

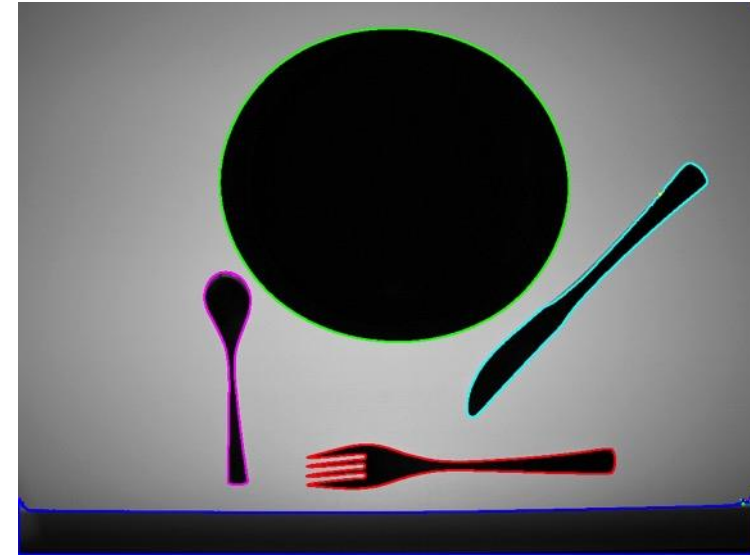
**Sensordatenverarbeitung**

# **SEGMENTIERUNGS- GETRIEBENE BV (4C)**

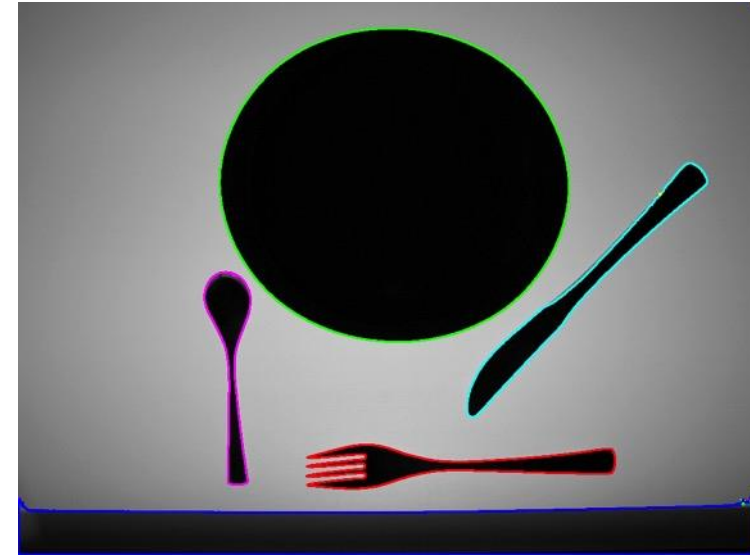
**(4-8..11.24)**



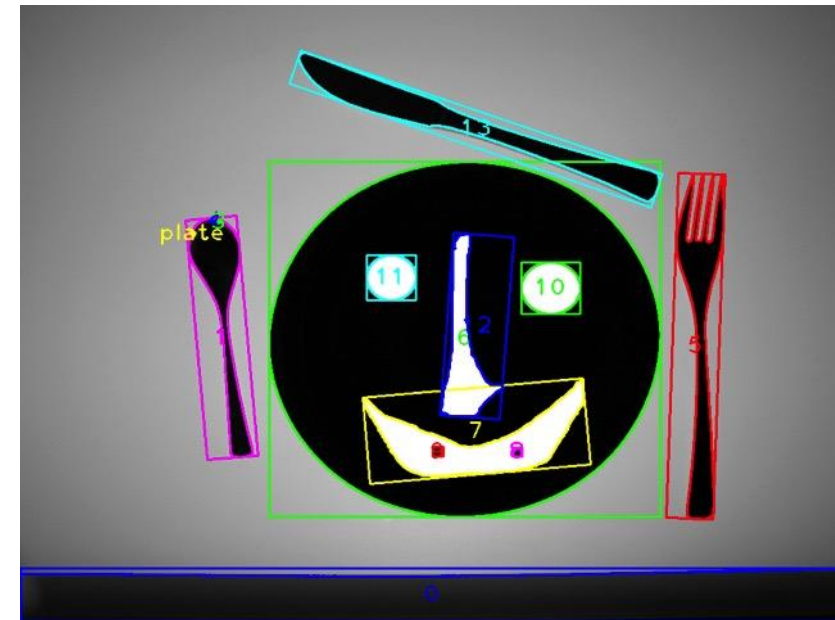
- Zu *einem* Objekt gehörende Pixel zusammenfassen
- Geht auch bei wenn Pixel einen Objekttyp angibt (Objektklassen)
- OpenCV: `findContours`
  - [https://docs.opencv.org/3.3.1/d3/d05/tutorial\\_py\\_table\\_of\\_contents\\_contours.html](https://docs.opencv.org/3.3.1/d3/d05/tutorial_py_table_of_contents_contours.html)
- Algorithmus: Satoshi Suzuki and others.  
Topological structural analysis of digitized binary images by border following. Computer Vision, Graphics, and Image Processing, 30(1):32–46, 1985.



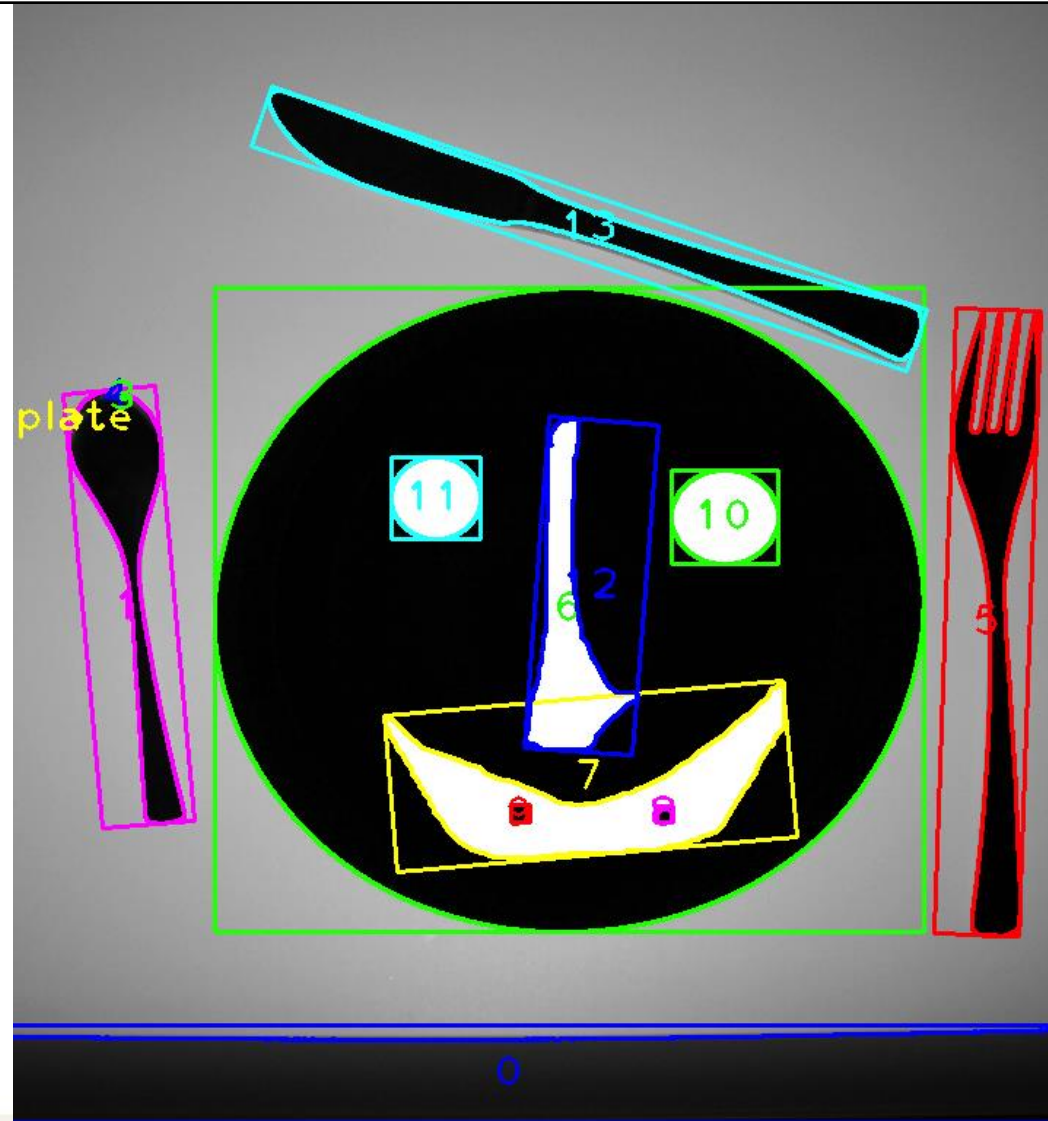
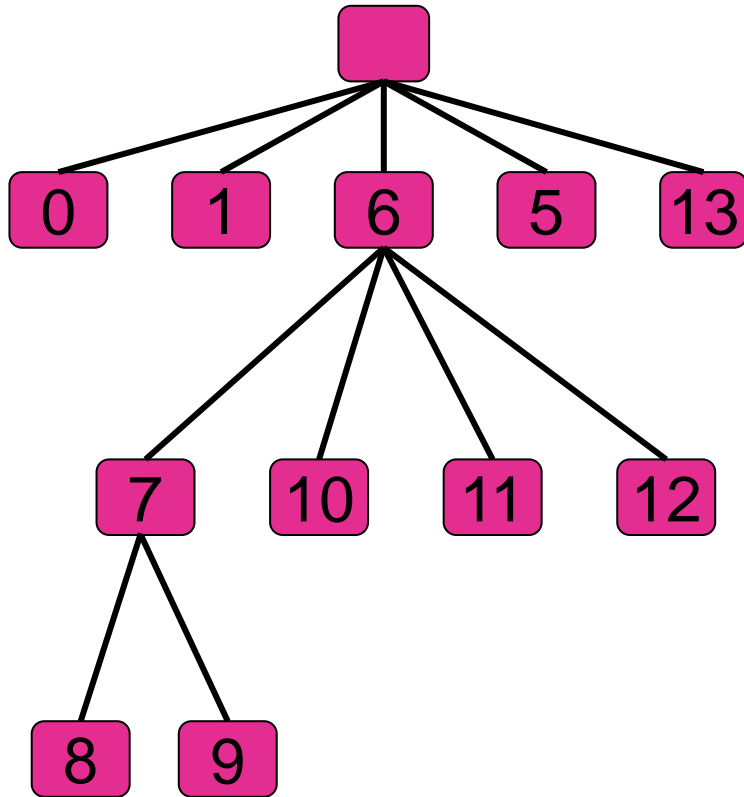
- Zu *einem* Objekt gehörende Pixel zusammenfassen
- Geht auch bei wenn Pixel einen Objekttyp angibt (Objektklassen)
- OpenCV: `findContours`
  - [https://docs.opencv.org/3.3.1/d3/d05/tutorial\\_py\\_table\\_of\\_contents\\_contours.html](https://docs.opencv.org/3.3.1/d3/d05/tutorial_py_table_of_contents_contours.html)
- Algorithmus: Satoshi Suzuki and others.  
Topological structural analysis of digitized binary images by border following. Computer Vision, Graphics, and Image Processing, 30(1):32–46, 1985.



- Kontur ist zusammenhängende geschlossene Linie um eine zusammenhängende Gruppe von Pixeln einer Objektklasse herum
- Sonderfall: Kontur hört am Rand auf
- Manchmal befinden sich Löcher oder Regionen anderer Objektklasse innerhalb (komplett umschlossen) einer Region
- Wichtige Information für Gruppierung
- Dargestellt als Baum / Hierarchie
- Beispiel:
  - Augen, Nase Mund im Teller
  - Schwarze Flecken im Mund



# Hierarchie von Regionen



- Frage an das Auditorium: Welche zwei typischen Probleme treten bei diesem Vorgehen auf?





- Frage an das Auditorium: Welche zwei typischen Probleme treten bei diesem Vorgehen auf?
- ▶ **Zwei Objekte berühren sich und werden zu einem**
- ▶ **Ein Objekt zerfällt durch Erkennungsfehler in mehrere Objekte**



- Morphologie zum Nachbearbeiten der binarisierten Bilder
- Erosion: Kleiner machen
- Dilation: Größer machen
- Opening: Kleine Flecken entfernen (erst kleiner dann größer machen)
- Closing: Kleine Löcher entfernen (erst größer dann kleiner machen)
- Bildquelle: OpenCV 3.0 Dokumentation
- [https://docs.opencv.org/3.0-beta/doc/py\\_tutorials/py\\_imgproc/py\\_morphological\\_ops/py\\_morphological\\_ops.html](https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_imgproc/py_morphological_ops/py_morphological_ops.html)



Erosion



Dilatation



Opening



Closing



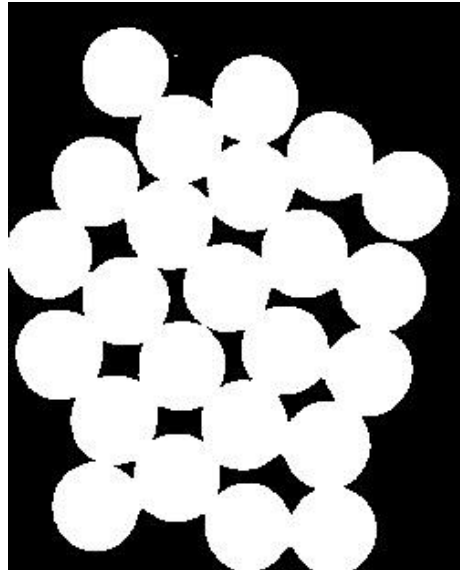
- Leitfrage: Welches Merkmal definiert die Trennung?
  - oft: Engstelle im Binärbild
  - oft: Leichte Linie im Originalbild
  - Objekte berühren sich an einer Stelle, nicht mit voller Breite
- Trennen durch Erosion / Opening?
  - ja, aber beeinträchtigt Größe (Erosion) / Form (Opening)
- Idee "Wasserscheidentransformation"
  - berechne Bild von Objektkernen ("sure Objects") durch aggressive Erosion
  - berechne Regionen nach Objektkernen (Objekte getrennt aber kleiner)
  - dehne Regionen aus, so dass die Grenze die Stelle mit dem höchsten Bildkontrast ist
  - Analogie: Bildkontrast entspricht Berghöhe. Der höchste Berg zwischen zwei Regionen trennt als Wasserscheide in welche Region der Regen fließt.



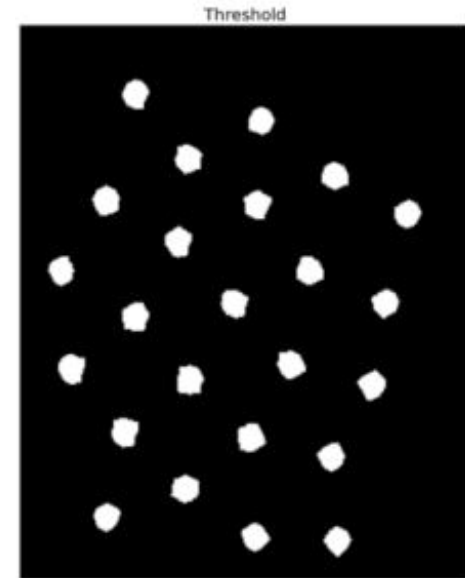
Quelle: [https://docs.opencv.org/3.3.1/d3/db4/tutorial\\_py\\_watershed.html](https://docs.opencv.org/3.3.1/d3/db4/tutorial_py_watershed.html)



Eingabe

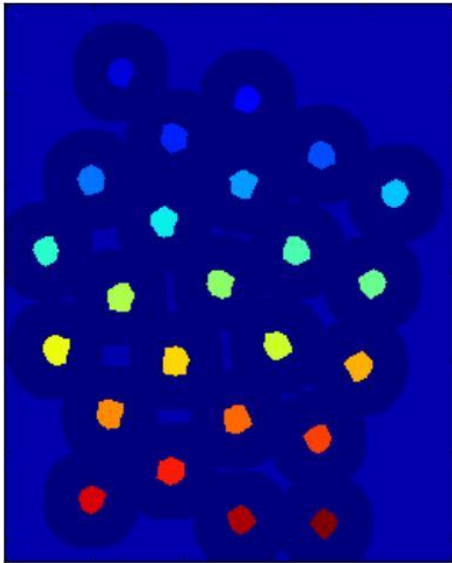


Segmentierung

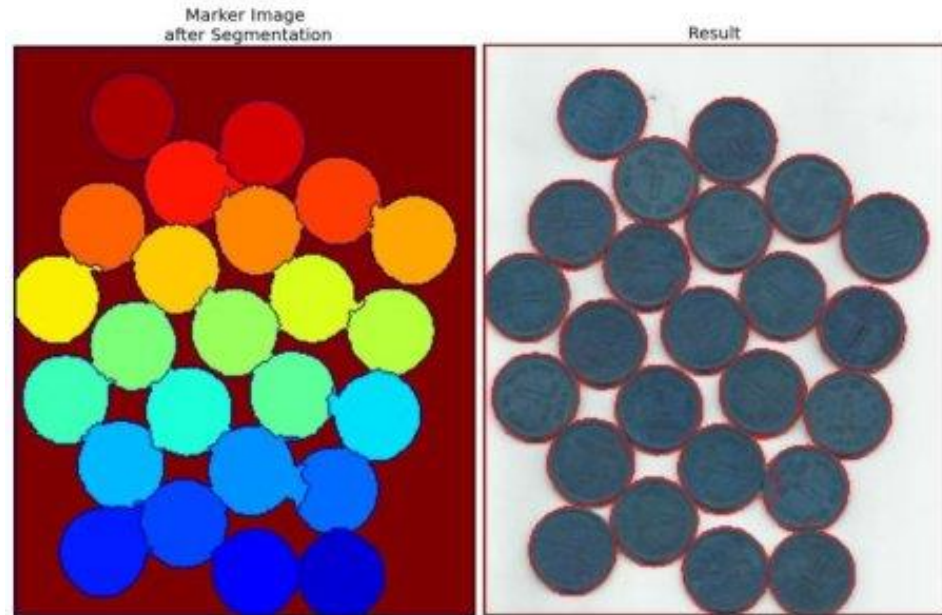


Kerne  
(mit Erosion  
oder Distanztransformation)

Quelle: [https://docs.opencv.org/3.3.1/d3/db4/tutorial\\_py\\_watershed.html](https://docs.opencv.org/3.3.1/d3/db4/tutorial_py_watershed.html)



Kerne → Regionen

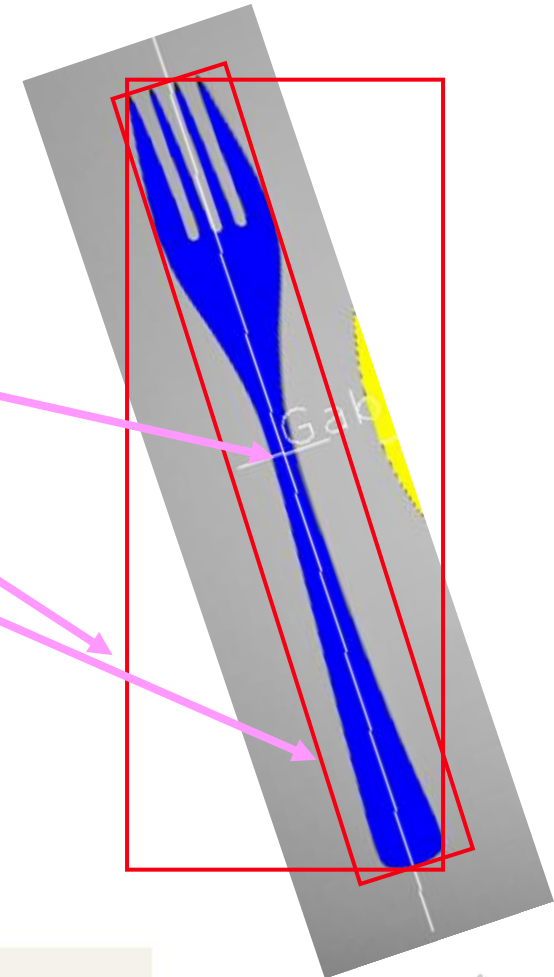


mit Wasserscheiden-  
transformation auf  
volle Segmentierung erweitert

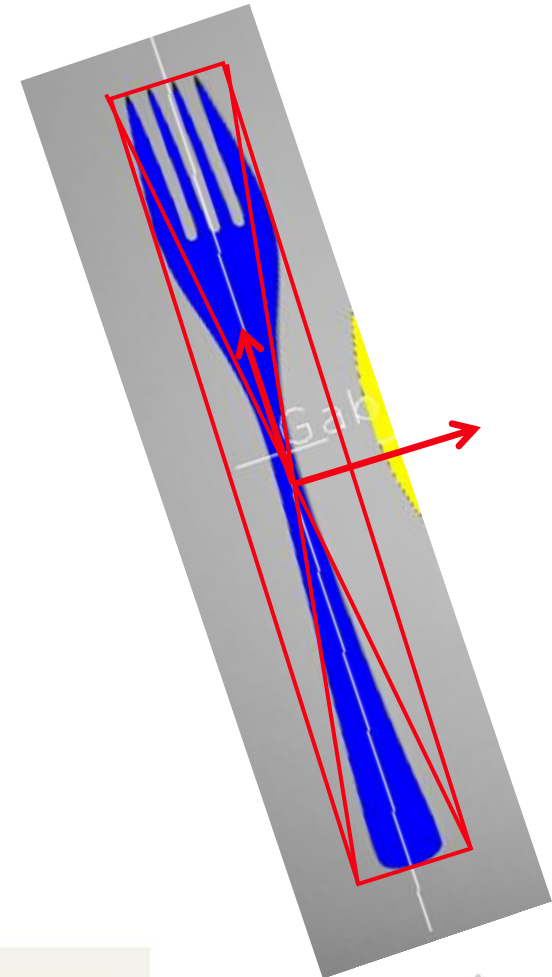
- Fälschlich vereinte Konturen
  - z.B. wenn es um Zählen geht, über Standardfläche entscheiden, ob eine Region ein oder zwei Objekte sind
  - oder Regionen nach Seitenverhältnis teilen
- Fälschlich getrennte Konturen
  - im Nachhinein vereinen
  - Abstandsschwellwert
  - Formähnlichkeit
  - konvexe Hülle über Teile, wenn wirkliches Objekt konvex



- aus Regionen (bzw. Konturen) Zahlen für Anwendungslogik bestimmen
  - für Klassifikation, Lokalisation, oder Maße
- OpenCV: Structural Analysis and Shape Descriptors
  - Schwerpunkt, Momente, Hu-Momente (beschreiben Form für Formvergleich)
  - Umfang, Fläche, Bounding-Box (achsenparallel, nicht drehinvariant)
  - konvexe Hülle / konvexe Defekte
  - passende Ellipse, **minimales umschließendes Rechteck** (→Position, Maße, Orientierung)
  - Minimal umschließendes Rechtecke wichtigstes Merkmale

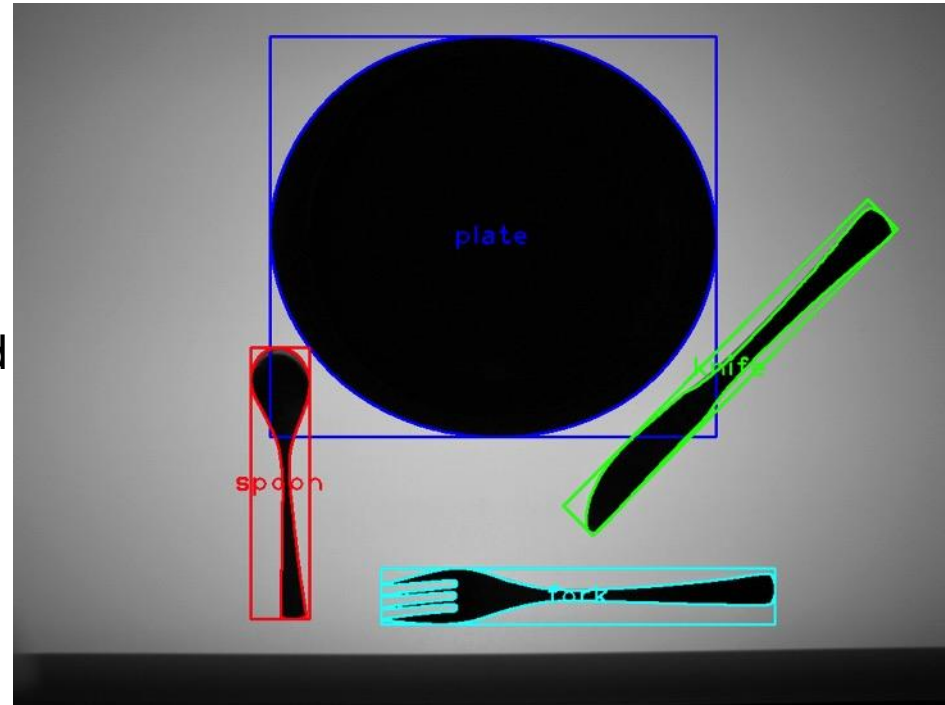


- Minimales umschließendes Rechteck als Referenzsystem
  - Entscheidung zwischen vorne / hinten benötigt
- Bewegt sich mit dem Objekt mit
  - translations / rotationsinvariant
- Definiert Objektkoordinatensystem
  - z.B. Mitte des Rechtecks, parallel zu Kanten
- Anwendungslogik in diesen Koordinaten formulieren
  - absolut in Pixeln
  - oder relativ zur Rechtecklänge → skalierungsinvariant
  - z.B. Objektteile ("Zinken im oberen Viertel") für weitere Überprüfungen ausschneiden und weiterverarbeiten
  - Punkte für Roboter zum Greifen
- Maske zum Vergleich an Koordinaten ausrichten





- Kriterien bzgl. Größe, Länge, Breite, Seitenverhältnis, ...
  - Filtern um Fehlregionen zu entfernen
  - Klassifizieren
- Relative vs. Absolut
  - absolute Zahlen hängen von Kameraabstand und Zoom ab
  - absolute Zahlen von anderen Objekten unabhängig
  - relative Zahlen ("Das größte ist ein Teller") funktionieren nur wenn alle Objekte da sind
  - relative Zahlen von Kameraabstand und Zoom unabhängig



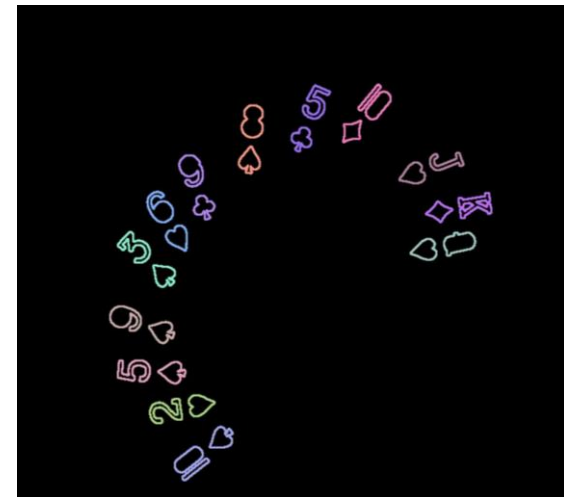
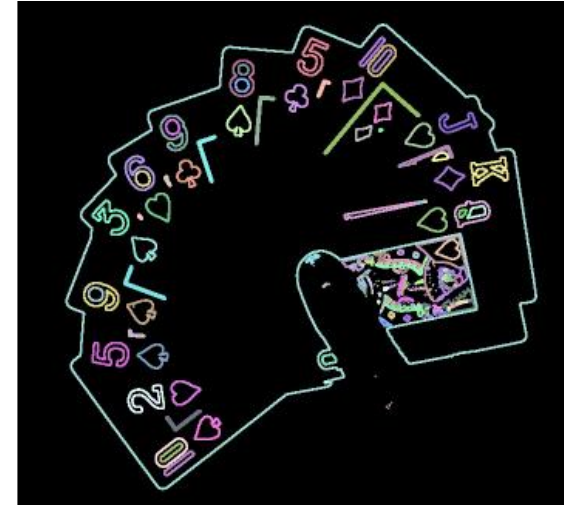


## Frage an das Auditorium

Wie komme ich von den  
Konturen zu  
den Karten?



- Konturen nach Fläche, ggf. Länge / Breite, filtern um Zahlen und Farbsymbole (Pik, Karo, König, Ass) vom Rest zu trennen
- Innere Konturen (bei 0, 6, 8, 9) ignorieren
- minimales umschließendes Rechteck
- für hypothetisches Zahl-Rechteck: längere Kante ist vertikal (im Sinne der Karte). Sollpunkt für Farbsymbol relativ zu Position / Orientierung des Rechtecks vorgeben. Die dazu nächste Kontur wählen und mit der Zahl zu einer Karte gruppieren.
- Ziffern / Farbsymbole erkennen durch Vergleich von Kennzahlen, z.B. Hu-Momente oder Pixeln
- 1 ignorieren und 0 als 10 werten oder analoge Gruppierungsoperation horizontal machen







## Anwendung: RoboCup

- früher zuerst Farbsegmentierung
- heute: formbasiert, aber Rasen grün



Segmentierungsgetriebene BV enthält sehr viele einstellbare Zahlenparameter

- "Fluch der Bildverarbeitung",
  - aber schwer zu vermeiden
  - meist neue Umgebung → veränderte Parameter
- Umgang mit Parametern
  - Namen und Erklärungen geben
  - aus Konfigurationsdatei laden
  - Intuition entwickeln, wovon sie abhängen (Geometrie, Beleuchtung, Algorithmus)
- Später: Klassifikation Lernen
- "Dafür gibt es einen Dr. Threshold", Zitat von Wolfram Burgard

- Hinweis: Wir fragen in der Klausur nach Lösungsideen für derartige einfache Anwendungen.
- Aufgabe bis zum interaktiven Repetitorium am 11.11.24:  
Überlegt Euch zu den nachfolgenden vier Anwendungen  
Lösungskonzepte

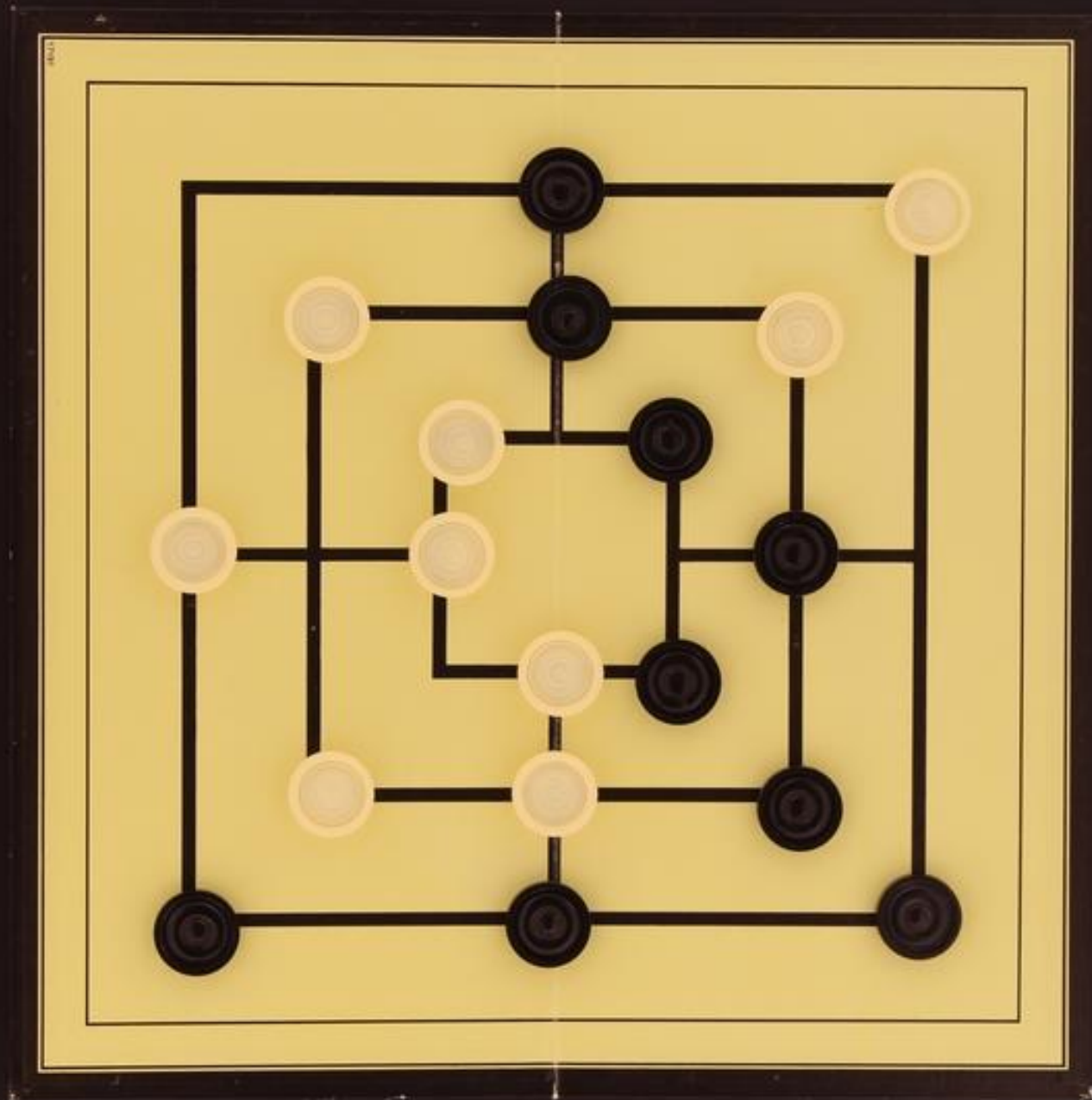




Die Würfel mit der jeweiligen Augenzahl  
sollen erkannt werden!

Was erwartest Du, das schief gehen  
könnte?





Der  
aktuelle  
Spiel-  
stand  
soll  
erkannt  
werden.





Der aktuelle  
Spielstand  
soll erkannt  
werden.



## Segmentierungsgetriebene Bildverarbeitung

- Kamera von oben, festes Licht, fester Hintergrund
- Segmentierung / Binarisierung
  - Grauwertschwellwert, HSV-Bereich, Kanten
- Gruppierung
  - Konturen mit Hierarchie finden, Morphologie, Wasserscheidentransf.
- Kennzahlen bilden
  - geometrische (Länge, Breite, Fläche, Krümmung, ...)
  - farbliche (mittlere Farbe, # Pixel eine Farbe)
  - minimales umschließendes Rechteck für Lage, definiert Objekt-Koordinatensystem
- Filtern & Klassifizieren
  - anwendungsspezifische Heuristiken, selbst ausgedacht
  - oft Schwellwerte für Größen, Gruppierung mit Nachbarn, Suchregionen relativ zu umschließendem Rechteck