarbeitung zu gewährleisten, da Bildpixel keine Werte außerhalb des zulässigen Bereichs annehmen können.



Histogramm

- **Definition:** Ein Histogramm ist eine grafische Darstellung der Häufigkeit von Pixelwerten in einem Bild.
- **Verwendung:** Wird zur Analyse von **Belichtungsfehlern**, **Bildverarbeitungsschritten** und **Bildqualität** genutzt.
- **Funktionsweise:** Zeigt die Häufigkeit von Grauwerten oder Farbwerten auf der vertikalen Achse und die Grauwert- bzw. Farbintensitäten auf der horizontalen Achse.
- **Einschränkungen:** Es ist **nicht möglich**, das Originalbild ausschließlich aus dem Histogramm zu rekonstruieren, da viele verschiedene Bilder dasselbe Histogramm aufweisen können.



Grauwert-Histogramme

- **Verwendung:** Besonders nützlich für **Graustufenbilder**.
- **Berechnung:** Zeigt, wie häufig jeder Grauwert x im Bild vorkommt.
- **Beispiel:** In einem 8-Bit-Bild ist das Histogramm von 0 bis 255 (für jeden Grauwert) unterteilt, wobei die vertikale Achse die Häufigkeit der Vorkommen dieses Grauwerts darstellt.

- **Verwendung in der Praxis:** Kann für die **Beurteilung von Belichtungsfehlern** oder die **Optimierung von Bildoperationen** verwendet werden.
- Definition: $H(x) = \text{card}\{(u, v) | I(u, v) = x\}$ für jedes $x \in \{0, \dots, q\}$

Farbhistogramme

- **Verwendung:** Werden für **Farbbilder** verwendet, z. B. im **RGB-** oder **HSV-Farbmodell**.
- **Berechnung:** Hierbei wird die Häufigkeit der Pixelwerte für jeden Farbkanal berechnet.
- **Typen:** Häufig verwendete Varianten sind das **RGB-Histogramm** und das **HS-Histogramm**.
- **Einschränkungen:** Bei 24-Bit Farbbildern können die Histogramme sehr groß werden, daher werden oft **zweidimensionale** Histogramme verwendet (z. B. **HS-Histogramm** für den Farbton und die Sättigung).

Auswirkungen von Bildoperationen auf Histogramme

- **Helligkeitserhöhung:** Verschiebt das gesamte Histogramm nach rechts.
- **Kontrasterhöhung:** Macht das Histogramm breiter und verteilt die Pixelwerte über einen größeren Bereich.
- **Invertieren des Bildes:** Spiegelt das Histogramm entlang der vertikalen Achse.
- **Verschmelzen von Histogrammlinien:** Führt zu einem Verlust von Bilddynamik und Information, wenn unterschiedliche Pixelwerte zusammengeführt werden.

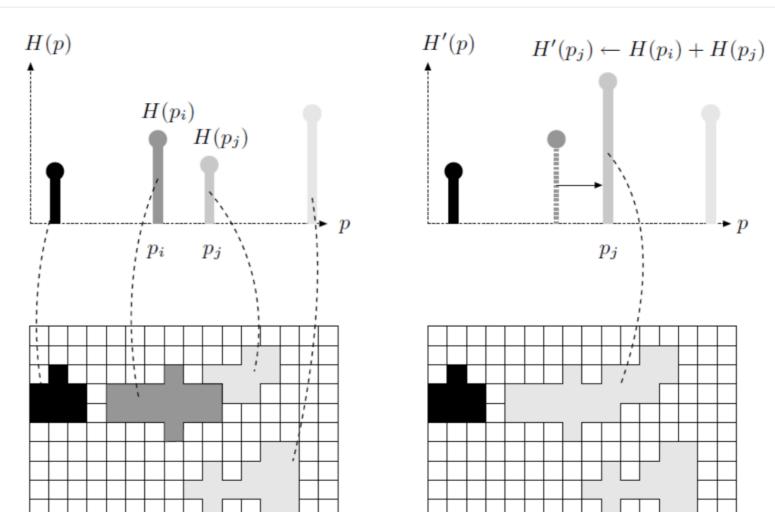


Abbildung 27: Histogrammbestimmung

Verlust von Bildinformationen

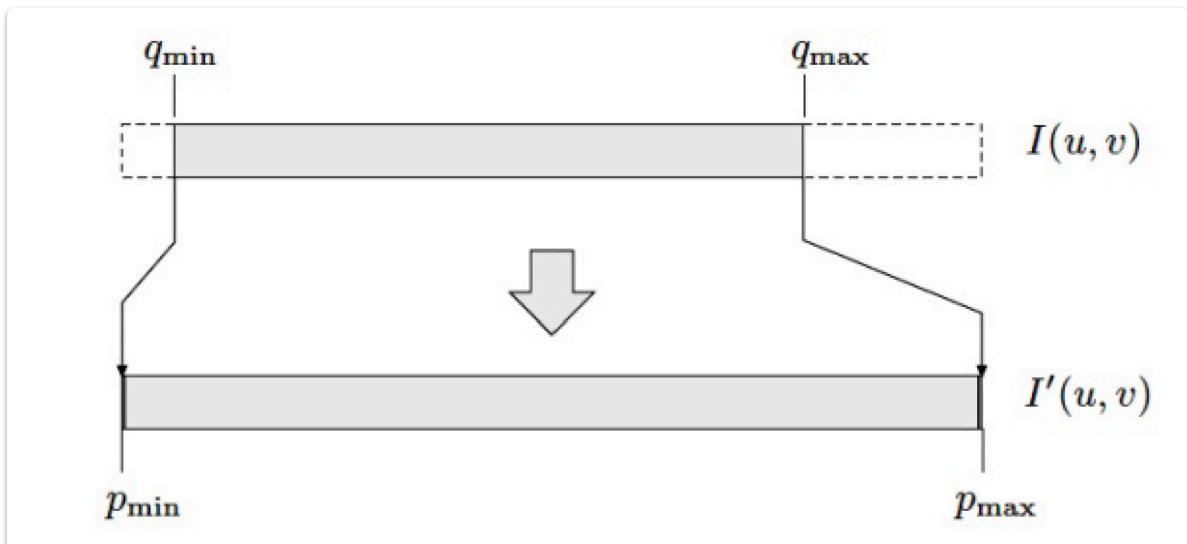
- **Clipping:** Wenn Werte außerhalb des definierten Bereichs (z. B. 0-255 für 8-Bit-Graustufenbilder) berechnet werden, werden diese auf den maximalen oder minimalen Wert beschränkt.
 - **Verlust durch Punktoperationen:** Wenn durch eine Bildoperation zwei Histogrammlinien zusammenfallen, werden diese Pixel nicht mehr voneinander unterschieden, was zu einem Verlust an Bilddynamik führen kann.
-

Histogrammnormalisierung

- **Ziel:**
 - Erhöhung des Kontrasts durch lineare Umverteilung der Pixelwerte.
 - Maximale Ausnutzung des Intensitätsbereichs.
- **Prinzip:**
 - Dunklerster Pixelwert (q_{min}) → niedrigster Intensitätswert (0).
 - Hellster Pixelwert (q_{max}) → höchster Intensitätswert (z. B. 255).
 - Lineare Verteilung der restlichen Pixelwerte dazwischen.
- **Formel zur Umrechnung:**

$$I'(u, v) = \frac{(I(u, v) - q_{min})}{(q_{max} - q_{min})} \cdot q$$

- $I(u, v)$: Pixelwert im Originabild.
 - q_{min} : Kleinster Pixelwert im Originalbild.
 - q_{max} : Größter Pixelwert im Originalbild.
 - q : Maximal möglicher Pixelwert im Zielbild (255 für 8-Bit Graustufenbilder).
 - $I'(u, v)$: Neuer Pixelwert im Bild nach der Normalisierung.



- **Problem der geringen Robustheit:**
 - **Ausreißer** (ein Pixel mit Intensität 0 oder q) beeinflussen die gesamte Streckung.

- Das führt dazu, dass viele Pixelwerte in einem engen Bereich bleiben und die Transformation weniger effektiv wird.

Vermeidung von Ausreißern durch Quantile

- **Lösung:** Bestimmung von **qmin** und **qmax** über **Quantile** statt extremen Werten.
- **p-Quantil:** Der Wert, unter dem **p*100%** der Werte einer Verteilung liegen.
 - Beispiel: **0.005-Quantil** (für 0.5% der Pixel an beiden Enden des Intensitätsbereichs).
- **Akkumulierte Histogramm (Ha):**

$$Ha(x) = \sum_{k=0}^x H(k)$$

- **H(k):** Häufigkeit des Grauwerts **k** im Histogramm.
- **Anwendung:**
 - In Programmen wie **Adobe Photoshop** werden Ausreißer durch Quantile minimiert.
 - **Auto-Kontrast:** Häufige Anwendung der Histogrammnormalisierung mit Quantilen.

Abbildung 28:

- Zeigt die Auswirkungen der Histogrammnormalisierung auf das Histogramm.
- **Regelmäßige Lücken** im Histogramm, die durch die lineare Streckung des Wertebereichs entstehen.

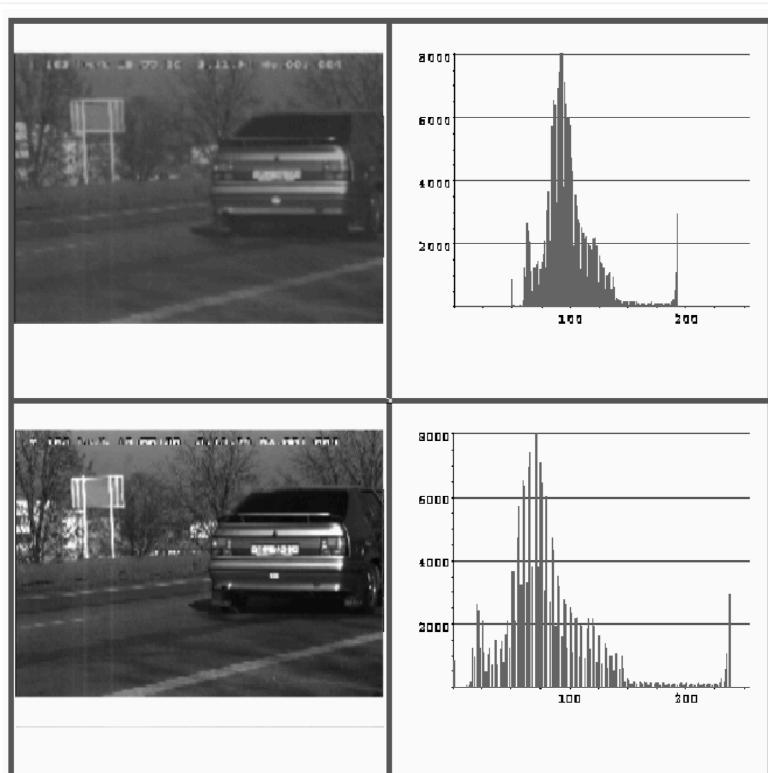


Abbildung 28: Histogrammnormalisierung

Test-ähnliches Beispiel:

Histogram Normalization Beispiel



- Angenommen, ein 10-Bit-Grauwertbild $I(u,v)$ weist einen minimalen Intensitätswert von 50 und einen maximalen Intensitätswert von 200 auf. Wie lautet in diesem Fall die affine (lineare) Punktoperation, die den Kontrast des Bildes auf den gesamten Intensitätsbereich verstärkt?

$$I'(u,v) = q \cdot \frac{I(u,v) - q_{\min}}{q_{\max} - q_{\min}}$$

Einsetzen in Formel:

$q = 10\text{-bit} = 1024$ Grauwerte = **1023**

$q_{\min} = 50$

$q_{\max} = 200$

$$I'(u,v) = \frac{1023((I(u,v) - 50))}{150}$$

Histogram Normalization Beispiel



- Angenommen, ein 10-Bit-Grauwertbild $I(u,v)$ weist einen minimalen Intensitätswert von 50 und einen maximalen Intensitätswert von 200 auf. Wie lautet in diesem Fall die affine (lineare) Punktoperation, die den Kontrast des Bildes auf den gesamten Intensitätsbereich verstärkt?

2. Möglichkeit:

Lineares Gleichungssystem lösen:

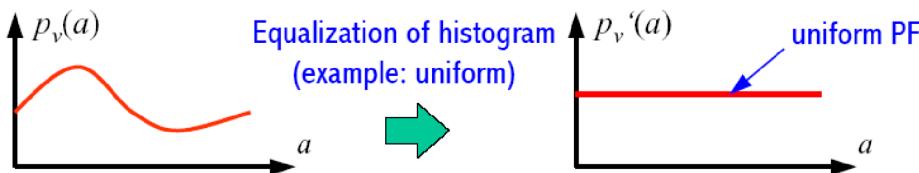
$$y = kx + d$$

$$0 = k \cdot 50 + d$$

$$1023 = k \cdot 200 + d$$

$$\Rightarrow d = 341, k = -6,82$$

Histogrammausgleich



- Ziel:**

- Erzeugung eines **gleichmäßig verteilten Histogramms** im Ergebnisbild durch eine homogene Punktoperation.
- Anwendung:** Verbesserung des Kontrasts in Bereichen mit stark vertretenen Grauwerten.

- Unterschied zur Histogrammnormalisierung:**

- Histogrammausgleich zielt darauf ab, **häufige Grauwertbereiche auseinanderzuziehen**
- Histogrammnormalisierung hingegen **streckt den gesamten Wertebereich gleichmäßig**.

- Prinzip:**

- **Ziel:** Annäherung an ein gleichverteiltes Histogramm (perfekte Gleichverteilung ist nicht möglich).
 - Einzelne **Spitzen im Histogramm** (stark konzentrierte Grauwertbereiche) können nicht vollständig entfernt werden, sondern werden nur auseinandergezogen.
 - **Veränderung des Kontrasts:** Es wird der Kontrast **in stark vertretenen Grauwertbereichen erhöht.**
-

Berechnung des Histogrammausgleichs

- **Akkumuliertes Histogramm (H_a):**
 - Das akkumulierte Histogramm wird verwendet, um das Bildkontrast zu modifizieren.
- **Normalisiertes, akkumuliertes Histogramm (H_n):**
 - Berechnung:
$$H_n(x) = \frac{q}{H_a(q)} \cdot H_a(x)$$
 - **$H_a(q)$:** Akkumuliertes Histogramm an der Stelle **q**.
 - **N:** Gesamtzahl der Pixel im Bild.
 - **q:** Maximaler Grauwert des Bildes (z. B. 255 bei 8-Bit).
- **Ziel:**
 - Durch diese Berechnung wird das akkumulierte Histogramm auf den **normalisierten Bereich zwischen 0 und q** abgebildet.

H_n dient als Lookup-Table (Umsetzungstabelle: die Werte einer Funktion werden vorab ermittelt und als Tabelle abgelegt) für die Neuordnung der Grauwerte. Der Ausgleich bewirkt, dass jedem Pixel mit Grauwert x der p -te Anteil des maximal kodierbaren Grauwerts q zugeordnet wird. Dabei bezeichnet p die relative Häufigkeit, mit der alle Grauwerte von 0 bis einschließlich x im Eingabebild vorkommen. Die Histogrammequalisierung ist - im Gegensatz zur Spreizung und Histogrammdehnung - keine affine Punktoperation. In Abbildung 29 ist eine Histogrammequalisierung veranschaulicht. Im Gegensatz zur Histogrammnormalisierung sind die Lücken zwischen den Balken der Grauwerte nicht mehr gleichverteilt, sondern in Bereichen mit hoher Anzahl von Grauwerten im Eingabebild sind die Lücken größer als in Bereichen mit wenig Grauwerten. Das bewirkt eine Kontrastverstärkung bei den Maxima und eine Kontrastabschwächung bei den Minima, z.B. ist dadurch der Schatten der Bäume auf der Hausmauer in Abbildung 29 besser erkennbar.

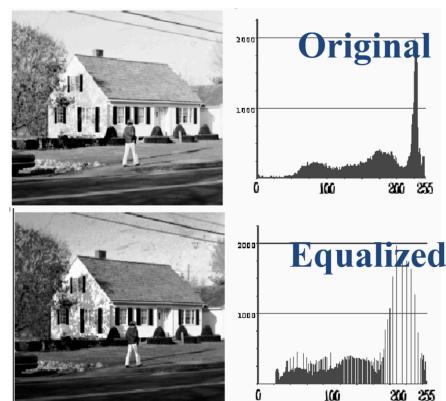


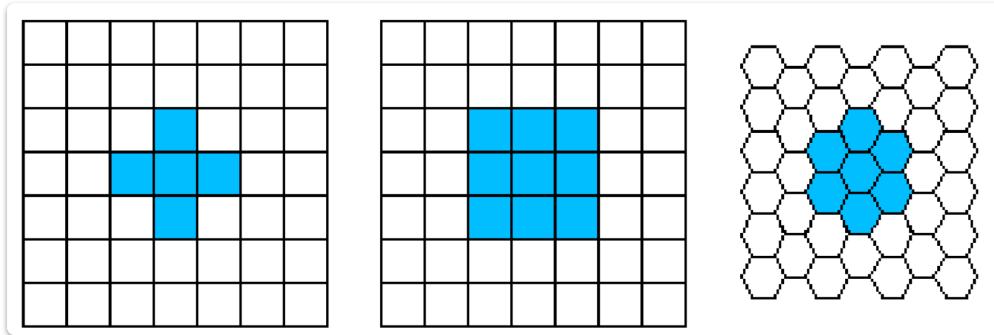
Abbildung 29: Histogrammausgleichung

5. Lokale Operationen

Quellen:

- [EVC_Skriptum_CV, p.24](#) bis [EVC_Skriptum_CV, p.28](#)

Nachbarschaften:



- Eine Nachbarschaft bezeichnet eine kleine, definierte Bildregion um ein Pixel, um Bildverarbeitungsoperationen durchzuführen.

Vierer-Nachbarschaft

- Jedes Pixel P hat **2 horizontale** und **2 vertikale** Nachbarn.
- Koordinaten des Pixels P : (u, v) .
- Koordinaten der vier D-Nachbarn:
 $(u - 1, v), (u + 1, v), (u, v - 1), (u, v + 1)$
- Eine Vierer-Nachbarschaft besteht aus **5 Punkten** (Pixel P + 4 D-Nachbarn).

Achter-Nachbarschaft

- Neben den D-Nachbarn hat jedes Pixel P auch **4 diagonale Nachbarn**.
- Koordinaten der diagonalen Nachbarn:
 $(u - 1, v - 1), (u + 1, v + 1), (u - 1, v + 1), (u + 1, v - 1)$
- Eine Achter-Nachbarschaft besteht aus **9 Punkten** (Pixel P + 4 D-Nachbarn + 4 diagonale Nachbarn).

Abstand der Nachbarn

- Der Abstand der Nachbarn wird durch die **Metrik** festgelegt:
 - **Euklidische Metrik**: Abstand beträgt $\sqrt{2}$.
 - **Manhattan-Metrik**: Abstand beträgt 2.

Was sind lokale Operationen

Punktoperationen

- Der neue Wert eines Bildelements hängt ausschließlich vom ursprünglichen Bildwert an derselben Position ab.
- siehe [4. Punktoperationen](#)

Lokale Operationen (Filter)

- **Ähnlichkeit zu Punktoperationen:** Auch hier besteht eine **1:1-Abbildung** der Bildkoordinaten, d. h., die Geometrie des Bildes bleibt unverändert.
- **Unterschied zu Punktoperationen:** Das Ergebnis wird nicht nur aus einem einzigen Ursprungspixel berechnet, sondern aus mehreren Pixeln des Originalbildes.
- Die Koordinaten der Quellpixel sind bezüglich der aktuellen Position (u, v) definiert und bilden eine zusammenhängende Region.

Filterregion

- Die **Größe der Filterregion** bestimmt, wie viele ursprüngliche Pixel zur Berechnung des neuen Pixelwerts beitragen und damit das **räumliche Ausmaß des Filters**.
- Eine gängige Filtergröße ist 3×3 , zentriert in der Achter-Nachbarschaft um die Koordinate (u, v) .
- Die Form der Filterregion muss nicht quadratisch sein, sondern kann beliebige Formen annehmen.

Lineare Filter

- **Bezeichnung:** Lineare Filter verbinden die Pixelwerte innerhalb der Filterregion in **linearer Form**, d. h., durch eine gewichtete Summation.
- **Beispiel:** Die **lokale Mittelwertbildung** ist ein einfaches Beispiel, bei dem alle neun Pixel der 3×3 Filterregion mit der Gewichtung $1/9$ summiert werden.

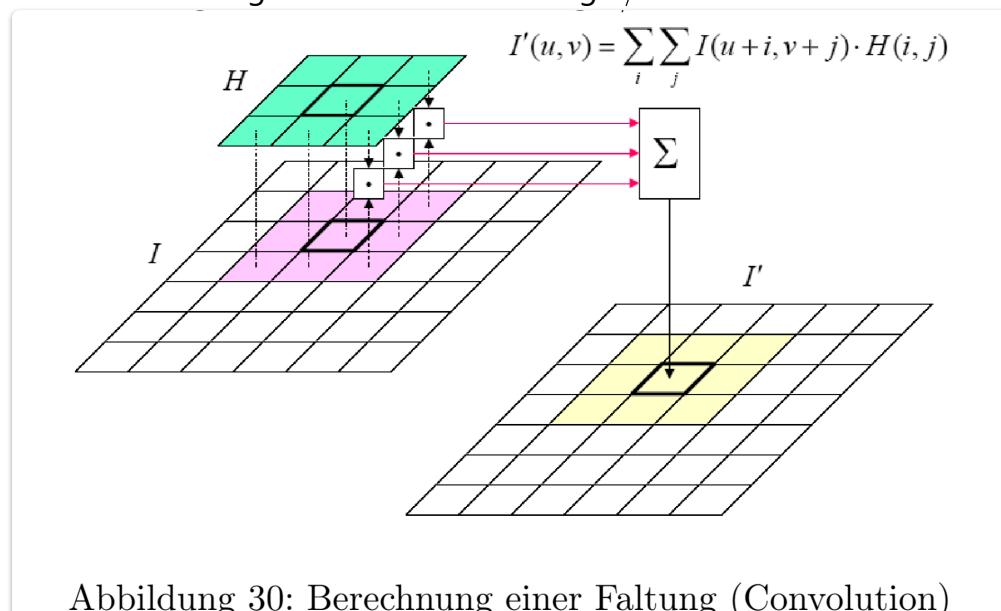


Abbildung 30: Berechnung einer Faltung (Convolution)

Filtermatrix

- **Definition:** Eine **Filtermatrix** oder **Filtermaske** $F(i, j)$ spezifiziert die Größe, Form und die zugehörigen **Gewichte** der Filterregion.
 - Die Größe der Matrix entspricht der Größe der Filterregion.
 - Jedes Element $F(i, j)$ der Matrix definiert das Gewicht des entsprechenden Pixels.
- **Einzigartigkeit:** Das Ergebnis eines linearen Filters ist eindeutig und vollständig durch die Koeffizienten in der Filtermatrix bestimmt.

Anwendung des Filters

- Die Anwendung eines linearen Filters auf ein Bild erfolgt durch folgende Schritte:
 1. **Positionierung der Filterfunktion F :** Die Filtermatrix F wird so über das Bild I positioniert, dass ihr Koordinatenursprung $F(0, 0)$ auf das aktuelle Bildelement $I(u, v)$ fällt.
 2. **Multiplikation und Summation:** Alle Bildelemente in der Filterregion werden mit den jeweils darüber liegenden Filterkoeffizienten multipliziert und die Ergebnisse werden summiert.
 3. **Speichern des Ergebnisses:** Die resultierende Summe wird an der entsprechenden Position im Ergebnisbild $I'(u, v)$ gespeichert.

Berechnung für den 3×3 Filter

- Für einen 3×3 Filter des neuen Bildes $I'(u, v)$ wird der Wert für jedes Pixel wie folgt berechnet:
 - Die Schritte 1–3 werden an jeder Position (u, v) im Bild wiederholt, um das gefilterte Bild zu erhalten.

Tiefpassfilter

Unterscheidung zwischen Tiefpass- und Hochpassfiltern

- **Tiefpassfilter:**
 - Filtern **hohe Frequenzen** heraus und lassen **niedrige Frequenzen** passieren.
 - Eignen sich für **Rauschunterdrückung** bzw. als **Glättungsoperatoren**.
 - Bekannte Tiefpassfilter: **Mittelwertfilter** und **Gauß-Filter**.
- **Hochpassfilter:**
 - Filtern **tiefe Frequenzen** heraus und lassen **hohe Frequenzen** passieren.
 - Eignen sich z. B. für die **Kantendetektion**. (siehe [8. Bildmerkmale - Interest Points](#))

Mittelwertfilter (Box-Filter)

- **Filtermaske:** Besteht aus lauter gleichen Gewichten (1), einfachste Form aller Tiefpassfilter.
- **Nachteile:**
 - Scharf abfallende Ränder und unoptimales Frequenzverhalten.

- Alle Bildelemente haben das gleiche Gewicht, wodurch das Zentrum nicht stärker gewichtet wird als die Ränder.

Gauß-Filter

- Filtermaske:** Entspricht einer diskreten, zweidimensionalen **Gauß-Funktion**.
 - Beispiel für eine Filtermaske (für $\sigma = 0.5$):
$$F_{Gauss} = \frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$
- Eigenschaften:**
 - Das mittlere Bildelement erhält das **maximale Gewicht**.
 - Die Werte der übrigen Koeffizienten nehmen mit zunehmender Entfernung zur Mitte **kontinuierlich und gleichmäßig** ab (isotrop).
 - Standardabweichung σ** bestimmt den „Radius“ der glockenförmigen Funktion und beeinflusst die Stärke der Glättung.

Filtermaske für einen 3×3 Gauß-Filter mit $\sigma = 0.5$

- Die resultierende Filtermaske lautet:
$$F_{Gauss} = \begin{bmatrix} 0.011 & 0.084 & 0.011 \\ 0.084 & 0.619 & 0.084 \\ 0.011 & 0.084 & 0.011 \end{bmatrix}$$
- Summe der Koeffizienten muss 1 betragen, was durch Division aller Koeffizienten durch deren Summe erreicht wird.

Wichtige Hinweise

- Größere Filtermasken** führen zu einer besseren **Approximation der Gauß-Funktion**, aber ändern nicht das **Glättungsverhalten**.
- Die **Stärke der Glättung** kann durch die **Standardabweichung σ** variiert werden.
- Ein **Mittelwertfilter** mit einer 3×3 Filtermaske führt zu einem **befriedigenden Ergebnis**, aber der **Gauß-Filter** wird im Allgemeinen bevorzugt.

Differenzfilter

Interpretation negativer Filterkoeffizienten

- Wenn **einzelne Filterkoeffizienten negativ** sind, kann die Filteroperation als **Differenz zweier gewichteter Summen** verstanden werden:
Summe positiver Gewichtungen – Summe negativer Gewichtungen
- Innerhalb der Filterregion R werden:
 - Positive Koeffizienten → positiv gewichtete Pixel.
 - Negative Koeffizienten → negativ gewichtete Pixel.

Beispiel: Laplace-Filter

- Filtermatrix:

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

- Berechnet die **Differenz zwischen dem zentralen Pixel** (-4) und den **4 umliegenden Pixeln** (Vierer-Nachbarschaft).
- **Übrige 4 Pixel** (diagonale Nachbarn) haben Koeffizienten 0 → **werden nicht berücksichtigt**.

Wirkung der Differenzbildung

- **Gegenteil** zur Durchschnittsbildung:
 - Durchschnitt → **Glättung** von Intensitätsunterschieden.
 - Differenz → **Verstärkung** von Intensitätsunterschieden.
- **Anwendungen:**
 - **Kanten- und Konturverstärkung**
 - **Bildschärfung**
- → **Differenzfilter sind Hochpassfilter**.

Mathematische Grundlage

- Hochpassfilter basieren auf Ableitungen der Bildfunktion $g(x, y)$:
 - **Erste Ableitung** → **Gradientenfilter**
 - **Zweite Ableitung** → **Laplace-Filter**

Nachbearbeitung des Ergebnisbildes

- Ergebnis enthält oft **positive und negative Grauwerte**.
- Mögliche Nachbearbeitungen:
 - **Normierung** auf z. B. $[0, 255]$
 - **Betragssbildung**: $|g(x, y)|$

Anwendung in Software (z. B. Adobe Photoshop)

- Umsetzung durch sogenannte „**Custom Filter**“
- Filter mit:
 - **Ganzzahligen Koeffizienten**
 - **Skalierungsfaktor (Scale)**
 - **Offset-Wert**, um negative Ergebnisse in den **sichtbaren Intensitätsbereich** zu verschieben.

Bildrandproblem

- Beim Anwenden von Filtern kann es zu **Problemen an den Bildrändern** kommen.
- Ursache: Die **Filterregion überschreitet den Bildbereich**, es fehlen passende Pixelwerte → das Filterergebnis **kann nicht berechnet werden**.
- Es gibt keine mathematisch exakte Lösung für das Problem
- andere Probleme mit Bildrand siehe: [7. Clipping und Antialiasing](#)

Methoden zum Umgang mit dem Randproblem

1. Einsetzen eines konstanten Werts

- Beispiel: 0 (schwarz)
- **Nachteil:** Verkleinert den sichtbaren Bildbereich.
- **Nicht akzeptabel** in den meisten Anwendungen.

2. Beibehalten der ursprünglichen (ungefilterten) Bildwerte

- Filter wird **nicht auf die Randpixel angewendet**.
- **Nachteil:** Inkonsistente Bildverarbeitung; ebenfalls nicht ideal.

3. Annahme künstlicher Pixelwerte außerhalb des Bildbereichs:

- (a) **Konstanter Wert** außerhalb des Bildes (z. B. schwarz oder grau)
 - Kann bei großen Filtern zu **starken Verfälschungen an den Rändern** führen.
- (b) **Fortsetzung der Randpixel**
 - Randwerte des Bildes werden **nach außen hin fortgeführt**.
 - **Geringe Verfälschung** → bevorzugte Methode
- (c) **Zyklische Wiederholung des Bildes**
 - Das Bild wird **horizontal und vertikal periodisch fortgesetzt**.

Formale Eigenschaften lineare Filter

Ursprung und Definition

- **Lineare Filter** basieren auf dem mathematischen Konzept der **linearen Faltung** (engl. *linear convolution*).
- Sie verknüpft zwei **Funktionen gleicher Dimensionalität**, kontinuierlich oder diskret.
- Für diskrete, 2D-Funktionen I (Bild) und F (Filtermatrix) ist die Faltung definiert als: $I' = I * FI'$
- Dabei gilt (mit Berücksichtigung der Koordinatenumkehr):

$$I'(u, v) = \sum_i \sum_j I(u - i, v - j) \cdot F(-i, -j)$$

- Die ursprüngliche lineare Filterdefinition entspricht einer **linearen Korrelation**, da hier **keine Spiegelung** der Filtermatrix erfolgt.

Eigenschaften der linearen Faltung

1. Kommutativität:

$$I * F = F * I$$

→ Reihenfolge von Bild und Filter spielt **keine Rolle**.

2. Linearität:

- Skalierung eines Bildes:

$$(a \cdot I) * F = a \cdot (I * F)$$

- Addition zweier Bilder:

$$(I_1 + I_2) * F = (I_1 * F) + (I_2 * F) \quad (I_1 + I_2) * F = (I_1 * F) + (I_2 * F)$$

3. Assoziativität:

$$A * (B * C) = (A * B) * C$$

→ Filter können **beliebig kombiniert** und **umgruppiert** werden.

Konsequenz: Separierbarkeit

- Ein Filterkern F kann als **Faltungsprodukt kleinerer Filterkerne** beschrieben werden:

$$F = F_1 * F_2 * \dots * F_n$$

- Besonders nützlich: Trennung in **zwei eindimensionale Filter**:

Beispiel:

- $F_x = [1 \ 2 \ 1]$
- $F_y = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix}$

Kombiniert:

$$F_{xy} = F_x * F_y$$

- Vorteil: **Reduktion der Rechenkomplexität**
 - Normal: $3 \times 5 = 15$ Operationen pro Pixel
 - Separiert: $5 + 3 = 8$ Operationen pro Pixel

Nicht lineare Filter

Nachteil linearer Filter

- Lineare Filter **glätten** nicht nur **Störungen**, sondern auch **gewollte Bildstrukturen** wie:
 - Punkte
 - Kanten
 - Linien
- → **Bildqualität leidet**: Strukturen werden verwischt.
- → Einschränkung ihrer Anwendung bei Struktur- oder Kantenerhaltung.

Rangordnungsfilter (engl. *rank value filters*)

- **Nichtlineare Filteroperationen**

- Kombination von benachbarten Pixeln durch **Vergleichen und Selektieren**, statt **Gewichten und Addieren**.
- Funktionsweise:
 - Alle Grauwerte innerhalb der Filtermaske werden **sortiert** (aufsteigend).
 - Es wird **ein bestimmter Rang** aus dieser Liste ausgewählt.
 - Dieser Wert ersetzt das zentrale Pixel.

Typen von Rangordnungsfilters

- **Medianfilter**:
 - Wählt den **mittleren Wert** (Median) der sortierten Grauwerte.
 - Besonders wirksam bei der **Entfernung von Ausreißern** (z.B. Salz-und-Pfeffer-Rauschen).
- **Minimumfilter**:
 - Wählt den **kleinsten** Grauwert in der Region.
- **Maximumfilter**:
 - Wählt den **größten** Grauwert in der Region.

Vorteile

- Besser geeignet zur **Erhaltung von Kanten und feinen Strukturen**.
- Besonders effektiv bei **nicht-gausschem Rauschen**.

Definition dieser Filter:

$I(u, v) = \min\{I(u + i, v + j) \text{ für } (i, j) \in R\} = \min(R_{u,v})$ bzw. $I(u, v) = \max\{I(u + i, v + j) \text{ für } (i, j) \in R\} = \max(R_{u,v})$, wobei $R_{u,v}$ die Region der Bildwerte bezeichnet, die an der aktuellen Position (u, v) von der Filterregion überdeckt werden. Die Abbildung 31 zeigt die Anwendung von 3x3-Min- und -Max-Filtren auf ein Grauwertbild, das künstlich mit SSalt-and-PepperSStörungen versehen wurde (zufällig platzierte weiße und schwarze Punkte). Der

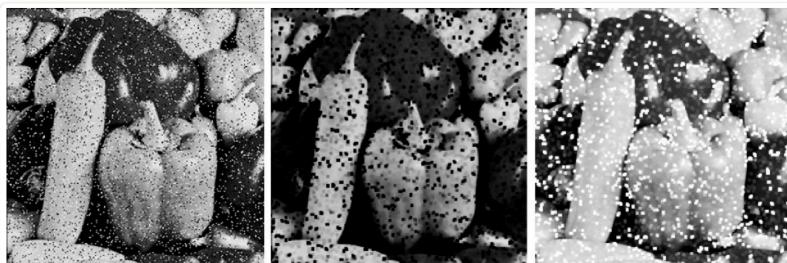
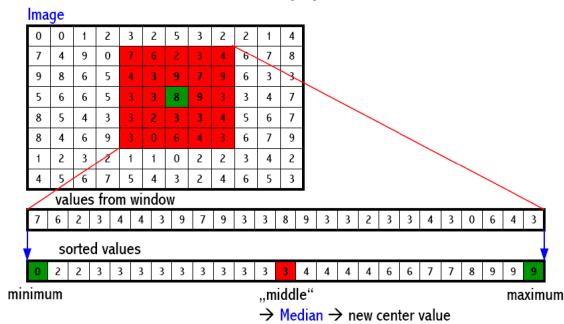


Abbildung 31: Min-/Max-Filter Anwendung

Min-Filter entfernt die weißen (Salt) Punkte, denn ein einzelnes, weißes Pixel wird innerhalb der 3x3-Filterregion R immer von kleineren Werten umgeben, von denen einer den Minimalwert liefert. Gleichzeitig werden durch das Min-Filter aber dunkle Strukturen räumlich erweitert. Der Max-Filter hat genau den gegenteiligen Effekt.



Grundproblem bei der Filterung

- Kein Filter kann *automatisch* unterscheiden zwischen:
 - wichtigen Strukturen (z.B. Kanten, Details)
 - unerwünschten Störungen (z.B. Rauschen)
- → Perfekter Filter existiert nicht.
- Jeder Filter trifft eine "blinde Entscheidung", ob ein Pixel zur Struktur oder zur Störung gehört.

Medianfilter – ein sinnvoller Kompromiss

- Ziel: **Störungen entfernen**, aber **Strukturen besser erhalten** als bei linearen Filtern.
- **Definition:**
Jedes Bildelement $I(u, v)$ wird durch den **Median** der Pixelwerte innerhalb einer Filterregion R ersetzt: $I(u, v) = \text{median}(R_{u, v})$
- Der **Median** aus einer sortierten Liste von $2K + 1$ Pixelwerten p_i ist der mittlere Wert:
 $\text{median}(p_0, \dots, p_{2K}) = p_K$ (sofern $p_i \leq p_{i+1}$)

Beispielhafte Wirkung

- **Abbildung 32** (gedanklich):
 - **Linkes Bild:** Originalbild mit **Salt-and-Pepper-Rauschen**.
 - **Mittleres Bild:** Nach Anwendung eines **Mittelwertfilters** – Störungen teilweise noch sichtbar.
 - **Rechtes Bild:** Nach Anwendung eines **Medianfilters** – Störungen **besser entfernt**, Strukturen **erkennbar erhalten**.



Abbildung 32: Anwendung von Mittelwert- und Medianfilter

Vorteile des Medianfilters

- Robust gegenüber Ausreißern
 - Besonders effektiv bei impulsartigem Rauschen wie Salt-and-Pepper-Noise
- Erhält Kanten besser als der Mittelwertfilter
- Nichtlinear, daher nicht anfällig für lineare Glättungsverluste

6. Kantenfilterung

Information:

- [EVC_Skriptum_CV, p.29](#) bis [EVC_Skriptum_CV, p.34](#)
- ist quasi ein Unterkapitel von [5. Lokale Operationen](#)

Kantenfilterung

- wichtiger Schritt in Bildverarbeitung
- um Kanten hervorzuheben
- Ansatz dafür ist Verwendung von Gradienten die man sich so berechnen kann:

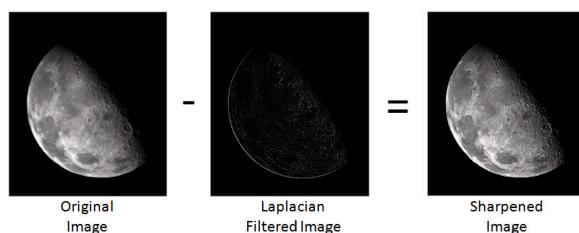
$$\nabla I = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\partial I}{\partial x} \\ \frac{\partial I}{\partial y} \end{bmatrix}$$

- G_x und G_y sind die Gradientenkomponenten in horizontaler bzw. vertikaler Richtung
- Stärke davon hängt durch Betrag von $|\nabla I|$ und dem Winkel ϕ ab

$$|\nabla I| = \sqrt{G_x^2 + G_y^2} \quad \text{und} \quad \theta = \text{atan2}(G_y, G_x)$$

- Es gibt verschiedene Filter die zur Kantenfilterung verwendet werden z.B.:
 - Sobel-Operator
 - Prewitt-Operator
 - ...

Bildschärfung



Ziel der Bildschärfung

- **Nachträgliches Schärfen von Bildern** dient dazu, **Unschärfen zu kompensieren**, die durch:
 - Scannen
 - Skalieren
 - Druck

- Bildschirmanzeige entstanden sind oder entstehen können.

Methode

- Hochfrequente Bildanteile enthalten die Bilddetails, die den Schärfeeindruck ausmachen.
- Beim Schärfen werden diese hochfrequenten Anteile angehoben.

Laplace-Operator

- Im zweidimensionalen Fall erfolgt das Schärfen mit dem Laplace-Operator.
- Der Laplace-Operator ∇^2 einer Funktion $f(x, y)$ ist definiert als:

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

wobei:

$$\frac{\partial f}{\partial^2 x}(x, y) = f(x + 1, y) + f(x - 1, y) - 2f(x, y)$$

in x Richtung und analog in y Richtung.

- Er nutzt die zweiten Ableitungen (∂^2 steht für 2. Partielle Ableitung) in horizontaler und vertikaler Richtung zur Hervorhebung von Kanten und feinen Strukturen.
- Daraus entsteht in der Kombination von x- und y-Richtung mit $[1 - 2 1]$ und $[1 - 2 1]^T$ der Laplace Operator
- Fürs Filtern von einem Bild, zuerst Laplace Filter und dann das Ergebnis vom Ursprungsbild subtrahieren:

$$I' = I - w * (H^L * I)$$

- H^L ist Laplace Filter
- w bestimmt Intensität der Schärfung

Kanten

Bedeutung von Kanten

- Umgangssprachlich: Kanten = abrupte Änderungen der Oberflächennormale (z.B. Tischkante).
- In Bildern: Kanten = Stellen mit lokalen Änderungen der Intensität oder Farbe (Diskontinuitäten).
- Visuelle Wahrnehmung:
 - Subjektive Schärfe hängt mit Deutlichkeit von Diskontinuitäten zusammen.

- Kanten sind **strukturell wichtig** für die Interpretation von Bildern.
- Unser Sehsinn erkennt Inhalte oft schon an **kantenförmigen Strukturen** (z.B. in Karikaturen).

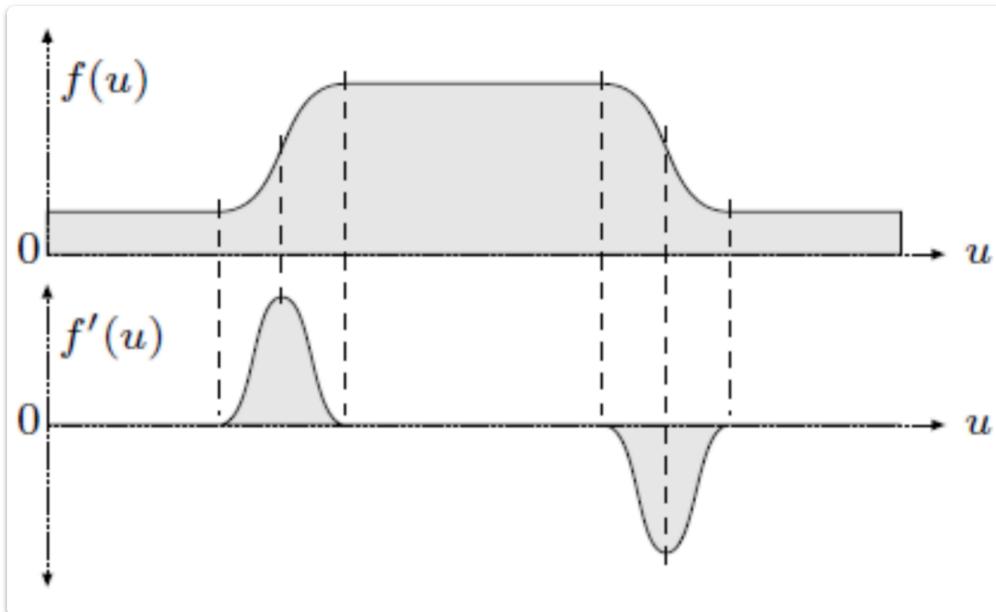
Bedeutung in der Bildverarbeitung

- Kanten enthalten **nicht-redundante Information**, d.h. wichtige Strukturinformationen.
- Definition: Kanten = Orte im Bild mit **starker Intensitätsänderung auf kleinem Raum** und entlang **ausgeprägter Richtung**.

Mathematische Beschreibung

- **Stärke der Änderung pro Distanz** = **erste Ableitung** der Intensitätsfunktion.
- Je größer die Ableitung, desto **deutlicher die Kante**.

Gradientenbasierte Kantendetektion



Motivation

- Betrachten wir ein eindimensionales **Intensitätsprofil** einer Bildzeile.
- Beispiel: Helle Region im Zentrum, dunkler Hintergrund → typische Kante im Bild.

Erste Ableitung – Bedeutung

- Bezeichnet als:

$$f'(u) = \frac{df}{du}(u)$$
- **Positive Werte** → Intensität steigt (z.B. dunkler zu heller Übergang).
- **Negative Werte** → Intensität fällt (z.B. heller zu dunkler Übergang).

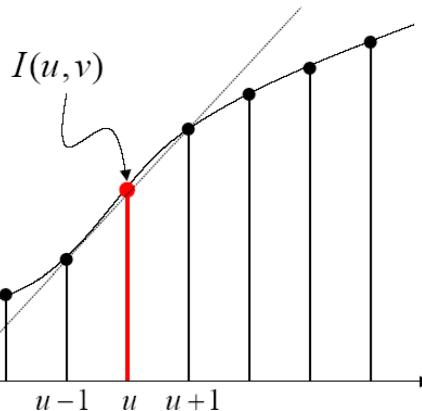
Problem bei diskreten Bildern

- In digitalen Bildern ist die Funktion **diskret** → Ableitung **nicht direkt definiert**.
- Wir benötigen eine **Schätzung** (Approximation).

Lösung: Differenzenapproximation

- Ableitung an Position u kann geschätzt werden durch:

$$\frac{df}{du}(u) \approx \frac{f(u+1) - f(u-1)}{2}$$



$$\frac{\partial I(u, v)}{\partial u} \approx I(u+1, v) - I(u-1, v)$$

Bei diskreten Signalen
nur Approximation möglich!

- Dies entspricht dem **Anstieg einer Geraden**, die durch die **benachbarten Abtastwerte** verläuft.

Interpretation

- Diese Approximation entspricht einem **Kantenfilter**.
- Wird oft in **Kantendetektion** (z.B. Sobel-Operator) verwendet.
- Erlaubt es, **Kanten als starke Intensitätsänderungen** zu erkennen – genau das, was unser Auge als „Kante“ wahrnimmt.

Gradienten und Kanten in zweidimensionalen Bildern

- Ein Bild wird als zweidimensionale Funktion $I(u, v)$ betrachtet
- Kanten im Bild = abrupte lokale Änderungen in Intensität oder Farbe
- Starke Änderungen = hohe Ableitungswerte → Hinweis auf Kante
- Erste Ableitung misst Stärke der Intensitätsänderung
- Für diskrete Funktionen ist die Ableitung nicht definiert → Approximation notwendig
- In 1D wird die erste Ableitung durch den Differenzenquotienten geschätzt:

$$\frac{df}{du}(u) \approx \frac{f(u+1) - f(u-1)}{2}$$

- In 2D: partielle Ableitungen entlang der Koordinatenachsen:

$$\frac{\partial I}{\partial u}(u, v), \quad \frac{\partial I}{\partial v}(u, v)$$

- Gradient Vektor:

$$\nabla I = \begin{pmatrix} \frac{\partial I}{\partial u} \\ \frac{\partial I}{\partial v} \end{pmatrix}$$

- Betrag des Gradienten (Kantenstärke):

$$\|\nabla I\| = \sqrt{\left(\frac{\partial I}{\partial u}\right)^2 + \left(\frac{\partial I}{\partial v}\right)^2}$$

- Betrag des Gradienten ist rotationsinvariant
- Horizontale Ableitung kann geschätzt werden durch linearen Filter:

$$H_{Dx} = \frac{1}{2} \cdot [-1 \quad 0 \quad 1]$$

- Vertikale Ableitung entsprechend:

$$H_{Dy} = \frac{1}{2} \cdot \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix}$$

- Filterantwort ist richtungsabhängig
- Horizontale Filter erkennen vertikale Kanten, vertikale Filter horizontale Kanten
- In flachen Bildregionen (konstante Intensität) ist die Filterantwort null

Kantendetektionsfilter

Allgemeine Eigenschaften von Ableitungsfiltern

- Ableitungsfilter beliebiger Ordnung dürfen keine Antwort auf konstante Intensitätswerte oder Signal-Offsets zeigen
- **Bedingung:** Summe der Filterkoeffizienten muss 0 sein
- Für gute Kantendetektion: Operatorantwort soll richtungsunabhängig sein → **isotrope Kantendetektion**

Arten von Kantenfiltern

- **Isotrope Filter:** Reagieren unabhängig von der Richtung, z. B. **Laplace-Operator**
- **Richtungsabhängige Filter:** Reagieren auf u- oder v-Richtung, z. B. **Sobel-** und **Prewitt-Operator**

Unterschiede der Kantenfilter

- Differenzieren sich durch:
 - die Filter zur Schätzung der Richtungsanteile
 - die Art der Kombination der Richtungsanteile zur Gesamtkanteninformation

Gradient und Kanteninformation

- Gradient enthält:
 - **Betrag** → Kantenstärke
 - **Richtung** → Kantenverlauf

Prewitt- und Sobel-Operator

- Beide nutzen Filterkerne mit 3 Zeilen bzw. 3 Spalten → reduzierte Rauschanfälligkeit
- **Prewitt-Operator:**
 - Filter: HP_x, HP_y
 - Einfache Box-Glättung vor der Ableitung
- **Sobel-Operator:**
 - Fast identische Filter, aber stärkere Gewichtung der mittleren Zeile/Spalte durch doppelte Glättung

Mathematische Formulierungen

- Gradienten in x- und y-Richtung:

$$D_x(u, v) = H_x * I$$

$$D_y(u, v) = H_y * I$$

- Kantenstärke:

$$E(u, v) = \sqrt{D_x(u, v)^2 + D_y(u, v)^2}$$

- Kantenrichtung:

$$\Phi(u, v) = \tan^{-1} \left(\frac{D_y(u, v)}{D_x(u, v)} \right)$$

Ablauf der Kantendetektion

1. Bild I wird mit Filtern H_x und H_y gefaltet
2. Aus D_x und D_y werden:
 - Kantenstärke $E(u, v)$
 - Kantenrichtung $\Phi(u, v)$ berechnet

Weitere Kantenoperatoren

- **Roberts-Operator** (ältester Kantenoperator)
 - Sehr kleine Filtergröße: 2×2
 - Schätzt Ableitungen entlang der Diagonalen
 - $H_{R1} = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}, H_{R2} = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$

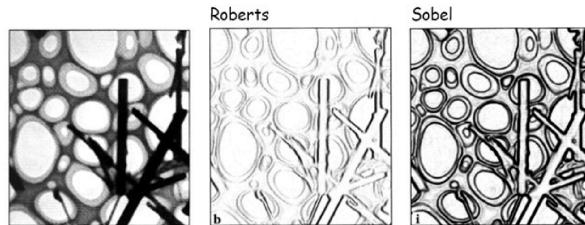
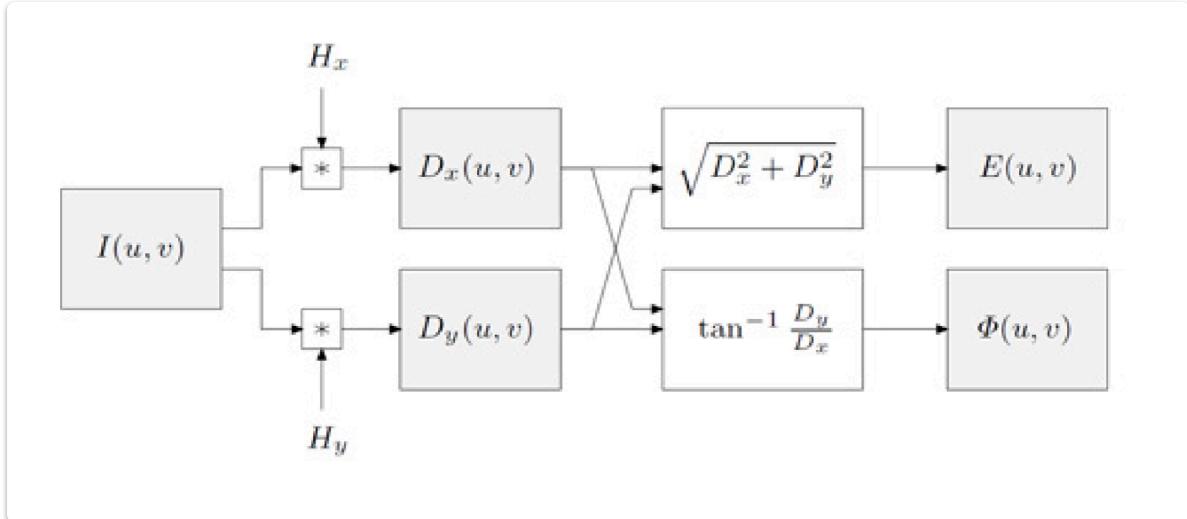


Abbildung 34: Roberts- und Sobelfilter

Roberts-Filter

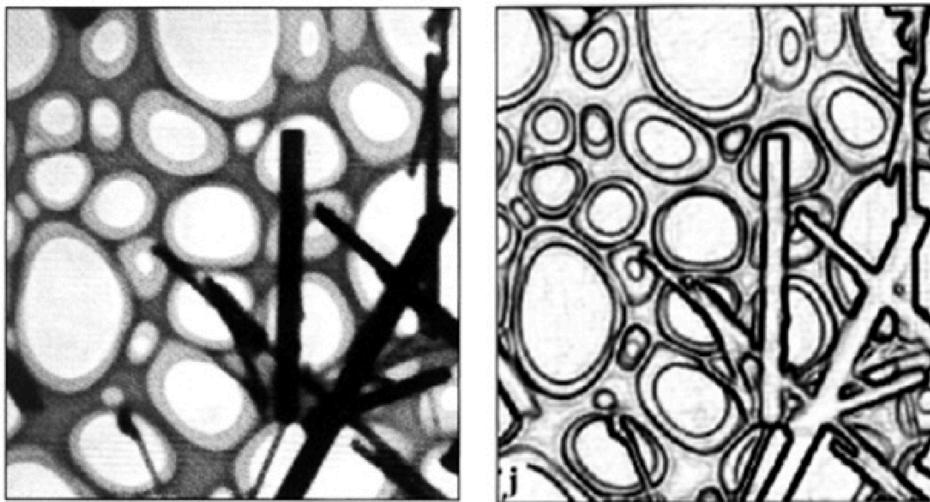
- **Starke Reaktion auf diagonale Kanten**
- Filter sind **weniger richtungsselektiv**: Reagieren über breites Orientierungsband ähnlich stark
- **Mängel bei horizontalen und vertikalen Kanten**:
 - Schlechte Filterantwort bei diesen Kanten, wodurch die Kantendetektion ungenau wird

Kompromiss im Design von Kantenfiltern

- **Besseres Filterdesign**:
 - Je besser ein Filter auf kantenartige Bildstrukturen reagiert, desto **richtungsabhängiger** wird es
 - **Breitere Filter** reagieren auf viele Orientierungen, aber mit geringerer Präzision in der Richtung
 - **Engere Filter** reagieren stärker in spezifischen Richtungen, bieten jedoch bessere Kantenerkennung

Kirsch-Operator

- **Kirsch-Operator:** Beispiel für den Einsatz mehrerer Filter in verschiedenen Richtungen



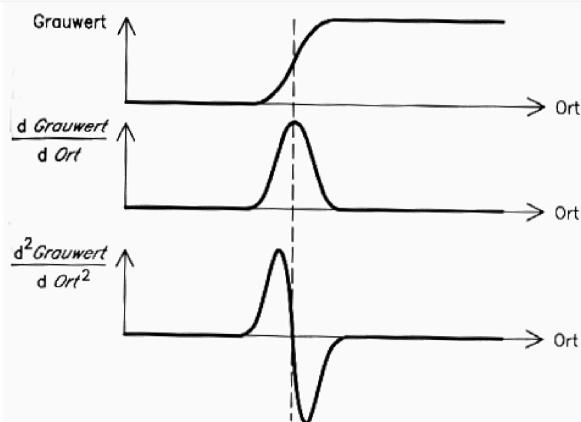
- Filter für **acht verschiedene Richtungen** im Abstand von **45°**
- Bietet höhere Präzision durch mehrere **enger aufgestellte Filter** für spezifische Richtungen
- Diese Acht Richtungen sehen so aus:

$$H_1^K = \begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix}, H_2^K = \begin{pmatrix} -2 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & 2 \end{pmatrix}, H_3^K = \begin{pmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{pmatrix}, H_4^K = \begin{pmatrix} 0 & -1 & -2 \\ 1 & 0 & -1 \\ 2 & 1 & 0 \end{pmatrix},$$

$$H_5^K = \begin{pmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{pmatrix}, H_6^K = \begin{pmatrix} 2 & 1 & 0 \\ 1 & 0 & -1 \\ 0 & -1 & -2 \end{pmatrix}, H_7^K = \begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix}, H_8^K = \begin{pmatrix} 0 & 1 & 2 \\ -1 & 0 & 1 \\ -2 & -1 & 0 \end{pmatrix}$$

Die **Kantenstärke** E^K an der Stelle (u,v) ist als **Maximum** der einzelnen Filterergebnisse definiert, d.h. $E^K(u,v) = \max(D_1(u,v), D_2(u,v), \dots, D_8(u,v))$, und der am stärksten ansprechende Filter bestimmt auch die zugehörige **Kantenrichtung**. Derartige **Kompass-Operatoren** bieten allerdings kaum Vorteile gegenüber einfacheren Operatoren, wie z.B. dem Sobel-Operator. Ein Vorteil des Kirsch-Operators ist, dass er keine Wurzelberechnung benötigt.

Kantendetektion mit 2. Ableitungen



Problem der ersten Ableitung bei Kantendetektion

- **Breite der Kanten:** Bei der ersten Ableitung wird die Kante genauso breit wie der Anstieg der Bildfunktion, was eine genaue Positionsbestimmung erschwert.

Nutzung der zweiten Ableitung

- Zweite Ableitung misst die **lokale Krümmung** der Funktion
- **Nulldurchgänge der zweiten Ableitung:** Diese stellen die tatsächlichen Kantenpositionen dar
 - **Vorteil:** Genauere Bestimmung der Kantenposition als bei der ersten Ableitung

Laplace-Operator

- Bekannter und einfachster Kantenfilter basierend auf der zweiten Ableitung
- **Richtungsunabhängig** (Vorteil)
- **Rauschanfällig** (Nachteil)
- **Lösung:** Vor Anwendung des Laplace-Operators sollte ein Glättungsfilter zur Rauschunterdrückung eingesetzt werden

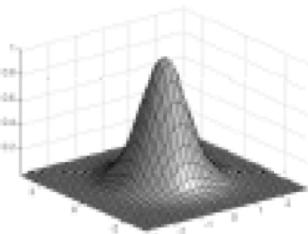
Laplacian-of-Gaussian-Operator (LoG)

- **LoG-Operator (Laplacian of Gaussian)** oder **Marr-Hildreth-Operator:**
 - Kombiniert **Glättung mit einem Gauß-Filter** und die **zweite Ableitung (Laplace-Filter)** in einem einzigen linearen Filter
 - Aufgrund seiner Form (ähnelt einem mexikanischen Sombrero) auch als **Mexican Hat** oder **Sombrerofilter** bekannt
 - **Diskrete Umsetzung:**
 - Zuerst wird das Bild mit einer **Binomialmaske** geglättet
 - Anschließend erfolgt die Anwendung des **diskreten Laplace-Operators**

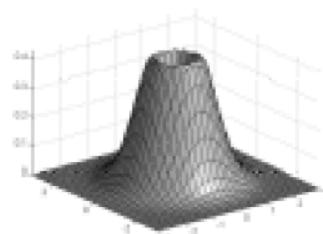
2D-Gauß-Funktion

- Ausgangspunkt für die Erzeugung des **LoG-Filterkernels** ist die **2D-Gauß-Funktion**

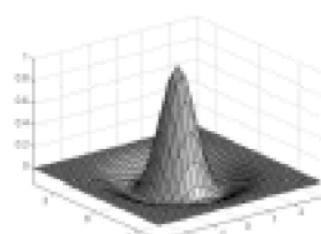
$$f(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$



Gauss' Bell
Function



First derivative



Minus second derivative
„Mexican Hat“

Wendet man den Laplace-Operator auf die Gauß-Funktion an, erhält man die kontinuierliche Repräsentation des LoG:

$$g(x, y) = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} = -\frac{1}{\pi\sigma^4} e^{-\frac{x^2+y^2}{2\sigma^2}} \left(1 - \frac{x^2+y^2}{2\sigma^2}\right)$$

Die diskrete Approximation sollte für Kernel ungerader Kantenlänge durchgeführt werden, wobei der Ursprung des Kernels jeweils in der Mitte liegt. Für die Kantenlänge 5 entspricht dies dem unten abgebildeten Filter H^{LoG} . Die Abbildung zeigt ein Anwendungsbeispiel, die detektierten Kanten sind jetzt Ein Pixel breit.

$$H^{\text{LoG}} = \begin{pmatrix} 0 & 0 & -1 & 0 & 0 \\ 0 & -1 & -2 & -1 & 0 \\ -1 & -2 & 16 & -2 & -1 \\ 0 & -1 & -2 & -1 & 0 \\ 0 & 0 & -1 & 0 & 0 \end{pmatrix}$$

Canny Operator



Kleinere Kantendetektoren (z.B. Sobel-Operator)

- **Begrenzte Reaktion:** Erkennen nur Intensitätsunterschiede innerhalb ihrer kleinen Filterregionen (z. B. 3×3)
- **Lösungsansatz für größere Unterschiede:**
 - Verwendung größerer Filter (größere Kantenoperatoren)
 - Anwendung der Operatoren auf **skalierte Bilder** (Mehrskalentechniken)

Multi-Resolution bzw. Multi-Scale-Techniken

- **Grundidee:**
 - Kanten werden auf verschiedenen Auflösungsebenen (Skalen) erkannt
 - Für jede Bildposition wird entschieden, welche Kante auf welcher Ebene dominant ist

Canny-Operator

- **Ziel:**
 - Verwendet einen Satz großer, **gerichteter Filter** auf mehreren Auflösungsebenen
 - Ergebnisse werden zu einem **gemeinsamen Kantenbild** (Edge Map) zusammengeführt
- **Ziele des Canny-Operators:**
 - Minimierung der **falschen Kantenmarkierungen**
 - **Optimale Lokalisierung** von Kanten
 - Sicherstellung, dass pro Kante nur **eine Markierung** erfolgt

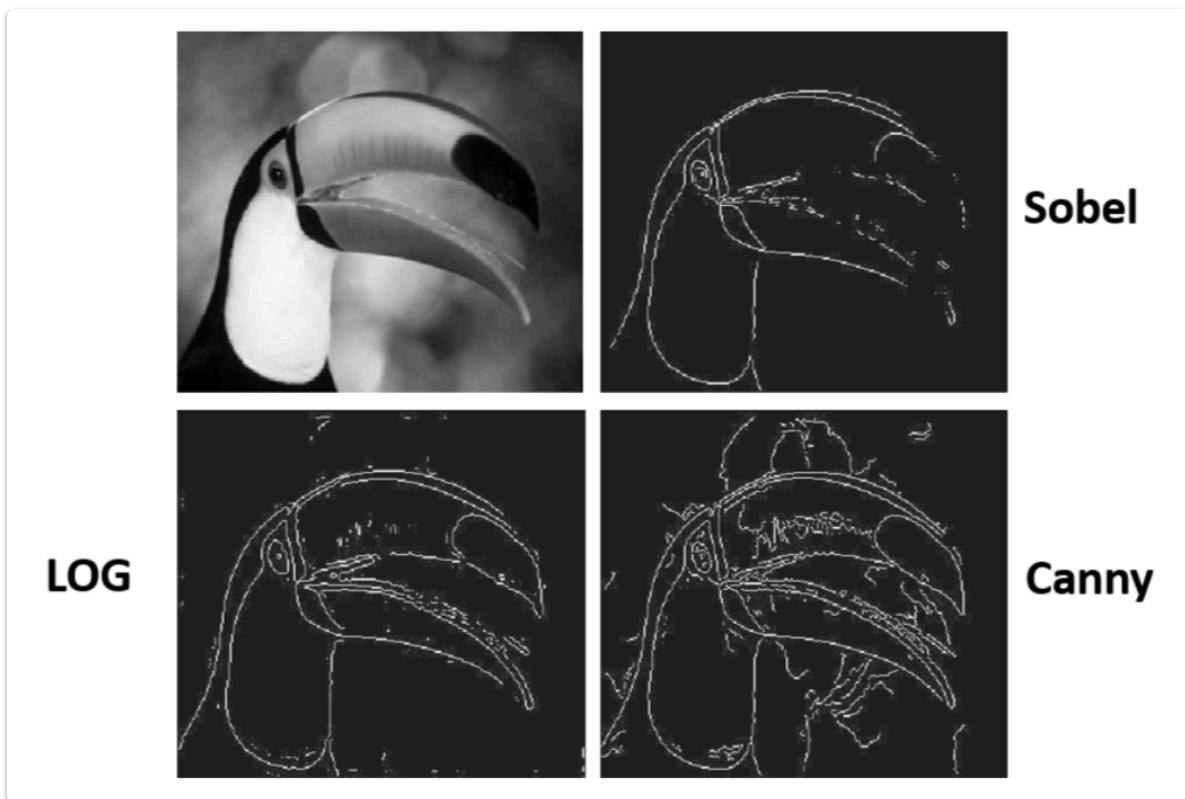
Funktionsweise des Canny-Operators

- **Gradientenverfahren:**
 - Der Canny-Operator basiert auf dem **Gradientenbetrag** des Bildes
 - **Lokale Maximums-Suche:** Sichert, dass Kanten nur **ein Pixel breit** sind

Nutzung von Single-Scale

- In der Praxis wird der Canny-Operator häufig in einer **Single-Scale-Version** angewendet
 - **Glättungsparameter σ** (Filterradius des Gaußfilters) wird angepasst, um die Kantendetektion zu verbessern
- **Vorteil:**
 - Verbesserte Kantendetektion im Vergleich zu einfachen Operatoren (wie Sobel)

Der Canny-Operator ist also besonders aufgrund seiner Fähigkeit, Kanten präzise zu erkennen, der Verwendung von Multi-Scale-Ansätze und der Optimierung durch die lokale Maximums-Suche gegenüber anderen Detektoren bevorzugt.



Kantenerkennung ist generell auch sehr wichtig für [8. Bildmerkmale - Interest Points](#)

7. Globale Operationen

Information:

- [EVC_Skriptum_CV, p.35](#) bis [EVC_Skriptum_CV, p.39](#)

Bildtransformationen

Punkt- und lokale Operationen

- **Bisherige Betrachtung:** Fokus auf **Punkt-** und **lokale Operationen**
 - Arbeitsweise: Entweder **einzelnes Pixel** oder **Pixelumgebung** wird bearbeitet
 - Siehe: [4. Punktoperationen](#), [5. Lokale Operationen](#)

Globale Operationen

- Nutzen das gesamte **Bild** als Ausgangsbasis
 - Bild wird aus dem **Bildraum** (Ortsraum) in einen anderen Raum (z. B. **Frequenzraum**) transformiert
 - **Vorteile** der Transformation:
 - Bessere Sichtbarkeit von bestimmten Bildmerkmalen
 - Bessere **Daten-Dekorrelation** und Anpassung an das menschliche visuelle System

Bild-Frequenzraum-Transformationen

- **Ziel:** Darstellung von Bilddaten in einem anderen Raum (z. B. Frequenzraum) zur besseren Analyse
 - Häufig verwendete Transformationen:
 - **Fourier-Transformation**
 - **Cosinus-Transformation**
 - Weitere Transformationen: **Wavelet-, Laplace-Transformation**

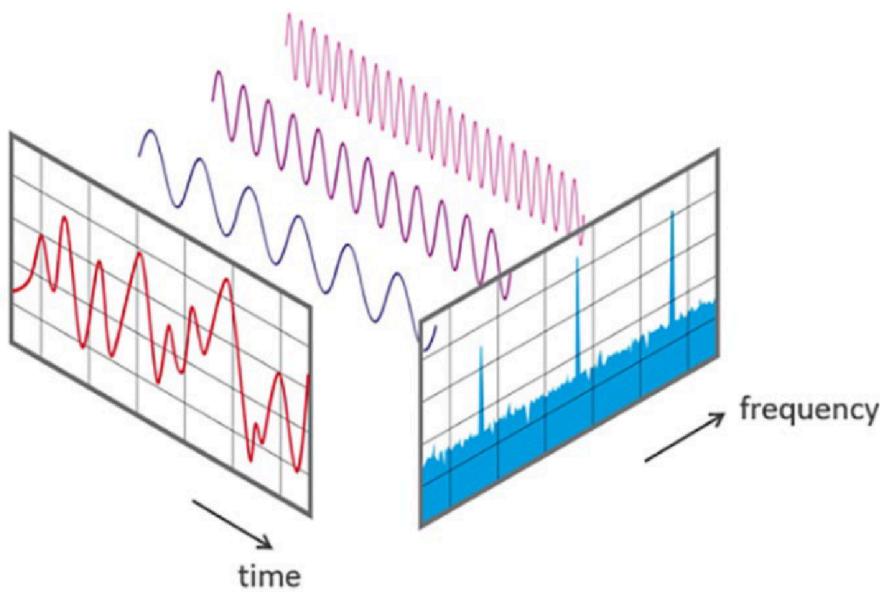
Fourier- und Cosinustransformation

- **Transformationen im eindimensionalen, kontinuierlichen Bereich:**
 - Betrachtung einer Funktion $y = f(x)$
 - Anwendung dieser Transformationen auf **zweidimensionale Bilder** (mit den Bildkoordinaten (x, y))
 - Bei der **diskreten** Betrachtung:
 - **Integrale** werden zu **Summen** (diskrete Werte)

Die Fourier- und Cosinus-Transformation sind die wichtigsten Methoden, um Bilddaten von ihrem Ortsraum in den Frequenzraum zu überführen, was viele Vorteile in der Analyse und

Verarbeitung von Bildern bietet.

Fouriertransformation



Wichtige Info vorab!

Diese Zusammenfassung bezieht sich hauptsächlich aufs [EVC_Skriptum_CV.pdf](#) und wird deshalb auch die im Skriptum verwendete Notation verwenden. In den Vorlesungsfolien wurde auf die Eulersche Identität zurückgegriffen um die \cos/\sin Ausdrücke abzukürzen.

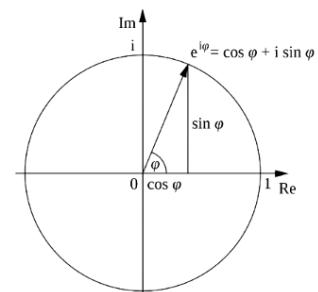
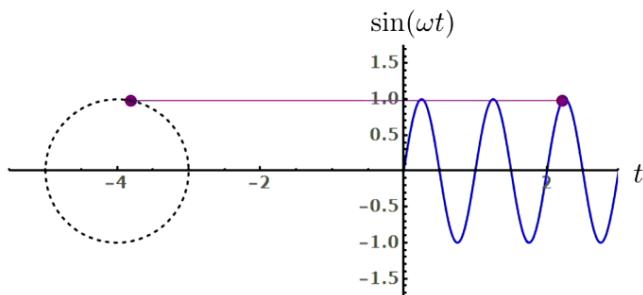
Falls beim Test die Notation aus den Slides verwendet wird hier das wichtigste zum herleiten:

Before we start - some math: Euler's Identity



Leonhard Euler
1707 - 1783

$$r e^{i \omega t} = r(\cos(\omega t) + i \sin(\omega t))$$



Frequenzbereich und harmonische Funktionen

- **Zerlegung von Bildsignalen:** Die Darstellung und Analyse von Bildern im Frequenzbereich basiert auf der Zerlegung von Bildsignalen in **harmonische Funktionen** (Sinus- und Kosinusfunktionen).

Sinus- und Kosinusfunktionen

- **Mathematische Darstellung:**
 - Sinusfunktion: $f(t) = A \sin(\omega t + \phi)$
 - Kosinusfunktion: $f(t) = A \cos(\omega t + \phi)$
- **Parameter:**
 - **A**: Amplitude (maximale Auslenkung)
 - **ϕ** : Phase (Verschiebung entlang der Zeitachse)

Beziehung zwischen Frequenz und Kreisfrequenz

- **Kreisfrequenz** (ω) und **Frequenz** (f) sind miteinander verknüpft:
$$\omega = 2\pi f$$
 - f : Frequenz in Zyklen pro Raum- oder Zeiteinheit, z. B. **1.000 Zyklen pro Sekunde** (Hertz)

Entstehung des Frequenzkonzepts

- **Ursprung des Frequenzbegriffs:** Das Konzept der Frequenzen und der Zerlegung von Schwingungen in harmonische Funktionen entstand ursprünglich aus der **Akustik** (Schall, Töne und Musik), da das menschliche Ohr ähnlich arbeitet.
 - **Schall**: Befindet sich in der **Zeitdomäne**
 - **Töne**: Haben eine **Lautstärke** (Amplitude) und eine **Frequenz** (Schwingungen pro Sekunde)

Fouriertransformation

- **Zentrale Aussage:**
 - Wellen, wie z.B. **Schallwellen**, **Wasserwellen** oder **elektromagnetische Wellen**, können mit beliebiger **Frequenz**, **Amplitude** und **Phasenlage** als **Summe von gewichteten Kosinus- und Sinusfunktionen** dargestellt werden.

Kosinus- und Sinusfunktionen

- **Kosinusfunktion:**
 - $\cos(x)$ hat den Wert **1** am Ursprung ($\cos(0) = 1$)
 - Durchläuft von $x = 0$ bis $x = 2\pi$ eine volle Periode
- **Sinusfunktion:**
 - $\sin(x)$ hat den Wert **0** am Ursprung ($\sin(0) = 0$)
 - Durchläuft ebenfalls von $x = 0$ bis $x = 2\pi$ eine volle Periode

Frequenz und Perioden

- **Periodenanzahl:**

- Für $\cos(x)$ innerhalb einer Strecke der Länge $T = 2\pi$ ist die Anzahl der Perioden 1:
 $\omega = \frac{2\pi}{T} = 1$

Verschiebung der Kosinusfunktion

- **Verschiebung:**
 - Wenn eine Kosinusfunktion entlang der x-Achse um eine Distanz ϕ verschoben wird ($\cos(x) \rightarrow \cos(x - \phi)$), ändert sich die **Phase** der Funktion.
 - ϕ bezeichnet den **Phasenwinkel** der resultierenden Funktion.
- **Sinus als verschobene Kosinusfunktion:**
 - Die **Sinusfunktion** ist eine **Kosinusfunktion**, die um $\frac{\pi}{2}$ (ein Viertelperiode) nach rechts verschoben ist.

Orthogonalität von Sinus- und Kosinusfunktionen

- **Orthogonalität:**
 - Kosinus- und Sinusfunktionen sind **orthogonal**, was bedeutet, dass sie in ihrer Form unabhängig voneinander sind.
 - Durch die Kombination von Kosinus- und Sinusfunktionen können neue sinusoidale Funktionen mit beliebiger **Frequenz**, **Amplitude** und **Phase** erzeugt werden.

Addition von Kosinus- und Sinusfunktionen

- **Addition:**
 - Wenn eine Kosinus- und eine Sinusfunktion mit der gleichen Frequenz ω und den Amplituden A bzw. B addiert werden, entsteht eine neue Sinusfunktion mit der gleichen Frequenz ω .
 - **Resultierende Amplitude** C und **Phasenwinkel** ϕ sind durch die Amplituden A und B bestimmt:

$$C = \sqrt{A^2 + B^2}, \quad \phi = \tan^{-1} \left(\frac{B}{A} \right)$$

Erweiterung auf beliebige Funktionen

- **Jean Baptiste Joseph de Fourier** (1768–1830) zeigte, dass jede **periodische Funktion** $g(x)$ mit einer Grundfrequenz ω_0 als **Summe von harmonischen Sinus- und Kosinusfunktionen** dargestellt werden kann:

$$g(x) = \sum_{k=0}^{\infty} A_k \cos(k\omega_0 x) + B_k \sin(k\omega_0 x)$$

- **Fourierkoeffizienten:**
 - A_k, B_k : Bestimmen das Gewicht der jeweiligen Kosinus- und Sinusfunktionen.

- Frequenzen der beteiligten Funktionen: Ganzzahlige Vielfache der Grundfrequenz ω_0
- **Fourieranalyse:** Berechnung der Fourierkoeffizienten aus der gegebenen Funktion $g(x)$.

Fourierintegral – Nicht periodische Funktionen

- **Fourierintegral:** Erweiterung auf **nicht periodische Funktionen**, die ebenfalls als Summe von Sinus- und Kosinusfunktionen dargestellt werden können:

$$g(x) = \int_0^\infty A_\omega \cos(\omega x) + B_\omega \sin(\omega x) d\omega$$

- **Koeffizienten** A_ω und B_ω :

- Beschreiben die Amplitude der entsprechenden **Kosinus- bzw. Sinusfunktion** bei der Frequenz ω .
- Bestimmung der Koeffizienten:

$$A_\omega = \frac{1}{\pi} \int_{-\infty}^{\infty} g(x) \cos(\omega x) dx$$

$$B_\omega = \frac{1}{\pi} \int_{-\infty}^{\infty} g(x) \sin(\omega x) dx$$

- **Spektrum der Funktion:**

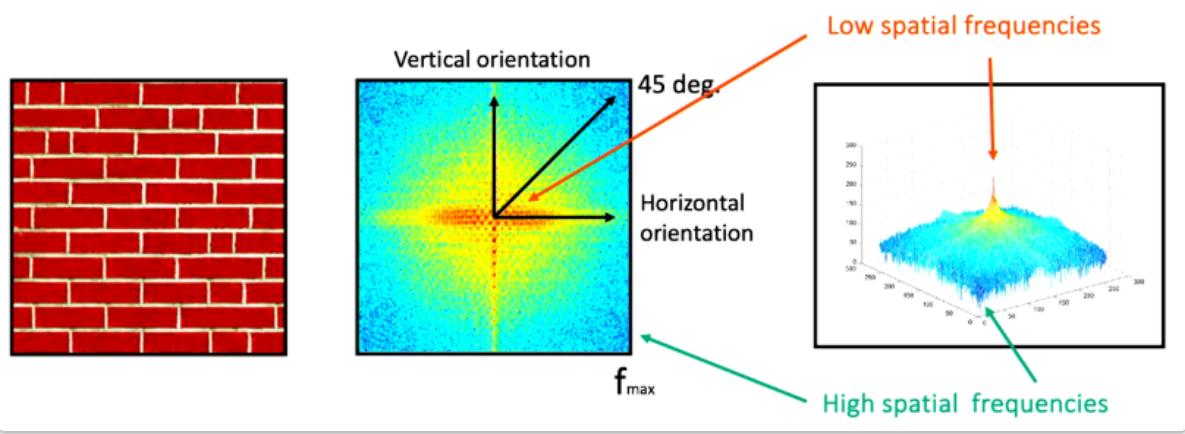
- $A(\omega)$ und $B(\omega)$ sind **kontinuierliche Funktionen**, die das **Spektrum** der Frequenzen im Signal repräsentieren.

Fouriertransformation und Fourier-Spektrum

- **Fouriertransformation:**
 - Vereinfacht die Darstellung, indem die Ausgangsfunktion $g(x)$ und das Spektrum als **komplexwertige Funktionen** betrachtet werden.
- **Fourierspektrum** $G(\omega)$:
 - Wird als **komplexe Funktion** dargestellt:

$$G(\omega) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} (\cos(\omega x) - i \sin(\omega x)) g(x) dx$$

- **Inverse Fouriertransformation:**
 - Umkehrung der Fouriertransformation zur Rekonstruktion der ursprünglichen Funktion $g(x)$ aus ihrem Spektrum $G(\omega)$.
- **Dualität von Orts- und Frequenzraum:**
 - Die Darstellung von Funktionen im Ortsraum und im Frequenzraum ist dual – sie beschreiben das gleiche Signal, jedoch auf unterschiedliche Weise.



Da das abstrakte Prinzip von Fouriertransformation Anfangs vielleicht bisschen unübersichtlich ist empfehle ich folgende Videos:

- https://www.youtube.com/watch?v=7IIR_BRkfPI (Kleine Einführung)
- <https://www.youtube.com/watch?v=spUNpyF58BY&t=477s> (Wurde auch in Vorlesung gezeigt)

Diskrete Fourier-Transformation (DFT)

Notwendigkeit der DFT

- Kontinuierliche Fouriertransformation (FT) ist für die numerische Berechnung am Computer nicht geeignet, da Bilder immer diskrete Daten enthalten.
- Die Diskrete Fourier-Transformation (DFT) stellt sowohl das Signal als auch sein Spektrum als endliche Vektoren dar, die eine diskrete, periodische Signal repräsentieren.

DFT: Vorwärtstransformation

Für ein diskretes Signal $g(u)$ der Länge M (mit $u = 0, \dots, M - 1$), wird das **Fourierspektrum** $G(m)$ für $m = 0, \dots, M - 1$ durch die **Vorwärtstransformation** berechnet:

$$G(m) = \frac{1}{\sqrt{M}} \sum_{u=0}^{M-1} g(u) e^{-i \frac{2\pi m u}{M}}$$

DFT: Inverse Transformation

Die **inverse DFT** zur Rekonstruktion des Signals $g(u)$ aus dem Spektrum $G(m)$:

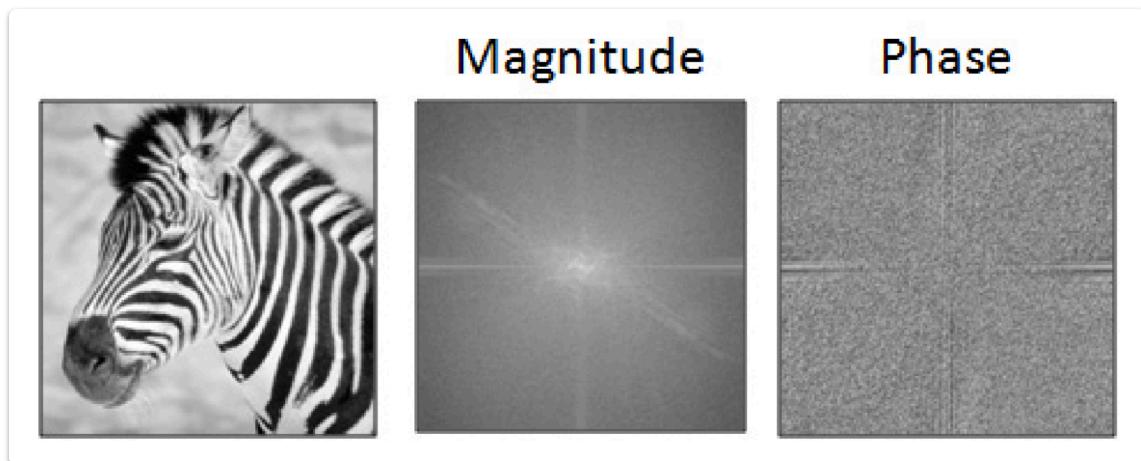
$$g(u) = \frac{1}{\sqrt{M}} \sum_{m=0}^{M-1} G(m) e^{i \frac{2\pi m u}{M}}$$

- **Beide Transformationen sind identisch:** Vorwärts- und inverse DFT sind mathematisch symmetrisch.

Eigenschaften des Fourierspektrums

- Sowohl das **Signal $g(u)$** als auch das **Fourierspektrum $G(m)$** sind **komplexwertige Vektoren** der Länge M .
- **Betrag des Fourierspektrums (Magnitude):**

$$\|G(m)\| = \sqrt{G_{\text{real}}^2(m) + G_{\text{imag}}^2(m)}$$



Wird als **Leistungsspektrum** (Powerspectrum) bezeichnet und beschreibt die Energie oder Leistung, die jede Frequenzkomponente zum Signal beiträgt.

- **Leistungsspektrum:**
 - Reellwertig und positiv.
 - Wird häufig zur **grafischen Darstellung** der Fouriertransformierten verwendet.
 - **Phaseninformation** geht verloren – das Signal kann daher nicht allein aus dem Leistungsspektrum rekonstruiert werden.
 - Unbeeinflusst von **Verschiebungen** des Signals: Ein zyklisch verschobenes Signal hat das gleiche Leistungsspektrum wie das ursprüngliche Signal.

Fast Fourier Transform (FFT)

- **Schnelle Berechnung** der DFT mit der **Fast Fourier Transform (FFT)**:
 - **Reduzierte Zeitkomplexität** von $O(M^2)$ auf $O(M \log_2 M)$.
 - **Wesentliche Zeitersparnis** bei größeren Signallängen, z.B., bei $M = 10^6$ wird die Berechnungszeit um den Faktor 10.000 reduziert.

Convolution Theorem:

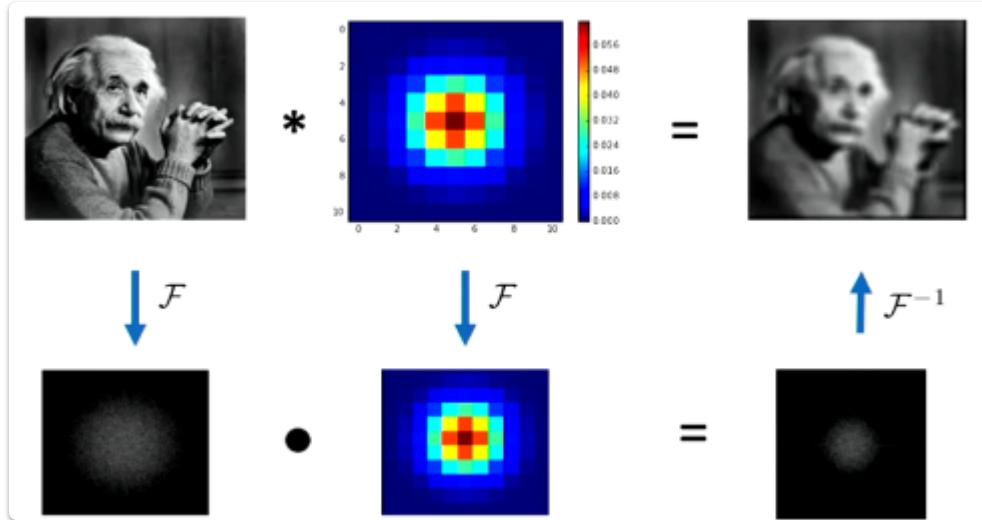
Das **Convolution Theorem** besagt, dass die Fouriertransformierte der Faltung zweier Funktionen im Zeitbereich gleich der Punktweise-Multiplikation ihrer Fouriertransformierten im Frequenzbereich ist. (Mehr dazu siehe [7. Clipping und Antialiasing](#))

Mathematisch:

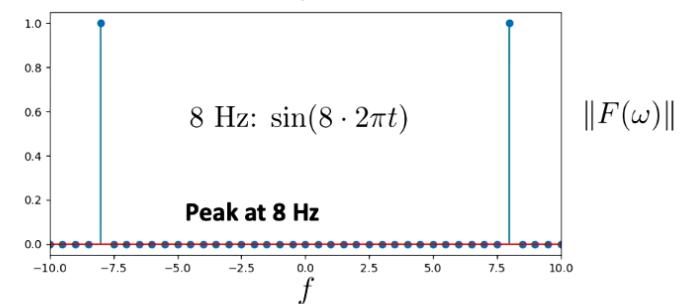
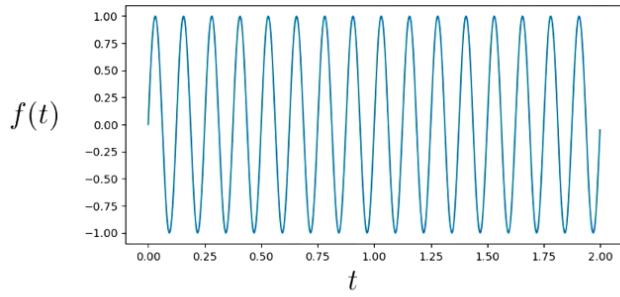
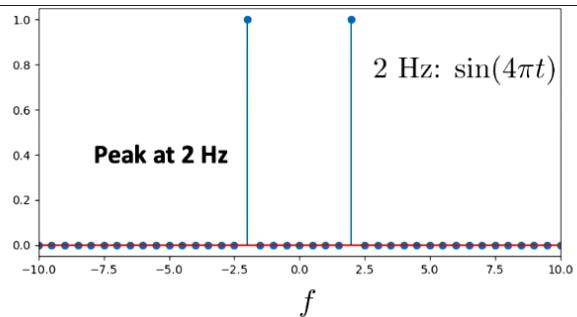
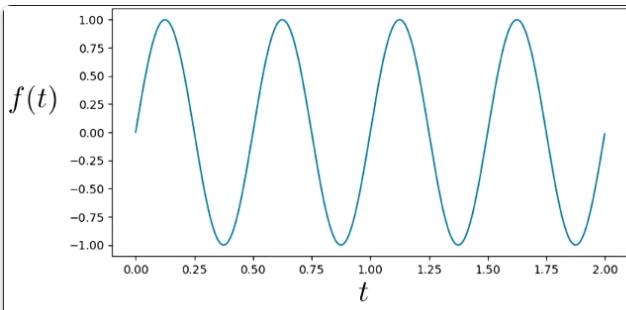
$$\mathcal{F}\{f * g\}(\omega) = F(\omega) \cdot G(\omega) Ff * g$$

Das bedeutet:

- Im Zeitbereich: Faltung der Funktionen $f(t)$ und $g(t)$.
- Im Frequenzbereich: Punktweise Multiplikation der Fouriertransformierten $F(\omega)$ und $G(\omega)$



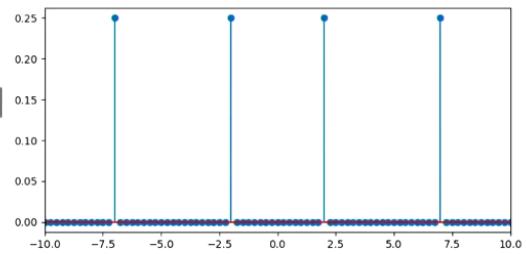
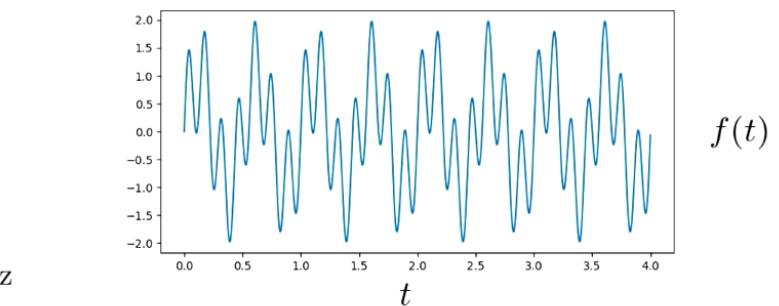
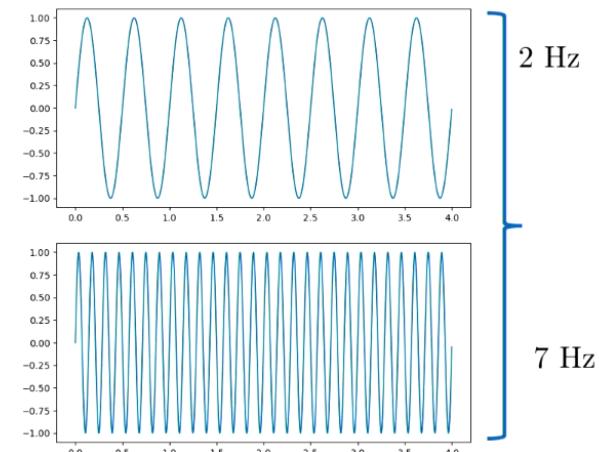
Hier noch einige Beispiele für Fourier Transformationen:



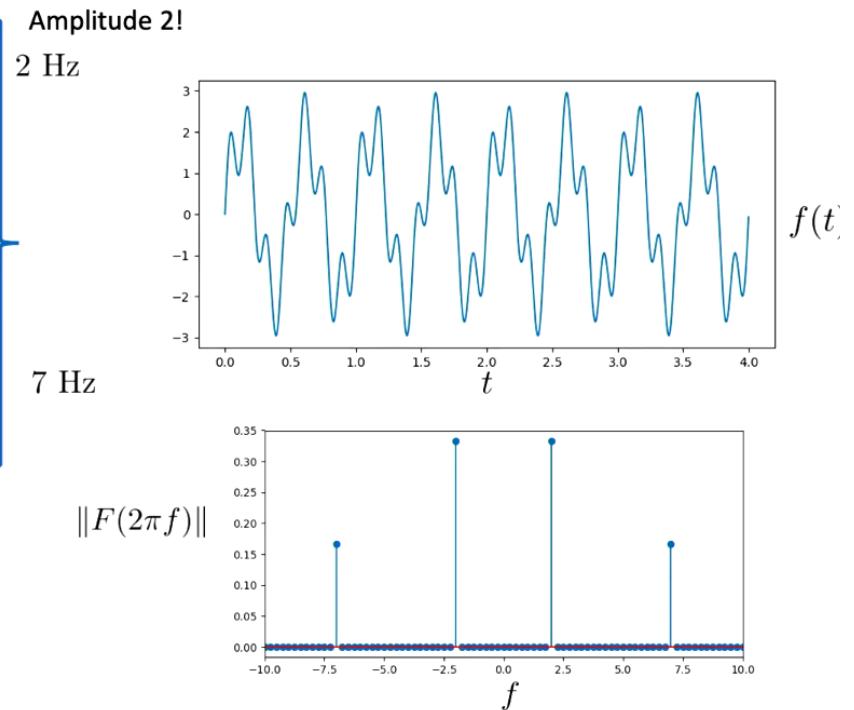
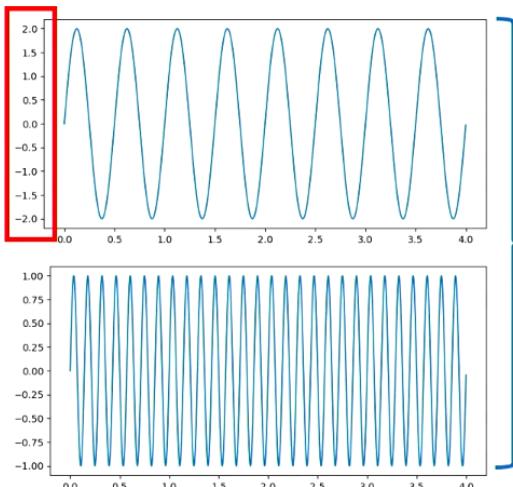
Let's look at some fourier transformations

21

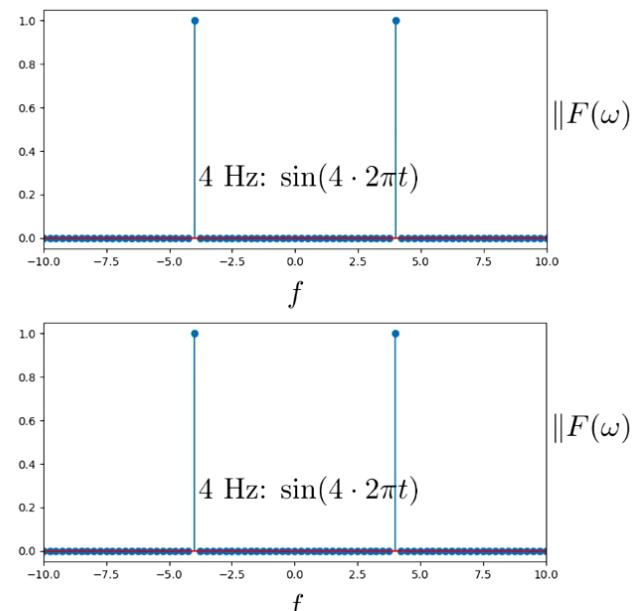
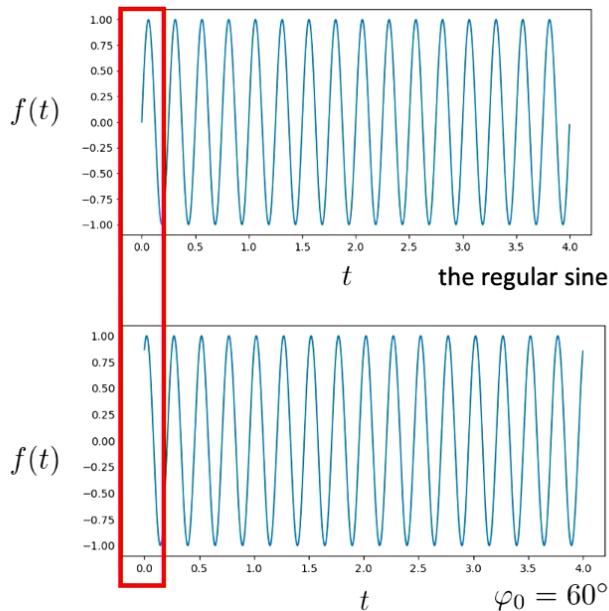
$$\omega = 2\pi f$$



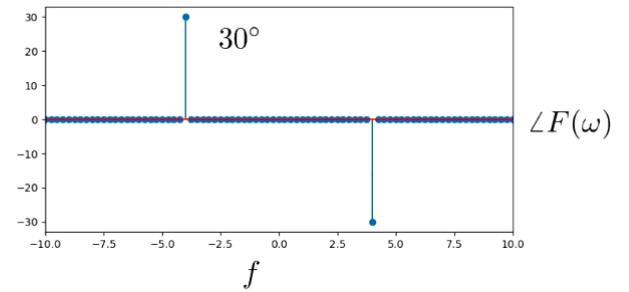
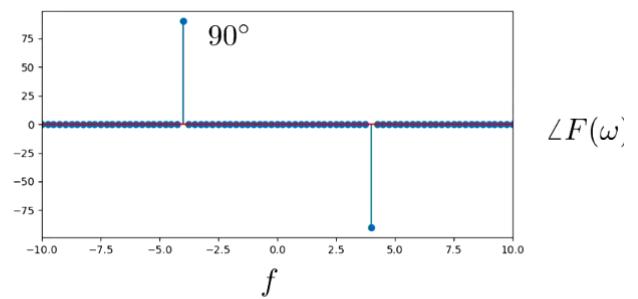
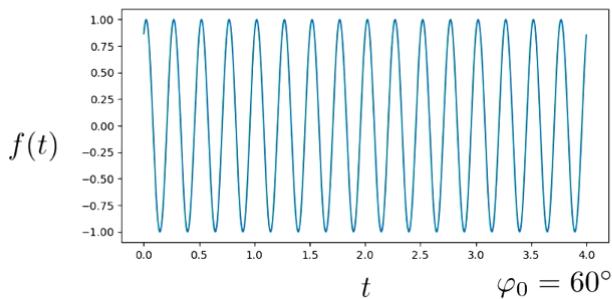
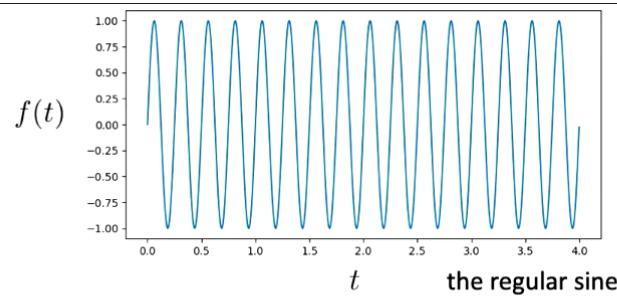
What about a sum of sines?



What about a sum of sines?



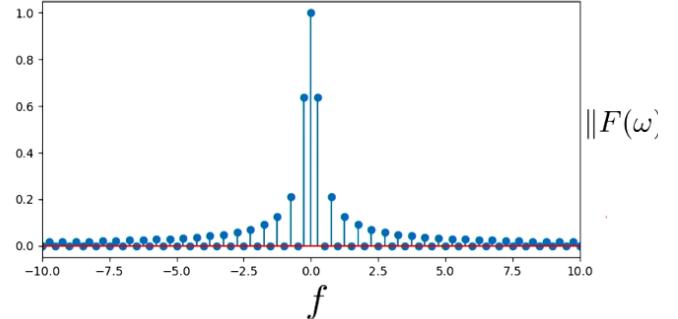
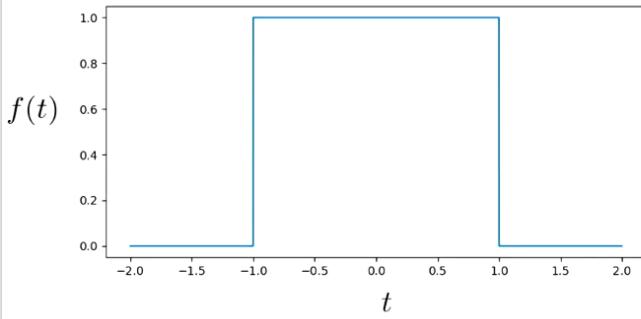
Let's look at a sine which starts at $\varphi_0 = 60^\circ$



Let's look at a sine which starts at $\varphi_0 = 60^\circ$

We can reconstruct offsets via the phase!

25



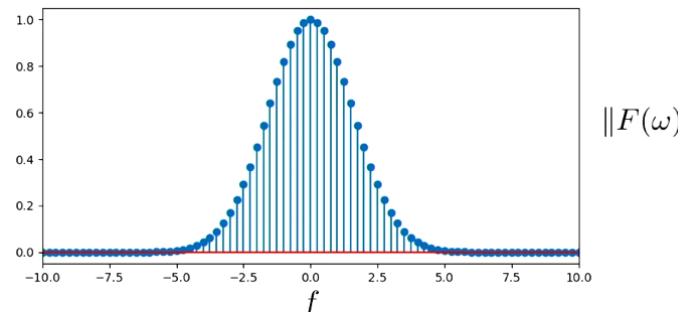
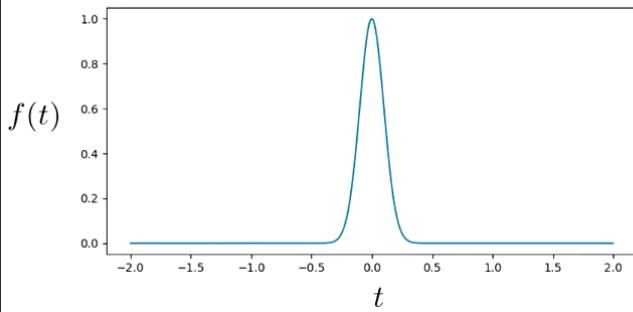
„bumping“ (harmonics) - approaches zero at ∞

Sinc-function (Spaltfunktion):

$$\text{si}(x) = \frac{\sin(\pi x)}{\pi x}$$

Fourier Transformation of a Rectangle

Gaussian with $\mu = 0, \sigma = 0.1$

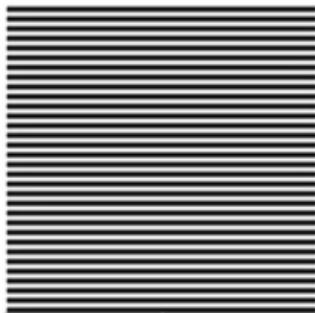


A Gaussian's spectrum is a Gaussian again! (but with σ rescaled)

Fourier Transformation of a Gaussian

Jetzt hier 2D Beispiele:

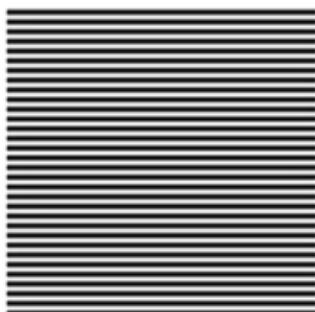
This image exclusively has **32 cycles** in **vertical** direction.



This image exclusively has **8 cycles** in **horizontal** direction.



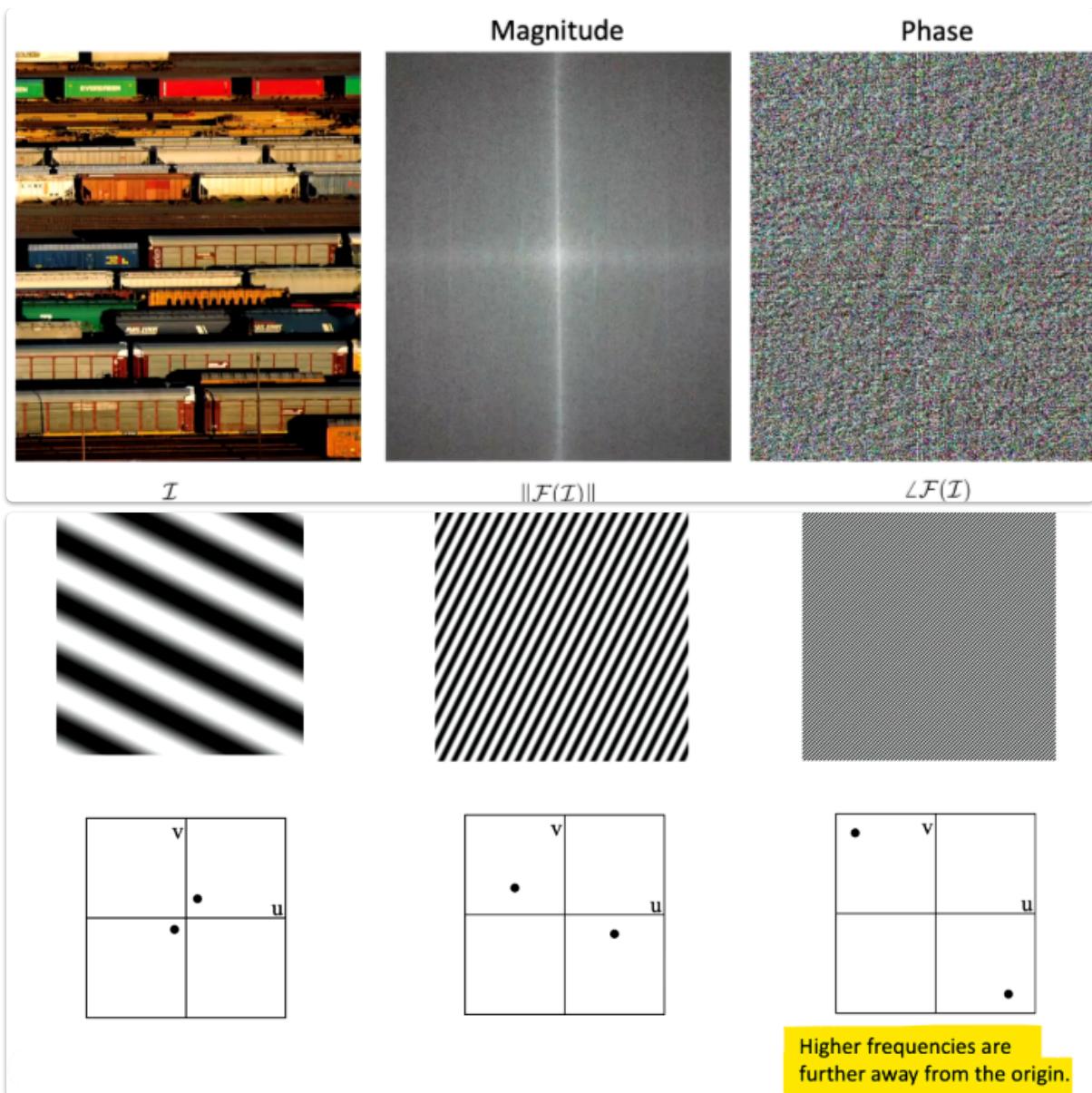
This image exclusively has **32 cycles** in **vertical** direction.



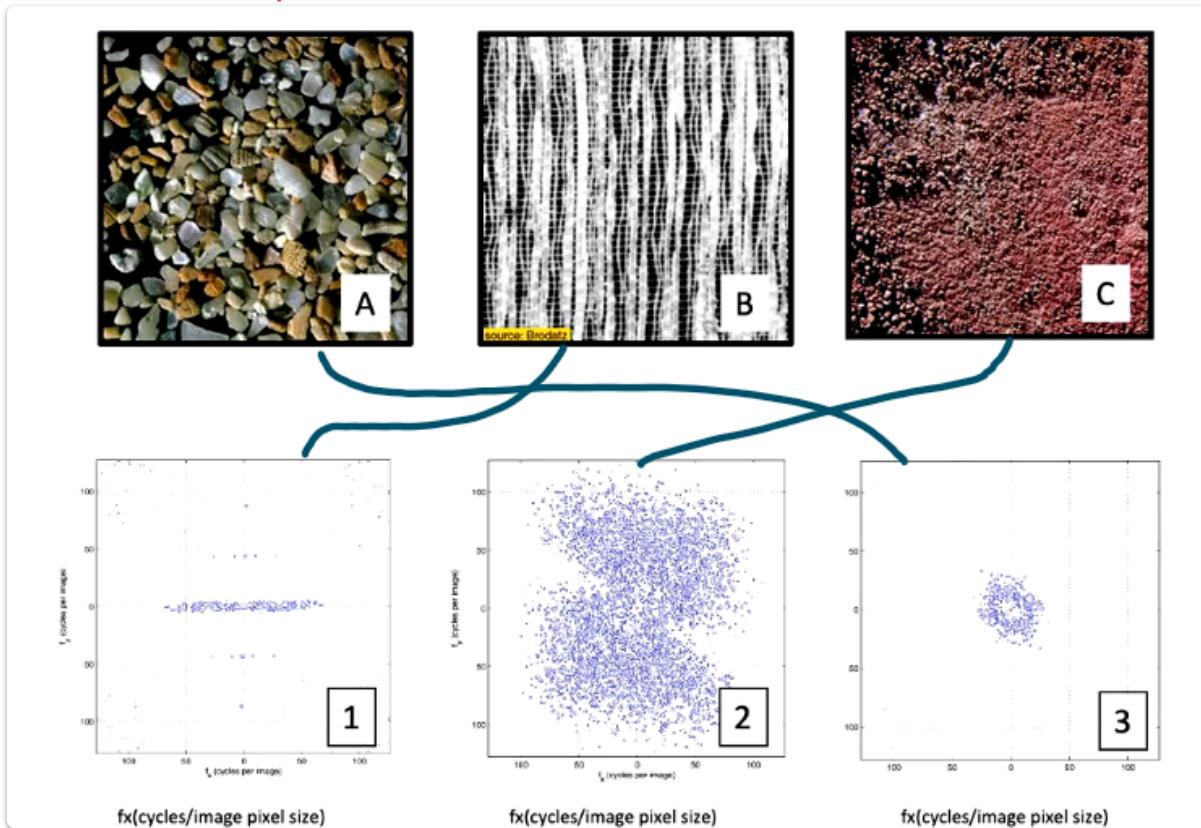
This image exclusively has **8 cycles** in **horizontal** direction.



Jetzt hier Beispiel für ein weiteres Bild:

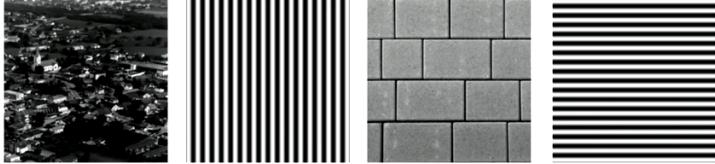


Testähnliches Beispiel:



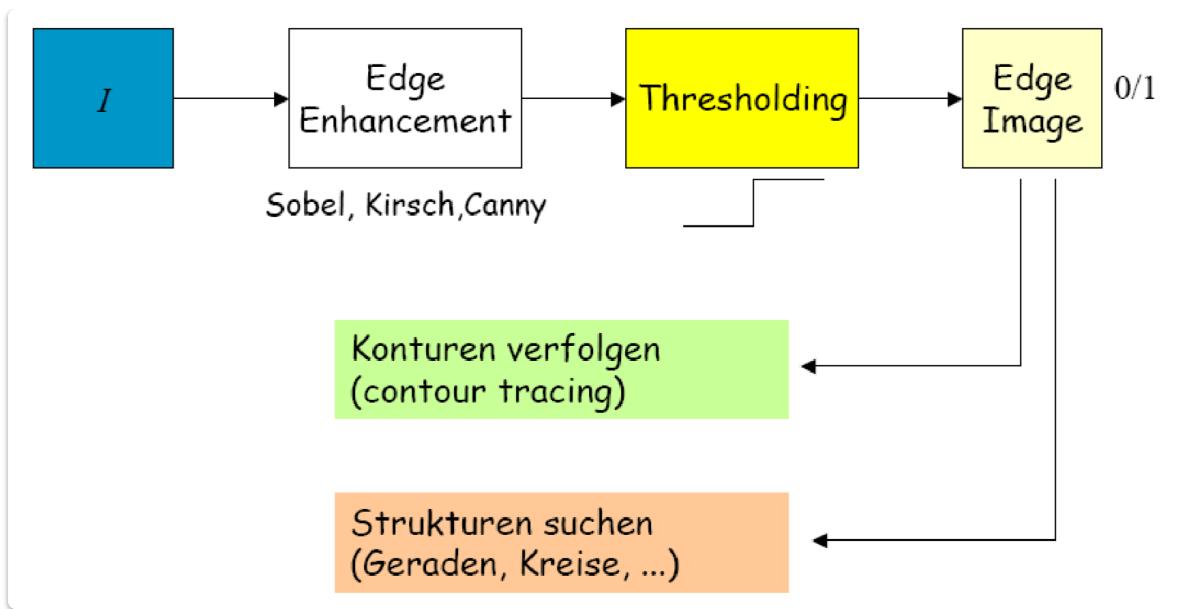
Beispiel Globale Operationen

- Die Bilder A-D zeigen den logarithmierten Betrag des Fourier-Spektrums eines Bildes. Ordnen Sie die Eingabebilder I_1 bis I_4 dem richtigen Spektrum aus A bis D zu.



Richtige Antwort: BCAD

Hough Transformation



Einführung in die Hough-Transformation

- **Ziel:** Analyse von Bildinhalten nach Kantendetektion, um größere Strukturen zu finden.
- **Problem der Kantendetektion:** Viele vermeintliche Kantenpunkte gehören in Wirklichkeit nicht zu echten Kanten und umgekehrt fehlen echte Kantenpunkte.
- **Lösungsansatz:** Kantenpunkte werden zu größeren Strukturen zusammengeführt (Konturen von Objekten), jedoch gibt es bei Unterbrechungen und Verzweigungen Schwierigkeiten.

Prinzip der Hough-Transformation

- **Begriff:** Methode zur Erkennung von geometrischen Formen in Punktverteilungen (z.B. Geraden, Kreise, Ellipsen).
- **Besonderheit:** Eignet sich besonders für die Analyse von Bildern mit geometrischen Formen, die in menschlich geschaffenen Objekten häufig vorkommen.
- **Geradenerkennung in binären Kantenbildern:**
 - Eine Gerade in 2D lässt sich mit zwei Parametern beschreiben (z.B. $y = kx + d$ mit Steigung k und Schnittpunkt d).
 - Ziel: Finden der Geradenparameter k und d , auf denen viele Kantenpunkte liegen.
 - **Problem:** Alle möglichen Geraden zu berechnen, die durch einen Punkt laufen, wäre ineffizient.

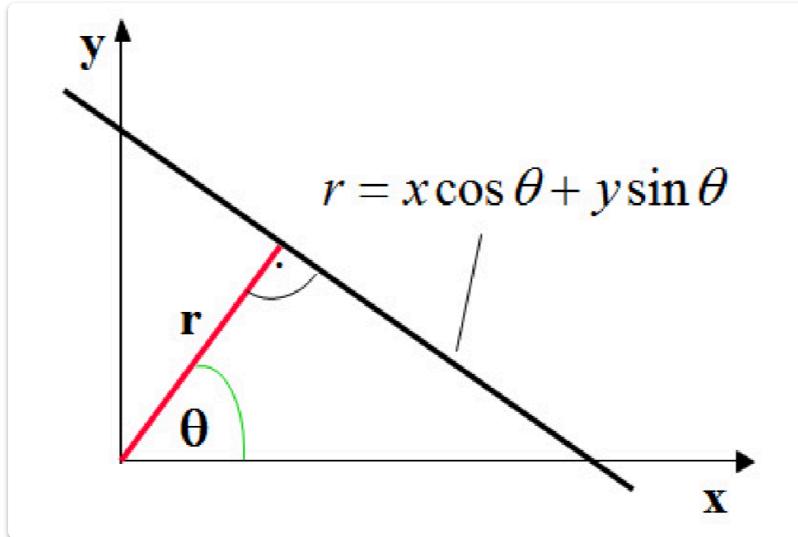
Umgekehrter Ansatz der Hough-Transformation

- **Geradendarstellung:** Statt klassischer Form $y = kx + d$ wird die Parameterform (Hessesche Normalform) verwendet:

$$r = x \cos(\theta) + y \sin(\theta)$$

- **Parameter:**
 - r : Normalabstand der Geraden zum Ursprung.

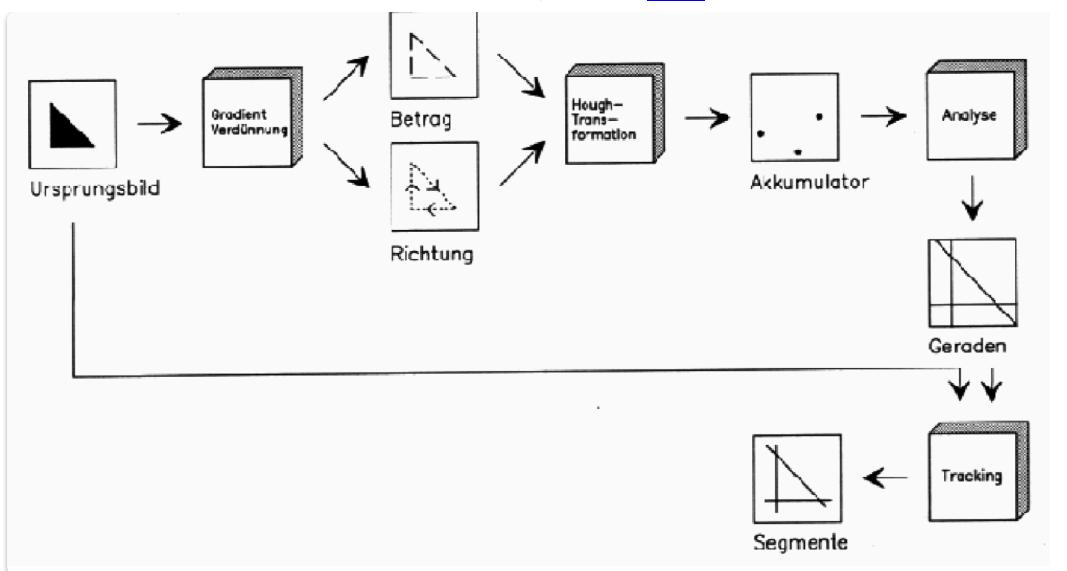
- θ : Winkel des Normalabstands zur x-Achse.



- **Erklärung:** Jede Gerade, die durch einen Punkt $p = (x, y)$ läuft, erfüllt die Gleichung. Dadurch entstehen unendlich viele mögliche Geraden, die durch den Punkt verlaufen, mit variierenden Parametern r und θ .
- **Hough-Raum:** Der Parameterraum (r, θ) wird als Hough-Raum bezeichnet. Jeder Punkt im Hough-Raum entspricht einer Geraden im Bild.

Vorgehensweise bei der Hough-Transformation

- **Kantenbild erstellen:** Aus dem Eingangsbild $I(u, v)$ wird ein Kantenbild erzeugt, wobei der Betrag und die Richtung des Gradienten (z.B. durch Sobel-Filter) berechnet wird.
- **Akkumulator-Array:**
 - Ein zweidimensionaler Akkumulator (Array) wird erzeugt, um den (r, θ) -Raum zu diskretisieren.
 - Für r und θ werden festgelegte Schritte (z.B. r in Pixeln, θ in 1° oder 5°) verwendet.
- **Eintragen der Kantenpunkte:**
 - Jeder Kantenpunkt wird in den Hough-Raum überführt und im Akkumulator eingetragen. Die Zellen des Akkumulators werden entsprechend inkrementiert.
- **Schwellwertanalyse:**
 - Wenn der Wert einer Akkumulator-Zelle einen bestimmten Schwellwert überschreitet, repräsentiert diese Zelle eine mögliche Gerade im Bild.



Nachbearbeitung der Ergebnisse

- **Tracking der Geraden:** Um die Anfangs- und Endpunkte der gefundenen Geraden zu ermitteln, wird ein Tracking-Verfahren angewendet.
 - Dazu wird das Originalbild entlang der gefundenen Geraden abgefahren, und es wird nach Stellen gesucht, an denen die Grauwertdifferenz den Schwellwert überschreitet.
 - An diesen Stellen befinden sich mit hoher Wahrscheinlichkeit Objektkonturen.

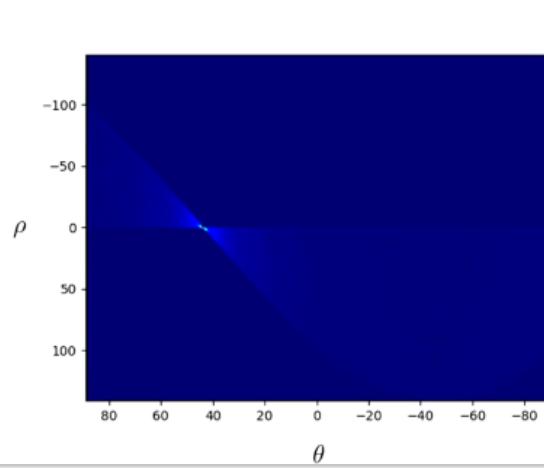
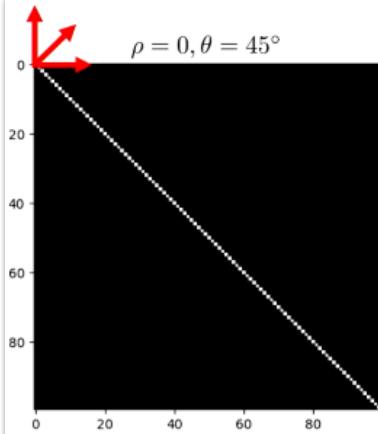
Vorteile der Hough-Transformation

- **Berücksichtigung der Kantenrichtung:** Die Richtung der Kanten im Originalbild ist bereits bekannt, was die Effizienz steigert.
- **Verwendung von Gradienteninformationen:** Die Hough-Transformation nutzt den Betrag und die Richtung der Kantenpixel für eine genauere Analyse.

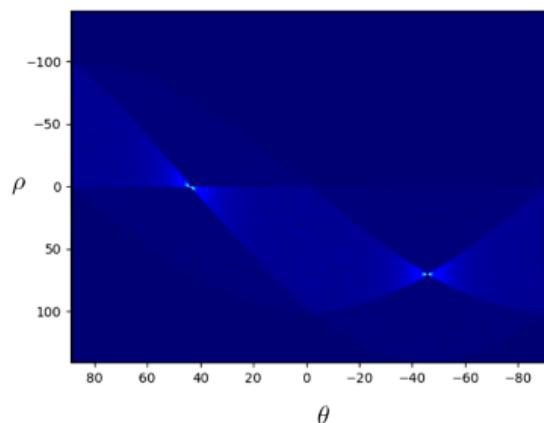
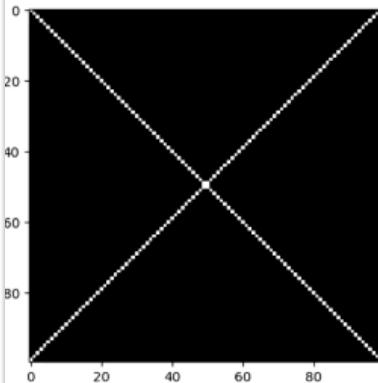
Einschränkungen

- **Komplexität:** Wenn Filter höherer Ordnung verwendet werden, müssen alle möglichen Geraden durch jeden Kantenpunkt eingetragen werden, was zu einer sinusförmigen Linie im Akkumulator führt und die Berechnungen erheblich verlangsamen kann.

Beispiele aus vo:



$$\rho_2 = \sqrt{(50^2 + 50^2)} \approx 70, \theta_2 = -45^\circ$$



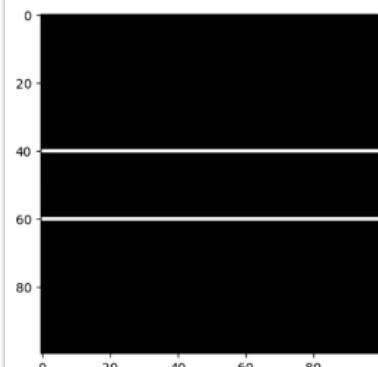
Orthogonal lines!

$$\theta_2 = -\theta_1$$

$$\theta_2 - \theta_1 = 90^\circ$$

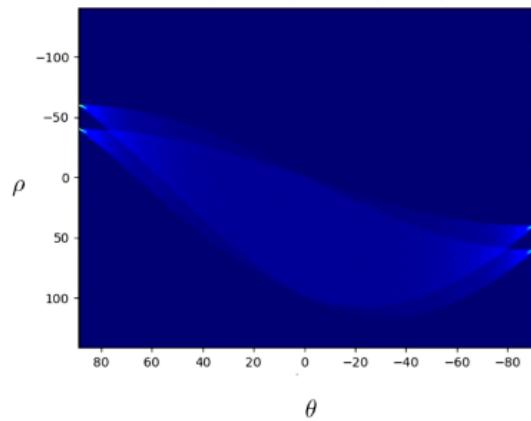
$$\theta_1 = \theta_2 = 90^\circ$$

$$\rho_1 = 40, \rho_2 = 60$$

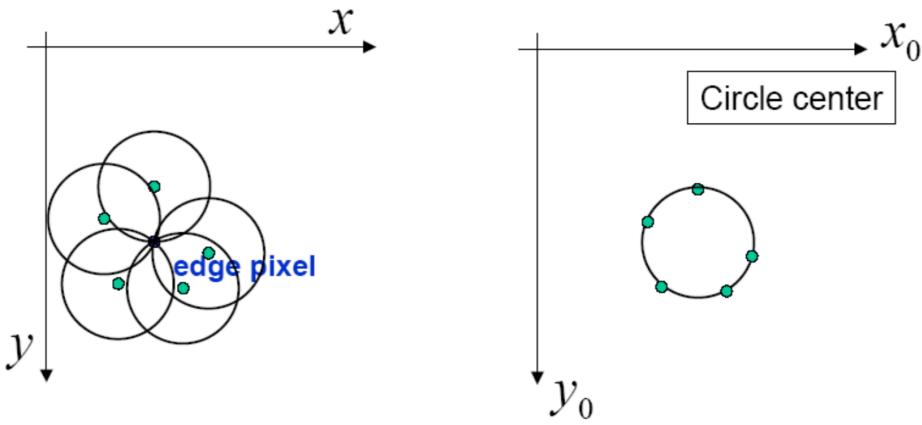


Keep in mind (depends on our parametrization)

$$\mathcal{H}(\rho, -90^\circ) = -\mathcal{H}(\rho, 90^\circ)$$



Weitere Hough Transformationen



1. Prinzip der Kreiserkennung

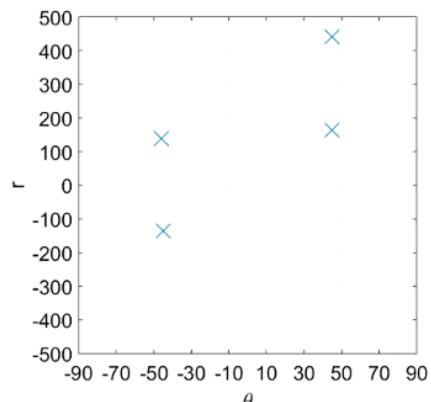
- **Ziel:** Erkennung von Kreisen im Bild.
- **Ähnlichkeit zur Geradenerkennung:**
 - Durch jeden Punkt im Originalbild wird ein Kreis mit einem festgelegten Radius gelegt.
 - Die Kreise im Hough-Transformationsraum schneiden sich in bestimmten Punkten, die Kandidaten für die Mittelpunkte der Kreise im Originalbild sind.
- **Bestimmung der Kreismittelpunkte:**
 - Die Punkte im Hough-Raum, an denen sich die meisten Kreise schneiden, entsprechen den Kreismittelpunkten.
 - Diese Punkte werden zurück in den Originalbereich transformiert, um die tatsächlichen Mittelpunkte der Kreise zu ermitteln.

2. Erweiterung auf Ellipsenerkennung

- **Ellipsenparameter:**
 - Eine Ellipse wird nicht nur durch den Mittelpunkt (x, y) beschrieben, sondern auch durch zwei Halbachsen a und b .
 - Dadurch ergibt sich ein 4-dimensionaler Hough-Raum, der berücksichtigt werden muss.
- **Hough-Raum:**
 - Für die Ellipsenerkennung müssen vier Dimensionen im Hough-Raum verwendet werden: die beiden Koordinaten des Mittelpunkts und die beiden Halbachsen.
 - Die Transformation ist komplexer als bei der Kreiserkennung, da sie mehr Parameter berücksichtigt.

Testähnliches Beispiel Hough Transformationen:

- Bei der Hough-Transformation zur Detektion von Linien werden diese in Hessescher Normalform repräsentiert: $r = x \cos(\theta) + y \sin(\theta)$. Im unten stehenden Diagramm sind 4 detektierte Linien im Hough-Raum (Akkumulator-Array) mit einem "X" markiert. Welche der folgenden Aussagen sind hier wahr bzw. falsch?



- | | | |
|--|--|--|
| Alle 4 Linien sind parallel zueinander | <input type="checkbox"/> wahr | <input checked="" type="checkbox"/> falsch |
| Mindestens eine Linie verläuft horizontal | <input type="checkbox"/> wahr | <input checked="" type="checkbox"/> falsch |
| Mindestens eine Linie verläuft vertikal | <input type="checkbox"/> wahr | <input checked="" type="checkbox"/> falsch |
| Die Start- und Endpunkte der Linien lassen sich aus dem Hough-Raum nicht bestimmen | <input checked="" type="checkbox"/> wahr | <input type="checkbox"/> falsch |
| Linien werden durch lokale Maxima im Hough-Raum repräsentiert | <input checked="" type="checkbox"/> wahr | <input type="checkbox"/> falsch |

8. Bildmerkmale - Interest Points

Quellen:

- [EVC_Skriptum_CV, p.40](#) bis [EVC_Skriptum_CV, p.45](#)

Interessenspunkte (Interest Points)

Definition

- Interessenspunkte sind charakteristische Punkte in einem Bild.
- Sie dienen der Beschreibung und dem Matching (Abgleich) von Bildern.

Eigenschaften

- Hervorstechend durch **einzigartige und unterscheidbare Merkmale**
- Basieren auf **lokalen Intensitätsunterschieden**, z. B.:
 - Ecken
 - Kanten
 - Strukturen

Zweck

- Werden zur **Bildbeschreibung** verwendet
- Besonders geeignet für **Vergleiche und Wiedererkennung** zwischen Bildern

Harris Corner Detection

Allgemeines

- Harris Corner Detection ist ein verbreiteter Algorithmus zur **Eckendetektion** in Bildern.
- Ziel: **Identifikation von Interessenspunkten**, die Ecken darstellen.

Kernidee

- Berechnung der **Harris-Matrix** M :
$$M = \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$
- I_x, I_y sind die **Ableitungen des Bildes** entlang der x- bzw. y-Achse.

Harris Response-Funktion

- Dient zur Bewertung der Matrix M an jedem Pixel:

$$R = \det(M) - k \cdot (\text{tr}(M))^2$$

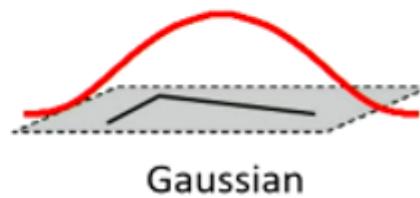
- $\det(M)$: Determinante von M
- $\text{tr}(M)$: Spur von M (Summe der Diagonaleinträge)
- k : Empirischer Parameter (typisch $0,04 \leq k \leq 0,06$)

Eckendetektion

- Für **jeden Pixel** im Bild wird R berechnet.
- Ein Pixel wird als **potenzielle Ecke** klassifiziert, wenn:
 - R ist größer als ein **vorgegebener Schwellenwert**

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

Window function $w(x, y) =$



SIFT (Scale-Invariant Feature Transform)

Allgemeines

- SIFT ist ein Verfahren zur **Dektection und Beschreibung lokaler Merkmale** in Bildern.
- Besonders **robust gegenüber**:
 - Skalierung
 - Rotation
 - Helligkeitsänderungen

Ablauf des Algorithmus

- Besteht aus mehreren Schritten wie der Skalenraumextrema-Dektection, Keypoint lokalisierung und Deskriptionsextraktion

SURF (Speeded-Up Robust Features)

Allgemeines

- SURF ist eine **Weiterentwicklung von SIFT**
- Ziel: **Schnellere und effizientere Berechnung** lokaler Merkmale

Eigenschaften

- Robust gegenüber:
 - Skalierung
 - Rotation
 - Helligkeitsänderungen

Technische Unterschiede zu SIFT

- Haar Wavelet-Responses werden zur Merkmalserkennung verwendet:
 - Vereinfachen und beschleunigen die Berechnungen
 - Geringerer Rechenaufwand bei vergleichbarer Robustheit
- SURF ist im Vergleich zu SIFT schneller, insbesondere bei großen Bildmengen oder Echtzeitanwendungen

Bildmerkmale

Definition

- Bildmerkmale sind mathematische Beschreibungen eines Bildes oder seiner Teile.
- Ziel: Repräsentation von Informationen, die besser unterscheidbar und nützlicher als reine Pixelwerte sind.
- Unterstützen die Extraktion relevanter Informationen für verschiedene Anwendungen.

Arten von Bildmerkmalen

- Globale Merkmale:
 - Beschreiben das gesamte Bild
 - Beispiel: Grauerthistogramm
 - Nachteile:
 - Schlecht unterscheidbar
 - Nicht robust genug für komplexe Anwendungen
- Lokale Merkmale:
 - Beschreiben kleine Bildregionen
 - Ziel: Mathematische Beschreibung von interessanten Bildbereichen und ihrer lokalen Nachbarschaft
 - Robust gegenüber Skalierung, Rotation, etc.
 - Anwendungen:
 - Bildregistrierung
 - Panorama-Stitching
 - 3D-Modellierung
 - Objekterkennung

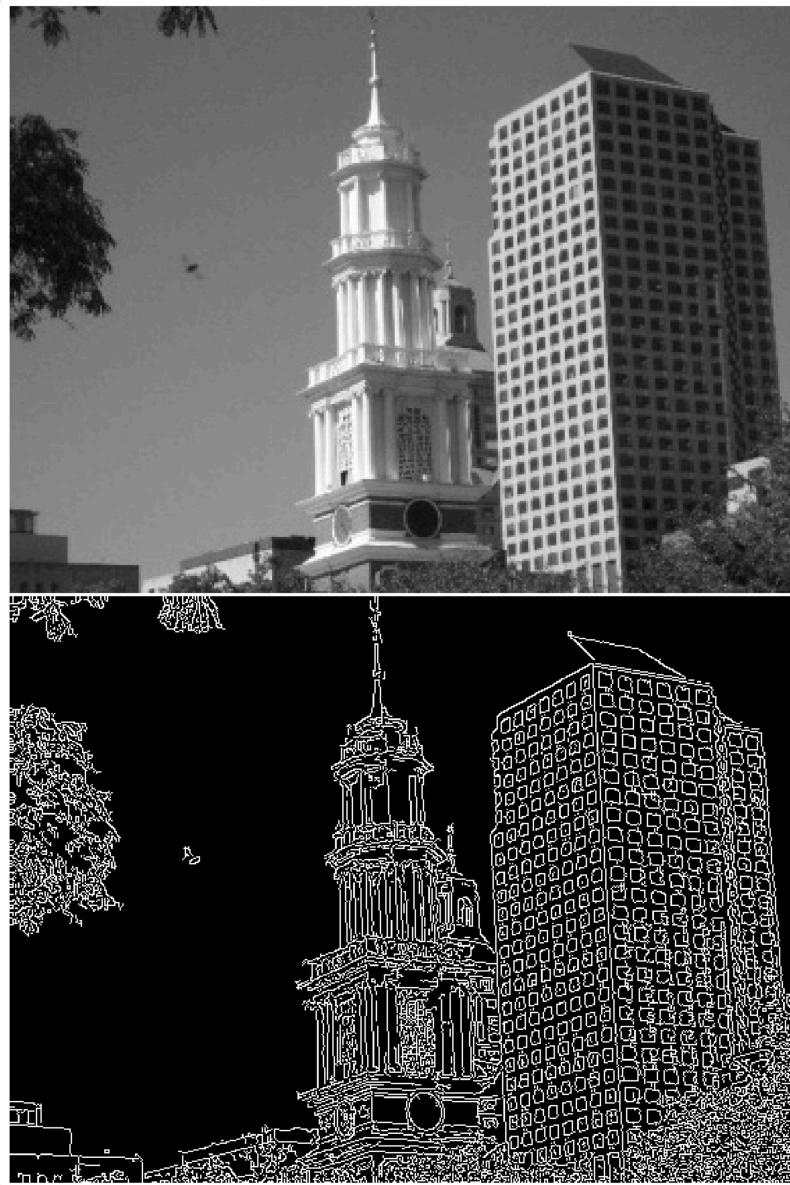
Schritte der Merkmalsextraktion

1. Merkmalsdetektion

- Identifikation von **Bildbereichen mit hohem Informationsgehalt**
- Bestimmung der **räumlichen Lage und Skalierung**
- Ergebnis: **Interest Points / Keypoints**

2. Merkmalsbeschreibung

- Erstellung eines **Merkmalsvektors**, der die **lokale Bildstruktur** beschreibt
- Die Gesamtheit aller Vektoren definiert den **Merkmalsraum** (engl. *feature space*)



Kantenerkennung

mehr siehe [6. Kantenfilterung](#)

Bedeutung von Kanten

- Kanten enthalten **nicht redundante** Informationen eines Bildes.
- Sie stellen **Unstetigkeiten** in der Helligkeit dar – also **starke Helligkeitsveränderungen**.

Ziel der Kantendetektion

- Finden von **Punkten mit plötzlichen Intensitätsänderungen** im Bild.
- Ergebnis: **Menge von Pixeln**, an denen sich **Unstetigkeitsstellen** befinden.

Vorteile

- **Reduktion der Datenmenge**:
 - Unwichtige, redundante Informationen werden **herausgefiltert**
 - Nur die **strukturell relevanten Bildinformationen** (hohe Frequenzen) bleiben erhalten

Einschränkungen

- **Nicht rotationsinvariant**
- **Nicht skalierungsinvariant**
- Dadurch sind gleiche Merkmale in unterschiedlichen Bildern **schwer vergleichbar**

Interest Points

Eigenschaften von Eckpunkten

- **Auffällige Bildstellen** – sowohl für Menschen als auch für Algorithmen



- **Robuste Merkmale**:
 - Entstehen **nicht zufällig** in 3D-Szenen
 - **Zuverlässig lokalisierbar** bei:
 - unterschiedlichen Blickwinkeln
 - verschiedenen Beleuchtungsbedingungen

Anforderungen an einen guten Corner Detector

- Zuverlässige Erkennung trotz Bildrauschen
- Hohe Lokalisierungsgenauigkeit
- Effiziente Berechnung

Grundlagen der Eckpunkt detektion

- **Kante:** Bereich mit starkem Helligkeitsgradient in **einer Richtung**, orthogonal dazu gering
- **Ecke:** Punkt mit **starkem Helligkeitsunterschied in mehreren Richtungen gleichzeitig**

Allgemeine Eigenschaften eines Interest Points

1. **Mathematisch eindeutig definierbar**
2. **Eindeutige Position** im Bildraum
3. **Hoher lokaler Informationsgehalt**
4. **Stabilität** gegenüber:
 - Lokalen Störungen
 - Globalen Störungen (z.B. perspektivische Verzerrung, Beleuchtungsvariation)
5. **Skalierungsinvianz**

Eckendetektion

Detektionsprinzip

- Eckendetektoren reagieren nicht nur auf **Ecken**, sondern auch auf Bildregionen mit **hoher Variabilität in alle Richtungen**.
- Ein **Eckpunkt liegt vor**, wenn der **Gradient in mehreren Richtungen** stark ausgeprägt ist.

Abgrenzung zu Kanten

- **Kanten:** Gradient stark **in einer Richtung** → **kein Eckpunkt**
- **Eckpunkt:** Starke Gradienten in **mehreren Richtungen gleichzeitig**

Definition eines Eckpunkts

- **Schnittpunkt zweier Kanten**
- **Punkt mit zwei dominanten Kantenrichtungen** in seiner lokalen Umgebung
- Kann auch sein:
 - **Endpunkt einer Linie**
 - **Punkt auf einer Kurve mit maximaler Krümmung**

Anforderungen an die Detektion

- **Isotrope Erkennung:** Unabhängig von Orientierung
- **Robustheit** gegenüber:
 - **Unterschiedlichen Lichtverhältnissen**

- Geometrischen Transformationen wie:

- Translation
- Rotation

Qualitätskriterium für Eckendetektoren

- Wiedererkennung derselben Ecke unter verschiedenen Bedingungen (Beleuchtung, Transformation)

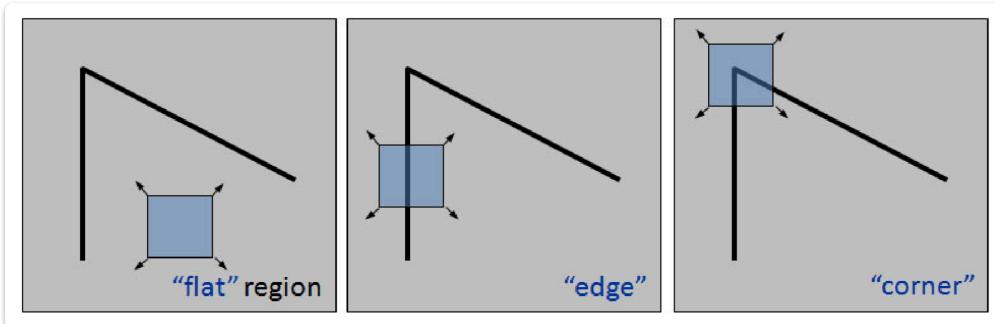
Moravec-Eckendetektor

Ursprung

- Einer der ersten Eckendetektoren
- Entwickelt von Hans P. Moravec
- Einführung des Begriffs "Points of Interest"

Grundidee

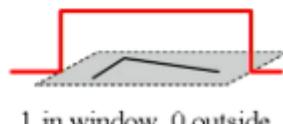
- Eine Ecke wird als Punkt mit geringer Ähnlichkeit zum verschobenen Fensterinhalt definiert.
- Verwendet ein lokales Fenster $w(x, y)$ zur Analyse der Intensitätsveränderungen.
- Fenster wird in vier Richtungen leicht verschoben:
 - Für jede Verschiebung wird die Intensitätsveränderung berechnet:
 - $I(x + u, y + v)$: Intensität an verschobener Stelle
 - $I(x, y)$: ursprüngliche Intensität



$$E(u, v) = \sum_{x, y} w(x, y) [I(x+u, y+v) - I(x, y)]^2$$

Window function Shifted intensity Intensity

Window function $w(x, y) =$



Interest Value

- Für jede Pixelposition wird ein Interest Value berechnet:
 - Entspricht der minimalen Intensitätsänderung bei allen Verschiebungen

Interpretation der Interest Values

- Geringe Veränderungen in alle Richtungen → flache Region
- Hohe Veränderung nur in einer Richtung → Kante
- Hohe Veränderung in mehreren Richtungen → Ecke

Nachteile des Moravec-Operators

- Anisotropische Antwort:
 - Kanten, die nicht exakt in Verschieberichtungen verlaufen, können fälschlich als Ecken erkannt werden
- Empfindlich gegenüber Bildrauschen:
 - Besonders entlang von Kanten
 - Betrachtet nur Minimale Intensitätsänderungen, nicht deren Verteilung

Harris Eckendetektor

Verbesserung gegenüber Moravec

- Statt absoluter Pixeldifferenzen (wie bei Moravec) wird die Variation der lokalen Bildstruktur analysiert.
- Verwendet die Bildgradienten zur Erkennung von Ecken:
 - An einem Eckpunkt sind die Gradienten sowohl in der Hauptrichtung als auch orthogonal dazu signifikant

Vorteile

- Zuverlässigere Detektion und Wiedererkennung von Ecken
- Isotrope Antwort:
 - Rotationsinvariant
 - Kein bevorzugter Richtungsverlauf wie bei Moravec

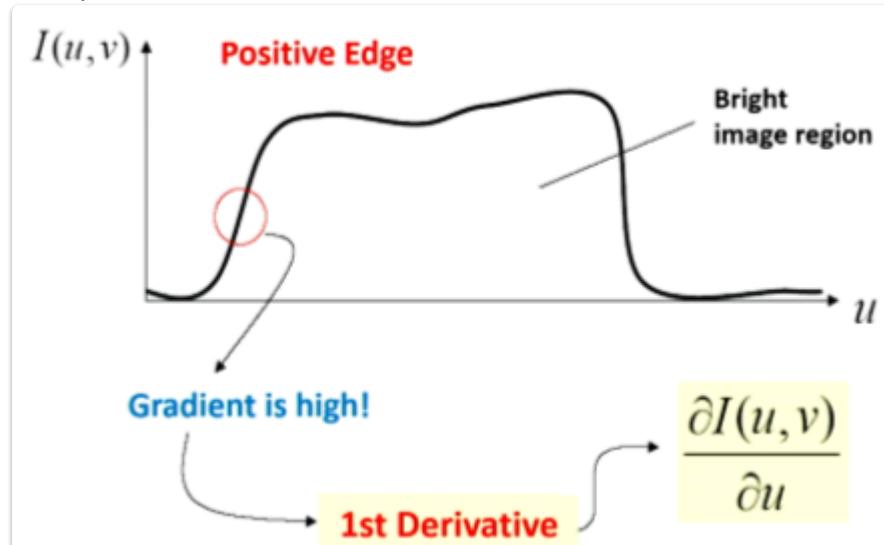
Nachteile

- Höherer Rechenaufwand durch komplexere Berechnungen
- Nicht skalierungsinvariant (wie auch der Moravec-Detektor)

Skaleninvariante Merkmalsumwandlung (SIFT)

Allgemeines

- **SIFT (Scale-Invariant Feature Transform)**: Algorithmus zur **Dektection und Beschreibung** lokaler Merkmale in Bildern.
 - Entwickelt von **David Lowe** im Jahr **1999** und in den **USA patentiert**.
 - **Eigenschaften**:
 - **Skalierungs invariant**: Unabhängig von der Bildgröße
 - **Rotationsinvariant**: Unabhängig von der Orientierung des Bildes
 - **Robust gegenüber**:
 - Affinen Transformationen
 - Beleuchtungsveränderungen
- Was passiert an einer Ecke?

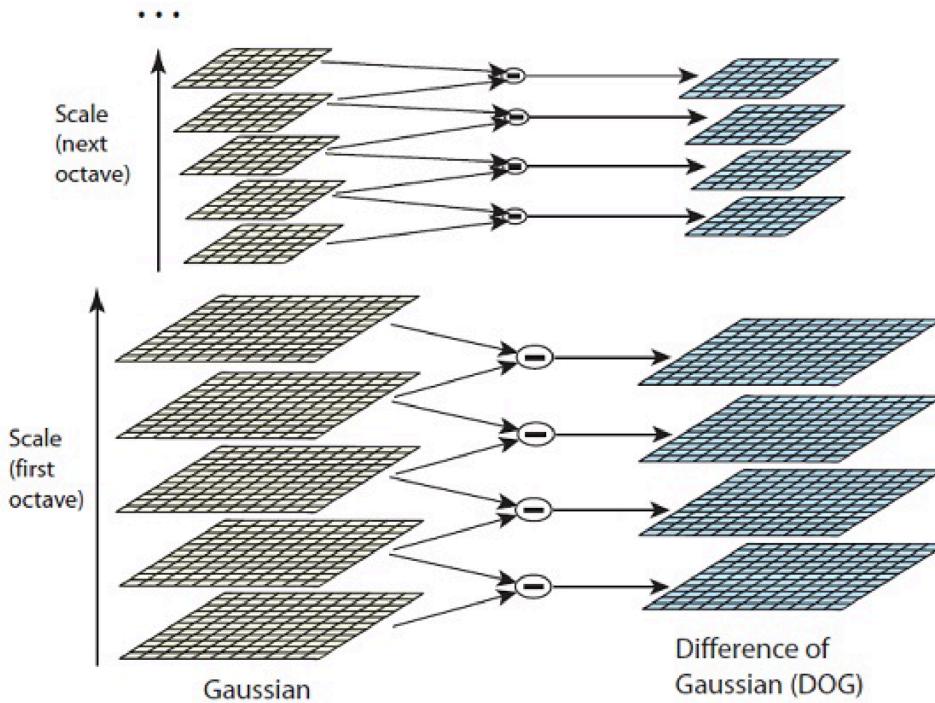


- Die Umwandlung erfolgt in **4 Schritten**:

1. Finden von Interest Points – Skalierung

Difference of Gaussians (DoG)

- SIFT verwendet eine **Difference-of-Gaussians (DoG)**-approximierte **Laplacepyramide**.
 - Der Hauptunterschied zur **ursprünglichen Laplacepyramide**:
 - Es erfolgen **keine Größenänderungen** zwischen den Ebenen, die Auflösung bleibt konstant.



- Functions for determining scale

$$f = \text{Kernel} * \text{Image}$$

Kernels:

$$L = \sigma^2 (G_{xx}(x, y, \sigma) + G_{yy}(x, y, \sigma))$$

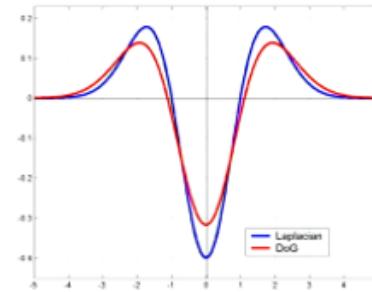
(Laplacian)

$$DoG = G(x, y, k\sigma) - G(x, y, \sigma)$$

(Difference of Gaussians)

where Gaussian:

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



DoG Filtering



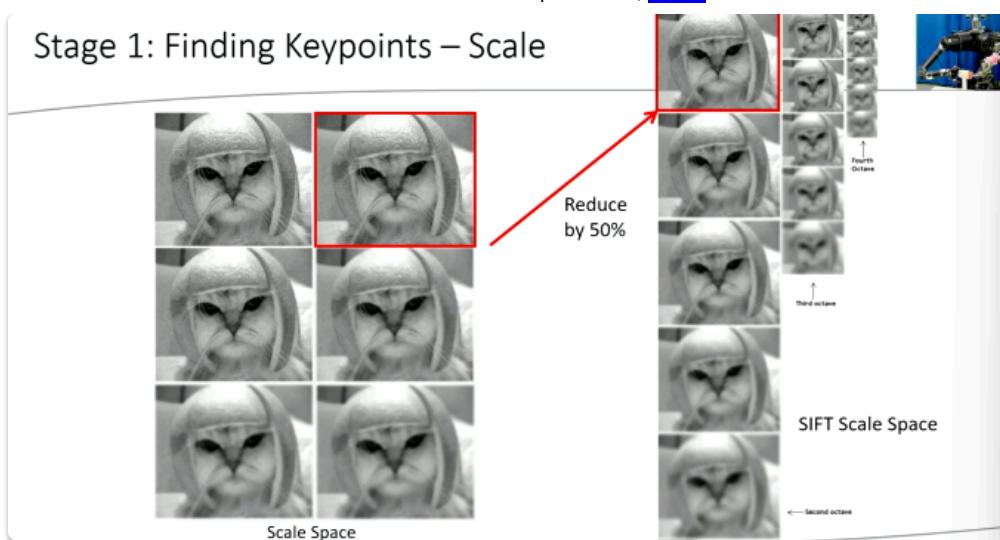
- Convolution with a variable-scale Gaussian

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y),$$

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

- Difference-of-Gaussian (DoG) filter $G(x, y, k\sigma) - G(x, y, \sigma)$

- Convolution with the DoG filter $D(x, y, \sigma) = L(x, y, k\sigma) - L(x, y, \sigma)$



Oktaven und Anwendung der Gaußfilter

- **Oktave:** Eine Reihe von **5 aufeinanderfolgenden Gaußfiltern**, die insgesamt **4 bandpassgefilterte Ergebnisse** erzeugen.
- Nach der ersten Oktave wird die **nächste Ebene der Gaußpyramide** (2. Oktave) auf die gleiche Weise behandelt.
 - Jede Oktave reduziert die **Auflösung** des Bildes:
 - Erste Oktave: **Originalauflösung**
 - Zweite Oktave: **halbe Auflösung** in beiden Richtungen, und so weiter.

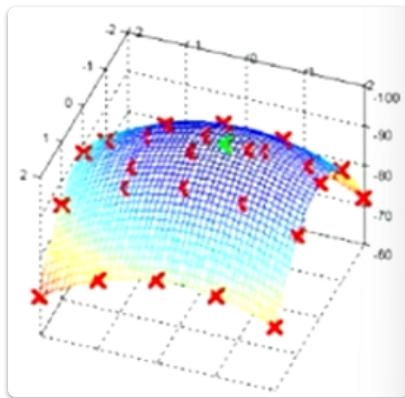
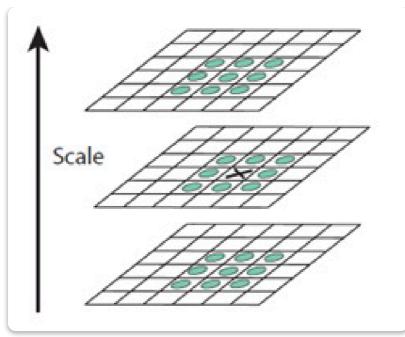
DoG und Lokalisierung von Merkmalen

- **DoG Bilder** (oder auch **Laplacebilder**):
 - Trennen die **Frequenzen** im Bild
 - Dies hilft bei der **Lokalisation von Kanten und Ecken**

Rolle der Oktaven

- **Extrema** werden durch das **Vergleichen von Bildpunkten** zwischen verschiedenen Oktaven lokalisiert.
- Die **verschiedenen Oktaven** sind entscheidend für die Erkennung von **Interest Points** auf unterschiedlichen Skalen.

2. Finden von Interest Points - Position



Dreiteiliger Prozess zur Positionierung von Interest Points

a) Lokalisierung der Extrema (Maxima/Minima) in den DoG-Bildern

- **Erster Schritt:** Grobe Lokalisierung der **Maxima und Minima**.
- **Vorgehensweise:**
 - Iterative Überprüfung der **Nachbarn** im DoG-Skalenraum.
 - Es werden **26 Abfragen** für jedes Pixel durchgeführt:
 - 9 Punkte in der gleichen Ebene
 - 9 Punkte in der Ebene darüber
 - 8 Punkte in der Ebene darunter
- **Kriterien:** Ein Interest Point ist **größer** oder **kleiner** als alle 26 benachbarten Punkte.

b) Bestimmung der Position der Extrema mit Subpixel-Genauigkeit

- **Ziel:** Die genauen Positionen der **Maxima und Minima** zu bestimmen, da sie nicht unbedingt auf einem Pixel liegen, sondern auch zwischen den Pixeln.
- **Vorgehensweise:**
 - Nutzung der **Taylorreihenentwicklung**, um Subpixelwerte um den approximierten Interest Point zu berechnen.

c) Eliminierung ungeeigneter Interest Points

- **Gründe für Eliminierung:**
 - Punkte liegen entlang von **Kanten**.
 - Punkte besitzen **nicht genügend Kontrast**.
- **Schritte zur Eliminierung:**

1. Kanten eliminieren:

- Ähnlich wie der Harris-Detektor:
 - Berechnung der **zwei Gradienten**, die senkrecht zueinander stehen.
 - **Möglichkeiten**:
 - **Flache Region**: Beide Gradienten sind klein.
 - **Kante**: Ein Gradient ist groß (senkrecht zur Kante), der andere klein (entlang der Kante).
 - **Ecke**: Beide Gradienten sind groß (wird als guter Interest Point betrachtet).

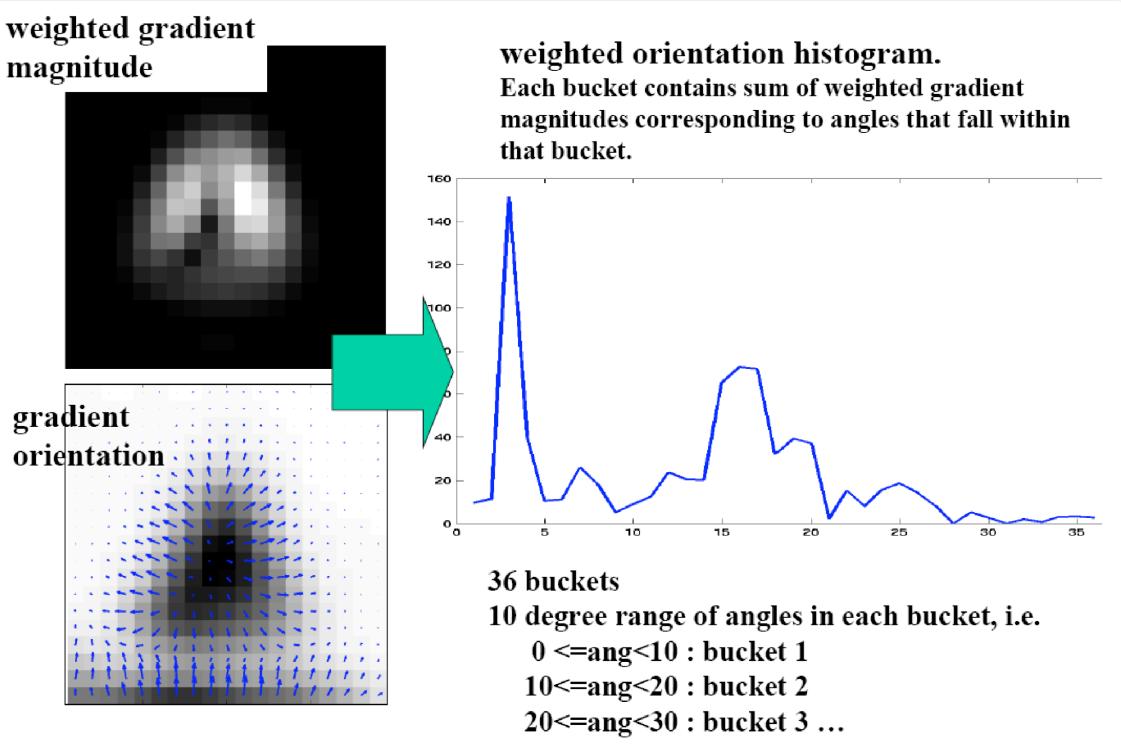
2. Geringer Kontrast:

- Kontrolle der **Intensitätswerte** im DoG-Bild.
- Wenn der Wert unter einem bestimmten **Schwellenwert** liegt, wird der Interest Point verworfen.

3. Finden von Interest Points – Orientierung

Ziel

- **Rotationsinvarianz**: Der SIFT-Merkalsvektor für denselben Interest Point soll auch in einem rotierten Bild denselben Wert haben.



Vorgehensweise

1. Erhebung der Gradienten:

- Um den **rotationsinvarianten Merkmalsvektor** zu erstellen, werden die **Längen (Gradientenbeträge)** und **Richtungen der Gradienten** rund um den Interest Point

gesammelt.

2. Bestimmung der dominanten Gradientenrichtung:

- Die **dominante Gradientenrichtung** einer Region wird durch Bestimmung der Richtung des stärksten Gradienten in der Umgebung des Interest Points ermittelt.

3. Histogramm der Gradientenrichtungen:

- Ein Histogramm wird erstellt, das die **360 Grad** der Gradientenrichtungen in **36 Bins** aufteilt (jeweils **10 Grad** pro Bin).
 - Beispiel: Eine Gradientenrichtung von **18,7 Grad** fällt in den Bin von **10-19 Grad**.
 - Der Wert des jeweiligen Bins entspricht der **Länge des Gradienten** an diesem Punkt.

4. Dominante Richtung:

- Der **Bin** mit dem höchsten Wert im Histogramm wird als die **dominante Gradientenrichtung** des Interest Points festgelegt.

Ergebnis

- Der Interest Point erhält eine **dominante Orientierung**, wodurch der SIFT-Merkalsvektor **rotationsinvariant** wird.

4. Erstellen einer Beschreibung der Merkmale

- Erstellung einer **eindeutigen Signatur** (Merkmalsvektor) für jeden Interest Point, die **robust gegenüber Störungen** und **rotations- sowie beleuchtungsinvariant** ist.

Schritte zur Signaturerstellung

1. 16x16 Fenster um den Interest Point:

- Ein **16x16 Fenster** wird um den Interest Point in seiner jeweiligen **Skalierung** definiert.
- Dieses Fenster wird in **16 4x4 Subfenster** unterteilt (siehe Abbildung 37).

2. Berechnung der Gradienten in den Subfenstern:

- Innerhalb jedes **4x4 Subfensters** werden die **Richtungen** und **Längen der Gradienten** berechnet.
- Die Gradientenrichtungen werden in einem **Histogramm mit 8 Bins** zusammengefasst:
 - 0–44 Grad → 1. Bin
 - 45–89 Grad → 2. Bin
 - usw.

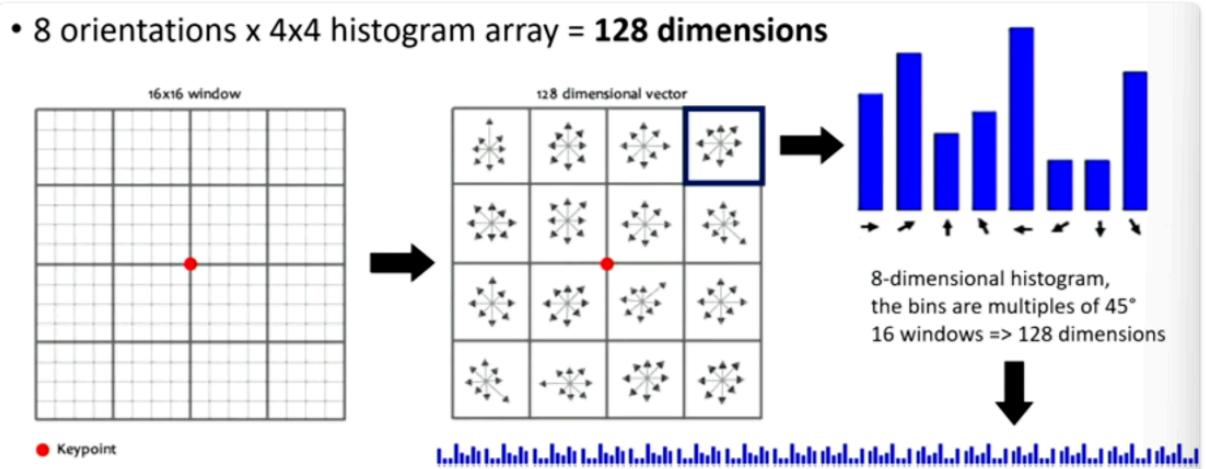
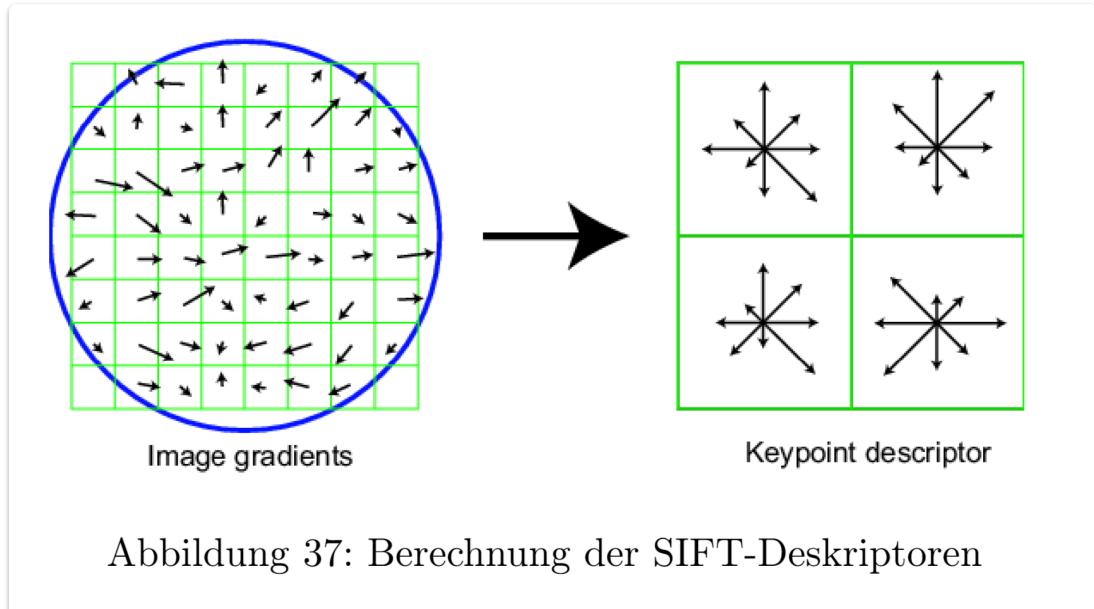
3. Gewichtung der Gradienten:

- Der Wert, der einem Bin hinzugefügt wird, hängt von:
 - Der **Länge des Gradienten** (größere Gradienten haben mehr Einfluss).
 - Der **Entfernung zum Interest Point**:

- Entferntere Gradienten werden mit einer **Gaußschen Gewichtungsfunktion** weniger stark gewichtet (siehe blauer Kreis in der Abbildung).

4. Berechnung des Merkmalsvektors:

- Wiederhole den Prozess für alle **16 4x4 Subfenster**.
- Der resultierende Vektor hat $4 \times 4 \times 8 = 128$ Werte.
- Diese 128 Werte bilden den **Merkmalsvektor** für den Interest Point.



Einschränkungen und Anpassungen

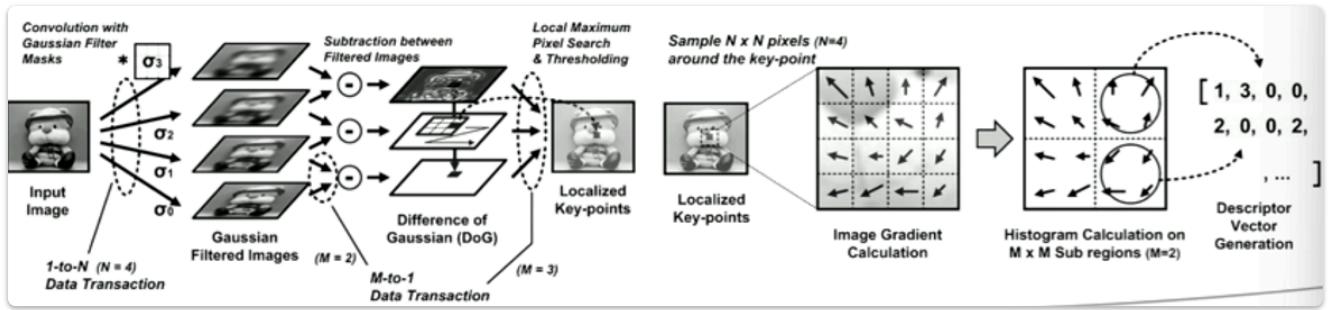
1. Rotationsabhängigkeit:

- Da der Merkmalsvektor auf **Gradientenrichtungen** basiert, verändern sich diese bei einer **Bildrotation**.
- Um **Rotationsinvarianz** zu erreichen, wird die dem Interest Point zugewiesene **dominante Orientierung** von jeder Gradientenrichtung subtrahiert. Somit wird jede Gradientenrichtung relativ zur Orientierung des Interest Points angegeben.

2. Beleuchtungsabhängigkeit:

- Lichteffekte können die **Länge einzelner Gradienten** beeinflussen.
- Um die Sensitivität gegenüber Beleuchtung zu verringern:

- Alle Werte im Merkmalsvektor, die größer als **0.2** sind, werden auf **0.2** gesetzt.
- Der Merkmalsvektor wird anschließend normalisiert, wodurch er weniger anfällig für Beleuchtungsvariationen wird.

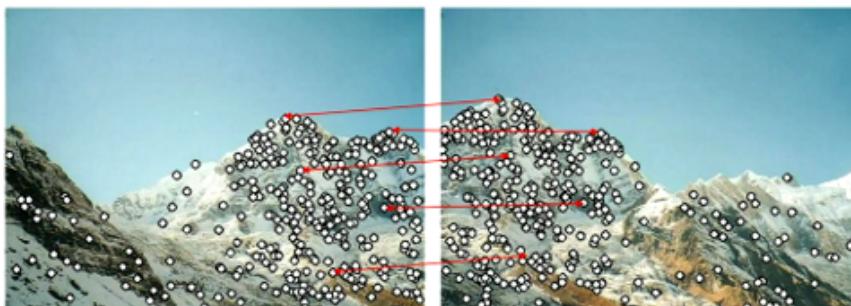


Möglicher Anwendungsfall: Panoramaerstellung

- We need to match (align) images
- Global methods are sensitive to occlusion, lighting, and parallax effects. So, look for local features that match well.
- How would you do it by eye?



- Detect feature points in both images
- Find corresponding pairs



- Use these pairs to align images



Testähnliches Beispiel

Beispiel: Moravec-Eckendetektor



- Gegeben ist ein 5x5 großer Bildausschnitt, auf den der Moravec-Eckendetektor angewendet werden soll. Berechnen Sie die Veränderungen der Intensitäten E für die mit Stern * markierte Stelle (3,3) und die 4 Verschiebungen (1,0), (1,1), (0,1) und (-1,1). Verwenden Sie dazu eine Fenstergröße von 3x3 und die Summe der quadrierten Differenzen (SSD, Sum of Squared Differences). Bestimmen Sie des weiteren aus den Veränderungen der Intensitäten den „Interest Value“.

y	1	70	60	70	60	60
1	70	80	80	90	80	80
2	80	80	90	80	80	80
3	80	90	100*	100	100	100
4	80	90	100	100	100	100
5	70	80	100	100	100	110
x	1	2	3	4	5	

$$E(u, v) = \sum_{x,y} w(x, y) [I(x+u, y+v) - I(x, y)]^2$$

E(1,0): SSD der Pixelwerte von $\boxed{80}$ und $\boxed{90}$:

$E(1,0) = \dots$

$E(1,1) = \dots$

$E(0,1) = \dots$

$E(-1,1) = \dots$

Interest value: \dots

Beispiel: Moravec-Eckendetektor



- Gegeben ist ein 5x5 großer Bildausschnitt, auf den der Moravec-Eckendetektor angewendet werden soll. Berechnen Sie die Veränderungen der Intensitäten E für die mit Stern * markierte Stelle (3,3) und die 4 Verschiebungen (1,0), (1,1), (0,1) und (-1,1). Verwenden Sie dazu eine Fenstergröße von 3x3 und die Summe der quadrierten Differenzen (SSD, Sum of Squared Differences). Bestimmen Sie des weiteren aus den Veränderungen der Intensitäten den „Interest Value“.

y	1	70	60	70	60	60
1	70	80	80	90	80	80
2	80	80	90	80	80	80
3	80	90	100*	100	100	100
4	80	90	100	100	100	100
5	70	80	100	100	100	110
x	1	2	3	4	5	

$$E(u, v) = \sum_{x,y} w(x, y) [I(x+u, y+v) - I(x, y)]^2$$

E(1,0): SSD der Pixelwerte von $\boxed{80}$ und $\boxed{90}$:

$$(90-80)^2 + (80-90)^2 + (80-80)^2 +$$

$$(100-90)^2 + (100-100)^2 + (100-100)^2 +$$

$$(100-90)^2 + (100-100)^2 + (100-100)^2 = 400$$

Beispiel: Moravec-Eckendetektor



- Gegeben ist ein 5x5 großer Bildausschnitt, auf den der Moravec-Eckendetektor angewendet werden soll. Berechnen Sie die Veränderungen der Intensitäten E für die mit Stern * markierte Stelle (3,3) und die 4 Verschiebungen (1,0), (1,1), (0,1) und (-1,1). Verwenden Sie dazu eine Fenstergröße von 3x3 und die Summe der quadrierten Differenzen (SSD, Sum of Squared Differences). Bestimmen Sie des weiteren aus den Veränderungen der Intensitäten den „Interest Value“.

y	1	70	60	70	60	60	$E(1,0)=$ <u>400</u>
1	70	60	70	60	60		
2	80	80	90	80	80		$E(1,1)=$ <u> </u>
3	80	90	100*	100	100		$E(0,1)=$ <u> </u>
4	80	90	100	100	100		$E(-1,1)=$ <u> </u>
5	70	80	100	100	110		Interest value: <u> </u>
x	1	2	3	4	5		

$$E(u, v) = \sum_{x,y} w(x, y) [I(x+u, y+v) - I(x, y)]^2$$

$E(1,1)$: SSD der Pixelwerte von und :

Beispiel: Moravec-Eckendetektor



- Gegeben ist ein 5x5 großer Bildausschnitt, auf den der Moravec-Eckendetektor angewendet werden soll. Berechnen Sie die Veränderungen der Intensitäten E für die mit Stern * markierte Stelle (3,3) und die 4 Verschiebungen (1,0), (1,1), (0,1) und (-1,1). Verwenden Sie dazu eine Fenstergröße von 3x3 und die Summe der quadrierten Differenzen (SSD, Sum of Squared Differences). Bestimmen Sie des weiteren aus den Veränderungen der Intensitäten den „Interest Value“.

y	1	70	60	70	60	60	$E(1,0)=$ <u>400</u>
1	70	60	70	60	60		
2	80	80	90	80	80		$E(1,1)=$ <u>1.200</u>
3	80	90	100*	100	100		$E(0,1)=$ <u>700</u>
4	80	90	100	100	100		$E(-1,1)=$ <u>1.400</u>
5	70	80	100	100	110		Interest value: <u> </u>
x	1	2	3	4	5		

$$E(u, v) = \sum_{x,y} w(x, y) [I(x+u, y+v) - I(x, y)]^2$$

Interest Value: Minimum der 4 Werte

