Computer Vision

Feature Engineering & Matching

Alexandra Posekany

December 2025

Wozu Features?

Szenario: Wir haben ein Foto von einem Buchcover und suchen dieses Buch in einem Video-Stream.

Problem: Das Buch im Video ist gedreht, kleiner (weiter weg), teilweise verdeckt oder anders beleuchtet. Pixel-Vergleich scheitert

Lösung: Wir brauchen stabile “Ankerpunkte” (Features), die immun gegen diese Änderungen sind.

Was sind Features?

Nicht alle Bildbereiche sind gleich informativ.

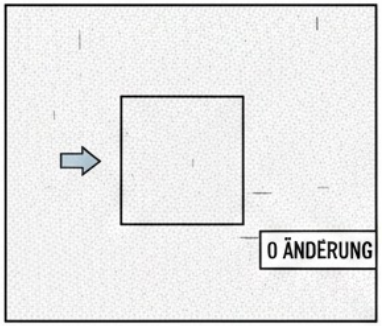
- **Flache** Regionen (Flat): Keine Information. Schwer zu lokalisieren (Wand, Himmel).
- **Kanten** (Edges): Eindimensionale Information. Wir wissen "wo" entlang der Kante wir sind, aber nicht genau "wo" auf der Linie (Aperture Problem).
- **Ecken** (Corners/Keypoints): Maximale Information. Änderungen in alle Richtungen. Einzigartig.

Was sind Features?

WAS IST EIN FEATURE?

BILDINFORMATION

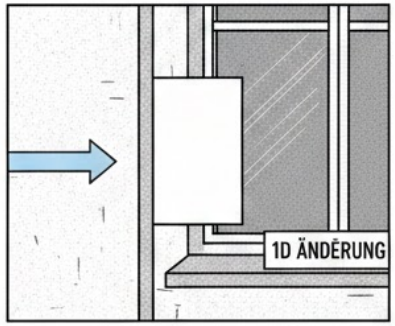
FLACH / FLAT
KEINE INFO



FLAT

SCHWER ZU FINDEN

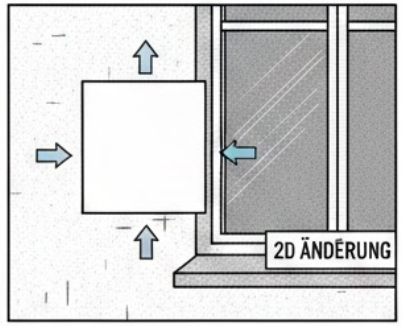
KANTE / EDGE
1D INFO



EDGE

RICHTUNG EMPFINDLICH

ECKE / CORNER
MAX. INFO



CORNER

EINZIGARTIGK

Harris Corner Detector (Der Klassiker)

Idee (1988): Suche kleine Fenster im Bild, bei denen sich die Intensität drastisch ändert, wenn man das Fenster leicht verschiebt (u,v).

Mathematik (vereinfacht): Analyse der Eigenwerte (λ_1,λ_2) der Strukturmatrix.

$$\lambda_1 \approx 0, \lambda_2 \approx 0 \rightarrow \text{ Flach}$$

$$\lambda_1 \gg 0, \lambda_2 \approx 0 \rightarrow \text{ Kante}$$

$$\lambda_1 \gg 0, \lambda_2 \gg 0 \rightarrow \text{ Ecke!}$$

Problem: Harris ist drehungsinvariant, aber nicht skalierungsinvariant (Zoom zerstört die Ecke).

Blobs vs. Ecken

Manchmal sind Ecken nicht gut (z.B. ein runder Punkt, eine Sonnenblume).

Blob: Eine Region, die sich von ihrer Umgebung unterscheidet (heller Fleck auf dunklem Grund).

Laplacian of Gaussian (LoG): Ein Filter, der auf "Blobs" verschiedener Größen reagiert.

Wichtig: SIFT und SURF basieren eher auf Blob-artigen Strukturen als auf harten Ecken.

Die Invarianz-Herausforderung (SIFT/SURF)

"Die heiligen Grale der Invarianz"

Damit ein Algorithmus robust ist, muss er folgende Änderungen ignorieren:

- **Rotationsinvarianz**: Objekt steht auf dem Kopf.
- **Skalierungsinvarianz**: Objekt ist nah oder fern.
- **Beleuchtungsinvarianz**: Tag vs. Nacht / Schatten.
- **Sichtpunktinvarianz** (Affine): Kamera schaut schräg auf das Objekt.

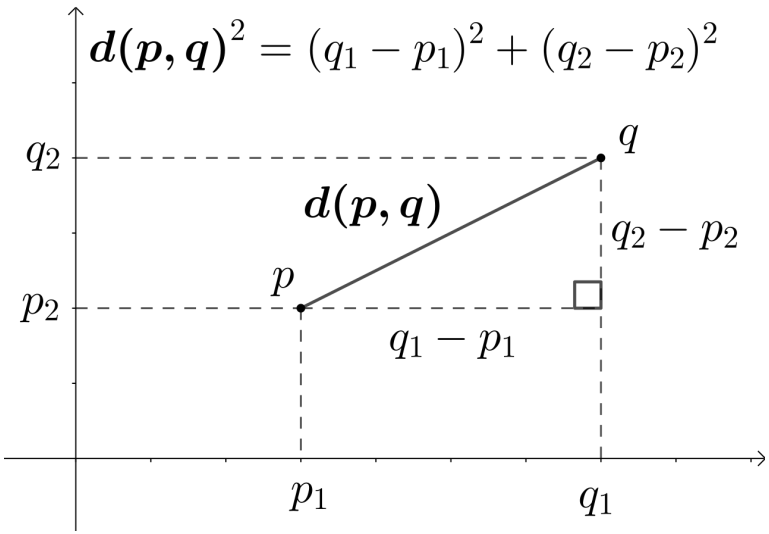
“Welcher Punkt gehört zu welchem?”

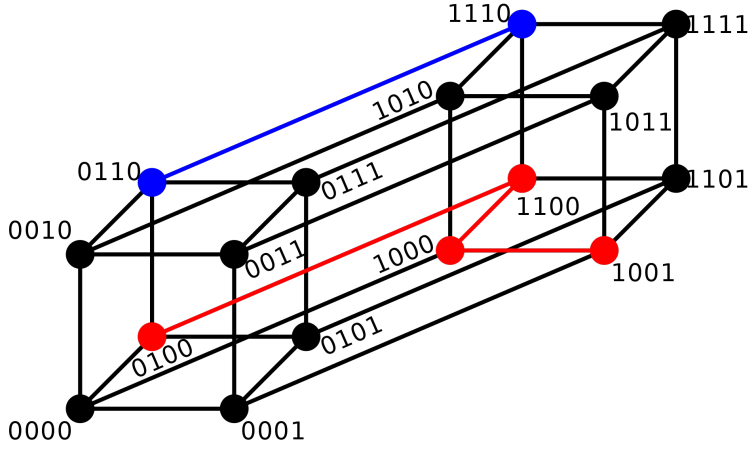
Aufgabe: Für jeden Deskriptor in Bild A den ähnlichsten Deskriptor in Bild B finden

Ähnlichkeitsmaße:

Euklidische Distanz für SIFT/SURF

Hamming-Distanz für binäre Deskriptoren (ORB, BRIEF)





SIFT (Scale-Invariant Feature Transform)

Der Durchbruch: Entwickelt von David Lowe (1999/2004). War lange der Goldstandard (und patentiert!).

Pipeline:

1. Scale-Space Extrema Detection: Suche Keypoints über verschiedene Bildgrößen (Bildpyramiden).
2. Keypoint Localization: Präzise Positionierung (Sub-Pixel-Genauigkeit).
3. Orientation Assignment: Bestimme die Hauptrichtung des Keypoints (für Rotationsinvarianz).
4. Keypoint Descriptor: Erstelle den Fingerabdruck.

SIFT – Scale Space (Der Zoom-Trick)

Wie finden wir Keypoints unabhängig von der Größe?

Difference of Gaussians (DoG):

1. Wir machen das Bild unscharf (Gauß) und ziehen es vom Original ab.
2. Das machen wir in verschiedenen Größen (Oktaver).

Lokale Maxima in diesem 3D-Raum (x, y, σ) sind unsere Keypoints.

Grafik: (Stapel von Bildern, die immer unschärfer und kleiner werden).

SIFT – Der Deskriptor (Der Fingerabdruck)

Wir haben den Punkt (x,y). Wie beschreiben wir ihn?

Wir betrachten ein 16x16 Pixel Fenster um den Punkt.

Unterteilung in 4x4 Sub-Blöcke.

Berechnung von Gradienten-Histogrammen (8 Richtungen) in jedem Block.

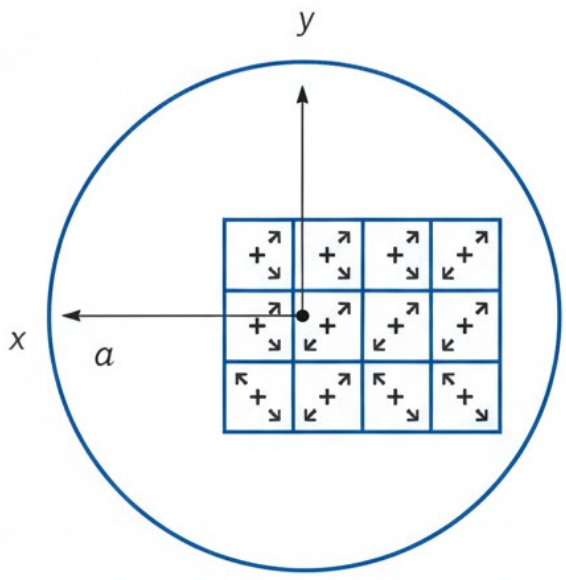
Ergebnis: Ein Vektor mit 4x4x8 = 128 Werten.

Dieser Vektor ist der “Ausweis” des Features.

SIFT Visualisierung

Ein SIFT-Feature besteht aus:

- Position (x,y).
- Größe (Kreisradius).
- Orientiretung (Strich im Kreis).
- Deskriptor (Datenvextor).



SURF (Speeded-Up Robust Features)

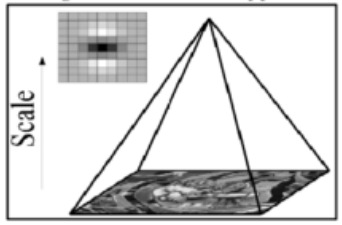
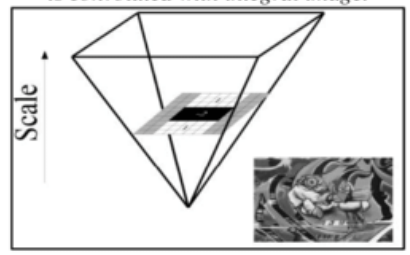
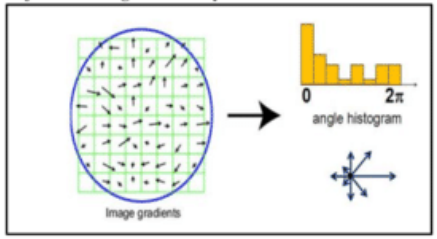
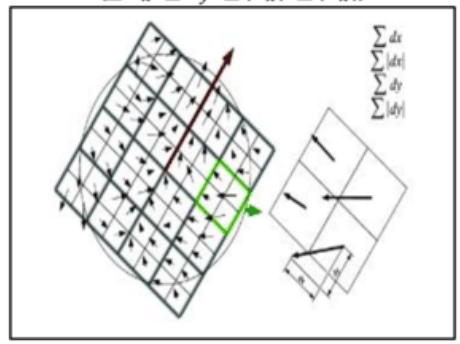
Motivation: SIFT war langsam.

Optimierung: Nutzt Integral Images (Summed Area Tables) und eine Näherung der Hesse-Matrix (Box Filter).

Unterschied:

- SIFT Deskriptor: 128 Dimensionen.
- SURF Deskriptor: 64 Dimensionen (schneller zu vergleichen).

War ebenfalls lange patentiert.

Scale Space	*Difference of Gaussian (DoG) is convolved with different size of images with same size of filter.*  *Fig. 6. Fix filter is convoluted with down sampling images*	*Different size of box filter(Laplacian of Gausian (LoG)) is convoluted with integral image.*  *Fig. 7. Fix image is convoluted with up sampling filters.*
Key point detection	*Using of local extrema detection, apply Non maxima suppression and eliminate edge response with Hessian matrix*	*Determine the key points with Hessian matrix and Non Maxima suppression.*
Orientation	*Image gradiant magnitude and orientations are sampled around the key point location, using the scale of the key point to select the level of Gaussian blur for the image. Orientation of histogram is used for same.*	*A sliding Orientation window of size $\pi/3$ detects the dominant orintation of the Gaussian weighted Haar Wavelet responses at every sample point with in a circular neighbourhood around the interest points.*
Descriptor	*The key point descriptor allows for significant shift in gradiant positions by creating orientation histograms over 4 x 4 sample regions. The figure shows 8 directions for each orientation histogram, with the length of each arrow corresponding to the magnitude of the histogram entry.*  *Fig. 8. Orientation assignment*	*An orientation quadratic grid with 4x4 square sub regions is laid over the interest point. For each square, the wavelet responses are computed from 5x5 samples.* *Descriptor of SURF is* $V = (\sum d_x, \sum d_y, \sum \lvert d_x \rvert, \sum \lvert d_x \rvert)$  *Fig. 9: Orientation assignments*
Size of descriptor	*128 bits*	*64 bits*

Moderne & Schnelle Alternativen (ORB)

Warum nicht SIFT/SURF?

Lizenz: Waren patentgeschützt (mittlerweile ausgelaufen, aber riskant für kommerzielle Produkte in Legacy-Code).

Performance: Zu langsam für Echtzeit-Anwendungen auf Handys oder Drohnen.

Die Lösung: Binäre Deskriptoren.

ORB (Oriented FAST and Rotated BRIEF)

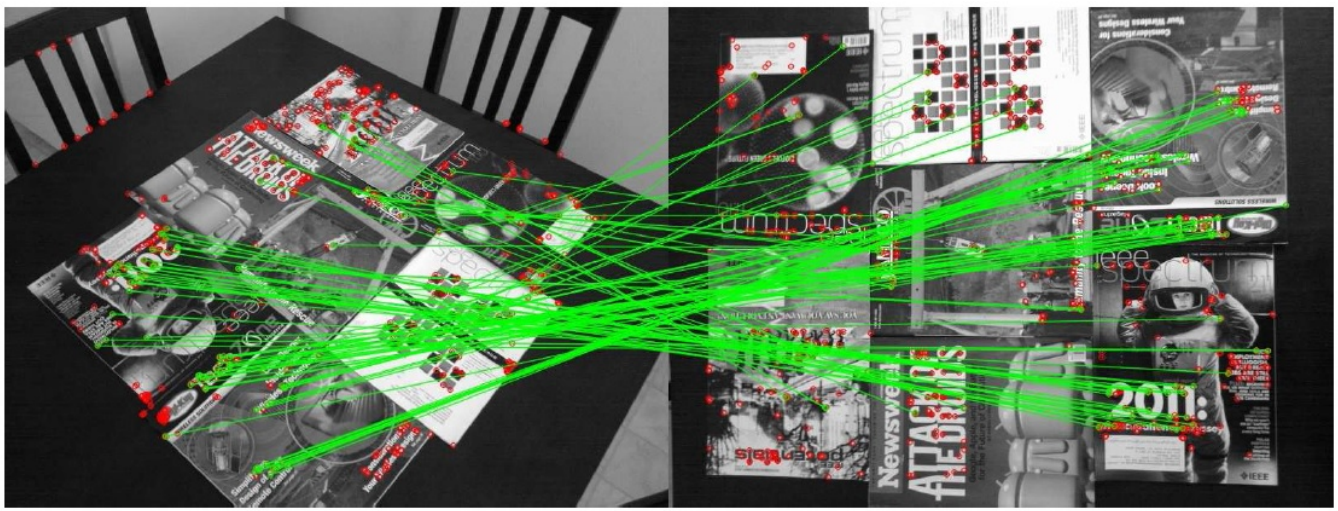
Entwickelt von OpenCV Labs (2011).

Ziel: Ein freier, effizienter Ersatz für SIFT/SURF.

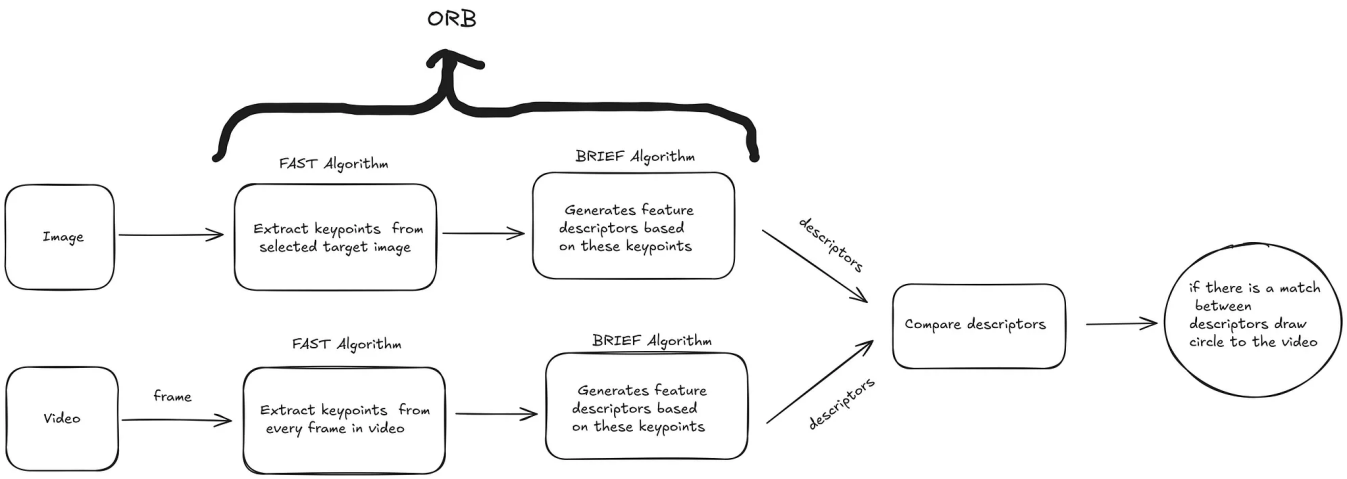
Kombination aus zwei Techniken:

- **Detektor**: FAST (Findet den Punkt).
- **Deskriptor**: BRIEF (Beschreibt den Punkt).

ORB fügt Rotationsinvarianz hinzu (das “Oriented” und “Rotated”).



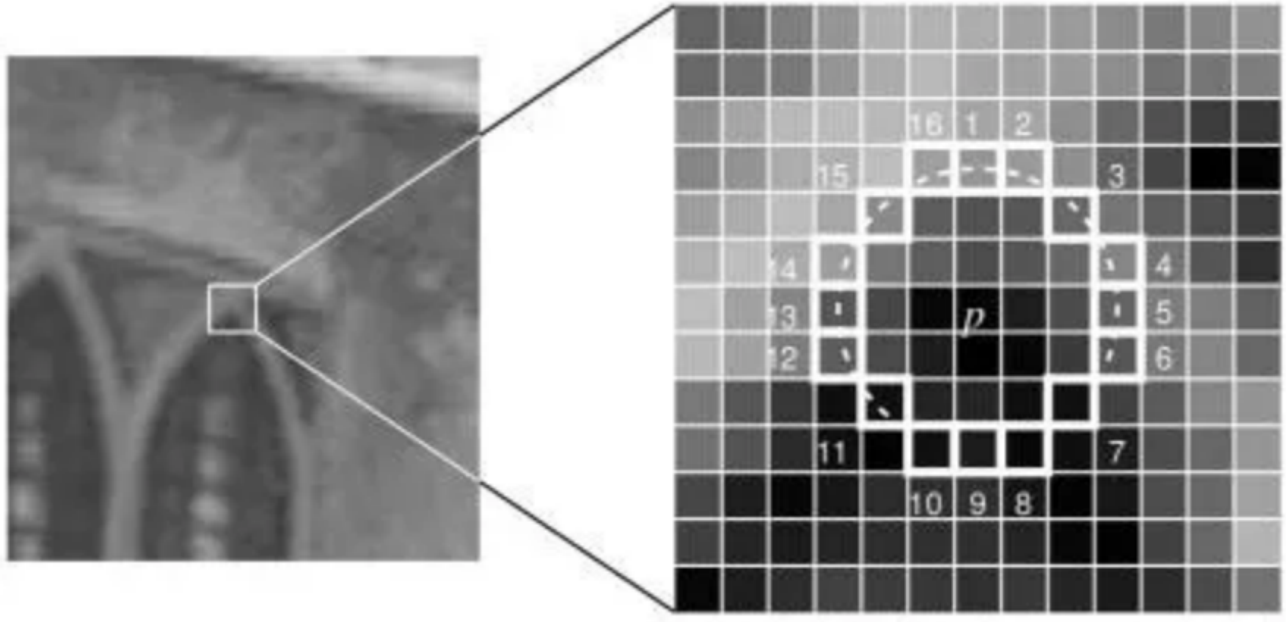
ORB (Oriented FAST and Rotated BRIEF)



FAST Detector (Features from Accelerated Segment Test)

Konzept: Nur auf Speed optimiert.

Der Test: Betrachte einen Kreis von 16 Pixeln um einen Punkt P.



Ist P eine Ecke? Ja, wenn N zusammenhängende Pixel im Kreis alle deutlich heller (oder dunkler) sind als P.

Keine Ableitungen, nur if ... $>$ ... then Vergleiche.

BRIEF Deskriptor (Binary Robust Independent Elementary Features)

SIFT nutzt Gleitkommazahlen (Vektor aus 128 Floats).

Das kostet Speicher und Rechenzeit.

Idee: Wir vergleichen einfach Pixel-Paare rund um den Keypoint.

Ist Pixel A heller als Pixel B? $\rightarrow$ 1Sonst $\rightarrow$ 0

Ergebnis: Ein Bitstring (z.B. 256 Bits: 100101110...).

Zusammenfassung Detektoren

Algorithmus	Genauigkeit	Geschwindigkeit	Deskriptor-Art
SIFT	Sehr Hoch	Langsam	Float Vektor (L2-Norm)
SURF	Hoch	Mittel	Float Vektor (L2-Norm)
ORB	Mittel-Hoch	Sehr Schnell	Binär (Hamming)

Matching-Strategien (Matches finden)

Matching – Wer passt zu wem?

Wir haben 1000 Features in Bild A und 1000 Features in Bild B.

Wie finden wir Paare?

Distanzmessung:

Für SIFT/SURF (Floats): Euklidische Distanz (L2-Norm).

Für ORB (Binär): Hamming Distanz (XOR Operation + Bit Count).
Sehr schnell auf CPUs!

Brute-Force Matcher

Strategie: Nimm Deskriptor 1 aus Bild A und vergleiche ihn mit allen Deskriptoren aus Bild B.

Nimm den mit der kleinsten Distanz.

Komplexität: $O(N \times M)$.

Eignung: Gut für kleine Datensätze (< 500 Features).

FLANN (Fast Library for Approximate Nearest Neighbors)

Strategie: Nutzt KD-Trees oder K-Means Bäume, um den Suchraum zu strukturieren.

Es wird nicht jeder Punkt geprüft, sondern nur die in der “Nähe”.

Trade-off: Viel schneller ($O(\log N)$), aber findet vielleicht nicht den absolut besten Match (Approximate).

Standard für große Anwendungen.

Filterung – Lowe's Ratio Test

Ein Match ist nicht immer ein guter Match.

Problem: Wenn ein Feature in Bild B gar nicht existiert, findet der Matcher trotzdem den "nächsten" (falschen) Nachbarn.

Lösung (Ratio Test):Finde die zwei besten Matches für einen Punkt (Nearest Neighbor 1 & 2).

Ist Distanz(1) viel kleiner als Distanz(2)? (z.B. $< 0.75\times$ Distanz2).

Wenn ja: Der Match ist eindeutig $\rightarrow$ Behalten.

Wenn nein: Der Match ist mehrdeutig $\rightarrow$ Verwerfen.

Bilregistrierung (Image Registration)

Ziel: Zwei oder mehr Bilder derselben Szene geometrisch ausrichten

Anwendungen:

- Panoramafotografie (Stitching)
- Medizinische Bildverarbeitung (MRT/CT Fusion)
- Satellitenbildanalyse
- Objekt-/Szenenerkennung aus verschiedenen Blickwinkeln
- Augmented Reality (Virtuelle Objekte in reale Welt einfügen)

Kernidee: Finde Transformation, die Bild B optimal an Bild A anpasst

Ablauf Bilregistrierung

1. FEATURE DETECTION Keypoints in beiden Bildern finden (SIFT, SURF, ORB von letzter Sitzung)
2. FEATURE MATCHING

   Korrespondierende Keypoints paaren
3. TRANSFORMATION SCHÄTZEN

   Beste geometrische Transformation berechnen (Homographie für ebene Szenen)
4. RE-PROJECTION

   Bild B in Koordinatensystem von Bild A transformieren

Das Problem mit False Matches

Nicht alle Matches sind korrekt!

Gründe für falsche Matches:

- Ähnliche Texturen an verschiedenen Stellen
- Wiederholende Muster
- Verdeckungen
- Perspektivische Verzerrungen

Konsequenz: Viele Matches sind “Outlier” (Ausreißer) $\rightarrow$ Verfälschen die Transformationsberechnung

Lösung: Wir brauchen einen robusten Algorithmus, der Outlier ignoriert!

Homographie-Schätzung

Projektive Transformation für ebene Szenen

Annahme: Szene ist planar oder Kamera rotiert um ihr Zentrum

Homographie-Matrix H (3×3):

$$H = \begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{pmatrix}$$

Transformation:

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

Eigenschaften: Erhält Kollinearität, aber nicht Längen/Winkel

Das Homographie-Problem mathematisch

Wie viele Matches brauchen wir? Jedes Punktpaar $(x, y) \leftrightarrow (x', y')$ gibt 2 Gleichungen:

$$x' = (h11 * x + h12 * y + h13)/(h31 * x + h32 * y + 1)$$

$$y' = (h21 * x + h22 * y + h23)/(h31 * x + h32 * y + 1)$$

8 unbekannte Parameter (h11..h32, da h33=1 skaliert)

Benötige mindestens 4 Punktpaare (4 x 2 = 8 Gleichungen)

Problem: Mit Ausreißern bekommt man falsche H!

RANSAC (RANdom SAmple Consensus) als Rettung!

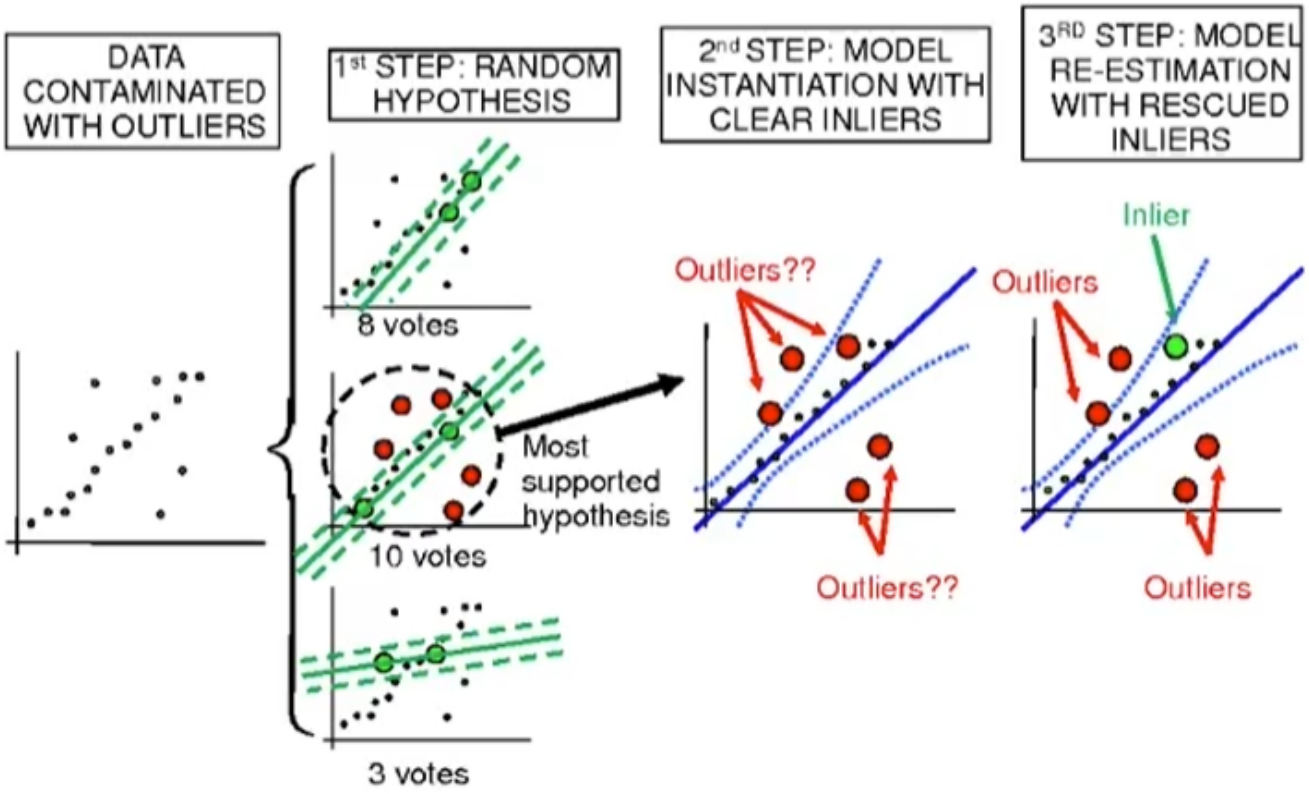
Robuster Schätzer für Modelle mit Ausreißern

Idee: Finde Modellparameter, die zu den meisten Datenpunkten passen

RANSAC-Algorithmus:

1. Wähle zufällig 4 Matches (Minimum Sample Set)
2. Berechne Homographie H aus diesen 4 Punkten
3. Teste H auf ALLEN Matches:
   - Projektion berechnen
   - Wenn Abstand $<$ Threshold $\rightarrow$ INLIER
   - Sonst $\rightarrow$ OUTLIER
4. Zähle Inlier
5. Wiederhole N-mal (iteriere)
6. Wähle H mit den meisten Inliern
7. Berechne H neu aus ALLEN Inliern (Least Squares)

RANSAC




```
Iteration 1:        Iteration 2:        Iteration 3:

o o o o o o         o o o o o o         o o o o o o
o o i i o o         o o o o o o         o o o o o o
o i R i i o         o o o i o o         o i i i i o
o o i o o o         o i i i i o         o i R i i o
o o o o o o         o o i o o o         o o i o o o
```

Performance-Optimierungen

1. **Vorverarbeitung**:

- Histogram Equalization für bessere Kontraste
- Rauschentfernung mit Gaussian/Median Filter

2. **Feature-Matching verbessern**:

- Lowe's Ratio Test: distance(match1) $<$ 0.7 * distance(match2)
- Symmetrie-Test: Bidirektionale Überprüfung

3. **RANSAC Parameter tuning**:

- reprojThreshold: 1-5 Pixel für genaue Registrierung
- Erhöhe maxIters bei vielen Outliern (>50%)

4. **Post-Processing**:

- Multi-Band Blending für unsichtbare Nahtstellen
- Exposure Compensation für gleichmäßige Helligkeit

Grenzen und Fallstricke

Wann funktioniert es NICHT?

- Nicht-planare Szenen: Homographie gilt nur für Ebenen!
- Parallaxe: Bei unterschiedlicher Kameraposition
- Extreme Verzerrungen: $> 45°$ Rotation oder starke Skalierung
- Zu wenige/zu ähnliche Texturen: Nicht genug unterscheidbare Features
- Dynamische Szenen: Bewegende Objekte stören

Key Takeaways

Feature Matching verbindet korrespondierende Punkte zwischen Bildern

- Brute-Force: Genau aber langsam
- FLANN: Schnell aber approximativ

Homographie beschreibt projektive Transformation zwischen Ebenen

- 3x3 Matrix mit 8 Freiheitsgraden
- Benötigt mindestens 4 Punktkorrespondenzen

RANSAC macht die Schätzung robust gegen Outlier

- Findet automatisch die beste Punktmenge