03-MB-709.03

Echtzeitbildverarbeitung (8)

Tim Laue
Prof. Dr. Udo Frese

Hough-Transformation für Linien

Bildverarbeitung im RoboCup

Was bisher geschah

- **Hough-Transformation**
  - Suche parametrisierte Kurven, z.B. Geraden, Kreise
  - Hough-Akkumulator im Parameterraum
  - Für jeden Pixel mit hinreichendem Kontrast erhöhe alle Akkumulatorpixel, deren Kurven diesen Pixel durchlaufen
- **Hough-Transformation für Kreise**
  - Houghraum beschreibt Kreise in Mittelpunktsform (x_c, y_c, r)
  - Ausnutzen des Gradienten: (x_c,y_c) = (x, y) +/-r (sobelX, sobelY)/sobelLen
  - Projektion in (x_c, y_c), also weglassen von r weil 3D-Houghraum sehr groß.
  - Für jeden (x_c, y_c) Mittelpunkt mit hinreichend großem Hough-Wert eine 1D. Hough-Transformation bzgl. r durchführen.
- **Derart technisch verwickelte Optimierung sind nur sinnvoll für Teilroutinen die sehr oft ausgeführt werden (z.B. jeden Pixel)**

Ziel der nächsten vier Übungszettel

Aufgabe 3:

Sobel-Filter

Aufgabe 7:

Ball als Kreis mit Hough erkennen

Aufgabe 10:

Linien auf dem Boden mit Hough erkennen

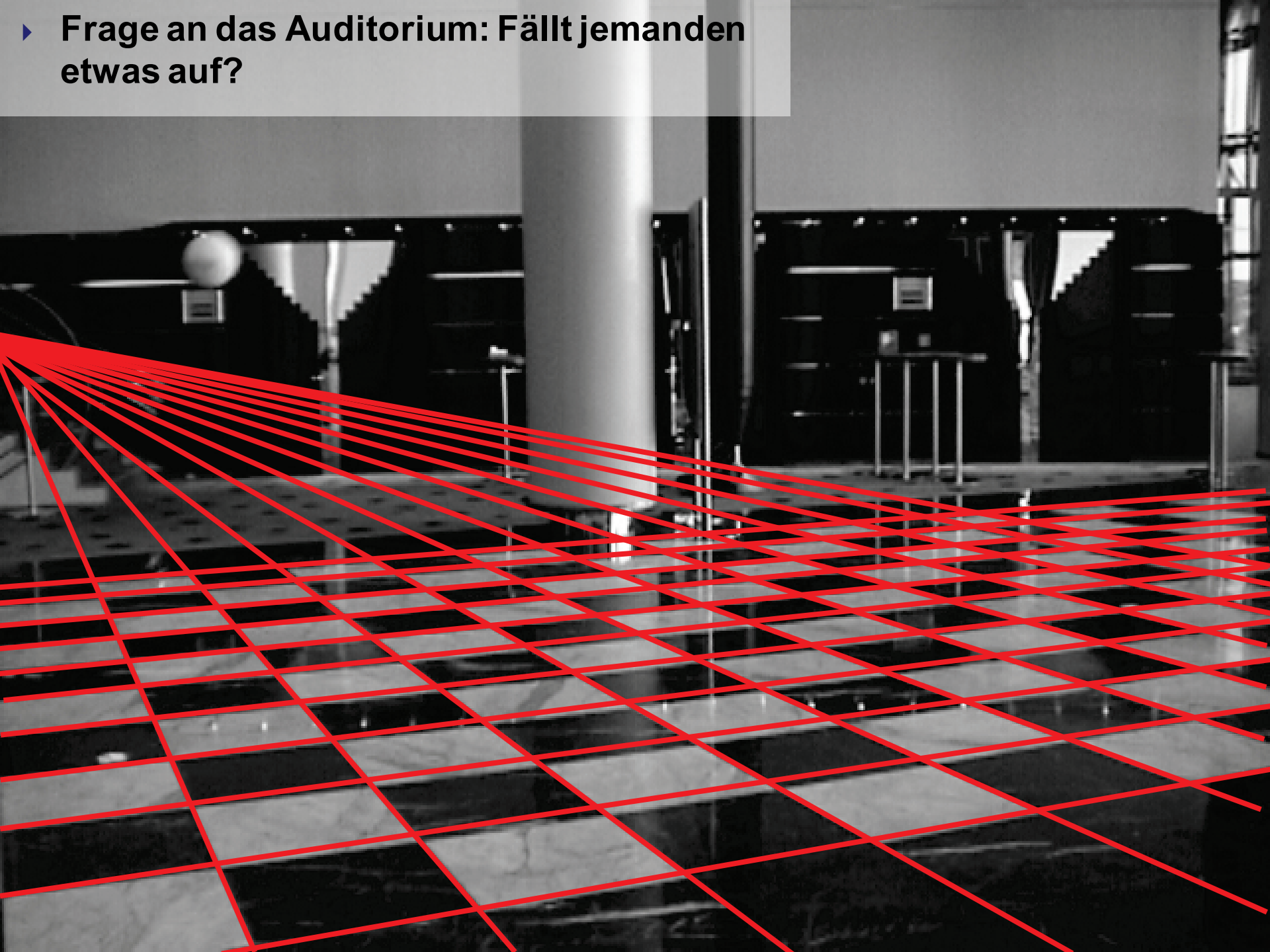
Aufgabe 13:

Kamerapose / Brennweite aus Linien berechnen

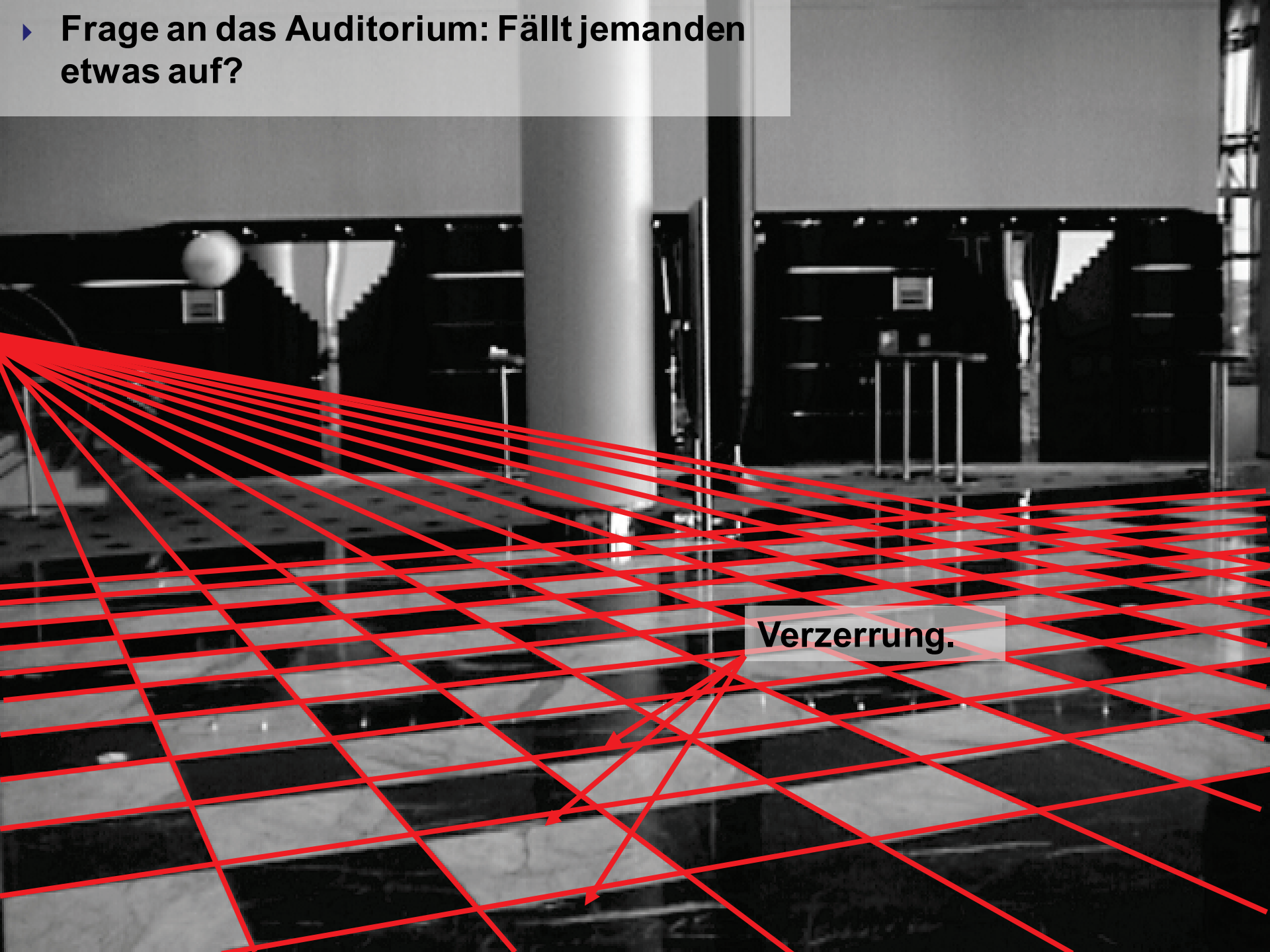
Aufgabe 14:

Ballflugbahn mit Partikel-Filter vorhersagen.

- **Frage an das Auditorium: Fällt jemanden etwas auf?**



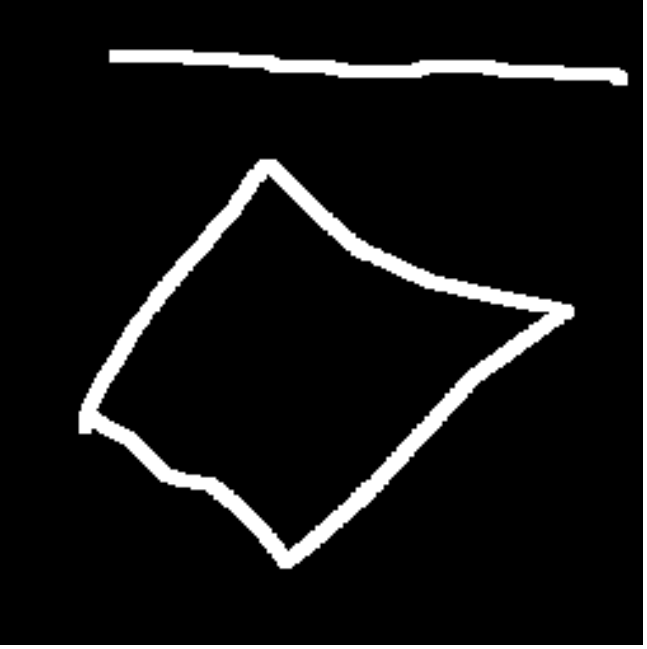
- **Frage an das Auditorium: Fällt jemanden etwas auf?**



Hough-Transformation für Linien

Grundidee

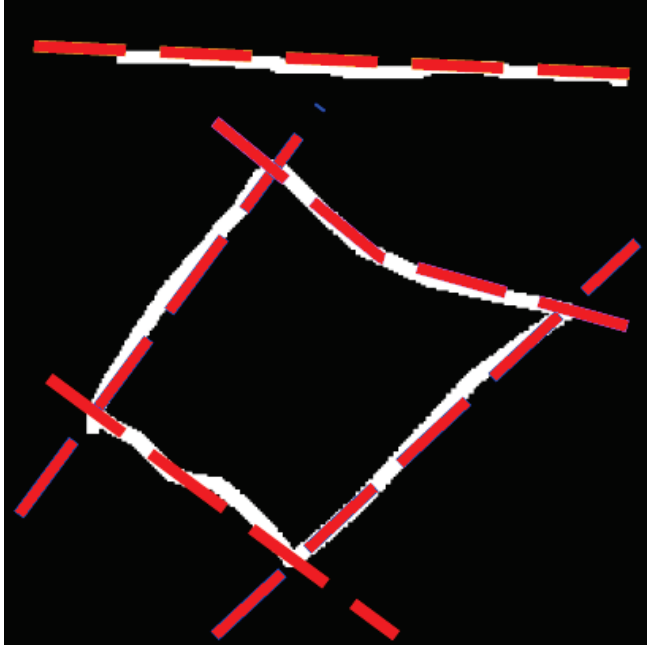
- **Hough-Algorithmus sucht Kurven**
  - in bekannter parametrisierter Form
  - mit unbekannten Parametern
  - für Linien: Hessesche Normalform
- **akkumuliert Evidenz im Parameterraum (Houghraum)**
- ***jeder Kantenpixel ist Evidenz für alle Kurven auf denen er liegt***
  - Houghakkumulator: ein Eintrag pro Parameterkombination
  - Pixel binär (ja/nein) als Kantenpixel klassifizieren (Schwellwert auf Sobellänge)
  - für jeden Kantenpixel alle Akkumulatoreinträge (um 1) erhöhen, deren Kurve durch diesen Pixel geht.
- **Einträge wirklich vorhandener Kurven akkumulieren einem hohen Wert, weil sie von vielen Kantenpixeln erhöht werden**

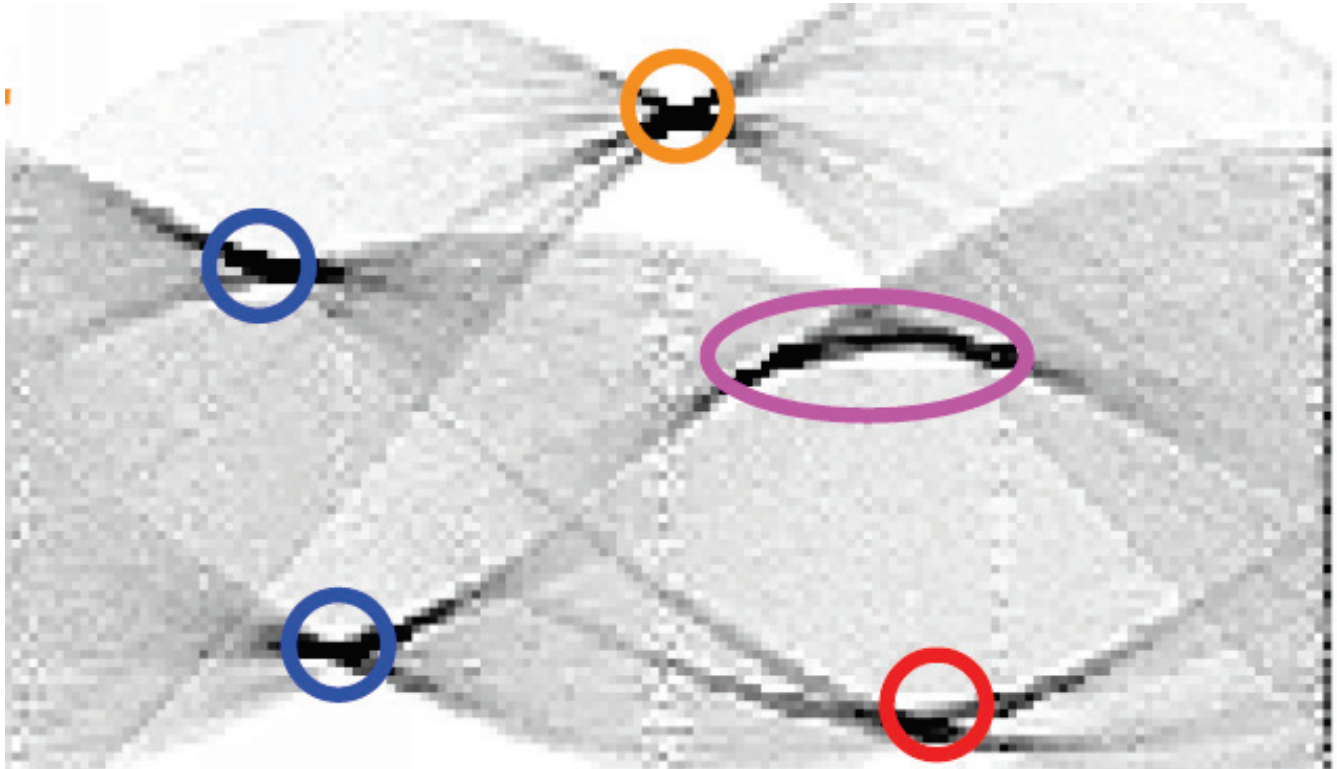


Eingabe (Bildraum)

Ausgabe (Bildraum)

Houghraum

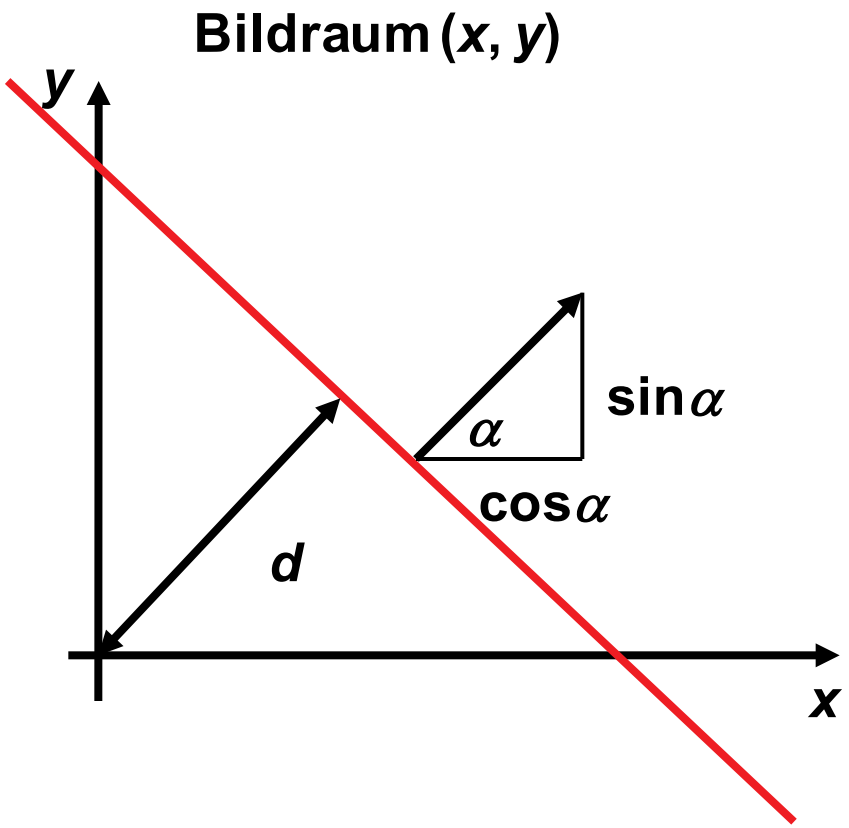




Quelle:
H. Neumann, Vorlesung Neuro-informatik (www.informatik.uni-ulm.de/ni/Lehre/SS03/CV1/)

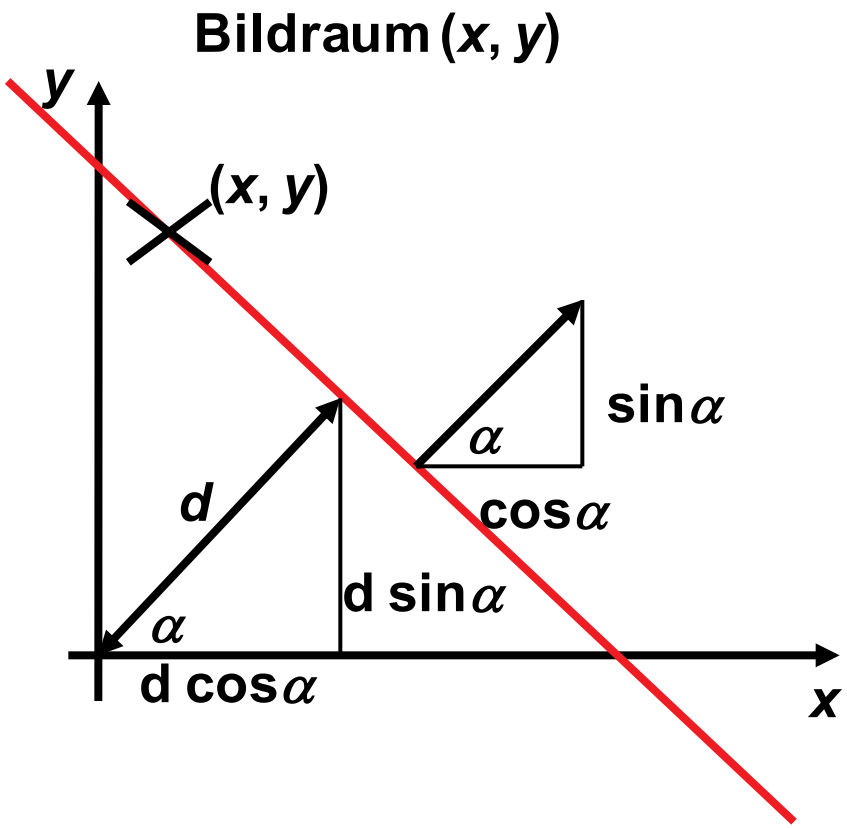
Hough-Transformation für Linien

Parametrisierung einer Geraden in Hessescher Normalform (d, α)



Hough-Transformation für Linien

Parametrisierung einer Geraden in Hessescher Normalform (d, α)



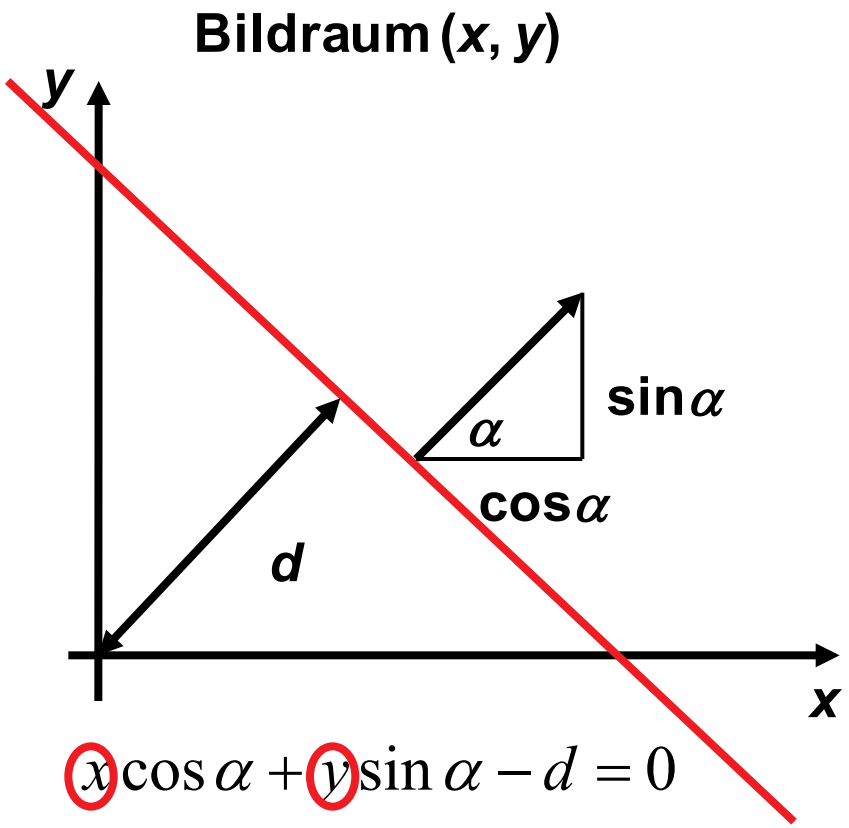
$$\left(\begin{pmatrix} x \\ y \end{pmatrix} - d\begin{pmatrix} \cos\alpha \\ \sin\alpha \end{pmatrix}\right) \cdot \begin{pmatrix} \cos\alpha \\ \sin\alpha \end{pmatrix} = 0$$

$$\begin{pmatrix} x \\ y \end{pmatrix} \cdot \begin{pmatrix} \cos\alpha \\ \sin\alpha \end{pmatrix} - d\begin{pmatrix} \cos\alpha \\ \sin\alpha \end{pmatrix}^2 = 0$$

$$x\cos\alpha + y\sin\alpha - d = 0$$

Hough-Transformation für Linien

Parametrisierung einer Geraden in Hessescher Normalform (d, α)

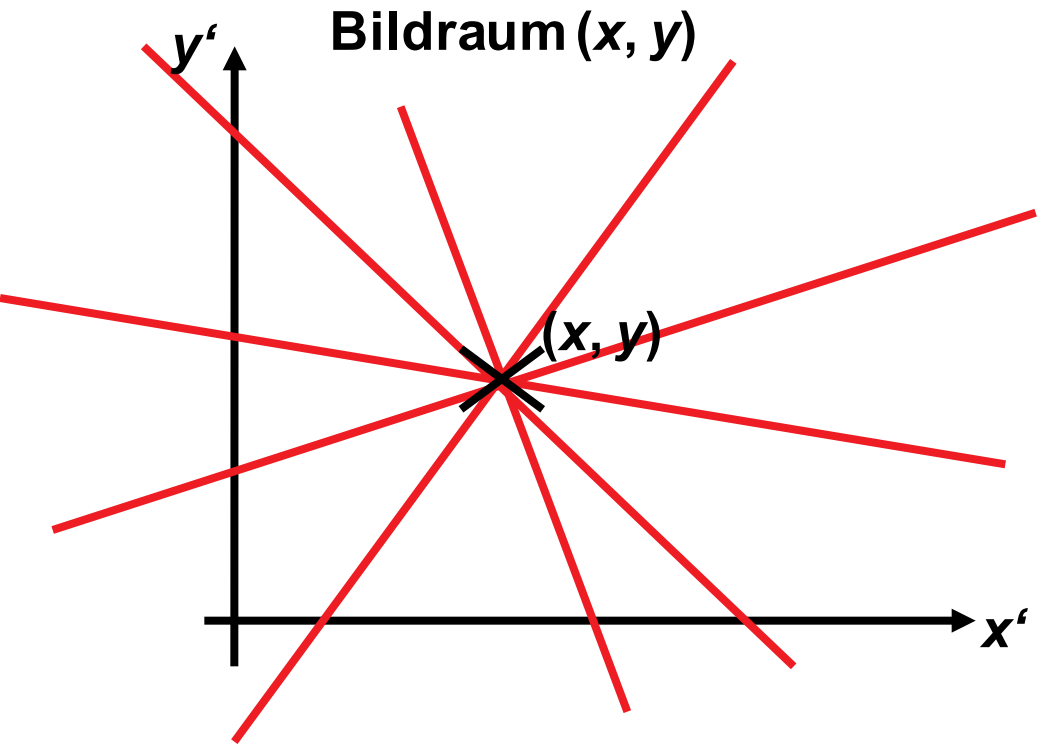


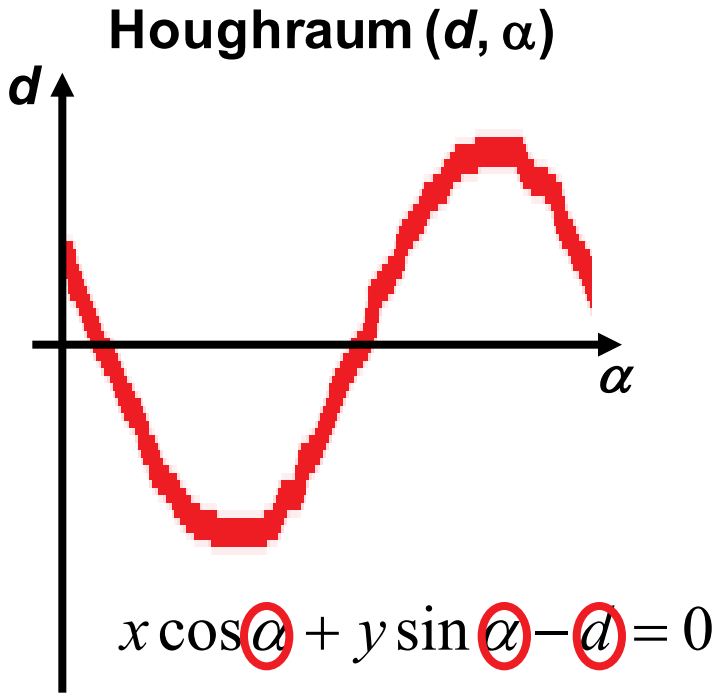
$$x\cos\alpha + y\sin\alpha - d = 0$$

Hough-Transformation für Linien

Parametrisierung einer Geraden in Hessescher Normalform (d, α)

- **Alle Geraden, die durch einen Punkt (x, y) durchgehen.**
  - Im Bildraum ein „Stern“
  - Im Houghraum eine Sinusfunktion

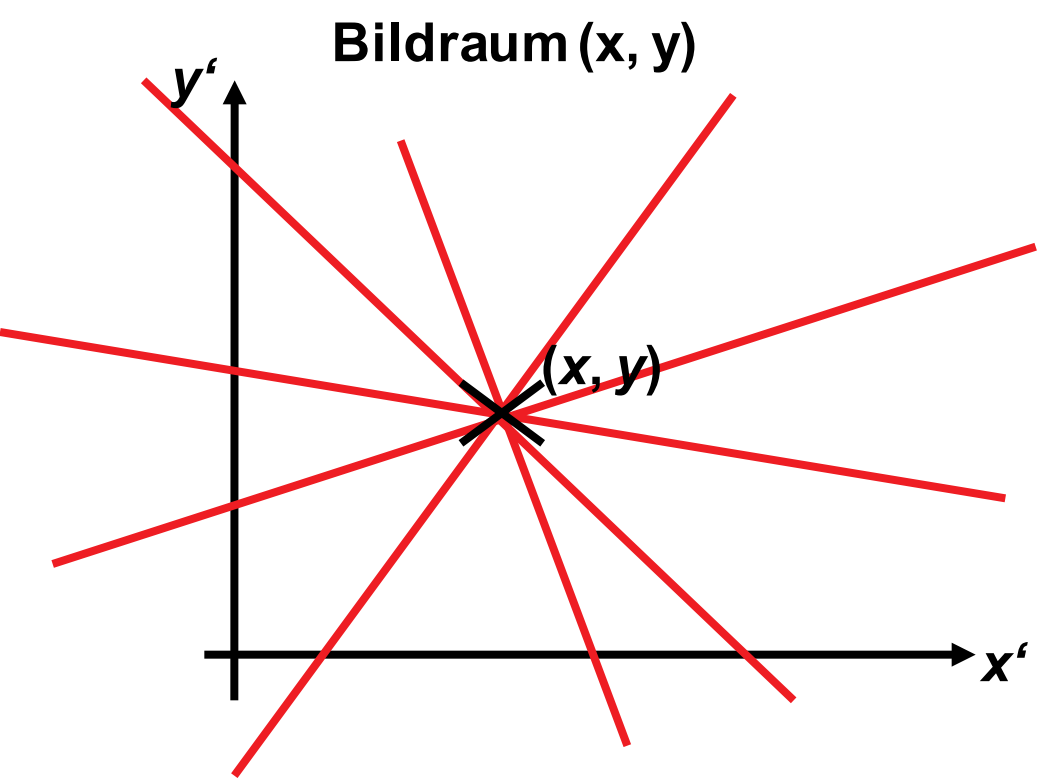


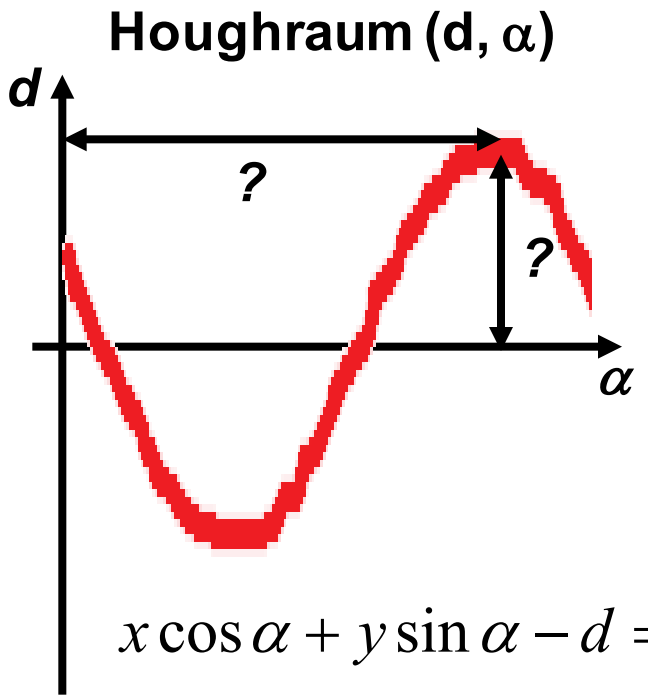


Hough-Transformation für Linien

Parametrisierung einer Geraden in Hessescher Normalform (d, α)

- **Frage an das Auditorium: Wie hängen die Parameter der Sinusfunktion im Houghraum von dem Punkt (x, y) ab?**



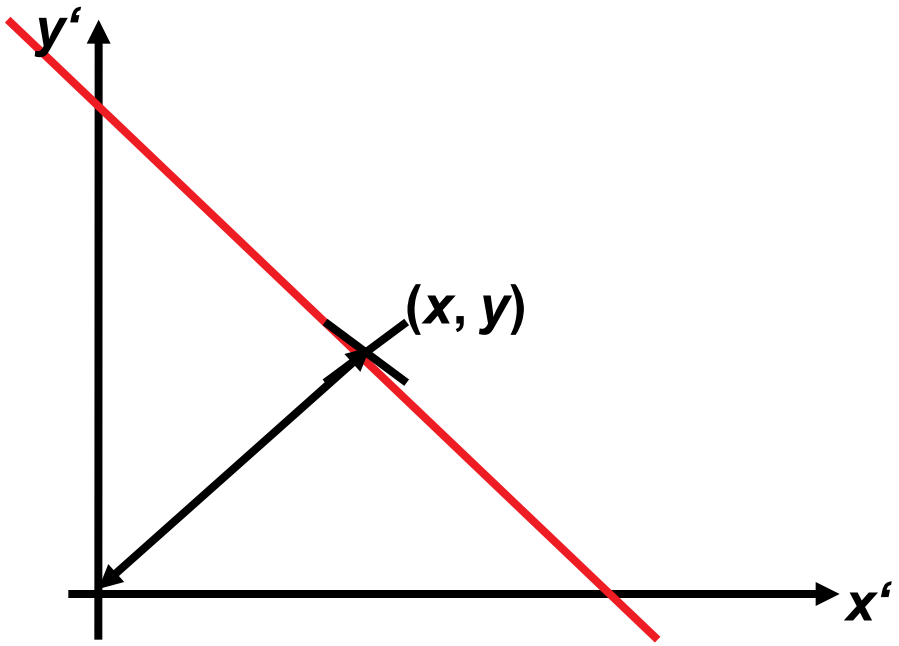


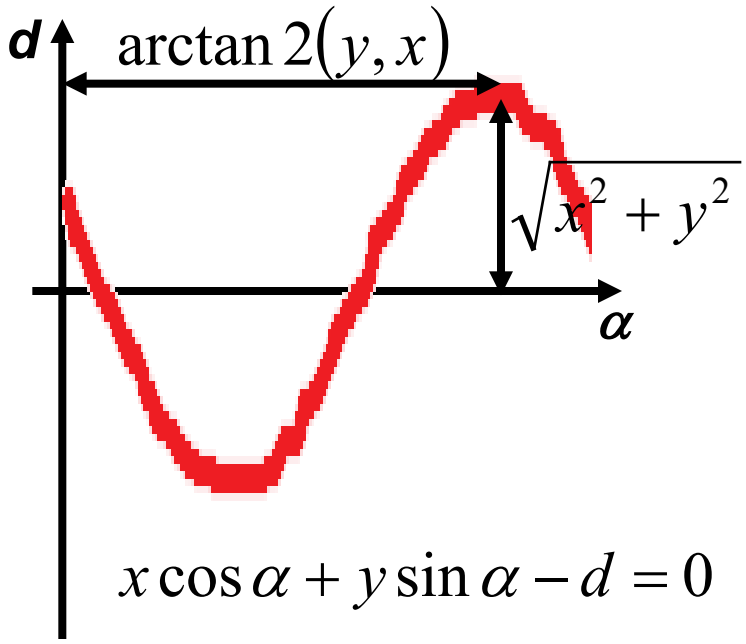
$$x\cos\alpha + y\sin\alpha - d = 0$$

Hough-Transformation für Linien

Parametrisierung einer Geraden in Hessescher Normalform (d, α)

- **Der maximale Abstand d wird erreicht, wenn (x, y) das Lot von (0, 0) auf die Geraden ist. Dies ist dann der Fall, wenn die Gerade senkrecht auf (0, 0) – (x, y) steht.**





Hough-Transformation für Linien

Hough-Transformation für Linien (Rohfassung)

```
addPointToHoughAccumulator (houghImg, x, y, sobelX, sobelY) {
   for (alpha=0;alpha<PI;alpha+=alphaStep) {
      d = x*cos(alpha) + y*sin(alpha);
      houghImg (alpha, d) += 1;
   }
}

lineHough (sobelImg, houghImg, sobelThreshold) {
   for (y=0;y<srcImg.height;y++)
      for (x=0;x<srcImg.width;x++) {
         sobelLen = sqrt(sobelImg(x, y).sobelX² + sobelImg(x, y).sobelY²);
         if (sobelLen>sobelThreshold)
            addPointToHoughAccumulator (houghImg, x, y, sobelX, sobelY);
      }
}
```

$$x\cos\alpha + y\sin\alpha - d = 0$$

Hough-Transformation für Linien

Hough-Transformation für Linien (Rohfassung)

- **Sehr langsam und noch unpräzise formuliert**

```
addPointToHoughAccumulator (houghImg, x, y, sobelX, sobelY) {
   for (alpha=0;alpha<PI;alpha+=alphaStep) {
      d = x*cos(alpha) + y*sin(alpha);
      houghImg (alpha, d) += 1;
   }
}

lineHough (sobelImg, houghImg, sobelThreshold) {
   for (y=0;y<srcImg.height;y++)
      for (x=0;x<srcImg.width;x++) {
         sobelLen = sqrt(sobelImg(x, y).sobelX² + sobelImg(x, y).sobelY²);
         if (sobelLen>sobelThreshold)
            addPointToHoughAccumulator (houghImg, x, y, sobelX, sobelY);
      }
}
```

$$x\cos\alpha + y\sin\alpha - d = 0$$

`double` als Array-Index

Hough-Transformation für Linien

Struktur des Linienhoughraumes

- **Wie diskretisieren, d.h. auf 2D-Array (Bild) abbilden?**
- **Bildgröße w×h**
- **Winkel α**
  - Periodisch [0..π) (Linien) bzw. [0..2π) Kanten
  - Sonderfälle am Rand oder % benutzen
  - Vorsicht: $\alpha \leftarrow \alpha + \pi$, führt zu $d \leftarrow -d$
  - Konservative Diskretisierung $\Delta\alpha$:
    ein Pixel am Bildrand entspricht $\Delta\alpha$ Winkel
  - Pragmatische Diskretisierung (Übungen):
    Feste Zweierpotenz z.B.: 256

$$\Delta\alpha = 1 \Big/ \sqrt{w^2 + h^2}$$

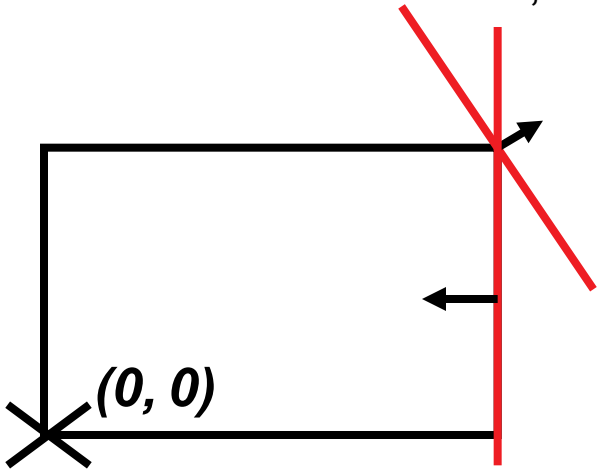
$$\Delta\alpha = \frac{\pi}{256}$$

Hough-Transformation für Linien

Struktur des Linienhoughraumes

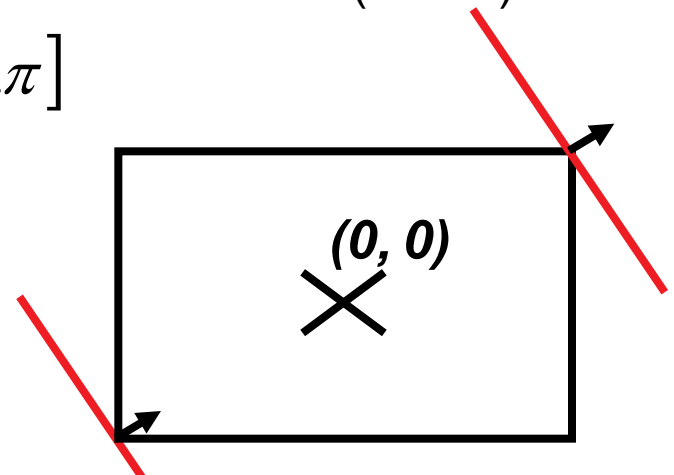
- **Distanz *d*:**
  - Einheit Pixel, natürliche Diskretisierung 1
  - Großer Wertebereich (links)
  - Kleinerer Wertebereich, wenn Referenzpunkt für d in Bildmitte (rechts)

$$\alpha \in [0..\pi]$$



$$d \in \left[-w..\sqrt{w^2+h^2}\right]$$

$$x\cos\alpha + y\sin\alpha - d = 0$$

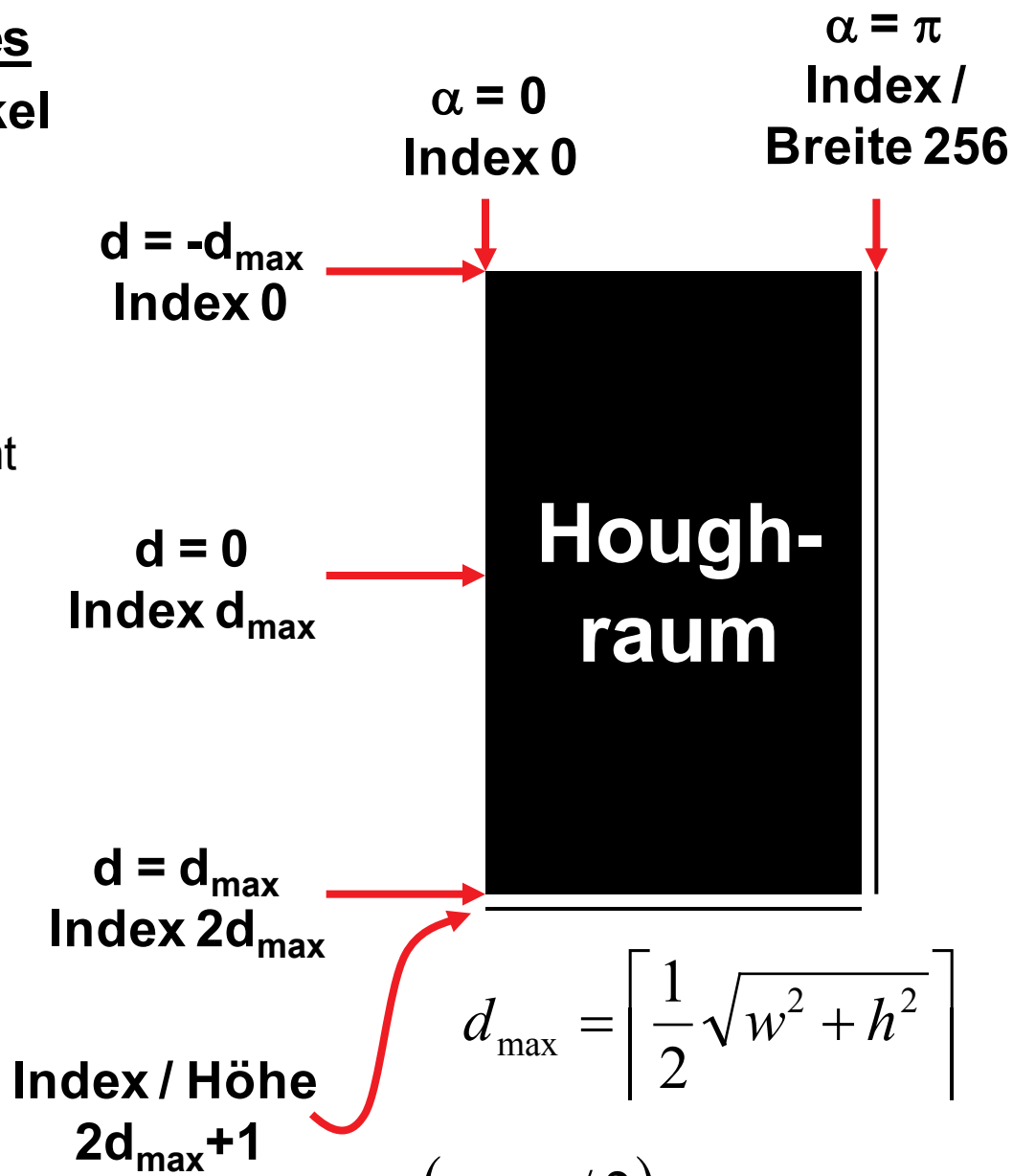


$$d \in \left[-\tfrac{1}{2}\sqrt{w^2+h^2}..\tfrac{1}{2}\sqrt{w^2+h^2}\right]$$

$$(x - w/2)\cos\alpha + (y - h/2)\sin\alpha - d = 0$$

Struktur des Linienhoughraumes

- **für Linienextraktion, 256 Winkel**
- **Winkel α**
  - α von 0 bis π (excl.)
  - Houghraum Breite 256
  - Indizes 0 bis 255 (incl.)
  - 256 entspricht π (excl.) äquivalent zu 0 (wrap around)
- **Distanz d**
  - d Bezug auf Bildmitte (w/2,h/2)
  - Diskretisiert 1 (1 Pixel d-Distanz Bildraum = 1 Pixel Houghraum)
  - d von $-d_{max}$ bis $+d_{max}$ (incl.)
  - Indices 0 bis $2d_{max}$ (incl.)
  - Houghraumhöhe $2d_{max}+1$

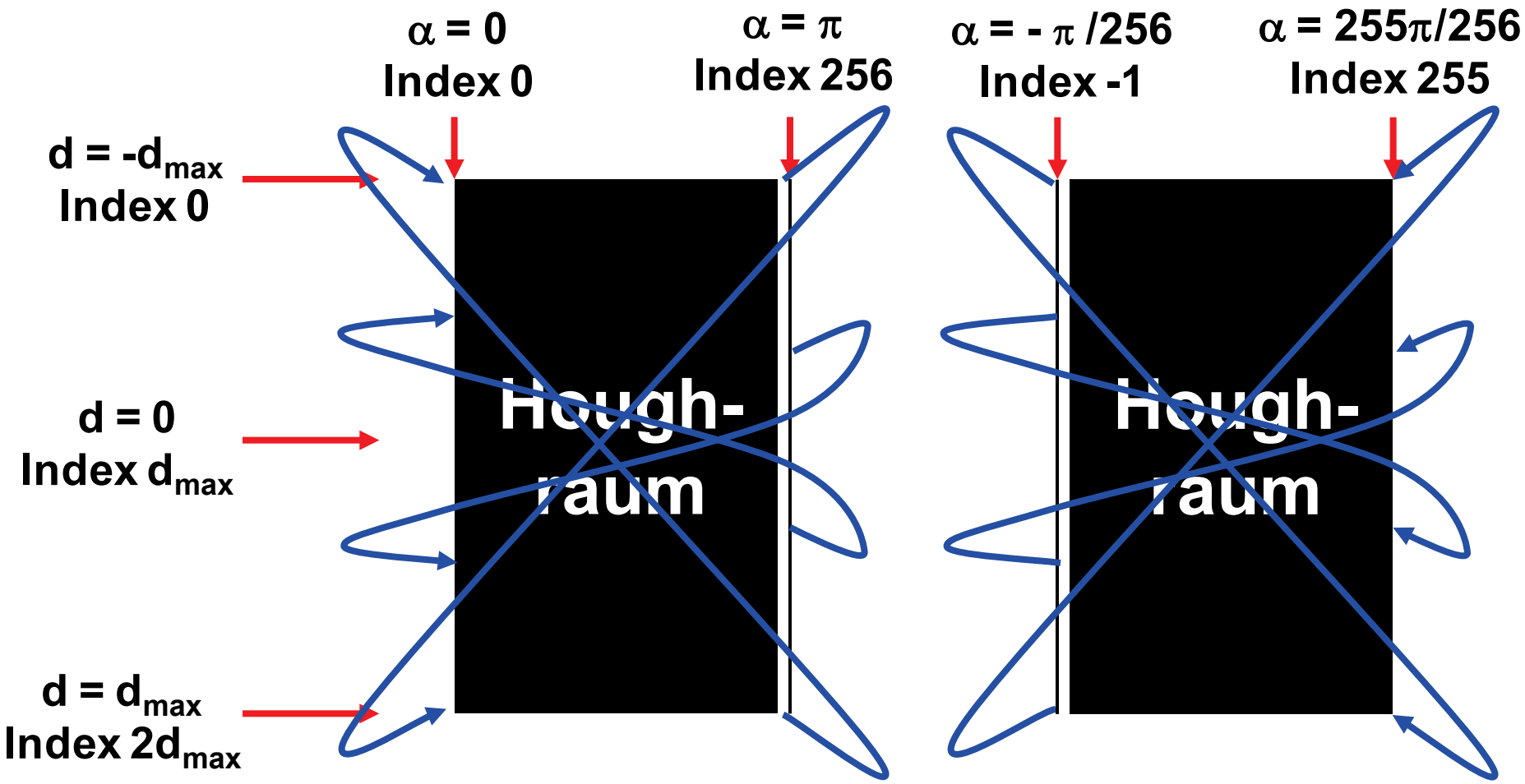


$$d_{\max} = \left\lceil \frac{1}{2}\sqrt{w^2 + h^2} \right\rceil$$

$$(x - w/2)\cos\alpha + (y - h/2)\sin\alpha - d = 0$$

Struktur des Linienhoughraumes

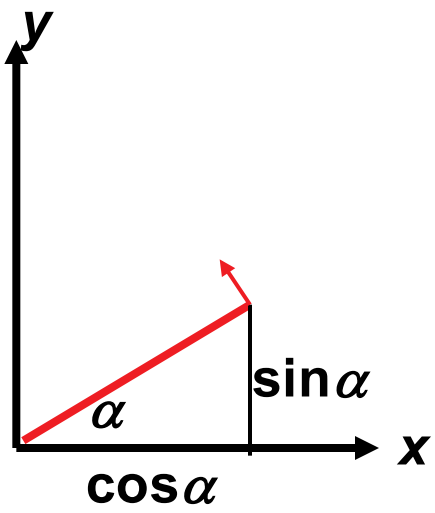
- α-wrap around mit Vorzeichenwechsel bei d
- (π, d) entspricht $(0, -d)$
- Index $(256, d_{idx})$ entspricht $(0, 2d_{max}-d_{idx})$
- Index $(-1, d_{idx})$ entspricht $(255, 2d_{max}-d_{idx})$

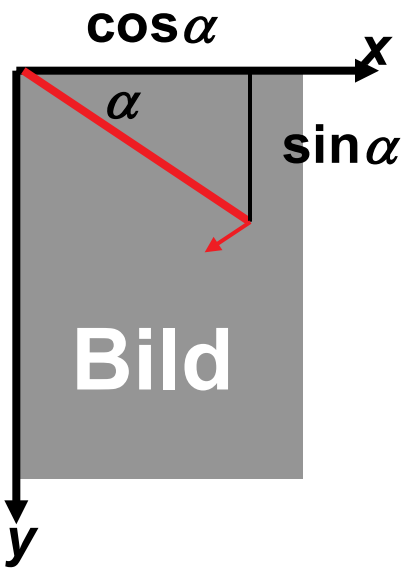


Hough-Transformation für Linien

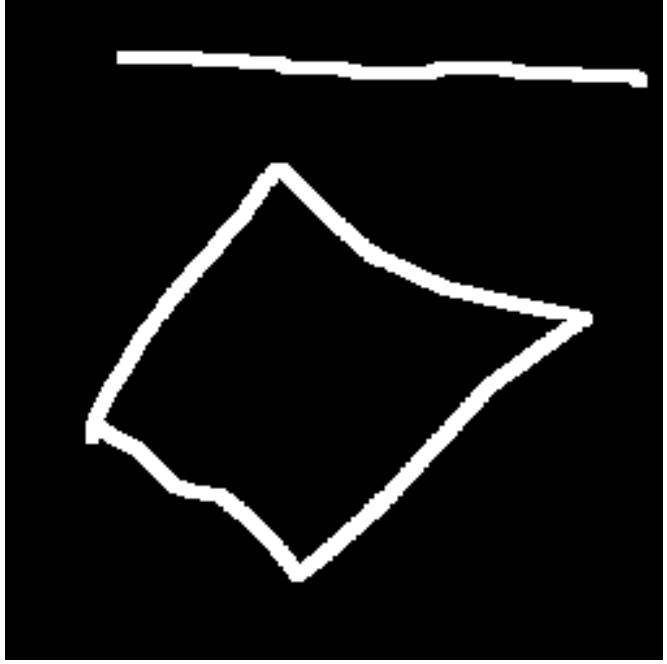
Struktur des Linienhoughraumes

- **Bildkoordinatensystem vs. geometrisches Koordinatensystem**
- **Bildkoordinaten: x zeigt nach rechts, y nach unten**
  - historisch Scanfolge des Elektronenstrahls im Fernseher
  - historisch vermutlich durch Schrift motiviert
- **Linkssystem (Daumen / Zeigerfinger) statt üblichen Rechtssystems**
  - gespiegelt
  - Winkel im Uhrzeigersinn
  - Flächenformeln wechseln Vorzeichen
- **Konsistent behandeln**
  - Herleitung im Rechtssystem
  - umrechnen bei Wechsel Bild- / Weltkoordinaten (Pixel -> mm)





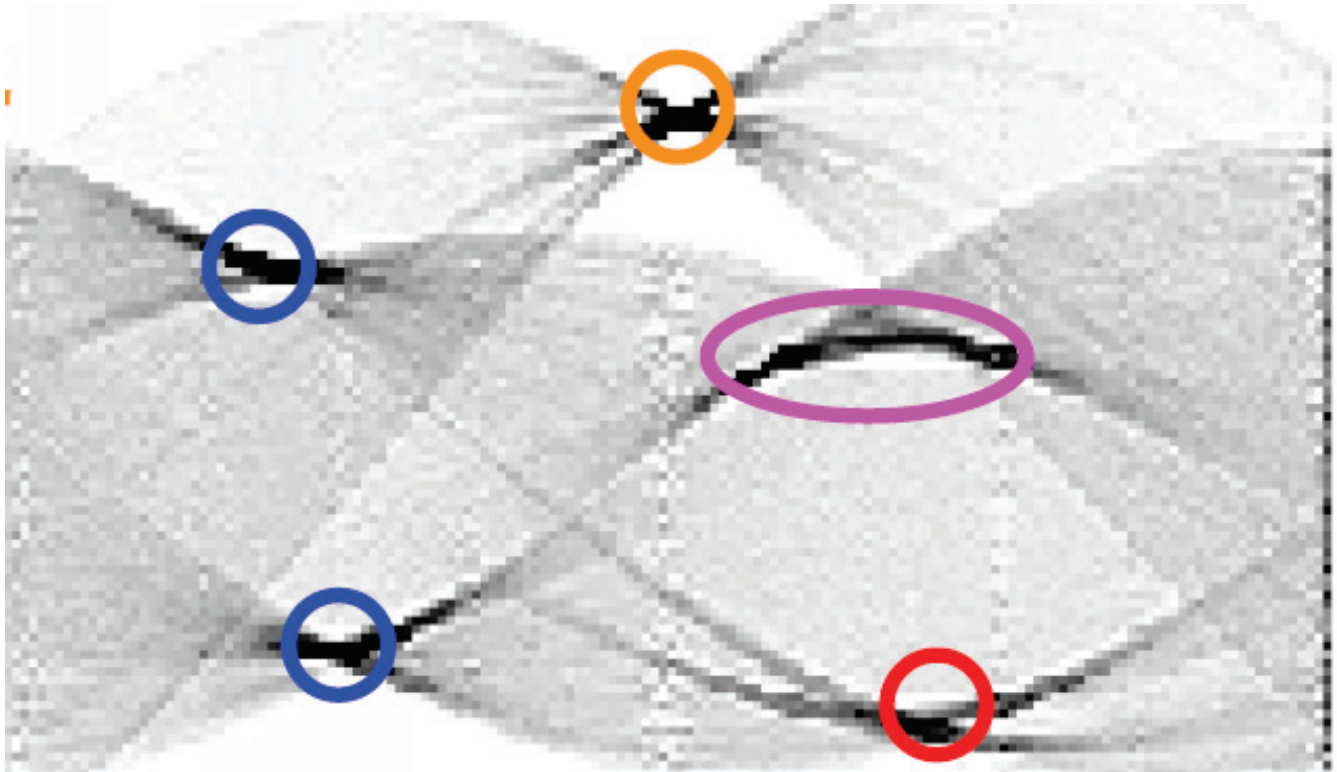
- **<u>Frage an das Auditorium</u>: Welches Maximum im Houghraum gehört zu welcher Linie?**



$\alpha = 0$

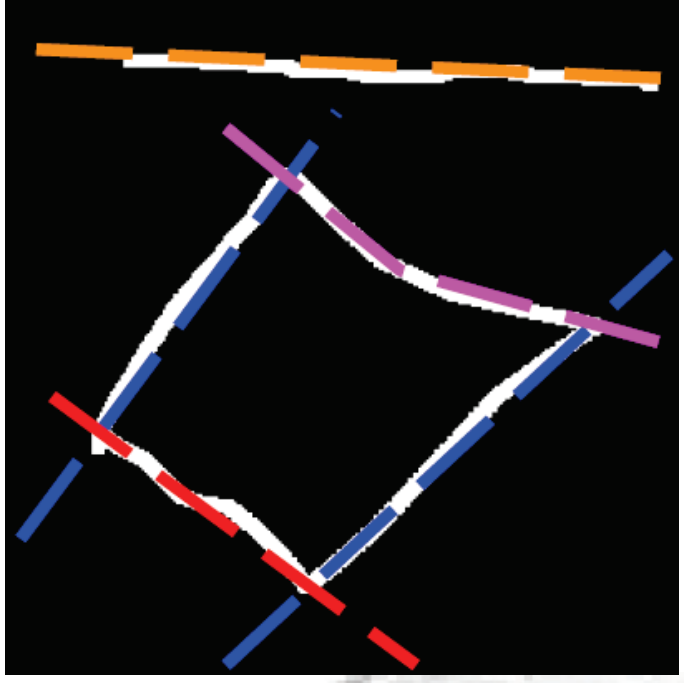
$\alpha = \pi$

$d = -d_{max}$



$d = d_{max}$

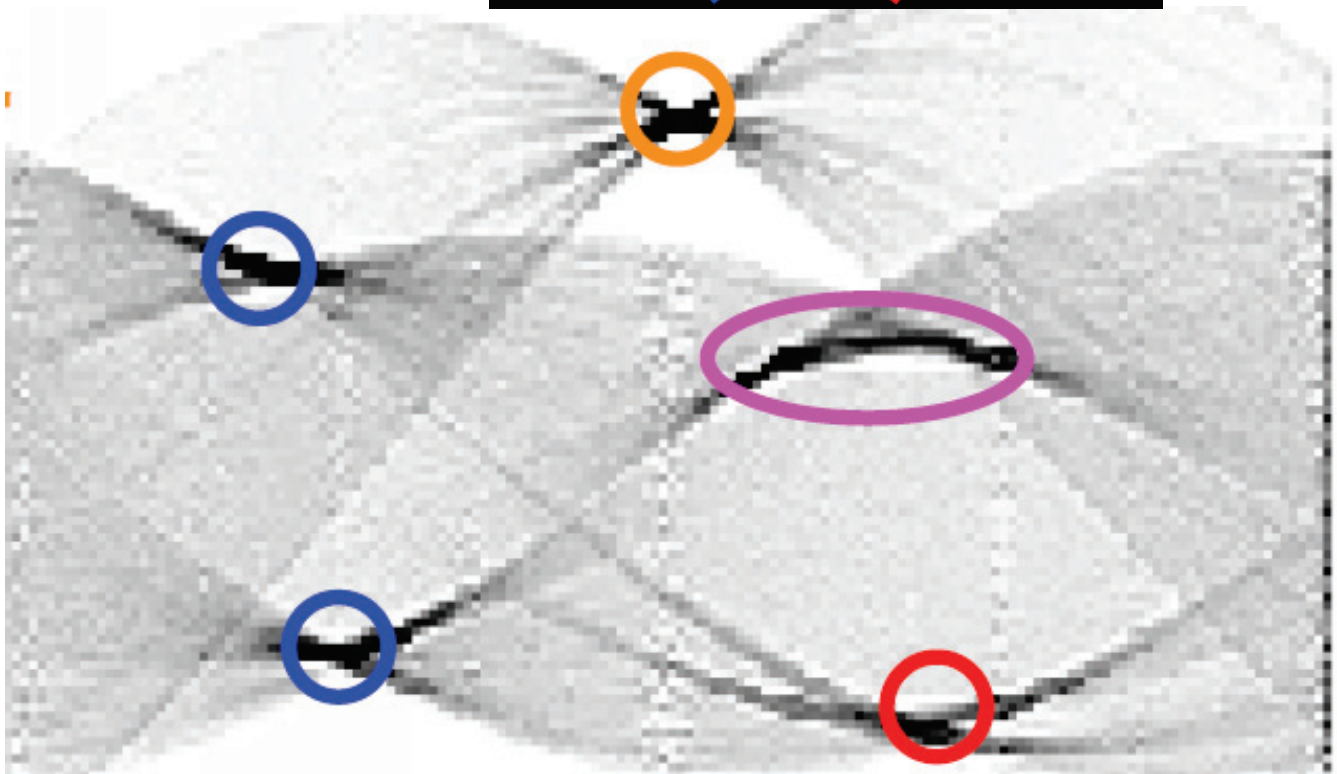
- **<u>Frage an das Auditorium</u>: Welches Maximum im Houghraum gehört zu welcher Linie?**



$\alpha = 0$

$\alpha = \pi$

$d = -d_{max}$



$d = d_{max}$

Hough-Transformation für Linien

Hough-Transformation Linien (1. optimierte Fassung)

- langsam weil alle `alphaIdx` durchlaufen werden

```
addPointToHoughAccumulator (houghImg, x, y, sobelX, sobelY) {
   for (alphaIdx=0;alphaIdx<NR_OF_ORIENTATIONS;alphaIdx++) {
      alpha = alphaIdx*PI/NR_OF_ORIENTATIONS;
      d = (x-width/2)*cos(alpha) + (y-height/2)*sin(alpha);
      dIdx = houghImg->height/2 + (int) d;
      houghImg (alphaIdx, dIdx) += 1;
   }
}
```

Hough-Transformation für Linien

Hough-Transformation Linien (1. optimierte Fassung)

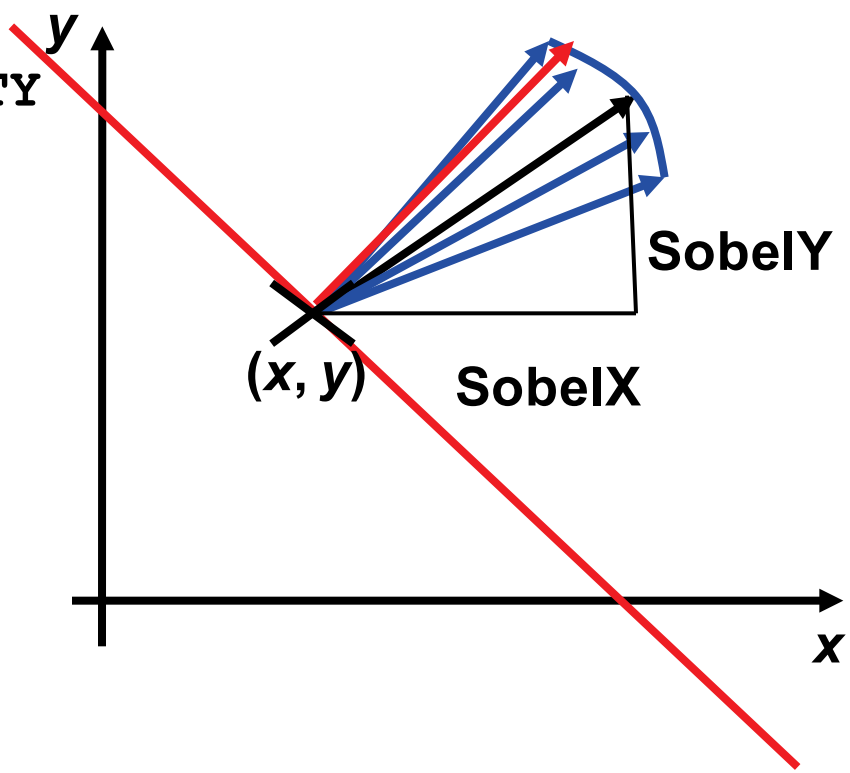
- **langsam weil alle `alphaIdx` durchlaufen werden**
- **Frage an das Auditorium:**
  Wie könnte man die Routine beschleunigen?

```
addPointToHoughAccumulator (houghImg, x, y, sobelX, sobelY) {
   for (alphaIdx=0;alphaIdx<NR_OF_ORIENTATIONS;alphaIdx++) {
      alpha = alphaIdx*PI/NR_OF_ORIENTATIONS;
      d = (x-width/2)*cos(alpha) + (y-height/2)*sin(alpha);
      dIdx = houghImg->height/2 + (int) d;
      houghImg (alphaIdx, dIdx) += 1;
   }
}
```

Hough-Transformation für Linien

Ausnutzen der Sobelrichtung

- Idee: Sobelvektors ist grob die Richtung der Geraden
- akkumuliere nur in Winkelbereich um Richtung des Sobelvektors.
- um die Richtung des Sobelvektors (schwarz) +/- `ANGLE_UNCERTAINTY` diskrete Winkel in den Houghraum eintragen (blau)
- echter Normalenwinkel (rot) sollte darunter sein
- Viel schneller, da nur `2*ANGLE_UNCERTAINTY+1` Einträge (z.B. 5 statt 256)
- weniger Überlagerung verschiedener Geraden



Hough-Transformation für Linien

Intervalle in periodischen Räumen ablaufen

- **umgangssprachlich „wrap-around“**
- **Intervalle in nichtperiodischen Räumen bekannt:**

```
for(int i=lo; i<=hi; i++) {
      // do something with i
}
```

- **für Periodizität N (`NR_OF_ORIENTATIONS`) gilt:**
  - i und i+N oder i-N sind äquivalent
  - nur normalisierte Indices [0..N-1] im Array vorhanden
- **Normalisieren (int) mit % (Vorsicht bei negativen Zahlen):**

```
iNorm = i % N;
if (iNorm<0) iNorm+=N;
```

- **Normalisieren (int) mit & (bei N=2^p)**

```
iNorm = i & (N-1);
```

- **Normalisieren (double) mit floor**

```
fNorm = f - N*floor(f/N);
```

Hough-Transformation für Linien

Intervalle in periodischen Räumen ablaufen

- **Lösung: i nichtperiodisch durchlaufen und normalisieren**

```
for(int i=lo; i<=hi; i++) {
      iN = i % N;
      if (iN<0) iN+=N;
      // do something with iN
}
```

- **schneller: vorher normalisieren und „wrap-around“ in Schleife**

```
lo = lo     % N; if (lo<0) lo+=N;
hi = (hi+1) % N;   if (hi<=0) hi+=N;
for(int i=lo; i!=hi; i++) {
      if (i==N) i=0;
      // do something with i
}
```

- **Vorsicht: hier ist `hi` exclusiv!**

Hough-Transformation für Linien

Hough-Transformation Linien (2. optimierte Fassung)

```
addPointToHoughAccumulator (houghImg, x, y, sobelX, sobelY) {
   sobelAngle = atan2(sobelY, sobelX);
   alphaLo = (int) (sobelAngle/PI*NR_OF_ORIENTATIONS);
   alphaLo-= ANGLE_UNCERTAINTY;
   if (alphaLo<0) alphaLo += NR_OF_ORIENTATIONS;
   if (alphaLo<0) alphaLo += NR_OF_ORIENTATIONS;
   alphaHi = alphaLo + 2*ANGLE_UNCERTAINTY +1;
   if (alphaHi>NR_OF_ORIENTATIONS) alphaHi-=NR_OF_ORIENTATIONS;
   for (alphaIdx=alphaLo;alphaIdx!=alphaHi;alphaIdx++) {
      if (alphaIdx==NR_OF_ORIENTATIONS) alphaIdx = 0;
      alpha = alphaIdx*PI/NR_OF_ORIENTATIONS;
      d = (x-width/2)*cos(alpha) + (y-height/2)*sin(alpha);
      dIdx = houghImg->height/2 + (int) d;
      houghImg (alphaIdx, dIdx) += 1;
}  }
```

Hough-Transformation für Linien

Hough-Transformation Linien (2. optimierte Fassung)

```
addPointToHoughAccumulator (houghImg, x, y, sobelX, sobelY) {
   sobelAngle = atan2(sobelY, sobelX);
   alphaLo = (int) (sobelAngle/PI*NR_OF_ORIENTATIONS);
   alphaLo-= ANGLE_UNCERTAINTY;
   if (alphaLo<0) alphaLo += NR_OF_ORIENTATIONS;
   if (alphaLo<0) alphaLo += NR_OF_ORIENTATIONS;
   alphaHi = alphaLo + 2*ANGLE_UNCERTAINTY +1;
   if (alphaHi>NR_OF_ORIENTATIONS) alphaHi-=NR_OF_ORIENTATIONS;
   for (alphaIdx=alphaLo;alphaIdx!=alphaHi;alphaIdx++) {
      if (alphaIdx==NR_OF_ORIENTATIONS) alphaIdx = 0;
      alpha = alphaIdx*PI/NR_OF_ORIENTATIONS;
      d = (x-width/2)*cos(alpha) + (y-height/2)*sin(alpha);
      dIdx = houghImg->height/2 + (int) d;
      houghImg (alphaIdx, dIdx) += 1;
} }
```

Aufwändige Fließ-komma-rech-nungen z.B. (sin, cos, atan2)

Effiziente Hough-Transformation

Effizienz durch Tabellen (LUT)

- **für: komplizierte Rechnungen mit wenigen Variablen**
- **Beispiel: Farbklassifikation aus R, G, B**
- **Idee: Tabelle mit Ergebnis für jede Kombination der Variablen**
- **mehrere Ergebnisse zur selben Variablenkombination möglich**
- **gut, wenn Variablen schon diskret / diskretisiert sind**
- **Zusatzrechnungen in Tabelle integrieren**
  - Diskretisierung
  - Normalisierung
  - Beschränkung des Ergebnisses
  - t.w. Adressberechnungen für z.B. Bildzugriff

Hough-Transformation für Linien

Frage an das Auditorium: Wo könnten hier Tabellen (LUTs) nützen?

```
addPointToHoughAccumulator (houghImg, x, y, sobelX, sobelY) {
   sobelAngle = atan2(sobelY, sobelX);
   alphaLo = (int) (sobelAngle/PI*NR_OF_ORIENTATIONS);
   alphaLo-= ANGLE_UNCERTAINTY;
   if (alphaLo<0) alphaLo += NR_OF_ORIENTATIONS;
   if (alphaLo<0) alphaLo += NR_OF_ORIENTATIONS;
   alphaHi = alphaLo + 2*ANGLE_UNCERTAINTY + 1;
   if (alphaHi>NR_OF_ORIENTATIONS) alphaHi-=NR_OF_ORIENTATIONS;
   for (alphaIdx=alphaLo;alphaIdx!=alphaHi;alphaIdx++) {
      if (alphaIdx==NR_OF_ORIENTATIONS) alphaIdx = 0;
      alpha = alphaIdx*PI/NR_OF_ORIENTATIONS;
      d = (x-width/2)*cos(alpha) + (y-height/2)*sin(alpha);
      dIdx = houghImg->height/2 + (int) d;
      houghImg (alphaIdx, dIdx) += 1;
   }
}
```

LUT 1: `(sobelX, sobelY) → (sobelLen, alphaLo, alphaHi)`

```
addPointToHoughAccumulator (houghImg, x, y, sobelX, sobelY) {
   sobelAngle = atan2(sobelY, sobelX);
   alphaLo = (int) (sobelAngle/PI*NR_OF_ORIENTATIONS);
   alphaLo-= ANGLE_UNCERTAINTY;
   if (alphaLo<0) alphaLo += NR_OF_ORIENTATIONS;
   if (alphaLo<0) alphaLo += NR_OF_ORIENTATIONS;
   alphaHi = alphaLo + 2*ANGLE_UNCERTAINTY + 1;
   if (alphaHi>NR_OF_ORIENTATIONS) alphaHi-=NR_OF_ORIENTATIONS;
   for (alphaIdx=alphaLo;alphaIdx!=alphaHi;alphaIdx++) {
      if (alphaIdx==NR_OF_ORIENTATIONS) alphaIdx = 0;
      alpha = alphaIdx*PI/NR_OF_ORIENTATIONS;
      d = (x-width/2)*cos(alpha) + (y-height/2)*sin(alpha);
      dIdx = houghImg->height/2 + (int) d;
      houghImg (alphaIdx, dIdx) += 1;
   }
}
```

LUT 2: `(alphaIdx) → (cos(alpha), sin(alpha))`

Hough-Transformation für Linien

Implementierung der LUTs

- **LUT 1: `(sobelX, sobelY)→ (sobelLen, alphaLo, alphaHi)`**
  - Tabelle (vector) von Objekten, nicht mehrere Tabellen
  - Wertebereich für `sobelX`, `sobelY`: [-1020..1020]
  - Eine Zeile: 2^{11}, dadurch Multiplikation `<<11`, Offset `1024`
  - Zugriff: `sobelTab[((sobelY+1024)<<11) + (sobelX+1024)]`
  - `+1024` herausziehen `sobelCenterTab[(sobelY<<11)+sobelX]`
- **LUT 2: `(alphaIdx)→ (cos(alpha), sin(alpha))`**
  - Wertebereich für alpha: [0..255], Zugriff direkt
  - Ergebnis mit 256 skaliert, dadurch `int` Arithmetik
  - ```
    CosSinTabEntry& cse = cosSinTab[alpha];
    d = ((x-width/2)*cse.cos + (y-height/2)*cse.sin) >> 8;
    ```
- **`-width/2, -height/2` aus Schleife ziehen**
- **`houghImg->height/2` herausziehen durch Pointer `houghCenter`**
    ```

Hough-Transformation für Linien

Heraussuchen der Linien aus dem Houghraum

- **Houghraum nach Pixeln über einem Schwellwert durchsuchen**
- **nur akzeptieren, wenn in einer Umgebung maximal**
- **in zurück gelieferten Linien *d* wieder auf (0,0) nicht Bildmitte beziehen. Einfacheres Interface.**
- **Maximalitätstest wird selten aufgerufen, daher nicht effizienzkritisch**
- **Vorsicht: Wraparound in α führt zu Negierung von *d***
  - vgl. Struktur des Houghraumes oben.
  - (π,*d*) entspricht (0,-*d*)
  - Index (256, d_{idx}) entspricht (0, $2d_{max}$-d_{idx})
  - Index (-1,d_{idx}) entspricht (255, $2d_{max}$-d_{idx})

Hough-Transformation für Linien

Zusammenfassung effiziente Implementierung

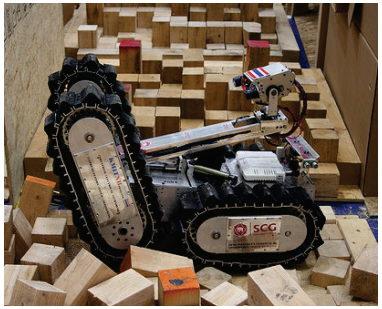
- nur Winkel, die ungefähr der Richtung des Sobel-Vektors entsprechen im Houghraum erhöhen.
- α in 256 Schritte $[0..\pi)$ diskretisieren, Periodizität beachten.
- *d* in Bezug auf Bildmitte um Houghraum klein zu halten.
- Sobel-Betrag und diskretisiertes & normalisiertes Winkel-Intervall für jede SobelX, SobelY Kombination vortabellieren (LUT1).
- $\cos(\alpha)$, $\sin(\alpha)$ für jeden diskretisierten Winkel tabellieren (*256 für Festkommaarithmetik) (LUT2).
- konstante Offsets aus Berechnungen herausziehen.
- Tabellen- / Houghraumdimension als Zweierpotenz, dadurch Multiplikation per <<.
- Derart technisch verwickelte Optimierung sind nur sinnvoll für Teilroutinen die sehr oft ausgeführt werden (z.B. jeden Pixel)

By mid-21st century, a team of fully autonomous humanoid robot soccer players shall win the soccer game, comply with the official rule of the FIFA, against the winner of the most recent World Cup.

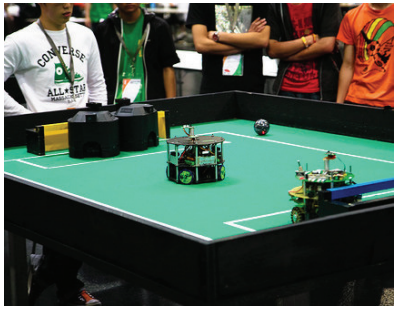
Kitano und Asada, 1998

RoboCup

Soccer



Rescue



Junior

@Home

RoboCup

Soccer

Rescue

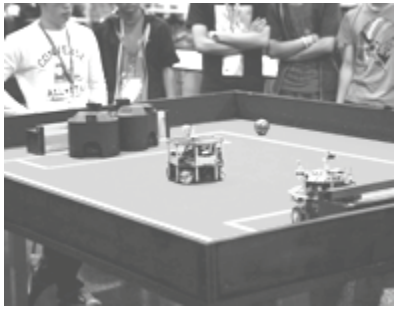
Junior

@Home

RoboCup

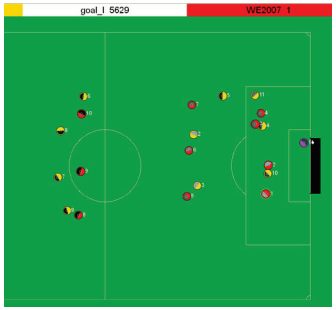
Soccer

Rescue

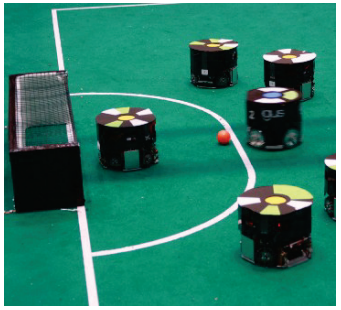


Junior

@Home

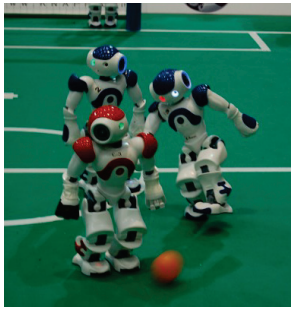


Simulation

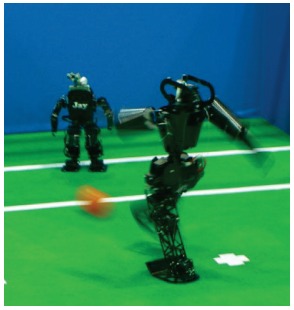


Small Size

Middle Size



SPL

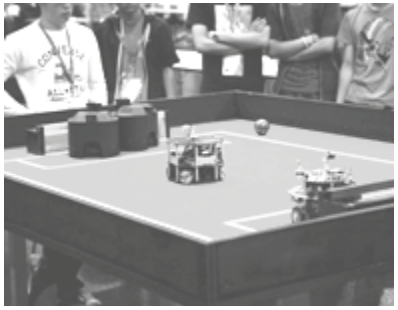


Humanoid

RoboCup

Soccer

Rescue



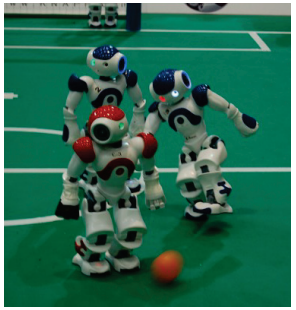
Junior

@Home

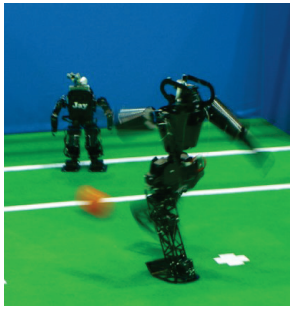
Simulation

Small Size

Middle Size



SPL



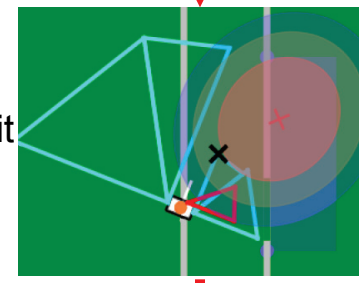
Humanoid

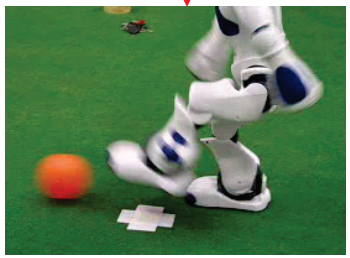
Standard Platform League

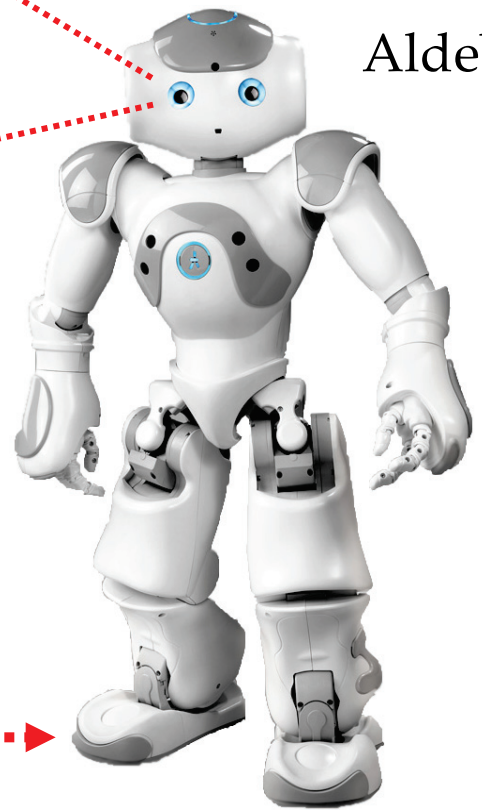
Bildverarbeitung
-320 x 240 @ 30 fps
-Farbsegmentierung
-Feldelemente, Ball, Roboter
-Relative Objektpositionen

Zustandsschätzung
-Eigene absolute Position
-Ballposition- und geschwindigkeit
-Positionen anderer Roboter
-Partikel- und Kalmanfilter

Handeln
-Verhaltensauswahl
-Laufen
-Schießen







Aldebaran Nao

Standard-Platform-League-Feld

- Feld grün
- Linien weiß
- Tor gelb / blau
- Ball orange
- Roboter mit „Trikots“

B-Human

RoboCup Standard Platform League

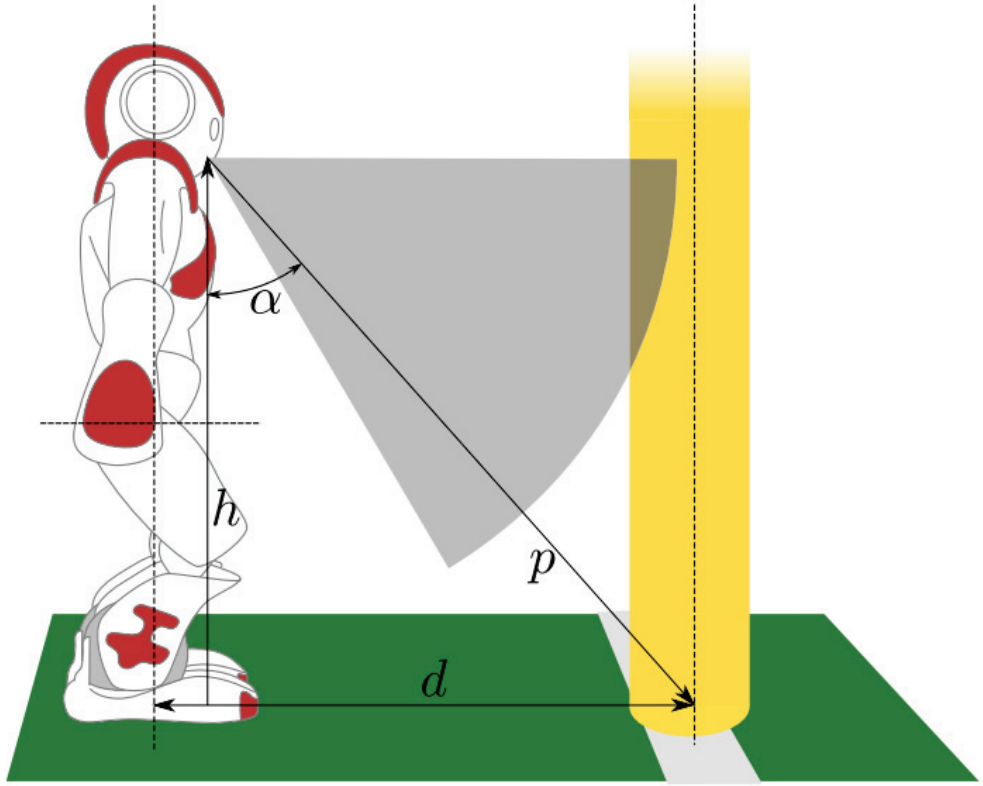
- **DFKI + Universität Bremen**
  - Studentisches Projekt
    - + 3 Wissenschaftler
    - + Diplomanden
  - RoboCup seit 2000
  - B-Human in SPL seit 2008
  - Web: http://www.b-human.de
- **Erfolge**
  - Deutscher Meister 2009 + 2010 + 2011
  - Weltmeister 2009 + 2010 + ???

Was sieht der Roboter?

Video: Thomas Röfer

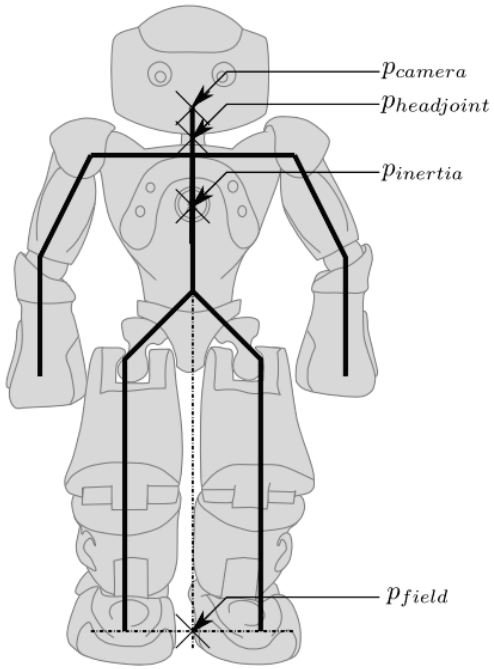
Perspektive

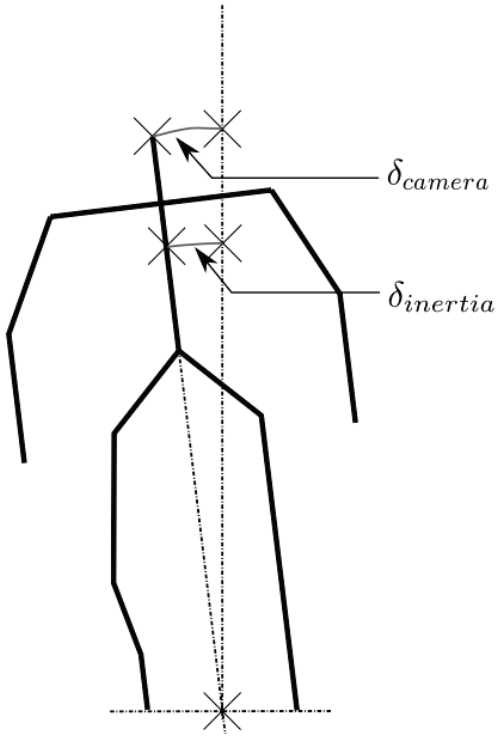
- **Intrinsische Parameter**
  - Brennweite, Verzerrung, …
  - Entfernungsbestimmung über Objektgröße
- **Extrinsische Parameter**
  - 3D-Pose der Kamera
  - Ändert sich durch Roboterbewegung
  - Entfernungsbestimmung über Winkel



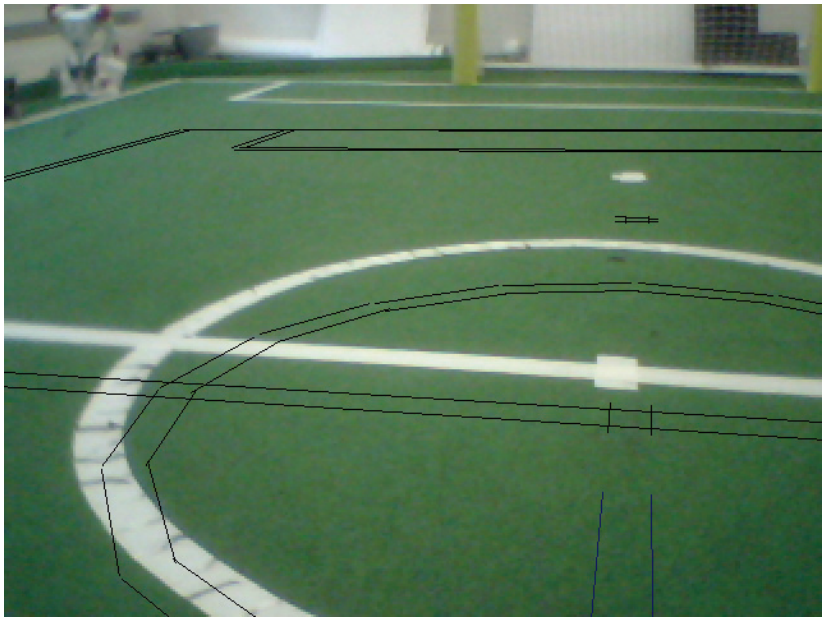
Schätzung der Torso-Lage

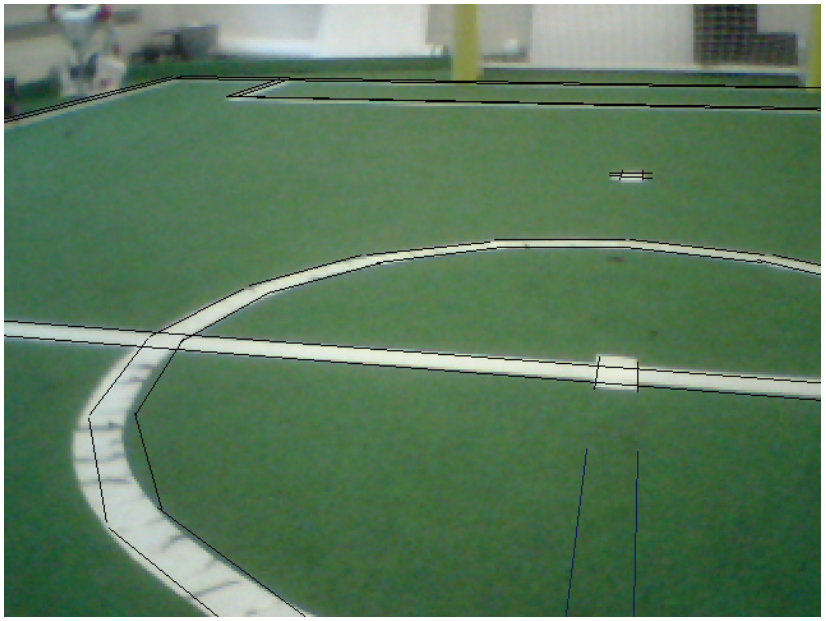
- **Vorwärtskinematik ist unpräzise**
  - Wackeln beim Laufen, Kollisionen, …
  - Winkelfehler haben starke Auswirkungen bei entfernten Objekten
- **Lageschätzung mittels Inertialsensor**





Roboterkalibrierung

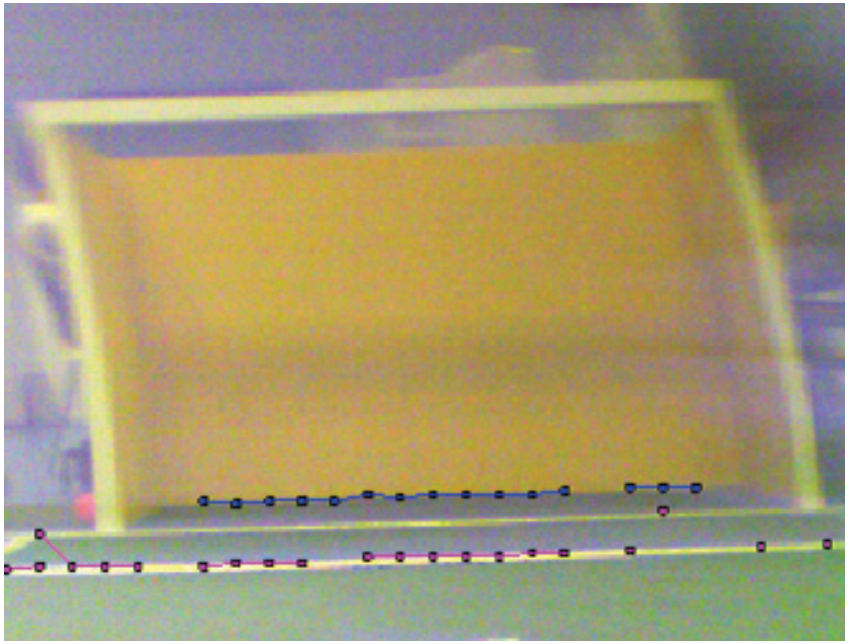




- **Vor der Kalibrierung**
  - Unpräziser Kameraeinbau
  - Spiel in den Fußgelenken
- **Nach der Kalibrierung**
  - Kameraverdrehung und -neigung
  - Gesamtkörperverdrehung und -neigung

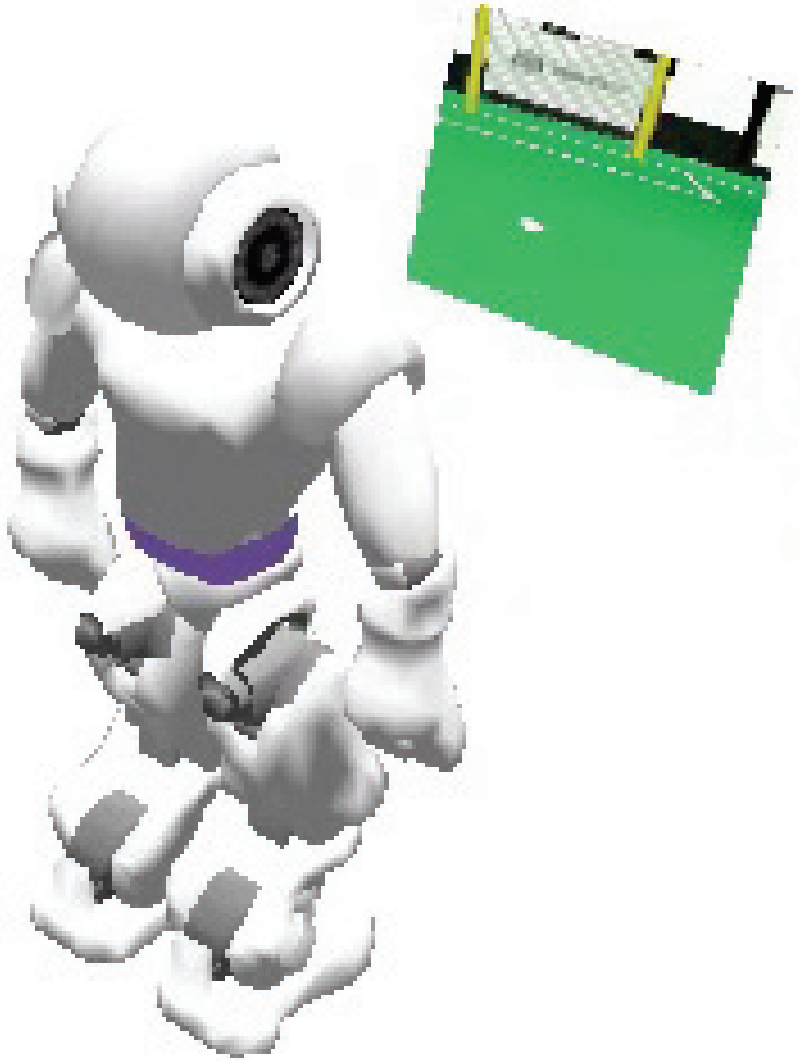
Verzerrungskorrektur

- **“Rolling shutter”**
  - CMOS-Chips belichten pixelweise
- **Zeitdifferenzen**
  - Bild
  - Gelenkwinkel
- **Korrektur**
  - Verrechnung der Geschwindigkeiten der Kopfgelenke
  - Nur Perzepte, nicht das ganze Bild



Aufnahme durch Bioloid-Roboter

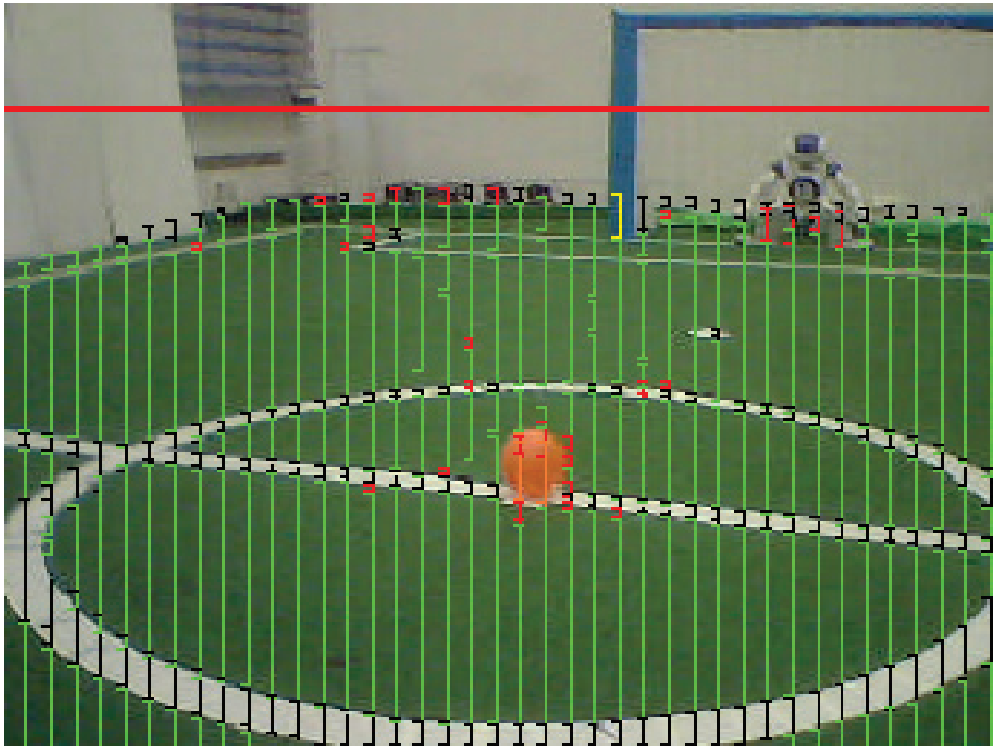
Perspektive (Video)



Video: Thomas Röfer

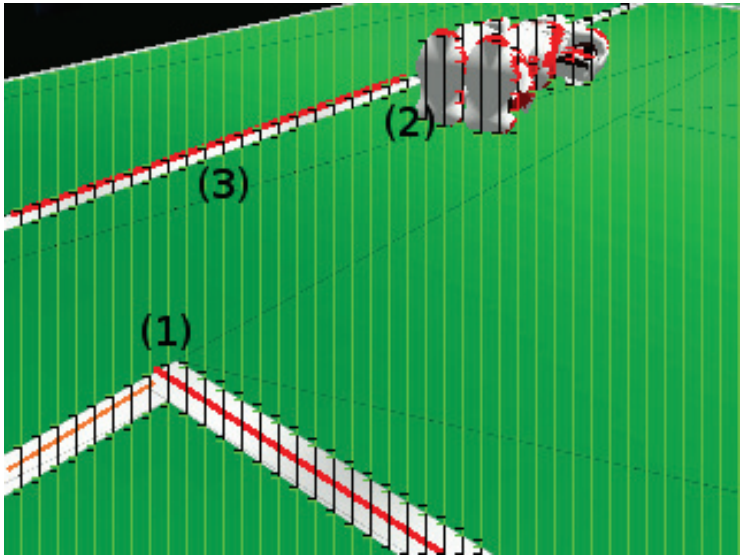
Grid-basierte Bildverarbeitung

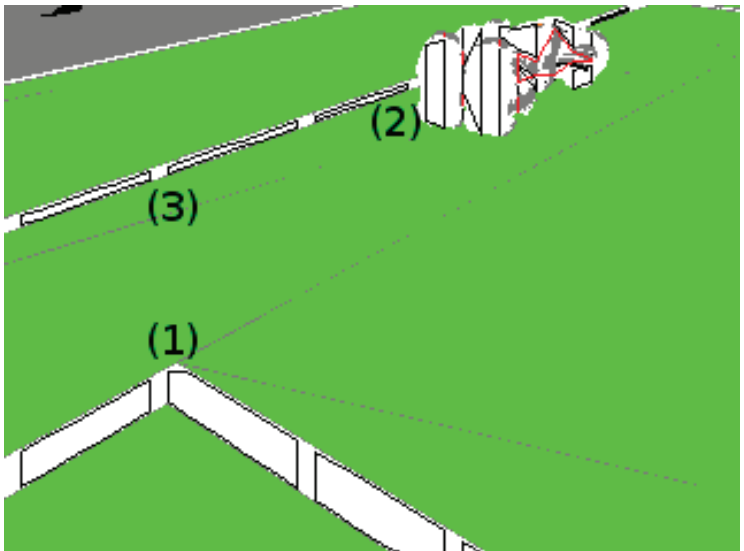
- **Ansatz**
  - Vertikale Segmente, beginnend beim Horizont, absuchen
  - In Horizontnähe horizontal suchen
  - Indikatoren für Features werden an “Spezialisten” weitergegeben
    - *z.B. Orange -> Lokale Ballsuche*
- **Vorteile**
  - Frühzeitige Beachtung räumlicher Relationen
  - Nur ein Bruchteil der Pixel wird tatsächlich verarbeitet



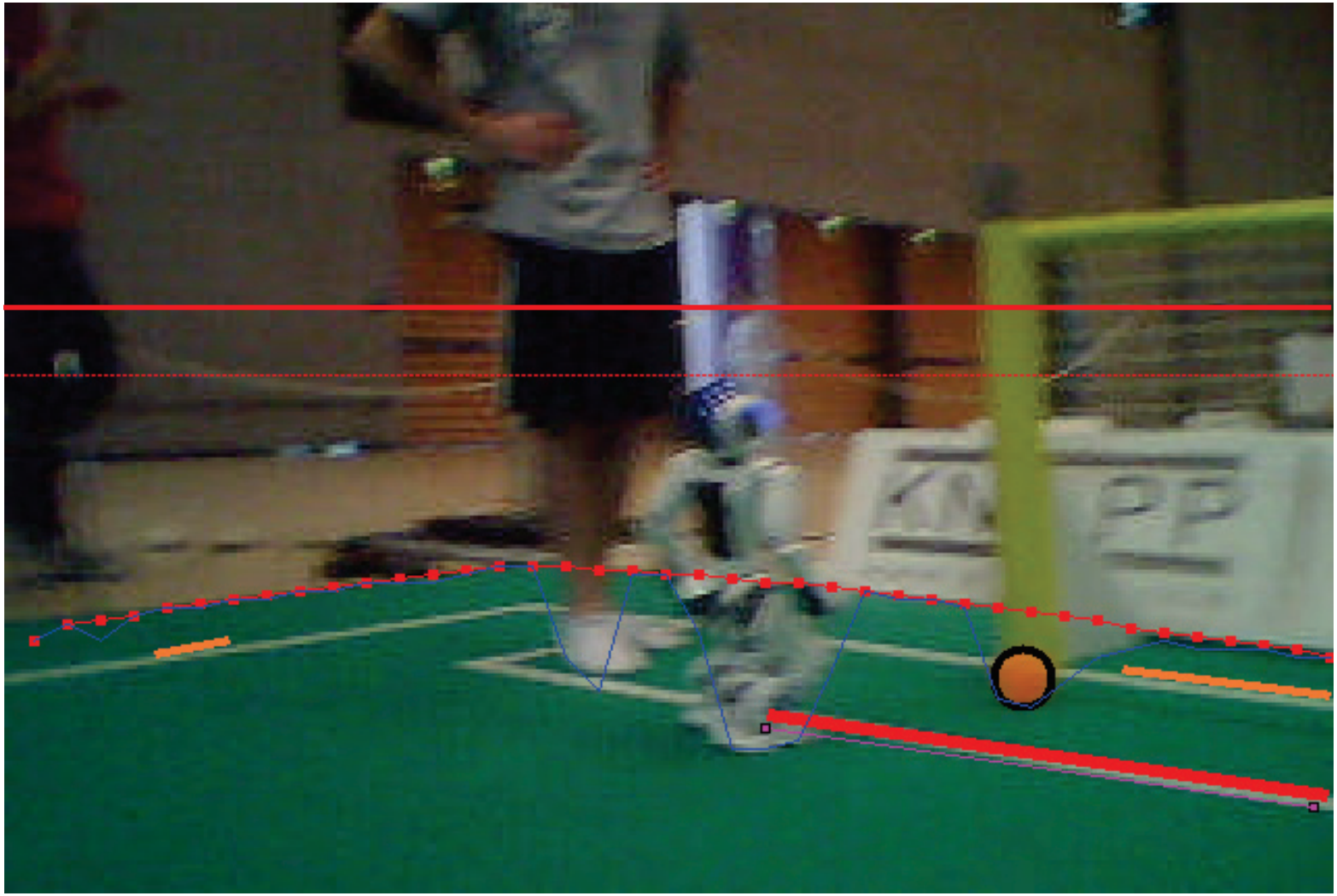
Von Segmenten zu Regionen

- **Ansatz**
  - Vereinigungssuche
  - Benachbarte Spalten durchlaufen
  - Kompatible Regionen vereinen
- **Keine Vereinigung wenn**
  - Farbklassen nicht übereinstimmen
  - Richtungen nicht zueinander passen
  - Durchschnittshöhe nicht passt
  - Region bereits zu groß ist

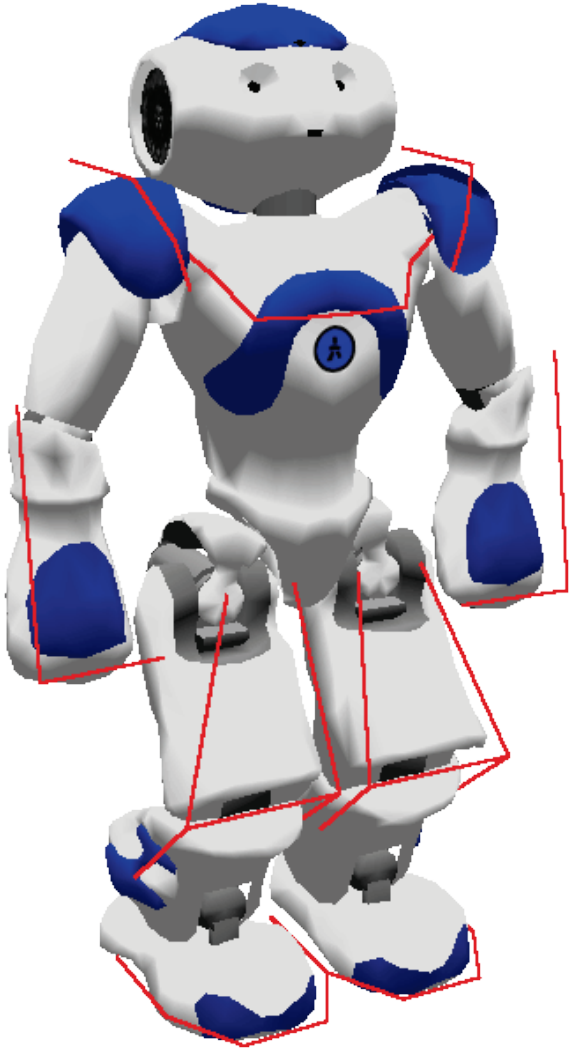


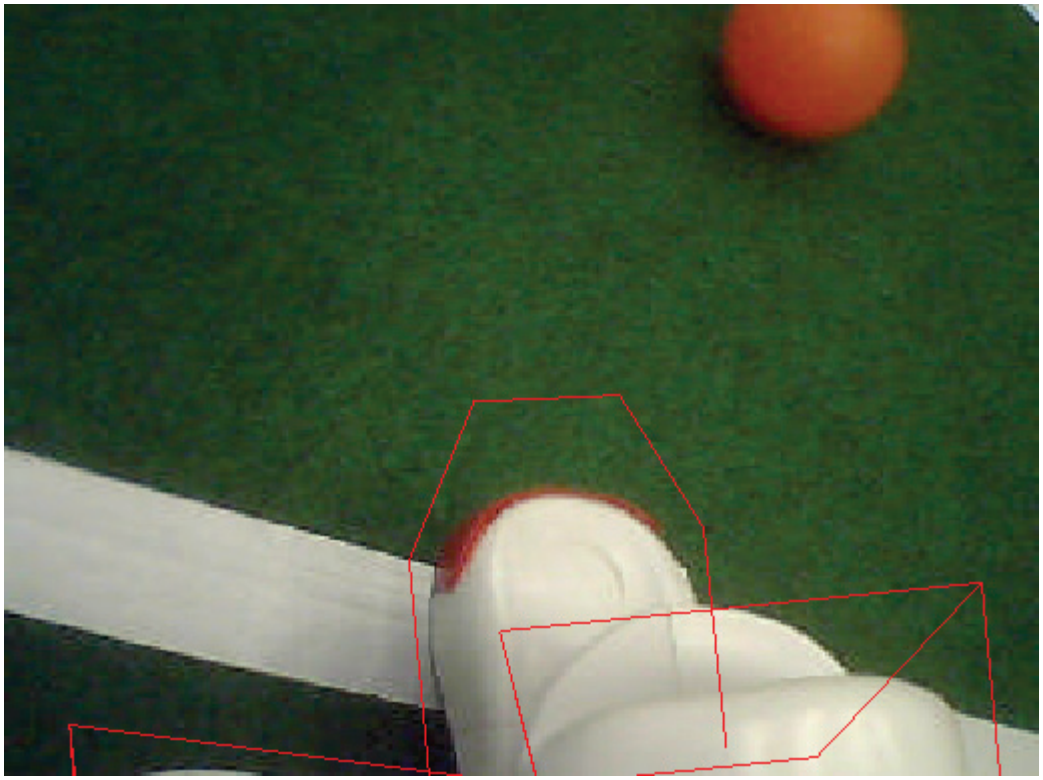


Bestimmung des Feldrands

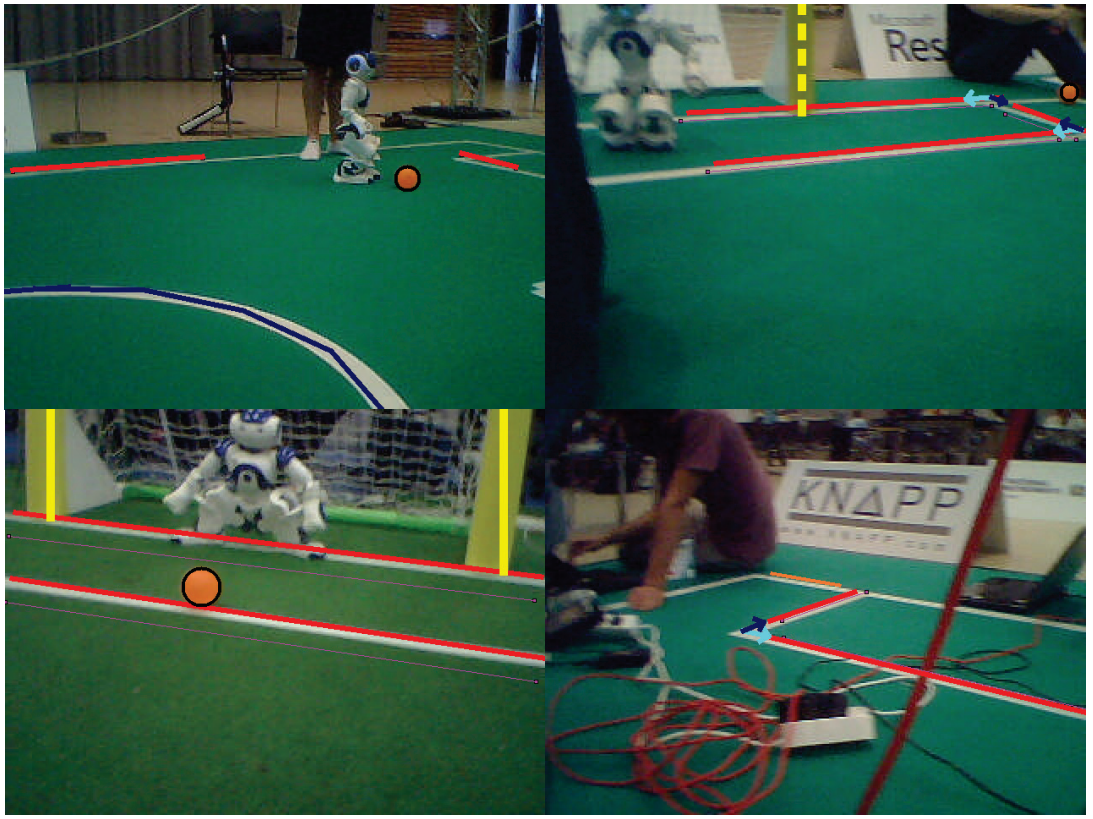


Den eigenen Körper ignorieren



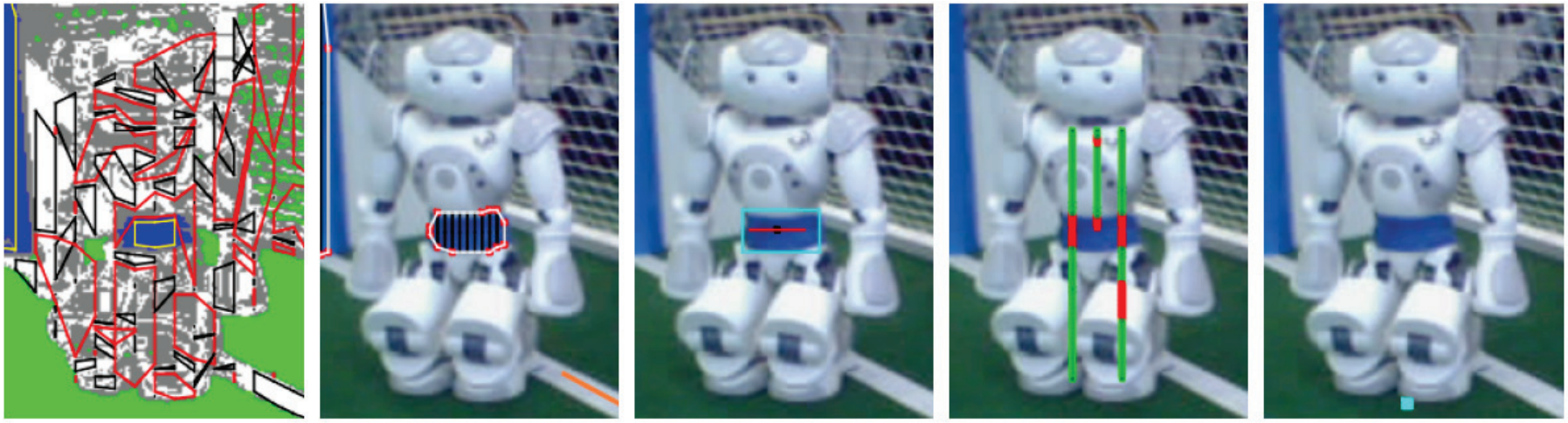


Aktuelle Perzepte



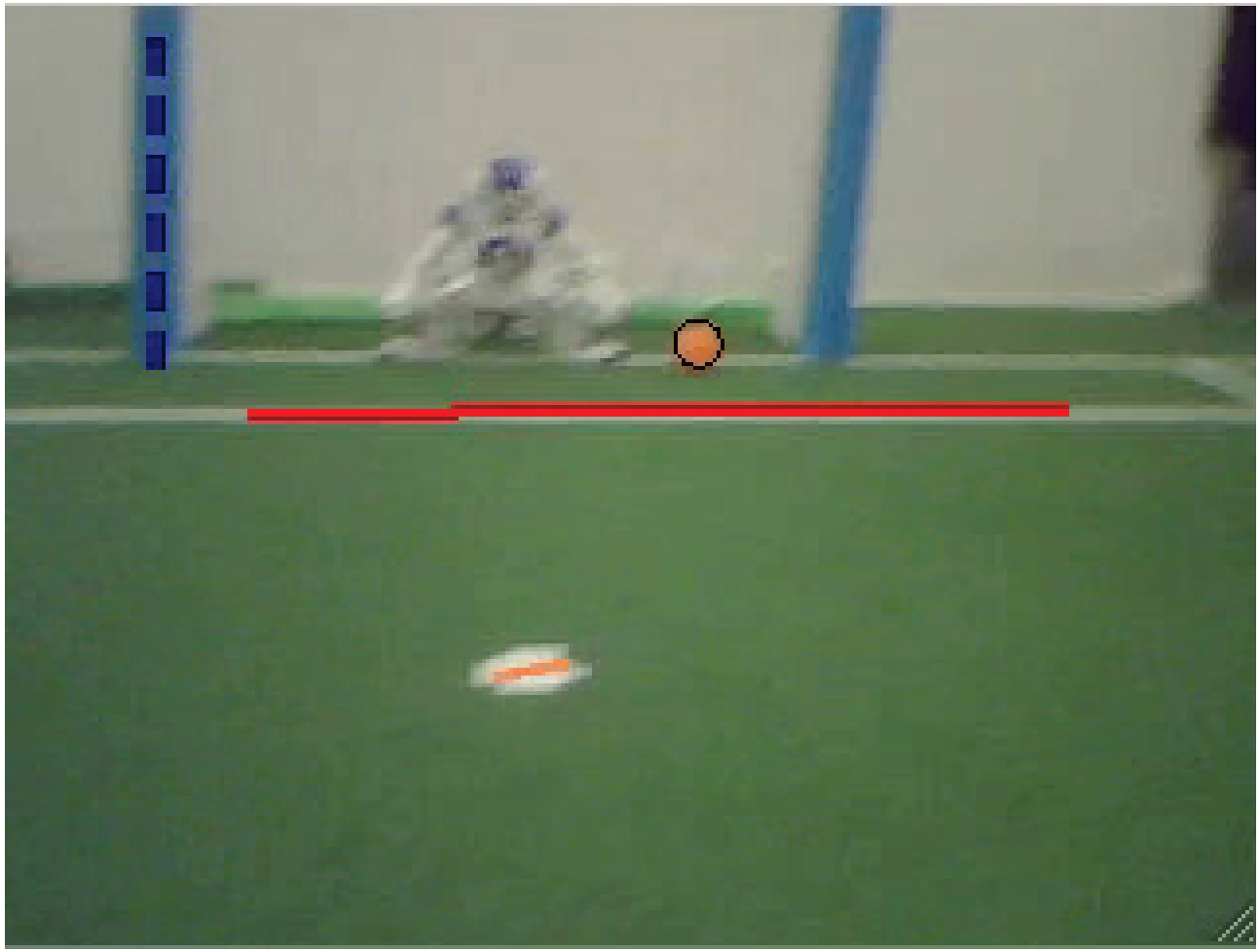
- **Erkannt werden**
  - Linien
  - Kreuzungen (L, T, X)
  - Mittelkreis
  - Torpfosten (und Latte)
  - Ball
  - Roboter
- **Ignoriert werden**
  - Alle Dinge außerhalb des Feldes
  - Linien in Robotern (eigener Körper und andere)
  - Sachen, die auf dem Feld rumliegen

Beispielspezialist: Roboter



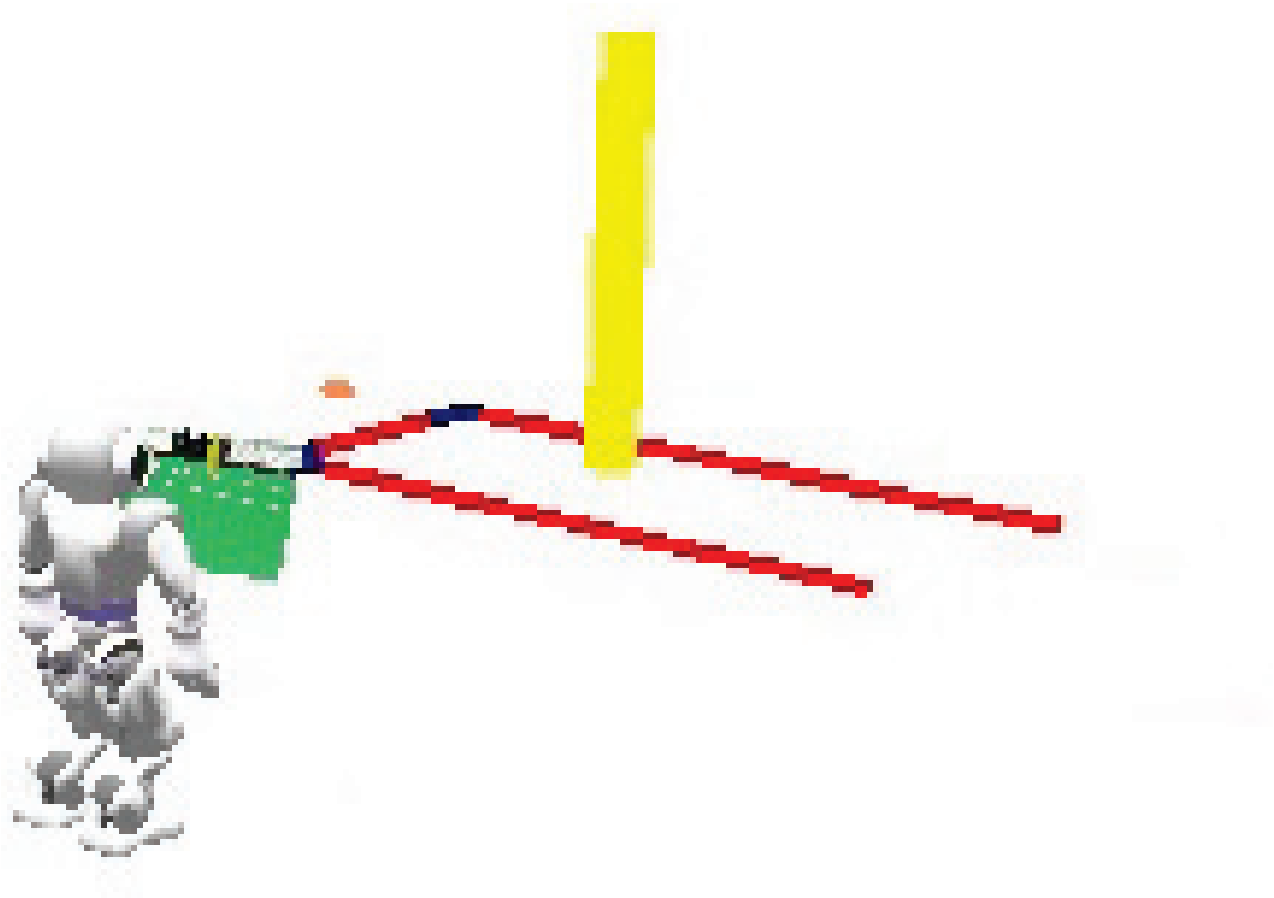
Blaue Region → Rand suchen → Flächenmerkmale → Weiß suchen → Fußpunkt suchen

Perzepte im Bild (Video)



Video: Thomas Röfer

Perzepte in der Welt (Video)

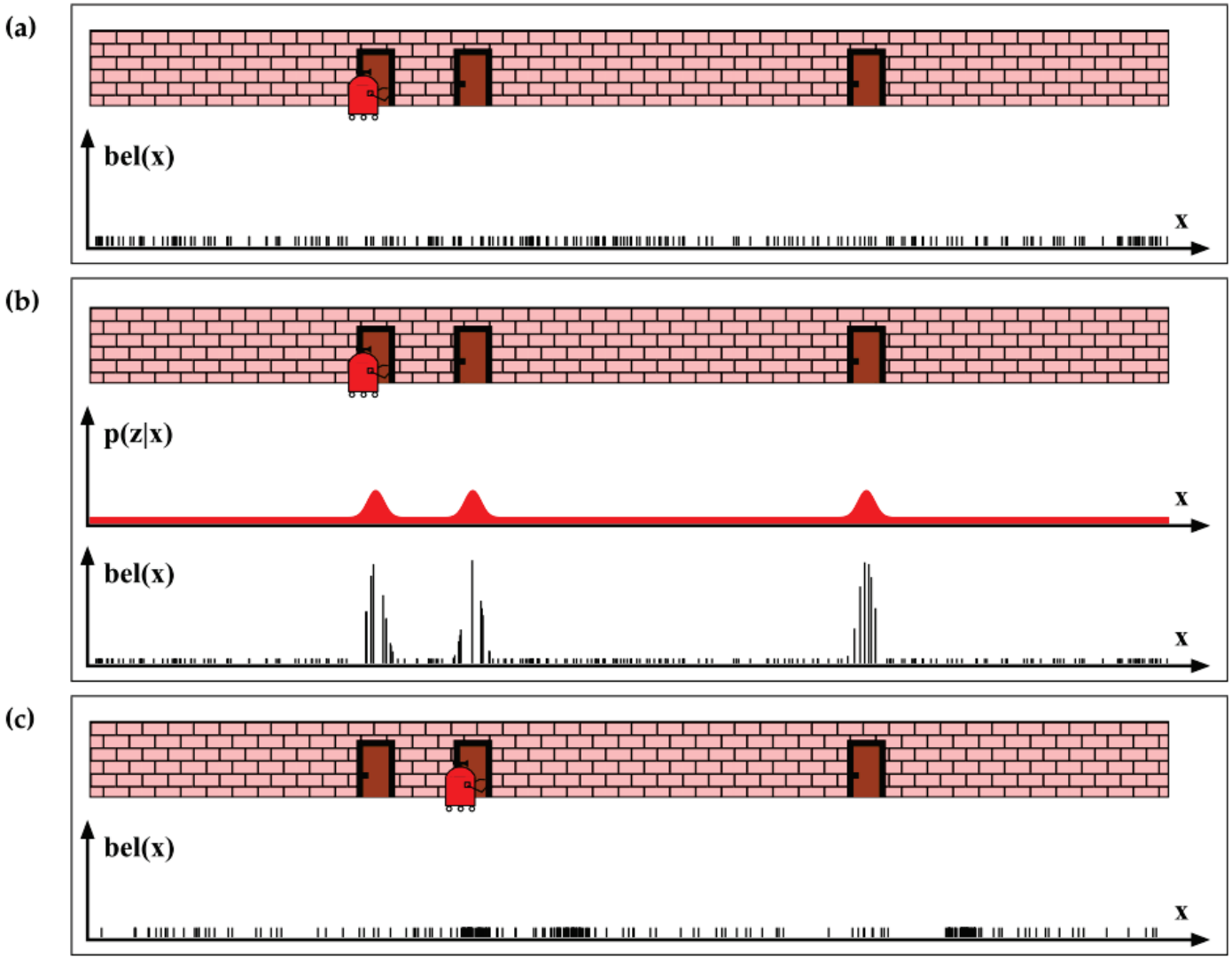


Video: Thomas Röfer

Selbstlokalisierung mit Partikelfilter

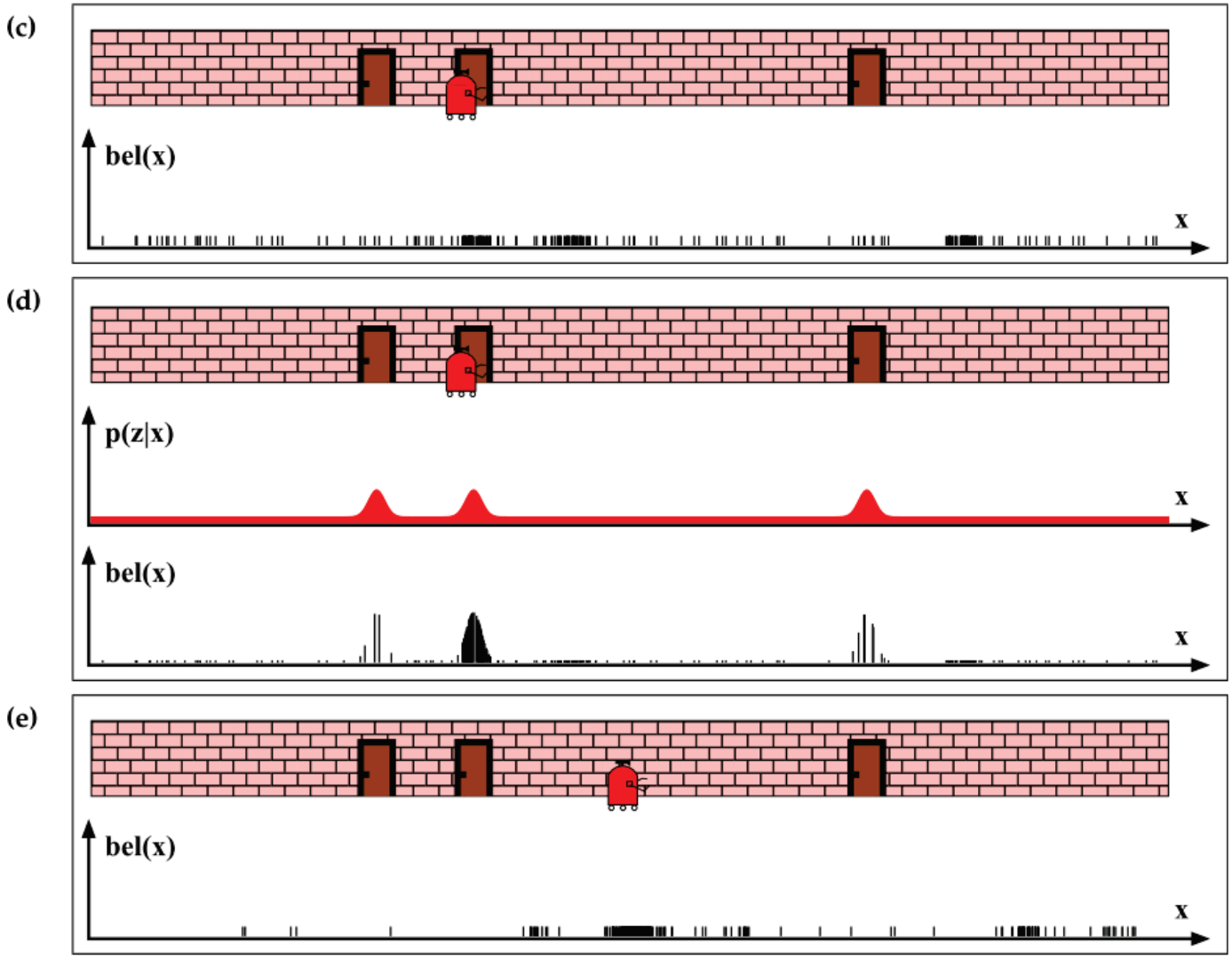
- **Eigene Position kann nicht robust und direkt auf der Basis einzelner Messungen bestimmt werden**
  - Features sind nicht eindeutig
  - Rauschen und Fehlmessungen
- **Partikelfilter: Approximation einer Wahrscheinlichkeitsverteilung durch Samples (Partikel)**
  - 100 Partikel
  - Beinhalten jeweils Position und Rotation in 2D
  - Standard-Implementierung nach “Probabilistic Robotics” (Thrun, Burgard, Fox)
  - + Sensor Resetting und Augmented-MCL
- **Vorteile**
  - Repräsentiert auch multimodale Wahrscheinlichkeitsverteilungen
  - Effiziente Behandlung des *“Kidnapped Robot Problem”*
- **De-facto Standard in einigen RoboCup-Ligen**
  - Oft kombiniert mit (Extended-/Unscented-) Kalman-Filter

Partikel-Filter (Beispiel)



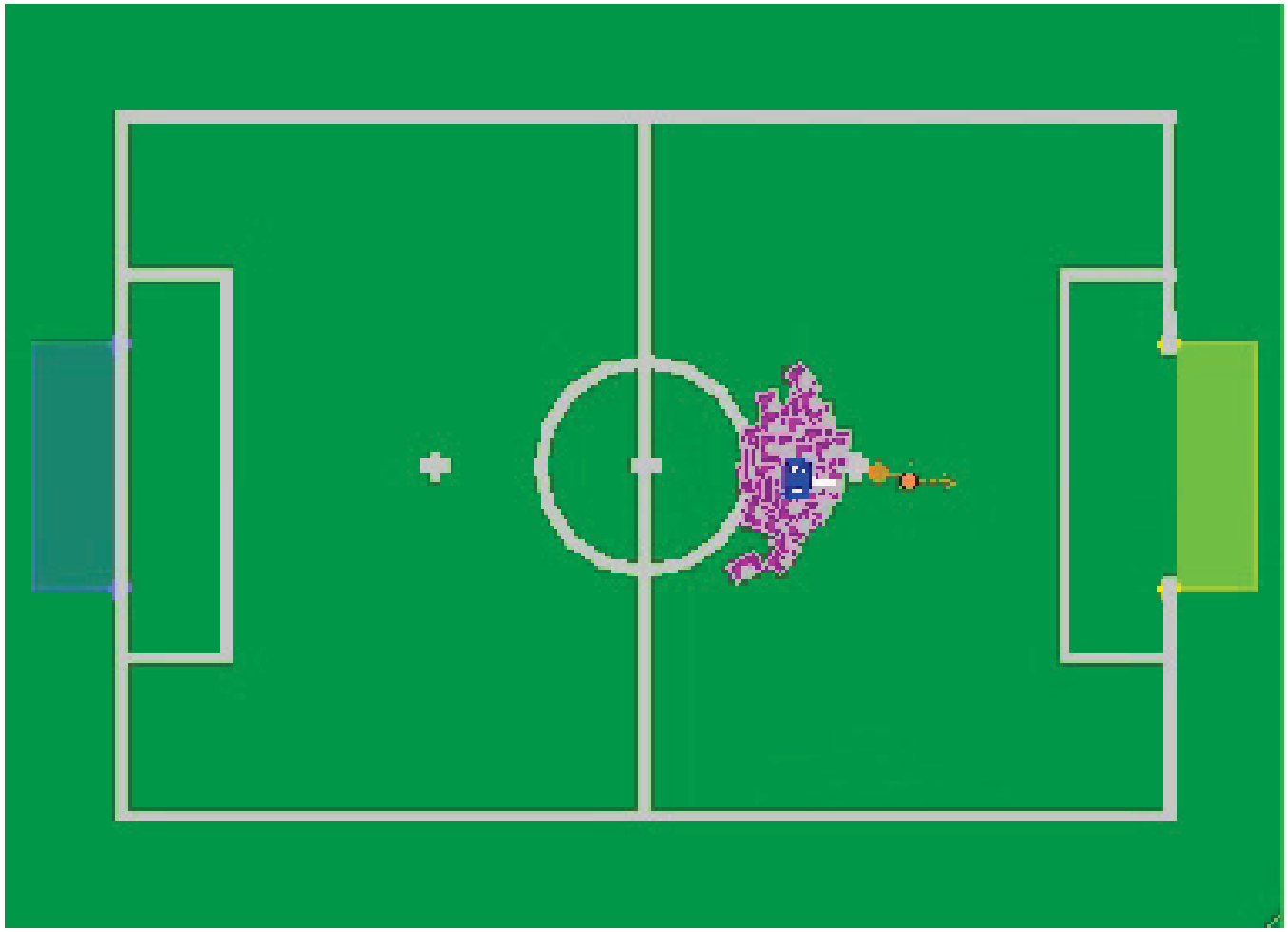
Quelle: http://www.probabilistic-robotics.org/

Partikel-Filter (Beispiel)



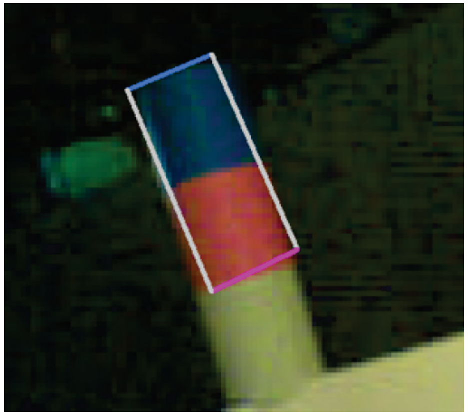
Quelle: http://www.probabilistic-robotics.org/

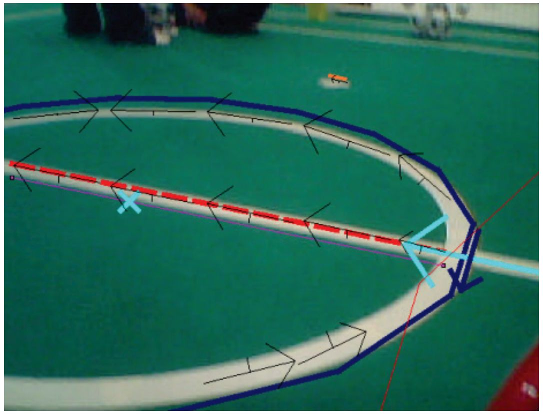
Selbstlokalisierung (Video)

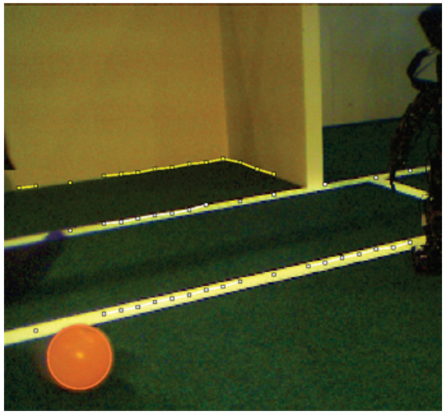


Video: Thomas Röfer

Klassen von Features für Lokalisierung

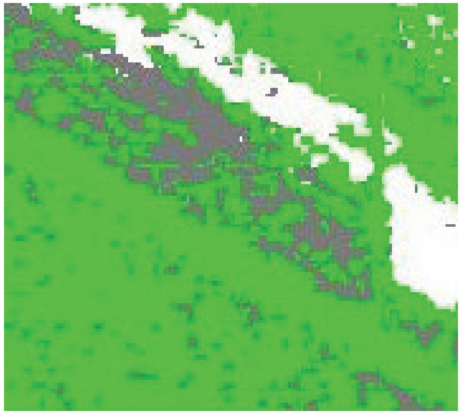


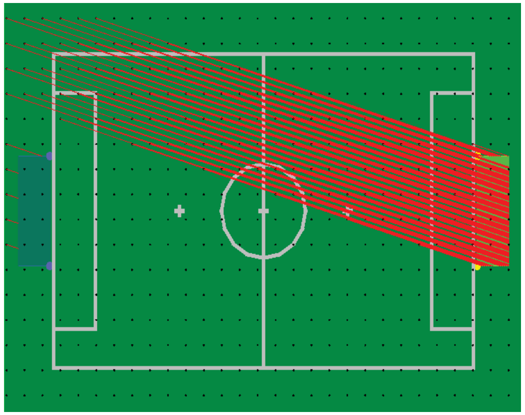




Eindeutige Features (früher) ←- → **Mehrdeutige Features**

Zustandsbasiertes Modell kompensiert schwache Bildverarbeitung





Zusammenfassung

- **Bildverarbeitung B-Human in der RoboCup Standard Platform League**
  - Tabellenbasierte Farbsegmentierung als Basis
  - Projektion auf den Boden durch bekannte Kamerapose
  - Durchsuchen einer Teilmenge (Grid) des Bildes
  - Spezielle Implementierungen für einzelne Objekte, um Effizienz und Robustheit zu verbessern
- **Lokalisierung mit Partikelfilter**