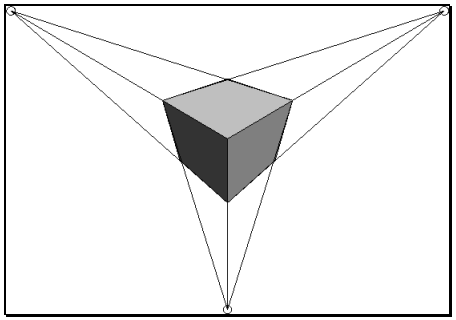
Vergleich von Algorithmen zur Fluchtpunktbestimmung

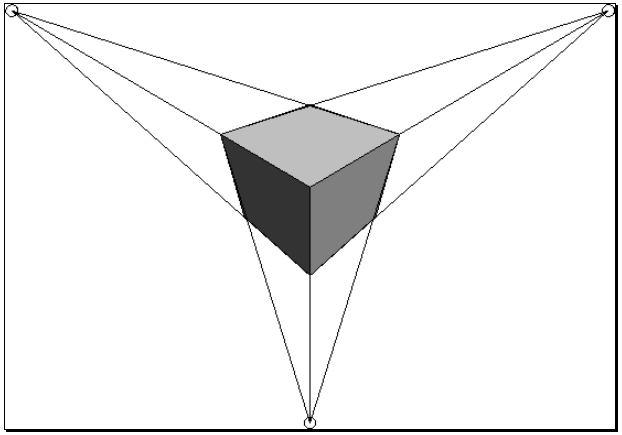


Studienarbeit

Heiko Abraham

- Aufgabenstellung
- Fluchtpunkte
- Algorithmen
- Ergebnisse
- Ausblick

abstract:



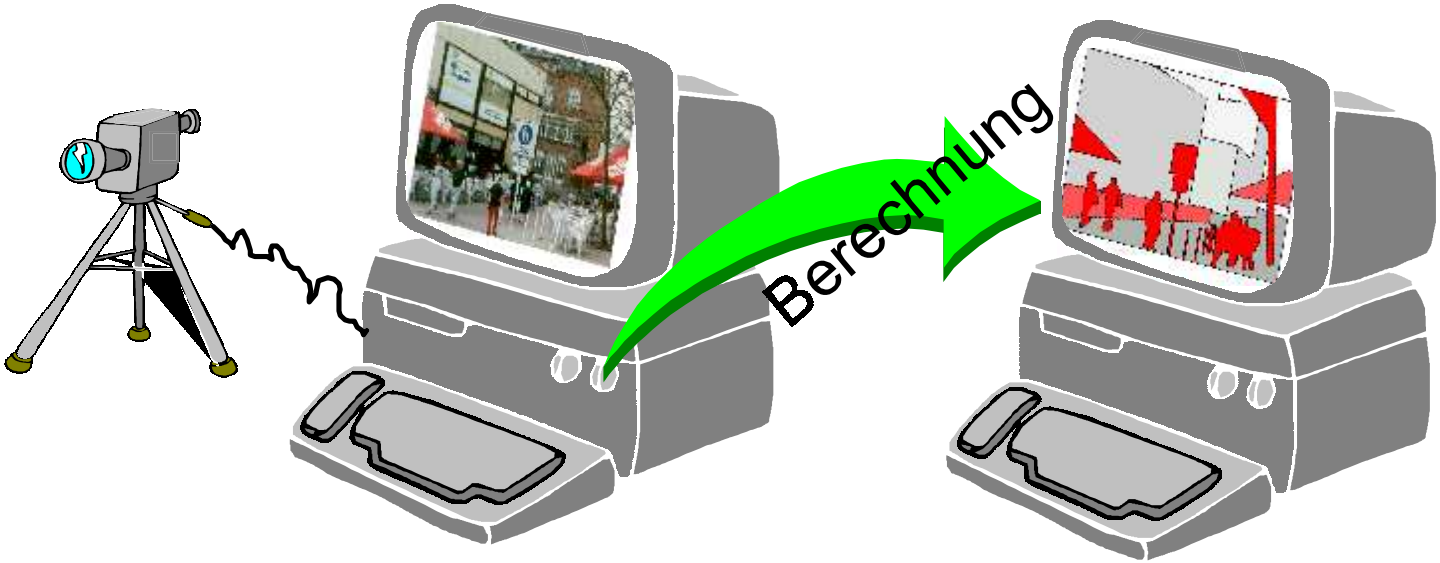
Mittels digitaler Bildkameras können 2D-Bilder von einer 3D-Bildumgebung gewonnen werden. Diese Projektion entspricht der monokularen Monographie. Wertvolle Informationen über die Struktur der Umgebung, ausgehend von den gewonnenen 2D-Bildern, können die sogenannten Fluchtpunkte liefern. Parallele Linien des 3D-Raum konvergieren unter perspektivischer Projektion in einem gemeinsamen Punkt der Bildebene, dem Fluchtpunkt. Seit die Bedeutung der Fluchtpunkte für die Interpretation klar wurde, sind eine Reihe von Algorithmen zu deren Bestimmung entwickelt worden.

In diesem Vortrag werden die grundlegenden Algorithmen kurz vorgestellt. Die praktisch ermittelten Laufzeiten und Charakteristika werden gegenübergestellt.

Aufgabenstellung:

Implementierung (in Komponenten) und Vergleich von Algorithmen zur Fluchtpunktbestimmung.

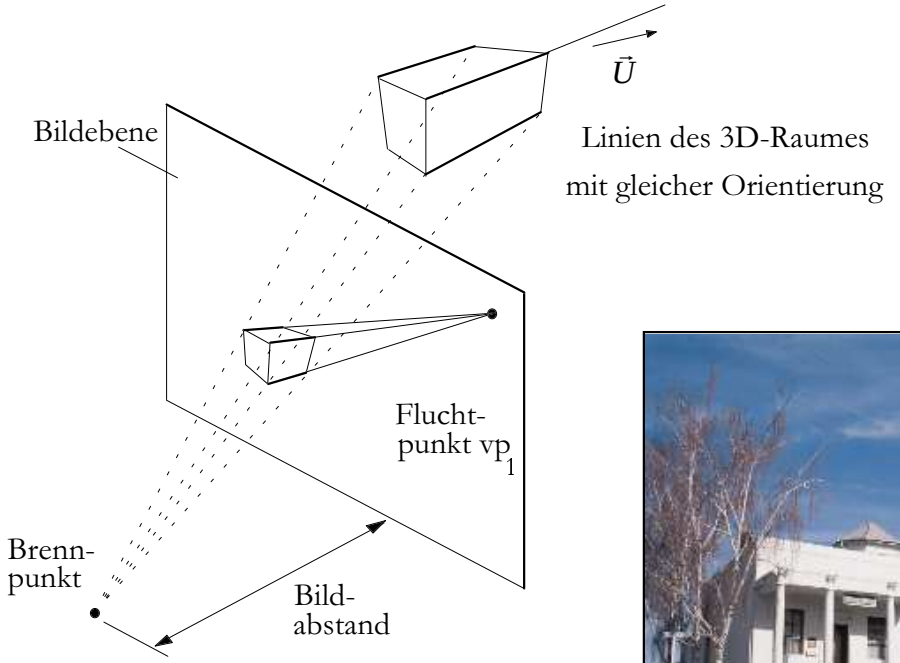
Endziel:
Gewinnung von Informationen aus einem Bild
zur Interpretation von Bildstrukturen.

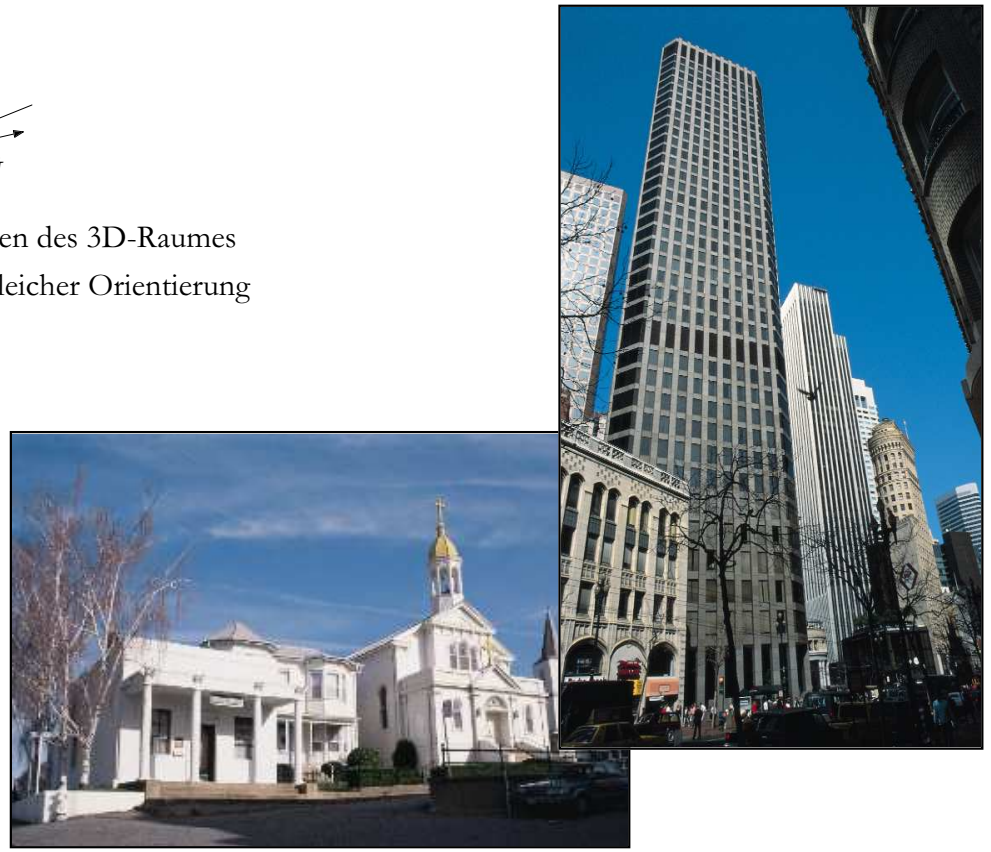


Teilziel dazu:
Bestimmung der Fluchtpunkte

Der Fluchtpunkt unter der Zentralprojektion:

Liniensegmente des 3D-Raumes, die jeweils gleiche Orientierung im Raum besitzen, schneiden sich unter der Zentralprojektion in einen Punkt der Bildebene, dem Fluchtpunkt.





Algorithmen zur Fluchtpunktbestimmung
nach *Erstautoren*, Veröffentlichungsjahr

Bestimmung der Schnittpunkte I_S durch

Hough-Array-Verfahren

❶ *S.T. Barnard*, 1983
(uniformes Hough-Array)

❷ *R.T. Collins*, 1989
(plane-fit - Verfahren)

❸ *E. Lutton*, 1994
(Non-uniformes Hough-Array)

direktes berechnen

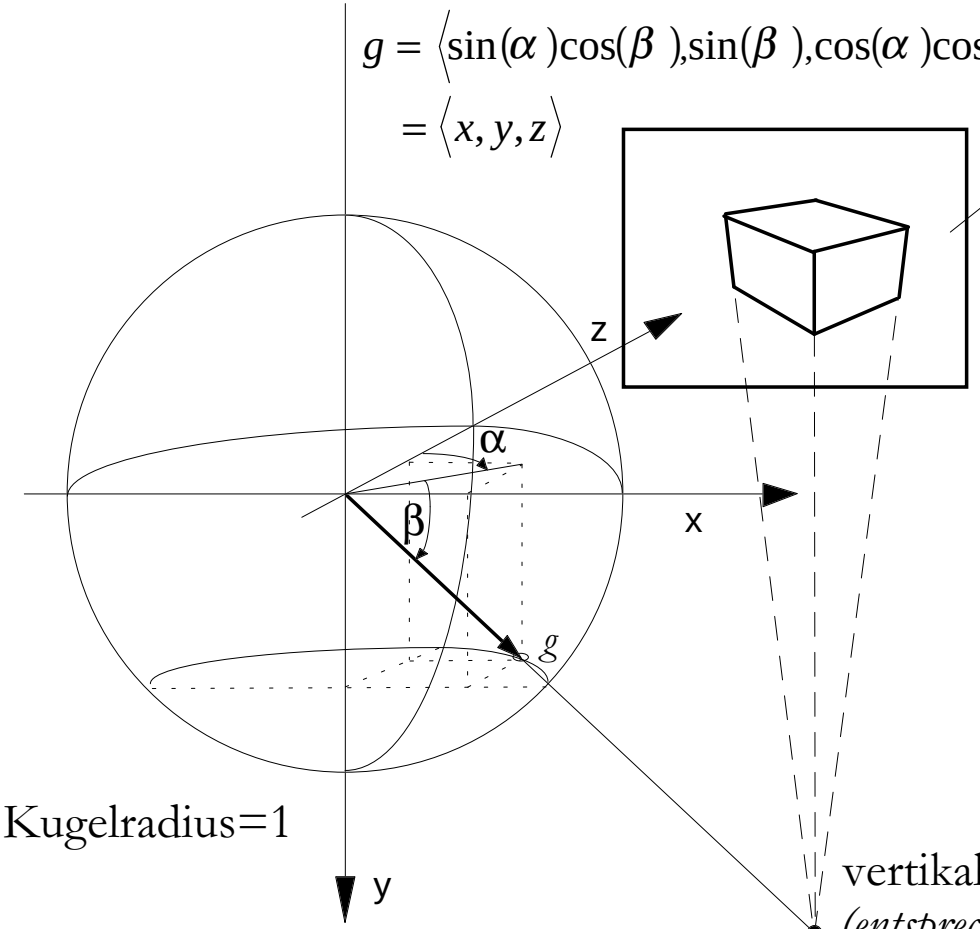
❹ *M.J. Magee*, 1984
(I_S nach Winkelabstand zusammenfassen)

❺ *P. Gamba*, 1996
(a priori Wissen verwenden)

❻ *P.Palmer*, 1993
(Winkel-Parametriesierung nutzen)

Abgeschlossenheit des Suchraumes durch Transformation auf Kugeloberfläche

$$g = \langle \sin(\alpha)\cos(\beta), \sin(\beta), \cos(\alpha)\cos(\beta) \rangle$$
$$= \langle x, y, z \rangle$$



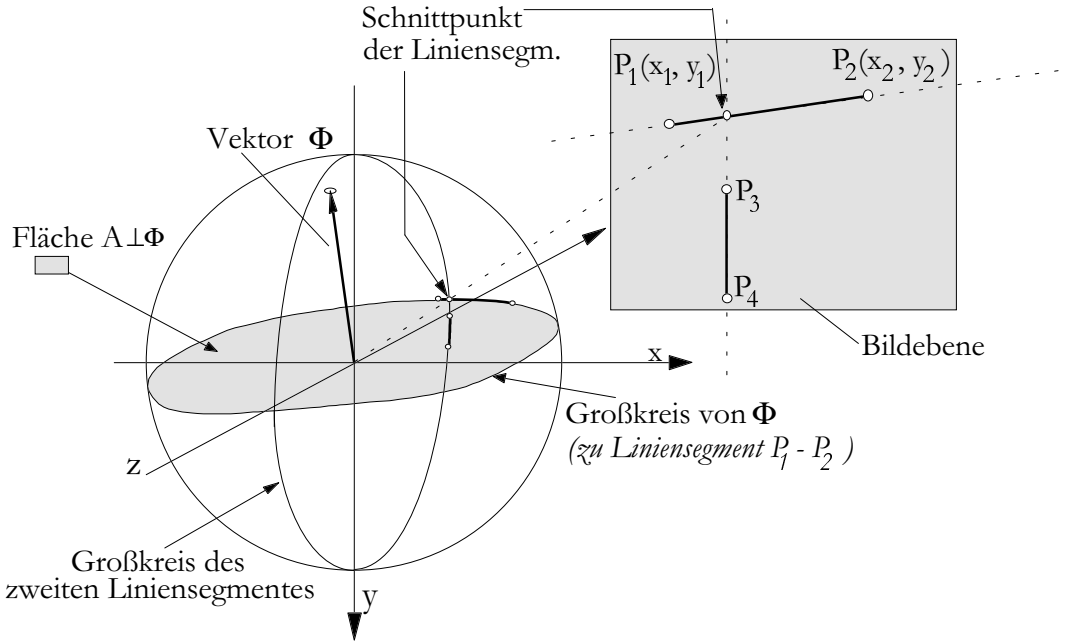
Bildebene
(parallel zur x-y-Achse)

- Keine (endliche) Bildebene ist groß genug, um jeden Fluchtpunkt darzustellen.

Kugelradius=1

vertikaler Fluchtpunkt
(entsprechend in Bildebene)

S.T. Barnard, Nov. 1983



- Jedes Liniensegment wird als Linie auf die Kugeloberfläche projeziert (Großkreis entsteht).
- Die Kugeloberfläche wird in Flächenstücke (*bins*) unterteilt, gemäß gleicher Winkel.
- Jeder belegte *bin*-Eintrag wird inkrementiert (Hough-Array).
- Lokale Maximasuche im Hough-Array.

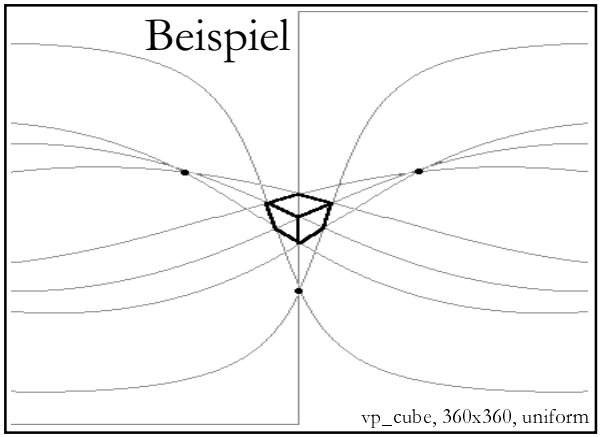
Laufzeit: O(n)

$$p_i = (x_i, y_i, f)^T$$

Vektor Φ: $$\Phi = \frac{p_1 \times p_2}{|p_1||p_2|} = \langle \varphi_x, \varphi_y, \varphi_z \rangle$$

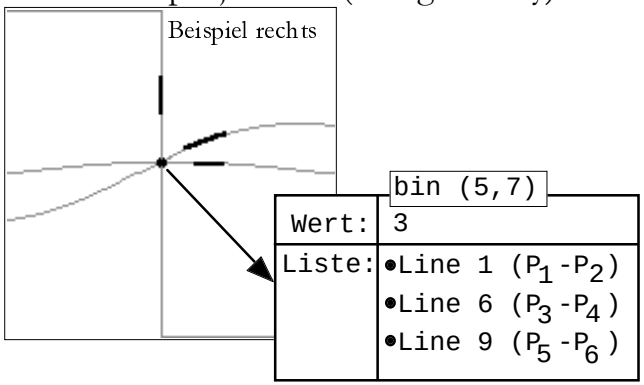
Großkreis: $g \cdot \Phi = 0$

$$\beta(\alpha, \Phi) = \arctan\left(\frac{-\Phi_x \cdot \sin(\alpha) - \Phi_z \cdot \cos(\alpha)}{\Phi_y}\right) \quad |\Phi_y \neq 0$$



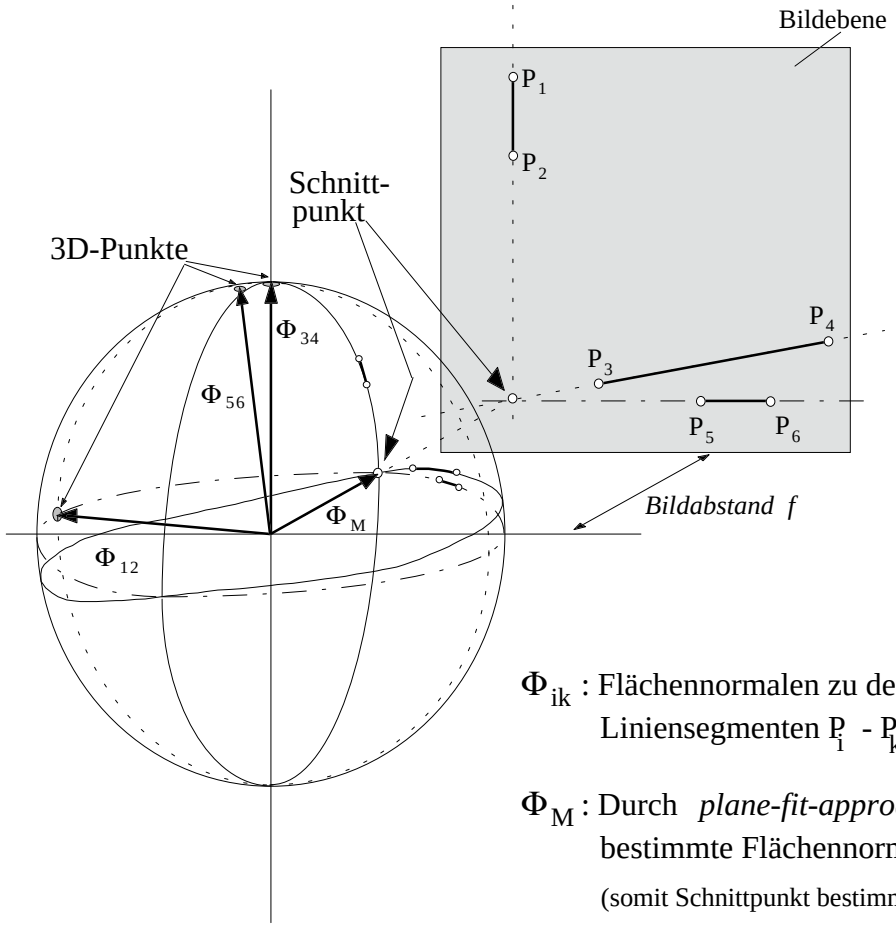
R.T. Collins und R. Weiss, Juni 1989

- Liniensegmente wie bei Barnard auf Kugeloberfläche projezieren (Hough-Array)



- Zu jeden Array-Eintrag in eine Liste Info über Herkunft speichern.
- In Array lokale Maxima suchen (bzw. Region).
- Zu jedem lokalen Maxima *plane-fit-approach* durchführen (Listeninformation nutzen). D.h. Ebene durch alle 3D-Punkte *fitten*.(k viele)

Laufzeit:
je nach Optimierungsverfahren: O(n)...O(n+k²)

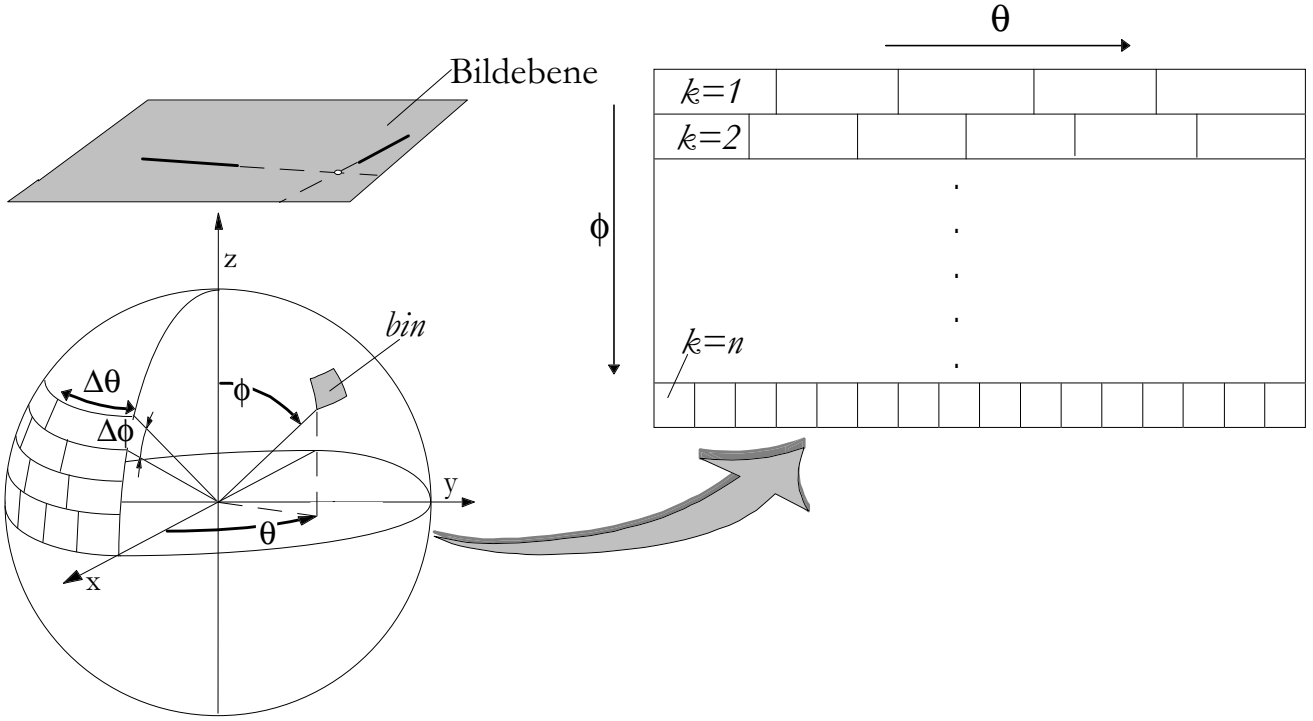


Φ_{ik} : Flächennormalen zu den Liniensegmenten P_i - P_k

Φ_M : Durch *plane-fit-approach* bestimmte Flächennormale

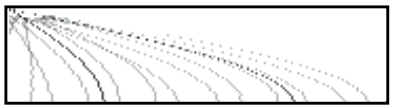
(somit Schnittpunkt bestimmt)

E. Lutton, H. Maître und J.Lopez-Krahe, April 1994



Beispiel: cube

uniform size:180x45



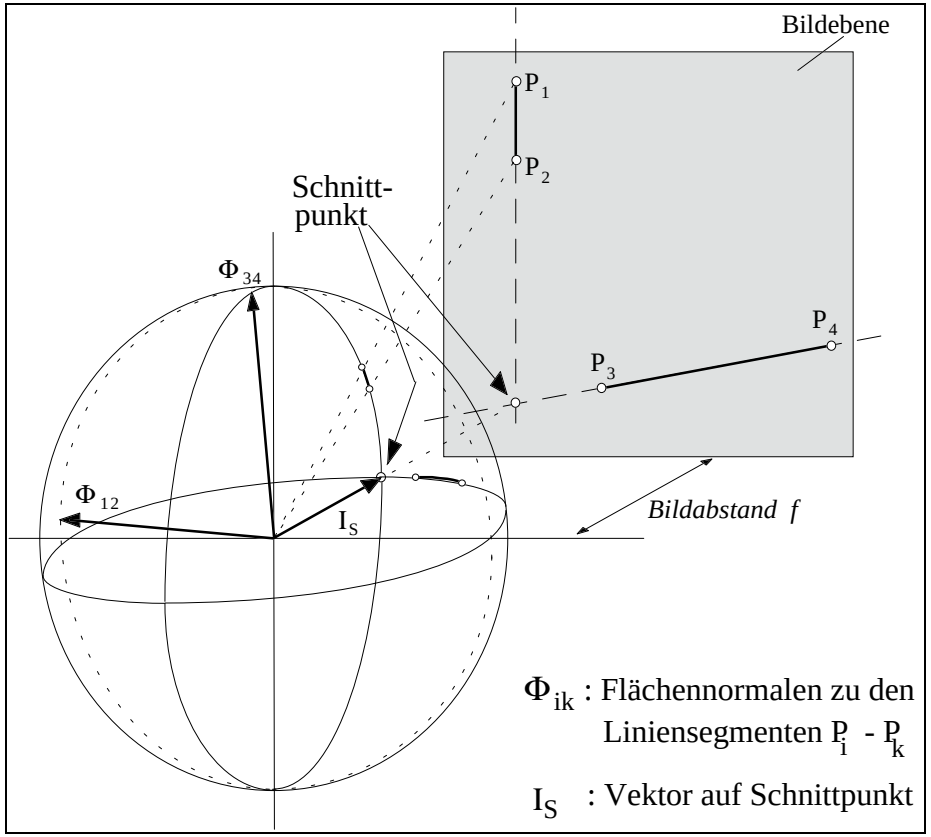
non-uniform

uniform, entzerrt

- Kugeloberfläche wird in *bin* eingeteilt mit ca. **gleichem Flächeninhalt** → andere Parametriesierung
  $\Delta\phi$ konstant, $\Delta\theta(\phi)$ variabel, der Parameter $\Delta\theta(90°)$ legt den Flächeninhalt eines *bin* fest
- dann weiter wie bei *Barnard* verfahren (Großkreise in Hough-Array eintragen, Werte inkrementieren)
- lokale Maximasuche (Implementierung schwierig) liefert die Fluchtpunkte

Laufzeit: O(n)

M.J. Magee und J.K. Aggarwal, 1984



- Schnittpunkte direkt berechnen

$$P_i = (x_i \; y_i)^T \qquad p_i = (x_i \; y_i \; f)^T$$

$$\Phi_{ik} = \frac{p_i \times p_k}{|p_i \times p_k|}$$

$$I_s = (I_x \quad I_y \; I_z)^T = \frac{\Phi_{12} \times \Phi_{34}}{|\Phi_{12} \times \Phi_{34}|}$$

- so entstehen $O(n^2)$ Schnittpunkte
- Finde Schnittpunkt-Cluster.
- Dazu Schnittpunkte zusammenfassen, wenn ihr Abstand δ zueinander kleiner einer Winkeldifferenz ist
- Laufzeit je nach Implementierung in $O(n^2)$ bis $O(n^4)$

$$\delta = \arccos\left(\cos\left(\frac{\pi}{2}-\beta_1\right)\cos\left(\frac{\pi}{2}-\beta_2\right)+\sin\left(\frac{\pi}{2}-\beta_1\right)\sin\left(\frac{\pi}{2}-\beta_2\right)\cos(\alpha_1-\alpha_2)\right)$$

P. Gamba, A. Mecocci und U. Salvatore, 1996

- ebenfalls direktes Berechnen der Schnittpunkte, aber

  ♦ Verwendung von *a priori* Wissen:

  ⇒ Selektierung aller horiz. / vert. Liniensegmente

  (drastische Reduzierung der Linienzahl n zu k)

  ♦ dann Schnittpunkte direkt bestimmen {in $O(k^2)$ }

  ♦ Verwendung der *voting rule*

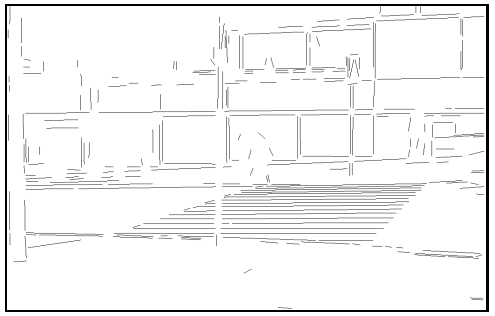
  ⇒ Schnittpunkte sich kreuzender Linien verwerfen

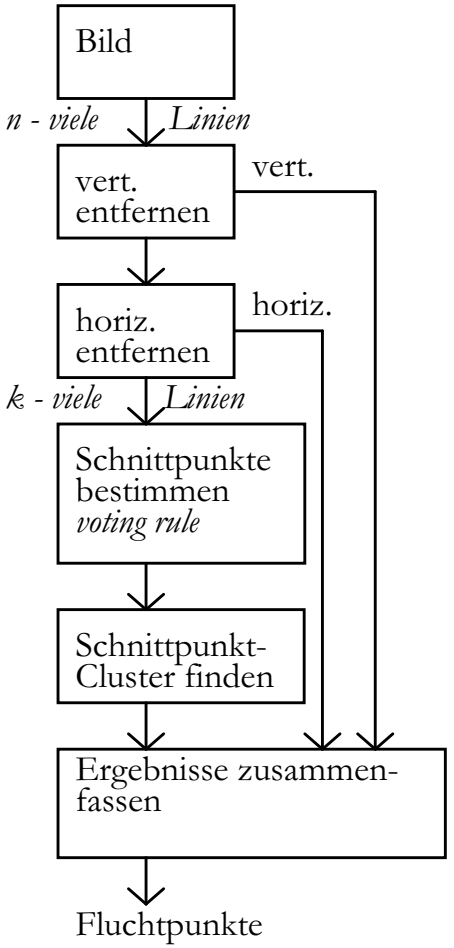
  ⇒ Jeden Schnittpunkt Gewichtswert vergeben

  - nach Länge der Liniensegmente
  - nach Entfernung der Linien zueinander

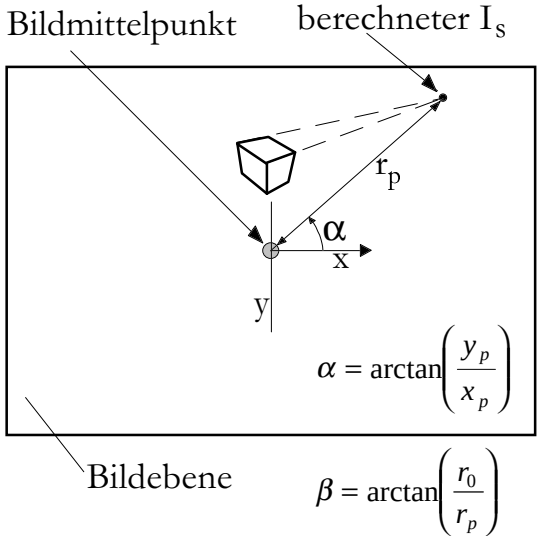
  ♦ Zusammenfassen der einzelnen Ergebnisse

  Laufzeit: $O(n^2)$





P. Palmer und A. Tai, 1993

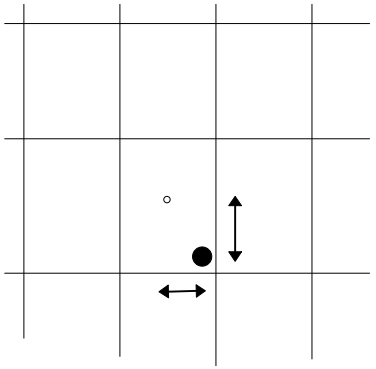


- alle (zueinander) parallele Liniensegmente entfernen
- Schnittpunkte direkt bestimmen
  (wie *Magee* mittels Kreuzprodukt)
- alle Schnittpunkte aus Bildzentrum entfernen
- alle Schnittpunkte in Hough-Array eintragen
  (Cluster finden)
  - ⇒ nutzt andere Parametriesierung
  - ⇒ verwendet geringe Hough-Array Auflösung
  - ⇒ inkrementiert jeden Array-Eintrag entsprechend einer *voting kernel*-Funktion

$$K(\delta\alpha, \delta\beta) = F\left(\frac{\delta\alpha}{K_\alpha}\right) \cdot F\left(\frac{\delta\beta}{K_\beta}\right)$$

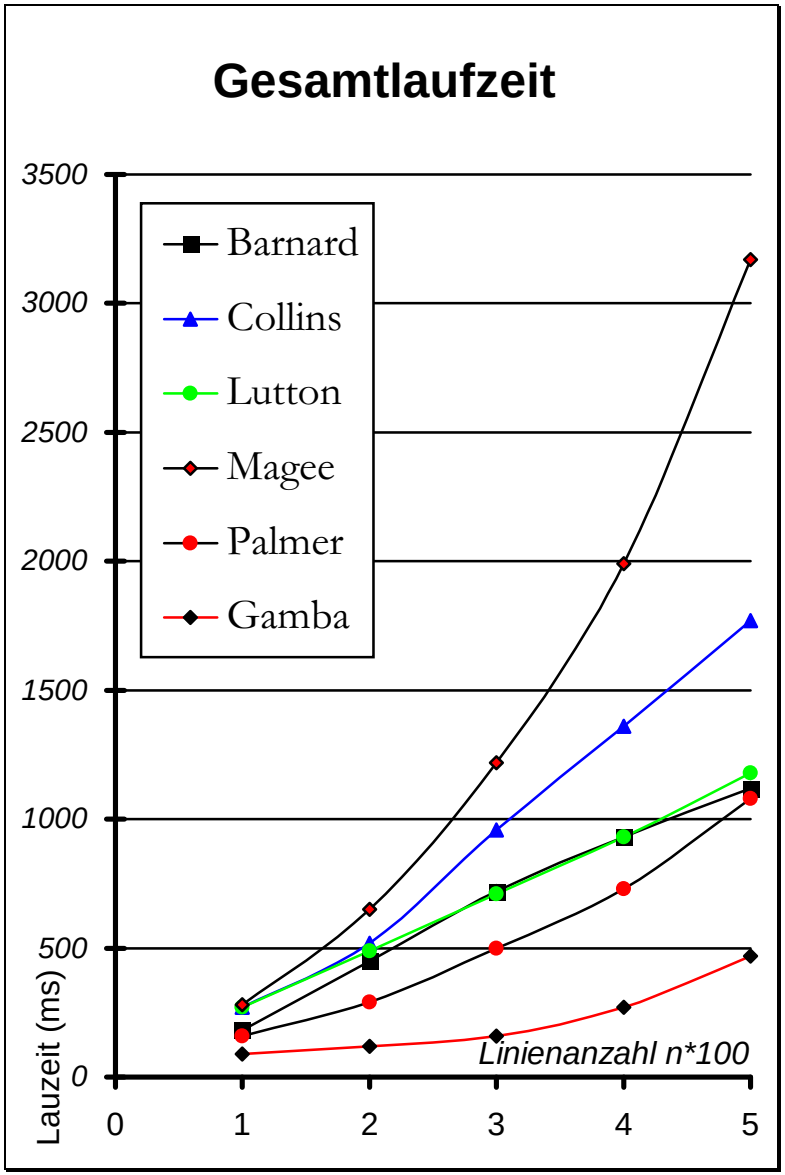
D.h. Eintrag stärker votiert, wenn Koordinaten dem Mittelpunkt des *bin* entsprechen.

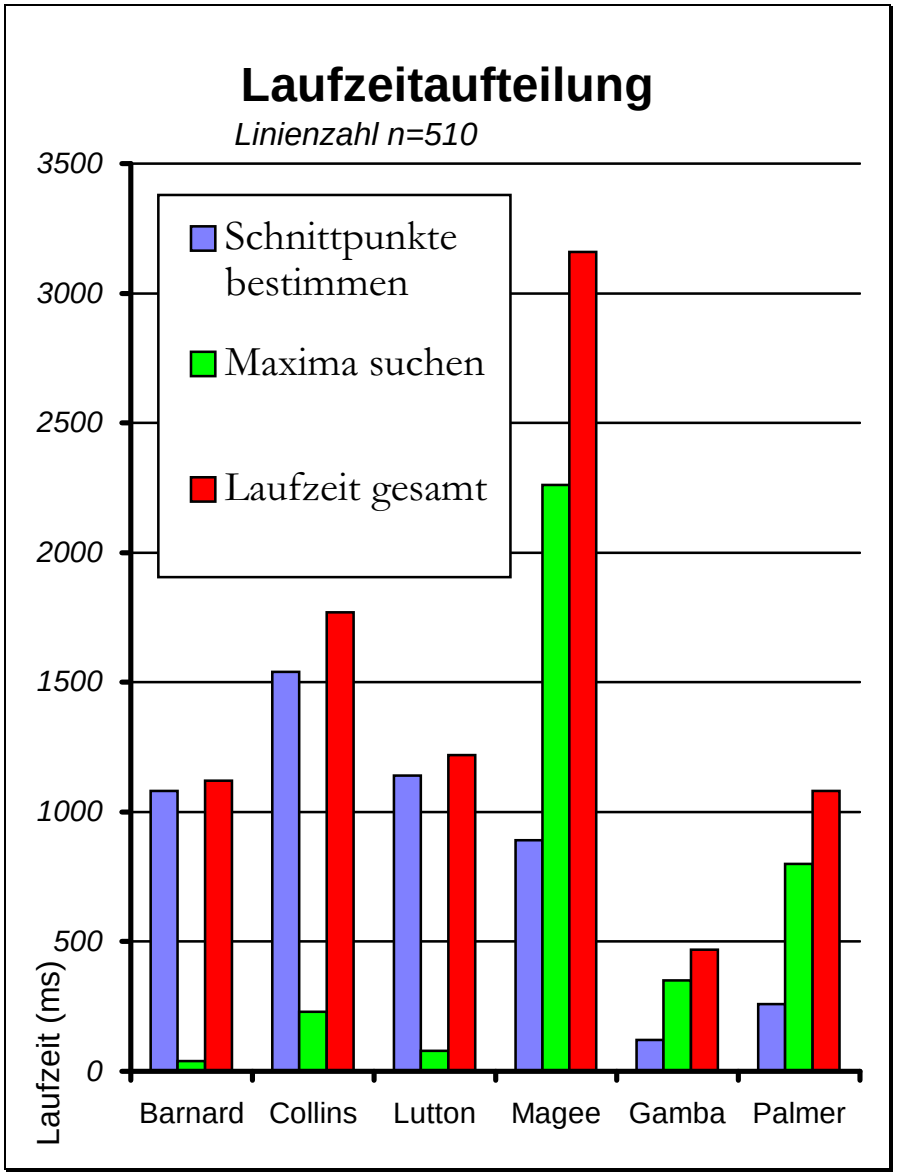
$$F(x) = 1 - 2x^2 + x^4 \text{ wenn } |x| < 1, \text{ sonst } F(x) = 0$$

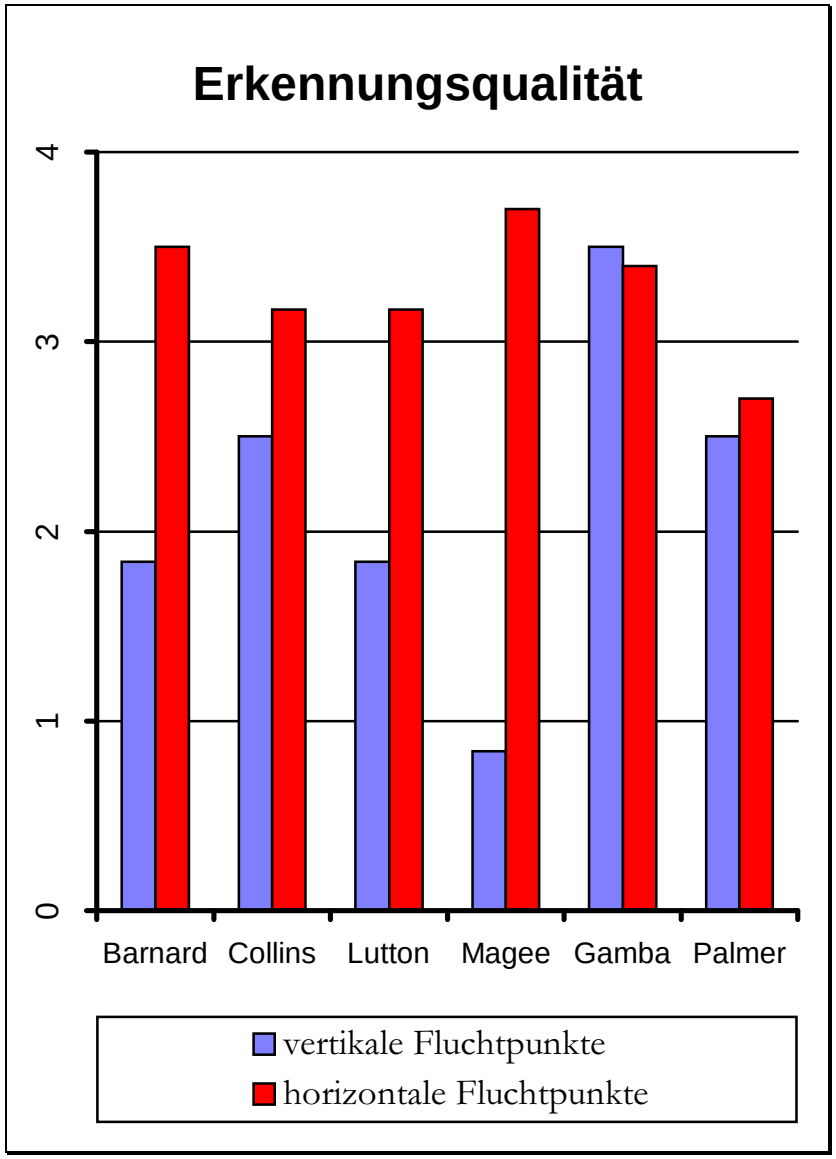


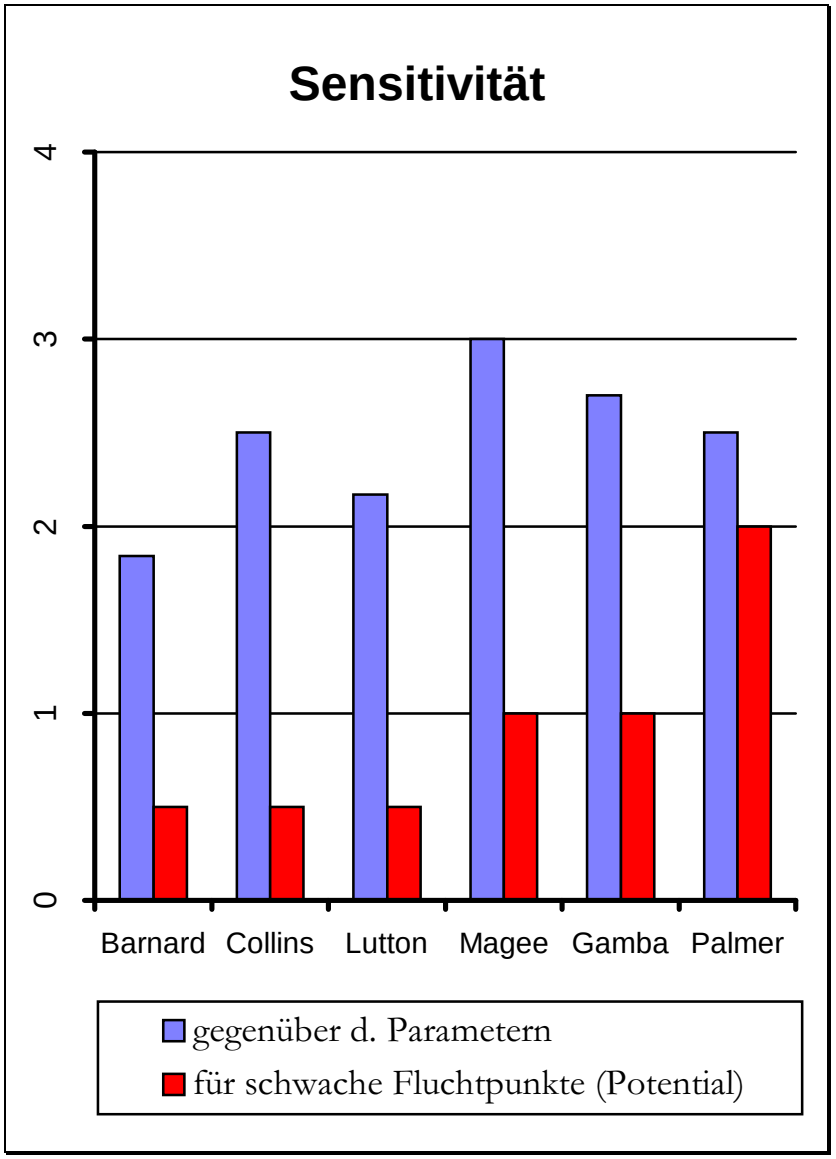
- bestimme lokale Maxima in Hough-Array
- wenn lokales Maxima bestimmt, dann
  - ⇒ genauer Bestimmung durch Einbeziehung der *voting kernel*-Funktion

Laufzeit: O(n²)


Gesamtlaufzeit
3500
3000
2500
2000
1500
1000
500
0
Lauzeit (ms)
Barnard
Collins
Lutton
Magee
Palmer
Gamba
0
1
2
3
4
5
Linienanzahl n*100


Laufzeitaufteilung
Linienzahl n=510
3500
3000
2500
2000
1500
1000
500
0
Laufzeit (ms)
Schnittpunkte bestimmen
Maxima suchen
Laufzeit gesamt
Barnard
Collins
Lutton
Magee
Gamba
Palmer





Bewertung in Punkten, Durchschnitt aus sechs Testbildern, höhere Punktezahl entspricht höherer Qualität

Ausblick

- generelles verwenden von *a priori*-Wissen
  (Reduzierung des Informationsgehaltes auf wichtige Fakten, Parallelisierung)

- Suche nach lokalen Maxima verbessern
  (z.B. Anstieg der Umgebung mit einbeziehen)

- Parametriesierung des Hough-Array ändern (vertikale Erkennungsqualität)

- Bewertungskriterium einführen
  Qualität des erkannten Liniensegmentes für *a priori*-Auswahl verwenden

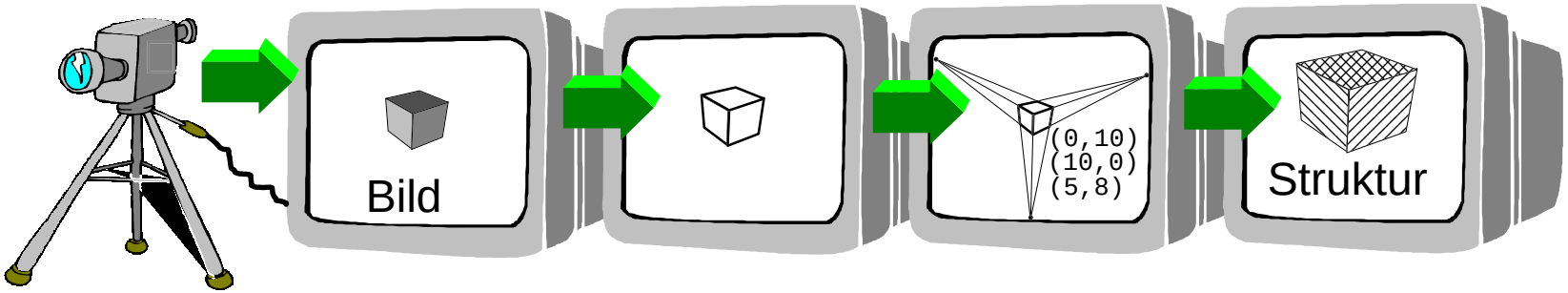
Voraussetzung:
Ein Bild der Umgebung, z.B. erzeugt durch

- digitale Bildkamera, oder
- SLR-Kamera und Flachbettscanner

Ziel:
Gewinnung von Informationen aus einem Bild zur Interpretation der Strukturen.

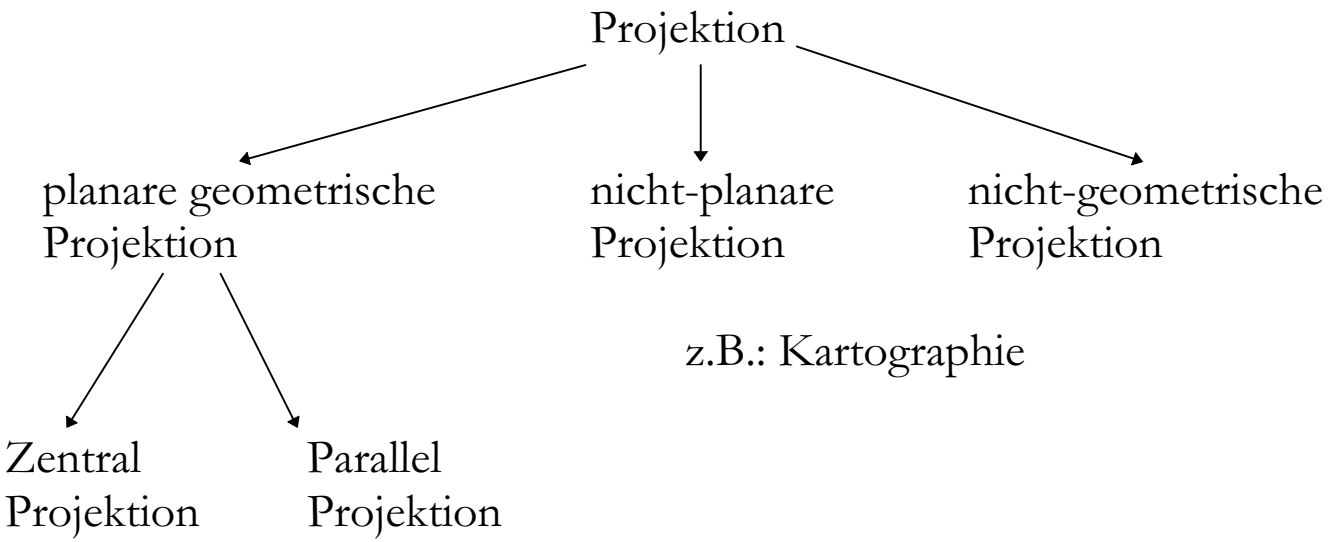
Teilziel:
Bestimmung der Fluchtpunkte des jeweiligen Bildes.

Vorgehensweise:

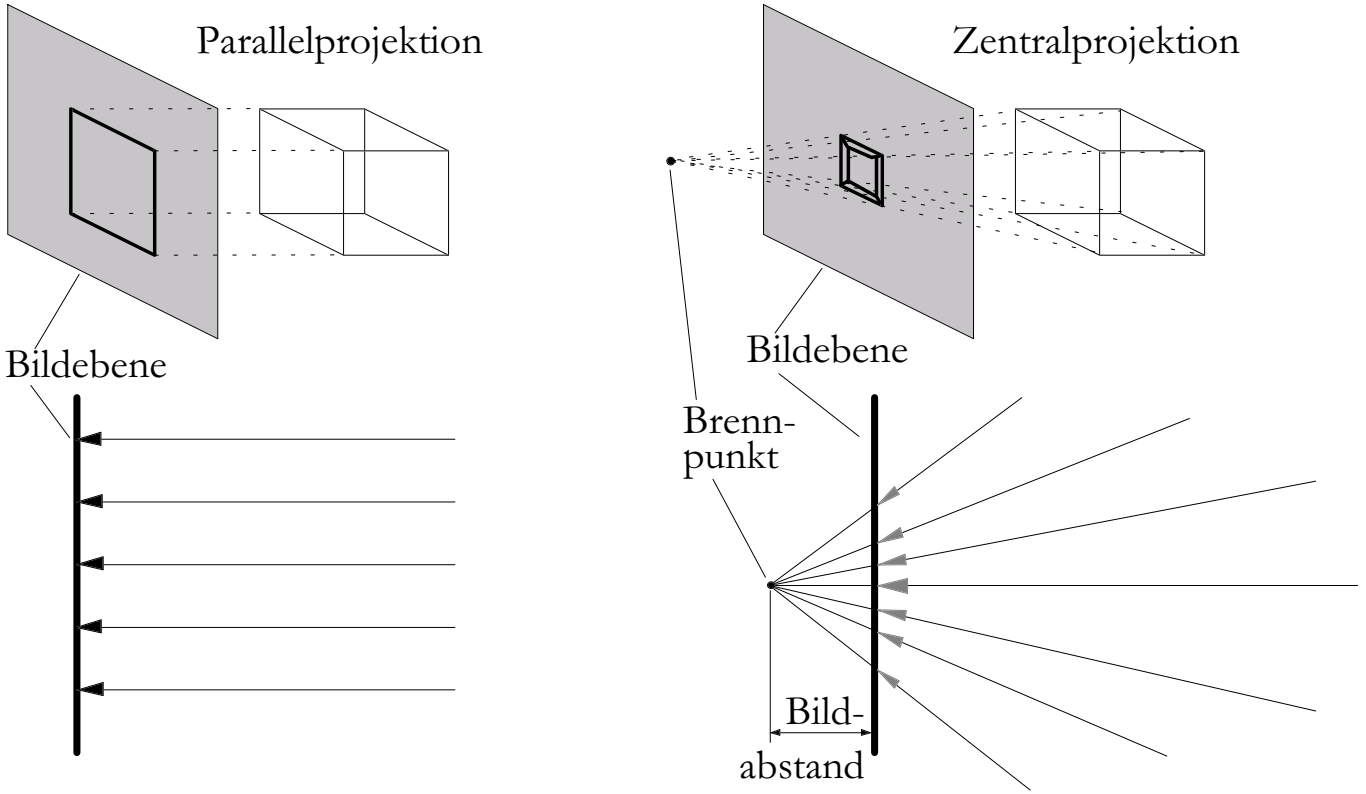


Grundlage, die Art der Projektion

- es gibt viele verschiedene Projektionsmethoden
- Um Informationen aus einem 2D-Bild über die Struktur des Raumes zu gewinnen, ist es wichtig zu wissen, welche Projektion verwendet wurde.



Beispiel: Zentral- und Parallelprojektion

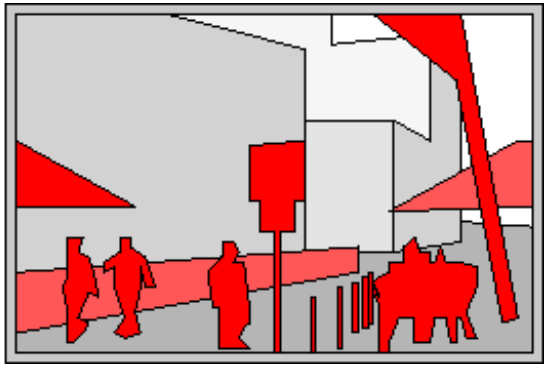


Das menschliche Auge und ein photographisches System verhalten sich ähnlich, wie die Zentralprojektion.

Wozu können Fluchtpunkte dienen?

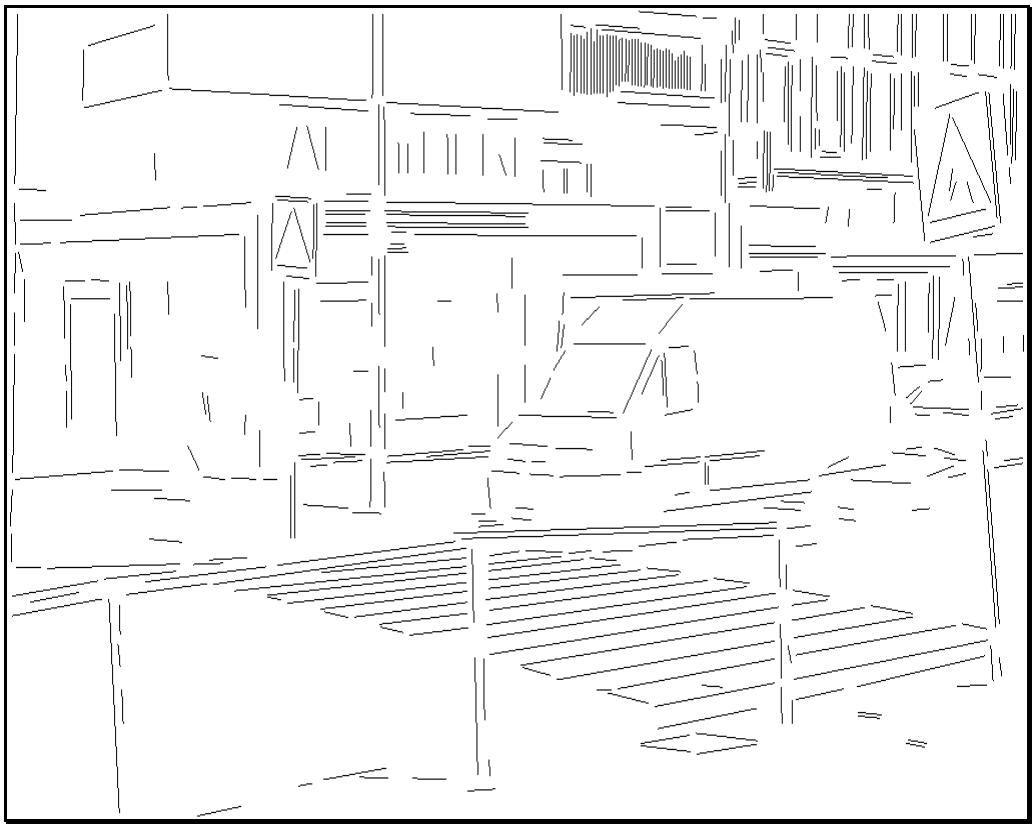
- Implementierung einer Kamera-Kalibrierung

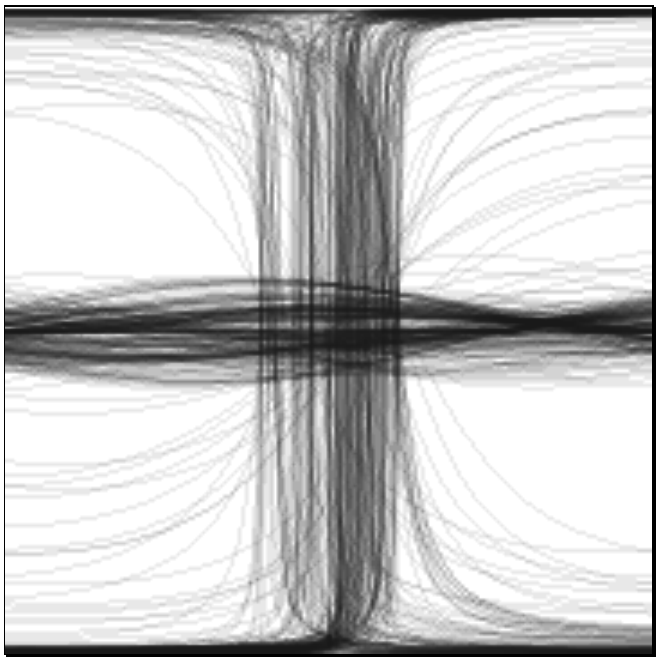
- 3D-Rekonstruktion monokularer Aufnahmen unter Benutzung von Fluchtpunkten



- Bestimmung der drei Koordinatenhauptachsen der 3D-Umgebung entsprechend der Hauptfluchtpunkte

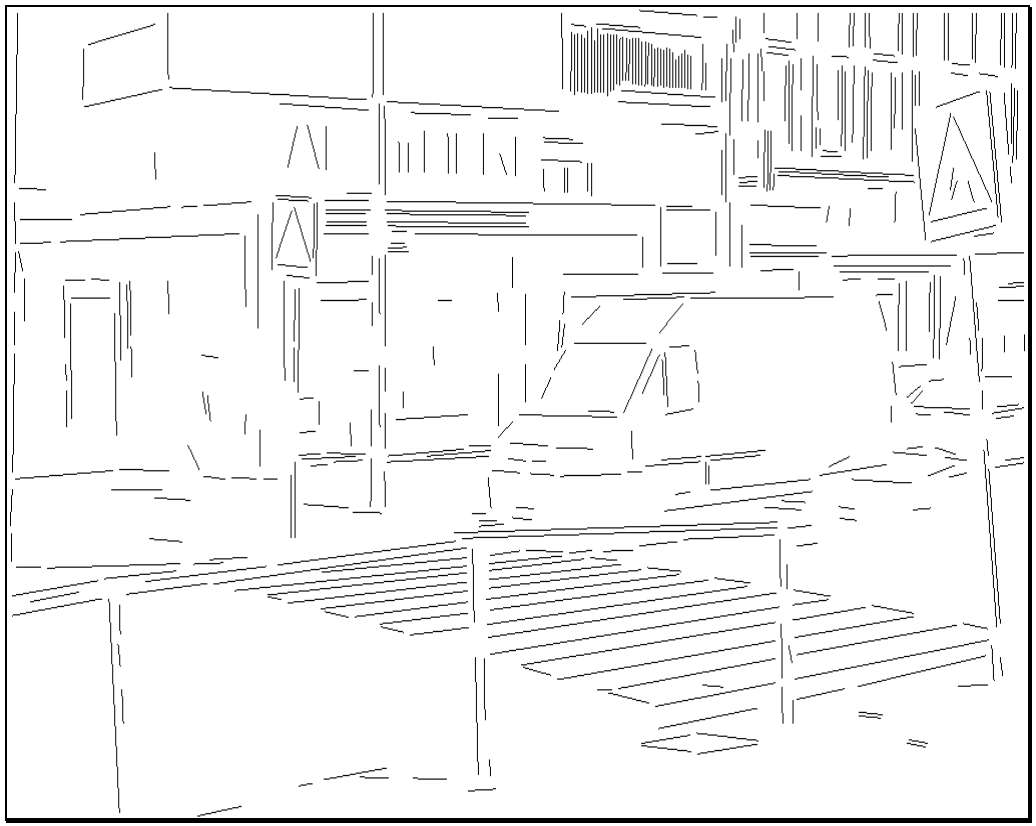
Wie ein Hough-Array nach Barnard aussehen kann:

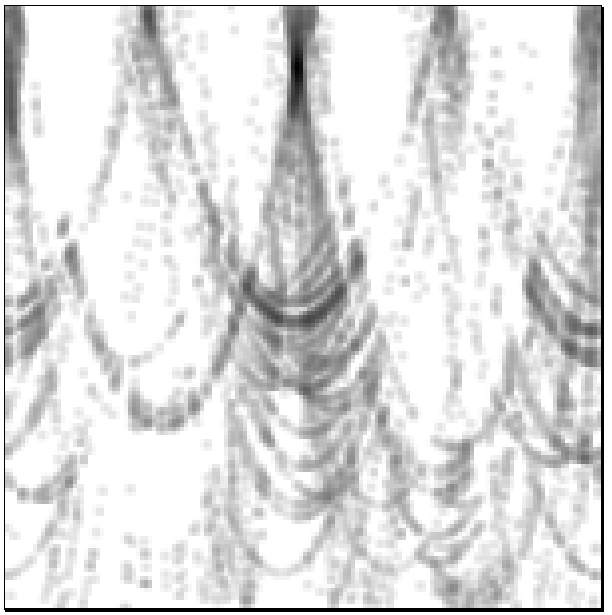




- Typisches Liniensegmentbild
- zweiter Fluchtpunkt nur schwach
- entsprechendes Hough-Array nach Barnard

Wie ein Hough-Array nach Palmer aussehen kann (Schnittpunkt - Cluster finden)





- Typisches Liniensegmentbild
- zweiter Fluchtpunkt nur schwach
- entsprechendes Hough-Array nach Palmer

Literaturangabe:

[1] S.T. Barnard. Interpreting perspective images. *Arteficial Intelligence*, 21(4):435-462, Nov. 1983.

[2] R. T. Collins und R. Weiss. An Efficient and Accurate Method for Computing Vanishing Points. *Image Understanding and Maschine Vision*, Technical Digest Serie, Optical Society of America, Juni 1989.

[3] M. A. Fischler, S. T. Barnard, R. C. Bolles, und M. Lowry. Modelling and using physical constraints in scence analysis. In Kaufmann, Hrsg*., Proceedings of the National Conference on Artifacial Inteligence*, Seite 30-35, Los Altos, Cal., 1982

[4] P. Gamba, A. Mecocci, und U. Salvatore. Vanishing point detection by a voting scheme. In P. Delogne, Hrsg., *Third International Conference on Image Processing*, volume 2, Seite 301-304, Lausanne, Switzerland, Sept. 1996. IEEE Signal Processing Society, Ceuterick, Leuven.

[5] E. Lutton, H. Maître, und J. Lopez-Krahe. Contribution to the determination of vanishing points using hough transform. *IEEE Transactions on Pattern Analysis Maschine Intelligence*, 16(4):430-438, Apr. 1994.

[6] M. J. Magee und J. K. Aggarwal. Determining vanishing points from perspective images. *Computer Vision, Graphics an Image Processing*, 26:256-267, 1984

[7] P. Palmer und A. Tai. An optimised vanishing point detector. In J. Illingworth, Hrsg., *Processing of the 1993 British Maschine Vision Conference*., Seite 529-538, Sheffield, UK, 1993. BMVA Press.

[8] J. D. Foley, A. van Dam, S. K. Feiner, J. F. Hoghes und R. L. Phillips, Grundlagen der Computergrafik, *Addison-Wesley Verlag*, 1994