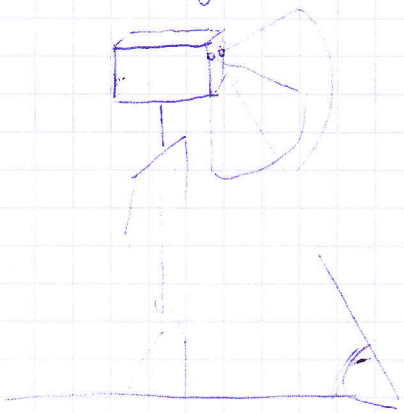
Computer Vision

Uns interessieren zwei große Themenbereiche

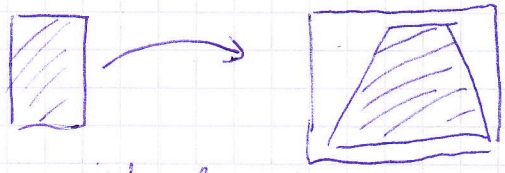
- Farbe
- Geometrie

1. Vorlesung 21.4.2009

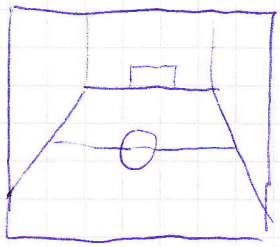
Wir fangen an mit FARBE.



Die Kamera des Roboters projizieren von den Objekten eine Projektion, da sie von oben raufschauen.



Ein Bild der Kamera auf einen Fussballfeld sieht dann ca. so aus:

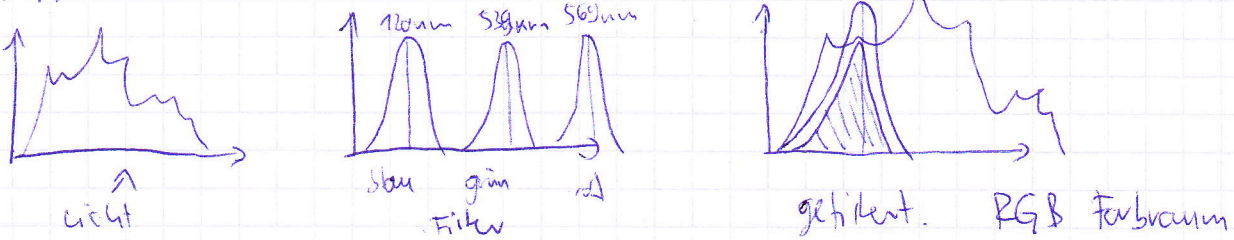


Farbsegmentierung

Farbe ist eine Empfindung des Menschen und muss durch die Maschine gemessen werden. Wie kann also das Gemessen?

Das Menschliche Auge hat Rezeptoren für die Farben rot, blau, grün sowie Rezeptoren, die auf Helligkeit reagieren.

Die Rezeptoren funktionieren wie Filter das ins Auge kommenden Lichts.



Durch das Filtern hat man natürlich schon ein Genauigkeitsverlust.

Schauen wir uns die Kamera an.

Bayer Filter

CCD - changed coupled device
CMOS

12 bit
8 - 10 bit

Chip

Licht-detektor

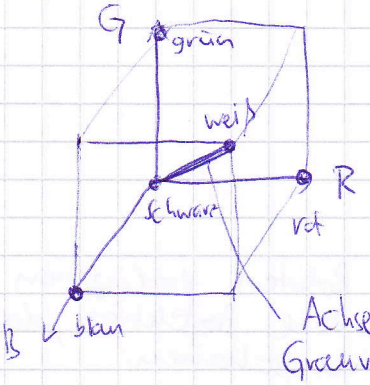
Man baut mehr grüne Filter ein, da das menschliche Auge in dem Bereich empfindlicher ist.

Der RGB Wert für einen Pixel wird interpoliert:

R	G	B
$r_1+r_2/2$	$g_{1+2+3+4}/4$	$b_1+b_2/2$

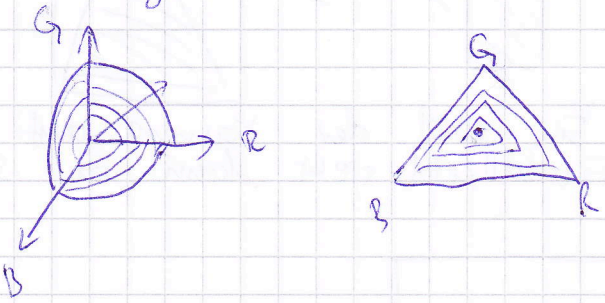
16 bit

Ein RGB Wert ist ein Punkt im RGB-Raum.



Der Abstand vom Ursprung bis zum Rand ist die Sättigkeit. Die Farbigkeit ist der Winkel des Vektors. (Hue)

Da die Farbe das entscheidende ist, kann man ohne Probleme von dem 3D RGB Raum in 2D gehen, in dem man die Oberfläche folgender Kugel betrachtet:



Der Radius gibt die Sättigung an und der Winkel die Farbigkeit. Die Intensität ist die Helligkeit.
Es gibt natürlich noch andere Farbräume: YUV, HSV, ISV, CIE ...

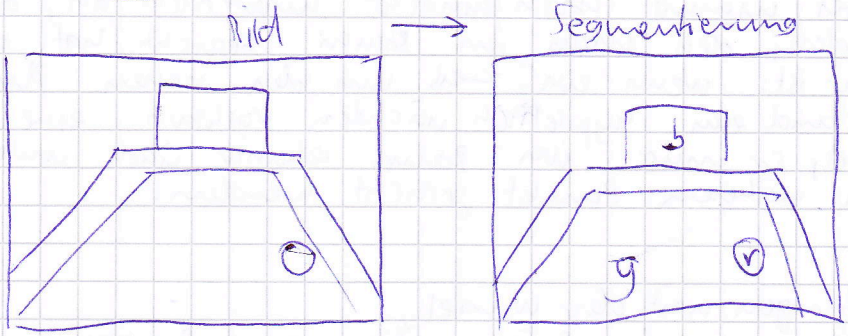
HSV-Raum

Intensität $V = \max(R,G,B)$

Sättigung $S = \begin{cases} 0 & R=G=B=0 \\ (\max[R,G,B] - \min(R,G,B)) / \max(R,G,B) & \text{sonst} \end{cases}$

$H = \begin{cases} 0^\circ & R=G=B \\ 60^\circ \cdot \frac{R-B}{\dots} & \end{cases}$

Übungsaufgabe

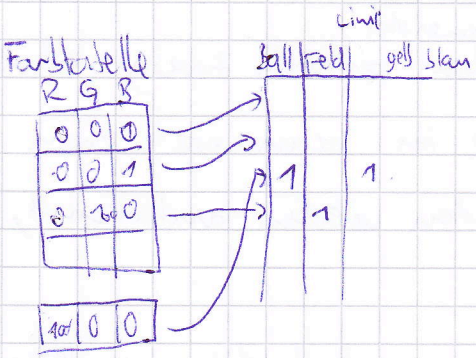


mit vielen Farben nur 5 Farben

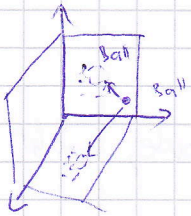
Mit Nearest Neighbours

Weiß und Schwarz einfach: weiß hohe Intensität

28.4.09
Vorlesung 2



Wie kann man nun die Farbtabelle ausfüllen?
Wir benutzen KD-Bäume



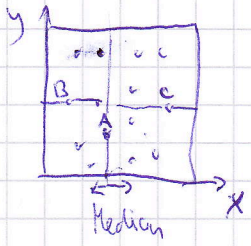
Man hat Punktwolken, die schon zu Objekten zugeordnet sind. Bei einem neuen Punkt wird geschaut, wer der nächste Nachbar ist, und dieser Kategorie zugeordnet.

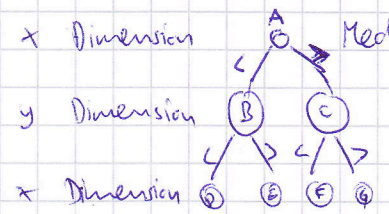
Das ist natürlich aufwendig für viele Punkte.

Wir machen das daher so:

a) Bild
b) Segmentieren
c) Farbtabelle füllen mit den Farben im segmentierten Bild
d) Farbtabelle mit Hilfe von KD-Tree ergänzen

Was ist ein KD-Tree? (K steht für die Anzahl der Dimensionen, hier also K = 3)





Solange rekursiv den Raum in x und y-Dimension teilen, bis in einer Region nur noch 1 Punkt übrig ist.

Ein neuer Punkt kann nun in logarithmischer Zeit eingeordnet werden. Dafür nimmt man zunächst hypothetisch an, dass der letzte Knoten, den man im Baum besucht hat, der nächste Nachbar ist. Wenn ein Kreis um den neuen Punkt mit Radius = Abstand zum hypothetisch nächsten Nachbarn eine andere Region schneidet, so muss im Baum rekursiv nach unten nach einem näheren Punkt gesucht werden.

Suche:

Start: Anfangen mit der Wurzel
Suchen von der Position des neuen Punktes im Raum

Current best: Wenn man ein Blatt erreicht, ist dies der current best nächste Nachbar.

Rekursion: Wir überprüfen, ob eine Kugel um Punkt zentriert und mit Radius = Abstand zum "current best" die nächstgelegene Region schneidet
a) wenn ja, folge dem anderen Zweig rekursiv bis unten
b) wenn nicht, laufe im Baum nach oben

Ende: Wenn man zum Wurzelknoten wieder ankommt, ist man fertig.

Test 1:
Beim Einfügen in die Farbtabelle kann man gleich die unmittelbaren Nachbarn mitmarkieren. Dann den KD-Tree bauen.

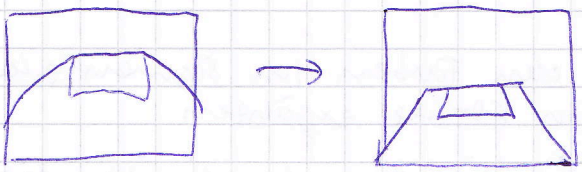
Bild → KD-Tree Farbtabelle → Neues Bild

tot: $\min(R-G, R-B) > 10$ oder so

Test 2: Das gleiche mal mit 7, 6, 5, 4 Bit pro Farbe, nicht mehr die 8 Bit.

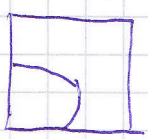
Optische Verzerrung

Wir wollen sowas:

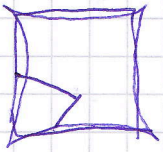


5.5.09 Radiale Verzerrung

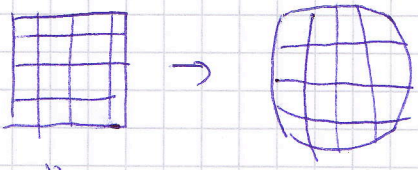
Vorlesung 3 Wie schafft man es, von



sowas auf



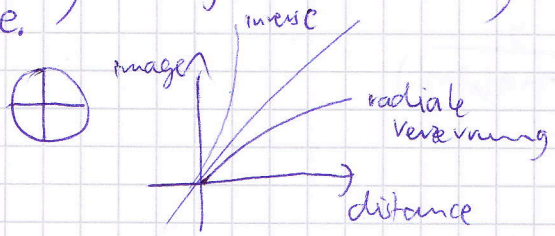
etwas zu kommen?



Es gibt mehrere Korrekturverfahren

1) Gitter. Man legt vor die Linse ein Gitter, dessen Koordinaten auf dem Bild dann Referenzpunkte darstellen.

2) Gitter-Kalibrierung. Bei radialer Verzerrung ist die Verzerrung in jeder Richtung bei gleichem Abstand dieselbe.



$(x,y) \to \left(\frac{x}{f(x,y)}, \frac{y}{f(x,y)}\right)$

$(x,y) \to \underbrace{\sqrt{x^2+y^2}}_{r}$

$f(x,y) = a_0 + a_1 r + a_2 r^2 + a_3 r^3 + \dots$

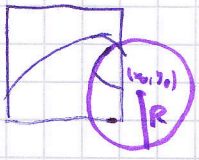
a_0 wenn Linse richtig zentriert; a_1

3) Wir arbeiten direkt auf dem Bild mit Hilfe von Geraden.

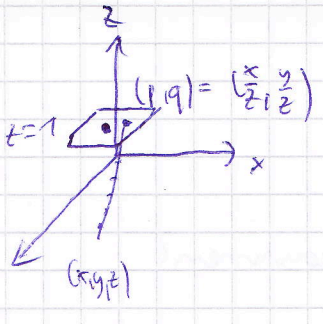
Modell: $1 + \lambda_1 r^2 + \lambda_2 r^4$

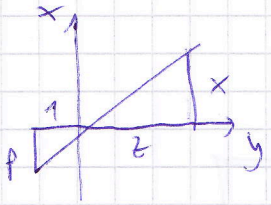
$$(x,y) \to \left(\frac{x}{1+\lambda_1 r^2}, \frac{y}{1+\lambda_1 r^2}\right)$$

Zu zeigen: Wenn man eine Linie im Original hat, muss sie zu einem Kreissegment abgebildet werden.



$\lambda = g(k, x_0, y_0)$





$\frac{p}{1} = \frac{x}{z}$

Projektive Geometrie

Homogene Koordinaten

$$\vec{P} = \begin{pmatrix} p \\ q \\ 1 \end{pmatrix} = \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

$$s\begin{pmatrix} p \\ q \\ 1 \end{pmatrix} = \begin{pmatrix} x \\ y \\ \frac{}{1+\lambda(x^2+y^2)} \end{pmatrix}$$

$$p = \frac{x}{1+\lambda(x^2+y^2)}$$

$$q = \frac{y}{1+\lambda(x^2+y^2)}$$

Modell

Gerade $ax' + by' + c = 0$

$$\begin{pmatrix} a \\ b \\ c \end{pmatrix} \cdot \begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = 0 \qquad \begin{matrix} 1 \neq 0 \\ c \neq 0 \end{matrix}$$

$$ax + by + c(1+\lambda(x^2+y^2)) = 0$$

Kreisgleichung $(x-x_0)^2 + (y-y_0)^2 = R^2$

$$\frac{a}{c\lambda}x + \frac{b}{c\lambda}y + x^2 + y^2 + \frac{1}{\lambda} = 0$$

$$\left(x + \frac{a}{2c\lambda}\right)^2 - \frac{a^2}{4c^2\lambda^2} + \left(y + \frac{b}{2c\lambda}\right)^2 - \frac{b^2}{4c^2\lambda^2} + \frac{1}{\lambda} = 0$$

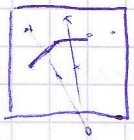
$$x_0 = -\frac{a}{2c\lambda} \qquad y_0 = -\frac{b}{2c\lambda}$$

$$R^2 = \frac{a^2}{4c^2\lambda^2} + \frac{b^2}{4c^2\lambda^2} - \frac{1}{\lambda}$$

$$= x_0^2 + y_0^2 - \frac{1}{\lambda}$$

↑ Mittelpunkt von Kreis

Wie findet man nun einen Kreis?



Mit Zirkel und Lineal ist es umgenau und fehleranfällig

Daher machen wir das numerisch

$$Ax^2 + By^2 + Cx + Dy + E = 0$$

(x_1, y_1), (x_2, y_2), (x_3, y_3) ... Daher

$$\begin{pmatrix} x_1^2 & y_1^2 & x_1 & y_1 & 1 \\ x_2^2 & y_2^2 & x_2 & y_2 & 1 \\ \vdots & & & & \end{pmatrix} \cdot \begin{pmatrix} A \\ B \\ C \\ D \\ E \end{pmatrix} = 0$$

Für einen Kreis kann man gleich $A = B = 1$ annehmen. Aber man nimmt am besten so viele Punkte wie möglich, um Fehler klein zu halten.

$$\begin{pmatrix} x_1^2 & y_1^2 & x_1 & y_1 \\ \vdots & & & \end{pmatrix} \cdot \underbrace{\begin{pmatrix} A \\ B \\ C \\ D \end{pmatrix}}_{P} = -\underbrace{\begin{pmatrix} 1 \\ 1 \\ 1 \\ \vdots \end{pmatrix}}_{Q} E$$

$n \times 4$ 4×1 $n \times 1$

$$X \cdot P = Q$$

$$X^T X P = X^T Q$$

$$P = (X^T X)^{-1} X^T Q$$

Anders gehen es über den Gradientenabstieg:

Minimiere $E^2 = \sum_{i=0}^{N} \left((x_i - x_0)^2 + (y_i - y_0)^2 - r^2 \right)^2$

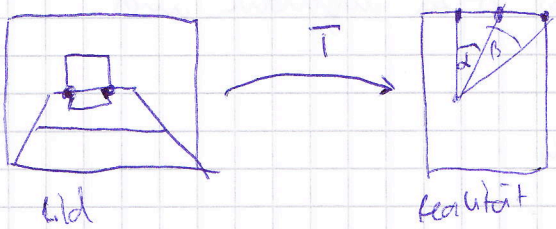
unbekannt

Lokalisierung

12.5.09

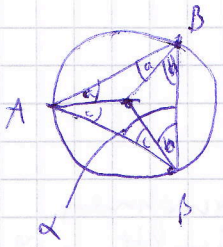
Vorlesung 4

①



Es reichen 3 Punkte aus, die man auf dem Bild lokalisieren kann und von denen man weiß, wo sie in der Realität liegen.

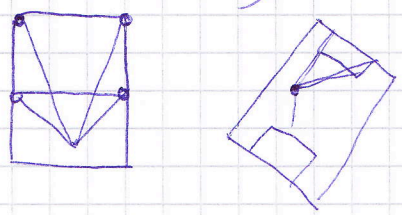
Das liegt daran, dass man aus zwei Punkten und dem Winkel einen Kreis konstruieren kann, auf dessen Rand der Roboter stehen muss. Durch noch einen dritten Punkt sieht sich ein zweiter, dessen Schnittpunkt mit dem anderen der Standort sein muss.



$$2a + 2b + 2c = \pi$$
$$a + b + \underbrace{c}_{\alpha} = \pi/2$$
$$\alpha = \frac{\pi}{2} - a$$

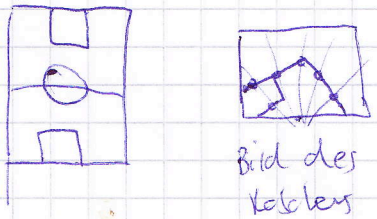
Jetzt Lokalisierung mit 4 Punkten.

②



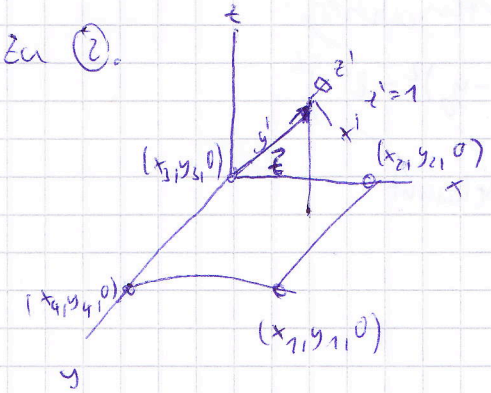
Das tolle hierbei ist, dass man durch die Punkte, die alle auf einer Ebene liegen müssen sogar die genaue Position der Kamera bestimmen kann, also die Höhe

③



Jetzt muss man versuchen, diese extrahierten Punkte auf das Feld zu matchen

Zu ②.



Weltkoordinaten

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

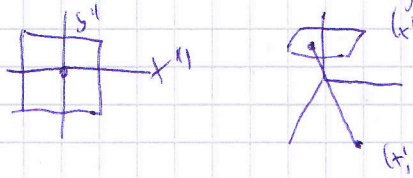
Kamerakoordinaten

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = R\left(\begin{pmatrix} x \\ y \\ z \end{pmatrix} - \vec{t}\right)$$

R: Rotation

$$= R\begin{pmatrix} x \\ y \\ z \end{pmatrix} - R\vec{t}$$

Im Kamerabild hat man ja nur zwei Koordinaten



Bekannt sind:

$$(x_1, y_1), (x_2, y_2), (x_3, y_3), (x_4, y_4)$$
$$(x''_1, y''_1), (x''_2, y''_2), (x''_3, y''_3), (x''_4, y''_4).$$

Unbekannt sind: $R, \vec{t}$

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = R\begin{pmatrix} x \\ y \\ 0 \end{pmatrix} - R\vec{t}$$

$$= \left(\begin{array}{cc|c} v_{11} & v_{12} & \\ v_{21} & v_{22} & -R\vec{t} \\ v_{31} & v_{32} & \end{array}\right) \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{pmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

$$x'' = \frac{x_1'}{z_1'} \qquad x'' z_1' = x_1' \qquad y'' z_1' = y_1'$$

$$y'' = \frac{y_1'}{z_1'} \qquad \} \text{ (i)}$$

$$x_1' = (x_1 \; y_1 \; 1 \; 0 \; 0 \; 0 \; 0 \; 0 \; 0) \cdot \vec{h}$$

$$y_1' = (0 \; 0 \; 0 \; x_1 \; y_1 \; 1 \; 0 \; 0 \; 0) \cdot \vec{h}$$

$$z_1' = (0 \; 0 \; 0 \; 0 \; 0 \; 0 \; x_1 \; y_1 \; 1) \cdot \vec{h} \qquad \text{(ii)}$$

Man benutzt nun (i) und (ii) um etwas Lösbares zu bekommen.

$$x_1'' (0 \; 0 \; 0 \; 0 \; 0 \; 0 \; x_1 \; y_1 \; 1)\vec{h} = (x_1 \; y_1 \; 1 \; 0 \; 0 \; 0 \; 0 \; 0 \; 0)\vec{h}$$

$$y_1'' (0 \; 0 \; 0 \; 0 \; 0 \; 0 \; x_1 \; y_1 \; 1)\vec{h} = (0 \; 0 \; 0 \; x_1 \; y_1 \; 1 \; 0 \; 0 \; 0)\vec{h}$$

$$P_L \begin{cases} (x_1 \; y_1 \; 1 \; 0 \; 0 \; 0 \; -x_1'' x_1 \; -x_1'' y_1 \; -x_1'')\vec{h} = 0 \\ (0 \; 0 \; 0 \; x_1 \; y_1 \; 1 \; -y_1'' x_1 \; -y_1'' y_1 \; -y_1'')\vec{h} = 0 \end{cases}$$

$$\& \left(\begin{array}{cccccc|c} \overbrace{x_1 & y_1 & 1 & 0 & 0 & 0 & -x_1'' x_1 & -x_1'' y_1}^{D} & \overbrace{-x_1''}^{B} \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -y_1'' x_1 & -y_1'' y_1 & -y_1'' \\ x_2 & y_2 & 1 & 0 & 0 & 0 & -x_2'' x_2 & -x_2'' y_2 & -x_2'' \\ 0 & 0 & 0 & x_2 & y_2 & 1 & -y_2'' x_2 & -y_2'' y_2 & -y_2'' \\ \vdots & & & & & & & & \end{array}\right) \cdot \begin{pmatrix} h_1 \\ h_2 \\ h_3 \\ h_4 \\ h_5 \\ h_6 \\ h_7 \\ h_8 \\ h_9 \end{pmatrix} = 0 \qquad \& A \cdot \vec{h} = 0$$

Da wir nur 8 Gleichungen aber 9 Unbekannte haben, setzt man einfach eine Unbekannte auf 1.

$h_9 = 1$ $\qquad D \cdot \begin{pmatrix} h_1 \\ \vdots \\ h_8 \end{pmatrix} = -B$ ist lösbar.

Jetzt kennen wir die Matrix mit den h_i.

$$c_0 \left(\begin{array}{cc|c} r_{11} & r_{12} & \\ r_{21} & r_{22} & -R\vec{t} \\ r_{31} & r_{32} & \end{array} \right) = \begin{pmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{pmatrix}$$

$\vec{r}_1$ $\vec{r}_2$

c_0 ist der Spalte sollte eins sein (wegen Rotation)

$$c = \sqrt{h_1^2 + h_4^2 + h_7^2}$$

Wenn man zwei Spalten der Rotationsmatrix hat, hat man automatisch die 3. Und zwar muss dieser Vektor normal zu den beiden anderen sein.

$$\vec{r}_1 \times \vec{r}_2 = \vec{r}_3$$

$$\vec{a} \times \vec{b} = \begin{pmatrix} a_2 b_3 - a_3 b_2 \\ a_3 b_1 - a_1 b_3 \\ a_1 b_2 - a_2 b_1 \end{pmatrix}$$

$$R = \left(\begin{array}{ccc} h_1/c & h_2/c & \\ h_4/c & h_5/c & \vec{r}_1 \times \vec{r}_2 \\ h_7/c & h_8/c & \end{array} \right)$$

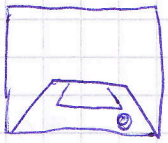
$$\begin{pmatrix} h_3/c \\ h_6/c \\ h_9/c \end{pmatrix} = -R\vec{t}$$

Jetzt hat man also R und $\vec{t}$ gefunden und ist fertig.

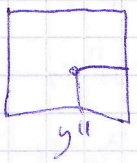
Diese Methode benutzt man besonders zum Kalibrieren.
So kann man gleich mehrere Schritte auf einmal machen:

- Entzerrung
- Kamerawinkel

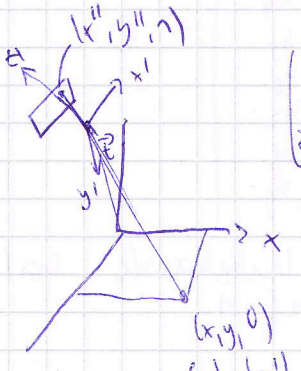
Jetzt will man natürlich auch andere Objekte finden.



Man hat im Bild die Position des Balls erkannt.



$\Rightarrow (x,y,z)$ und $R, \vec{t}$ bekannt



$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = R \begin{pmatrix} x \\ y \\ 0 \end{pmatrix} - R\vec{t}$$

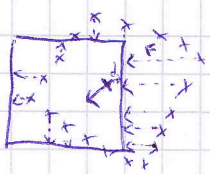
Vorlesung 5
26.5.2009

Lokalisierung

matching ?

EM-Algorithmus (Expectation-Maximization)

Bsp: E



Die Punkte werden mit den Linien assoziiert, die am nächsten stehen. Man könnte die Linien nun als magnetisch annehmen, so dass die Punkte von ihnen angezogen werden

$F = c \sum F_i$

$D = d \sum_i d_i$

M Rotation, Translation

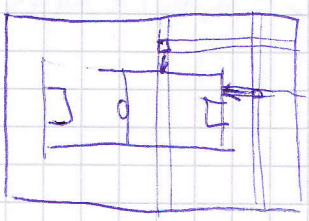
Das iteriert man dann, bis zur Konvergent oder bis zu einer bestimmten Anzahl von Iterationen.

Die Position des Roboters ist der Ursprung der Punktwolke. Durch das Matching wird dann die korrekte Position gefunden.

So wäre der Algorithmus relativ langsam, da man viele Abstände berechnen muss.

Der EM-Tabellen geht das schneller.

EM-Tabelle



Man berechnet im voraus für Gitterpunkte, die man über das Modell legt, die wirkende Kraft. Alle werte speichert man in einer Tabelle.

Rotationen

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

Rotation um z-Achse

Rotationsmatrix ist multiplikation aller Einzelmatrizen

$R^3 = R_z R_y R_x$

Für die Eigenvektoren gilt sei Rotation $R\vec{x} = \vec{x}$

Quaternionen (ℍ) Verallgemeinerung der komplexen Zahlen

$a + bi$
$i^2 = -1$

$(a+bi) + (c+di) = (a+c) + (b+d)i$

$(a+bi)(c+di) = (ac - bd) + (ad + bc)i$

ℝ	ℂ	ℍ
a	$a+bi$	(a,b,c,d)
+	+	$a + bi + cj + dk$
−	−	$i^2 = j^2 = k^2 = ijk = -1$
·	·	$(a+bi+cj+dk) + (p+qi+rj+sk) = (a+p) + (b+q)i + (c+r)j + (d+s)k$
÷	÷	$(1+2i)(i+j) = i + j + 2i^2 + 2ij$
		$= -2 + i + j + 2k$

$ij = k$ $\quad ji = -k$
$jk = i$ $\quad kj = -i$
$ki = j$ $\quad ik = -j$

$(u_x, u_y, u_z) = \vec{u}$ Vektor

↳ $u_x i + u_y j + u_z k$ ⋯ Repräsentation des Vektors als Quaternion

$\vec{v} = (v_x, v_y, v_z) \quad v_x i + v_y j + v_z k$

$\vec{u}\vec{v} = -u_x v_x - u_y v_y - u_z v_z + i(u_y v_z - u_z v_y) + j(u_z v_x - u_x v_z) + k(u_x v_y - u_y v_x)$

$uv = -(u \cdot v) + (u \times v)$

$vu = (u \cdot v) - (u \times v)$
$= -(v \cdot u) + (v \times u)$

$u \cdot v = -\frac{uv + vu}{2}$

$u \times v = \frac{uv - vu}{2}$

$$v \rightarrow R v R^{-1}$$

$$R = \left(\cos \frac{\Theta}{2}\right) + \left(\sin \frac{\Theta}{2}\right) \vec{n}$$

↑ Einheitsvektor Rotationsachse

(alles als Quaternionen)

2.6.09
CV

Rotation mit Quaternion

$R = \left(\cos\frac{\alpha}{2}\right) + \left(\sin\frac{\alpha}{2}\right)\vec{u}$

$v \to RvR^{-1}$

zu rotierender

Quaternion

Identitäten

$uv = \frac{uv + vu}{2}$ Skalarprodukt

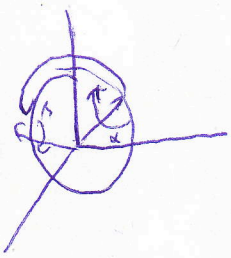
$u \times v = \frac{uv - vu}{2}$

$uvw = -[uvw] - (vw)u + (uw)v - (uv)w$

$[uvw] = \frac{wvu - uvw}{2}$

$(u \times v) \times w = \frac{uvw - wvu}{2}$

$u \times (v \times w) = \frac{uvw - vwu}{2}$



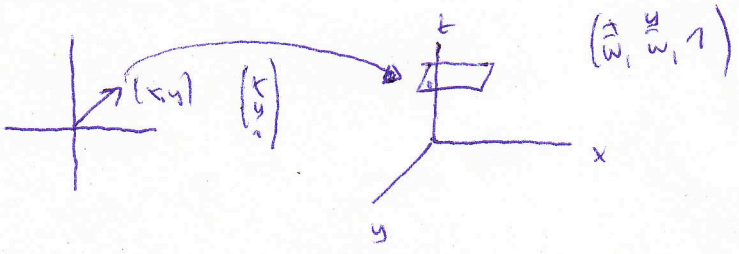
norm $|(1-t)\vec{u} + \vec{v}.t)$

$t = 0 \ldots 1$

Geometrien	Euklidisch	Ähnlichkeit	Affine	Projektiv
Rotation	✓	✓	✓	✓
Translation	✓	✓	✓	✓
Skalierung		✓	✓	✓
Skalierung (unterschiedliche Faktoren)			✓	✓
Scherung			✓	✓
Perspektive				✓
Komposition von Perspektiven				✓

Bei der projektiven Geometrie bleiben Linien, Flächen und Cross-ratio erhalten.

Homogene Koordinaten



$(\frac{x}{w}, \frac{y}{w}, 1)$

$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} x \\ y \\ w \end{pmatrix}$ $\quad \frac{x}{w} = x$

$\frac{y}{w} = y$

Linien

$ax+by+c=0$

$a \cdot w + b y w + c w = 0$

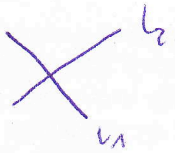
$aX + bY + cW = 0$

$\begin{pmatrix} x \\ y \\ w \end{pmatrix} \cdot \begin{pmatrix} a \\ b \\ c \end{pmatrix} = 0$ $\qquad \vec{p} \cdot \vec{l} = 0$ Punktitat

Zwei Linien

$l_1 = (a_1, b_1, c_1)^T$

$l_2 = (a_2, b_2, c_2)^T$



Schnittpunkt der Linien ist $\vec{p} = \vec{l_1} \times \vec{l_2}$

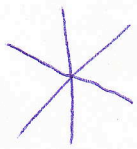
Linie aus 2 Punkten

$\vec{l} = \vec{p_1} \times \vec{p_2}$

$\vec{p_1}, \vec{p_2}, \vec{p_3}$ kollinear!

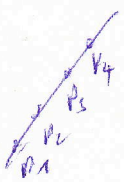
$\vec{l} = \vec{p_2} \times \vec{p_3}$ $\qquad \vec{p_1} \cdot (\vec{p_2} \times \vec{p_3}) = 0$

$\vec{l_1}, \vec{l_2}, \vec{l_3}$



$\vec{l_1} \cdot (\vec{l_2} \times \vec{l_3}) = 0$ $\qquad [l_1\ l_2\ l_3] \hat{=}$ Determinante der 3 Vektoren

Cross-ratio



$CR = \frac{\Delta_{13}\,\Delta_{24}}{\Delta_{14}\,\Delta_{23}}$ $\qquad \Delta_{ij}$ ist Abstand zwischen Punkten p_i zu p_j

bei der projektiven Transformation ist die Cross-Ratio invariant.

Abstand Gerade / Punkt

$d = (a_1 x + b_1 y + c_1) \cdot \gamma$

mit

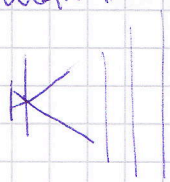
9.6.2009

Vorlesung 7

Stereo Vision

Wir wollen Tiefeninformationen

- ~~Tiefeninformation~~
- Fokus
  - nah
  - mittel
  - fern



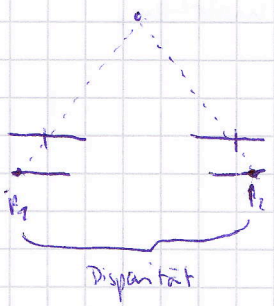
- Kamera scannen

optical flow

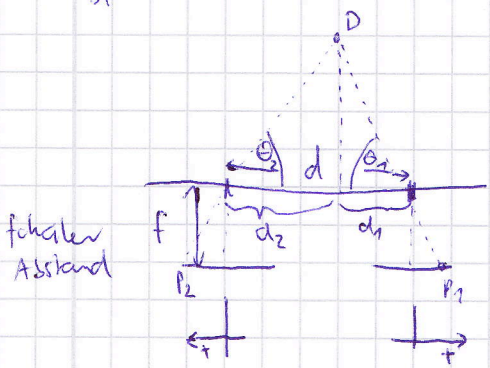
Je weiter weg die Gegenstände sind, umso weniger bewegen sie sich

- Mehrere Kameras

(Binokular)
(Trinokular)



Binokular



$$\frac{D}{d_2} = \tan\theta_2 = \frac{f}{P_2}$$

$$\frac{D}{d_1} = \tan\theta_1 = \frac{f}{P_1}$$

$$d_2 = D\frac{P_2}{f} \qquad d_1 = D\frac{P_1}{f}$$

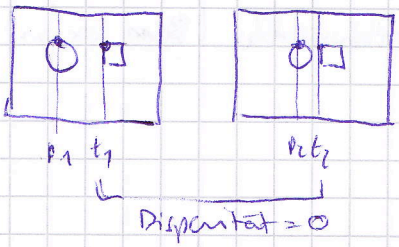
$$d = D\left(\frac{P_2}{f} + \frac{P_1}{f}\right)$$

$$D = \frac{df}{P_1 + P_2}$$

mit selben Vorzeichen

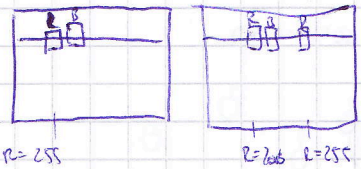
$$D = \frac{df}{P_1 - P_2}$$

Man hat jetzt zwei Bilder



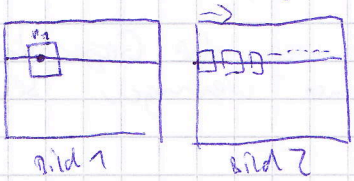
Wie kann man nun die Pixel in Korrespondenz bringen?

Matching



- Verdeckungen
- Reihenfolge der Pixel

- Block matching

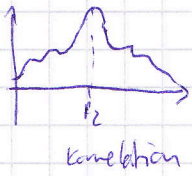


Scannen der gleichen Zeile mit dem Block aus Bild 1 in Bild 2.

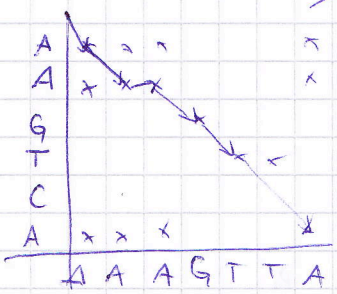
Die Blöcke ~~sind~~ berechnet man das Skalarprodukt.

Ein großer Wert sagt aus, dass die Korrelation gut ist, ein geringerer sagt aus, dass die Blöcke nicht gut matchen.

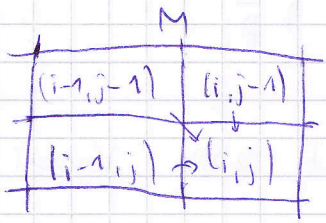
Das Skalarprodukt ist also die Kostenfunktion.



- Dynamic Programming Ansatz aus der Bioinformatik

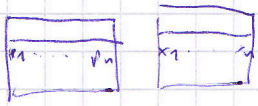


kosten(A,A) = 0
kosten(T,C) = 1



$$M(i,j) = \min\begin{cases} M(i-1,j-1) + \text{kosten}((i-1,j),(i,j)) \\ M(i,j-1) + \text{Verdeckungskosten}, \\ M(i-1,j) + \text{Verdeckungskosten} \end{cases}$$

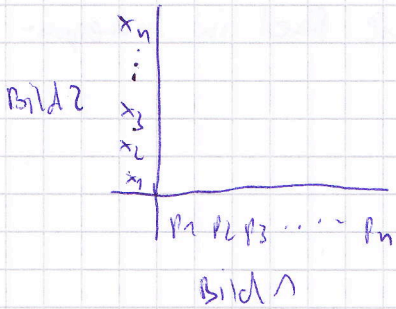
Dynamische Programmierung für Stereo



A) Kostenfunktion

Kosten(x_1, p_1) = -Korrelationsindex (Box(p_1), Box(x_1))

B) Maschinerie



16.6.09

Vorlesung 8

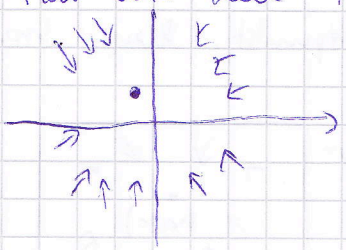
Optischer Fluss

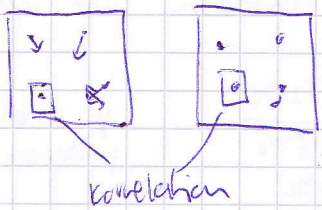
$$\text{Disparität} \approx \frac{d}{D}$$

$$D \approx \frac{d}{\text{Disparität}} = \frac{d}{\text{opt. Fluss}}$$

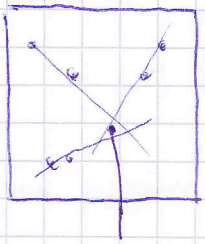
Optischer Fluss ist ein Spezialfall von Stereo Vision.

Man will den Fluchtpunkt berechnen





Korrelation



(x_c, y_c) Fluchtpunkt

Jede Linie ist durch ihre Koeffizienten gegeben.

$$a_i x + b_i y + c_i = 0$$

$$a_i x + b_i y + c_i = 0 \quad \Longleftrightarrow \quad \underbrace{\frac{a_i}{\sqrt{a_i^2+b_i^2}}}_{a_i'} x + \underbrace{\frac{b_i}{\sqrt{a_i^2+b_i^2}}}_{b_i'} y + \underbrace{\frac{c_i}{\sqrt{a_i^2+b_i^2}}}_{c_i'} = 0$$

normalisiert $a_i^2 + b_i^2 = 1$

$$E = \sum_{i=1}^{N} \left| \frac{a_i x_c + b_i y_c + c_i}{\sqrt{a_i^2 + b_i^2}} \right|^2$$

$$= \sum_{i=1}^{N} (a_i' x_c + b_i' y + c_i')^2$$

will man minimieren

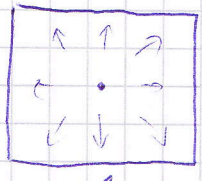
$$\frac{\partial E}{\partial x_c} = 0 \qquad \sum 2 (a_i' x_c + b_i' y_c + c_i') a_i' = 0$$

$$\frac{\partial E}{\partial y_c} = 0 \qquad \sum 2 (a_i' x_c + b_i' y_c + c_i') b_i' = 0$$

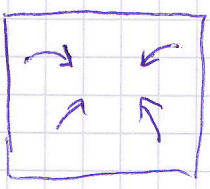
$$\begin{pmatrix} \sum a_i'^2 & \sum a_i' b_i' \\ \sum b_i' a_i' & \sum b_i'^2 \end{pmatrix} \begin{pmatrix} x_c \\ y_c \end{pmatrix} = - \begin{pmatrix} \sum a_i' c_i' \\ \sum b_i' c_i' \end{pmatrix}$$

Das ist der Optimalfall.

Wie ist es mit optischer Verzerrung?



Bei Verzerrung ist der optische Fluss auch gekrümmt



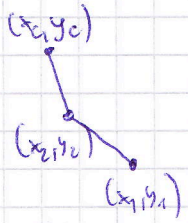
Wir wollen mit dem optischen Fluss die radiale Entzerrung wegbekommen.

$\gamma = a_3 r^2 + a_2 r + a_1$ Abstand zur Mitte des Bildes

Wir wenden den E/M-Algorithmus an.

Erwarten als Fluchtpunkt (x_c, y_c).

Im Maximierungsschritt verbessern wir den Fluchtpunkt. Dazu braucht man

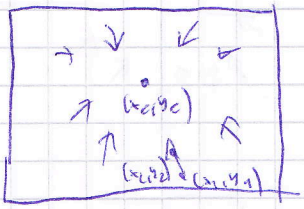


$$g_1 = \frac{\gamma_1 y_1 - \gamma_2 y_2}{\gamma_1 x_1 - \gamma_2 x_2}$$

$$g_2 = \frac{\gamma_c y_c - \gamma_2 y_2}{\gamma_c x_c - \gamma_2 x_2}$$

γ sind Korrekturfaktoren

$a_3 = 0 \quad a_2 = 0 \quad a_1 = 1$



$E = \frac{1}{2} (g_1 - g_2)^2$ Fehler für eine Linie

$E = \sum_j^{\text{opt. Linien}} \frac{1}{2} (g_{1j} - g_{2j})^2$ Gesamtfehler. Den müssen wir minimieren.

$\gamma_1 = a_3 r_1^2 + a_2 r_1 + a_3$

$r_1 = \sqrt{x_1^2 + y_1^2}$

Gradientenabstieg um die Korrekturen zu finden.

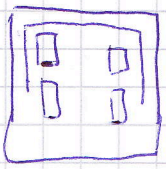
$\Delta a_3 = \lambda \frac{\partial E}{\partial a_3}$ ...

30.6.09

Vorlesung 10

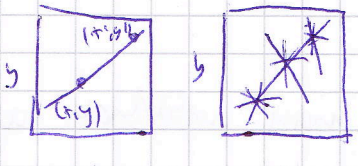
Hough-Transformation

Man hat ein Bild und will z.B. Geraden finden



Die Überlegung ist nun, welche Linien alle durch den Punkt (x,y) gehen können. Also

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \cdot \begin{pmatrix} a \\ b \\ 1 \end{pmatrix} = 0 \qquad ax + by + 1 = 0$$



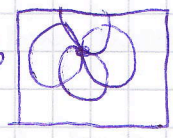
Wenn man das jetzt ganz oft mit vielen Punkten macht, bekommt man im Parameterraum ganz viele Geraden, und dort, wo wirkliche Geraden im Bild sind, hat man Anhäufungen von Schnittpunkten.

Die Hough-Transformation entspricht der dualen Darstellung der Datenpunkte.

Man kann auch Kreise finden.

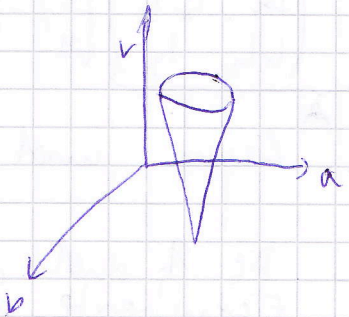
$$(x-a)^2 + (y-b)^2 = r^2$$

Einfachheitshalber fixieren wir den Radius.



Im Parameterraum bilden wieder alle möglichen Mittelpunkte einen Kreis

Mit Betrachtung des Radius ist der Parameterraum 3-dimensional.

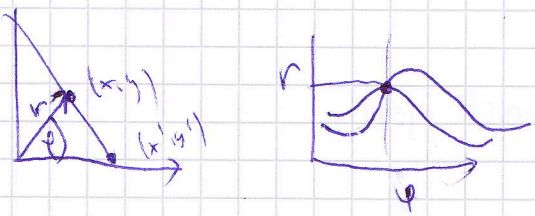


Drei Kegel im Parameterraum, die zu 3 auf einem Kreis liegenden Punkten gehören, schneiden sich in einem Punkt.

Man kann die Hough-Transformation auf beliebige Objekte verallgemeinern.

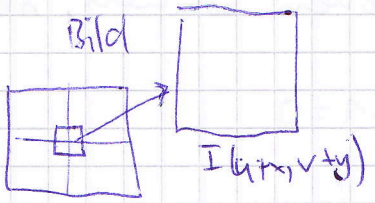
Für Linien gibt es auch noch eine andere Darstellung

$$+ \cos\varphi + y \sin\varphi = r$$



Harris Corner Detector

Ist ein Feature.



$I(u,v)$ wir verschieben um (x,y)

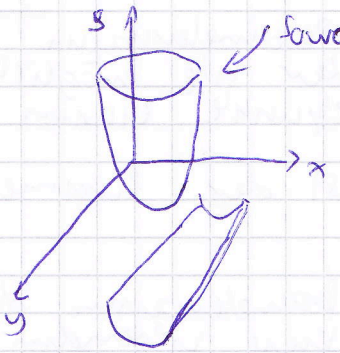
$$S = \sum_{u,v} \left(I(u+x, v+y) - I(u,v) \right)^2$$

Das könnte man natürlich auch Gewichten

Das Bild $I(u+x, v+y)$ kann über eine Taylor-Reihe approximiert werden: $I(u+x, v+y) \approx I(u,v) + \frac{\partial I(u,v)}{\partial x} x + \frac{\partial I(u,v)}{\partial y} \cdot y$

$$S = \sum_{u,v} \left(\underbrace{\frac{\partial I(u,v)}{\partial x}}_{I_x} x + \underbrace{\frac{\partial I(u,v)}{\partial y}}_{I_y} y \right)^2$$

Es ergibt sich sowas:

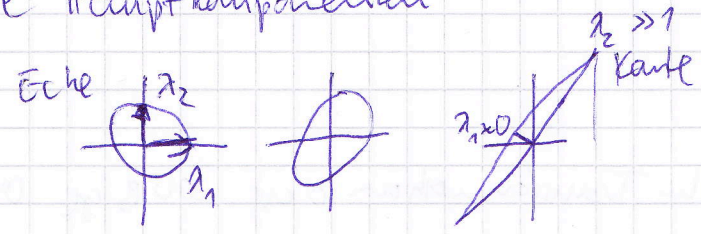


sowas interessiert uns

$$S = \sum_{u,v} (x,y) \begin{pmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

$$= (x,y) \underbrace{\sum_{u,v} \begin{pmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{pmatrix}}_{A} \begin{pmatrix} x \\ y \end{pmatrix}$$

Wir suchen von allen auf dem Raum und betrachten die Hauptkomponenten



Die λ_i sind die Eigenwerte der obigen Matrix A

Die uninteressanten Fälle sind:

$\lambda_1 = \lambda_2 \approx 0$ $\Rightarrow$ alles homogen

$\lambda_1 \approx 0$ $\Rightarrow$ Kante

$\lambda_2 \approx 0$ $\Rightarrow$ Kante

Der Harris corner detector macht jetzt

$$\lambda_1 \lambda_2 - K(\lambda_1 + \lambda_2)^2$$

$\det A = \lambda_1 \lambda_2$

$\text{Spur } A = \lambda_1 + \lambda_2$

Damit ergibt sich $\det A - K(\text{Spur } A)^2$

↑

empirisch testen