

Robotik

Priv.-Doz. Dr. Thomas Wiemann
Institut für Informatik
Autonome Robotik

SoSe 2021



Von Kanten zu Ecken

- ▶ Kanten haben einen hohen Grauwertgradienten in einer Richtung
- ▶ Kanten bestehen aus Linienzügen von vielen Pixeln
- ▶ Die bisher vorgestellten Filter versagen an Ecken
- ▶ Ecken zeichnen haben einen starken Gradienten in zwei Richtungen
- ▶ Ecken / Herausstehende Punkte sind genau lokalisiert
- ▶ Anforderungen an einen Operator zur Eckenerkennung:
 - Regionen mit homogener Textur werden als nicht interessant erkannt
 - Kantenpunkte sollen einen starken Gradienten in einer Richtung aufweisen
 - Punkte an Ecken und herausragende Punkte sollen signifikante Gradienten in beiden Richtungen aufweisen
- ▶ „Harris Operator“

Harris Operator (1)

- ▶ Betrachte in einem Grauwertbild I eine Region der Größe (u, v)
- ▶ Verschiebe die Region im Bild und betrachte die gewichtete Summe der Grauwertunterschiede:

$$S_I(x, y) = \sum_u \sum_v w(u, v) (I(u, v) - I(u - x, v - y))^2$$

- ▶ Leite die Harris-Matrix \mathbf{A} durch eine Taylorreihenapproximation von S_I ab:

$$S_I(x, y) \approx S_I(0, 0) + (x, y) \nabla S_I + \frac{1}{2}(x, y) \mathbf{A} \begin{pmatrix} x \\ y \end{pmatrix}$$

- ▶ Vereinfacht ergibt sich also

$$S_I(x, y) \approx \frac{1}{2}(x, y) \mathbf{A} \begin{pmatrix} x \\ y \end{pmatrix} = 0$$

Harris Operator (2)

- ▶ S_I ist eine Funktion der Grauwerte von I
- ▶ Definition der Ableitungen I_x und I_y

$$\mathbf{I}_x := \partial \mathbf{I} / \partial x$$

$$\mathbf{I}_y := \partial \mathbf{I} / \partial y$$

- ▶ Somit ergibt sich:

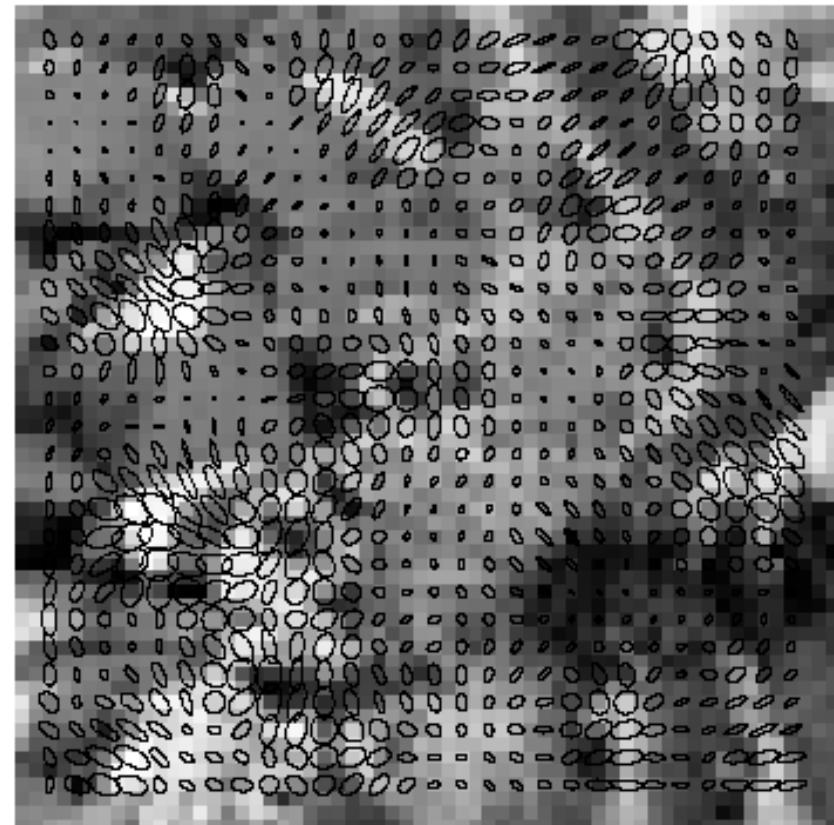
$$\mathbf{A} = \sum_u \sum_v w(u, v) \begin{pmatrix} \mathbf{I}_x^2 & \mathbf{I}_x \mathbf{I}_y \\ \mathbf{I}_x \mathbf{I}_y & \mathbf{I}_y^2 \end{pmatrix}$$

- ▶ Eine Ecke / Interessanter Punkt zeichnet sich durch eine Variation von S_I in beide Richtungen von (x, y) aus

Harris Operator (3)

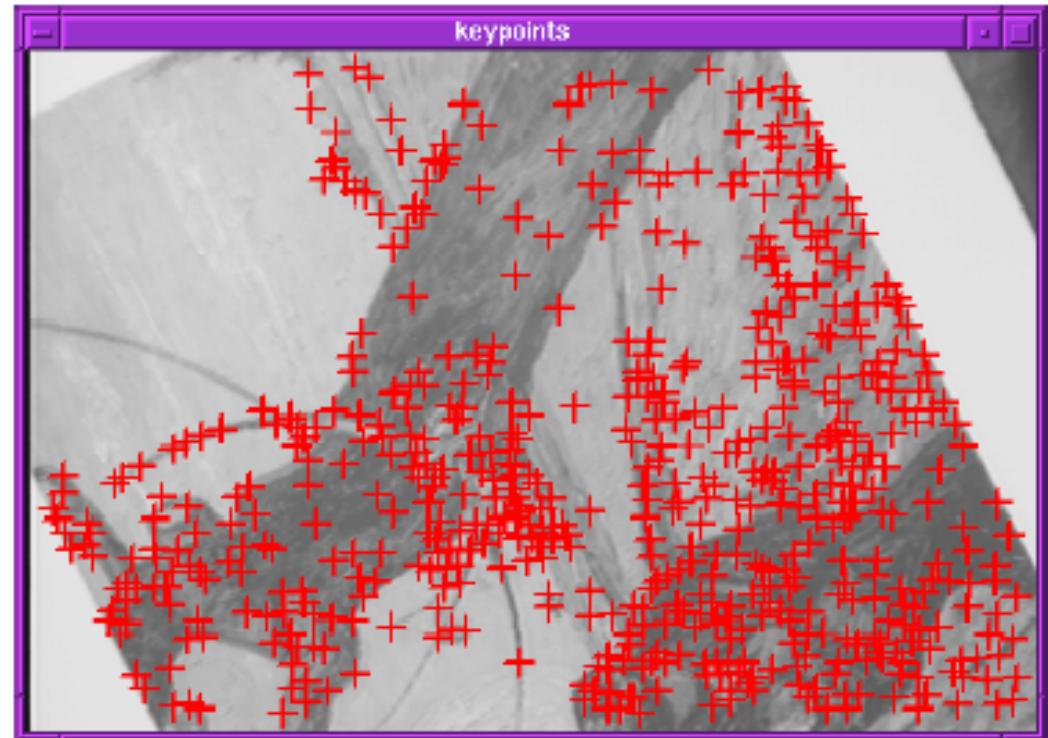
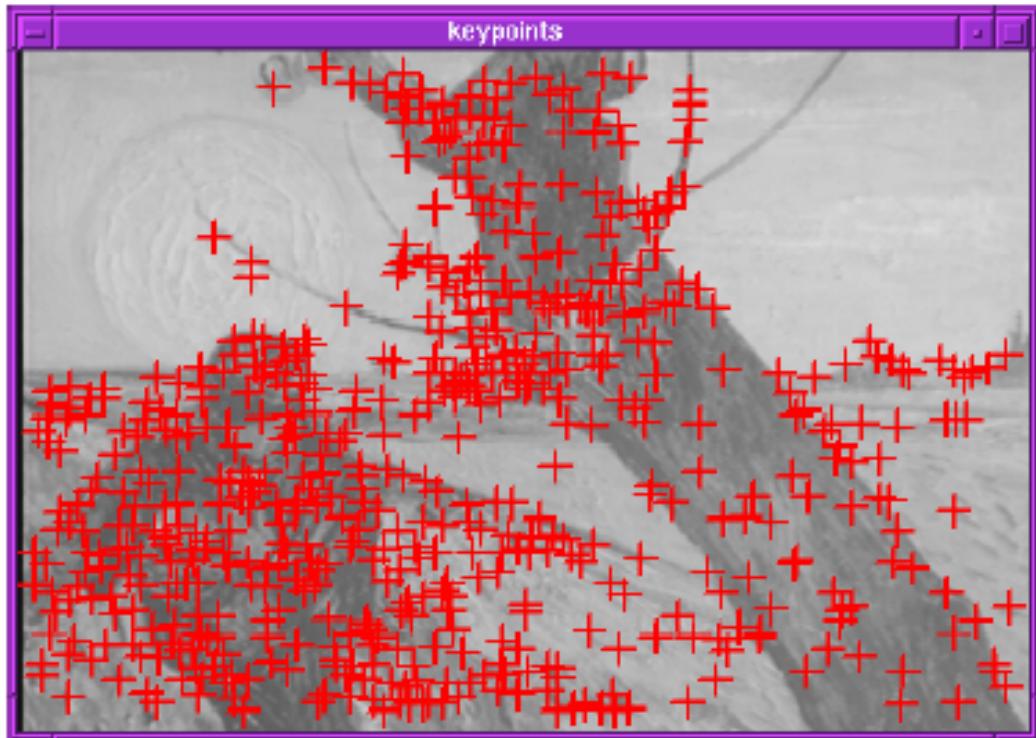
- ▶ Wie erkenne ich einen interessanten Punkt?
- ▶ Analyse der Eigenwerte λ_1 und λ_2 von A
- ▶ Es gelten folgende Eigenschaften:
 1. Wenn $\lambda_1 \approx 0$ und $\lambda_2 \approx 0$ liegt kein interessanter Punkt vor.
 2. Eine Kante liegt vor, wenn $\lambda_1 \approx 0$ und $\lambda_2 = c_1$ und $c_1 \gg 0$ ist.
 3. Eine Ecke liegt vor, wenn $\lambda_1 = c_1$, $\lambda_2 = c_2$ und $c_1 \neq c_2$, sowie $c_1 \gg 0$ und $c_2 \gg 0$.
- ▶ Das Verfahren erfüllt also formal die eingangs gestellten Anforderungen
- ▶ A ist rotationssymmetrisch!

Harris Corner Detection - Beispiel



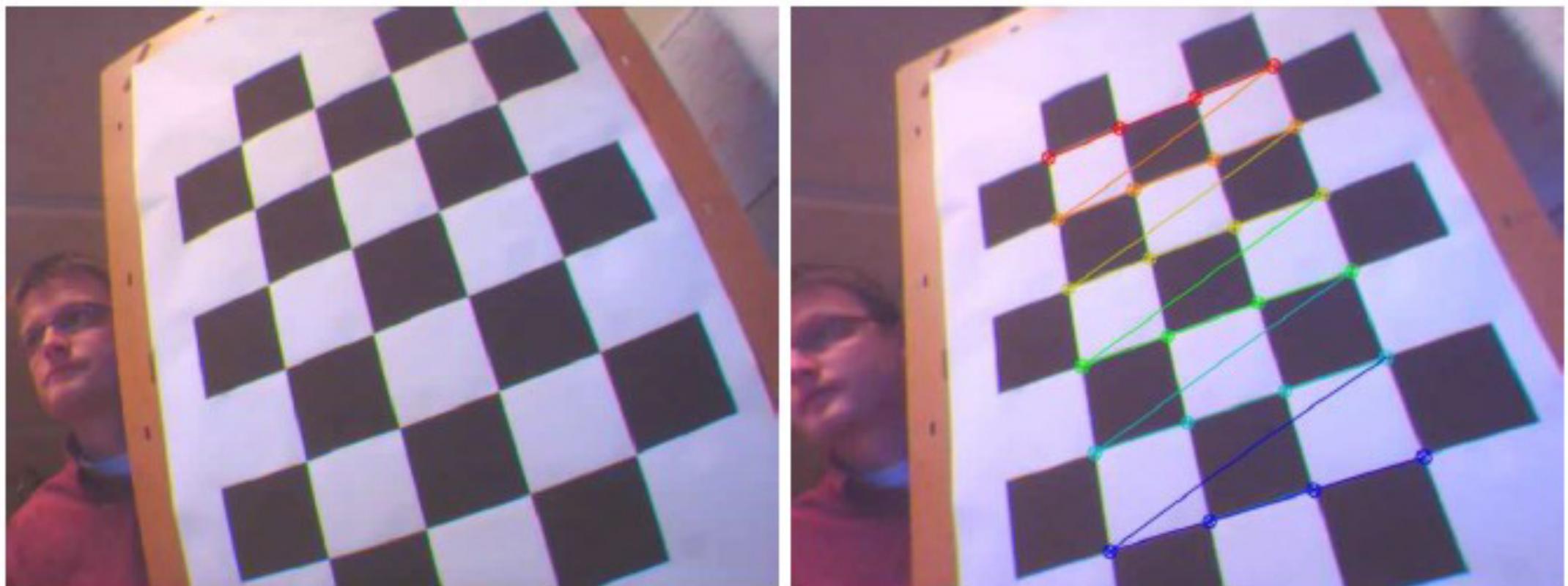
Corners are detected where the product of the ellipse axis lengths reaches a local maximum.

Harris Corner Detection - Beispiel



- ▶ Ursprünglich für Motion Tracking entwickelt
- ▶ Reduziert die Rechenzeit erheblich verglichen mit Pixel-Pixel-Tracking
- ▶ Translations- und Rotationsinvariant (nicht skalierungsinvariant)

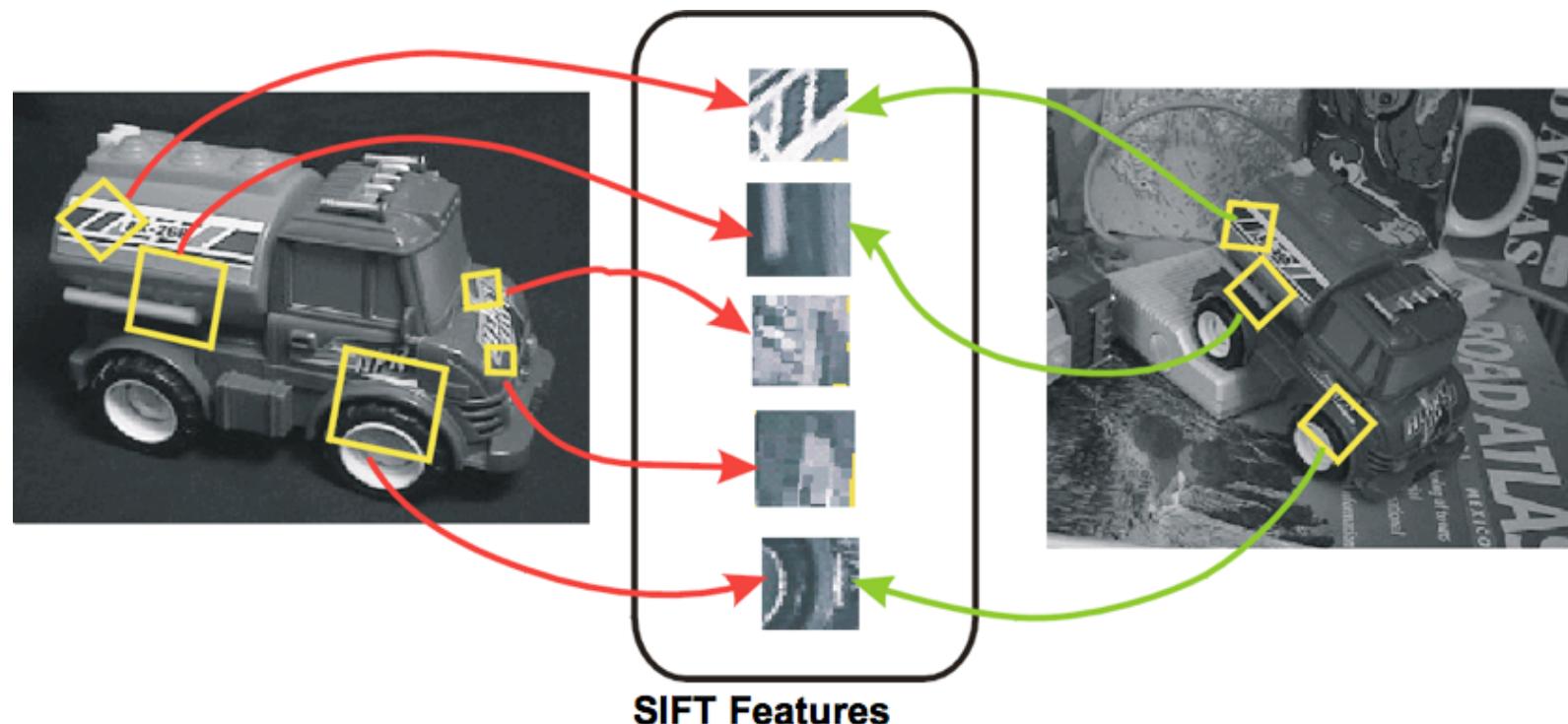
Harris Corners zu Kamerakalibrierung



SIFT-Features (1)

Distinctive image features from scale-invariant keypoints. David G. Lowe,
International Journal of Computer Vision, 60, 2 (2004), pp. 91-110.

SIFT = Scale Invariant Feature Transform



Vorteile von lokalen, invariante Features

- ▶ Lokalität: Bildmerkmale sind lokal, also robust gegen Verdeckungen und Clutter
- ▶ Unterscheidbarkeit: Individuelle Features können gegen eine große Datenbank gematcht werden
- ▶ Anzahl: Auch für kleine Objekt können viele Features erzeugt werden
- ▶ Effizienz: Nah an Echtzeit
- ▶ Erweiterbarkeit: Können durch andere Verfahren erweitert werden, um das Verfahren robuster zu machen

SIFT-Algorithmus - Zusammenfassung

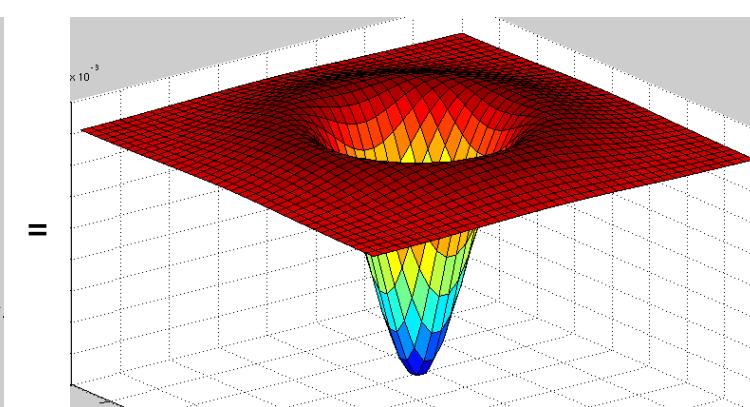
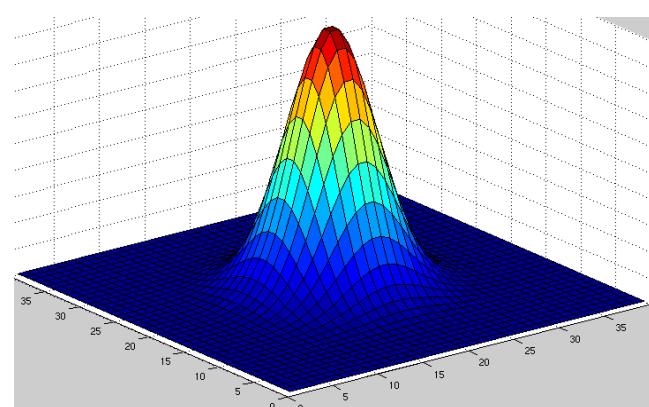
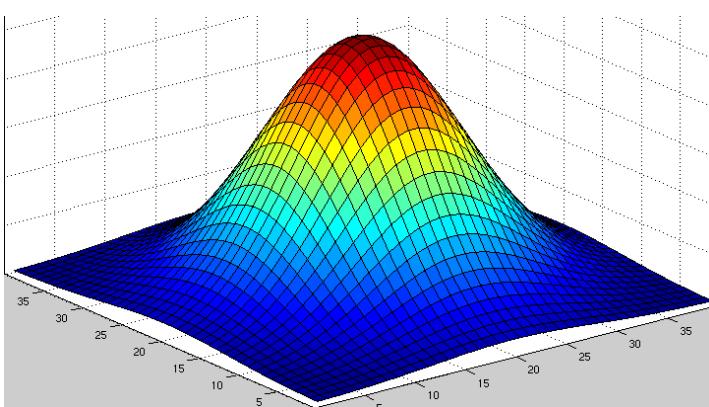
1. **Erzwinge Skalierungsinvarianz:** Berechne „Difference of Gaussians“ Maximum für viele verschiedene Skalierungen. Unterdrücke Nicht-Maxima und detektiere lokale Maxima als Keypoint-Kandidaten
2. **Lokalisiere die Ecke:** Fitte eine quadratische Funktion. Berechne den Mittelpunkt mit Subpixel-Genauigkeit.
3. **Entferne Merkmale an Kanten:** Berechne das Verhältnis der Eigenwerte. Entferne Keypoints, deren Verhältnis unterhalb eines Schwellenwertes ist
4. **Erzwinge Rotationsinvarianz:** Berechne die Orientierung der Merkmalsumgebung indem der Patch in Richtung der stärksten Divergenz gedreht wird. Drehe die Region so, dass diese Richtung nach oben zeigt.
5. **Berechne Merkmalsdeskriptor:** Berechne 16 4x4-Gradientenhistogramme in der Umgebung. Drehe das Histogramm so, dass der stärkste Gradient nach oben zeigt.
6. **Erzwinge Invarianz gegen Belichtungswechsel und Sättigung:** Normalisiere auf Einheitslänge. Begrenze die Gradienten um gegen Übersättigung geschützt zu sein

SIFT-Algorithmus - Zusammenfassung

1. **Erzwinge Skalierungsinvarianz:** Berechne „Difference of Gaussians“ Maximum für viele verschiedene Skalierungen. Unterdrücke Nicht-Maxima und detektiere lokale Maxima als Keypoint-Kandidaten
2. **Lokalisiere die Ecke:** Fitte eine quadratische Funktion. Berechne den Mittelpunkt mit Subpixel-Genauigkeit.
3. **Entferne Merkmale an Kanten:** Berechne das Verhältnis der Eigenwerte. Entferne Keypoints, deren Verhältnis unterhalb eines Schwellenwertes ist
4. **Erzwinge Rotationsinvarianz:** Berechne die Orientierung der Merkmalsumgebung indem der Patch in Richtung der stärksten Divergenz gedreht wird. Drehe die Region so, dass diese Richtung nach oben zeigt.
5. **Berechne Merkmalsdeskriptor:** Berechne 16 4x4-Gradientenhistogramme in der Umgebung. Drehe das Histogramm so, dass der stärkste Gradient nach oben zeigt.
6. **Erzwinge Invarianz gegen Belichtungswechsel und Sättigung:** Normalisiere auf Einheitslänge. Begrenze die Gradienten um gegen Übersättigung geschützt zu sein

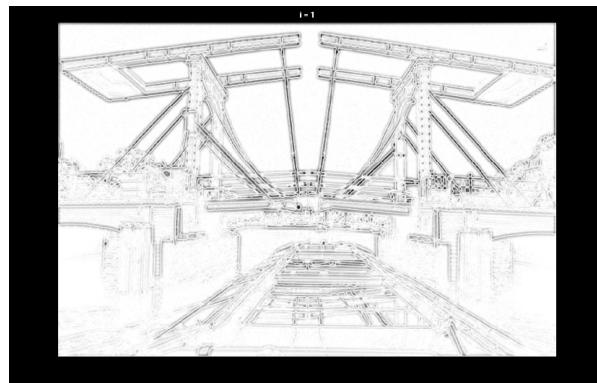
SIFT - Keypointkandidaten (1)

- ▶ Berechne „Difference of Gaussians“ für verschiedene Skalierungen
- ▶ Erinnerung:

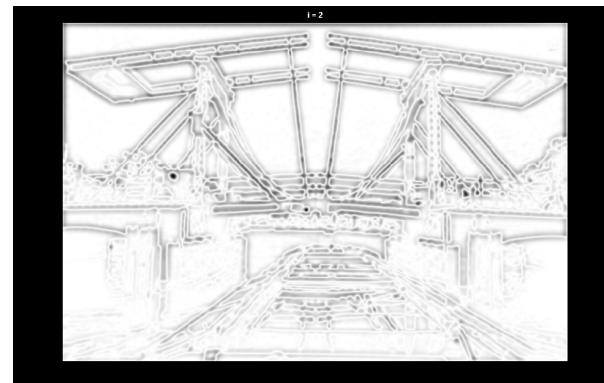


SIFT - Keypointkandidaten (2)

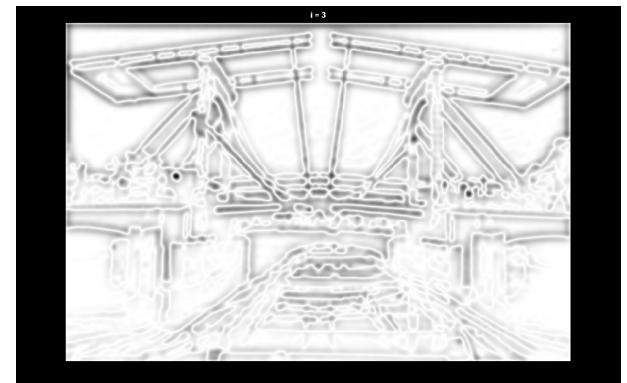
- Berechne eine Serie von DoG mit steigender Varianz:



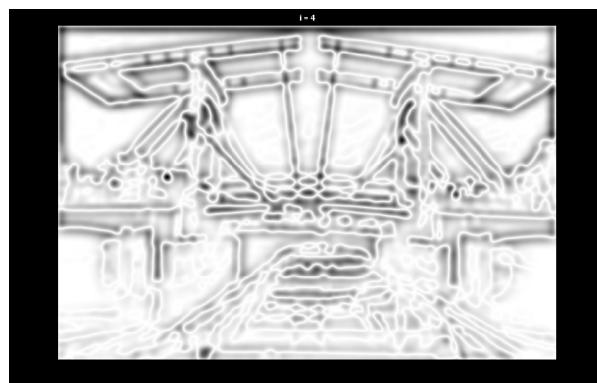
$K = 1$



$K = 2$



$K = 3$



$K = 4$



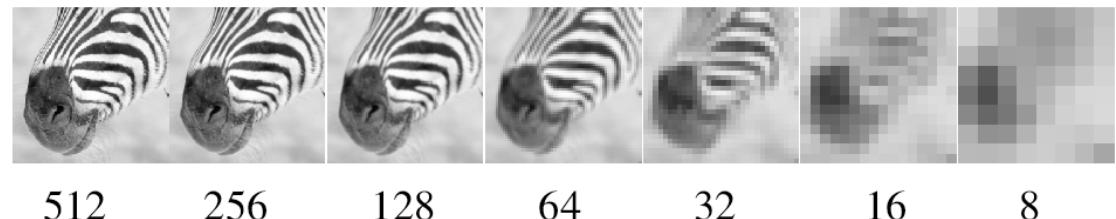
$K = 5$



$K = 10$

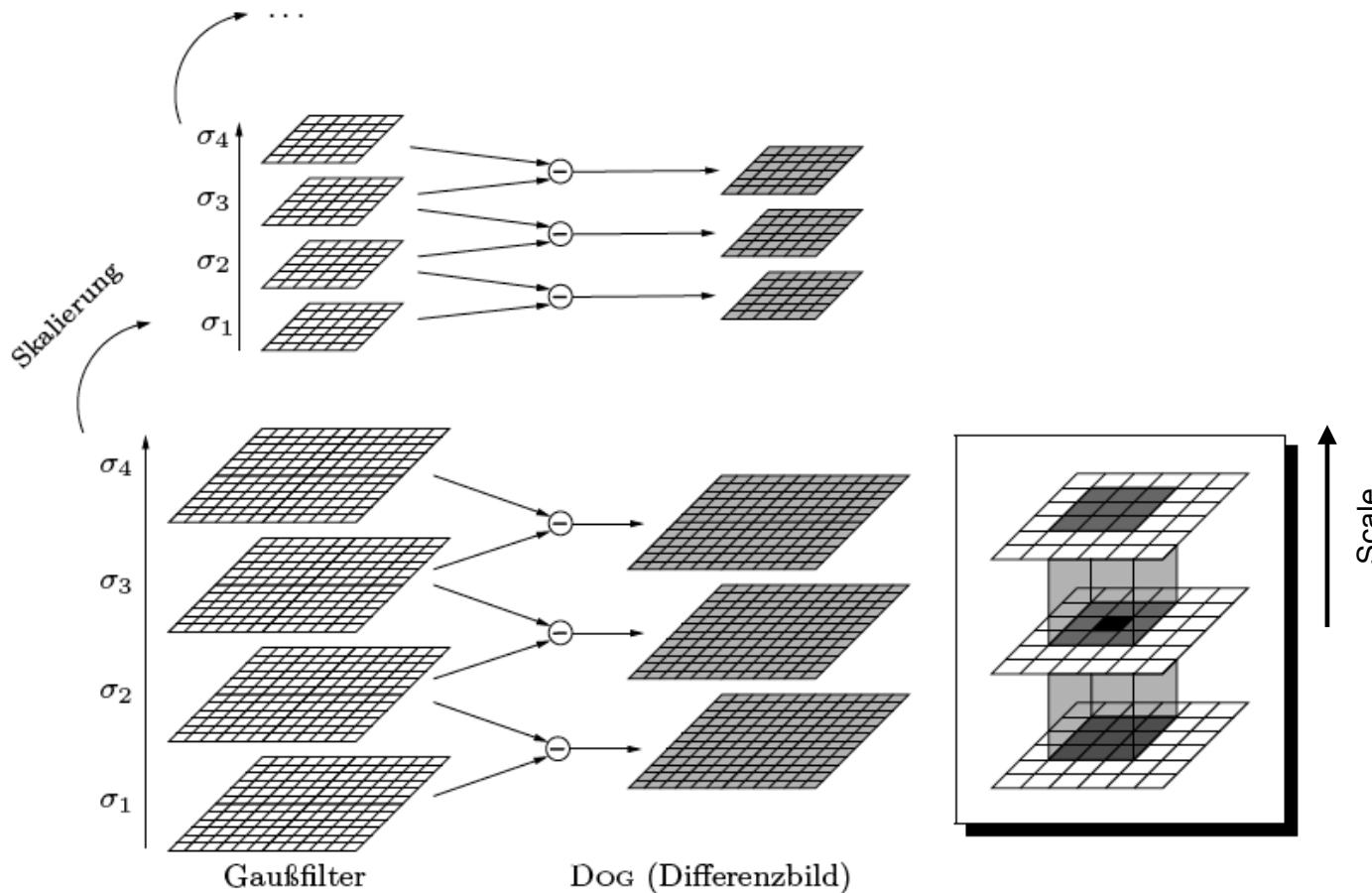
SIFT - Keypointkandidaten (3)

- Erzwinge Skalierungsinvarianz durch
“Bildpyramide”



SIFT - Keypointkandidaten (4)

- ▶ Halbiere die Größe der Bilder und wiederhole das Verfahren



Keypoint-Kandidaten sind in ihrer 26er-Nachbarschaft maximal oder minimal

SIFT - Keypointkandidaten - Beispiel

- ▶ Bildgröße 233x189
- ▶ 832 DOG Extrama
- ▶ 729 über vorgegebenen Schwellwert



SIFT-Algorithmus - Zusammenfassung

1. **Erzwinge Skalierungsinvarianz:** Berechne „Difference of Gaussians“ Maximum für viele verschiedene Skalierungen. Unterdrücke Nicht-Maxima und detektiere lokale Maxima als Keypoint-Kandidaten
2. **Lokalisiere die Ecke:** Fitte eine quadratische Funktion. Berechne den Mittelpunkt mit Subpixel-Genauigkeit.
3. **Entferne Merkmale an Kanten:** Berechne das Verhältnis der Eigenwerte. Entferne Keypoints, deren Verhältnis unterhalb eines Schwellenwertes ist
4. **Erzwinge Rotationsinvarianz:** Berechne die Orientierung der Merkmalsumgebung indem der Patch in Richtung der stärksten Divergenz gedreht wird. Drehe die Region so, dass diese Richtung nach oben zeigt.
5. **Berechne Merkmalsdeskriptor:** Berechne 16 4x4-Gradientenhistogramme in der Umgebung. Drehe das Histogramm so, dass der stärkste Gradient nach oben zeigt.
6. **Erzwinge Invarianz gegen Belichtungswechsel und Sättigung:** Normalisiere auf Einheitslänge. Begrenze die Gradienten um gegen Übersättigung geschützt zu sein

Keypointlokalisierung

- ▶ Filtere Kandidaten auf Kanten
- ▶ Verfahren analog zu Harris-Corners

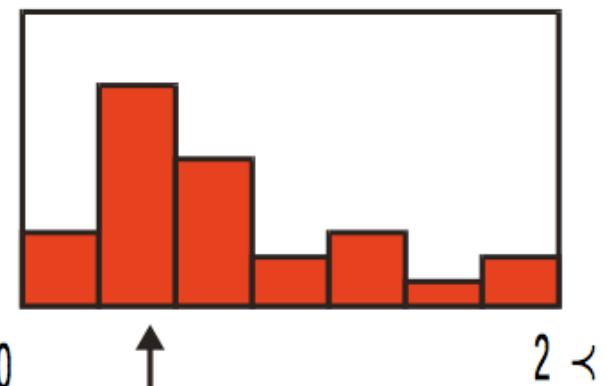
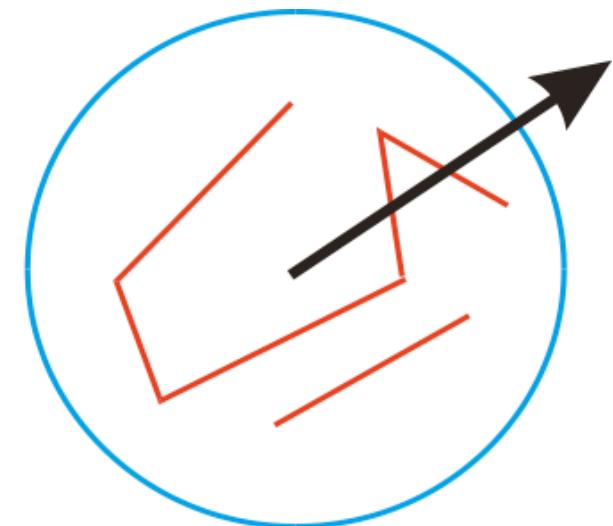


SIFT-Algorithmus - Zusammenfassung

1. **Erzwinge Skalierungsinvarianz:** Berechne „Difference of Gaussians“ Maximum für viele verschiedene Skalierungen. Unterdrücke Nicht-Maxima und detektiere lokale Maxima als Keypoint-Kandidaten
2. **Lokalisiere die Ecke:** Fitte eine quadratische Funktion. Berechne den Mittelpunkt mit Subpixel-Genauigkeit.
3. **Entferne Merkmale an Kanten:** Berechne das Verhältnis der Eigenwerte. Entferne Keypoints, deren Verhältnis unterhalb eines Schwellenwertes ist
4. **Erzwinge Rotationsinvarianz:** Berechne die Orientierung der Merkmalsumgebung indem der Patch in Richtung der stärksten Divergenz gedreht wird. Drehe die Region so, dass diese Richtung nach oben zeigt.
5. **Berechne Merkmalsdeskriptor:** Berechne 16 4x4-Gradientenhistogramme in der Umgebung. Drehe das Histogramm so, dass der stärkste Gradient nach oben zeigt.
6. **Erzwinge Invarianz gegen Belichtungswechsel und Sättigung:** Normalisiere auf Einheitslänge. Begrenze die Gradienten um gegen Übersättigung geschützt zu sein

Erzwingen von Rotationsinvarianz

- ▶ Erstelle ein Histogramm lokaler Gradienten auf einer bestimmten Skalierungsebene
- ▶ Detektiere Peak
- ▶ Maxima sowie weitere Peaks innerhalb von 80% der Magnitude werden als weitere Keypoints hinzugefügt
- ▶ Jeder Schlüssel spezifiziert stabile 2D Koordinaten (x, y, Skalierung, Ausrichtung)

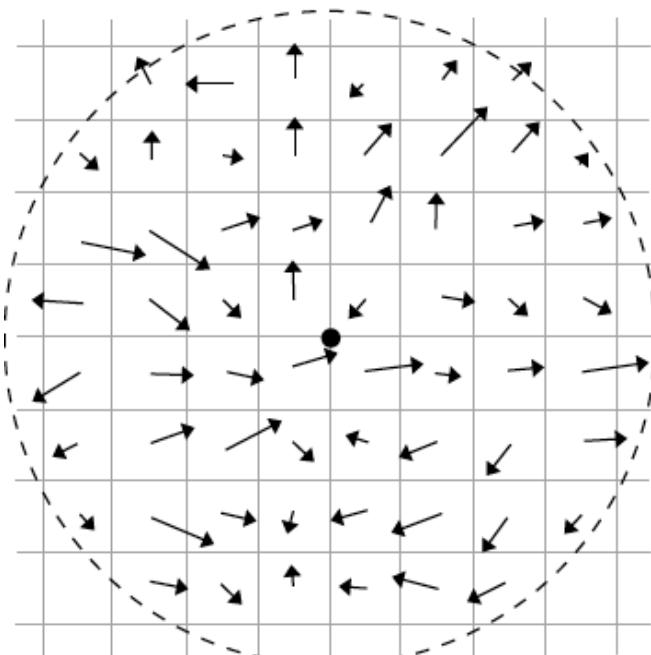


SIFT-Algorithmus - Zusammenfassung

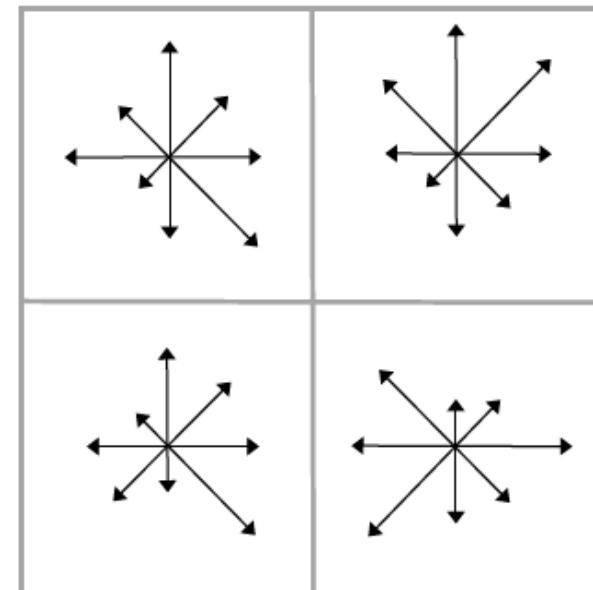
1. **Erzwinge Skalierungsinvarianz:** Berechne „Difference of Gaussians“ Maximum für viele verschiedene Skalierungen. Unterdrücke Nicht-Maxima und detektiere lokale Maxima als Keypoint-Kandidaten
2. **Lokalisiere die Ecke:** Fitte eine quadratische Funktion. Berechne den Mittelpunkt mit Subpixel-Genauigkeit.
3. **Entferne Merkmale an Kanten:** Berechne das Verhältnis der Eigenwerte. Entferne Keypoints, deren Verhältnis unterhalb eines Schwellenwertes ist
4. **Erzwinge Rotationsinvarianz:** Berechne die Orientierung der Merkmalsumgebung indem der Patch in Richtung der stärksten Divergenz gedreht wird. Drehe die Region so, dass diese Richtung nach oben zeigt.
5. **Berechne Merkmalsdeskriptor:** Berechne 16 4x4-Gradientenhistogramme in der Umgebung. Drehe das Histogramm so, dass der stärkste Gradient nach oben zeigt.
6. **Erzwinge Invarianz gegen Belichtungswechsel und Sättigung:** Normalisiere auf Einheitslänge. Begrenze die Gradienten um gegen Übersättigung geschützt zu sein

SIFT-Dekcriptor

- ▶ Begrenze die Gradienteneinträge (Sättigung)
- ▶ Erstelle ein Array von Gradientenhistogrammen
- ▶ Beispiel: 8 Richtungen, 2x2 Histogrammarray \Rightarrow 32 Einträge



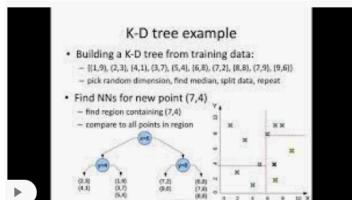
Bildgradienten



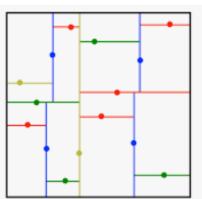
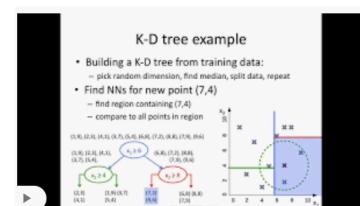
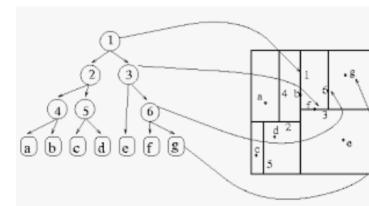
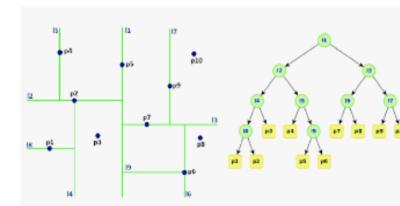
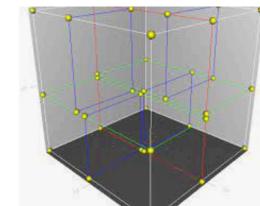
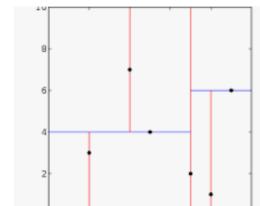
Deskriptor

Matchen von Deskriptoren

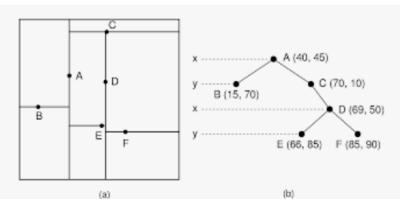
- ▶ ... via „Approximate Nearest Neighbor“-Suche
- ▶ ... mittels modifizierten kd-Baum
- ▶ ... Speedup ~ 1000
- ▶ ... mehr zur kd-Baum-Suche in Teil 7 der Vorlesung



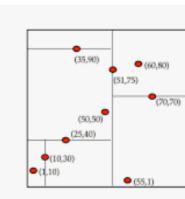
KD tree algorithm: how it works - YouTube
youtube.com



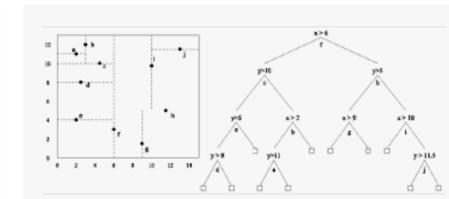
Algorithm Repository
algorist.com



20.3. KD Trees – OpenDSA Data Structures and Algorithms M...
openda-server.cs.vt.edu



K-d tree: nearest neighbor search algorithm with tractable pseudo ...
stackoverflow.com



The scheme of the KD-Tree Search algorithm. | Download Scientific...
researchgate.net

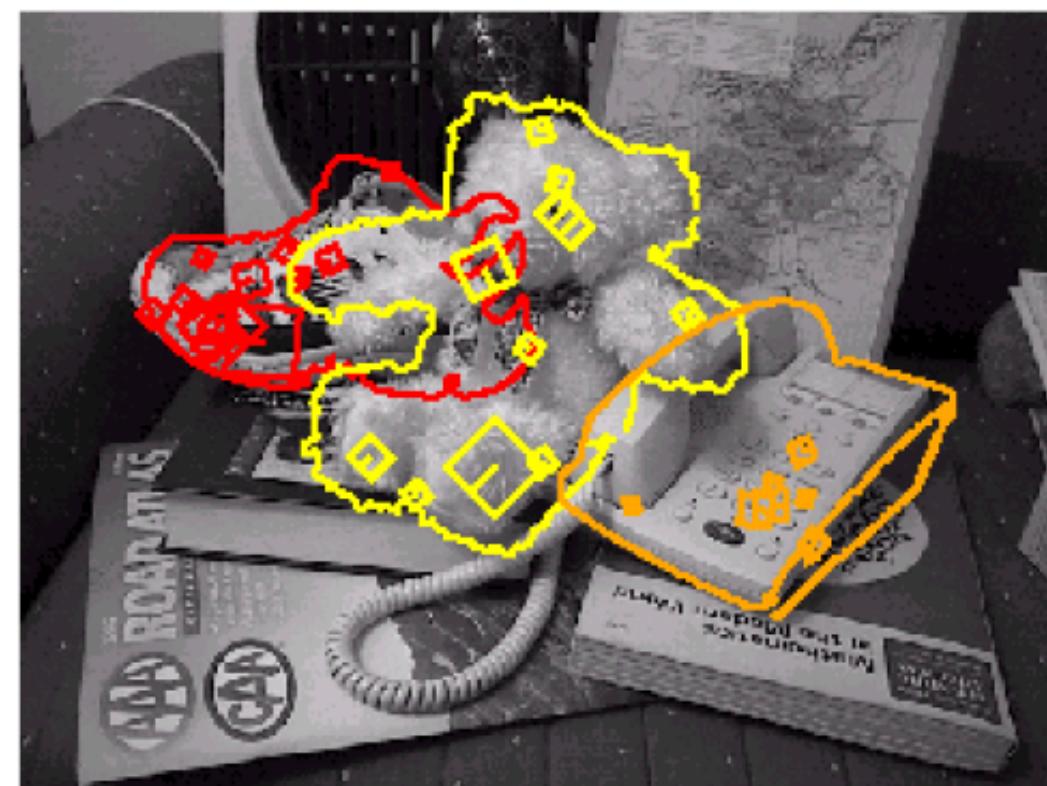
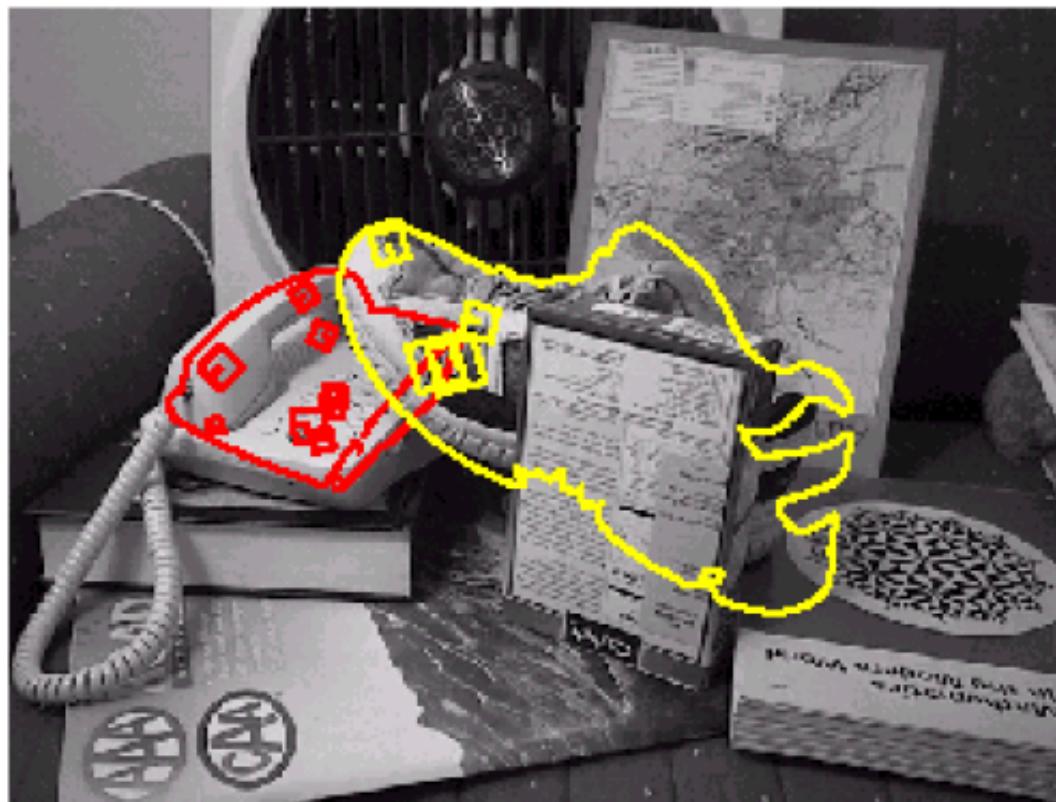


k-d tree - Wikipedia
en.wikipedia.org

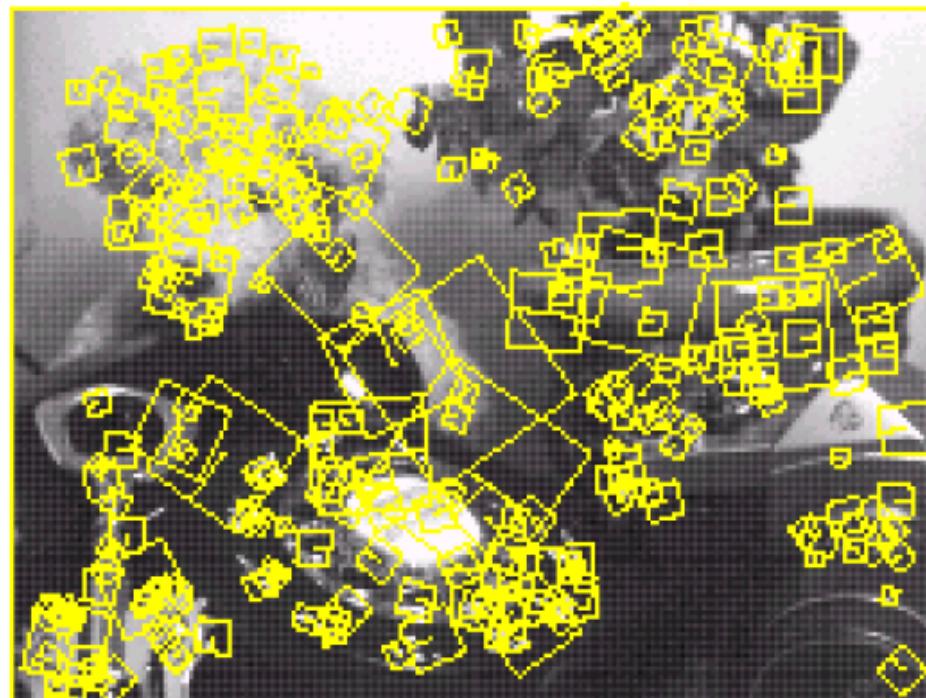
SIFT - Anwendungsbeispiel (1)



SIFT - Anwendungsbeispiel (2)

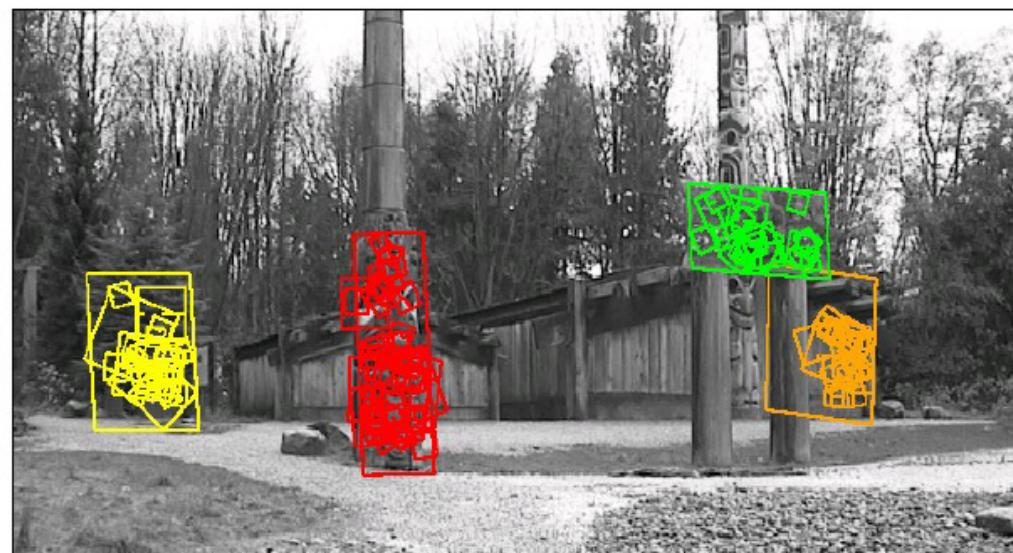
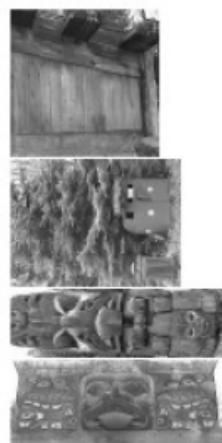


SIFT - Anwendungsbeispiel (3)

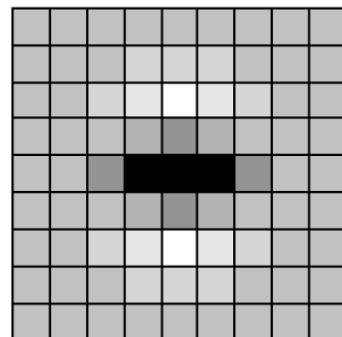


273 keys verified in final match

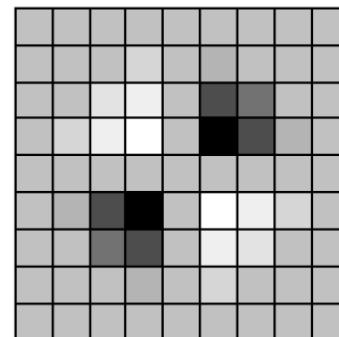
SIFT - Anwendungsbeispiel (4)



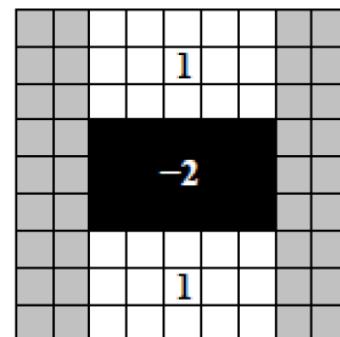
- ▶ “Speeded Up Robust Feature”
- ▶ DoB (Difference of Boxes) anstelle von DoG mit Non-Maximum-Supression



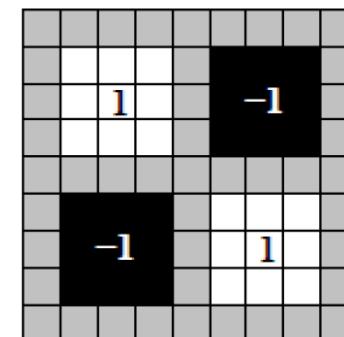
(a) L_{uu}



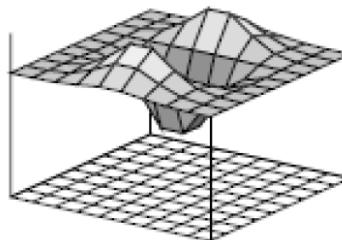
(b) L_{uv}



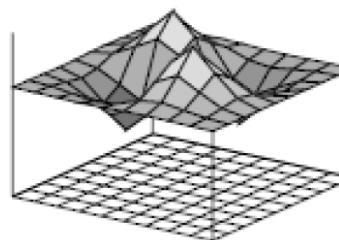
(c) D_{uu}



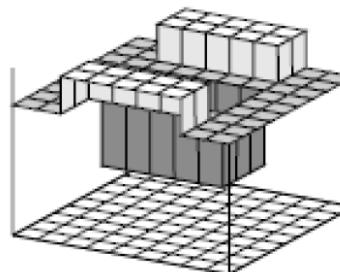
(d) D_{uv}



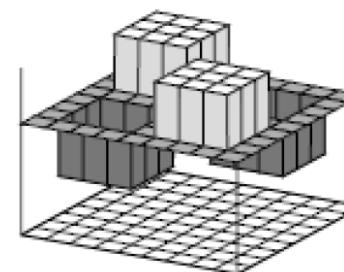
(e) L_{uu}



(f) L_{uv}



(g) D_{uu}



(h) D_{uv}

Der “Feature Zoo”

► Ein paar Beispiele:

- Harris Corner Detection
- Shi-Tomasi Corner Detector, Good Features to Track
- Scale Invariant Feature Transform (SIFT)
- Speeded-up robust Features (SURF)
- Features from Accelerated Segment Test (FAST)
- KAZE und AKAZE
- Binary Robust Independent Elementary Features (BRIEF)
- Binary Robust Invariant Scalable Points (BRISK)
- Oriented FAST and Rotated Brief (ORB)
- Center Surround Extremas for Realtime Feature Detection and Matching (CenSURE)

Aber welches Feature und welcher Deskriptor eignen sich wofür?

Das muss man ja nach vorliegenden Anwendungszweck individuell betrachten. SIFT / SURF eher für klassische Bildverarbeitung (Stitching). BRIEF / ORB eher für Tracking und Structure from Motion. Literaturstudium und Evaluation notwendig!

Evaluationsbeispiel

► Kontext “Image Registration”

TABLE I. OPENCV SETTINGS AND DESCRIPTOR SIZES OF THE FEATURE-DETECTOR-DESCRIPTORS USED

Algorithm	OpenCV Object	Descriptor Size
SIFT	cv.SIFT('ContrastThreshold',0.04,'Sigma',1.6)	128 Bytes
SURF(128D)	cv.SURF('Extended',true,'HessianThreshold',100)	128 Floats
SURF(64D)	cv.SURF('HessianThreshold',100)	64 Floats
KAZE	cv.KAZE('NOctaveLayers',3,'Extended',true)	128 Floats
AKAZE	cv.AKAZE('NOctaveLayers',3)	61 Bytes
ORB	cv.ORB('MaxFeatures',100000)	32 Bytes
ORB(1000)	cv.ORB('MaxFeatures',1000)	32 Bytes
BRISK	cv.BRISK()	64 Bytes
BRISK(1000)	cv.BRISK(); cv.KeyPointsFilter,retainBest(features,1000)	64 Bytes

Tareen, Shaharyar Ahmed Khan, and Zahra Saleem. "A comparative analysis of sift, surf, kaze, akaze, orb, and brisk." *2018 International conference on computing, mathematics and engineering technologies (iCoMET)*. IEEE, 2018.

Evaluationsbeispiel

► Kontext “Image Registration”



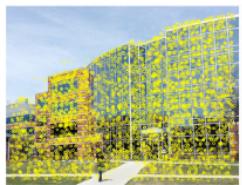
(a) SIFT Features (1907, 2209)



(b) Image Matching with SIFT



(c) Image Registration with SIFT



(d) SURF Features (3988, 4570)



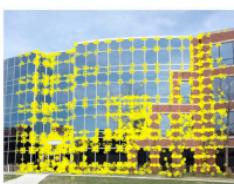
(e) Image Matching with SURF



(f) Image Registration with SURF



(m) ORB Features (5095, 5345)



(n) Image Matching with ORB



(o) Image Registration with ORB



(p) BRISK Features (3288, 3575)



(q) Image Matching with BRISK



(r) Image Registration with BRISK

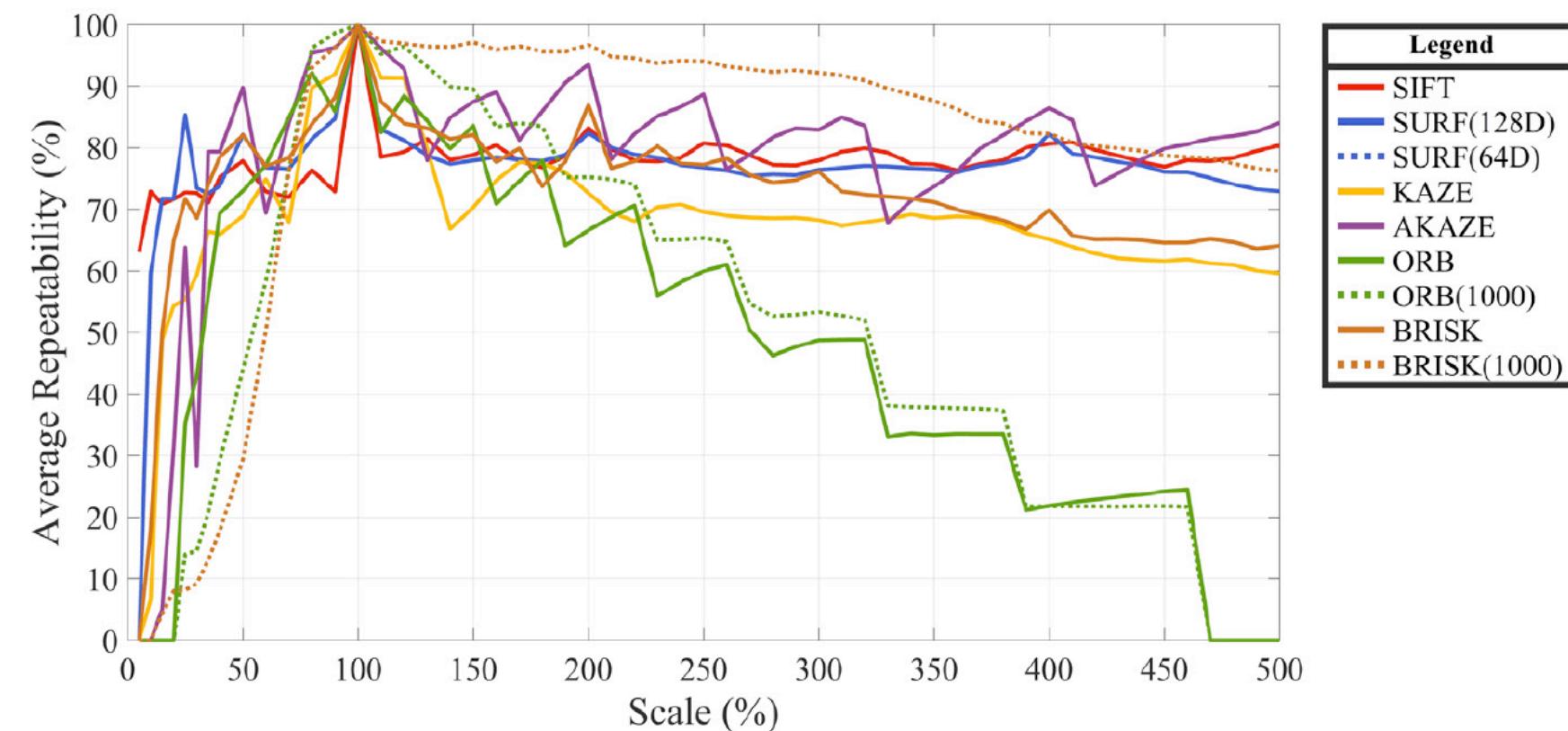
Tareen, Shaharyar Ahmed Khan, and Zahra Saleem. "A comparative analysis of sift, surf, kaze, akaze, orb, and brisk." *2018 International conference on computing, mathematics and engineering technologies (iCoMET)*. IEEE, 2018.

Evaluationsbeispiel

Algorithm	Features Detected in the Image Pairs		Features Matched	Outliers Rejected	Feature Detection & Description Time (s)		Feature Matching Time (s)	Outlier Rejection & Homography Calculation Time (s)	Total Image Matching Time (s)
	1 st Image	2 nd Image			1 st Image	2 nd Image			
Building Dataset (Image Pair # 1)									
SIFT	1907	2209	384	51	0.1812	0.1980	0.1337	0.0057	0.5186
SURF(128D)	3988	4570	319	58	0.1657	0.1786	0.5439	0.0058	0.8940
SURF(64D)	3988	4570	612	73	0.1625	0.1734	0.2956	0.0052	0.6367
KAZE	1291	1359	465	26	0.2145	0.2113	0.0613	0.0053	0.4924
AKAZE	1458	1554	475	36	0.0695	0.0715	0.0307	0.0055	0.1772
ORB	5095	5345	854	149	0.0213	0.0220	0.1586	0.0067	0.2086
ORB(1000)	1000	1000	237	21	0.0103	0.0101	0.0138	0.0049	0.0391
BRISK	3288	3575	481	47	0.0533	0.0565	0.1236	0.0056	0.2390
BRISK(1000)	1000	1000	190	33	0.0188	0.0191	0.0158	0.0049	0.0586

Tareen, Shaharyar Ahmed Khan, and Zahra Saleem. "A comparative analysis of sift, surf, kaze, akaze, orb, and brisk." *2018 International conference on computing, mathematics and engineering technologies (iCoMET)*. IEEE, 2018.

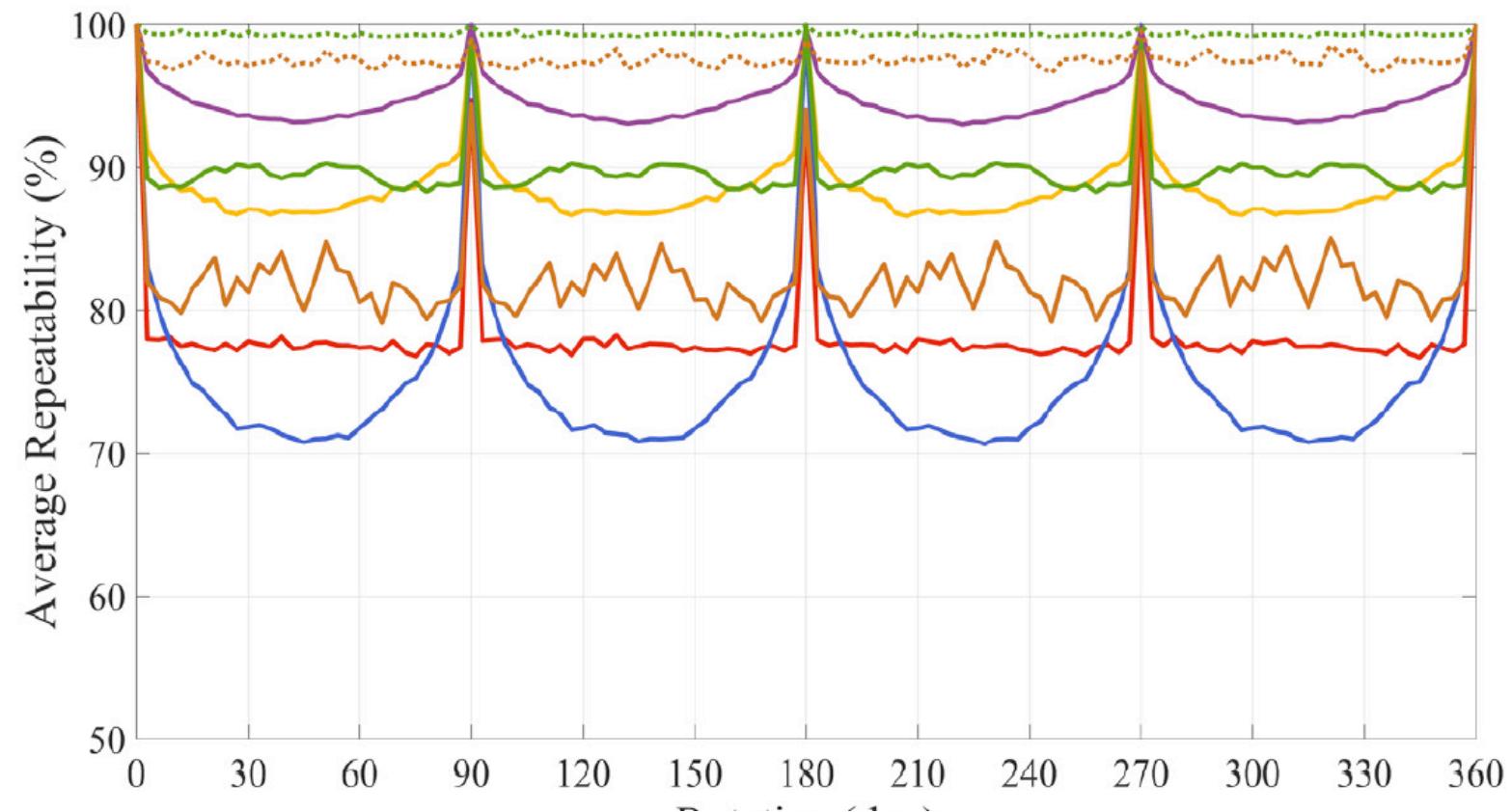
Evaluationsbeispiel



Tareen, Shaharyar Ahmed Khan, and Zahra Saleem. "A comparative analysis of sift, surf, kaze, akaze, orb, and brisk." *2018 International conference on computing, mathematics and engineering technologies (iCoMET)*. IEEE, 2018.

Evaluationsbeispiel

Legend	
SIFT	—
SURF(128D)	—
SURF(64D)	···
KAZE	—
AKAZE	—
ORB	—
ORB(1000)	···
BRISK	—
BRISK(1000)	···



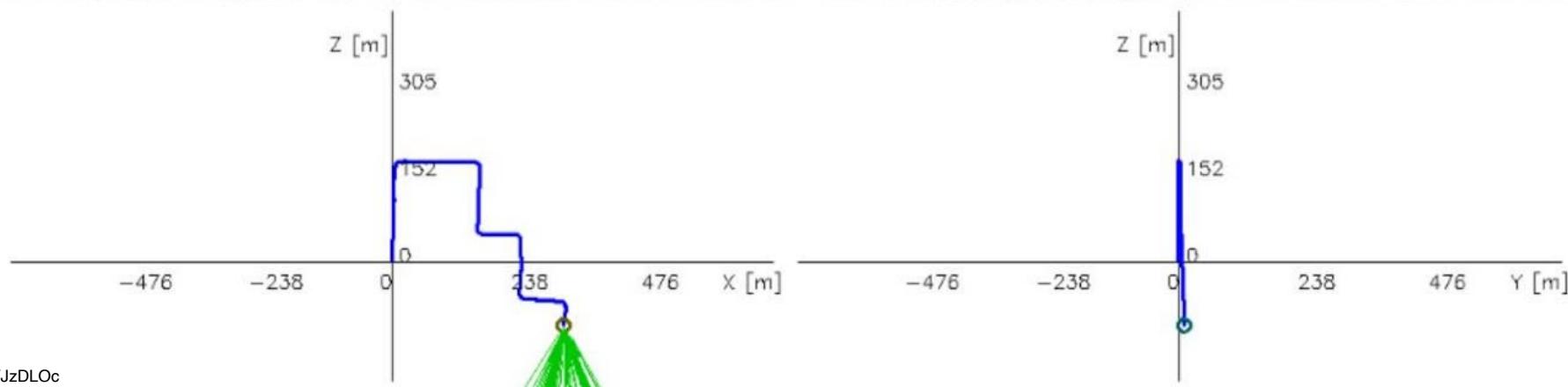
Tareen, Shaharyar Ahmed Khan, and Zahra Saleem. "A comparative analysis of sift, surf, kaze, akaze, orb, and brisk." *2018 International conference on computing, mathematics and engineering technologies (iCoMET)*. IEEE, 2018.

Eigenschaften

- ▶ Anzahl detekтирter Features
 - ORB > BRISK > SURF > SIFT > AKAZE > KAZE
- ▶ “Computational Efficiency”
 - ORB > ORB₁₀₀₀ > BRISK > BRISK₁₀₀₀ > SURF_{64D} > SURF_{128D} > AKAZE > SIFT > KAZE
- ▶ “Feature Matching”
 - ORB₁₀₀₀ > BRISK₁₀₀₀ > AKAZE > KAZE < SURF_{64D} < ORB < BRISK < SIFT < SURF_{128D}
- ▶ “Total Image Matching” (Speed)
 - ORB₁₀₀₀ > BRISK₁₀₀₀ > AKAZE > KAZE > SURF_{64D} > SIFT > ORB > BRISK > SURF_{128D}

Tareen, Shaharyar Ahmed Khan, and Zahra Saleem. "A comparative analysis of sift, surf, kaze, akaze, orb, and brisk." *2018 International conference on computing, mathematics and engineering technologies (iCoMET)*. IEEE, 2018.

Beispiel Monocular SLAM



<https://youtu.be/XpTHYJzDLoc>

ORB SLAM 2



<https://youtu.be/ufvPS5wJAx0>

Zusammenfassung

- ▶ Verarbeitung von Entfernungsdaten
 - Filterung mit Mittelwert- und Medianfilter
 - Heuristische Linienerkennung
 - Hough-Transformation
- ▶ Bildverarbeitung
 - Ziel: Stabile, gut lokalisierte Features und eindeutige Deskriptoren
 - Kantenerkennung mit Filtern 1. oder 2. Ordnung
 - Dünne Kanten mit Canny-Edge
 - Ecken mit Harris-Corners
 - SIFT-Features und Deskriptoren

Sensordatenvorverarbeitung ist ein wichtiger Schritt in der Verarbeitungspipeline für "höherwertige" Verfahren wie Kartierung oder Lokalisierung.

Inhalt



Gliederung

1. Einleitung
2. Robot Operating System
3. Sensorik
4. **Sensordatenverarbeitung**
5. Fortbewegung
6. Lokalisierung
7. Mapping
8. Navigation
9. Ausblick

Inhalt



Gliederung

1. Einleitung
2. Robot Operating System
3. Sensorik
4. Sensordatenverarbeitung
5. Fortbewegung
6. Lokalisierung
7. Mapping
8. Navigation
9. Ausblick



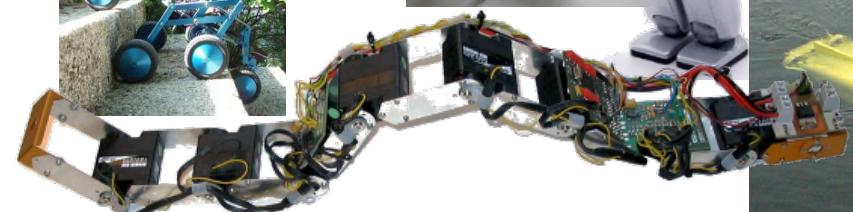
Gliederung

5. Fortbewegung

1. Einleitung
2. Bewegungsschätzung
3. Bayes- und Kalmanfilter
4. Fusion von Odometriedaten
5. Fortbewegung
6. Lokalisierung
7. Mapping
8. Navigation
9. Ausblick

Einleitung

- ▶ In dieser Vorlesung sind mobile Roboter Radfahrzeuge mit starrem Körper
- ▶ Es gibt viele andere Bewegungsformen, sie sind sinnvoll (oft), aber sie kommen hier nicht vor



(Aktiver) Freiheitsgrad

Zahl/Art der Bewegungen, die eine einzelne Komponente (Rad, Gelenk, ...) ausführen kann (translatorische, rotat. DOF). Summe der DOF der Komponenten eines Systems (Roboter).

Effektiver Freiheitsgrad

Dimensionalität der Pose.

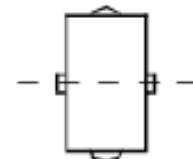
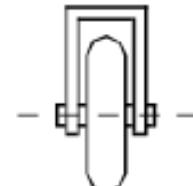
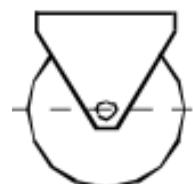
Holonomie (informell):

Jeder Wert jeder Dimension des eff. DOF ist unabhängig von den anderen einstellbar durch Änderungen in den aktiven DOF. (System kommt „direkt“ von jeder möglichen Pose in jede mögliche andere.) Ist die Anzahl der effektiven DOF größer als die der DOF, so ist das System nicht-holonom.

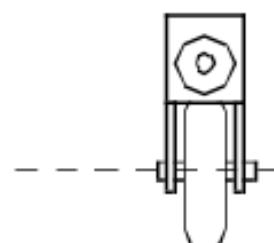
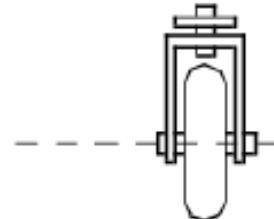
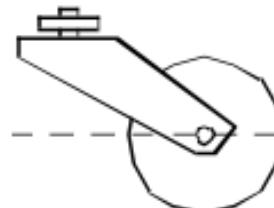
► Beispiele

- Starres Rad / Kette: 1 akt. DOF
- Differentialantrieb (Ceres, KURT2): 2 aktive DOF
- Ceres, KURT2 in d. Ebene: 3 effektive DOF (3D-Pose (x, y, θ_y))

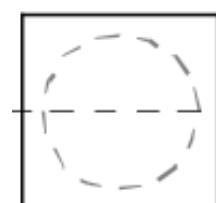
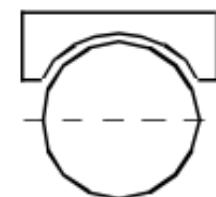
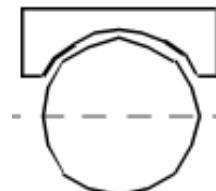
Räder



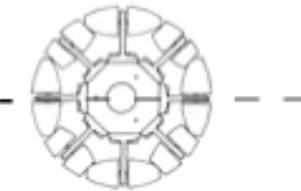
Standard-Rad
aktiv, 1-2DOF



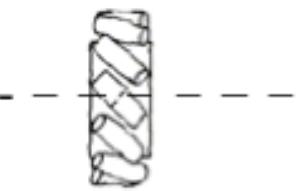
Laufrad (*castor wh.*)
passiv, 2 DOF



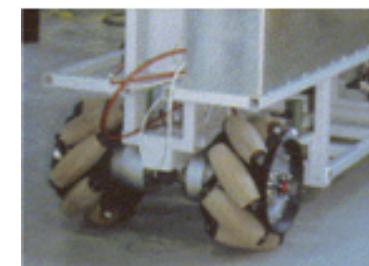
Sphärisches Rad
meist passiv, 3 DOF



swedish 90°

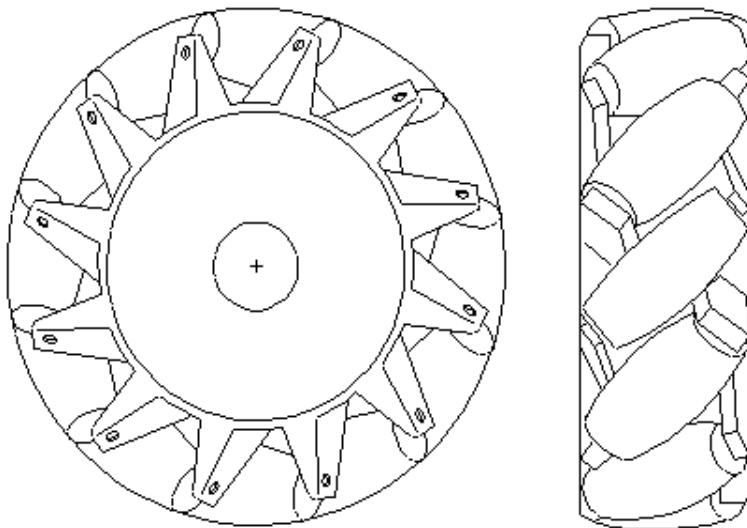


swedish 45°



**Mecanum-Rad
(*Swedish*)**
aktiv, 3 DOF

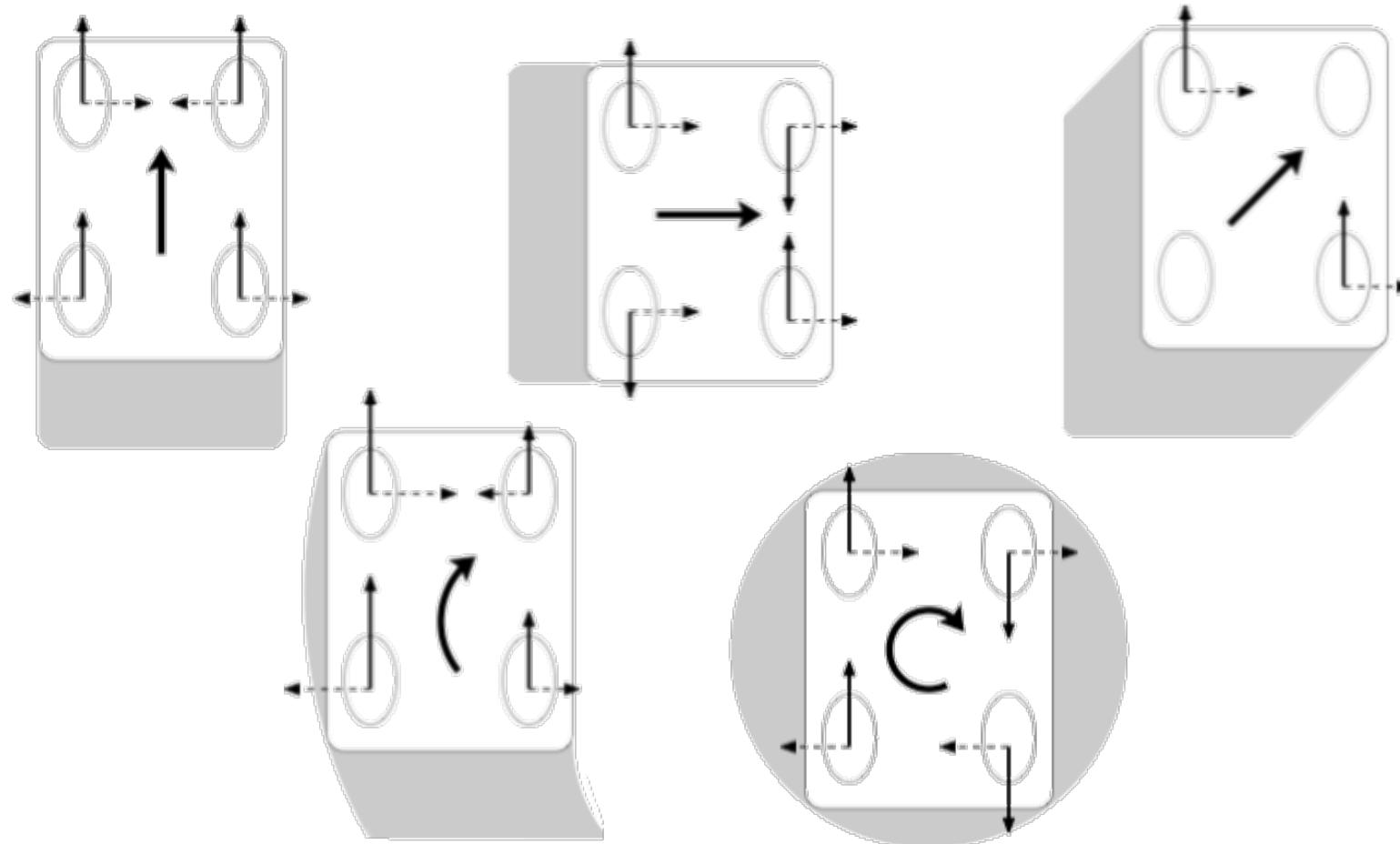
Meccanum-Räder



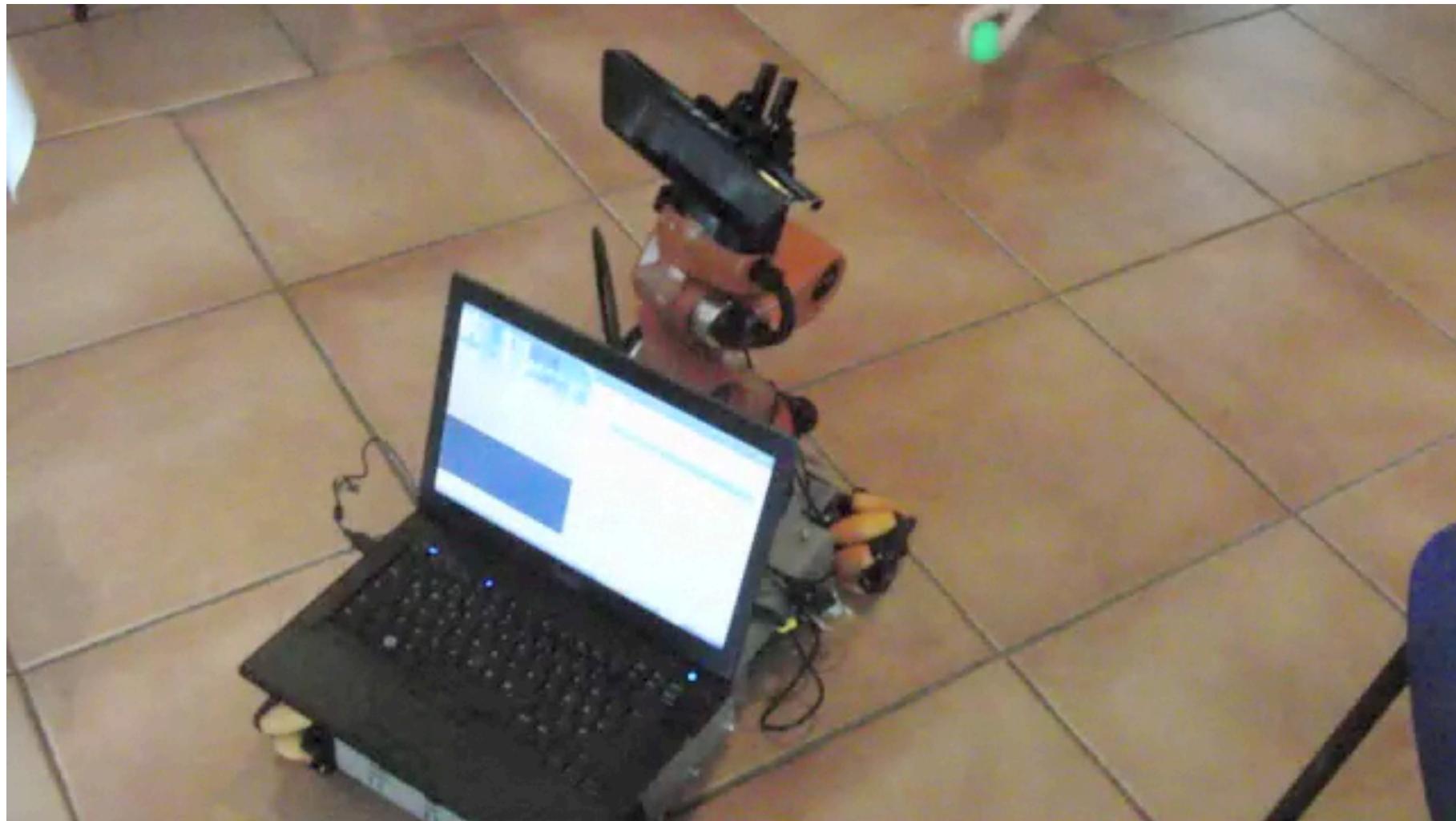
- ▶ Gesamträder und Rollen angetrieben
 - \rightarrow Holonom in der Ebene
 - Klappt nur auf planem Boden!



Bewegungssynchronisation

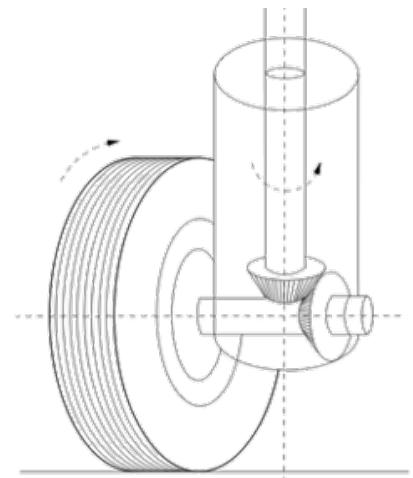
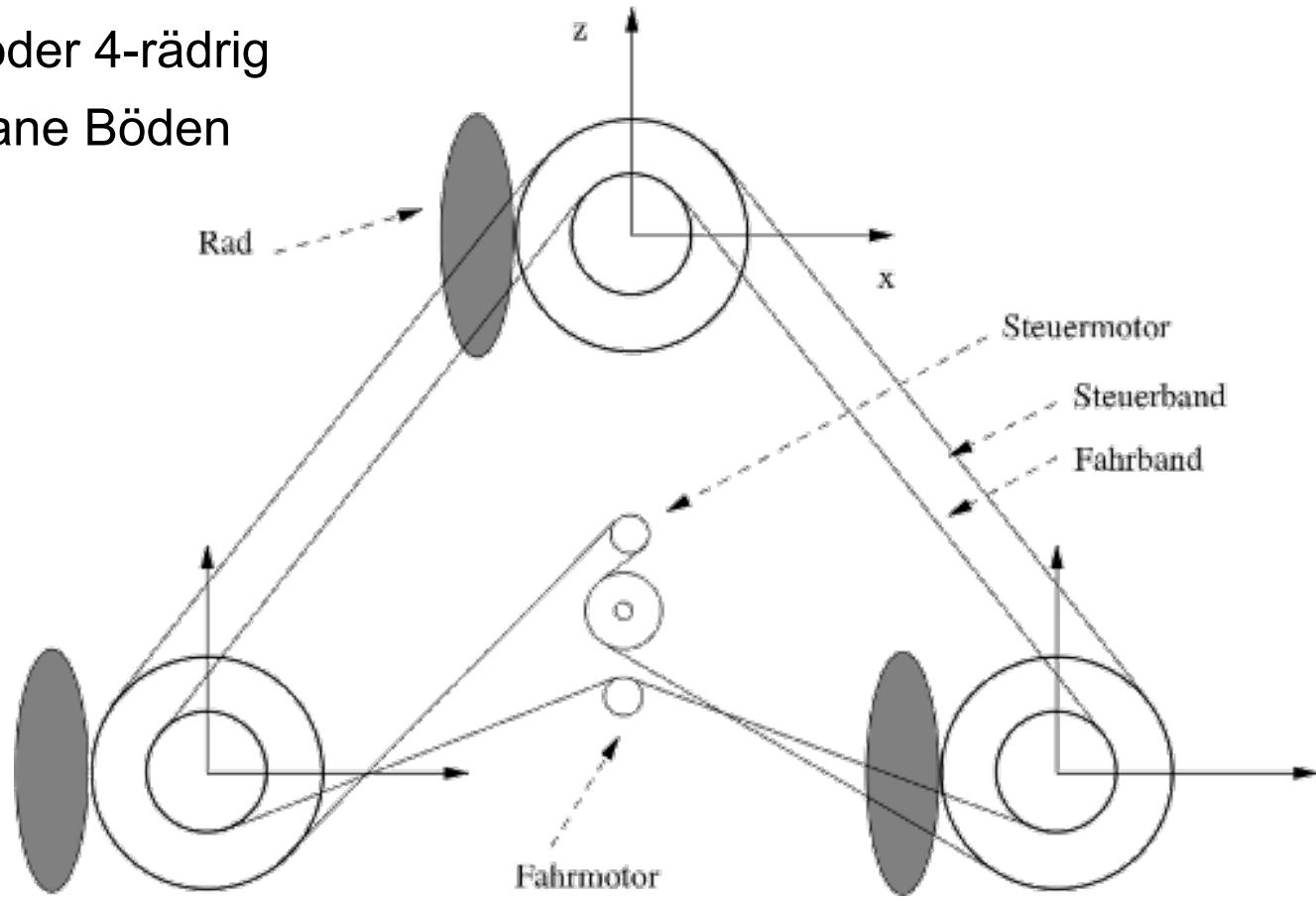


Beispiel: KUKA Youbot



Synchro-Antrieb

- ▶ Holonom
- ▶ Gibt's 3- oder 4-rädrig
- ▶ Nur für plane Böden



Beispiel

