03-MB-709.03

Echtzeitbildverarbeitung (9)

Prof. Dr. Udo Frese

Zusammenfassung 2D-Bildverarbeitung

Auffrischung Matrizenrechnung

Homogene Koordinaten

Einladung

3 Jahre Echtzeitbildverarbeitung
16.6.2011, 16:00 s.t.(!) - 18:00
Universität Bremen, Cartesium, Rotunde

http://www.informatik.uni-bremen.de/agebv/de/3JahreEBV

16:00 Hilfe für die Helfer - 3D Echtzeitbildverarbeitung zur Suche nach Verschütteten (René Wagner)

16:15 Sichere Algorithmen zur Sensordatenverarbeitung (Holger Täubig)

16:30 Sportrobotik - Das Runde trifft das Eckige (Tim Laue)

16:45 Wie geht es weiter? (Udo Frese)

17:00 Kaffee, Kuchen und Demos

- Physikalisch Interaktives verGnügunGsrobotersYstem (PIGGY)
- Kollisionsvermeidung für fahrerlose Transportsysteme
- 3D Echtzeitbildverarbeitung im nachgestellten Trümmerhaufen
- Roboterfußball mit B-Human
- Humanoider Roboter fängt zwei Bälle gleichzeitig (Video)
- Kollisionsvermeidung für humanoide Roboter (Video)

Was bisher geschah

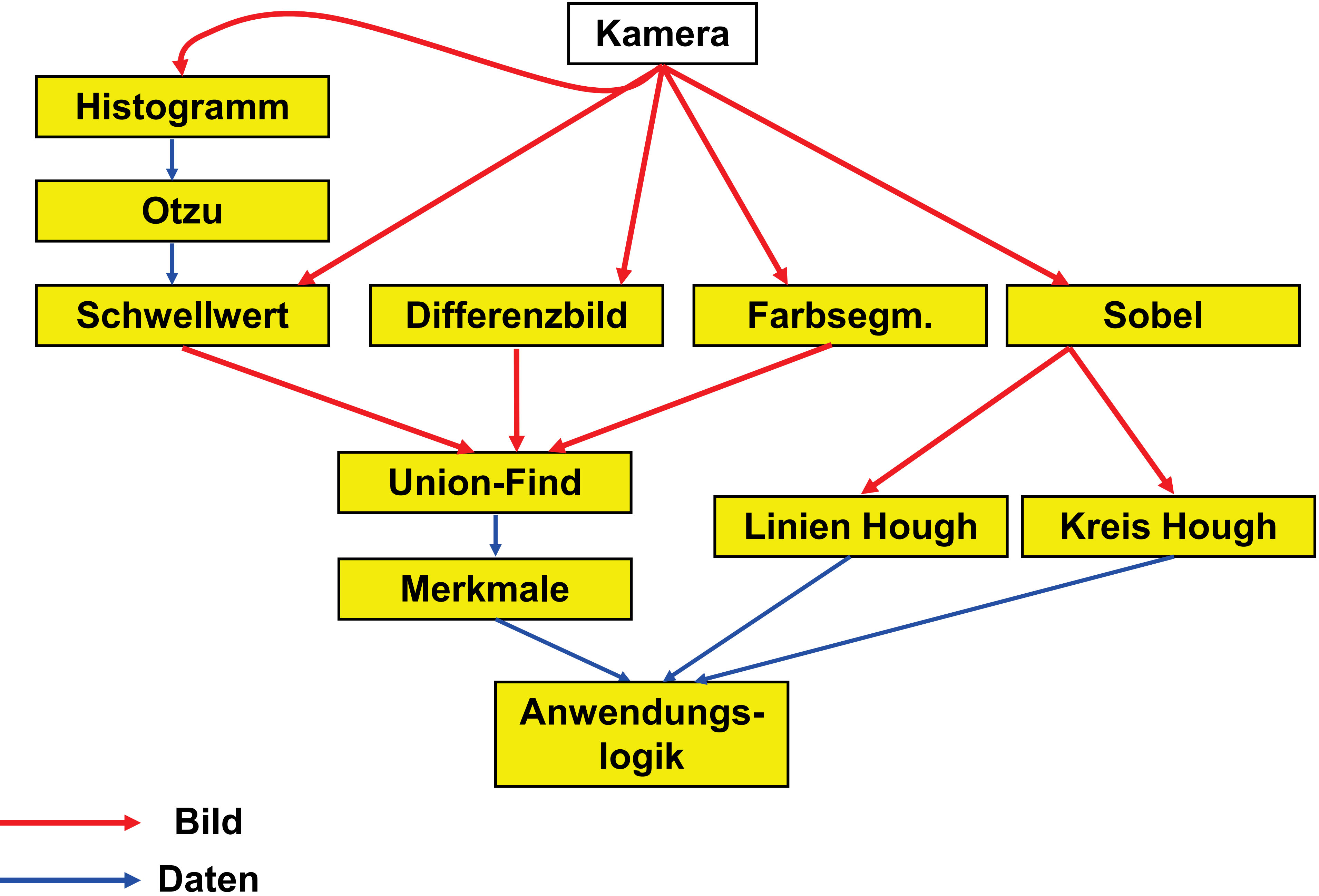
- **Linien Hough Transformation:**
  - Nur Winkel ungefähr in Sobelrichtung im Houghraum erhöhen.
  - α in 256 Schritte $[0..\pi)$ diskretisieren, Periodizität beachten.
  - *d* in Bezug auf Bildmitte um Houghraum klein zu halten.
  - Look-up-table (LUT 1) für Länge / Richtung aus SobelX, SobelY
  - Look-up-table (LUT 2) für sin/cos in Festpunktarithmetik
  - Konstanten herausziehen, >>, <<, & nutzen
- **Derart technisch verwickelte Optimierung sind nur sinnvoll für Teilroutinen die sehr oft ausgeführt werden (z.B. jeden Pixel)**

Tipps für Hough-Transformation

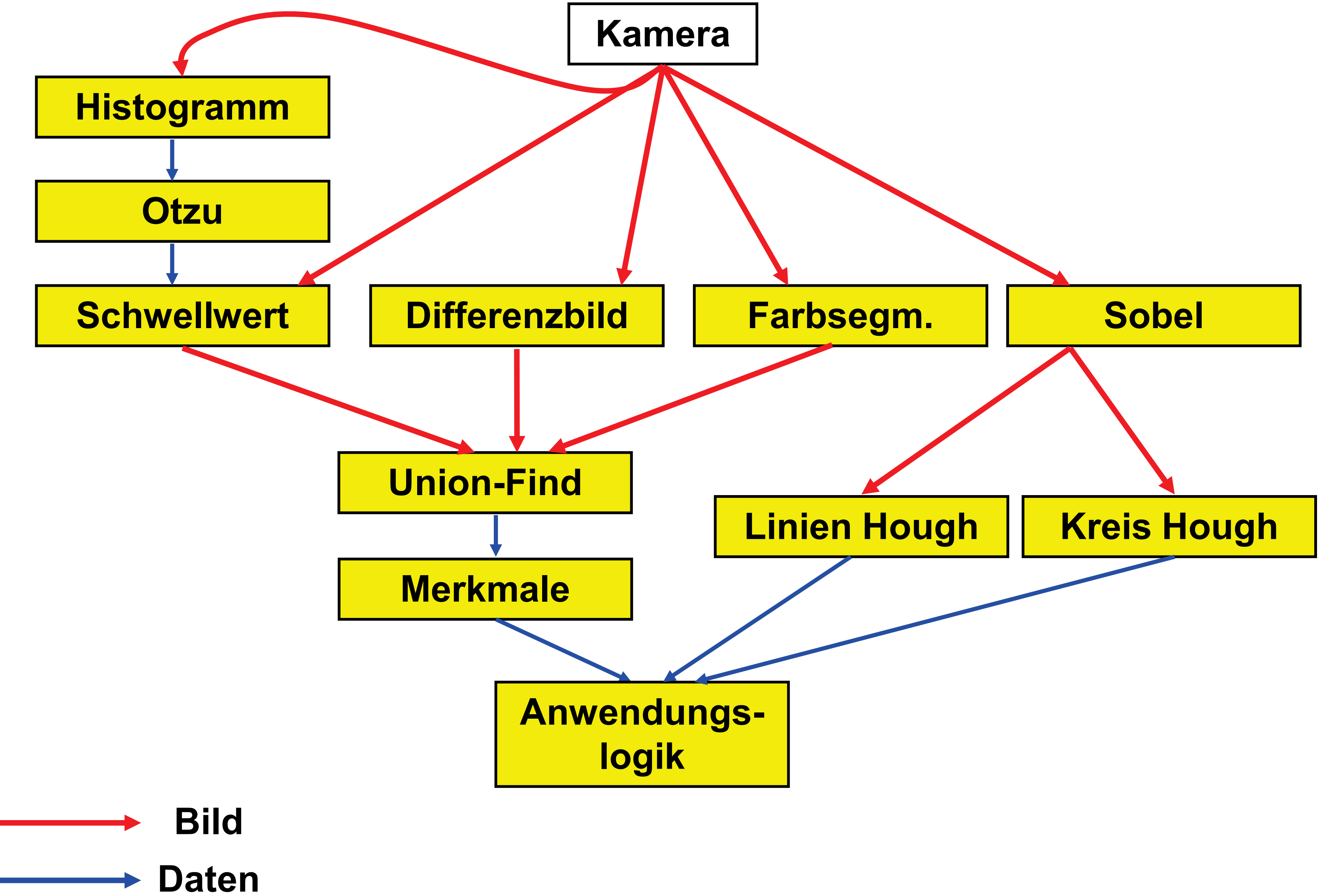
Implementierungstricks

- Von einfach zu schwer schrittweise
- Einfache Testbilder bei denen man beurteilen kann was herauskommt
- Genau analysieren
- assertions
- Debugger benutzen (z.B. DDD, kdevelop)
- Speicherschützer verwenden (z.B. libefence, valgrind, kcachegrind)

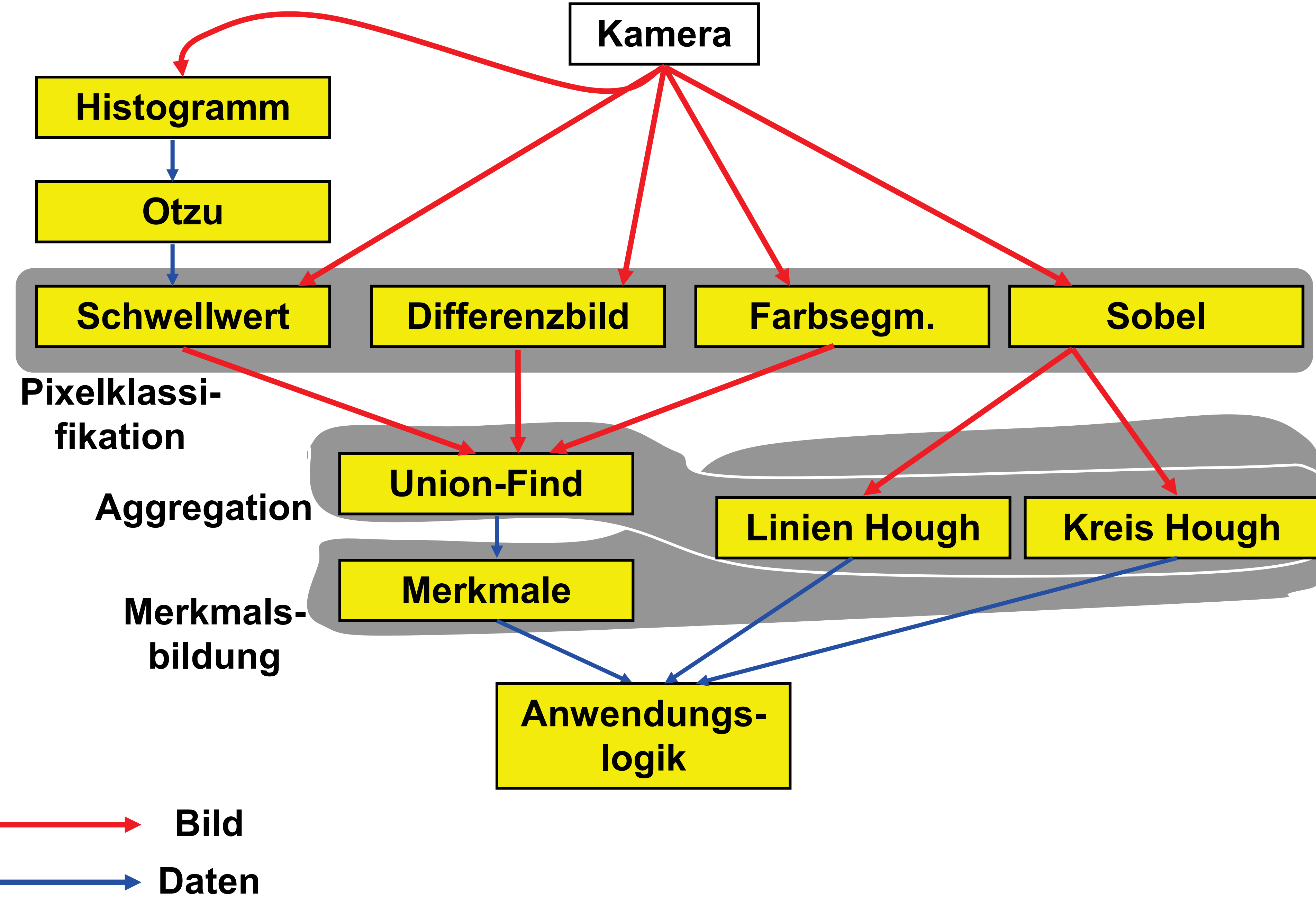
Rekapitulation 2D-Bildverarbeitung


Kamera
Histogramm
Otzu
Schwellwert
Differenzbild
Farbsegm.
Sobel
Union-Find
Linien Hough
Kreis Hough
Merkmale
Anwendungs-
logik
Bild
Daten

Frage an das Auditorium: Gibt es ein Schema oder ein Prinzip?



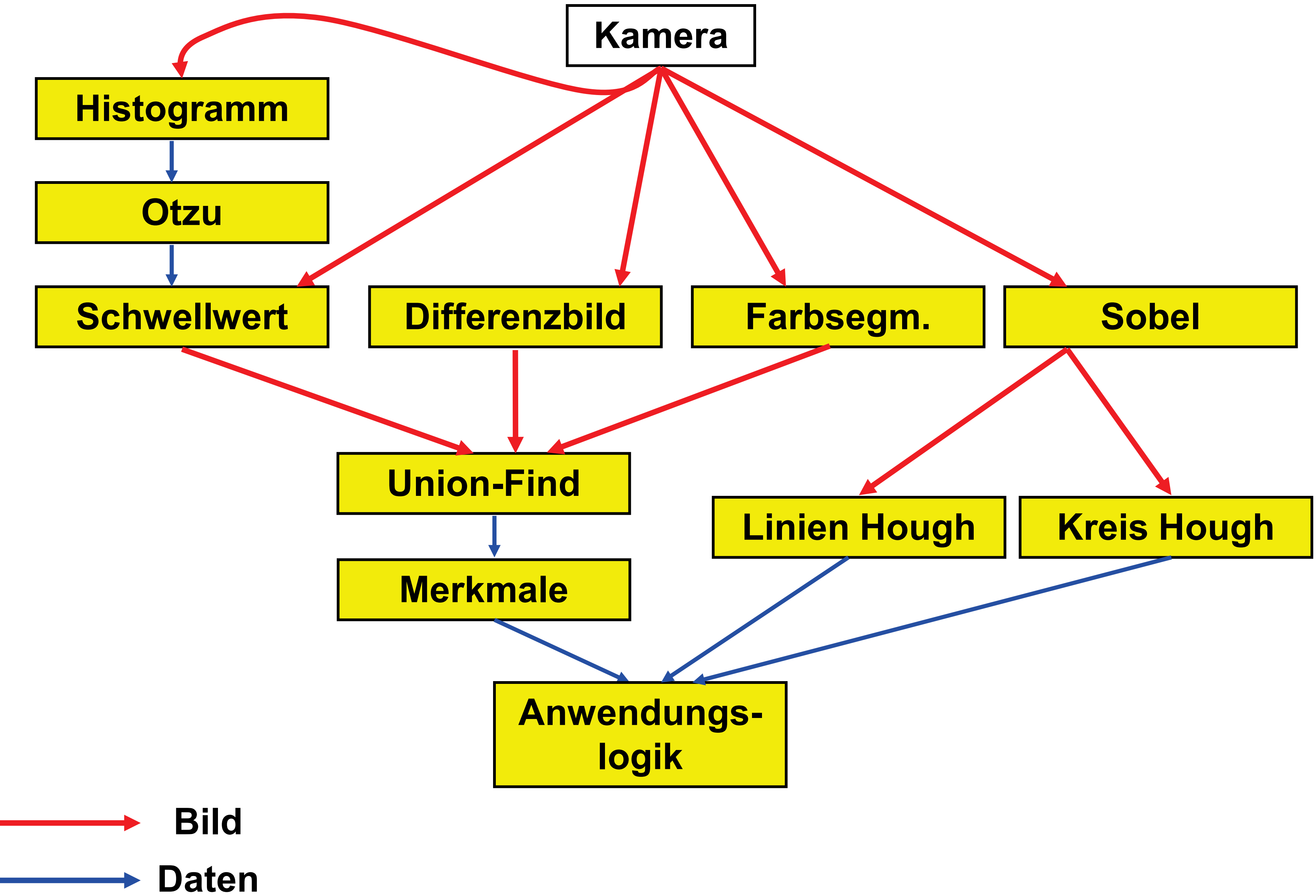
<u>Frage an das Auditorium:</u> Gibt es ein Schema oder ein Prinzip?



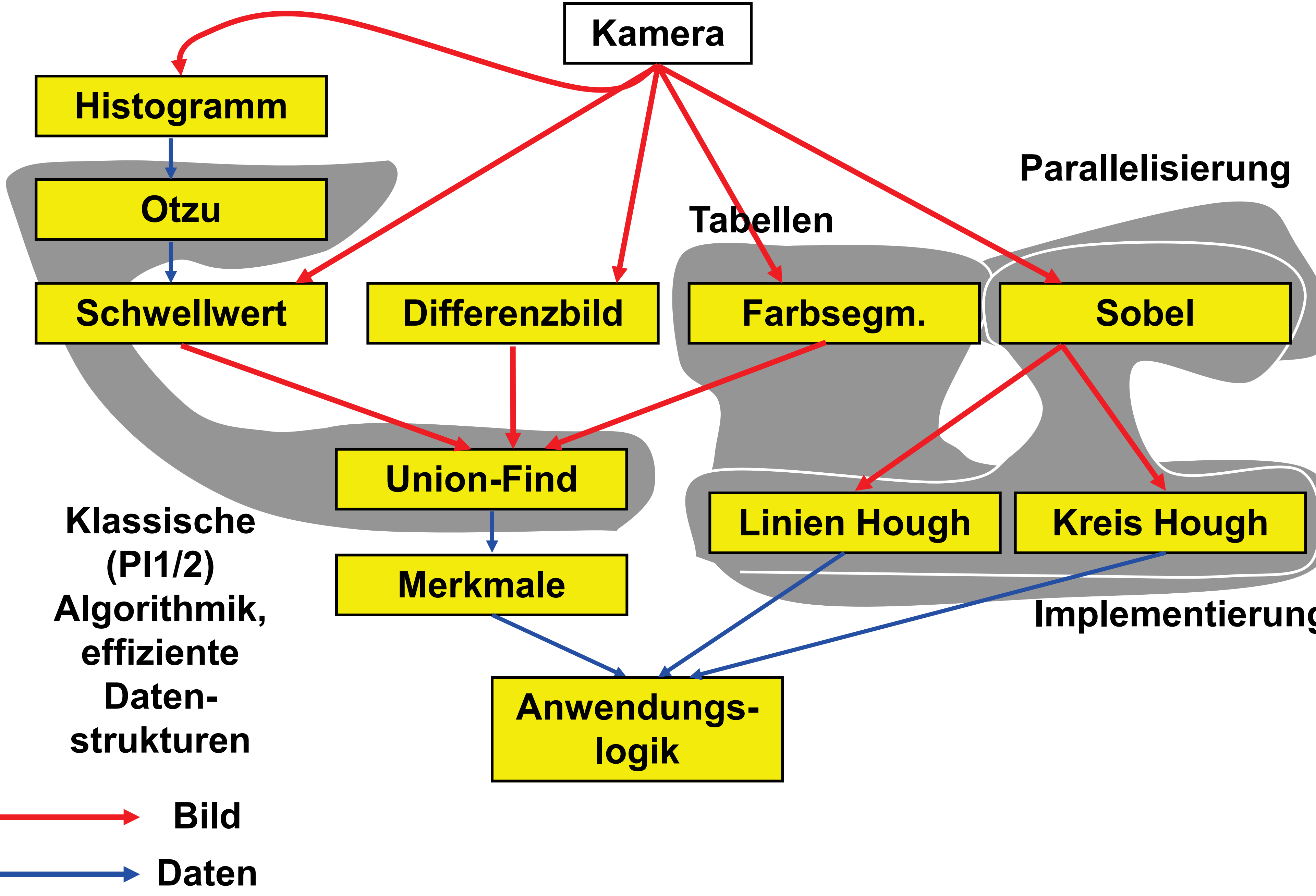
Rekapitulation 2D-Bildverarbeitung

- **Schema „Datengetriebene Bildverarbeitung“**
  - „Früh die Datenmenge reduzieren“
  - Pixelklassifikation (Schwellwert, Farbsegm., Sobel): Finden der Pixel, die lokal so wie gesucht aussehen.
  - Aggregation (Union-Find, Hough): Zusammenfassen von Pixeln, die zum selben Objekt gehören.
  - Merkmalsbildung (Hauptträgheitsachsen, impl. Hough): Herunter brechen auf endlich viele Parameter.
  - Nachteil: Wissen auf höheren Stufen könnte niedrigere Stufen stützen, passiert aber nicht.
- **Gegenschema „Optimierung einer Wahrscheinlichkeitsfunktion“**
  - Alle Stufen bilden gemeinsame zu optimierende Wahrscheinlichkeitsfunktion
  - Entscheidungen auf unterer Stufe fallen anders aus, wenn es dadurch oben besser passt.
  - Viel robuster – viel langsamer, dazu später mehr

Frage an das Auditorium: Techniken zur Beschleunigung



Frage an das Auditorium: Techniken zur Beschleunigung



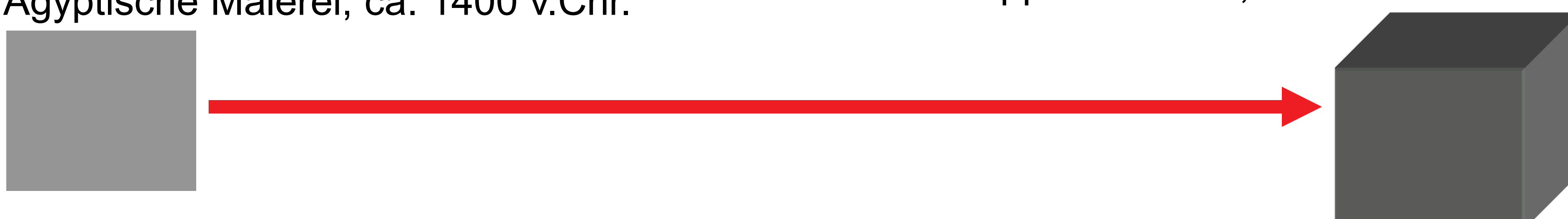
Techniken zur Beschleunigung

- **Vorberechnete Tabellen**
  - Aufwändige Teilrechnung, die nur von wenigen (1-3) Eingaben abhängt
  - Hilfsaufgaben integrieren (Diskretisierung, Normalisierung, Clipping, Umrechnen, Festkommaarithmetik)
  - Oft extrem viel schneller
- **Klassische Algorithmik**
  - Laufsumme, Wurzelbäume, Binärbäume, Divide & Conquer, …
  - Endlose Zahl an Techniken
- **Implementierungstricks**
  - Zeiger statt Indizes
  - Randtest außerhalb der Schleife
  - Teilausdrücke aus Schleife herausziehen
- **openMP Parallelisierung**
  - (Blöcke von) Zeilen mit openMP auf Kerne verteilen
  - Rechnung der Zeilen muss unabhängig sein (meist der Fall)
- **SIMD**
  - parallel auf 4/8/16 Pixeln rechnen
  - festes Zugriffsschema, ohne Bedingungen

Was nun geschieht

Ägyptische Malerei, ca. 1400 v.Chr.

Christus händigt Petrus die Schlüssel aus, Pietro Perugino (1481-82) Freske, 335 x 550 cm Cappella Sistina, Vatikan



Was nun geschieht

- **2D Bildverarbeitung**
  - Kein praktikables mathematisches Modell, wie ein Objekt im Bild erscheint.
  - Deshalb nicht: Erkennung durch Gleichung aus dem Modell.
  - Deshalb: Detektion von Merkmalen (Kontur, Linien, Kreise, Punkte).
- **3D Bildverarbeitung**
  - räumlichen Lage von Objekten anhand ihrer 2D Bildmerkmale.
  - Z.B: Kamerapose (Position & Orientierung) aus dem Bild eines Schachbrettgitters.
  - Perfektes mathematisches Modell wo ein Punkt mit bekannter Raumposition im Bild erscheint
  - Die perspektivische Abbildung.
  - Deshalb: 3D Pose als Gleichung aus der perspektivischen Abbildung.

Auffrischung Matrizenrechnung

<u>Vektor $v \in R^n$</u>

- **Zusammenstellung von *n* Zahlen**
- **Komponenten v_i mit Subskript Index**
- **Addition, Subtraktion komponentenweise.**
- **Multiplikation mit Skalar kompontenweise**
- **Skalarprodukt $v \cdot w$,**
  - definiert als $\Sigma_i v_i w_i$
  - Matrixschreibweise: $v^T w$
  - Euklidische Norm, Länge: Wurzel aus $|v| = \sqrt{v^T v}$.
  - geometrisch: $v^T w = |v|\ |w|\ \cos(\angle_v^w)$

Auffrischung Matrizenrechnung

Matrix $A \in R^{m \times n}$

- rechteckige Zusammenstellung von $m \times n$ Zahlen.
- Komponenten A_{ij} mit Subskript Index (Zeile *i*, Spalte *j*)
- $A_{i\bullet}$ ist i-te Zeile, $A_{\bullet j}$ ist j-te Spalte.
- Vektoren sind $n \times 1$ Matrizen.
- Addition, Subtraktion komponentenweise
- Multiplikation mit Skalar komponentenweise.
- Matrix mal Vektor ergibt Vektor: $(Av)_i = \Sigma_k A_{ik} v_k$

Auffrischung Matrizenrechnung

Matrix-Vektor Multiplikation

$$\begin{pmatrix}1 & 2 & 3\end{pmatrix}$$

$$\begin{pmatrix}12\\5\\10\end{pmatrix} = \begin{pmatrix}1 & 1 & 3\\2 & 0 & 1\\3 & 2 & 1\end{pmatrix} \bullet \begin{pmatrix}1\\2\\3\end{pmatrix}$$

$$(Av)_i = \sum_k A_{ik} v_k$$

Auffrischung Matrizenrechnung

- *i*-te Spalte ist der Funktionswert des *i*-ten Einheitsvektors.
- $A_{\bullet i} = Ae_i$
- Wichtig für intuitives Verstehen von Matrizen.

Matrix-Vektor Multiplikation

$$\begin{pmatrix} 1 & 0 & 0 \end{pmatrix}$$

$$\begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 3 \\ 2 & 0 & 1 \\ 3 & 2 & 1 \end{pmatrix} \bullet \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$$

$$(Av)_i = \sum_k A_{ik} v_k$$

Auffrischung Matrizenrechnung

Matrix $A \in R^{m \times n}$

- **Matrix mal Matrix ergibt Matrix: $(AB)_{ij} = \Sigma_k A_{ik} B_{kj}$**
  - Nicht kommutativ ($AB \neq BA$, in der Regel)
  - Assoziativ: ABC = A(BC) = (AB)C
- **0-Matrix 0:**
  - Einträge 0
  - 0v=0, 0A=0
- **Einheitsmatrix:**
  - I nur 0 und 1 auf Diagonale,
  - Iv=v, IA=A, AI=A
- **Transponierte A^T:**
  - Zeilen und Spalten vertauscht: $(A^T)_{ij} = A_{ji}$
- **Inverse: A^{-1}:**
  - $AA^{-1} = A^{-1}A = I$
  - Nicht alle Matrizen haben Inverse

Auffrischung Matrizenrechnung

Lineare Abbildung $f: R^n \rightarrow R^m$

- Addition: $f(v+w) = f(v) + f(w)$
- Multiplikation mit Skalar: $f(\lambda v) = \lambda f(v)$.
- Jede lineare Abbildung entspricht einer Matrix: $f(v) = Av$.
- Spalten von A sind Funktionswerte der Einheitsvektoren (Sehr wichtig)
- Verkettung von Abbildungen entspricht Matrixmultiplikation:
  $(f \circ g)(v) = f(g(v))$, $f(v)=Av$, $g(v)=Bv$,
  $f(g(v)) = A(Bv) = (AB)v$,
  also $f \circ g$ durch AB dargestellt.
- Reihenfolge von rechts nach links (Sehr wichtig):
  $ABv = A(Bv)$, also erst B auf v angewandt, dann A auf das Ergebnis.
- Umkehrabbildung entspricht Inversen.
  $f^{-1}(f(v))=v$, $A^{-1}Av=Iv=v$.

Auffrischung Matrizenrechnung

- **Frage an das Auditorium: Welche Matrix dreht einen 3D Vektor um α um die X-Achse?**

$$\begin{pmatrix} ? & ? & ? \\ ? & ? & ? \\ ? & ? & ? \end{pmatrix}$$

Matrix-Vektor Multiplikation

$$\begin{pmatrix} 1 & 2 & 3 \end{pmatrix}$$

$$\begin{pmatrix} 12 \\ 5 \\ 10 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 3 \\ 2 & 0 & 1 \\ 3 & 2 & 1 \end{pmatrix} \bullet \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}$$

$$(Av)_i = \sum_k A_{ik} v_k$$

Auffrischung Matrizenrechnung

- **Frage an das Auditorium: Welche Matrix dreht einen 3D Vektor um α um die X-Achse?**

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{pmatrix}$$

Matrix-Vektor Multiplikation

$$\begin{pmatrix} 1 & 2 & 3 \end{pmatrix}$$

$$\begin{pmatrix} 12 \\ 5 \\ 10 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 3 \\ 2 & 0 & 1 \\ 3 & 2 & 1 \end{pmatrix} \bullet \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}$$

$$(Av)_i = \sum_k A_{ik} v_k$$

Homogene Koordinaten

Homogene Koordinaten

- **Lineare Abbildungen (+, -, ·const),**
  - dargestellt durch Matrizen
  - bestverstandene Klasse mehrdimensionaler Abbildungen.
- **Projektive Abbildung**
  - auch Division aller Komponenten durch den selben Nenner.
  - z.B. Perspektive
- **Homogene Koordinaten:**
  - Erweitere Vektoren um eine Komponente
  - betrachte Vielfache als identisch.
  - „Normale“ Vektoren eingebettet durch anhängen einer 1 als letzte Komponente.

$$\begin{pmatrix}1\\2\\3\end{pmatrix} \rightarrow \begin{pmatrix}1\\2\\3\\1\end{pmatrix} \equiv \begin{pmatrix}2\\4\\6\\2\end{pmatrix} \equiv \begin{pmatrix}\frac{1}{2}\\1\\1\frac{1}{2}\\\frac{1}{2}\end{pmatrix}$$

$$\begin{pmatrix}0\\0\\1\end{pmatrix} \rightarrow \begin{pmatrix}0\\0\\1\\1\end{pmatrix} \equiv \begin{pmatrix}0\\0\\2\\2\end{pmatrix} \equiv \begin{pmatrix}0\\0\\\frac{1}{2}\\\frac{1}{2}\end{pmatrix}$$

Homogene Koordinaten

- **In homogenen Koordinaten sind projektive Abbildung linear, d.h. durch Matrizen darstellbar.**
- **Wertvolles Werkzeug für anspruchsvolle Bild->3D Berechnungen.**
- **Referenz: Hartley & Zisserman: Multiple View Geometry**
- **Hier: Beschränkung auf 1 oder 0 als letzte Komponente.**
  - Darstellung von Pose, Koordinatensystemen, Starrkörperbewegungen.
  - Eigentliche Perspektive (X/Z, Y/Z) wieder klassisch, nicht als Matrix.

Beispiel

$$f\begin{pmatrix} x \\ y \end{pmatrix} = \frac{x}{y}, \quad f_{\text{hom}}\begin{pmatrix} x \\ y \\ w \end{pmatrix} = \begin{pmatrix} x \\ y \end{pmatrix}, \quad \begin{pmatrix} x \\ y \end{pmatrix} \to \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}, \quad f_{\text{hom}}\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} x \\ y \end{pmatrix} \equiv \begin{pmatrix} x/y \\ 1 \end{pmatrix} \leftarrow \frac{x}{y}$$

Homogene Koordinaten

- **Eingeschränkte homogene Koordinaten**
  - Ortsvektor (mit 1)
  - Richtungsvektor (mit 0)
- **Konsistentes Rechnen**
  - Differenz zweier Ortsvektoren ist ein Richtungsvektor
  - Richtungsvektor können mit Skalaren multipliziert werden
  - Summe aus Orts- und Richtungsvektor ist Ortsvektor

$$\begin{pmatrix} x \\ y \\ z \\ 0 \end{pmatrix} \quad \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

$$\begin{pmatrix} * \\ * \\ * \\ 1 \end{pmatrix} - \begin{pmatrix} * \\ * \\ * \\ 1 \end{pmatrix} = \begin{pmatrix} * \\ * \\ * \\ 0 \end{pmatrix} \qquad \lambda \begin{pmatrix} * \\ * \\ * \\ 0 \end{pmatrix} = \begin{pmatrix} * \\ * \\ * \\ \lambda 0 \end{pmatrix} = \begin{pmatrix} * \\ * \\ * \\ 0 \end{pmatrix} \qquad \begin{pmatrix} * \\ * \\ * \\ 1 \end{pmatrix} + \begin{pmatrix} * \\ * \\ * \\ 0 \end{pmatrix} = \begin{pmatrix} * \\ * \\ * \\ 1 \end{pmatrix}$$

Homogene Koordinaten

Frage an das Auditorium: Was kann man zur Summe zweier Ortsvektoren sagen?

Homogene Koordinaten

<u>Frage an das Auditorium: Was kann man zur Summe zweier Ortsvektoren sagen?</u>

- **Kein ordentlicher Vektor (wegen der 2 als homogene Komponente)**
- **Denn das Ergebnis hängt vom Koordinatensystem ab.**
- **Verschiebt man Koordinatensystem 1 nach +X verschiebt, verringern sich X-Koordinaten um 1, die Summe also um 2.**
- **Kann aber als Zwischenergebnis auftauchen (z.B. (v+w)/2).**

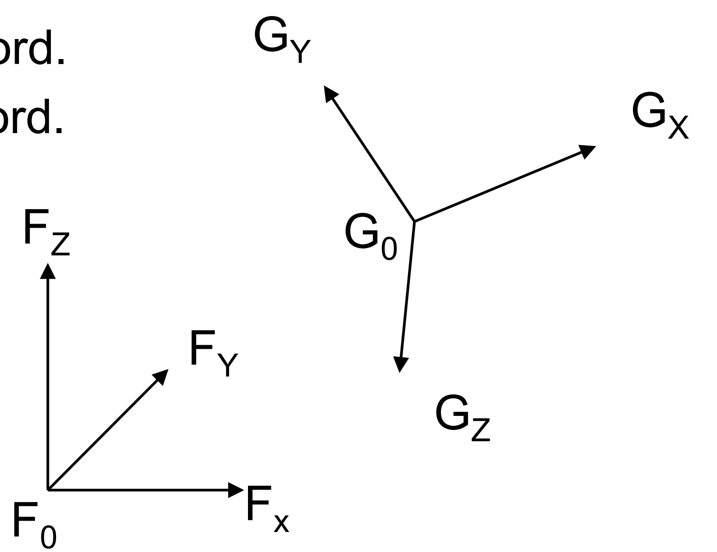
$$\begin{pmatrix} * \\ * \\ * \\ 1 \end{pmatrix} + \begin{pmatrix} * \\ * \\ * \\ 1 \end{pmatrix} = \begin{pmatrix} * \\ * \\ * \\ 2 \end{pmatrix}$$

Homogene Koordinaten

Starrkörperpose oder Koordinatentransformation

- **Transformation „GInF“ wandelt Orts- oder Richtungsvektor von G-Koordinaten in F-Koordinaten um.**
  - Spalte 1: Bild von $(1,0,0,0)^T$, also G_X in F-Koord.
  - Spalte 2: Bild von $(0,1,0,0)^T$, also G_Y in F-Koord.
  - Spalte 3: Bild von $(0,0,1,0)^T$, also G_Z in F-Koord.
  - Spalte 4: Bild von $(0,0,0,1)^T$, also G_0 in F-Koord.

$$\begin{pmatrix} x_F \\ y_F \\ z_F \\ w_F \end{pmatrix} = \begin{pmatrix} * & * & * & * \\ * & * & * & * \\ * & * & * & * \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x_G \\ y_G \\ z_G \\ w_G \end{pmatrix}, \vec{v}_F = T_F^G \vec{v}_G$$



Homogene Koordinaten

Starrkörperpose oder Koordinatentransformation

- 3*3 Untermatrix Q Rotation bzw. Orientierung
- 4. Spalte Translation t

$$\begin{pmatrix} x_F \\ y_F \\ z_F \\ w_F \end{pmatrix} = \left(\begin{array}{ccc|c} & & & t_x \\ & Q & & t_y \\ & & & t_z \\ \hline 0 & 0 & 0 & 1 \end{array}\right) \cdot \begin{pmatrix} x_G \\ y_G \\ z_G \\ w_G \end{pmatrix} = Q \begin{pmatrix} x_G \\ y_G \\ z_G \end{pmatrix} + \begin{pmatrix} t_x \\ t_y \\ t_z \end{pmatrix} w_G$$

- *Q* erhält Winkel und Längen (orthonormal).
- Spalten von *Q* haben Länge 1 und stehen senkrecht aufeinander.
- *Q*: 9 Zahlen, 6 Zwangsbedingungen, 3 Freiheitsgrade

$$v^T w = (Qv)^T (Qw) = v^T Q^T Q w \quad \Rightarrow \quad Q^T Q = I \quad \Rightarrow Q^{-1} = Q^T$$

$$Q_{\bullet 1}^T Q_{\bullet 1} = 1,\ Q_{\bullet 2}^T Q_{\bullet 2} = 1,\ Q_{\bullet 3}^T Q_{\bullet 3} = 1,\ Q_{\bullet 1}^T Q_{\bullet 2} = 0,\ Q_{\bullet 1}^T Q_{\bullet 3} = 0,\ Q_{\bullet 2}^T Q_{\bullet 3} = 0$$

Homogene Koordinaten

Inverse einer Starrkörpertransformation

- **Durch besondere Form der Matrix einfache Formel für Inverse:**

$(GInF)^{-1}$ **FInG**

$$\left(\begin{array}{ccc|c} & Q & & t \\ \hline 0 & 0 & 0 & 1 \end{array}\right)^{-1} = \left(\begin{array}{ccc|c} & Q^T & & -Q^T t \\ \hline 0 & 0 & 0 & 1 \end{array}\right)$$

transponieren, nicht rechnen, weil Q orthonormal.

t, also Vektor von F0 nach G0 von F in G Koordinaten umrechnen. Negieren, weil Vektor von F0 nach G0.

Homogene Koordinaten

Praktische Tipps für Programmieren in 3D

- **Formale Genauigkeit beim Benennen ist praktisch sehr hilfreich.**
- **Punkte mit Koordinatensystem benennen**
  - `pInA` ist Punkt p in A-Koordinaten
- **(Relativ-) Posen mit zwei Koordinatensystemen benennen**
  - `AInB` Pose von `A` in `B`-Koordinaten
  - bildet einen Vektor/Punkt in `A`-Koordinaten auf denselben Vektor/Punkt in `B`-Koordinaten ab.
  - `pInB = AInB*pInA`
  - Spalten von `AInB` sind Einheitsvektoren/Ursprung von `A` in `B`-Koordinaten
  - Absolutposen `AInWorld`, ggf. `InWorld` weglassen
- **`ObjectInWorld` not `WorldInObject`**

Homogene Koordinaten

Praktische Tipps für Programmieren in 3D

- **Konsistenz von Koordinatenrechnung an den Namen ablesbar**
- **Inverse vertauscht Rollen von A und B**
  - `AInB`$^{-1}$ = `BInA`.
- **Verkettung muß von rechts nach links gelesen Sinn machen, weil Vektoren von rechts multipliziert werden.**
  - Bsp.: `BInC*AInB = AInC`
  - Bsp.: `RobotInCamera = CameraInWorld.inv()*RobotInWorld`
  - Begründung:
    - *`(BInC*AInB)*pInA = BInC*AInB*pInA = BInC*pInB = pInC`*

Homogene Koordinaten

Koordinatensystemwahl bei physischen Geräten

- **Koordinatensysteme sinnvoll wählen (gerne Z nach oben).**
- **Koordinatensysteme physisch sichtbar wählen, ggf. anzeichnen.**
- **Kanten gegen die man messen kann**
- **Wahl der Koordinatensysteme in einer Skizze dokumentieren**

Homogene Koordinaten

<u>Frage an das Auditorium</u>: Die Lage des Tisches im Raum soll beschrieben werden. Welche Koordinatensysteme sollte man wählen? Wie lautet die Transformationsmatrix `TableInWorld`?

$$\begin{pmatrix} ? & ? & ? & ? \\ ? & ? & ? & ? \\ ? & ? & ? & ? \\ ? & ? & ? & ? \end{pmatrix}$$

Zusammenfassung

- **Rekapitulation 2D Bildverarbeitung**
  - Prinzip: Pixelklassifikation, Aggregation, Merkmalsbildung
  - Optimierungsmethoden: Klassische Algorithmik, Tabellen, Implementierungstricks
- **Homogene Koordinaten**
  - 4. Komponente, hier entweder 0 (freier Vektor) oder 1 (Ortsvektor).
  - Koordinatentransformationen als 4×4 Matrix.
  - Matrix immer als AInB benennen, dann ist Konsistenz ablesbar.