Symmetrieerkennung in Theorie und Praxis

FH Wedel Entwicklung trifft FU Berlin Forschung

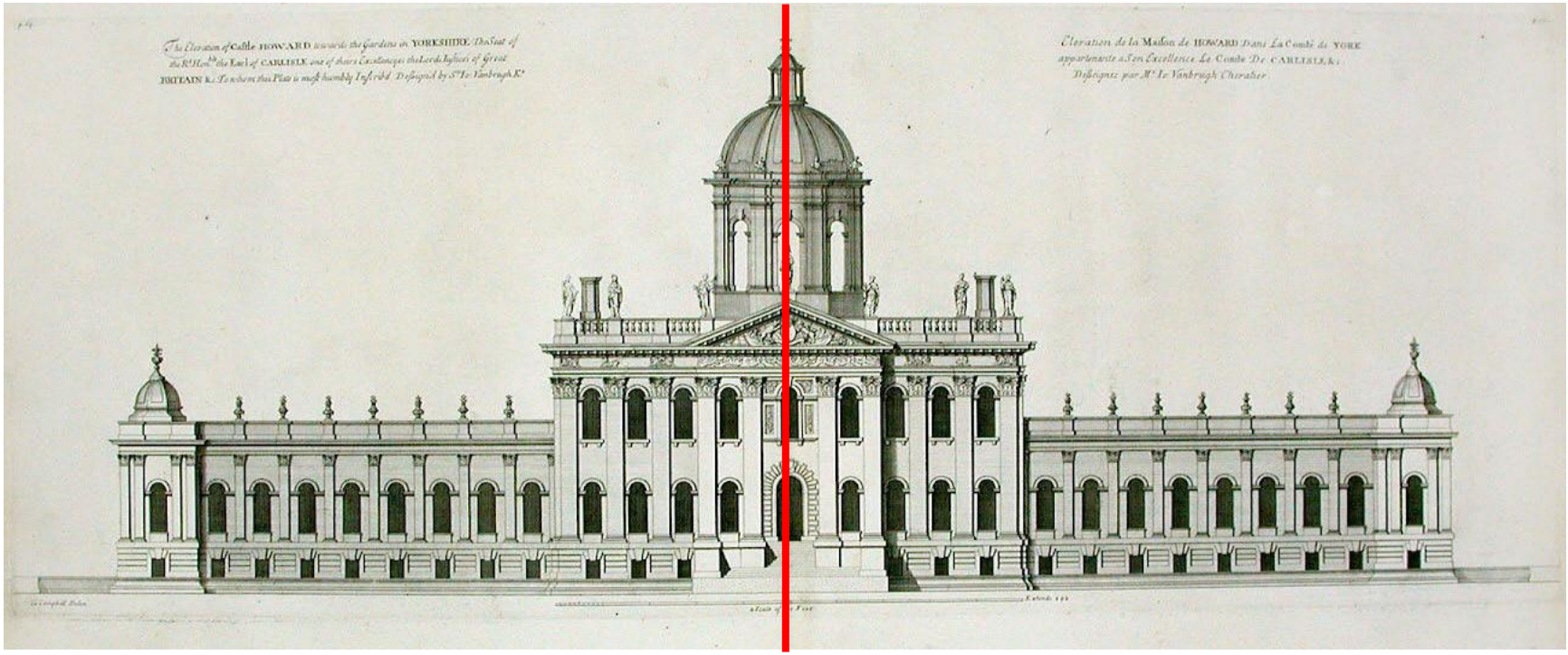
Timm Hoffmann & Marcus Riemer

“Symmetry is a complexity-reducing concept […]; seek it everywhere”

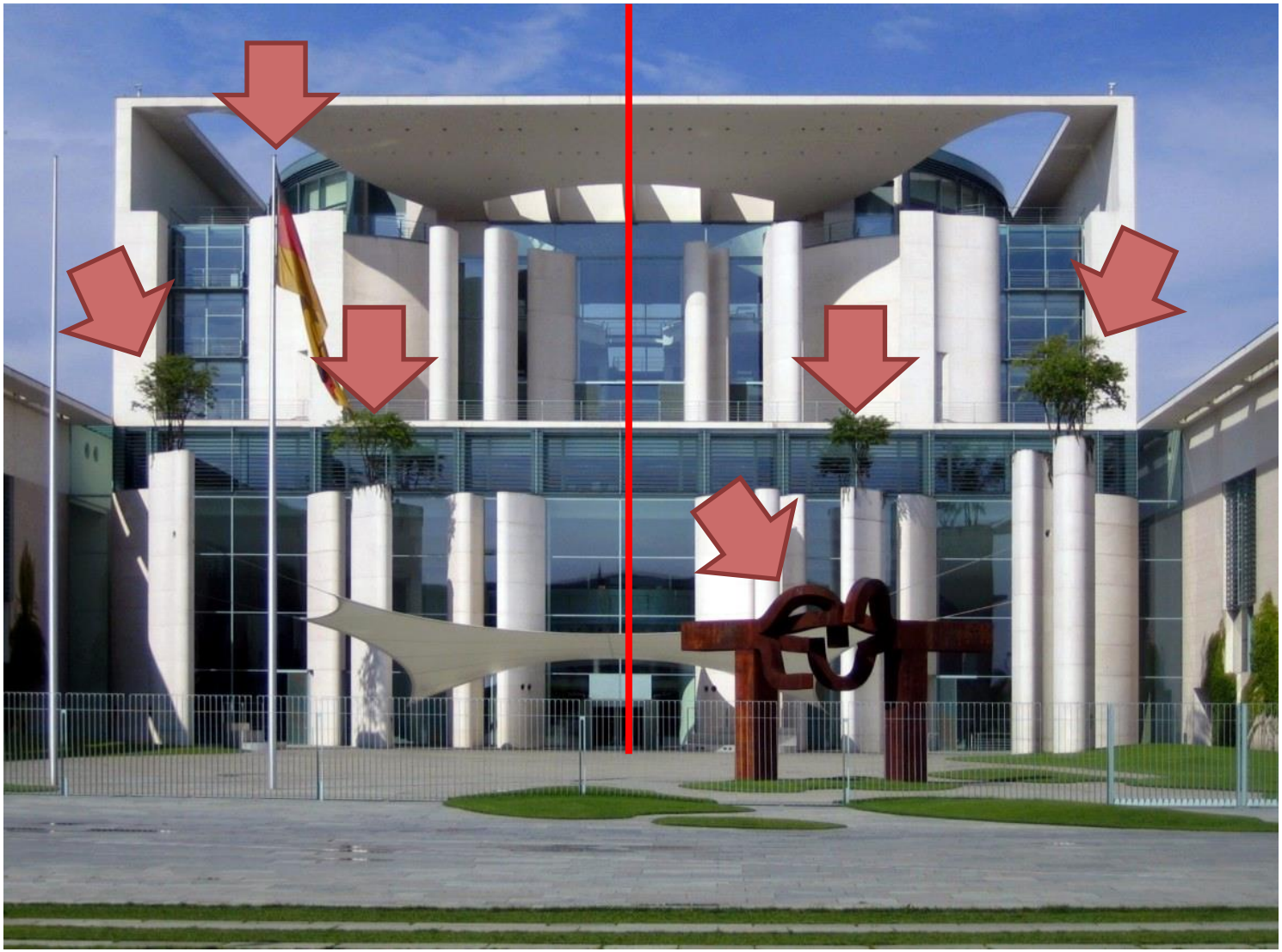
- Alan J. Perlis

EINFÜHRUNG

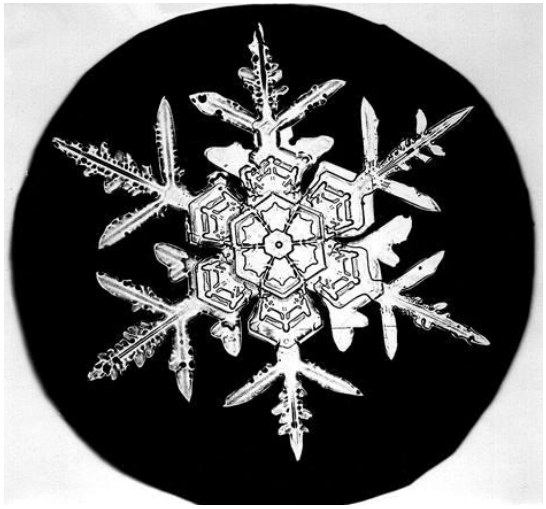
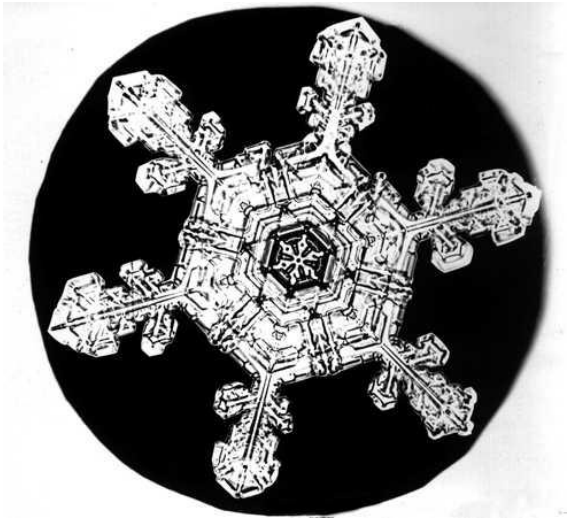
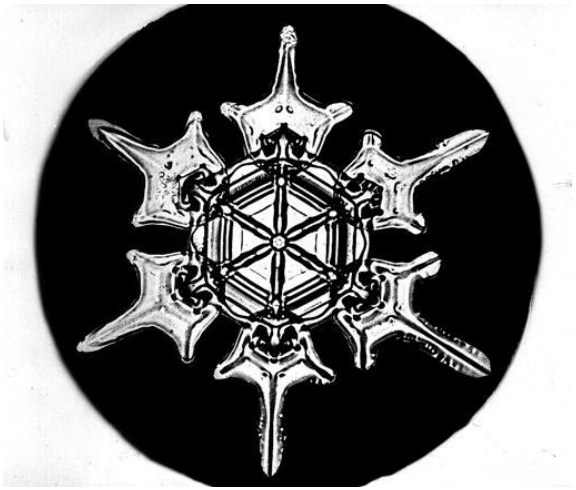
Symmetrien im Alltag

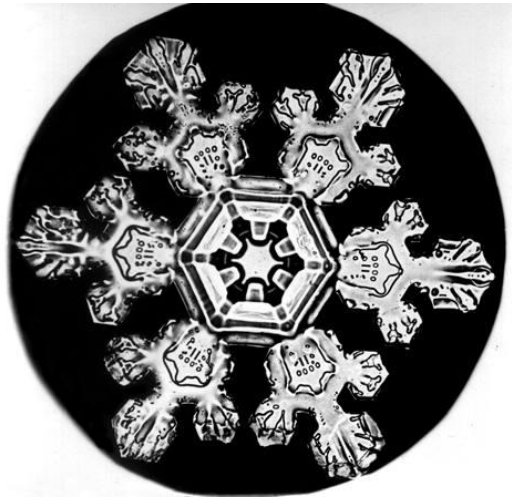
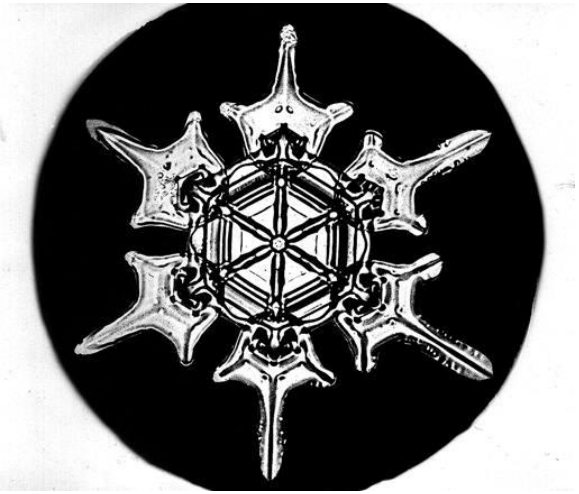


Symmetrien im Alltag

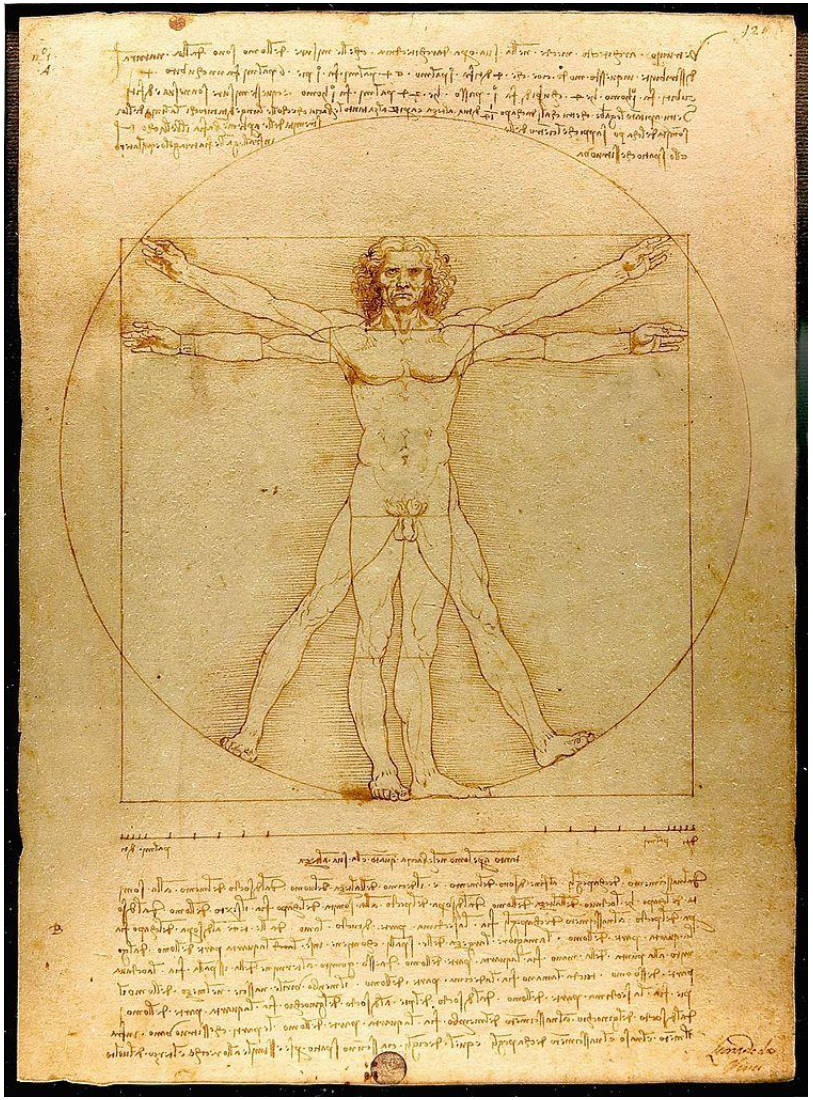


Symmetrien im Alltag

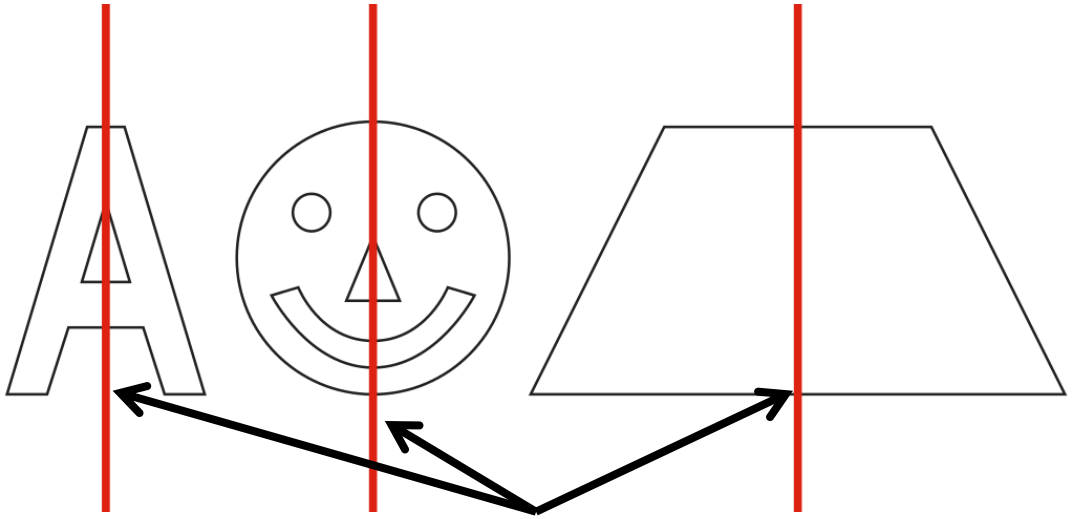




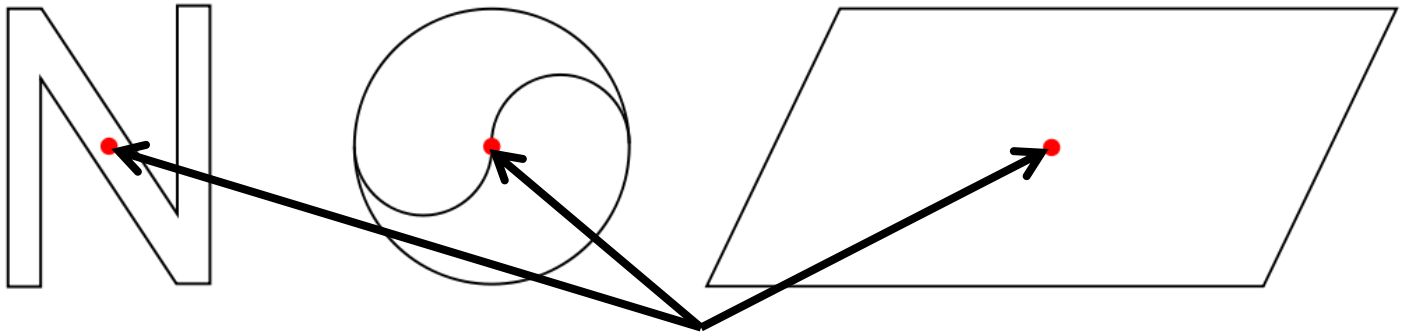
Symmetrien im Alltag



Wiederholung Analysis



Achse



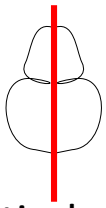
Punkt

Achsen- und Rotationssymmetrie

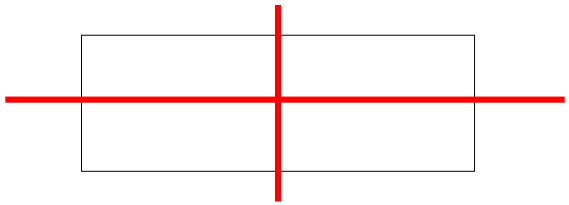
Diedergruppe D_n

n Spiegelachsen durch den Mittelpunkt

D_1 = Einzelne Achsenspiegelung



D_2 = Nicht quadratisches Rechteck



D_n = Regelmäßiges n-Eck

Zyklische Gruppe C_n

Alle Drehungen um einen Punkt um Vielfache von $\frac{360°}{n}$

C_1 = Komplett unsymmetrisches Objekt

C_2 = Punktspiegelung

C_3 = Triskele

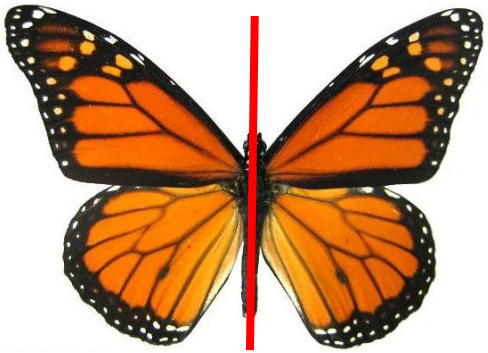
C_4 = Swastika

$D_n \rightarrow C_n$

Symmetrie

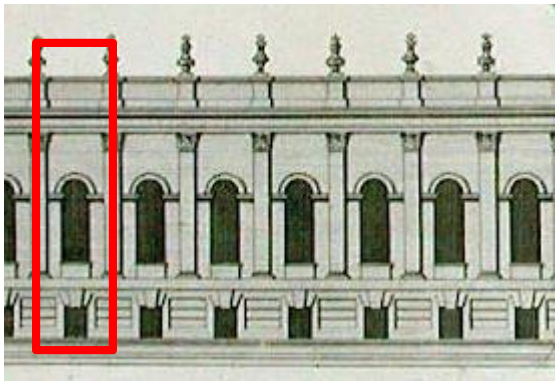
Symmetrie ist die Gleichheit bezüglich bestimmten Klassen von Transformationen

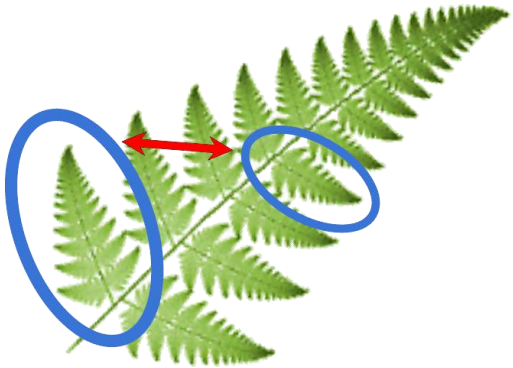
Spiegelung



Rotation

Translation





Spiegelung
Rotation
Translation
Skalierung

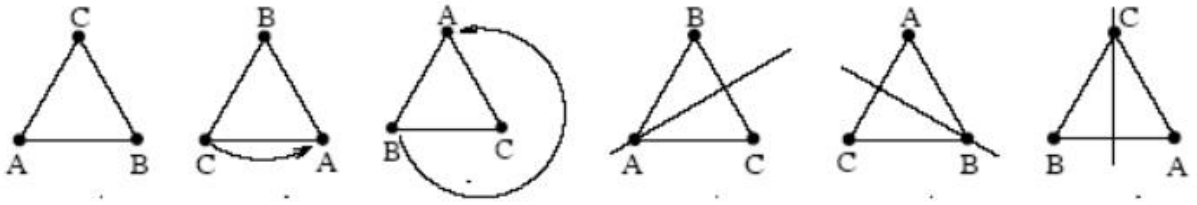
Symmetriegruppe

In der mathematischen Gruppentheorie ist die Symmetriegruppe eines geometrischen Objektes die Gruppe, die aus der Menge aller Kongruenzabbildungen besteht, die das Objekt auf sich selbst abbilden, zusammen mit der Verkettung von Abbildungen als Gruppenoperation.

Symmetriegruppe

In der mathematischen **Gruppentheorie** ist die Symmetriegruppe eines geometrischen Objektes die Gruppe, die aus der **Menge** aller **Kongruenzabbildungen** besteht, die das Objekt auf sich selbst abbilden, zusammen mit der **Verkettung** von Abbildungen als **Gruppenoperation**.

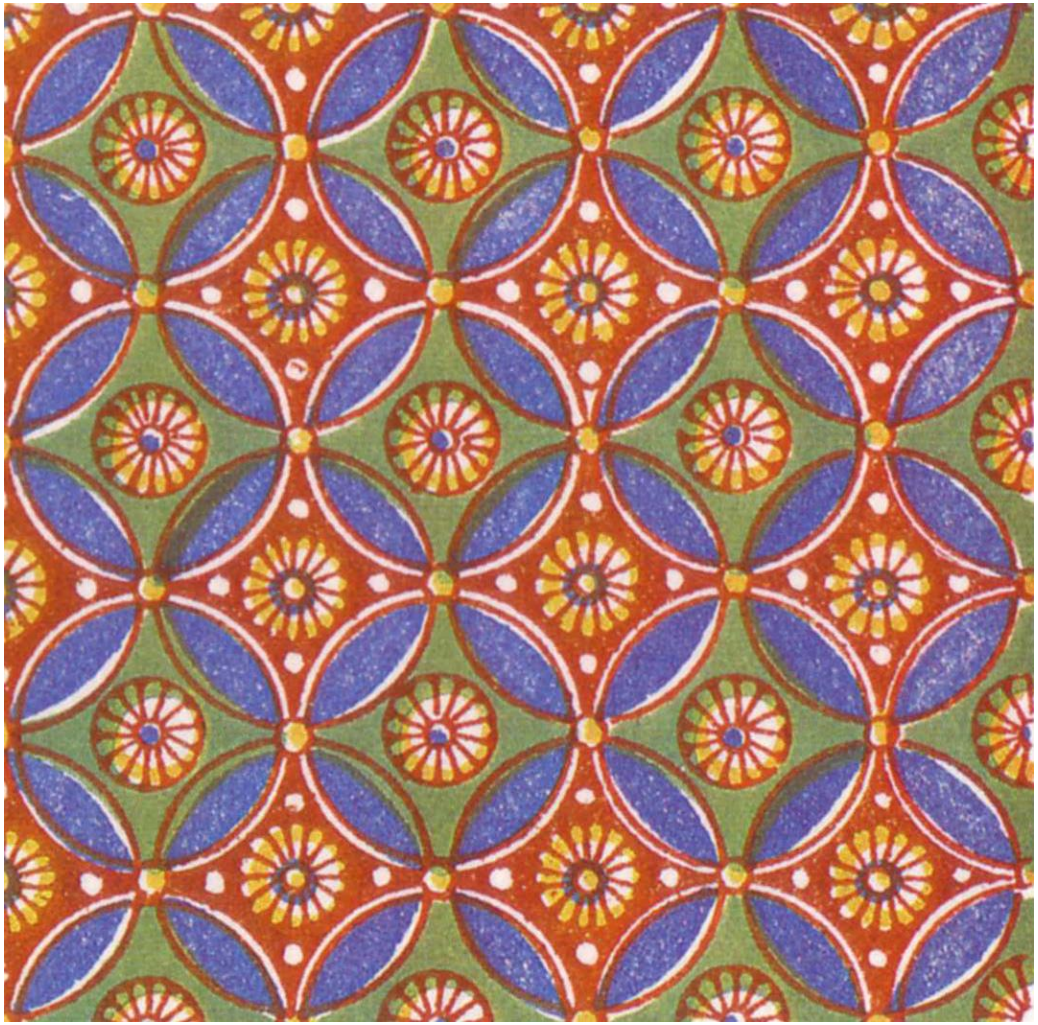
Endliche Symmetriegruppen



$(S_3, \circ)$:

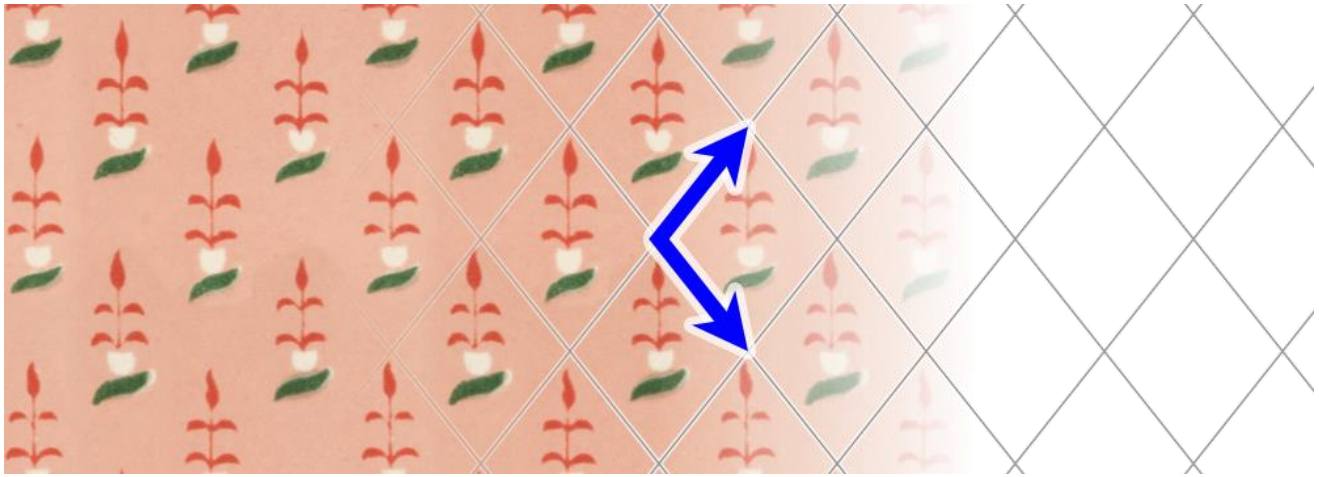
$\circ$	ρ_0	ρ_1	ρ_2	σ_l	σ_r	σ_o
ρ_0	ρ_0	ρ_1	ρ_2	σ_l	σ_r	σ_o
ρ_1	ρ_1	ρ_2	ρ_0	σ_o	σ_l	σ_r
ρ_2	ρ_2	ρ_0	ρ_1	σ_r	σ_o	σ_l
σ_l	σ_l	σ_r	σ_o	ρ_0	ρ_1	ρ_2
σ_r	σ_r	σ_o	σ_l	ρ_2	ρ_0	ρ_1
σ_o	σ_o	σ_l	σ_r	ρ_1	ρ_2	ρ_0

Unendliche Symmetriegruppen

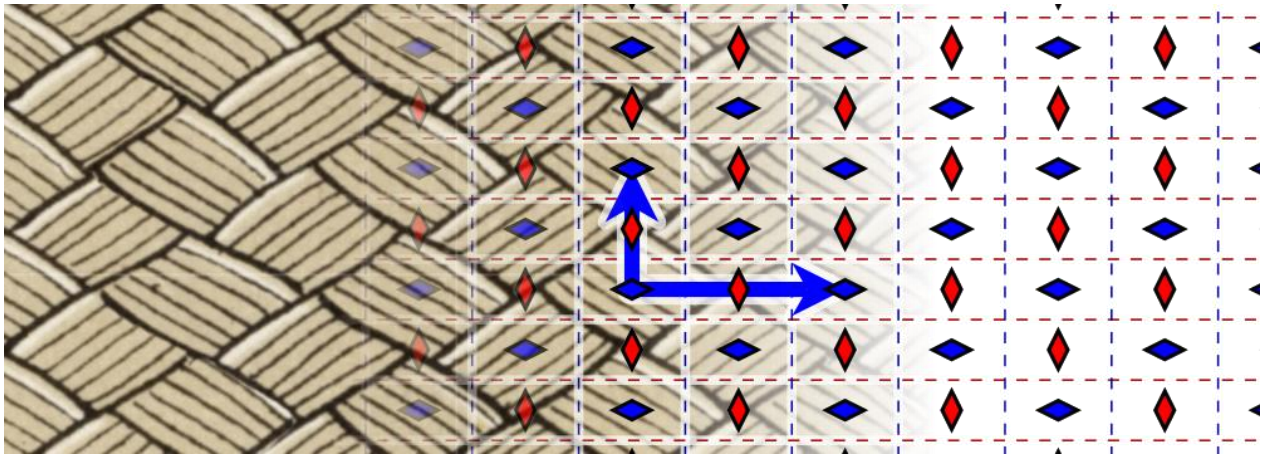


Unendliche Symmetriegruppen

p1
(Translation)



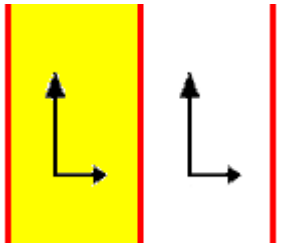
p2gg
(2 Rotationszentren + Verschiebung)

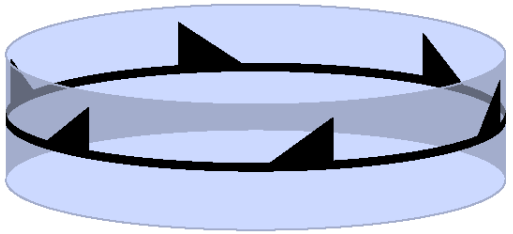


Frieze Gruppe

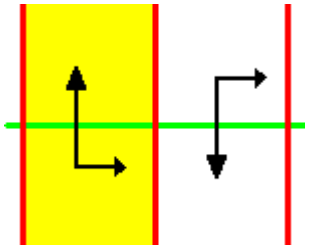
Unendlich, aber nur in eine Dimension

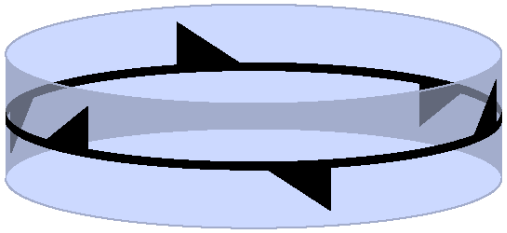
p1
(Translation)



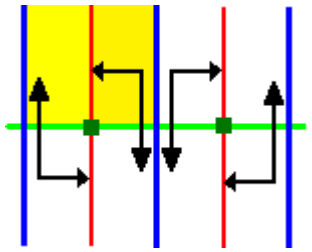


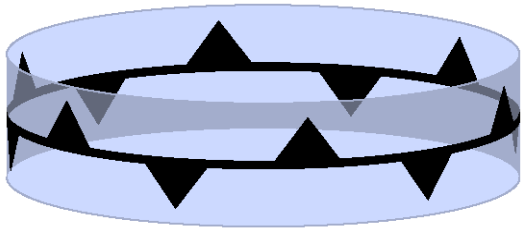
p11g
(Translation, Spiegelung)





p2mg
(Translation, Spiegelung, Rotation)





ANWENDUNG

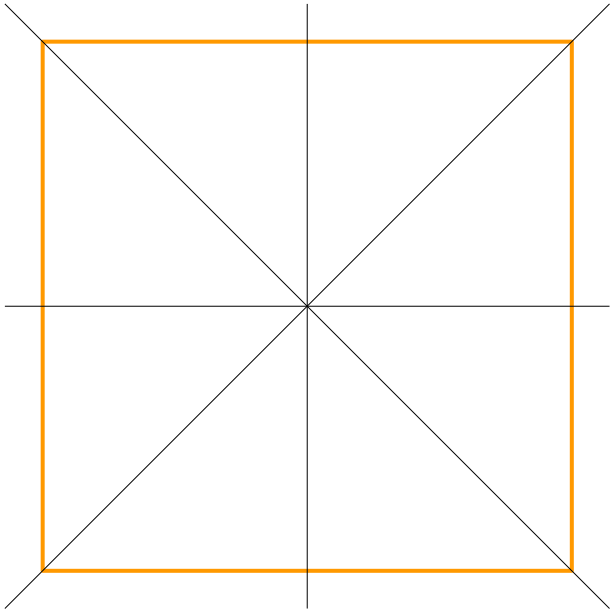
Fragestellungen

1. Erkennung einer Symmetrie

2. Beschreibung der Symmetrie (Approximation)

Erkennung

- Gegeben: Bild
- Gesucht: Symmetriegruppe



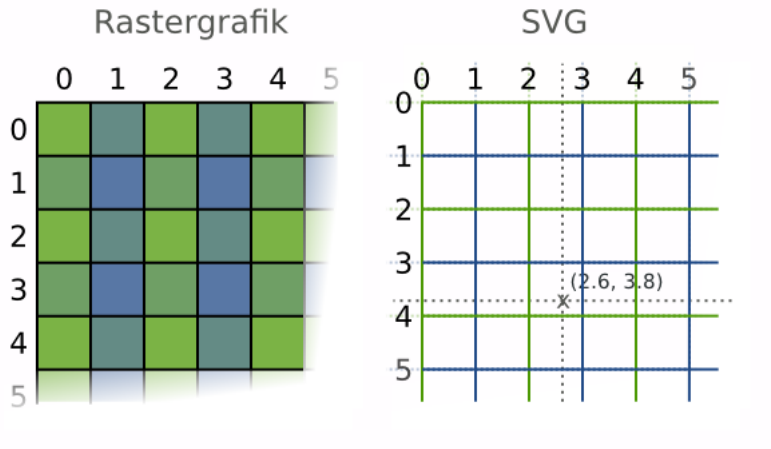
D_4

Probleme bei der Erkennung

- Bild nicht „mathematisch“ symmetrisch
  → nur intuitiv erkennbar
- Weiche Erkennung erforderlich

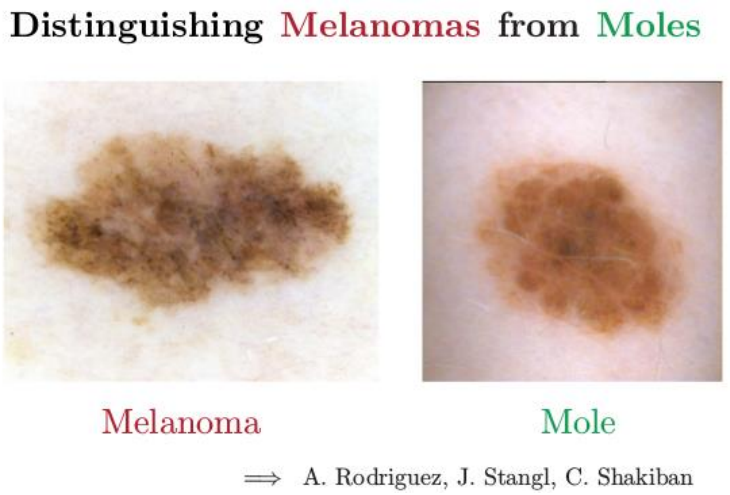
Eingabe für die Erkennung

- Vektorgrafik
  - Geometrische Beschreibung von Pfaden
  - Stetige Werte
- Pixelgrafik
  - (Farb)Wert pro Pixel
    (Zelle in einem 2D-Raster)
- Punktwolke (reduzierte Vektorgrafik)
  - „Liste von Koordinaten“



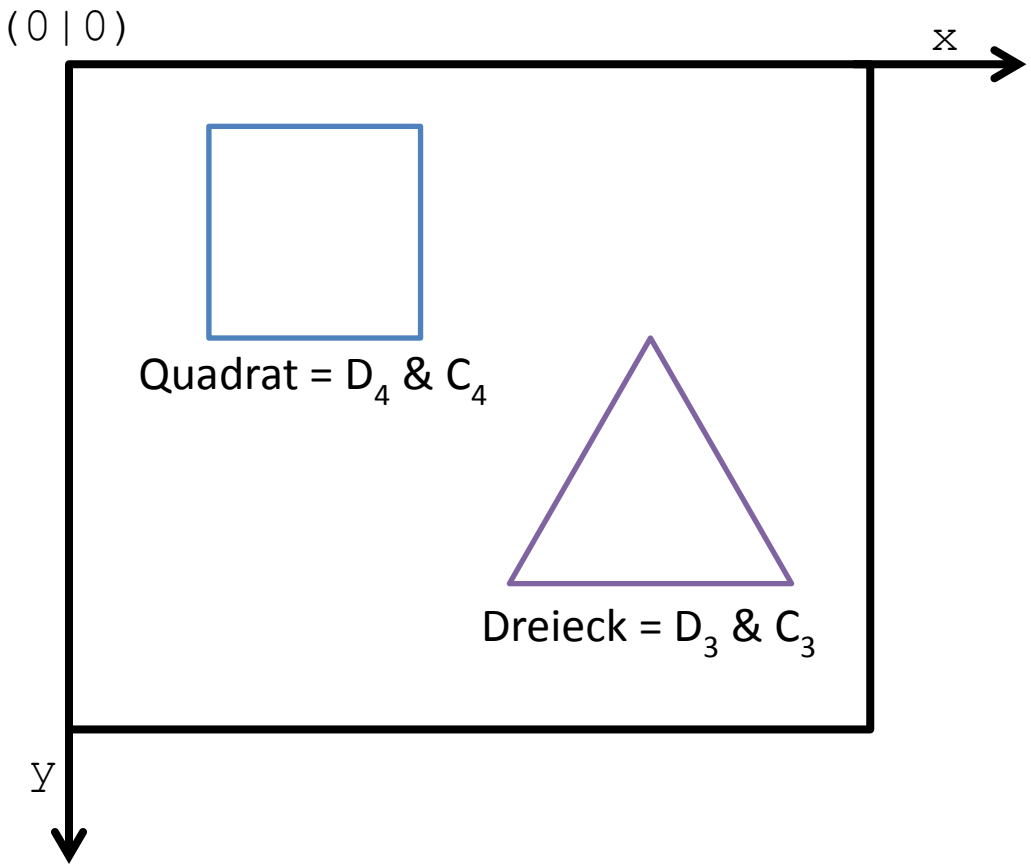
Beispiele für Einsatzbereiche

- Objekterkennung in Bildern
- Astronomische Objekte klassifizieren
- Medizin: Muttermale von Hautkrebs unterscheiden
- Bildkompression



Objekterkennung

Keinerlei Symmetrie im Gesamtbild



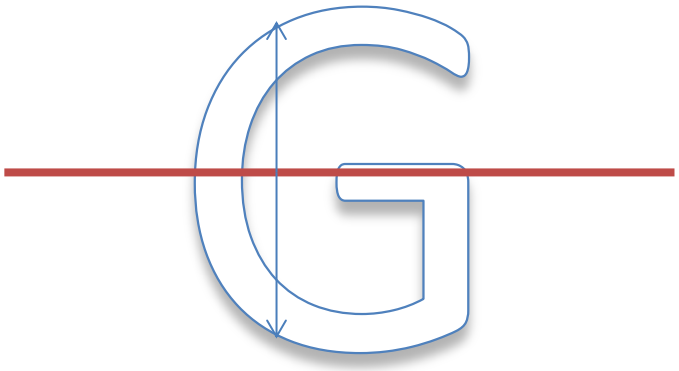
Approximation

- Gegeben:
  - Bild
  - Symmetrie-Eigenschaft (z. B. Symmetriegruppe)
- Gesucht:
  - Symmetrie-Beschreibung (z. B. konkrete Achsen)
  - Soll vorhandene Symmetrie möglichst genau annähern

Approximation

- NP-schwer für Punkte, die „zu“ nah beisammen liegen
- Ursprünglich von S. Iwanowski in seiner Doktorarbeit bewiesen
  „Approximate congruence and symmetry detection in the plane“ (1990)
- Verfeinert von C. Dieckmann
  „Symmetry Detection and Approximation“ (2012)

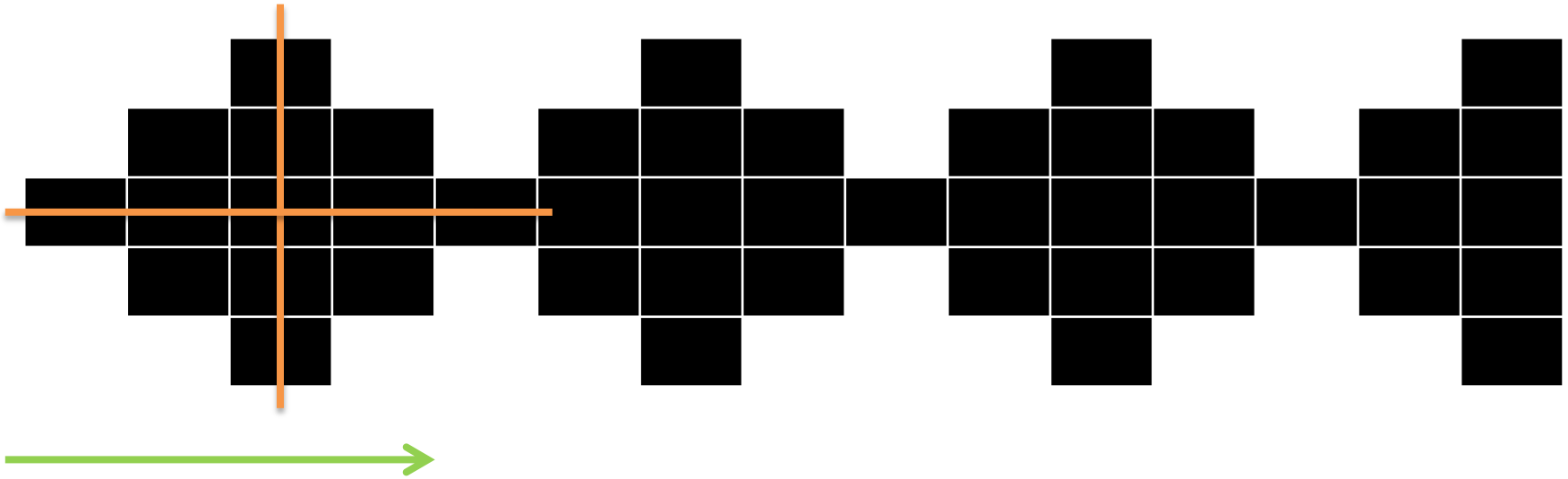
Beispiel



Erkannte Symmetriegruppe: D_1

ERKENNUNG (1)

Frieze Patterns

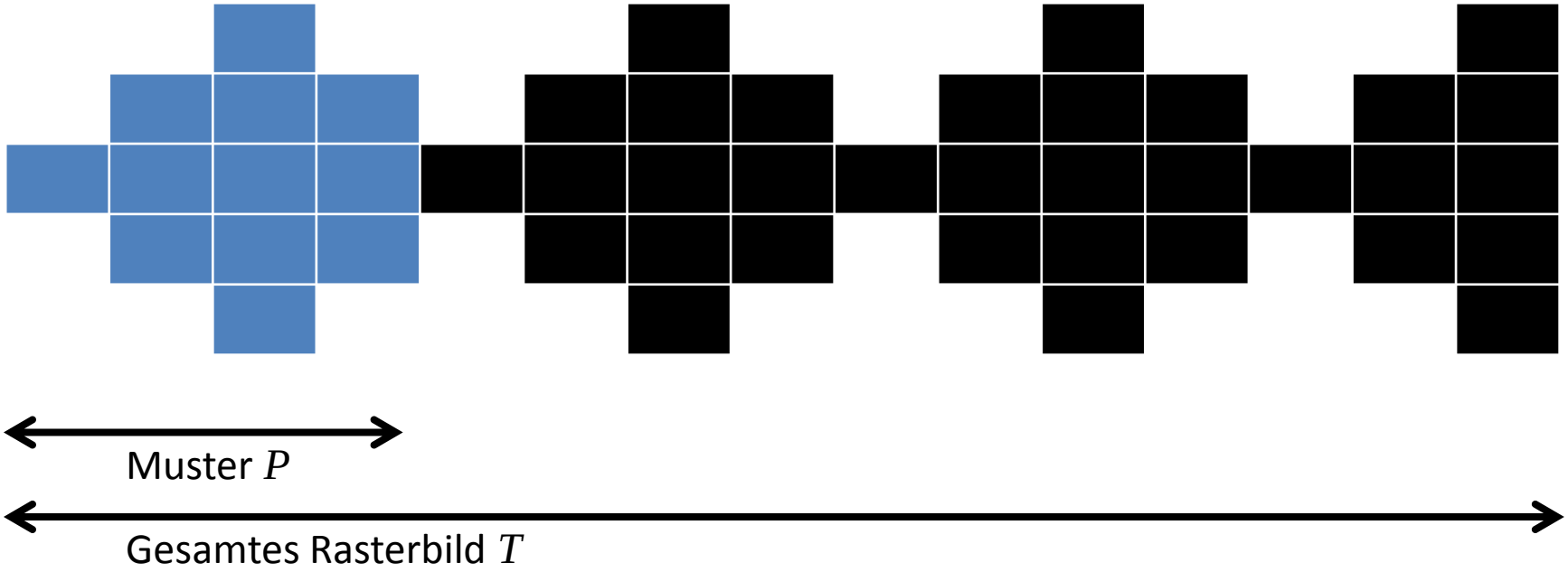


Rotation? **Nein**

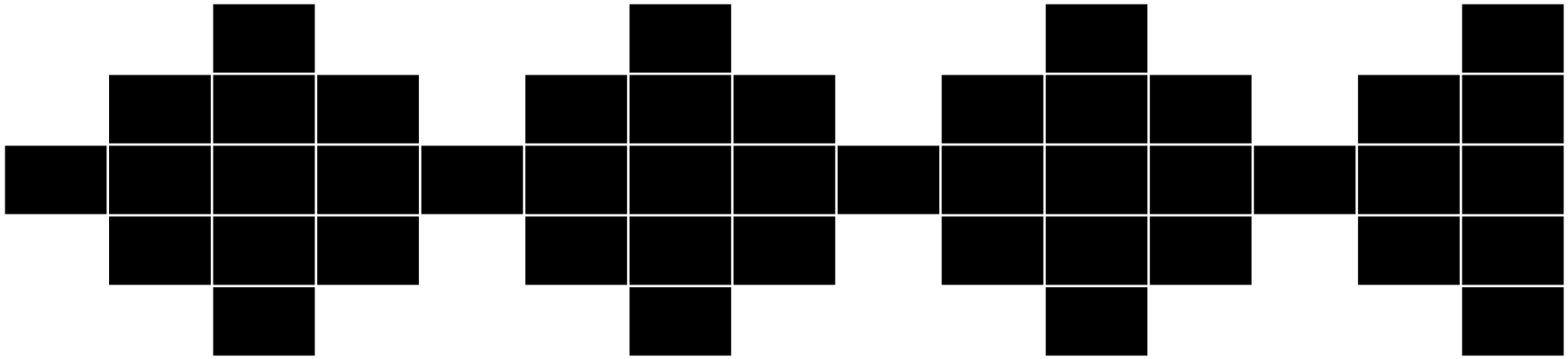
Spiegelung? **Nein**

Translation? **Ja**

Frieze Patterns



Frieze Patterns und String Matching



m

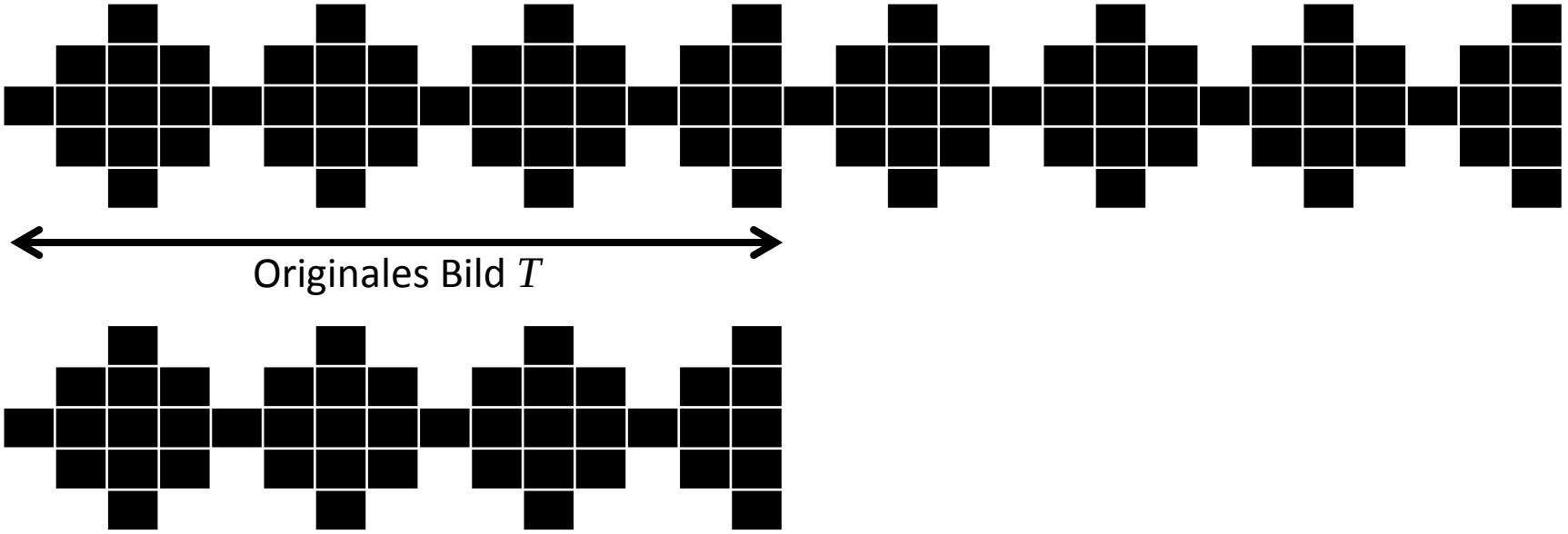
n

0	0	1	0	0	0	1	0	0	0	1	0	0	0	1
0	1	1	1	0	1	1	1	0	1	1	1	0	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0	1	1	1	0	1	1	1	0	1	1	1	0	1	1
0	0	1	0	0	0	1	0	0	0	1	0	0	0	1

Alphabet $\Sigma = \{0, 1\}$

Zeile R_i wobei $1 \leq i \leq m$ und $R_i \subset T_i \in \Sigma^*$

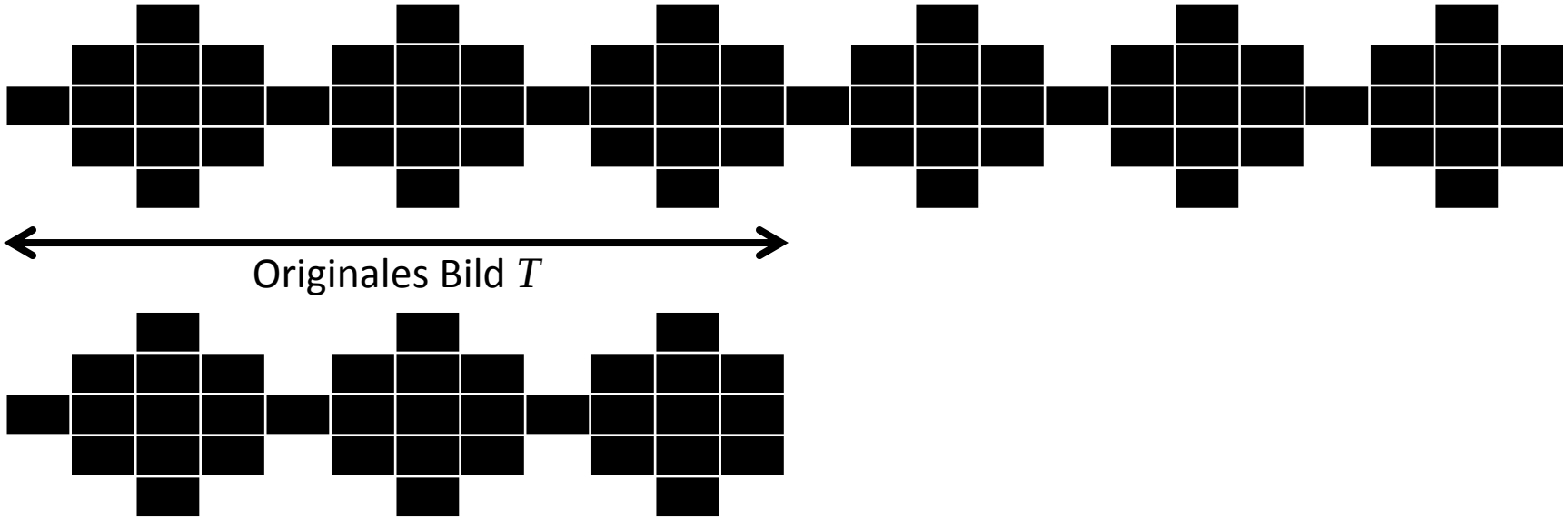
Breite des Musters ermitteln




```
1)   For R_i, R_i' in T, T + T
2)        W_i    = Calc Second match of R_i in R_i'
3)        P_width = lcm(W_i,P_width)
```

Nur Erkennung von ganzzahligen Wiederholungen

Breite des Musters ermitteln




```
1)   For R_i, R_i' in T, T + T
2)        W_i    = Calc Second match of R_i in R_i'
3)        P_width = lcm(W_i,P_width)
```

Laufzeit des Verfahrens

```
1)  For R_i, R_i' in T, T + T
2)      W_i    = Calc Second match of R_i in R_i'
3)      P_width = lcm(W_i,P_width)
```

1) $T + T \in O(1)$
   #Durchläufe = N (Anzahl der Zeilen)

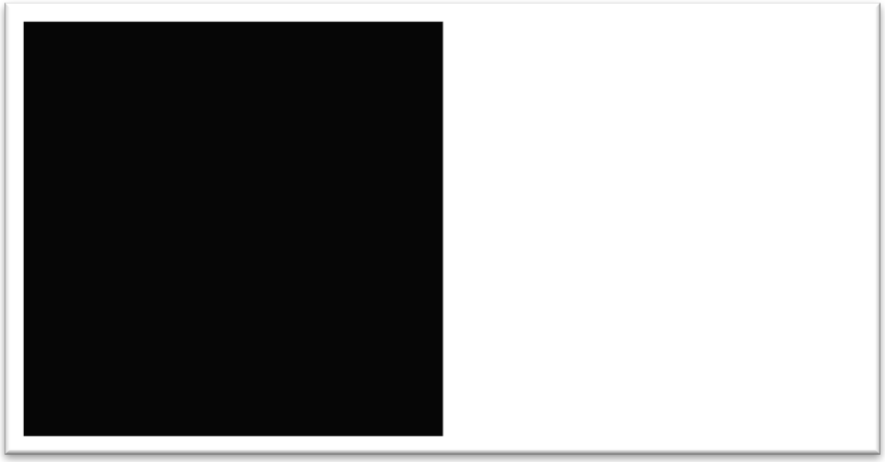
2) String Matching $\in O(|T| + |P|)$
   $|T| = 2*M$ (Anzahl der Spalten)
   $|P| = M$

   Insgesamt also O(m) Aufwand

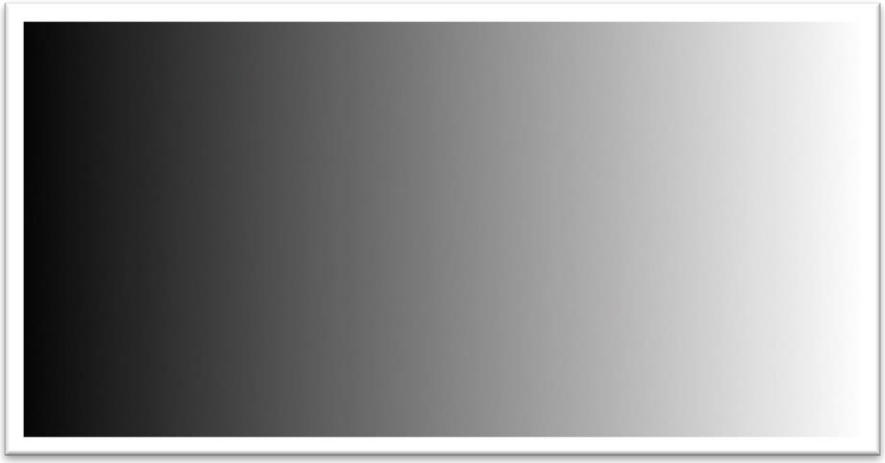
3) gcd $\in O(\log x)$ wobei x = Anzahl Bits für Parameter

$$\mathbf{O(n * (m + \log x)) \subseteq O(mn)}$$

Umgang mit Graustufen



```
Def eq(l, r)
      l=r
```



```
Def eq(l, r, δ)
      |l − r| < δ
```

Unscharfes Matching

Bisher: Lokale Ähnlichkeit, Jetzt: Globale Ähnlichkeit

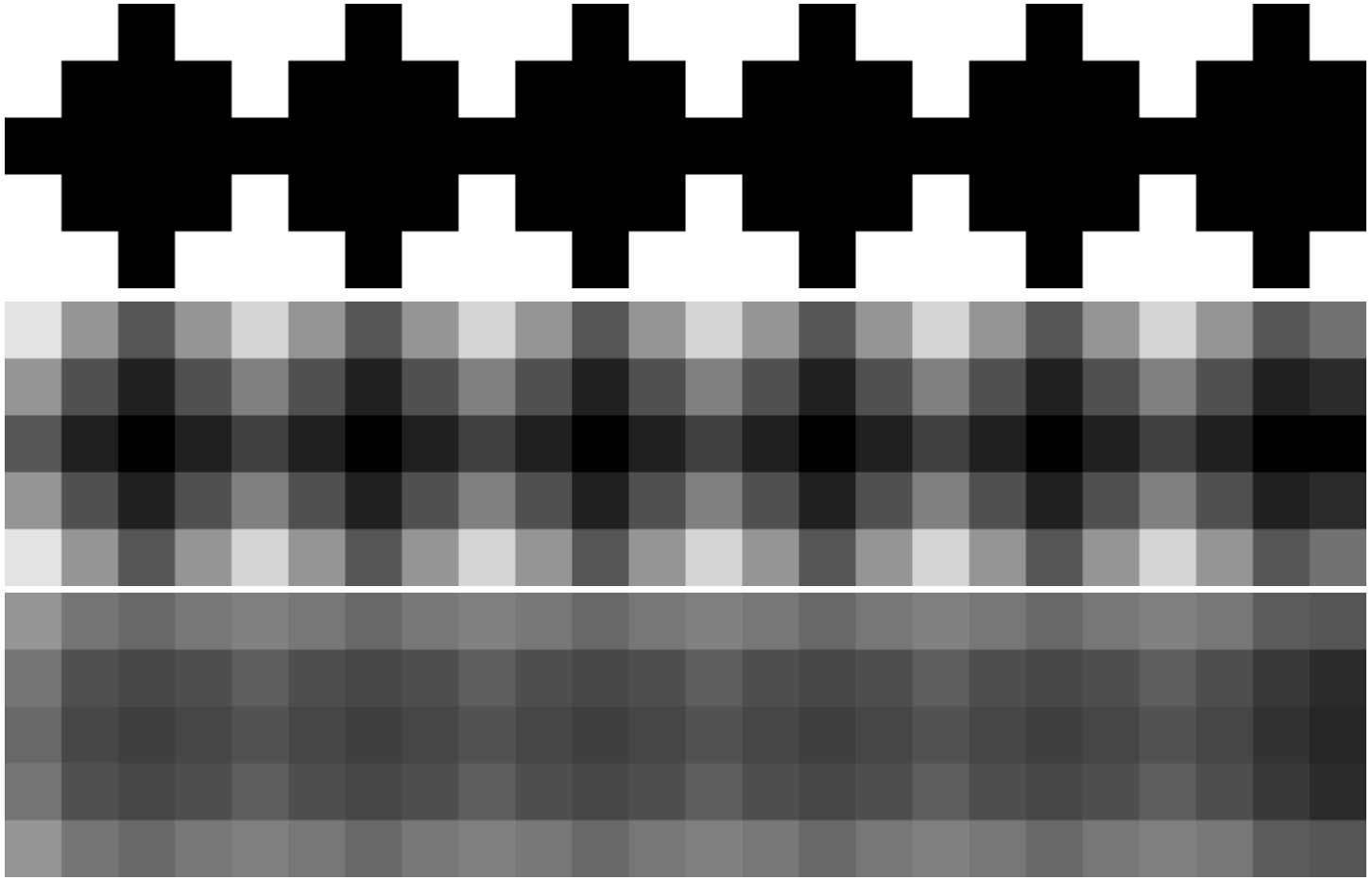
$$filter(j) = \frac{1}{m}\sum_{i=1}^{m}|R[i] - R'[i+j]|$$

```
1)  For R_i, R_i' in T, T + T
2)      For pos in R_i
3)          R_i[pos] = filter(pos)
4)      W_i   = Second match of R_i in R_i' with δ
5)      P_width = lcm(W_i, P_width)
```

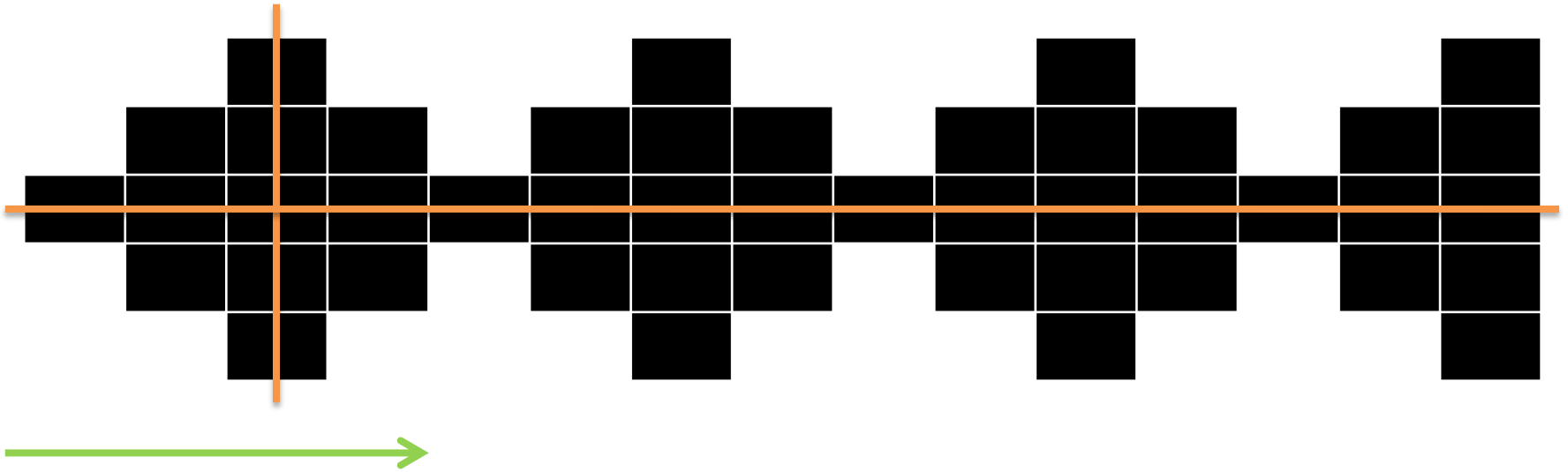
$O(n * (m^2 + m + \log x)) \subseteq O(nm^2)$

Unscharfes Matching

Bisher: Lokale Ähnlichkeit, Jetzt: Globale Ähnlichkeit



Zusammenfassung: Frieze Pattern Matching

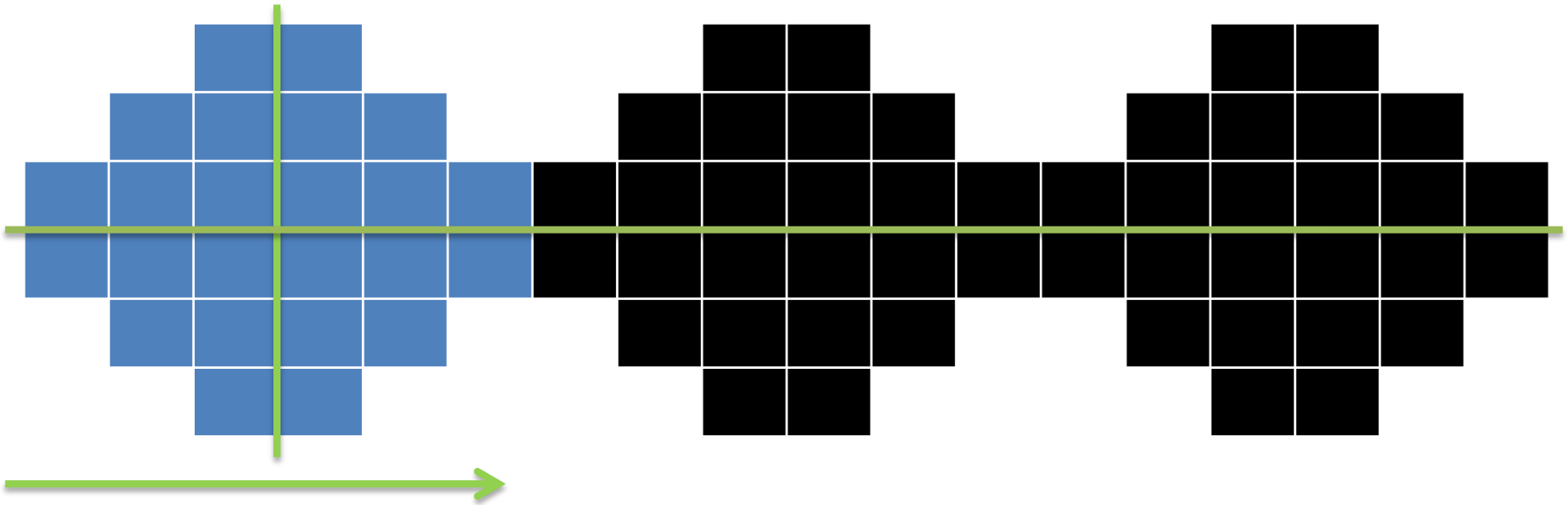


Rotation? **Nein**

Spiegelung? **Nein**

Translation? **Ja**

Zusammenfassung: Frieze Pattern Matching



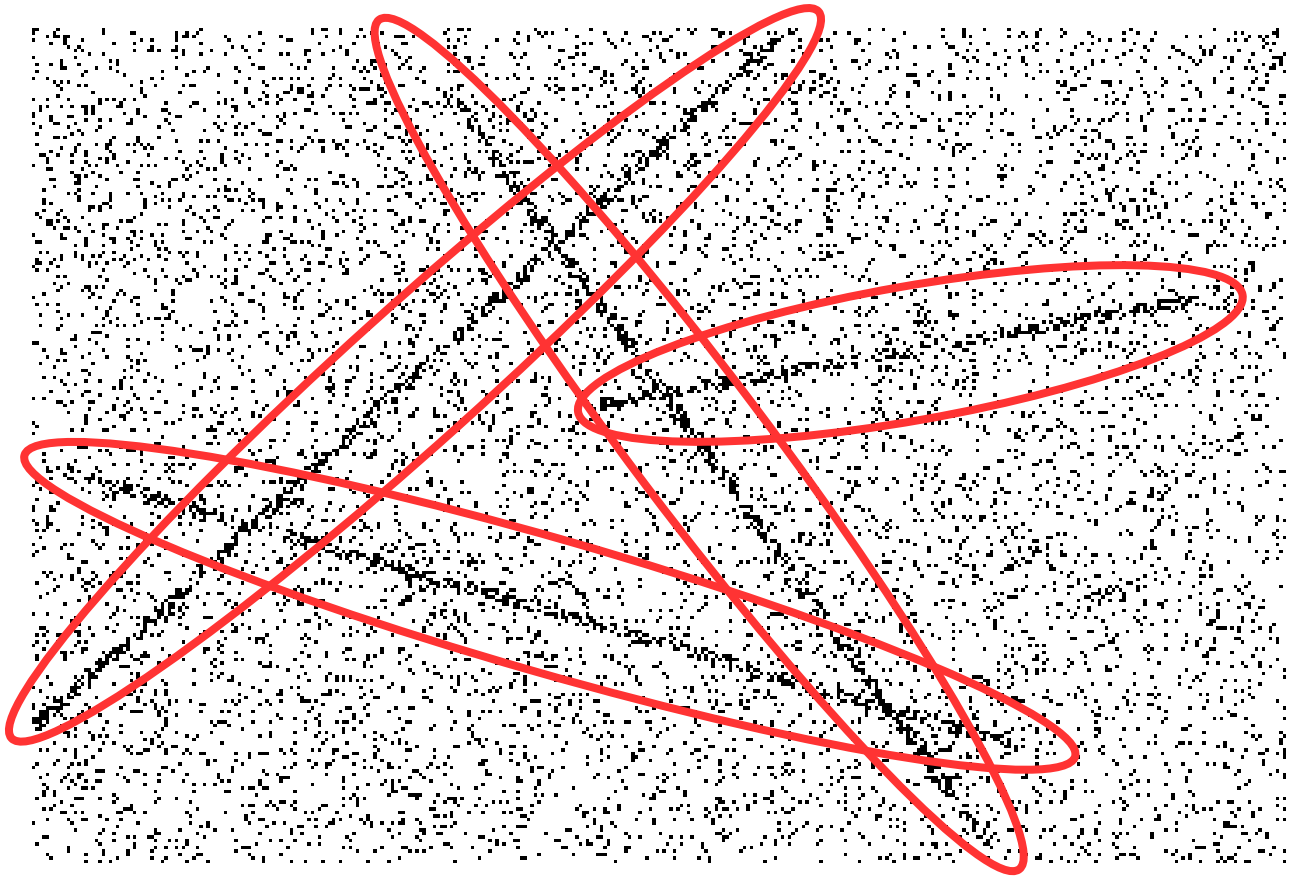
Rotation? Ja

Spiegelung? Ja

Translation? Ja

ERKENNUNG (2)

Hough Transformation



Hough Transformation

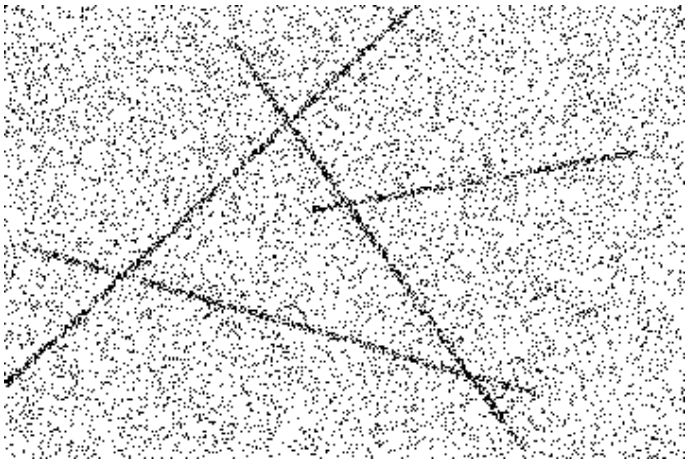
- Methode von Paul V. C. Hough
- US-Patent 3,069,654 (1962)
  „Method and means for recognizing complex patterns“

➔ allgemeiner Ansatz beliebige, parametrisierbare Formen zu finden

Linienerkennung

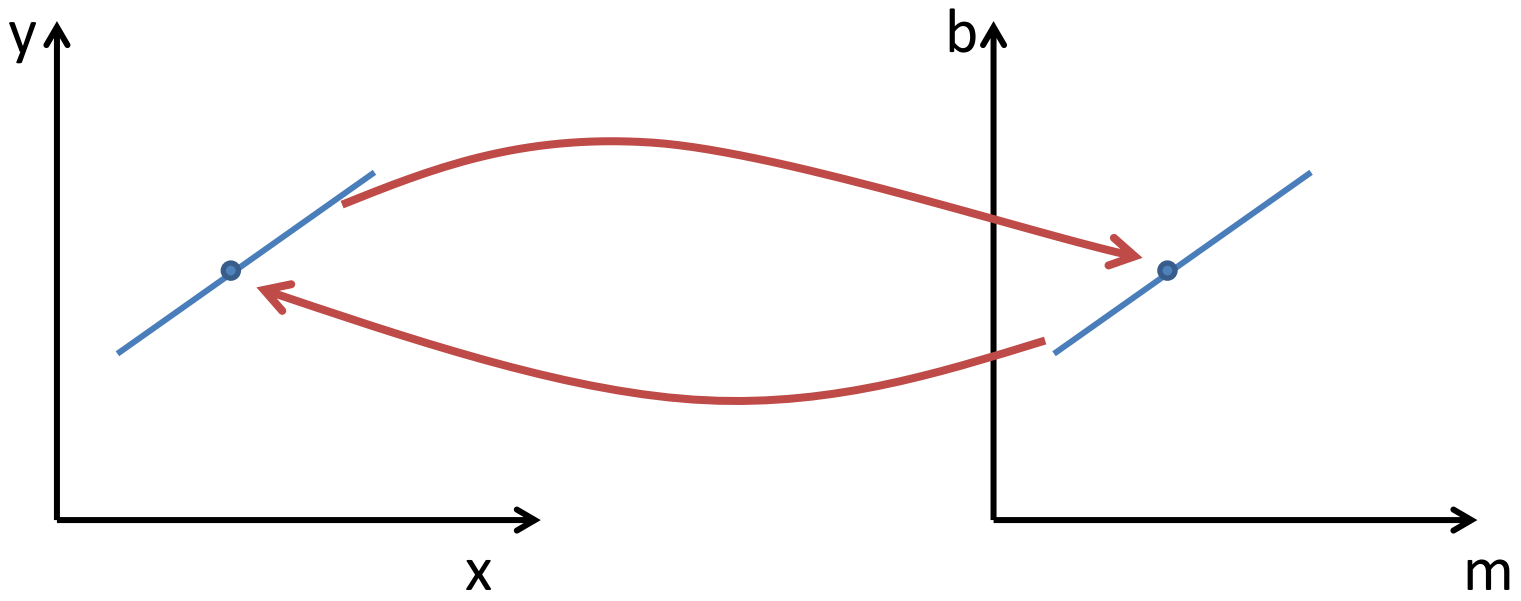
Hough-Transformation häufig hierfür eingesetzt

Binärbild → Linie, die möglichst viele Punkte überdeckt



Bildraum ⇔ Parameterraum

Bildraum (x , y)		Parameterraum (m , b)	
Punkt	(x_i , y_i)	$b = -m \cdot \mathrm{x_i} + \mathrm{y_i}$	Gerade
Gerade	$y = m_j \cdot x + b_j$	(n_j , b_j)	Punkt



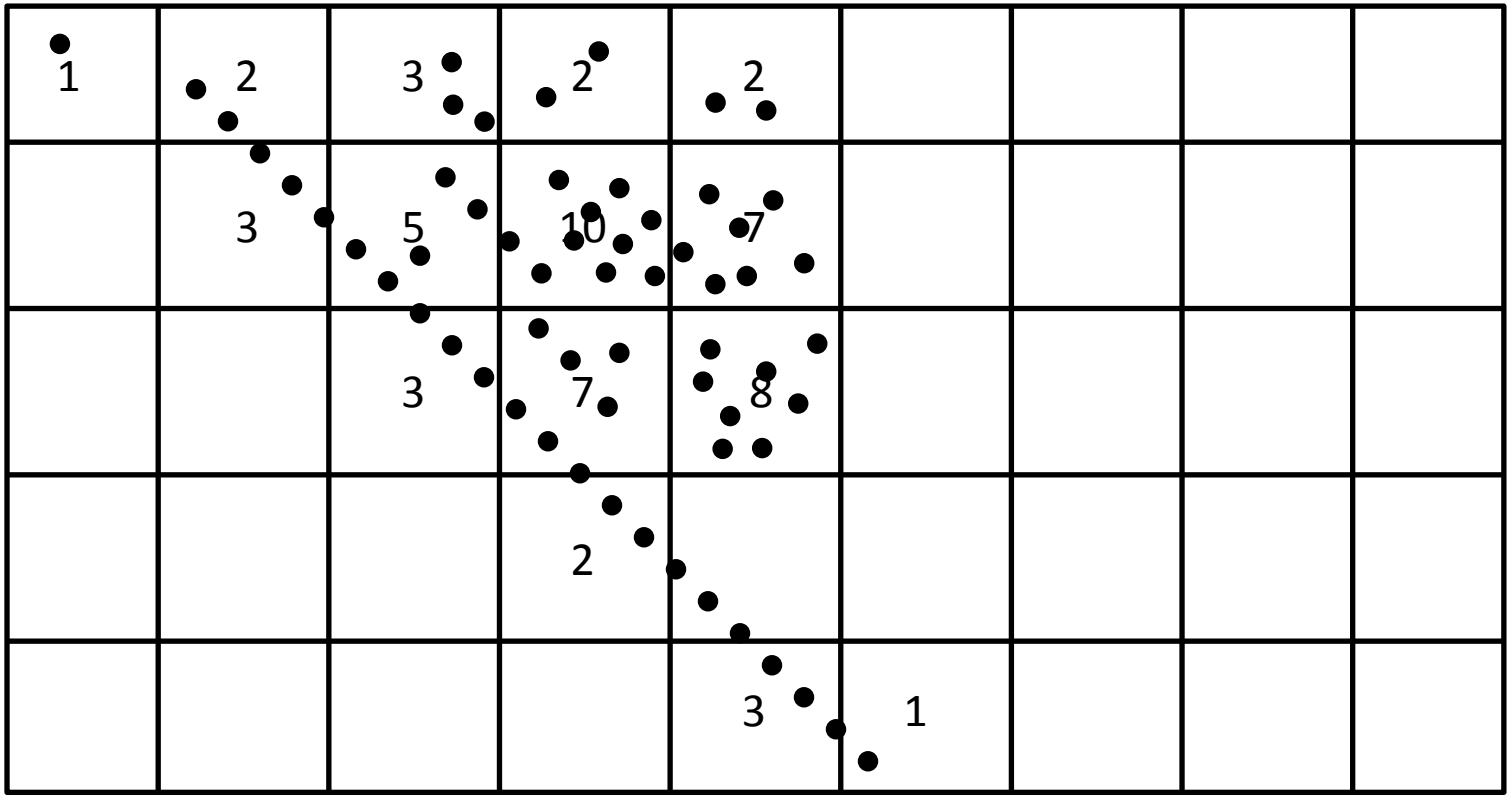
Parameterraum

- Problem: Beispiele für Parameter einer Linie nicht genau gleich

- daher: diskretisieren
- ➔ Datenstruktur: 2D-Array

		1					
1		1	3	1			
		4	6	2			1
		1	1	1			
					1		
2							

Diskretisierung

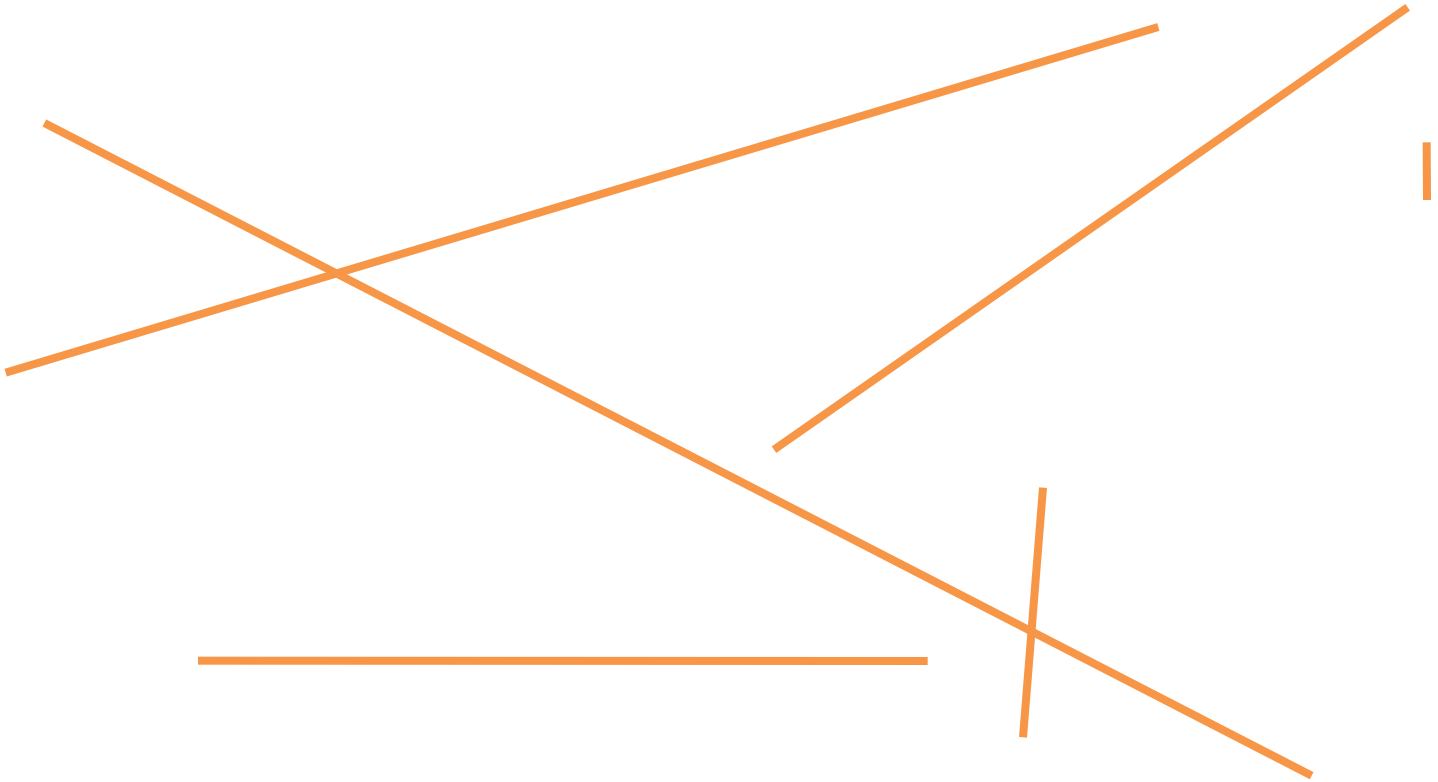


Geradenrepräsentation

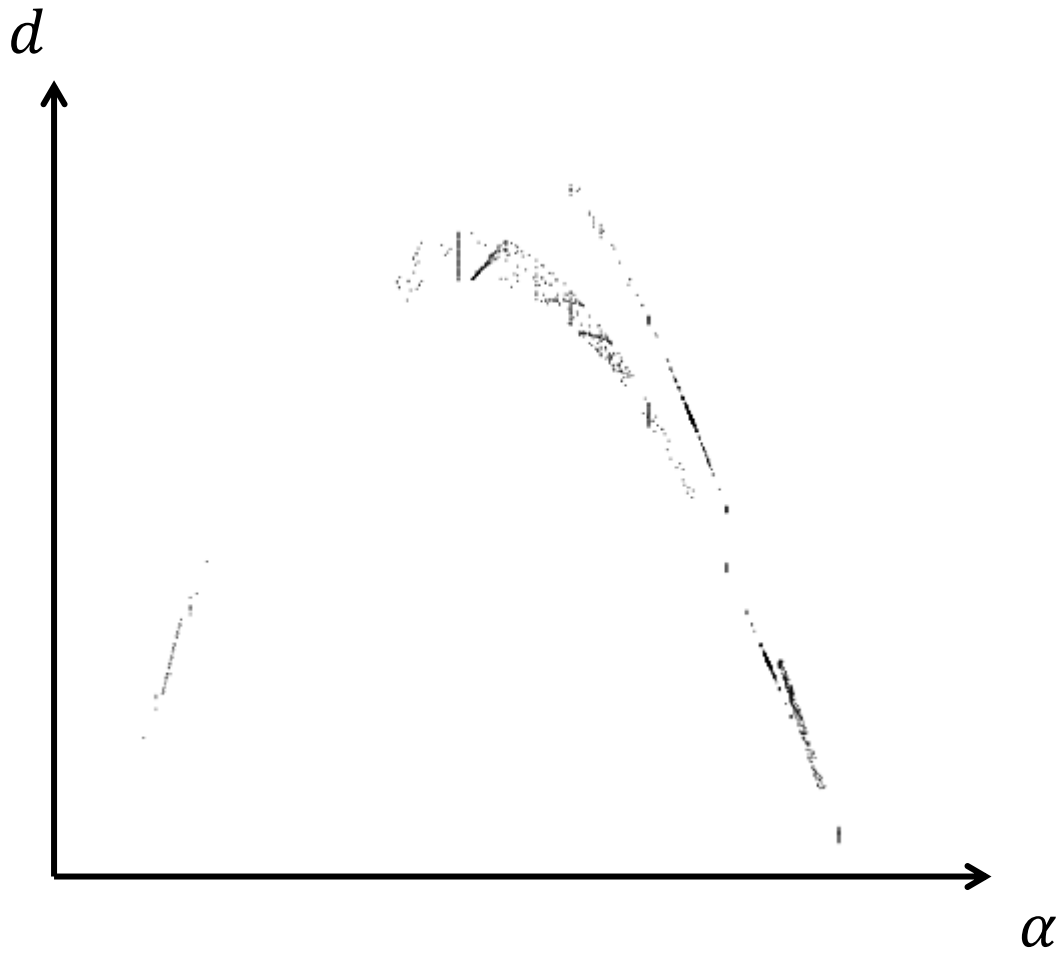
- bisher: $y = m \cdot x + b$
- ➔ vertikale Geraden?

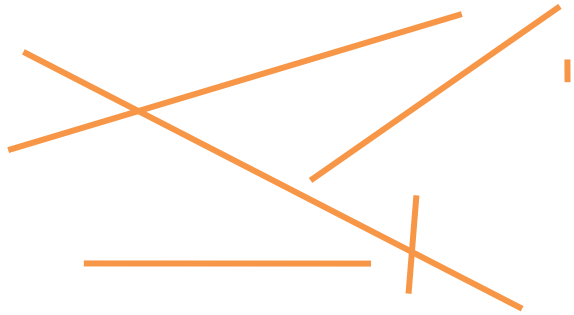
- besser: Hesse'sche Normalform
- ➔ Winkel, Distanz vom Ursprung (Polarkoordinaten)

Beispiel



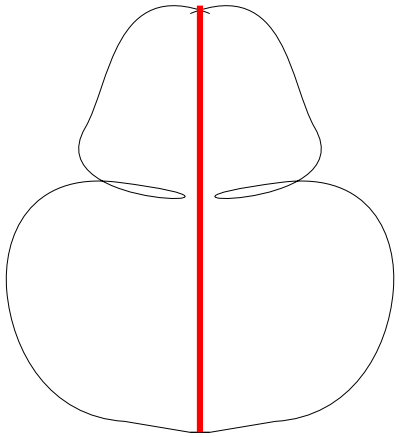
Resultat



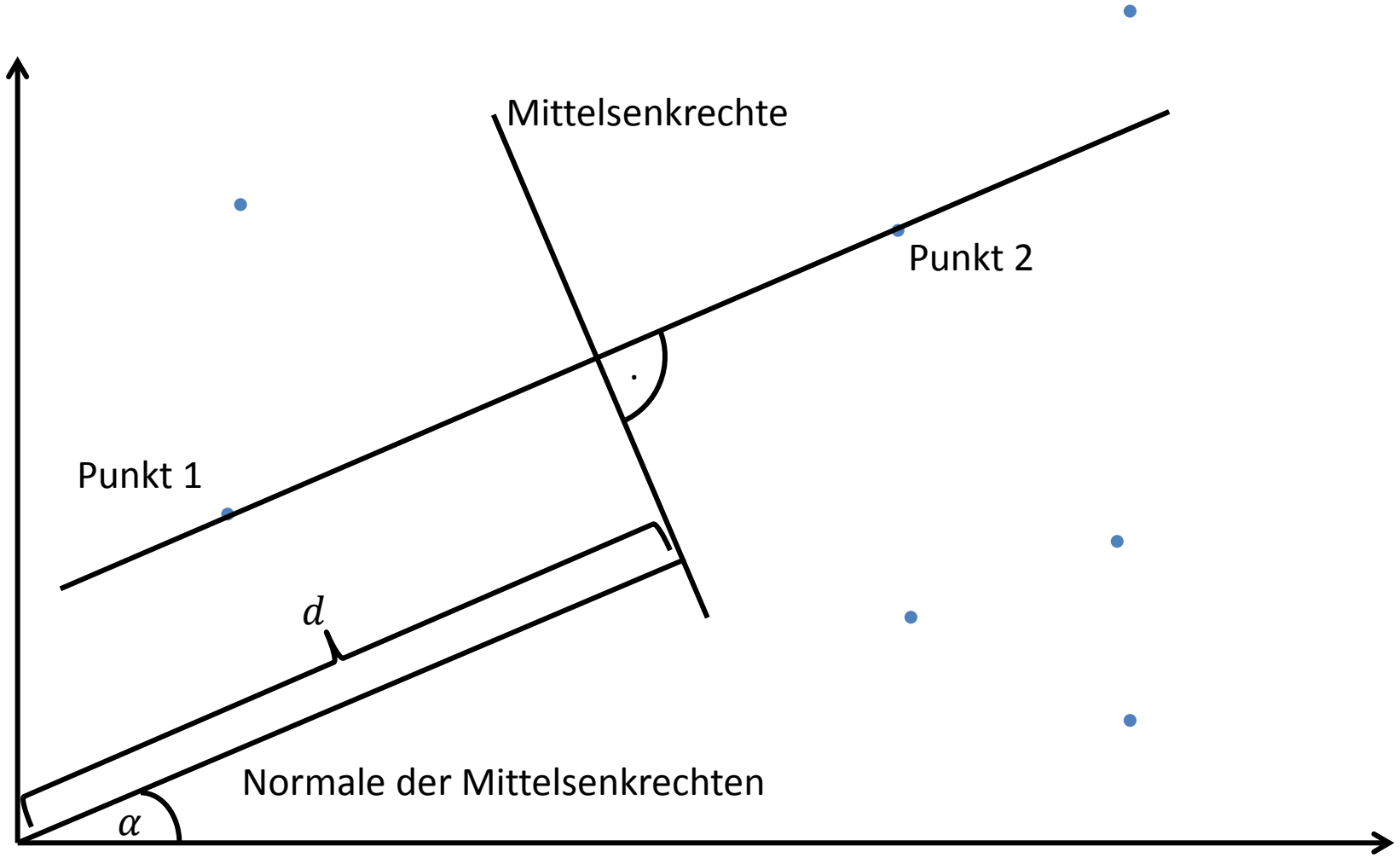


Linien → Symmetrie

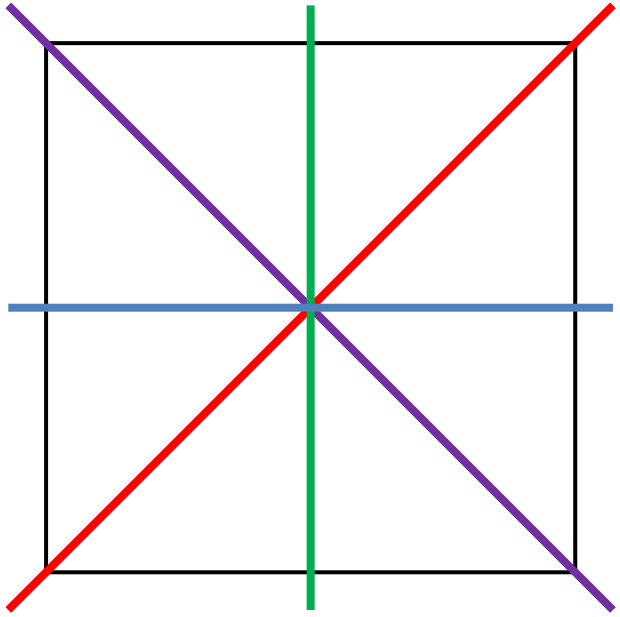
- Diedergruppe beschreibt Symmetrieachsen
- Symmetrieachse ist eine Gerade
- Suche nach einer „anderen“ Geraden

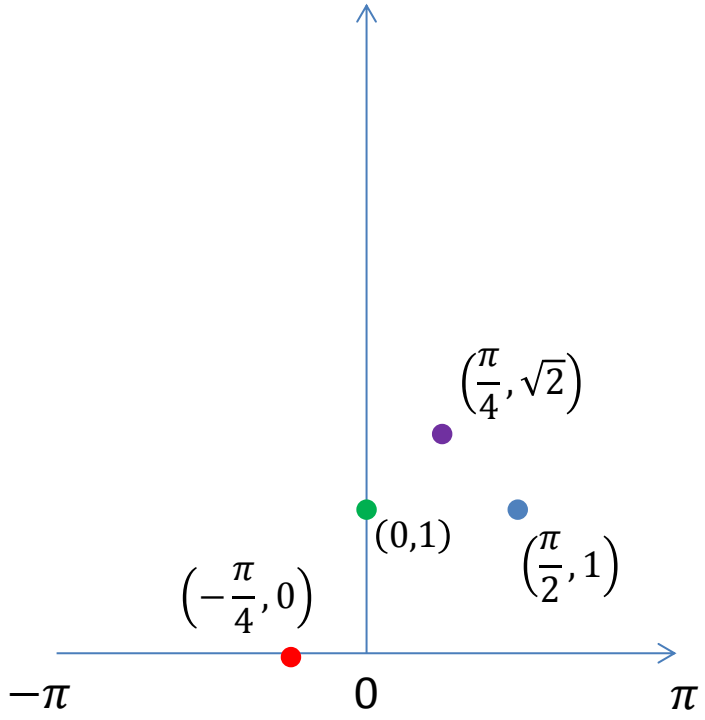


Berechnung


Mittelsenkrechte
Punkt 2
Punkt 1
d
Normale der Mittelsenkrechten
α

Beispiel




$\left(\frac{\pi}{4}, \sqrt{2}\right)$
$(0,1)$
$\left(\frac{\pi}{2}, 1\right)$
$\left(-\frac{\pi}{4}, 0\right)$
$-\pi$
0
π

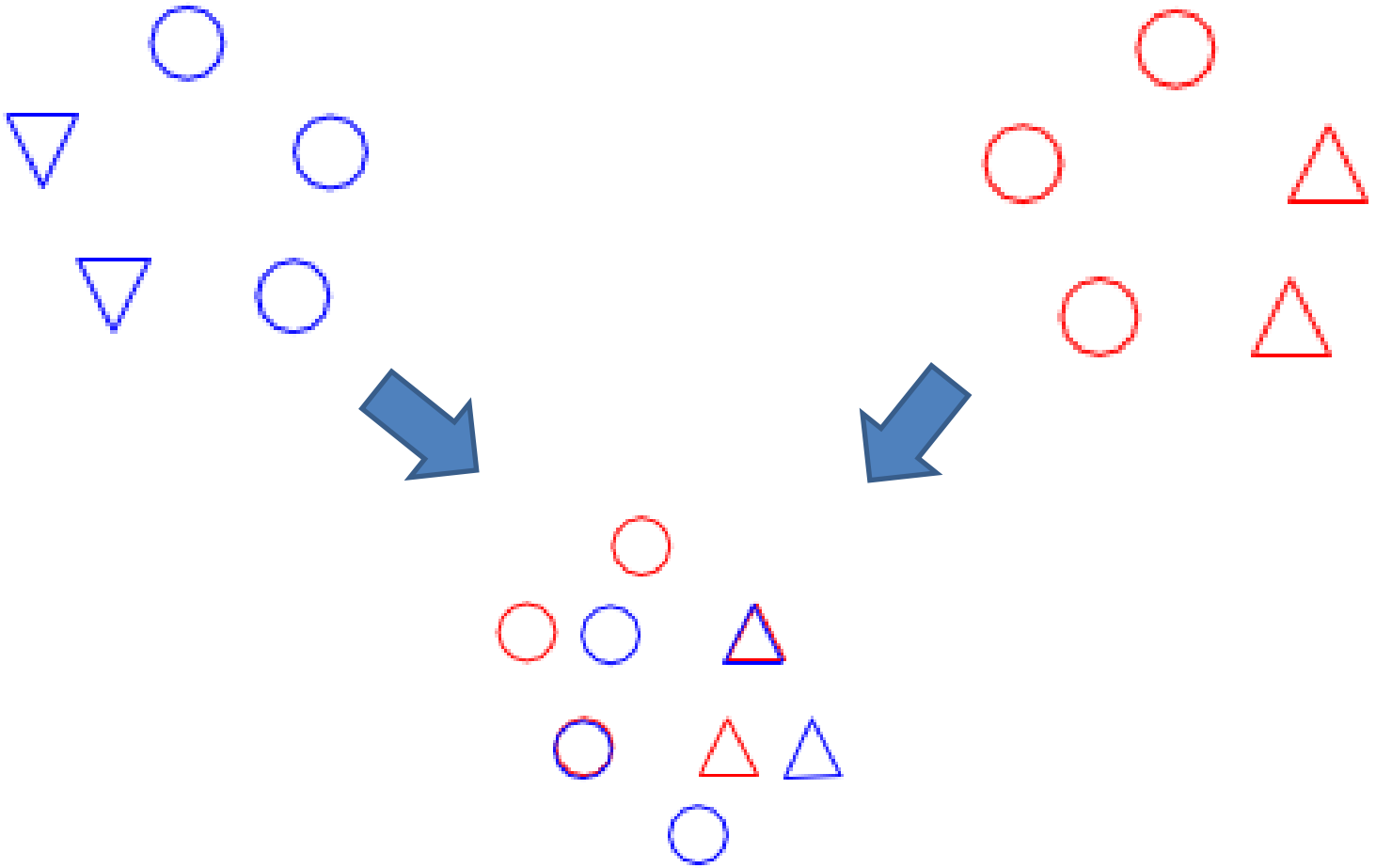
Algorithmus

$n \triangleq \#Punkte$

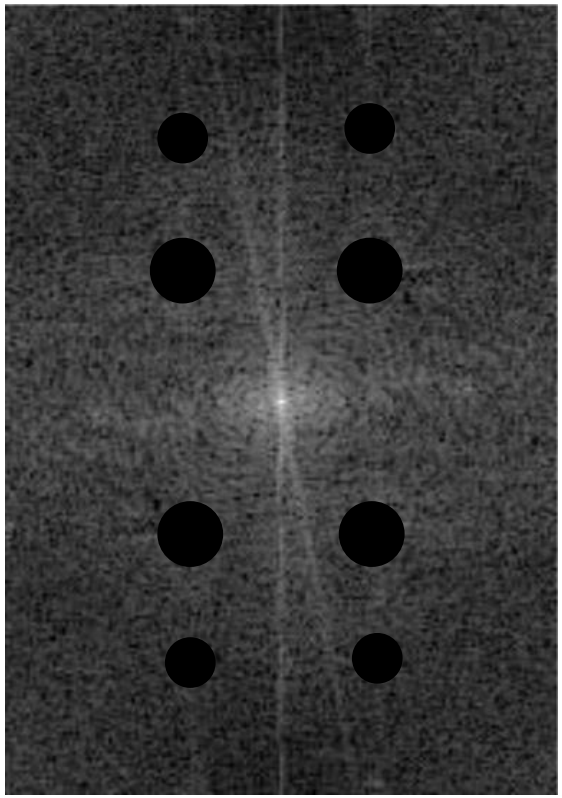
```
1)  For (p1,p2) in Punkte X Punkte                    O(n²)
2)      Calc Parameter für Parameterraum              O(1)
       (hier: Polarkoordinaten der Mittelsenkrechten)
3)      Inc Akkumulationsraum[diskretisierte Parameter]  O(1)
4)  Calc Häufungen im Akkumulationsraum suchen        O(n²)
```

➔ Laufzeit: $O(n^2)$

Kongruenz



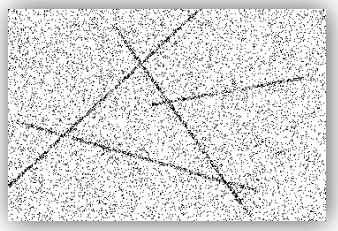
Symmetrieerkennung und die Fourier Transformation

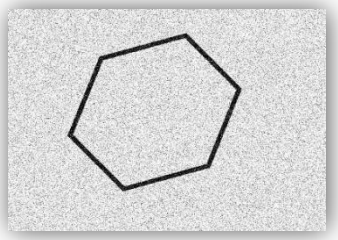


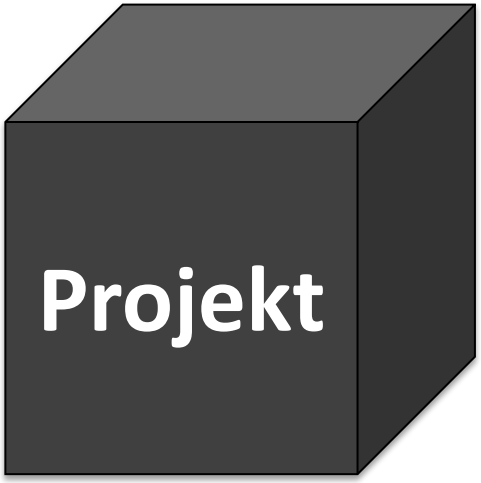

MASTER-PROJEKT

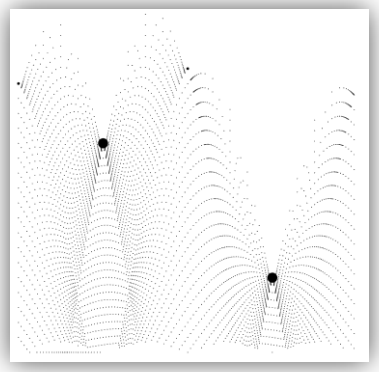
Master Projekt

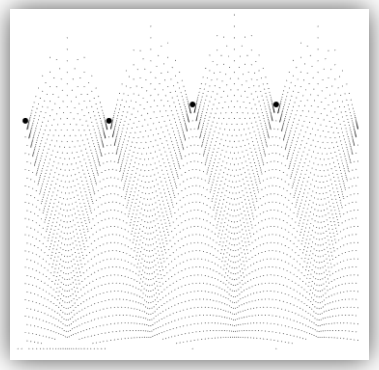
Graustufenbild











Symmetriegruppe

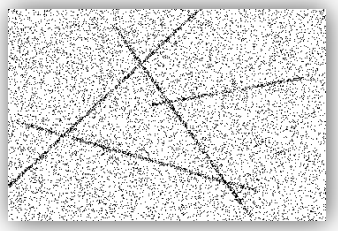
Keine Symmetrie

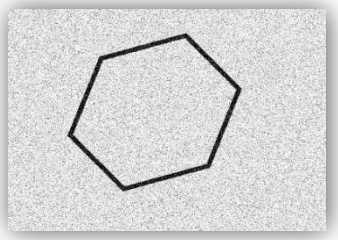
Keine Symmetrie

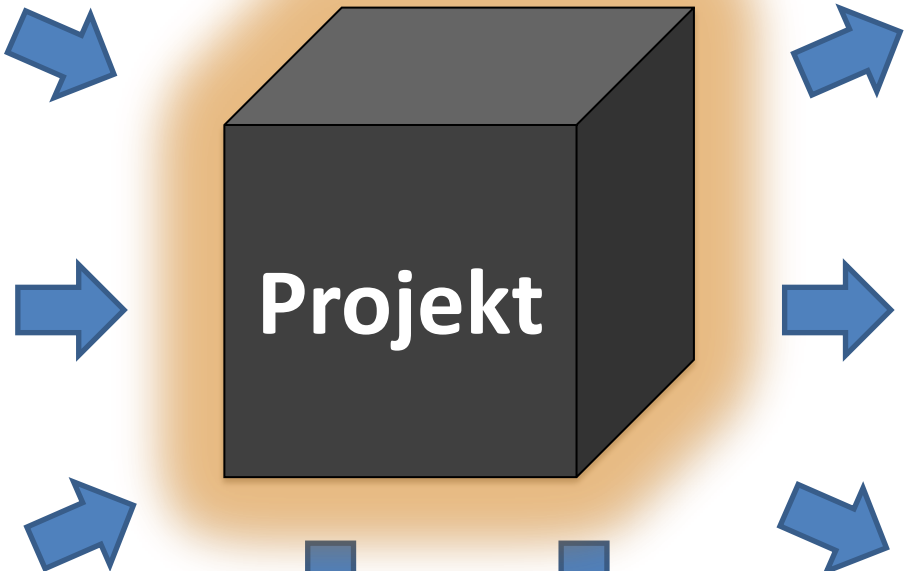
D_6 & C_6

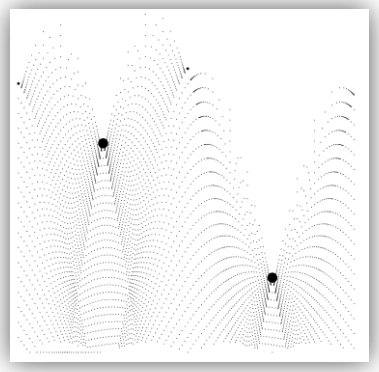
Master Projekt

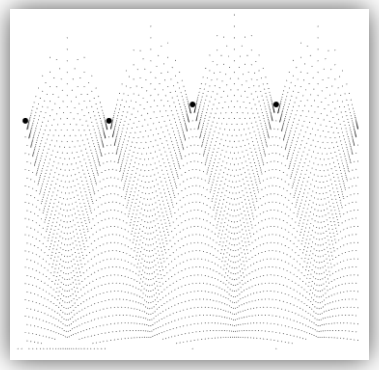
Graustufenbild











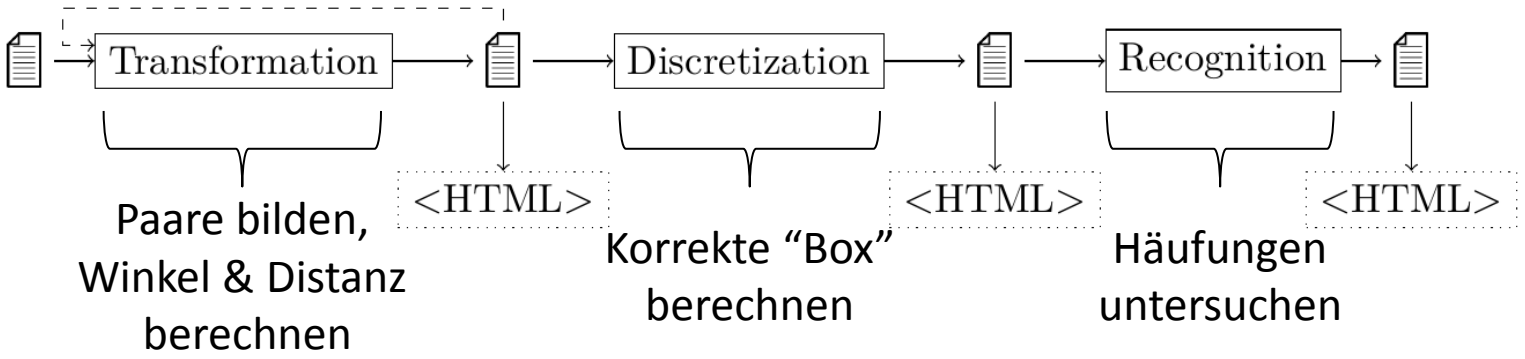
Symmetriegruppe

Keine Symmetrie

Keine Symmetrie

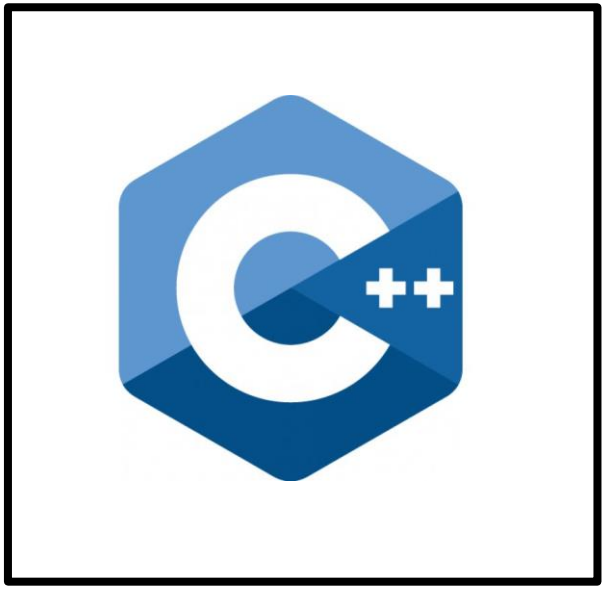
D_6 & C_6

Die Pipeline




```
1) For p1, p2 in Punkte X Punkte
2)       g_param = mx + b - y aus (p1, p2) konstruieren
3)       g_polar = polarcoord(g_param)
4)       Parameterraum += g_polar
5) For p1', p2' in Parameterraum
6)       DiskreterRaum += interpolated(p1', p2')
7) Suchen von Häufungen(DiskreterRaum)
```

Software Architektur



Backend

Protocol Buffer

Frontend

Warum C++ und HTML?

Harte Vorgaben von der FU Berlin

- Kein Haskell ☹
- C++ oder Java

Weichere Vorgaben von uns

- Viele Experimente mit der Oberfläche
  - Strikte Trennung zwischen Berechnung und Visualisierung
  - Qt & GTK (C++) bzw. Swing, AWT & JavaFX (Java) sind eher “klassische” UI Bibliotheken
- Platformübergreifend (sogar auf mobilen Geräten)

Warum Protocol Buffer?

	JSON	**Protobuf**
Typisierung • Zahlen bestimmter Größen • Strukturen, Listen • Automatische Validierung	**-** (keine Validierung)	**++**
Einfach (Javascript) • Externe Abhängigkeiten?	**++** (nativ unterstützt)	**-** (Bibliotheken, aber keine nativen Integer)
Einfach (C++) • Externe Abhängigkeiten?	**-** (Bibliotheken, eigene Datenstrukturen)	**+** (Offizielle Code Generatoren)
Größe der Nachrichten	**--**	**++**

Warum Protocol Buffer?

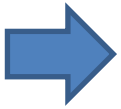
```
message ExactSample {
  required double angle = 1;
  required double distance = 2;
  required Intensity i1 = 3;
  required Intensity i2 = 4;
}
```

```
message Intensity {
   repeated int32 channels = 1;
}
```

Datentyp	Protocol Buffer	JSON
double	8 Byte	16 ASCII Zeichen
int32 (0 – 255)	1 – 2 Byte	1 – 3 ASCII Zeichen

Protocol Buffer $^1/_5$ bis $^1/_2$ der Größe

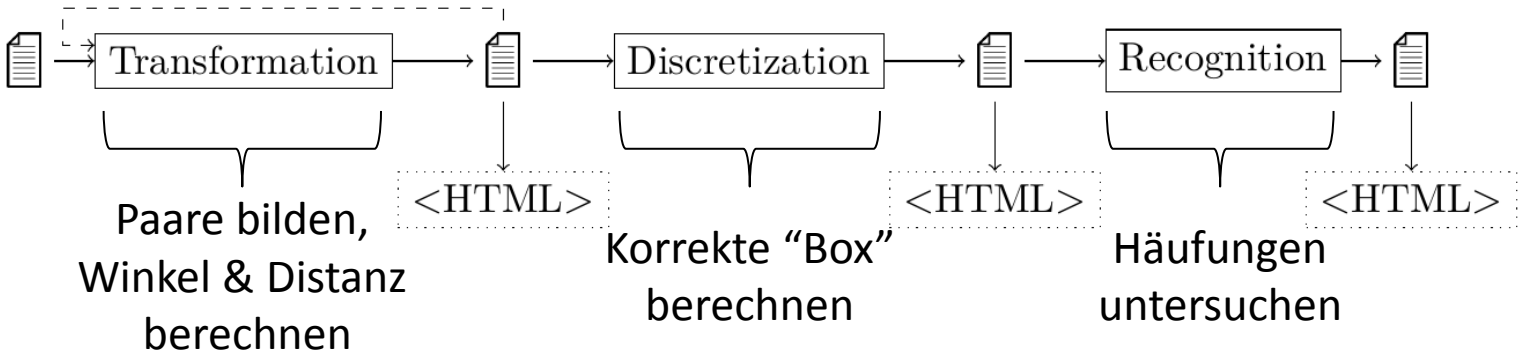
+ 33% für Base64 Encoding



$$x * \frac{1}{5} * \frac{13}{10} \sim 0.26x$$

$$x * \frac{1}{2} * \frac{13}{10} \sim 0.65x$$

Parallelisierung mit der Pipeline



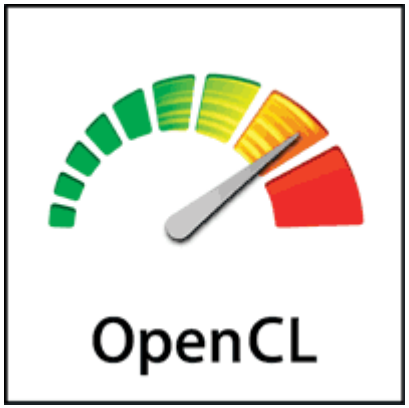
Gewinne durch die Pipeline eher marginal ☹

- 3 Stufen, also auch maximal 3 Threads
- **Stufen nicht gleichermaßen aufwändig**
- Overhead durch Synchronisierung bei naiver Implementierung recht groß
- Trotzdem softwaretechnisch eine gute Struktur

Parallelisierung mit Partitionierung

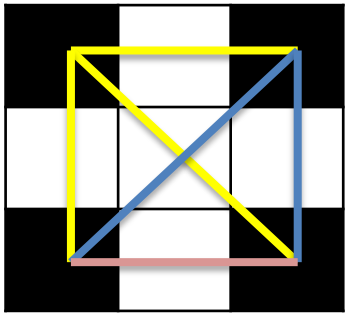
Annahmen

- Erste Pipelinestufe dauert am Längsten
- Statische Partitionierung ist unkritisch

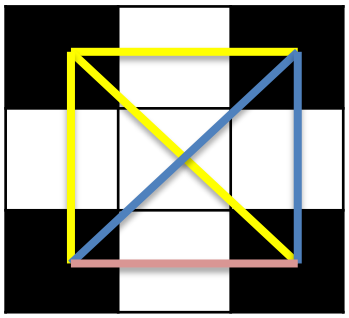



```
1) partitions = divide(Punkte X Punkte, #Cores)
2) For thread in Threads
3)   thread.compute
4)     For p1, p2 in partitions[thread.num]
5)       g_param = mx + b - y aus (p1, p2) konstruieren
6)       g_polar = polarcoord(g_param)
7)       thread.Parameterraum += g_polar
8)     Parameterraum += thread.Parameterraum
9) Threads.join()
```

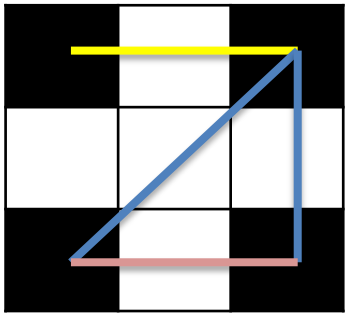
Auswahl von Paaren



Alle Paare
- 3 Gelb
- 2 Blau
- 1 Rot



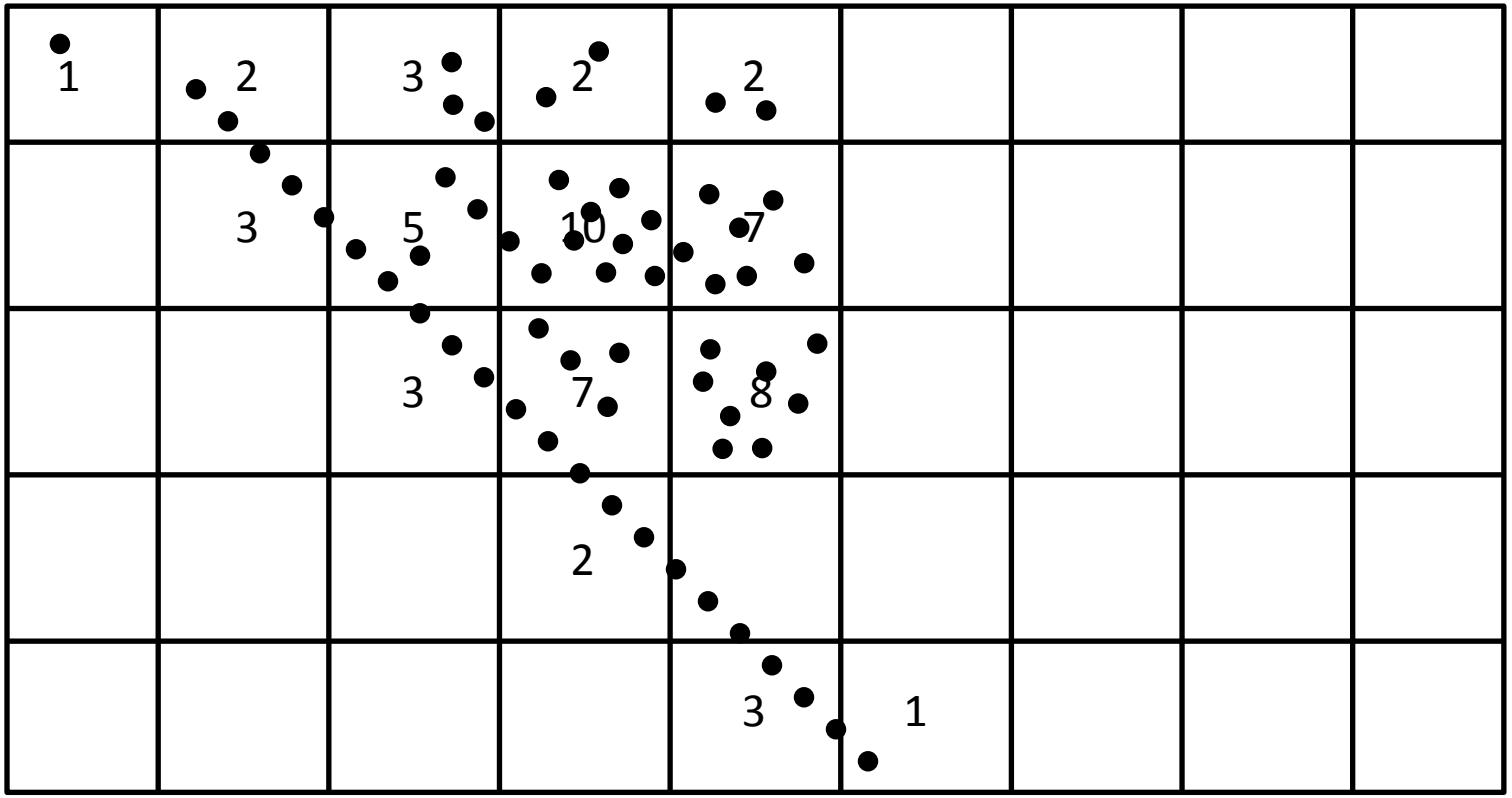
Jedes 2. Paar
- 2 Gelb
- 1 Blau
- 0 Rot



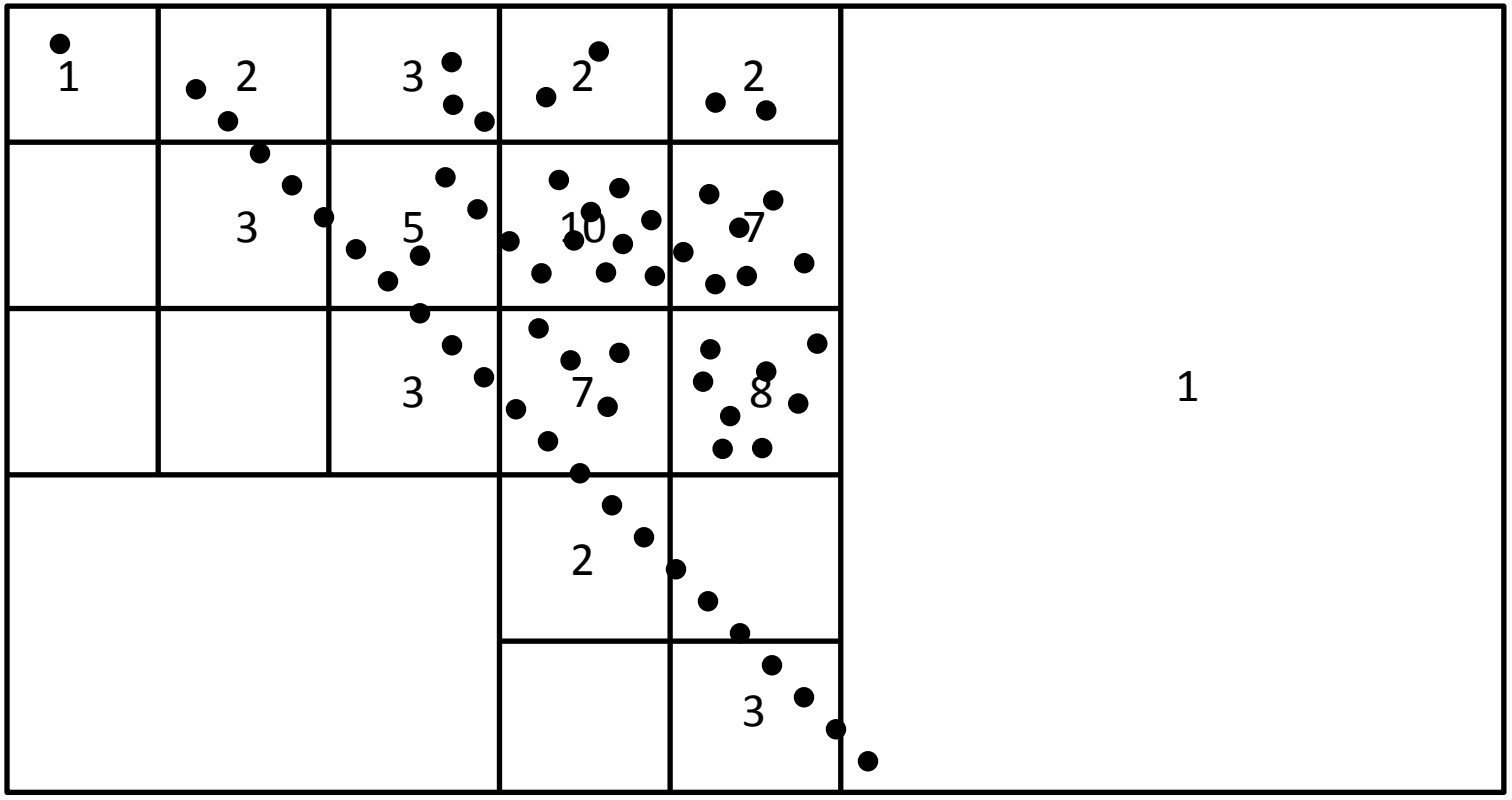
Zufällige Paare
- 1 Gelb
- 2 Blau
- 1 Rot

Leicht zu implementieren, schwierig zu untersuchen

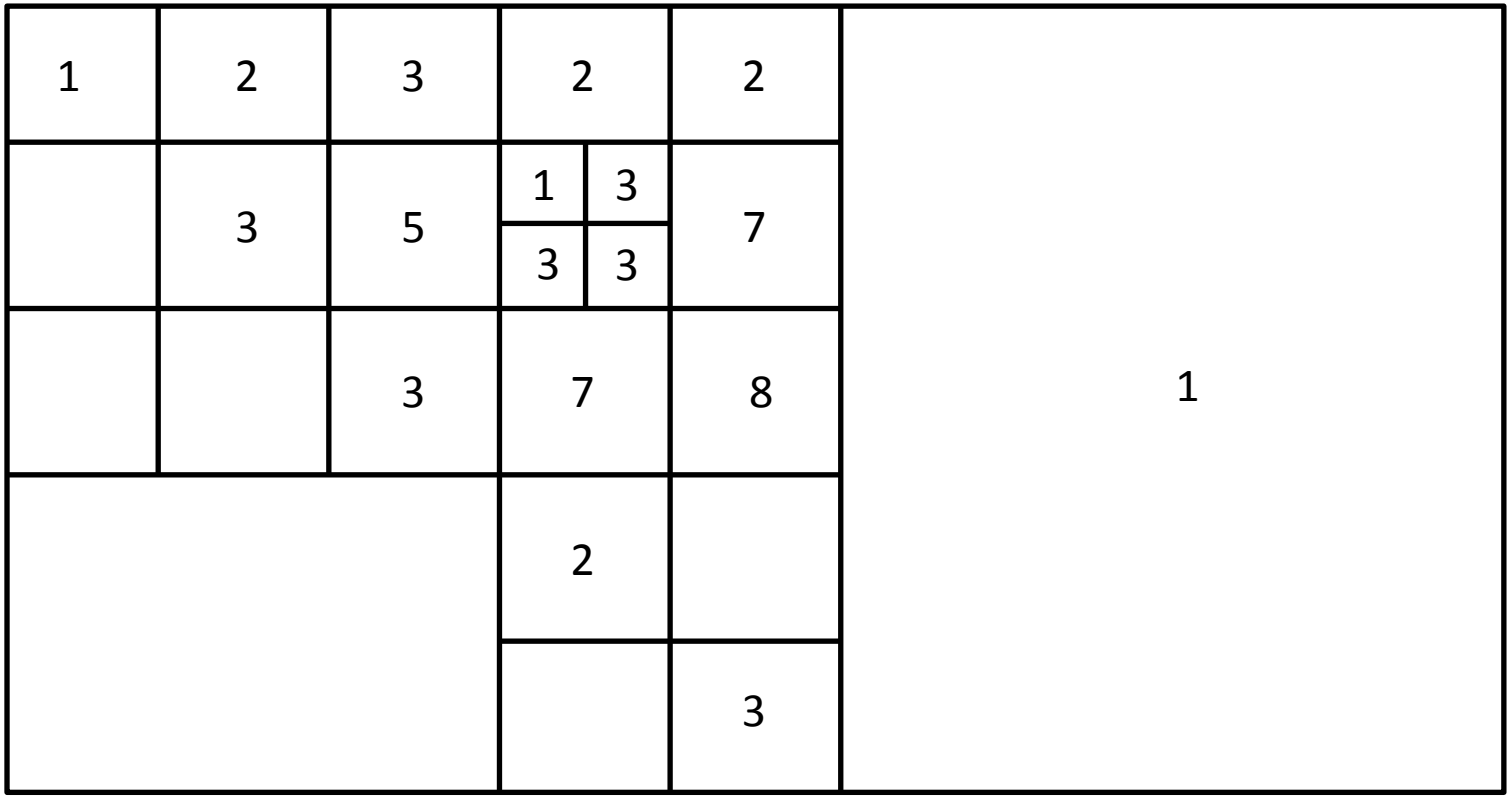
Diskretisieren der Werte



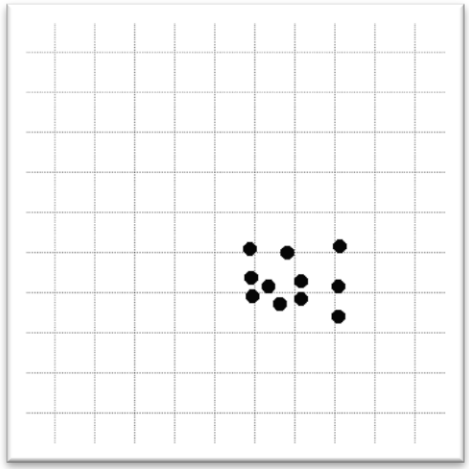
Diskretisieren der Werte



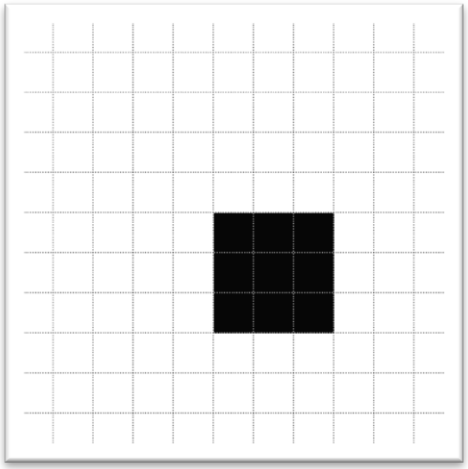
Diskretisieren der Werte



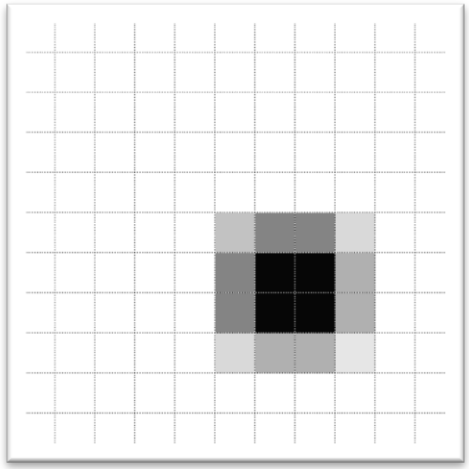
Diskretisieren der Werte



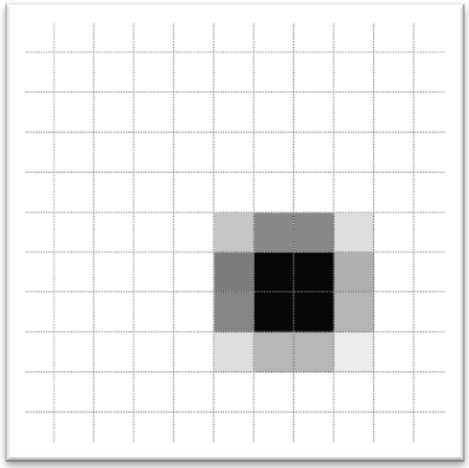
Nearest Neighbour



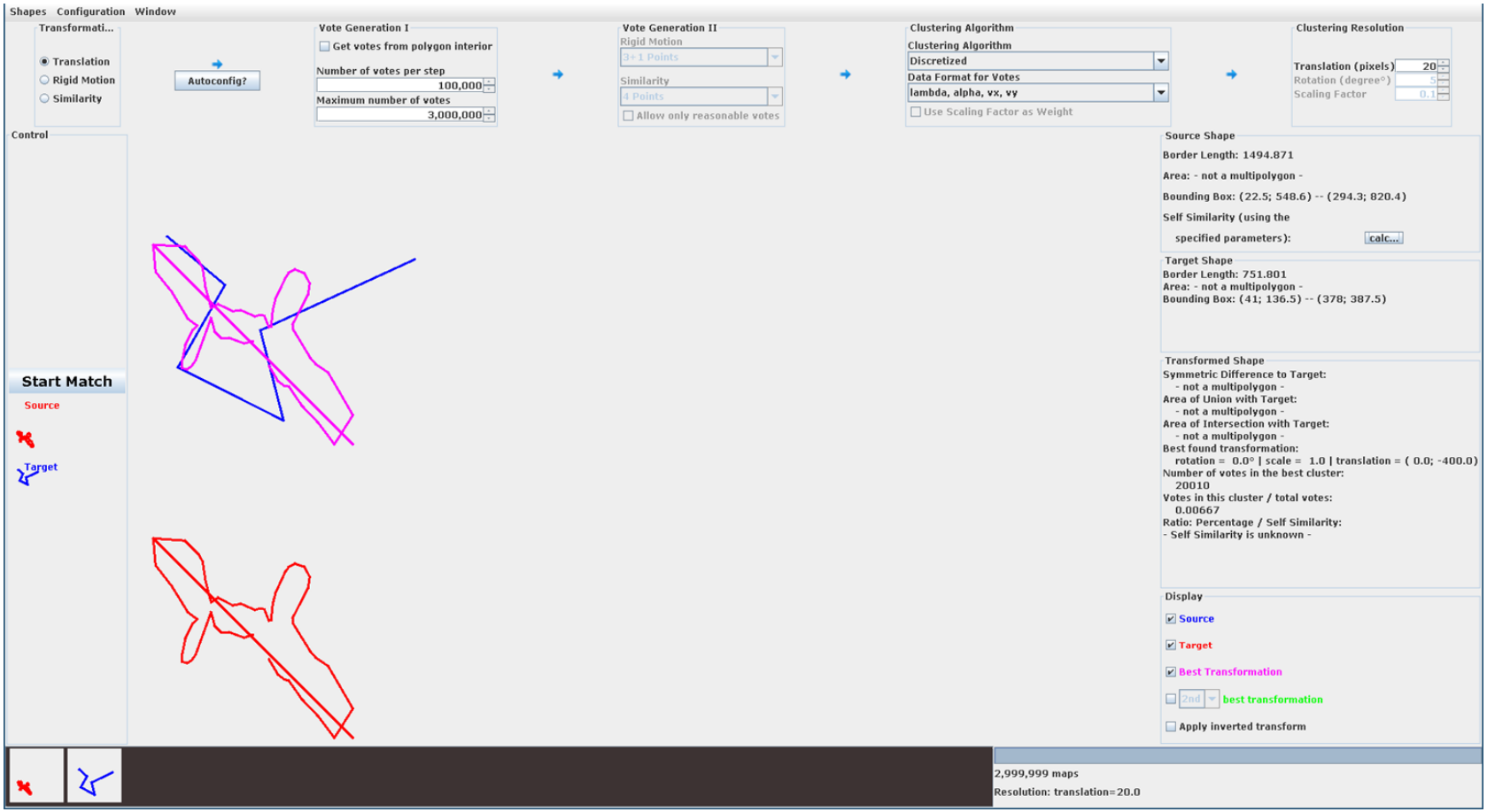
Bilinear



Bikubisch



Programme der FU



User-Interface

Responsive Design

SymmRecog Transformation Discretization Recognition

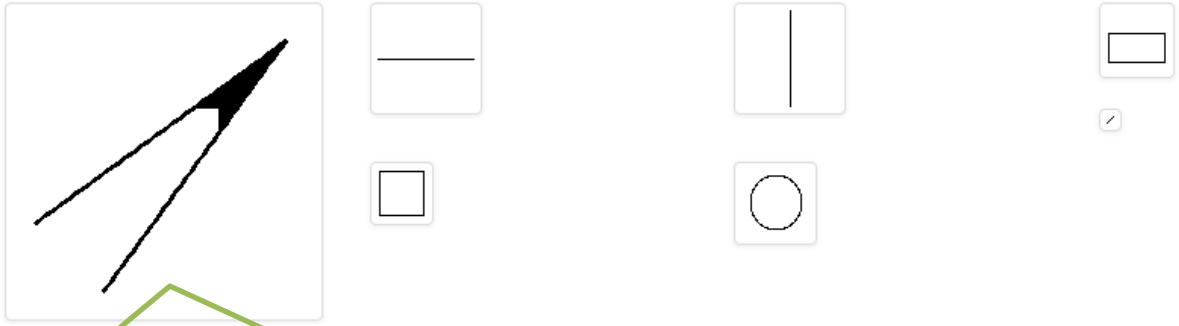
Discretization step size
Interpolation

No image selected! Please upload an in ery.

Pair selection method
All pairs

Dynamisch einblendbare Optionen

Gallery



Eine Aktion für Berechnung

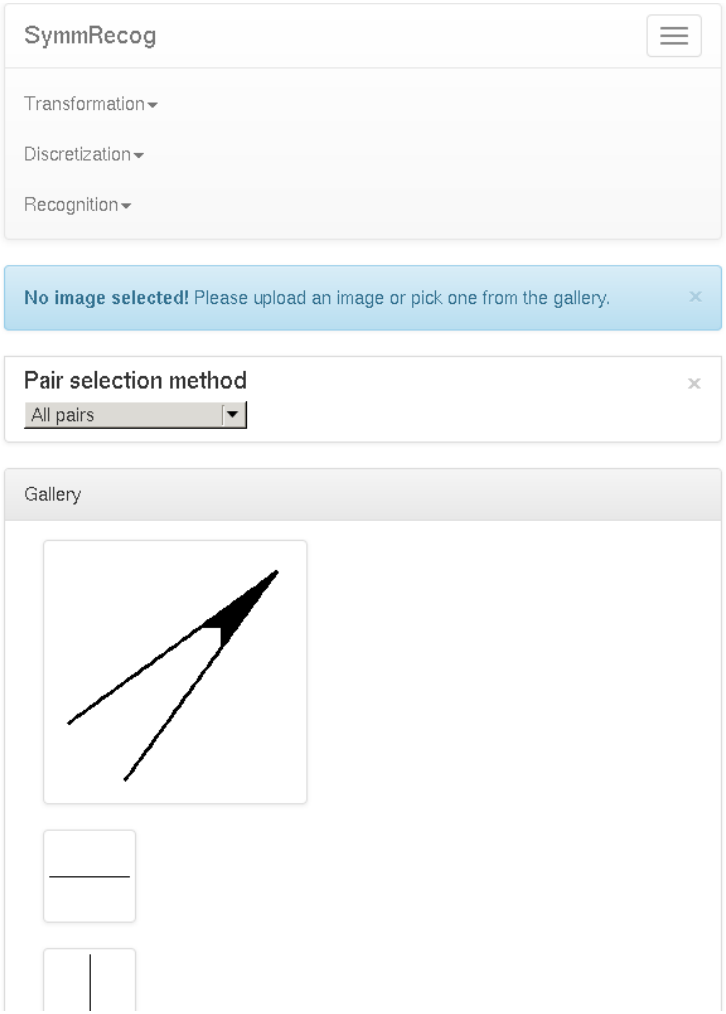
Original graphic	Transformation space
Please select an image from the gallery!	Please select an image from the gallery!

Platz für Inhalte

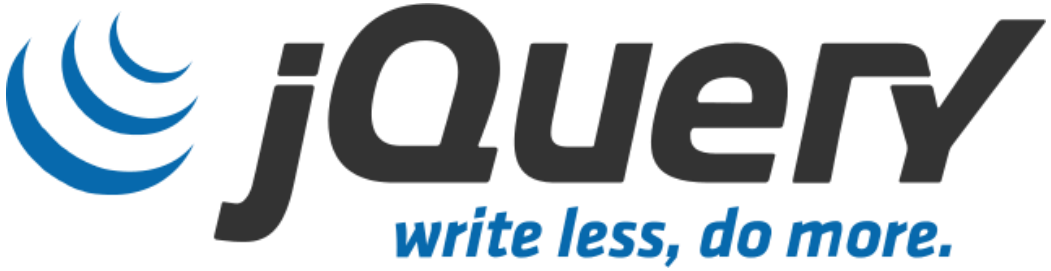
Log output

```
Image at images/big-mess.png (800x800) with 24 bits per pixel, maximum intensity is 1
```

User - Interface (mobil)

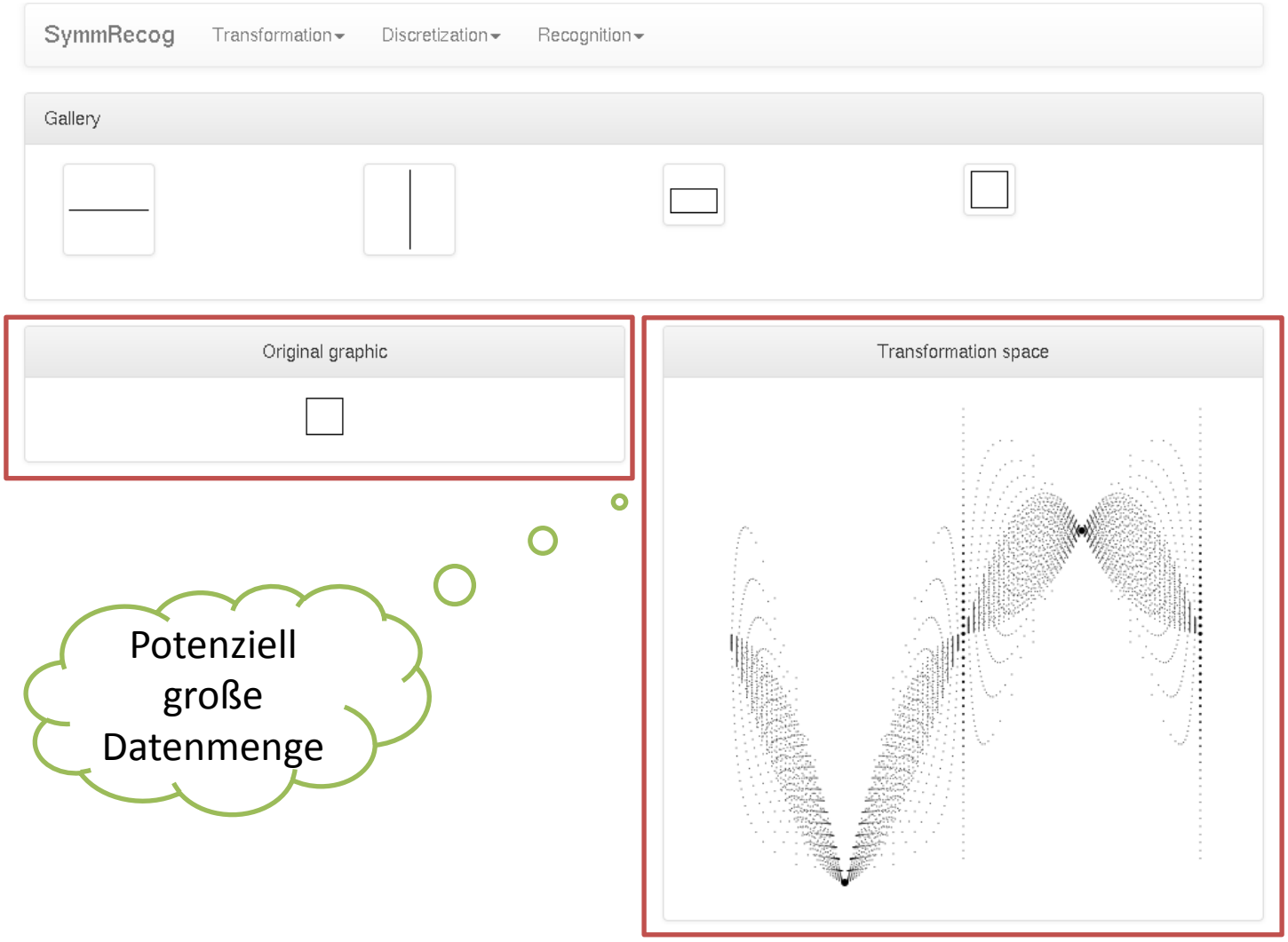


User - Interface



Javascript

Visualisierung

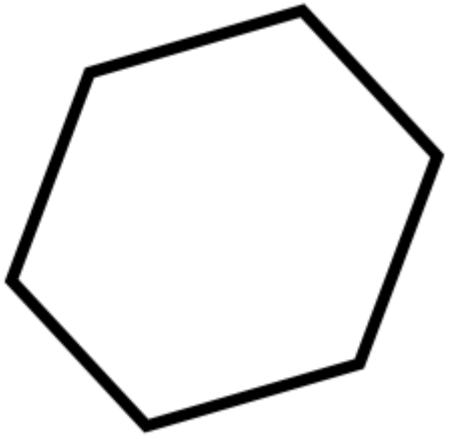


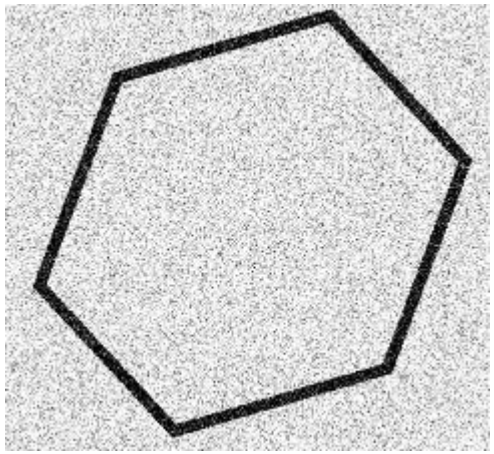
Canvas ⇔ SVG

	HTML5 Canvas	Scalable Vector Graphic
Zeichnen-Beschreibung	Canvas Primitive (Funktionen)	XML Beschreibung
Einbettung	HTML Element	SVG Namespace
Bildformat	Rastergrafik	Vektorgrafik
Interaktion	Browserevents des Canvas	Element-individuelle Events
Manipulation	Neuzeichnen	DOM-Manipulation
Performance	Gut	Langsamer als Canvas

AUSBLICK UND FAZIT

Ausblick: Störungen





Rauschen

Verformen

Ausblick: Zu untersuchende Größen

- Störungen
- Generierung der Punktepaare
  - Einfluss der verschiedenen Methoden der Paarbildung?
  - Anzahl nötiger Paare für gute Erkennung?
- Diskretisierung
  - Einfluss der variabel großen Zellen?
  - Einfluss der verschiedenen Interpolationsmethoden?

Ende