

03-05-H
-709.53

Echtzeitbildverarbeitung (2)

Prof. Dr. Udo Frese

Der Weg des Bildes in den Rechner
Industrieller Ansatz
Schwellwert
Regionenbildung

Was bisher geschah

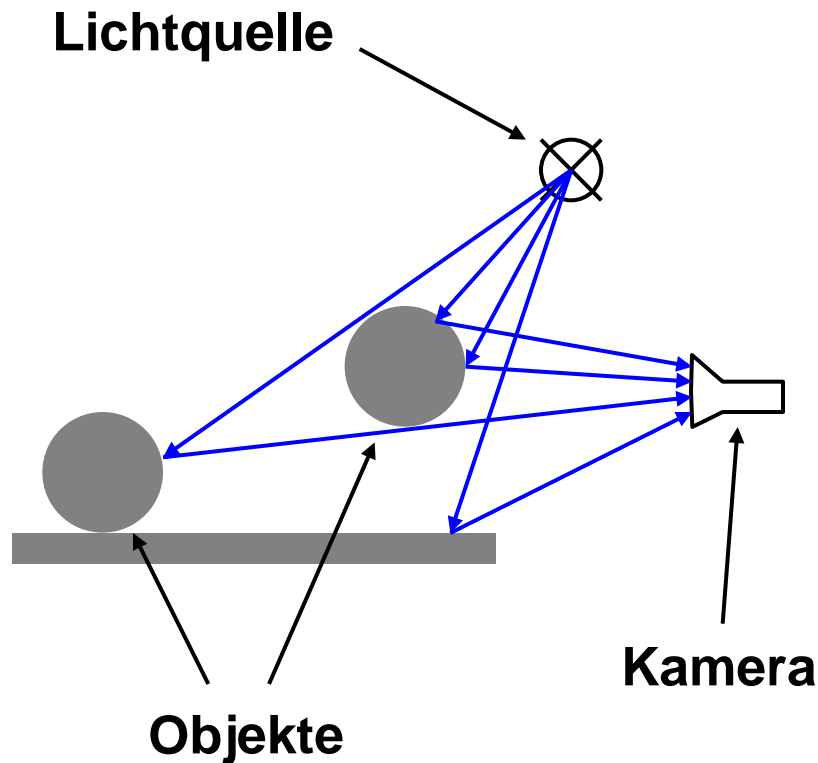
- ▶ **Bildverarbeitung ist automatisches Erkennen von Objekten in Bildern mit 2D / 3D Lagebestimmung**
 - ▶ „Umkehrproblem zur Computergrafik“ ist keine hilfreiche Sichtweise
 - ▶ Schwierig, weil viele „seltsame“ Effekte Bilder beeinflussen
- ▶ **Echtzeit bedeutet, so schnell wie der Vorgang der die Daten erzeugt**
 - ▶ Echtzeitbildverarbeitung 20ms – 200ms
 - ▶ Rechenzeit ist dominante Einschränkung
- ▶ **Industrielle Anwendungen**
 - ▶ Qualitätskontrolle: Maße, Vollständigkeit, Farbe, Oberfläche
 - ▶ Montage: Position von Werkstückteilen, Robotersteuerung
 - ▶ Die Lösung ist eine geschickt entworfene Umgebung
- ▶ **Forschungsanwendungen**
 - ▶ Vielfältig: Navigation, Kartierung, Sportrobotik, Medizin, Virtual Reality

Der Weg des Bildes in den Rechner

Wirkungskette bis zum Bild im Rechner (Normalfall, Monochrom)

- ▶ **Beleuchtung sendet Lichtstrahl aus ...**
- ▶ **... interagiert mit dem Objekt und wird zurückgestrahlt ...**
- ▶ **... trifft auf das Objektiv ...**
- ▶ **... wird vom Objektiv auf den CCD Chip abgebildet ...**
- ▶ **... erzeugt für jeden Pixel eine Spannung prop. zur Lichtmenge.**
- ▶ **Spannung wird digitalisiert und üblicherweise 8 Bit konvertiert ...**
- ▶ **... zum Rechner gesendet ...**
- ▶ **... in den Speicher geschrieben.**

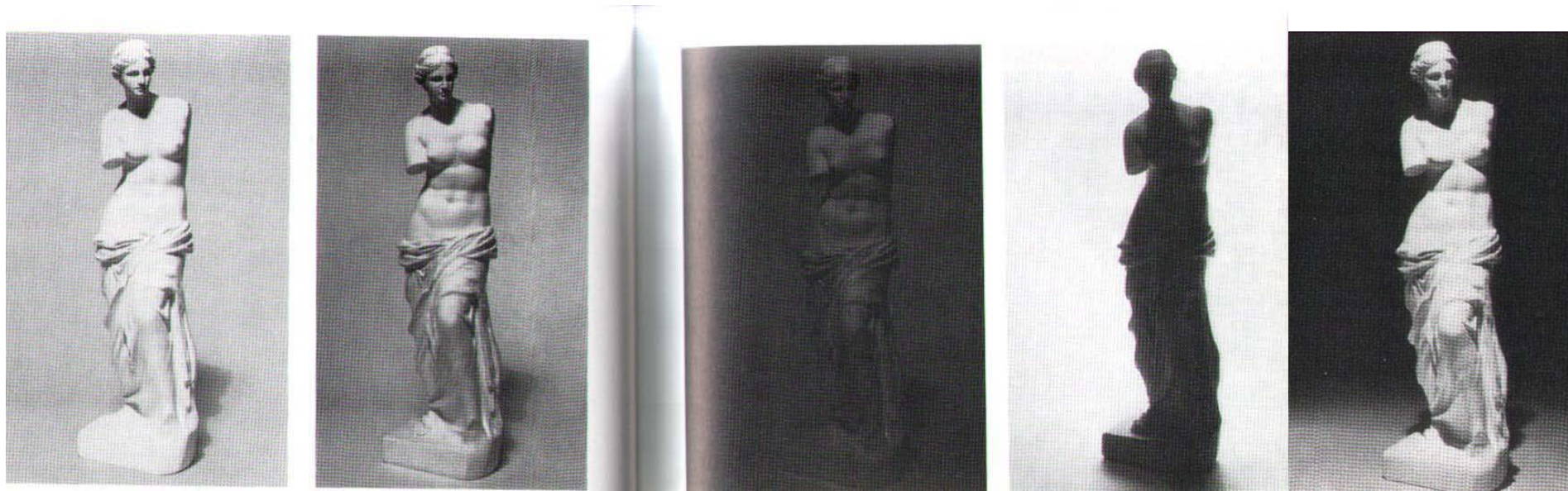
Der Weg des Lichtes



Der normale Weg:

- ▶ Licht wird von der Lichtquelle ausgesandt
- ▶ Trifft auf das Objekt und wird zurückgeworfen
- ▶ Stärke hängt von Oberfläche, Winkel zur Lichtquelle und t.w. Winkel zur Kamera ab
- ▶ Lichtstrahl trifft das Objektiv der Kamera

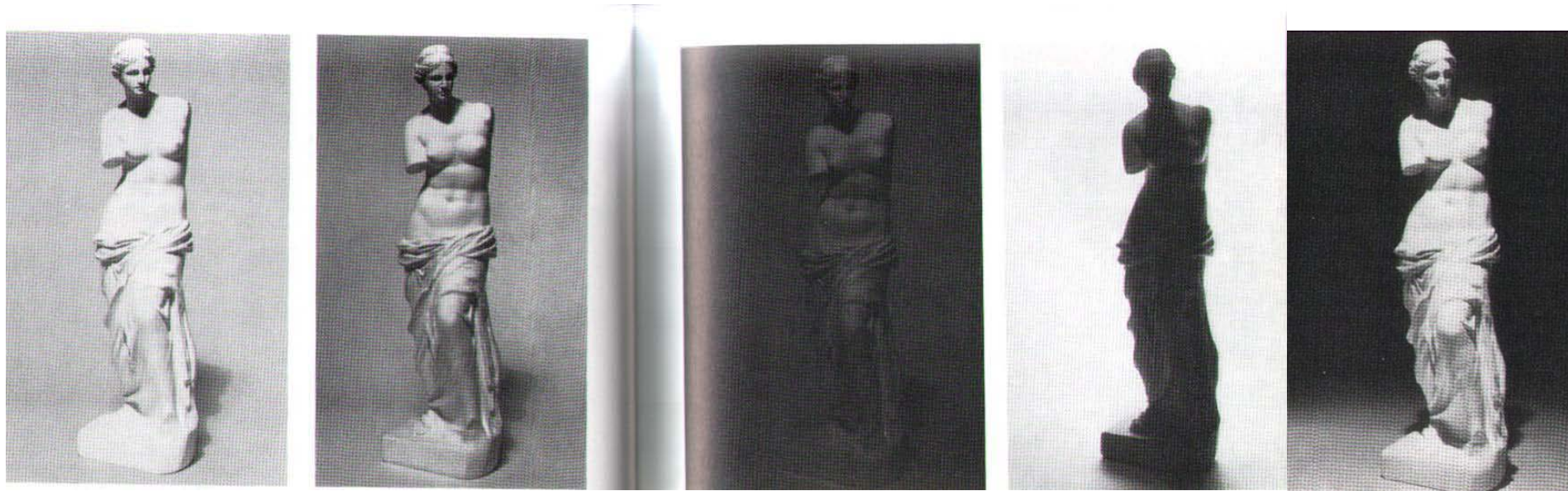
Der Weg des Lichtes



- ▶ Beleuchtung macht einen sehr großen Unterschied
- ▶ Besonders bei Industriellen Anwendungen:
Lieber bessere Beleuchtung als mehr Intelligenz!
- ▶ Frage an das Auditorium: Welche Beleuchtung eignet sich für wofür?

Quelle: A. Feininger: Die hohe Schule der Photographie, 1961

Der Weg des Lichtes



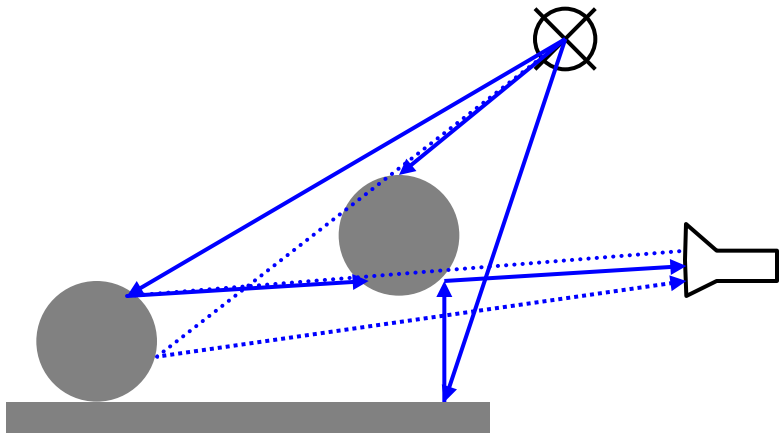
**Merkmale
auf der Figur**

**Kontur-
segmentierung**

- ▶ Frage an das Auditorium: Welche Beleuchtung eignet sich für wofür?

Quelle: A. Feininger: Die hohe Schule der Photographie, 1961

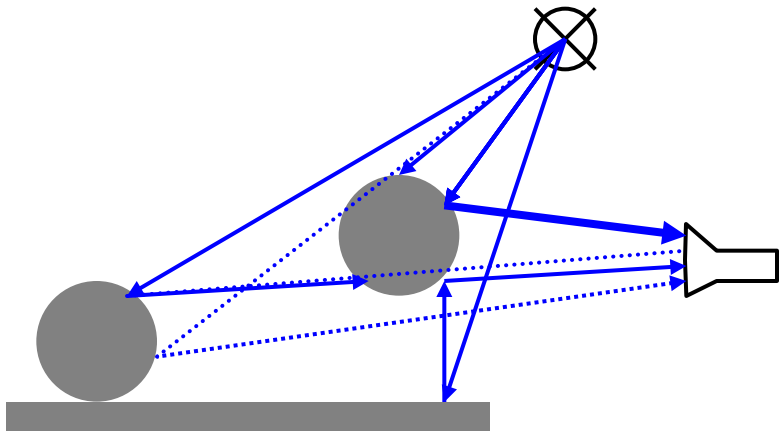
Der Weg des Lichtes



Effekte, die Probleme machen

- ▶ Frage an das Auditorium: Könnt Ihr die skizzierten Effekte benennen?

Der Weg des Lichtes

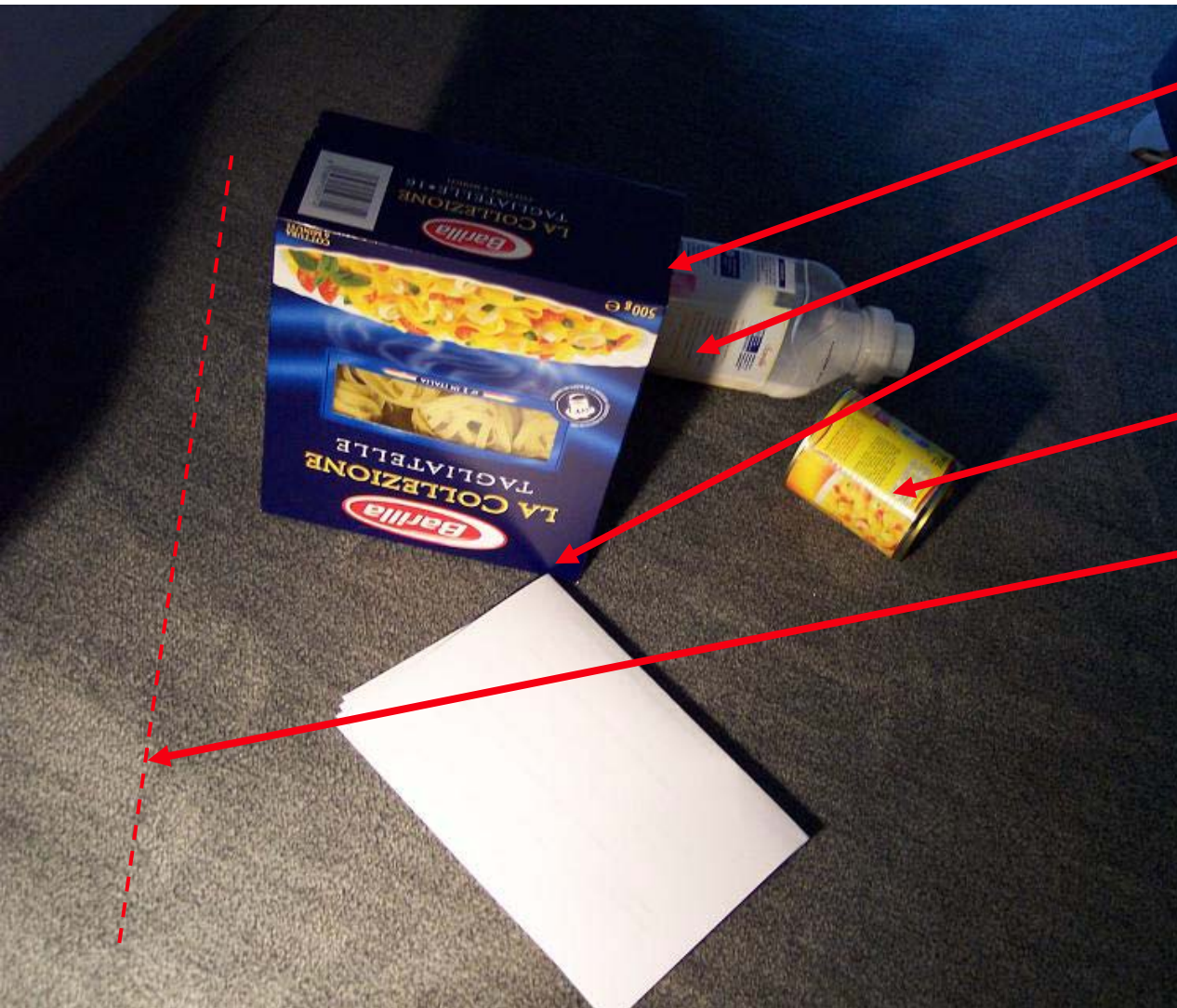


Effekte, die Probleme machen

(an der Lichtquelle v.l.n.r.):

- ▶ Verdeckung
- ▶ Schatten
- ▶ Reflexionen
- ▶ Licht von einem Objekt (meist gespiegelt)
- ▶ Beleuchtungsgradient (Licht einer Punktquelle ist $\sim r^{-2}$)

Der Weg des Lichtes



- ▶ Verdeckung
- ▶ Schatten
- ▶ Licht von einem Objekt (meist gespiegelt)
- ▶ Reflexionen
- ▶ Beleuchtungsgradient

Der Weg des Lichtes

Frage an das Auditorium: Was tut man gegen...

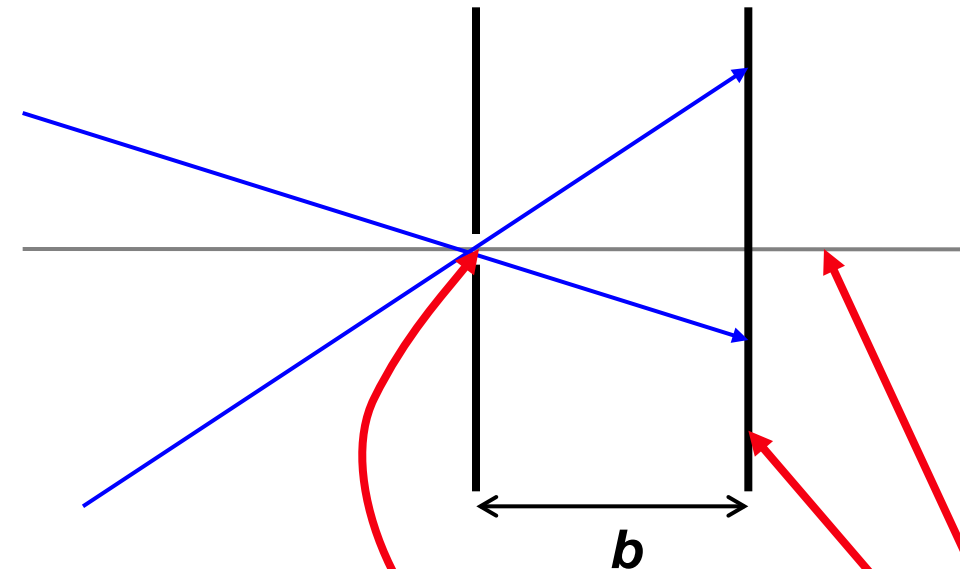
- ▶ Verdeckungen:
- ▶ Schatten
- ▶ Reflexionen
- ▶ Licht von einem Objekt
- ▶ Beleuchtungsgradient

Der Weg des Lichtes

Frage an das Auditorium: Was tut man gegen...

- ▶ **Verdeckungen:**
 - ▶ Bessere Kameraposition; Mehrere Kameras; Objekte besser plazieren
- ▶ **Schatten**
 - ▶ Licht nah an die Kamera
 - ▶ Ausgedehnte Lichtquellen (indirektes Licht, Röhren, Ringlicht, Flächenlicht, Lichtkuppel, LED Licht)
- ▶ **Reflexionen**
 - ▶ Matte Oberflächen; Licht von besserem Winkel;
 - ▶ ggf. Flächenlicht, Lichtkuppel; Polarisationsfilter
- ▶ **Licht von einem Objekt**
 - ▶ Matte Oberflächen
- ▶ **Beleuchtungsgradient**
 - ▶ Senkrecht beleuchten; Flächenlicht

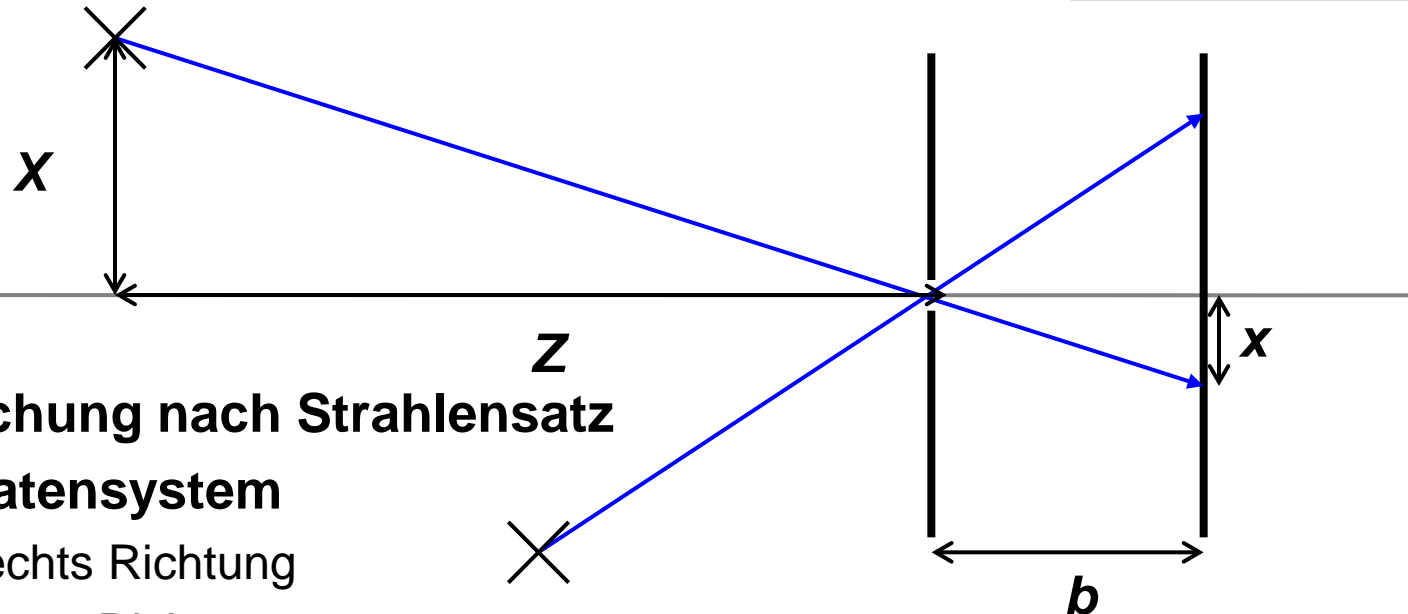
Der Weg des Lichtes



Lochkamera (Camera obscura)

- ▶ Lichtstrahlen fallen durch das Loch in der Vorderseite
- ▶ Auf der Rückseite entsteht das (umgedrehte) Bild
- ▶ definiert abstrakte perspektivische Abbildung
- ▶ Abstand ist die Bildweite b (ähnlich Brennweite f)
- ▶ Optische Achse
- ▶ Bildebene
- ▶ Optisches Zentrum der Kamera

Der Weg des Lichtes



▶ **Abbildungsgleichung nach Strahlensatz**

▶ **Kamerakoordinatensystem**

- ▶ X zeigt in Bild-rechts Richtung
- ▶ Y zeigt in Bild-unten Richtung
- ▶ Z zeigt in die Tiefe (optische Achse)

▶ **(X, Y, Z) Koordinaten des abgebildeten Punktes im Kamerasystem**

▶ **(x, y) Koordinaten des Abbildes im *physikalischen* Bild**

$$x = b \frac{X}{Z} \quad y = b \frac{Y}{Z}$$

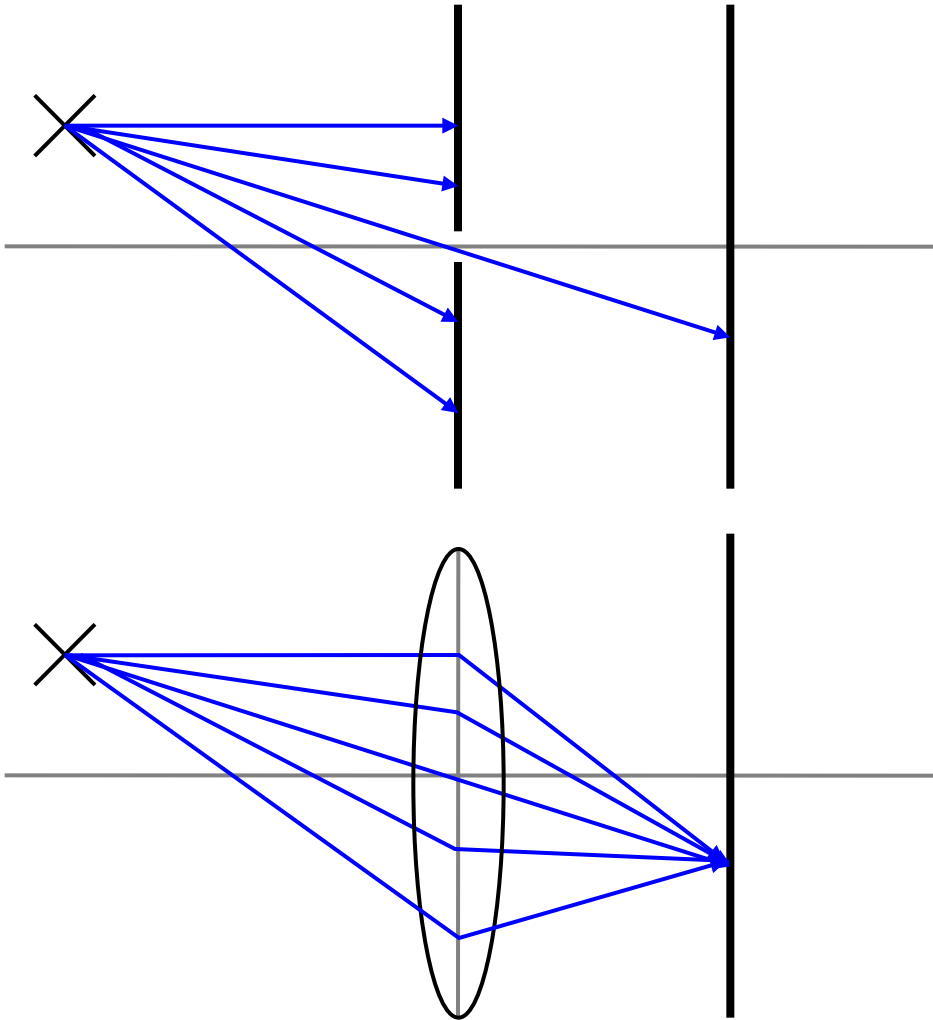
Der Weg des Lichtes

Bild-/ Brennweiten

- ▶ Durch verschiedene Bildweiten kann der Bildausschnitt / Vergrößerung verändert werden
- ▶ Praktisch: Objektive mit verschiedenen Brennweiten oder Zoomobjektiv
- ▶ Dieselbe Szene mit $\approx 80^\circ$, 45° , 15° Öffnungswinkel
- ▶ Mensch: $\approx 170^\circ$
- ▶ Verzerrungen bei über 45°



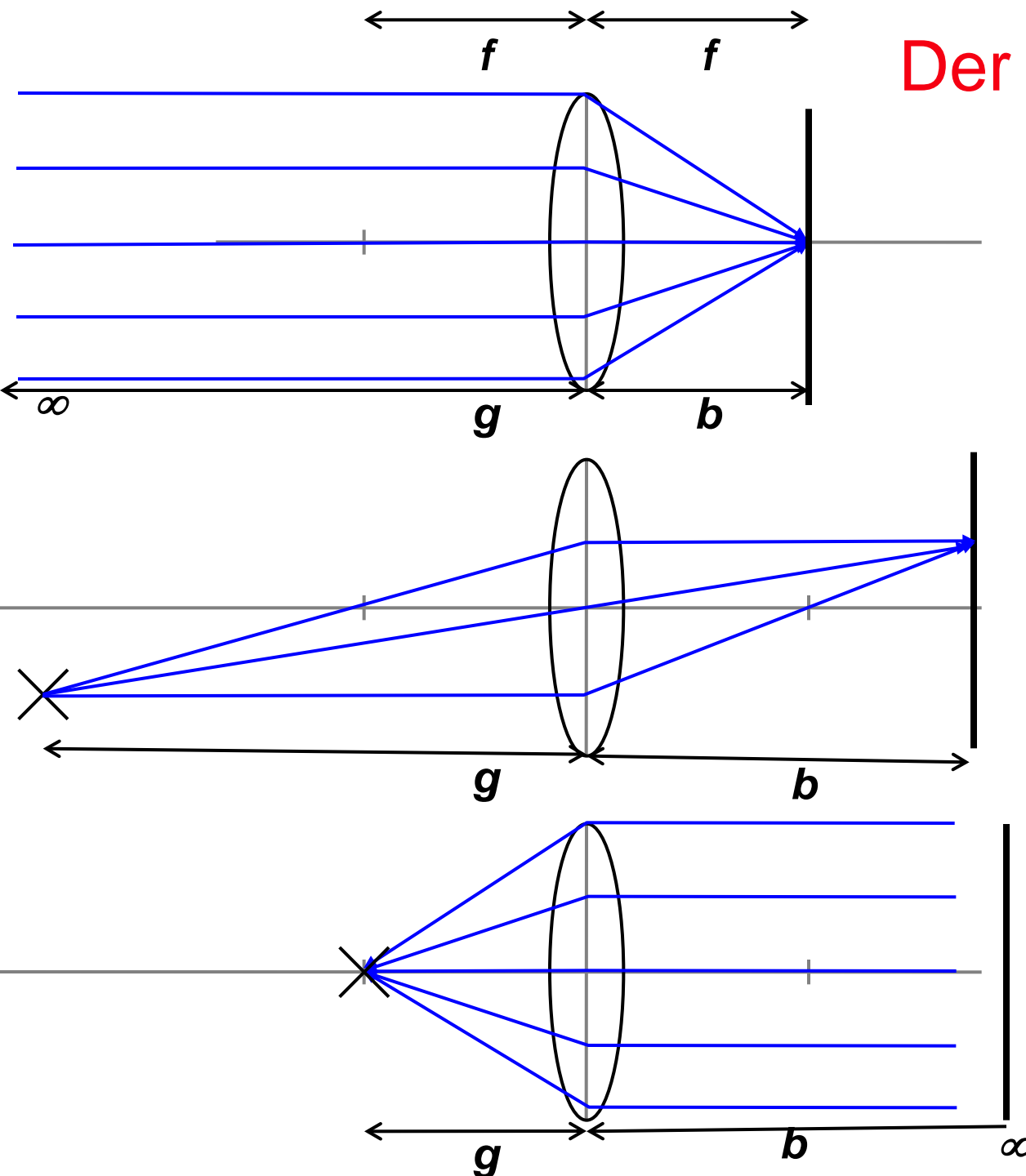
Der Weg des Lichtes



Linsenkamera

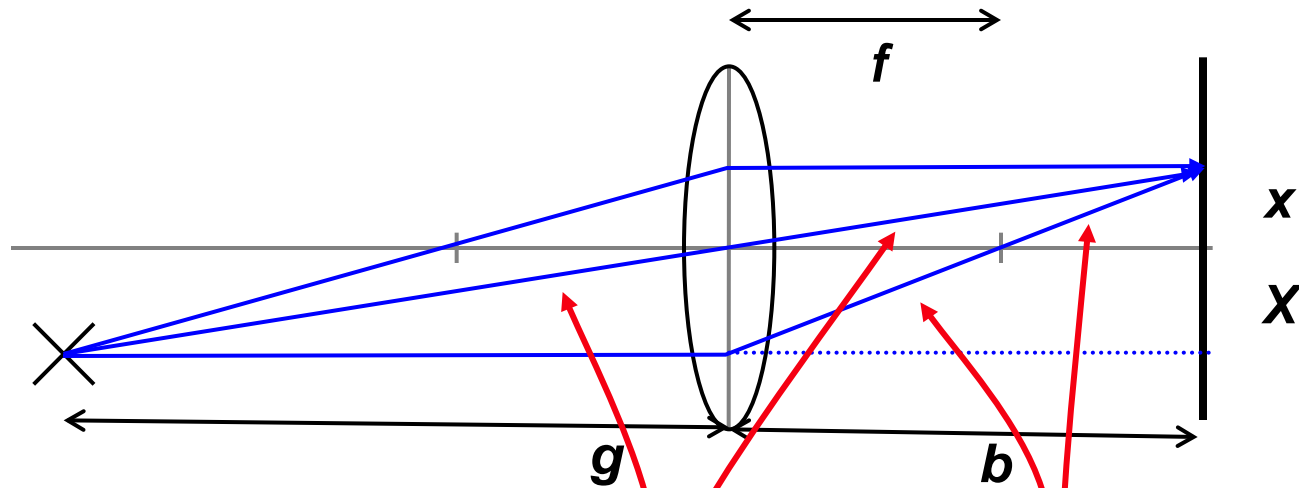
- ▶ Camera obskura erzeugt ein scharfes Bild in der Bildebene, weil von jedem Objektpunkt nur ein Strahl durch das punktförmige optische Zentrum auf die Bildebene führt.
- ▶ Praktisch muss das Loch groß genug Licht durchzulassen, dadurch Unschärfe
- ▶ Linsenkamera erzeugt ein scharfes Bild, weil die Linse alle Strahlen eines Objektpunktes in einem Punkt der Bildebene sammelt.

Der Weg des Lichtes



- ▶ Linse konzentriert parallele Strahlen (Entfernung ∞) in der Brennebene (f)
- ▶ Strahlen parallel zur optischen Achse im Brennpunkt
- ▶ Strahlen durch das optische Zentrum bleiben unbeeinflusst.
- ▶ Daher opt. Zentrum wie bei Lochkamera

Der Weg des Lichtes



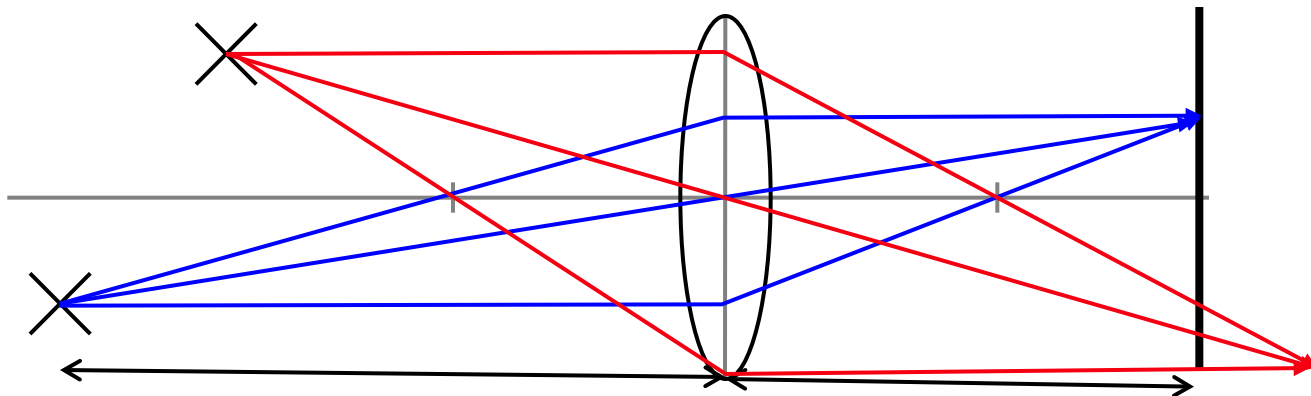
► Linsengesetz

$$\frac{1}{g} + \frac{1}{b} = \frac{1}{f}, \text{ weil } \frac{X}{g} + \frac{X}{b} = \frac{x}{b} + \frac{X}{b} = \frac{x+X}{b} = \frac{X}{f}$$

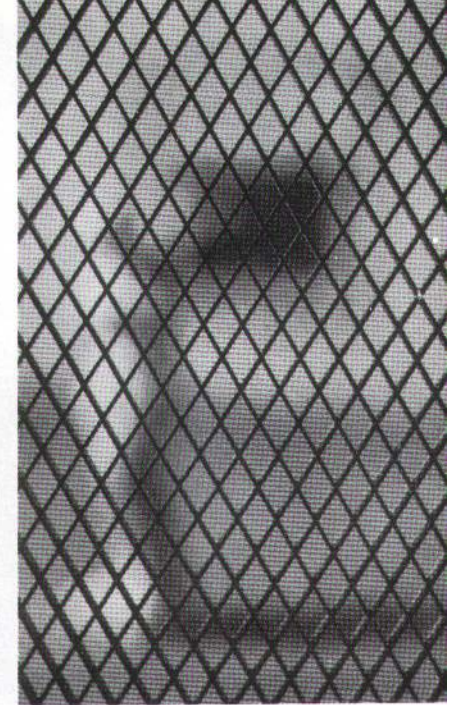
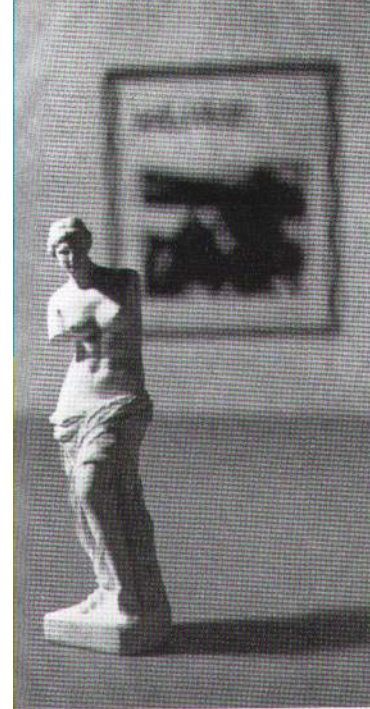
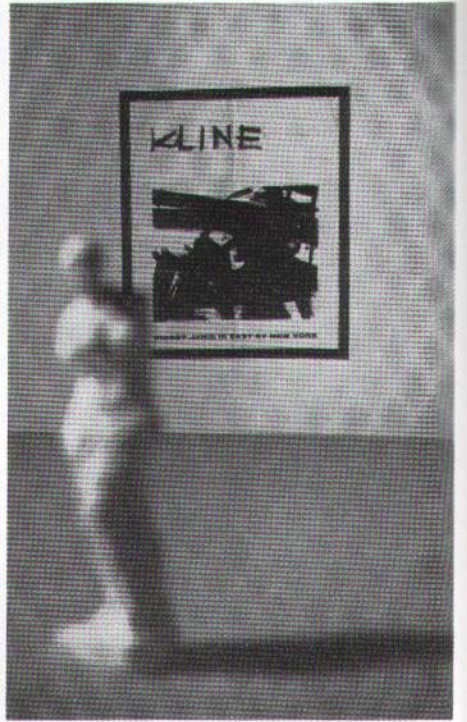
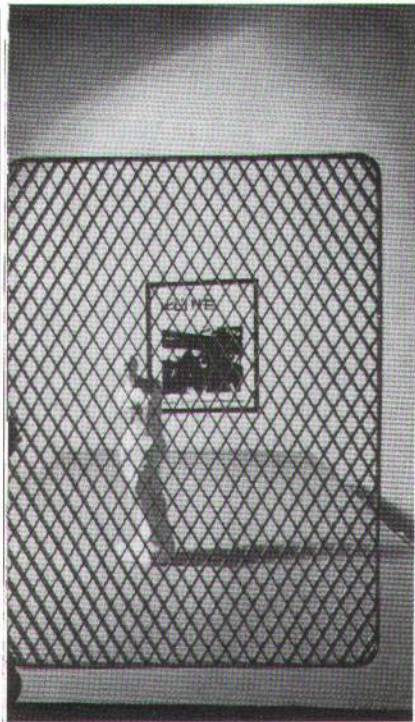
► Beweis durch zwei mal Strahlensatz

Der Weg des Lichtes

- ▶ Nur Objekte in einer festen Entfernung g werden (theoretisch) scharf abgebildet (vgl. **blaue** vs. **rote** Pfeile)
- ▶ Objektive müssen scharfgestellt, fokussiert werden (Ändern von b)
- ▶ Im Normalfall $g \gg f$, dadurch sind Änderungen in b klein und ein ganzer Bereich wird praktisch scharf abgebildet (Schärfentiefe)
- ▶ Je größer die Linse (kleine Blende) desto größer wird die Unschärfe, d.h. desto kleiner die Schärfentiefe
- ▶ Schärfentiefe vor allen Dingen ein Problem im Nahbereich



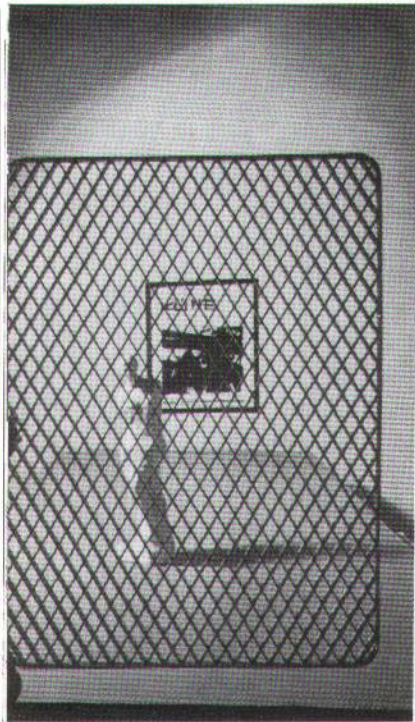
Der Weg des Lichtes



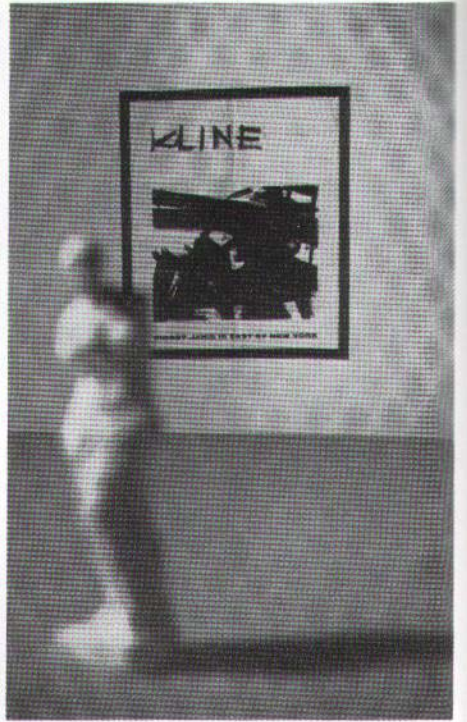
**Frage an das Auditorium:
Welche Blende (klein, gross), welcher Fokus (vorn, mitte, hinten)?**

Quelle: A. Feininger: Die hohe Schule der Photographie, 1961

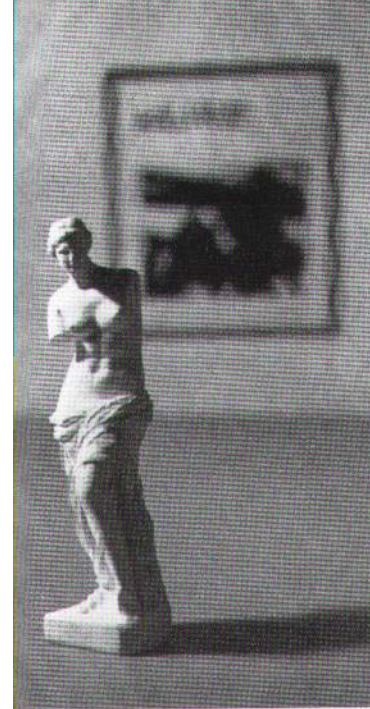
Der Weg des Lichtes



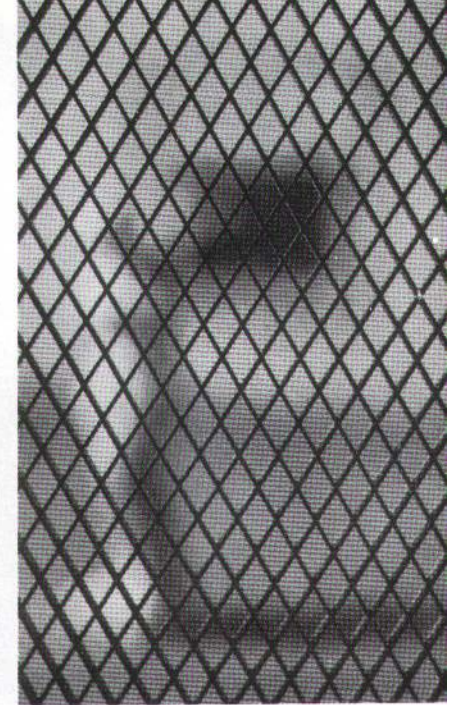
**Große Blende
/ Schärfentiefe**



**Kleine Blende
Fokus hinten**



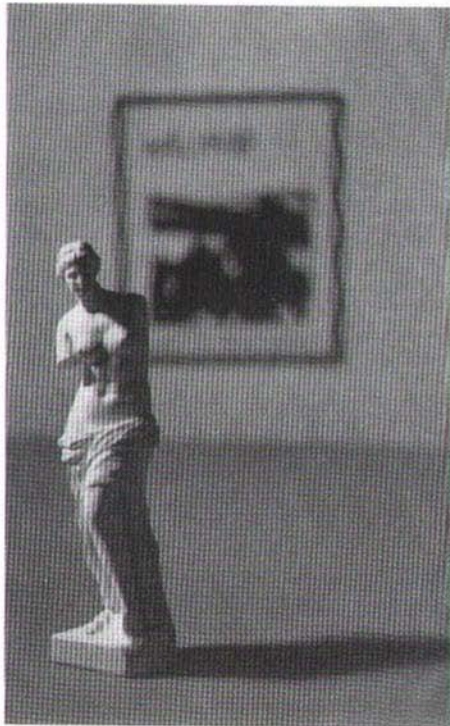
**Kleine Blende
Fokus mitte**



**Kleine Blende
Fokus vorne**

Quelle: A. Feininger: Die hohe Schule der Photographie, 1961

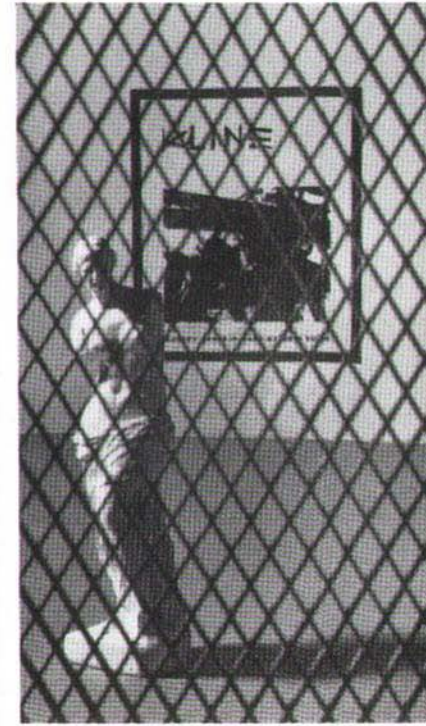
Der Weg des Lichtes



**Kleine Blende
/ Schärfentiefe**



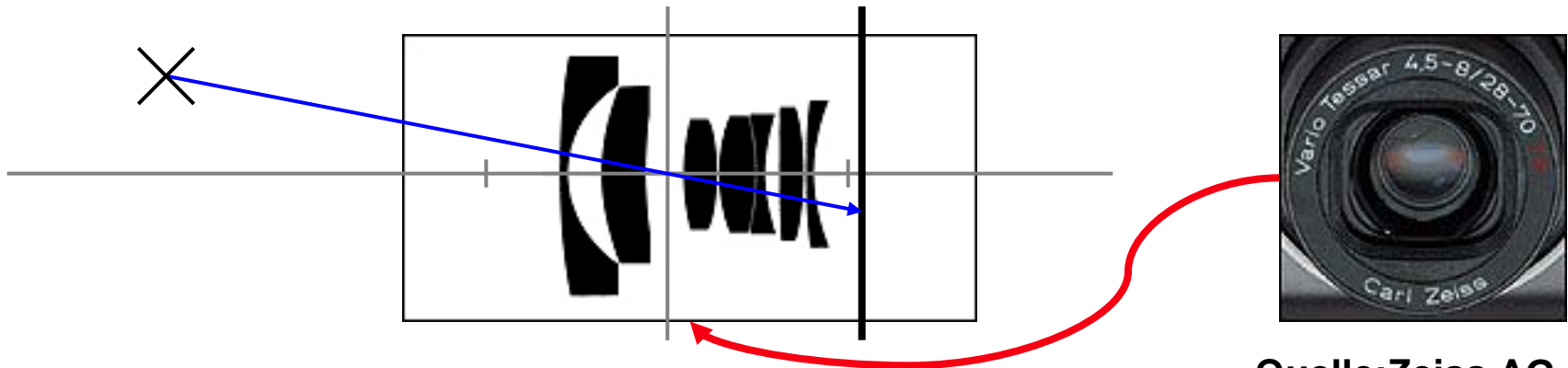
**Mittlere Blende
/ Schärfentiefe**



**Große Blende
/ Schärfentiefe**

Quelle: A. Feininger: Die hohe Schule der Photographie, 1961

Der Weg des Lichtes



Quelle: Zeiss AG
(www.zeiss.de)

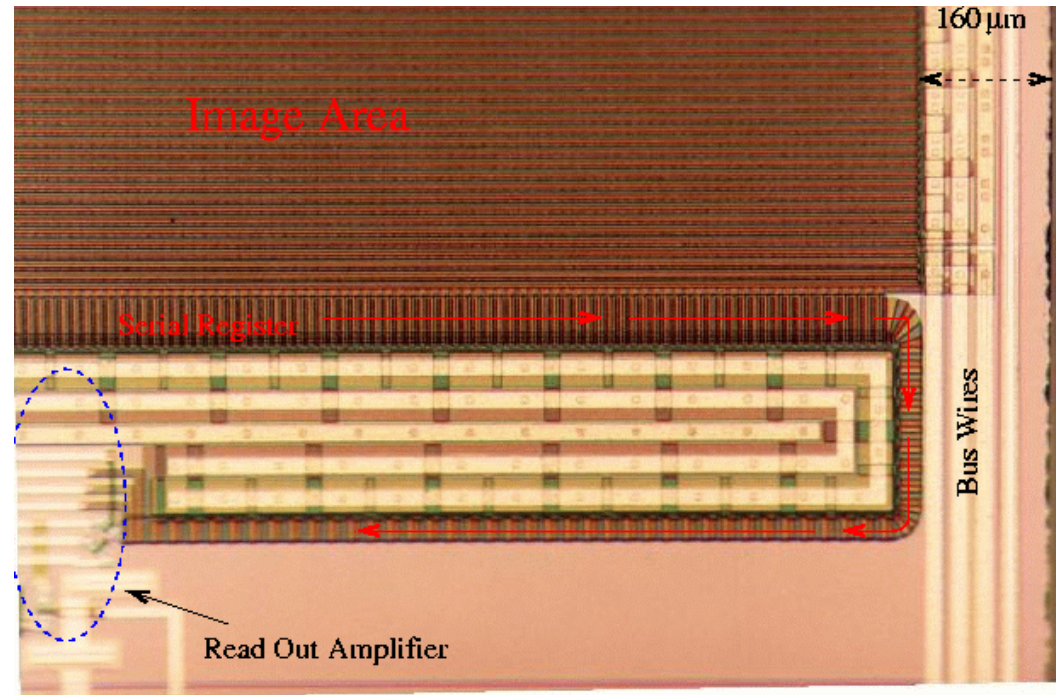
Komplexe Objektive

- ▶ Moderne Objektive haben mehrere (3-9) Linsen
- ▶ Zoom, Fokus, Korrektur von Linsen- und Farbfehlern
- ▶ Aus Bildverarbeitungssicht wie eine einzelne Linse mit...
 - ▶ Optischem Zentrum
 - ▶ Optischer Achse
 - ▶ Brennweite

Der Weg des Lichtes

Charge Coupled Device (CCD)

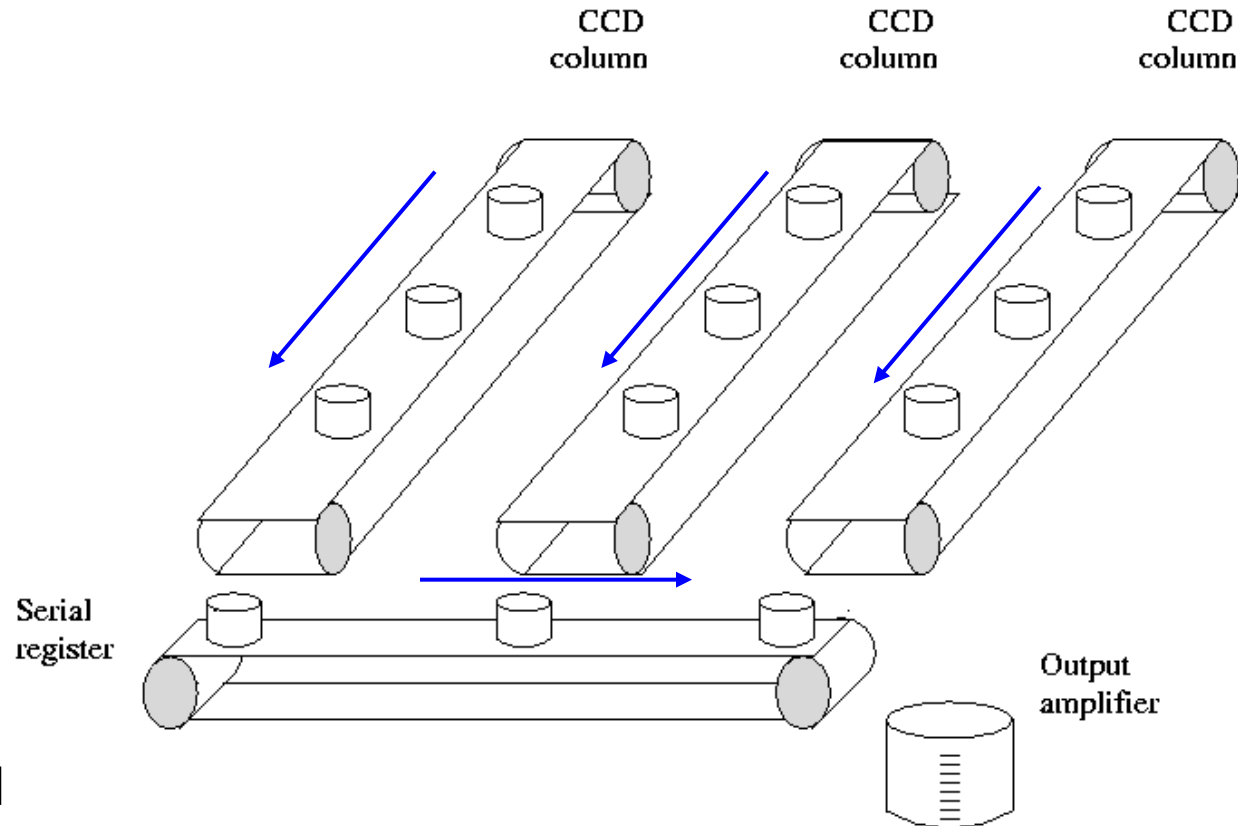
- ▶ **Sitzt in der Bildebene**
- ▶ **Rechteckiges Feld lichtempfindlicher Zellen (Pixel)**
- ▶ **Wandelt für jeden Pixel die Helligkeit in Spannung um (analog)**
- ▶ **Ende des Lichtweges, danach elektrisch**
- ▶ **Auslesen durch Verschieben der Ladung (Charge Coupled)**
- ▶ **Monochrom, Farbe später.**



Quelle (diese Grafik und folgende): M. Richmond, S. Tulloch: Introduction to CCDs (<http://spiff.rit.edu/classes/phys445/lectures/ccd1/ccd1.html>)

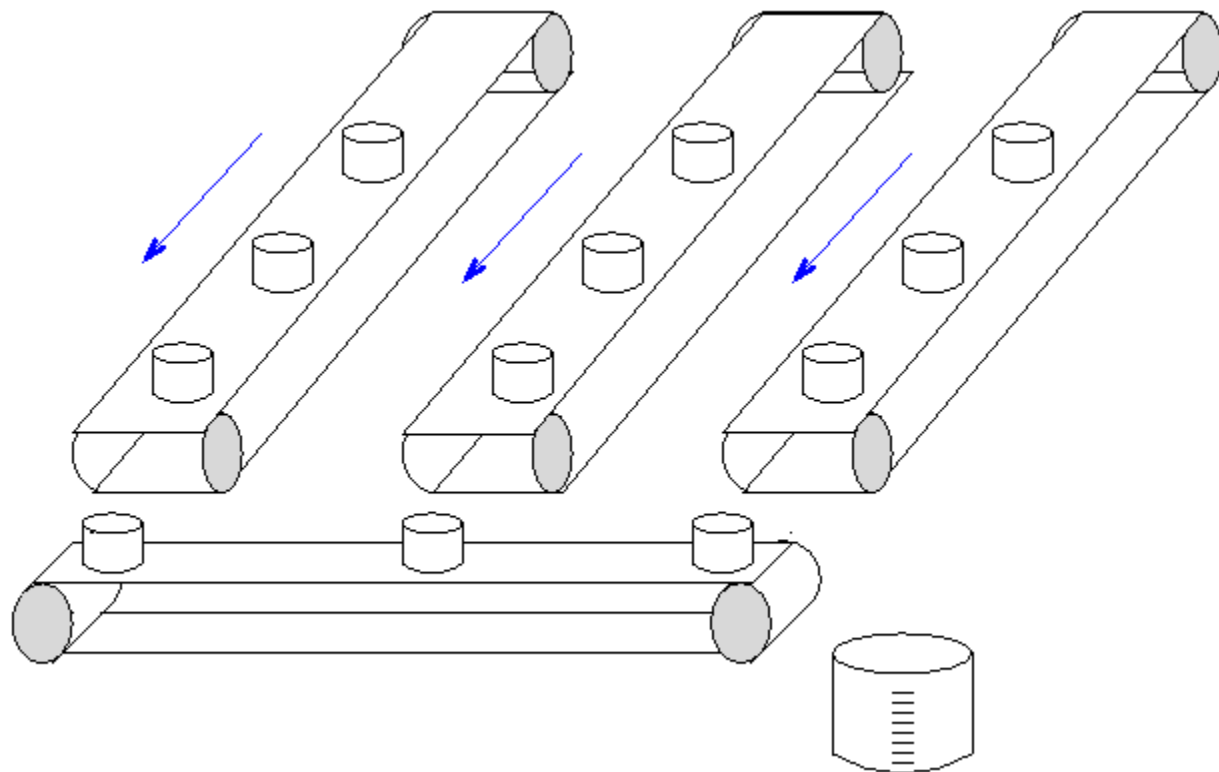
Der Weg des Lichtes

- ▶ **Analogie:**
Förderbänder von Eimern
- ▶ **Im 2D Feld von Eimern (Pixel) wird Regen (Ladung) gesammelt.**
- ▶ **Die Spaltenförderbänder schieben die Eimer zeilenweise nach unten**
- ▶ **Ein Zeilenförderband schiebt die Eimer zum Ausgang**



Der Weg des Lichtes

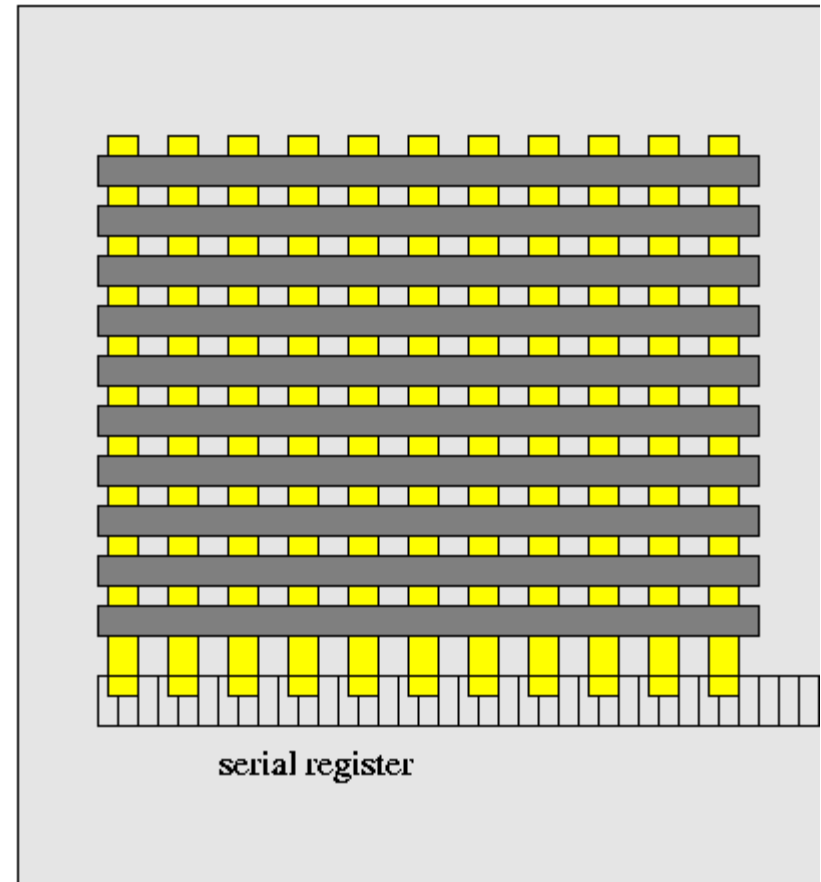
- ▶ **Analogie:**
Förderbänder von Eimern
- ▶ **Im 2D Feld von Eimern (Pixel) wird Regen (Ladung) gesammelt.**
- ▶ **Die Spaltenförderbänder schieben die Eimer zeilenweise nach unten**
- ▶ **Ein Zeilenförderband schiebt die Eimer zum Ausgang**



Der Weg der Ladung

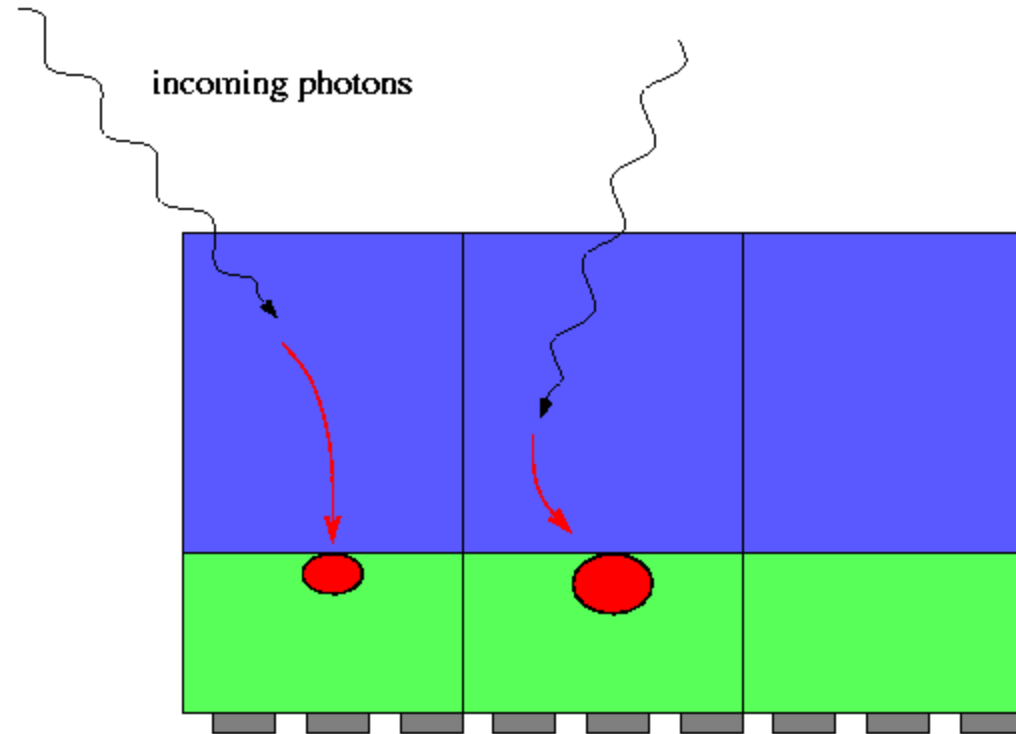
Charge Coupled Device (CCD)

- ▶ Pixel-Spalten durch isolierte Halbleiterregionen (**gelb**)
- ▶ Pixel-Zeilen durch jeweils 3 transparente Elektroden (**grau**)
- ▶ Ladung entsteht unter den Elektroden durch Licht (photoelektrischer Effekt)
- ▶ Vertikaler Transport durch Spannungsmuster an Elektroden
- ▶ Horizontaler Transport im „serial Register“ ebenfalls durch Spannungsmuster
- ▶ Serialisierung eines 2D Feldes



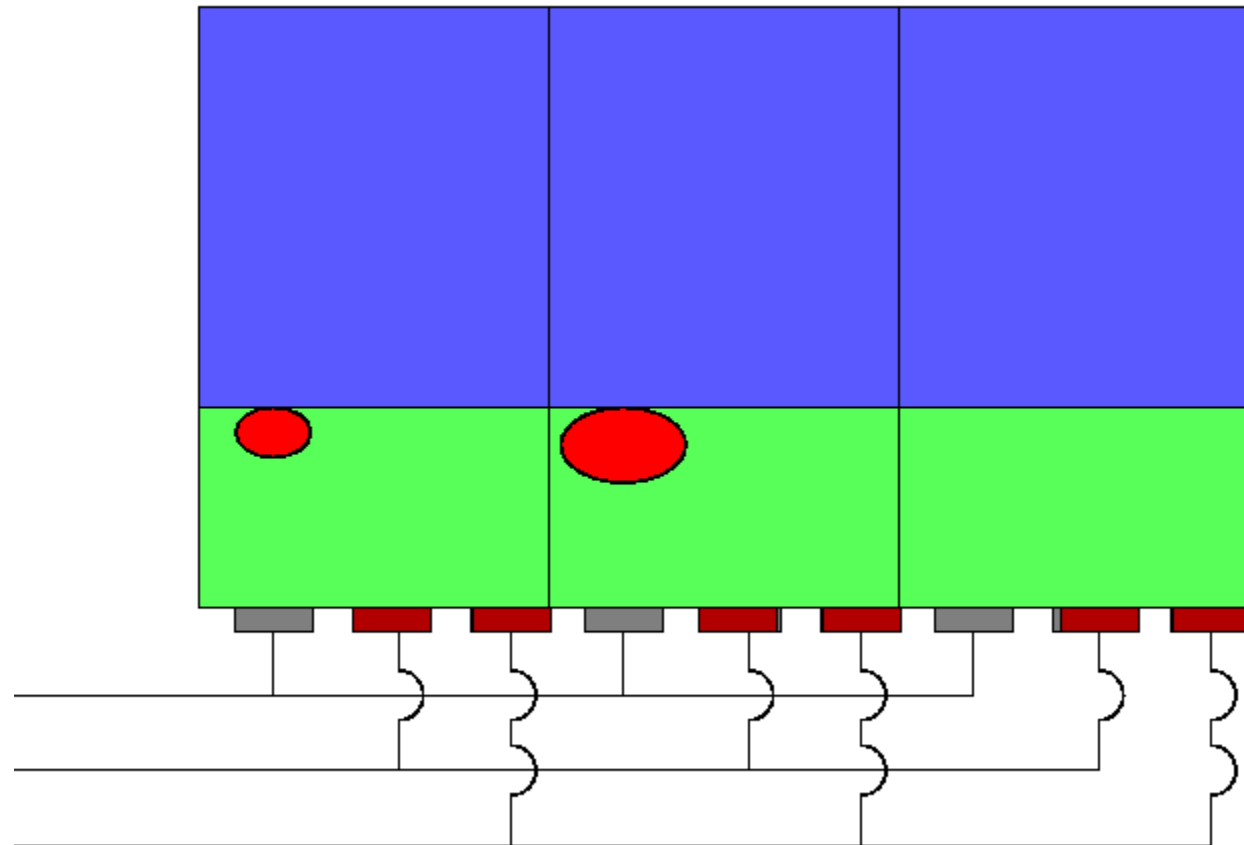
Der Weg der Ladung

- ▶ Photonen (Licht) erzeugt Ladung (**rot**), die sich zwischen n-dotierter Zone (**grün**) und p-dotierter Zone (**blau**) sammelt.



Der Weg der Ladung

- ▶ Ladung sammelt sich unter positiver Elektrode
- ▶ Durch Spannungsmuster an Elektroden kann die Ladung bewegt werden



► Frage an das Auditorium: Woher kommen die Lichter auf der Säule.

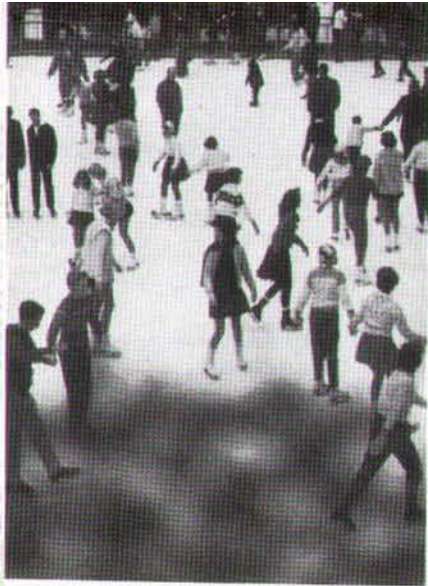


Video 1

Der Weg der Ladung

- ▶ **Ladung wird von den lichtempfindlichen Zellen synchron in lichtgeschützte übertragen**
- ▶ **Dadurch Belichtungszeit (Shutter, Exposure) elektronisch steuerbar**
- ▶ **Digitalisieren der Ladungsmenge in Pixeldaten mit steuerbarer Verstärkung (Gain)**
- ▶ **Meistens Gammakurve: $\text{pixel} = \text{intensität}^{0.45}$**
- ▶ **Zusammenfassung CCD:**
 - ▶ Licht wird in Ladung umgewandelt
 - ▶ Ladung zeilenweise nach unten geschoben
 - ▶ Zeilen pixelweise nach rechts geschoben
 - ▶ Pixelweise digitalisiert
 - ▶ Belichtungszeit (Shutter), Verstärkung (Gain) einstellbar

Der Weg der Ladung



1/1000s, Blende 2.8 1/250s, Blende 5.6

1/50s, Blende 12.5 1/10s, Blende 29

- ▶ Lange Belichtungszeit führt zu unscharfer Bewegung
- ▶ Niedrige Blende (großes Objektiv) führt zu geringer Schärfentiefe

Quelle: A. Feininger: Die hohe Schule der Photographie, 1961

Der Weg der Daten

- ▶ Bei analogen (TV-Video) Kameras werden die Daten erheblich durch Bandfilterung, Farbaufmodulierung, Pixeljitter und viele andere Probleme beeinflusst.
- ▶ Bei digitalen Kameras direkt im Chip digitalisiert und per USB, FireWire (IEEE 1394) oder Ethernet an den Rechner geschickt
- ▶ Daten vom Treiber angenommen
- ▶ Per Bus-Master im Speicher abgelegt
- ▶ Für ein $w \times h$ Monochrombild:
1 Byte je Pixel, w Pixel je Zeile, h Zeilen je Bild
rel. Adresse von Pixel x, y ist $x+w*y$
- ▶ `IplImage image;`
`image.imageData[x+image.width*y]`



Der Weg des Bildes in den Rechner

Faktoren, die die Bildhelligkeit beeinflussen.

	Probleme wenn groß	Probleme wenn klein
Objektivdurchmesser (1/Blende)	geringe Schärfentiefe	wenig Licht
Belichtungszeit (Shutter)	Bewegungsunschärfe	wenig Licht
Verstärkung (Gain)	hohes Rauschen	benötigt viel Licht
Zoom	wenig Licht	wenig Auflösung
Beleuchtungsstärke	aufwändig	wenig Licht
Beleuchtungsentfernung	wenig Licht	ungleichmäßige Beleuchtung

Der Weg des Bildes in den Rechner

Frage an das Auditorium:

Welche Kombination Beleuchtung, Blende, Belichtung, Verstärkung:

- ▶ **In einer urigen Kneipe das Billardspiel verfolgen**
- ▶ **Fußballspielender Roboter**
- ▶ **Kamera an der Spitze eines Flugzeugs**
- ▶ **Laufroboter im Gebüsch**

Der Weg des Bildes in den Rechner

Frage an das Auditorium:

Welche Kombination Beleuchtung, Blende, Belichtung, Verstärkung:

- ▶ **In einer urigen Kneipe das Billardspiel verfolgen**
 - ▶ Kamera über Tisch, großes Objektiv / kleine Blende, kurze Belichtung
- ▶ **Fußballspielender Roboter**
 - ▶ kleines Objektiv / große Blende, kurze Belichtung, viel Licht
- ▶ **Kamera an der Spitze eines Flugzeugs**
 - ▶ großes Objektiv / kleine Blende (alles Gesehene ist weit entfernt), kurze Belichtung
- ▶ **Laufroboter im Gebüsch**
 - ▶ kleines Objektiv / große Blende, lange Belichtung

Der industrielle Ansatz

Der industrielle Ansatz

- ▶ **Übungsblatt 1**
- ▶ **Standardvorgehensweise für einfache BV Probleme in der Qualitätssicherung**
- ▶ **Objekt einzeln vor Durchlichtkasten**
- ▶ **Rechenschritte:**
 - ▶ Binarisierung mit Intensitätsschwellwert (dunkel = Objekt, hell = Hintergrund)
 - ▶ Regionenbildung (Run Length Encoding, union-find Algorithmus)
 - ▶ Merkmale der Regionen (Fläche, Länge, Breite, Schwerpunkt, Hauptträgheitsachsen, etc.)
 - ▶ Anwendungsspezifische Beurteilung (stimmen Position, Masse, etc.)

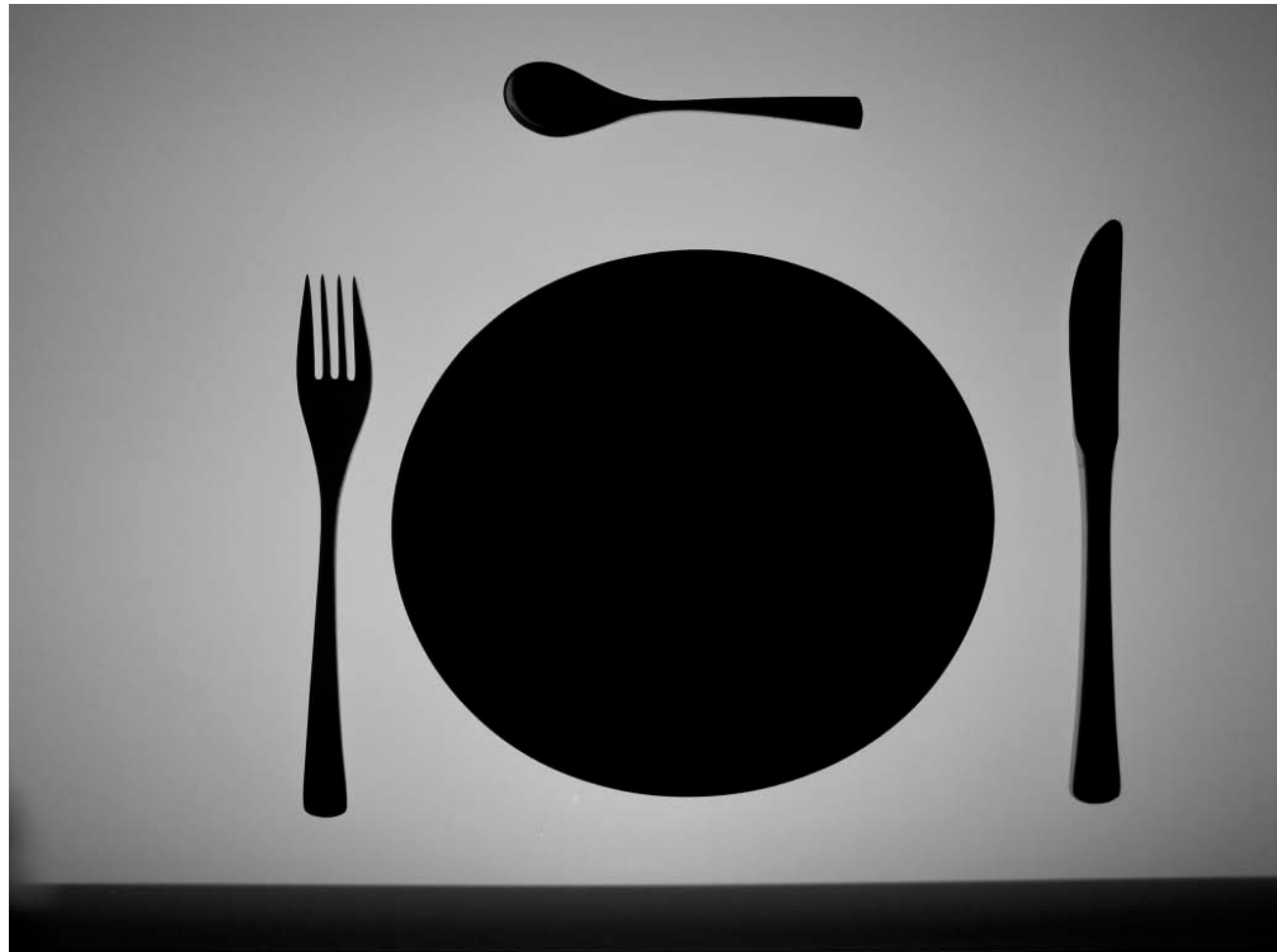


Quelle:Leutron Vision Inc.,
(www.leutron.com)

Der industrielle Ansatz

Bildaufnahme vor Durchlichtkasten

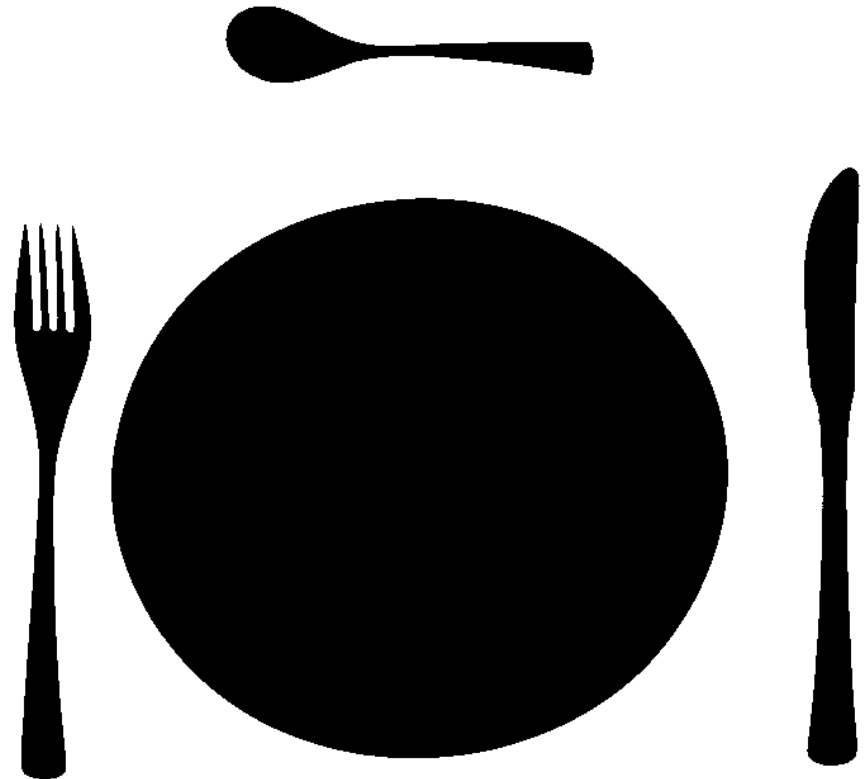
- ▶ Beispielszene aus den Übungen
- ▶ Fast perfektes Bild
- ▶ Leichter Helligkeitsgradient
- ▶ Leichte Reflexionen an Löffel und Messer



Schwellewert & Morphologie

Binarisierung

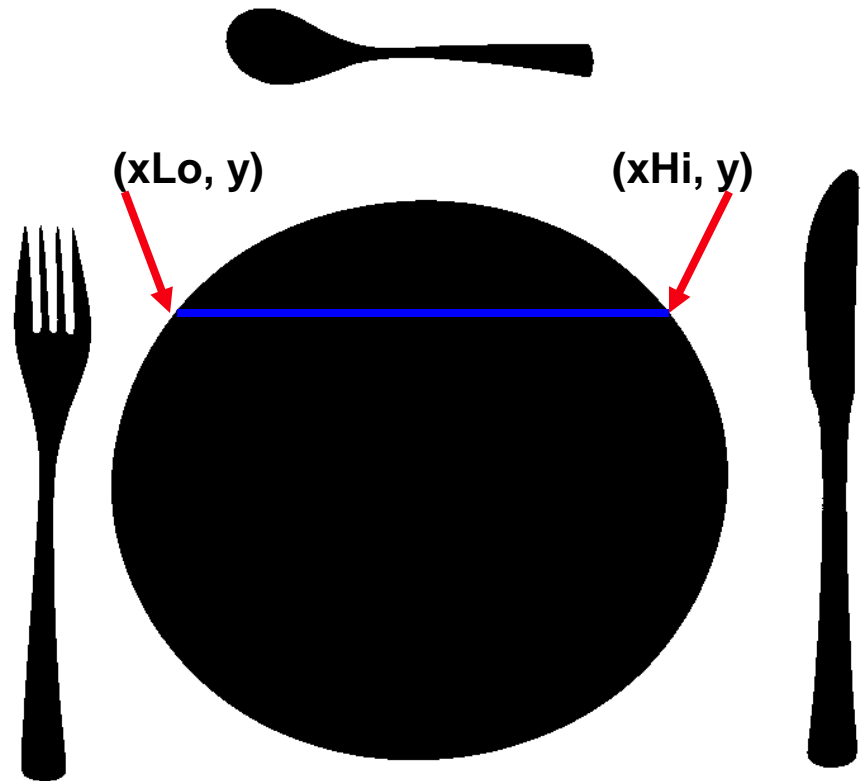
- ▶ **Festen Schwellwert einstellen**
- ▶ **oder automatischer Schwellwert (nächste Vorlesung)**
- ▶ **oder Beleuchtungsgradienten bestimmen**
- ▶ **Resultierendes Bild mit einem Bit / Pixel speichern**



Schwellewert

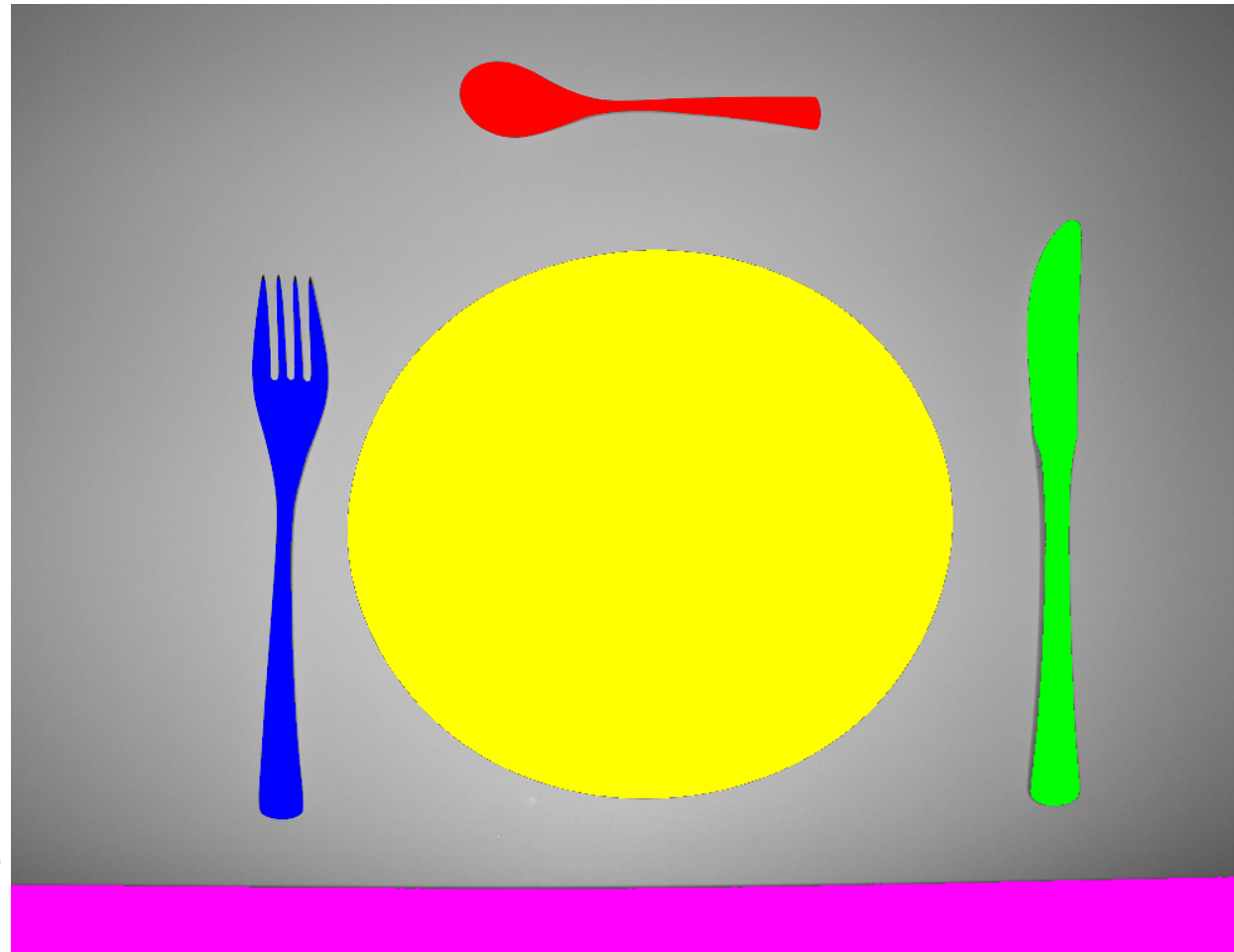
Binarisierung

- ▶ Bit / Pixel speichern
- ▶ oder in horizontale Streifen (Intervalle) zerlegen (run length encoding)
- ▶ Jeder Streifen (xLo , xHi , y)
- ▶ Streifen nach y und innerhalb einer Zeile nach x sortiert
- ▶ Zuerst alle Regionen gemischt, später einzelne Zusammenhangskomponenten finden



Regionenbildung

- ▶ **Aufgabe:** Region aus zusammenhängenden schwarzen Pixeln bilden, d.h. mit einheitlicher Zahl/Farbe versehen
- ▶ **Grundidee:** „Ist Dein Nachbarpixel noch schwarz, fülle ihn!“
- ▶ **Grassfire Algorithmus:** Breitensuche
- ▶ **Floodfill Algorithmus:** (rekursive) Tiefensuche



Regionenbildung (Floodfill Alg.)

```
fill (image, x, y, regionColor) { // fülle (x,y) und seine schwarze Region
    if (image(x,y)!=black) return; // schon gefüllt oder weiß
    image(x,y) = regionColor;
    if (x>0) fill (image, x-1, y, regionColor);
    if (x<image.width-1) fill (image, x+1, y, regionColor);
    if (y>0) fill (image, x, y-1, regionColor);
    if (y<image.height-1) fill (image, x, y+1, regionColor);
}

groupRegions (image) { // fülle Regionen mit unterschiedlichen Farben
    regionColor = firstColor;
    for (int y=0; y<image.height; y++)for (int x=0; x<image.width; x++)
        if (image(x,y)==black) // neue ungefüllte Region
            fill (image, x, y, regionColor++);
}
```

Regionenbildung (Floodfill Alg.)

- ▶ Rechenzeit $O(\text{width} \cdot \text{height})$
- ▶ Schneller als Grassfire
- ▶ Obige Implementierung langsam wegen Rekursions-Overhead
- ▶ Schneller: Binärbild zuerst in run-length-code Intervalle konvertieren, dann Regionenbildung dort durchführen.
- ▶ Grundidee: „Laufe mit zwei Zeigern im Zeilenabstand durch die Intervalle, immer wenn sich zwei Intervalle berühren, vereinige die Regionen“

Regionenbildung (Union-Find-Alg.)

```
groupRegions (vector<Interval>& rle) { // Setze .region in Regionen
    initializeRegions();
    run = flw = rle.begin();
    while (run!=rle.end()) {
        while (flw->y < run->y-1 || // flw links/oberhalb kein Kontakt
                flw->y == run->y-1 && flw->xHi < run->xLo) flw++;
        if (flw->y == run->y-1 && flw->xLo <= run->xHi) { // run-flw Kontakt
            unite (run, flw);
            if (flw->xHi>run->xHi) run++; // run kann keinen flw++ Kontakt haben
            else flw++; // flw kann keinen run++ Kontakt haben
        }
        else run++; // flw hat keinen iv Kontakt
    }
}
```

- ▶ run (**roter** Rahmen) läuft durch die Intervalle
- ▶ flw (**blauer** Rahmen) läuft eine Zeile hinterher



Regionenbildung (Union-Find-Alg.)

```
groupRegions (vector<Interval>& rle) { // Setze .region in Regionen
    initializeRegions();
    run = flw = rle.begin();
    while (run!=rle.end()) {
        while (flw->y < run->y-1 || // flw links/oberhalb kein Kontakt
                flw->y == run->y-1 && flw->xHi < run->xLo) flw++;
        if (flw->y == run->y-1 && flw->xLo <= run->xHi) { // run-flw Kontakt
            unite (run, flw);
            if (flw->xHi>run->xHi) run++; // run kann keinen flw++ Kontakt haben
            else flw++; // flw kann keinen run++ Kontakt haben
        }
        else run++; // flw hat keinen iv Kontakt
    }
}
```



Regionenbildung (Union-Find-Alg.)

```
groupRegions (vector<Interval>& rle) { // Setze .region in Regionen
    initializeRegions();
    run = flw = rle.begin();
    while (run!=rle.end()) {
        while (flw->y < run->y-1 || // flw links/oberhalb kein Kontakt
                flw->y == run->y-1 && flw->xHi < run->xLo) flw++;
        if (flw->y == run->y-1 && flw->xLo <= run->xHi) { // run-flw Kontakt
            unite (run, flw);
            if (flw->xHi>run->xHi) run++; // run kann keinen flw++ Kontakt haben
            else flw++; // flw kann keinen run++ Kontakt haben
        }
        else run++; // flw hat keinen iv Kontakt
    }
}
```



Regionenbildung (Union-Find-Alg.)

```
groupRegions (vector<Interval>& rle) { // Setze .region in Regionen
    initializeRegions();
    run = flw = rle.begin();
    while (run!=rle.end()) {
        while (flw->y < run->y-1 || // flw links/oberhalb kein Kontakt
                flw->y == run->y-1 && flw->xHi < run->xLo) flw++;
        if (flw->y == run->y-1 && flw->xLo <= run->xHi) { // run-flw Kontakt
            unite (run, flw);
            if (flw->xHi>run->xHi) run++; // run kann keinen flw++ Kontakt haben
            else flw++; // flw kann keinen run++ Kontakt haben
        }
        else run++; // flw hat keinen iv Kontakt
    }
}
```



Regionenbildung (Union-Find-Alg.)

```
groupRegions (vector<Interval>& rle) { // Setze .region in Regionen
    initializeRegions();
    run = flw = rle.begin();
    while (run!=rle.end()) {
        while (flw->y < run->y-1 || // flw links/oberhalb kein Kontakt
                flw->y == run->y-1 && flw->xHi < run->xLo) flw++;
        if (flw->y == run->y-1 && flw->xLo <= run->xHi) { // run-flw Kontakt
            unite (run, flw);
            if (flw->xHi>run->xHi) run++; // run kann keinen flw++ Kontakt haben
            else flw++; // flw kann keinen run++ Kontakt haben
        }
        else run++; // flw hat keinen iv Kontakt
    }
}
```



Regionenbildung (Union-Find-Alg.)

```
groupRegions (vector<Interval>& rle) { // Setze .region in Regionen
    initializeRegions();
    run = flw = rle.begin();
    while (run!=rle.end()) {
        while (flw->y < run->y-1 || // flw links/oberhalb kein Kontakt
                flw->y == run->y-1 && flw->xHi < run->xLo) flw++;
        if (flw->y == run->y-1 && flw->xLo <= run->xHi) { // run-flw Kontakt
            unite (run, flw);
            if (flw->xHi>run->xHi) run++; // run kann keinen flw++ Kontakt haben
            else flw++; // flw kann keinen run++ Kontakt haben
        }
        else run++; // flw hat keinen iv Kontakt
    }
}
```



Regionenbildung (Union-Find-Alg.)

```
groupRegions (vector<Interval>& rle) { // Setze .region in Regionen
    initializeRegions();
    run = flw = rle.begin();
    while (run!=rle.end()) {
        while (flw->y < run->y-1 || // flw links/oberhalb kein Kontakt
                flw->y == run->y-1 && flw->xHi < run->xLo) flw++;
        if (flw->y == run->y-1 && flw->xLo <= run->xHi) { // run-flw Kontakt
            unite (run, flw);
            if (flw->xHi>run->xHi) run++; // run kann keinen flw++ Kontakt haben
            else flw++; // flw kann keinen run++ Kontakt haben
        }
        else run++; // flw hat keinen iv Kontakt
    }
}
```



Regionenbildung (Union-Find-Alg.)

```
groupRegions (vector<Interval>& rle) { // Setze .region in Regionen
    initializeRegions();
    run = flw = rle.begin();
    while (run!=rle.end()) {
        while (flw->y < run->y-1 || // flw links/oberhalb kein Kontakt
                flw->y == run->y-1 && flw->xHi < run->xLo) flw++;
        if (flw->y == run->y-1 && flw->xLo <= run->xHi) { // run-flw Kontakt
            unite (run, flw);
            if (flw->xHi > run->xHi) run++; // run kann keinen flw++ Kontakt haben
            else flw++; // flw kann keinen run++ Kontakt haben
        }
        else run++; // flw hat keinen iv Kontakt
    }
}
```



Regionenbildung (Union-Find-Alg.)

```
groupRegions (vector<Interval>& rle) { // Setze .region in Regionen
    initializeRegions();
    run = flw = rle.begin();
    while (run!=rle.end()) {
        while (flw->y < run->y-1 || // flw links/oberhalb kein Kontakt
               flw->y == run->y-1 && flw->xHi < run->xLo) flw++;
        if (flw->y == run->y-1 && flw->xLo <= run->xHi) { // run-flw Kontakt
            unite (run, flw);
            if (flw->xHi>run->xHi) run++; // run kann keinen flw++ Kontakt haben
            else flw++; // flw kann keinen run++ Kontakt haben
        }
        else run++; // flw hat keinen iv Kontakt
    }
}
```



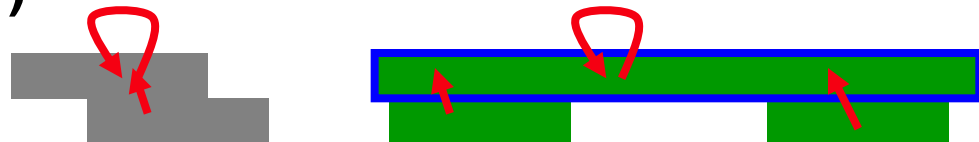
Regionenbildung (Union-Find-Alg.)

```
groupRegions (vector<Interval>& rle) { // Setze .region in Regionen
    initializeRegions();
    run = flw = rle.begin();
    while (run!=rle.end()) {
        while (flw->y < run->y-1 || // flw links/oberhalb kein Kontakt
                flw->y == run->y-1 && flw->xHi < run->xLo) flw++;
        if (flw->y == run->y-1 && flw->xLo <= run->xHi) { // run-flw Kontakt
            unite (run, flw);
            if (flw->xHi>run->xHi) run++; // run kann keinen flw++ Kontakt haben
            else flw++; // flw kann keinen run++ Kontakt haben
        }
        else run++; // flw hat keinen iv Kontakt
    }
}
```



Regionenbildung (Union-Find Alg.)

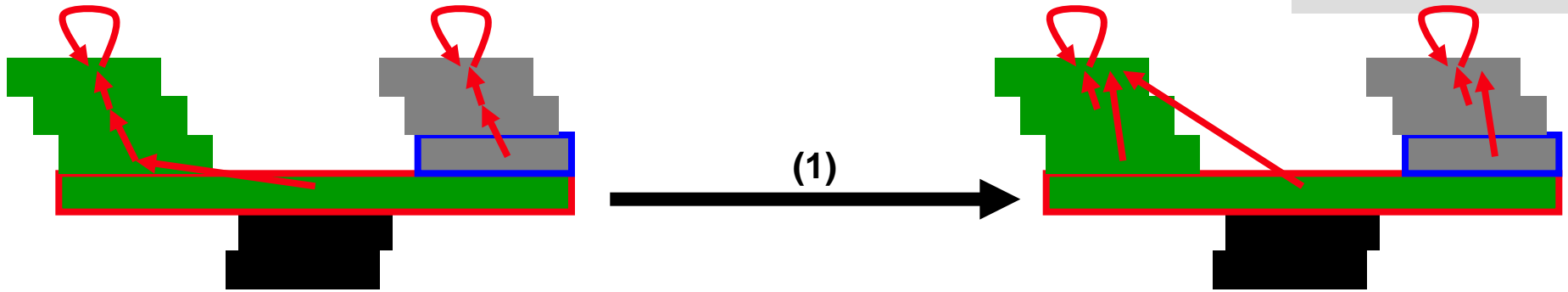
- ▶ Wie macht man `unite` effizient? Union-Find Datenstruktur
- ▶ Intervall hält Zeiger auf ein früheres Intervall der selben Region
- ▶ Erstes Intervall (Wurzel) hält aus technischen Gründen einen Zeiger auf sich selbst statt eines Nullzeigers
- ▶ Eine Region sind all die Intervalle mit gemeinsamen Wurzelknoten
- ▶ Vereinigen zweier Regionen `run` und `flw`
 - ▶ Wurzelintervall von `run` und `flw` finden.
 - ▶ Den späteren von beiden auf den früheren zeigen lassen
- ▶ Pfadkomprimierung: Während des Suchens des Wurzelintervalls alle Intervalle direkt auf die Wurzel zeigen lassen.
- ▶ Rechenzeit amortisiert fast $O(1)$



Regionenbildung (Union-Find-Alg.)

```
class Interval {
    int xLo, xHi, y; // horizontaler Streifen (xLo,y)-(xHi,y)
    int region;      // Nummer der Region
    Interval* parent; // Für union-find-Baum, initialisiert: auf sich selbst
}
initializeRegions (vector<Interval>&rle) { // Mach jedes Interval zu einer Region
    for (it=rle.begin(); it!=rle.end(); it++) (*it).parent = &(*it);
}
pathCompress (Interval* iv) { // Hänge Knoten bis zur Wurzel direkt an die Wurzel
    root = iv;
    while (root->parent!=root) root = root->parent; // Zur Wurzel laufen
    while (iv!=root) {
        buffer = iv->parent;
        iv->parent = root; // Direkt an die Wurzel hängen
        iv = buffer;
    }
}
unite (Interval* run, Interval* flw) { // Vereinige Regionen von run und flw
    pathCompress (run); pathCompress (flw);
    if (run->parent<flw->parent) flw->parent->parent = run->parent;
    else run->parent->parent = flw->parent;
}
```

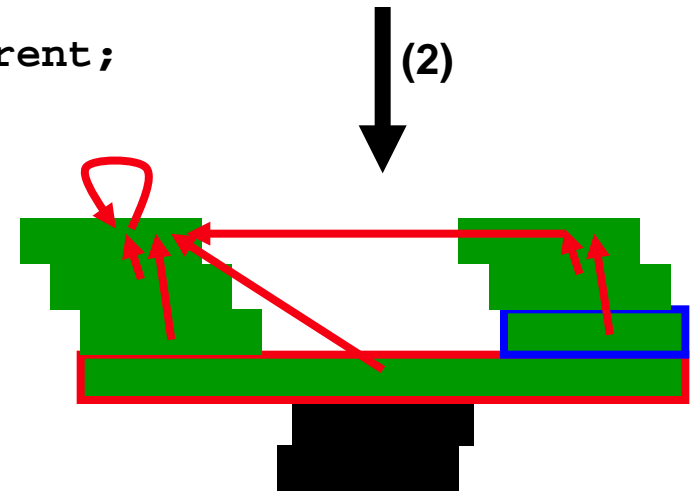
Regionenbildung (Union-Find-Alg.)



```

pathCompress (iv)
  root = iv;
  while (root->parent!=root) root = root->parent;
  while (iv!=root) {
    buffer = iv->parent; iv->parent = root;
    iv = buffer;
  }
}
unite (run, flw) {
(1) pathCompress (run); pathCompress (flw);
(2) if (run->parent<flw->parent)
    flw->parent->parent = run->parent;
    else run->parent->parent = flw->parent;
}

```



Regionenbildung (Union-Find-Alg.)

```
void setRegionIndex (vector<Interval>& rle)
{
    iv = rle.begin();
    regionCtr = 0;
    while (iv!=rle.end()) {
        if (iv->parent==iv) {
            iv->region = regionCtr;
            regionCtr++;
        }
        else iv->region = iv->parent->region;
        iv++;
    }
}
```

- ▶ **Setzen der Indices für die jeweiligen Regionen im Rahmen eines abschließenden Durchlaufes**
- ▶ **Da nicht alle Pfade voll komprimiert sind**

Zusammenfassung

- ▶ **Weg des Bildes in den Rechner wird durch viele Faktoren beeinflusst**
 - ▶ Optik der Szene: Beleuchtung, Verdeckung, Schatten, Reflexionen
 - ▶ Objektiv: Öffnungswinkel, Blende, Belichtung
 - ▶ Beleuchtung, Blende und Belichtung gut wählen, keine Über- / Unterbelichtung
 - ▶ CCD Chip, USB / Firewire Datenübertragung: Beschränkt Auflösung / Bildrate
- ▶ **„Industrieller Ansatz“**
 - ▶ Aufnahme vor Durchlichtkasten, Segmentierung über Schwellwert
 - ▶ Darstellen als Vereinigung von Intervallen (run length encoding)
 - ▶ Regionenbildung über Union-Find-Algorithmus
 - ▶ Merkmale und Applikationsabhängige Bewertung der Merkmale
- ▶ **Union-Find-Algorithmus (fast $O(\text{Intervallzahl})$)**
 - ▶ Zwei Zeiger laufen durch die Intervalle im Abstand einer Zeile
 - ▶ Regionen sich berührende Intervalle werden vereinigt
 - ▶ Region dargestellt durch zur Wurzel gerichteten Baum