Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

MIN-Fakultät
Fachbereich Informatik
Arbeitsbereich SAV/BV (KOGS)

Nützliche Bildverarbeitungs-Verfahren

BV-Praktikum im Sommersemester 2014

Benjamin Seppke und Susanne Germer

Agenda

- Wiederholung: Bilder in Racket
- Schwellenwertverfahren/Binarisierung
  - Manuell
  - Histogramm-basiert
- Kantenfinder
  - Morphologisch
  - Canny
- Festlegen des freizustellenden Bereichs
- Koordinatensysteme und Projektionen auf unterschiedliche Spielfeldgrößen

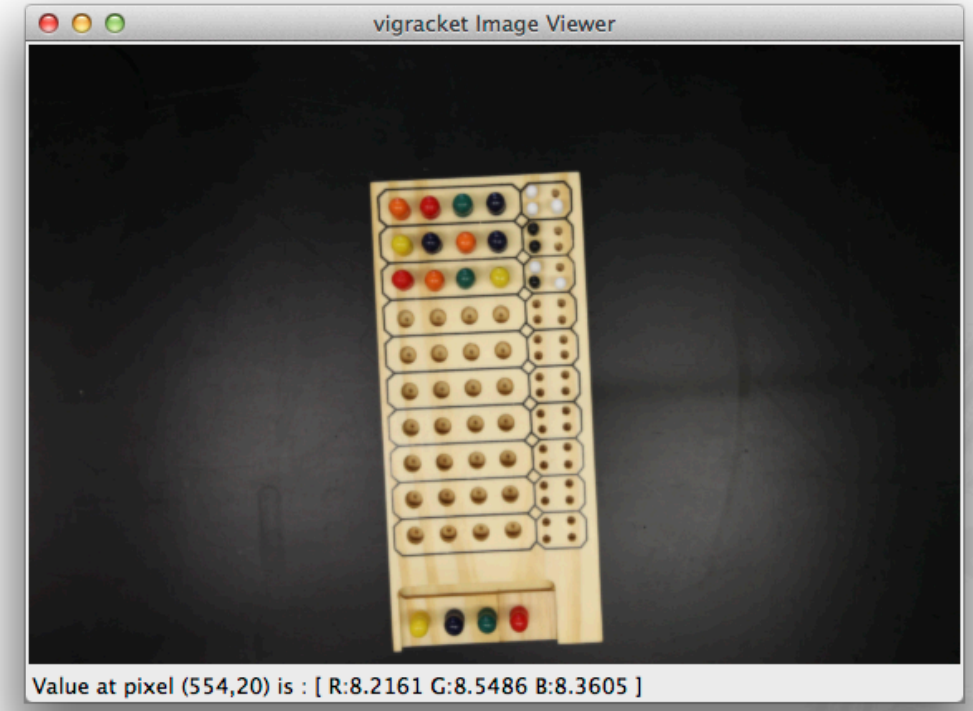
Wiederholung: Bilder in Racket

- Bilder werden in Racket als Listen von „carrays“ dargestellt:

  ‘(<carray>) für Grauwertbilder

  ‘(<carray> <carray> <carray>) für RGB-Farbbilder
- Bedeutung nur implizit, zum Beispiel beim Viewer vorhanden.
- Die meisten Algorithmen können auf beliebig lange Listen von carrays angewendet werden.
- Trennung zwischen Bild-Datensatz (oben) und Bild-Bitmap (mit image->racket-image)
  - VigRACKET-Operationen nur mit obiger Repräsentation kompatibel!
  - 2htdp/image-Operationen nur nach Konvertierung möglich!

Das Beispiel für heute:



- Vorverarbeitung: Verkleinern des Ausgangsbildes
- Aufgabe 1: Isolieren (Freistellen) des Spielfeldes aus dem Bild
  - Schwellenwertbasierte Verfahren
  - Kantenbasierte Verfahren
  - Segmentierung und Labelling
- Aufgabe 2: „Abtasten“ des freigestellten Spielfeldes in Bezug auf den Spielzustand
  - Koordinatentransformationen
  - Gitter festlegen

```
(define image_dir "/Users/seppke/...“)
(define img (load-image (build-path image_dir "IMG_8553.JPG")))
(define resized_img (resizeimage-percent img 10))
```

Binarisierung

Bei vielen Anwendungen ist es nützlich, nur zwischen zwei diskreten Grauwerten zu unterscheiden, "schwarz“ und "weiß", oder "1" und "0", "Objekt" und "Hintergrund".

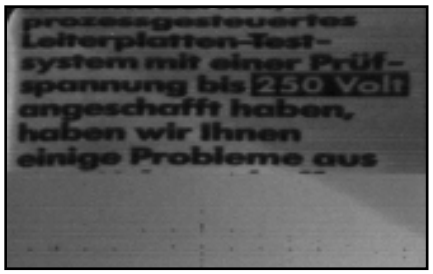
Beispiel: Objekt von Hintergrund trennen ("segmentieren")

Binarisieren = Bild in Binärbild transformieren

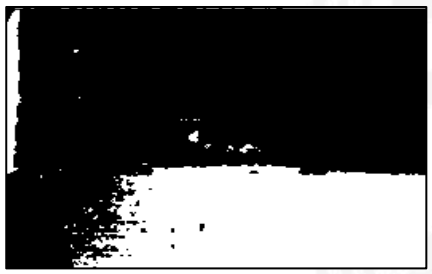
Binarisieren kann bei kontinuierlichem oder höher aufgelöstem Grautonbild erfolgen, auch bei verarbeiteten Bildern, z.B. Gradientenbetragsbildern.

Schwellenwertvergleich: $g(x, y) = \begin{cases} 0 & \text{für } g(x, y) < T \\ 1 & \text{für } g(x, y) \geq T \end{cases}$ T ist Schwellenwert

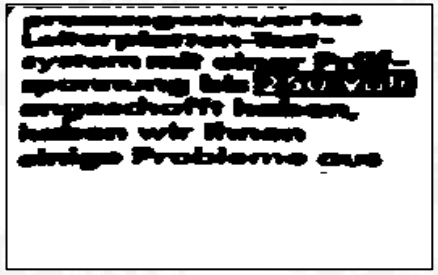
Grauwerte 0 – 255



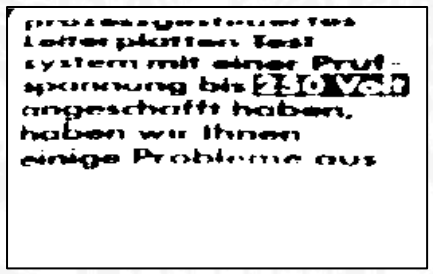
Schwellenwert 64



Schwellenwert 32



Schwellenwert 16



Schwellenwertbestimmung durch Probieren

Ein Schwellenwert, der Objekt und Hintergrund perfekt trennt, existiert nicht immer.

Auswahl durch Probieren:
Wähle Schwellenwert, bis eine Bildeigenschaft erfüllt ist, z.B.

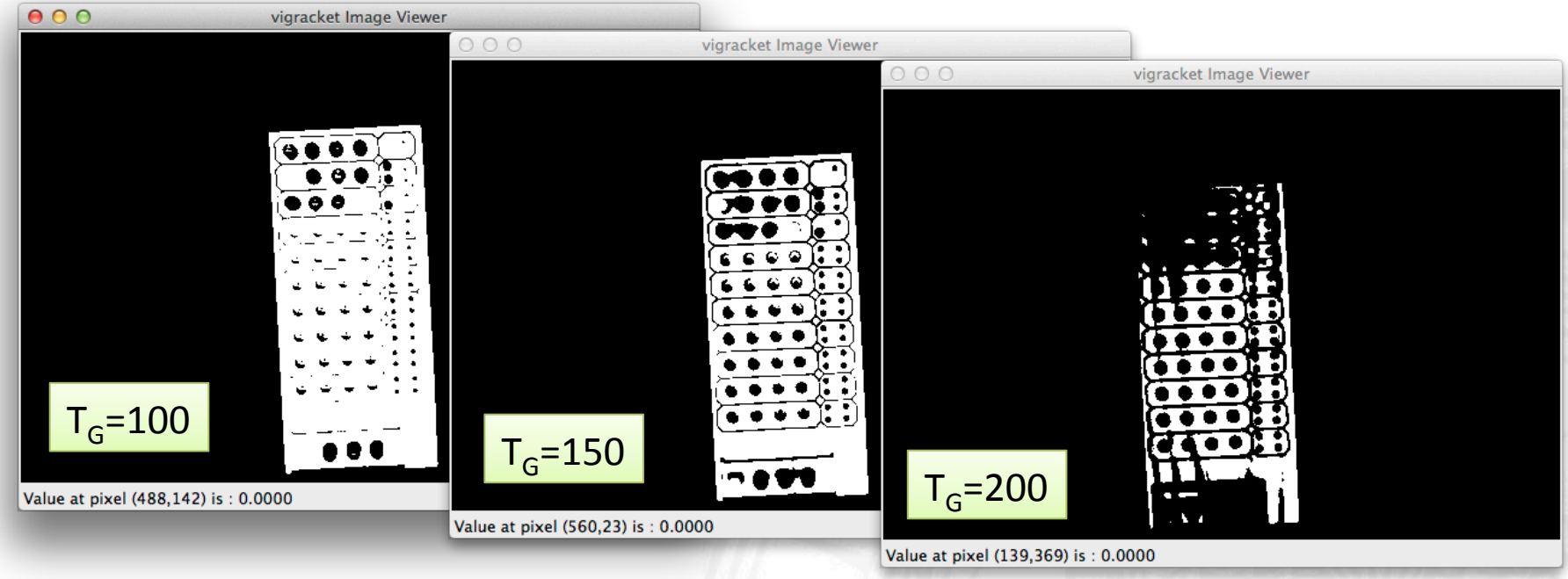
- Verhältnis von schwarzen zu weißen Pixeln $q = \frac{\#\ \text{weiße Pixel}}{\#\ \text{schwarze Pixel}} \Rightarrow q_0$
- Linienbreite $\textit{Linienbreite} \Rightarrow \mathrm{d}_0$
- Zahl der zusammenhängenden Komponenten $\textit{Komponentenzahl} \Rightarrow \mathrm{n}_0$

Bei logarithmischer Suche kann die Zahl der Probeversuche klein gehalten werden.

Beispiel:

Um einen Schwellenwert $0 \leq T \leq 255$ auszusuchen, braucht man höchstens 8 Versuche zur Bestimmung der besten Annäherung an q_0.

Beispiel mit manuell ermittelten Schwellenwerten

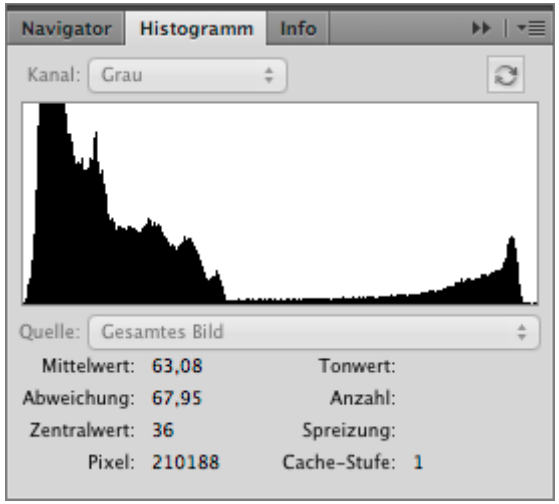


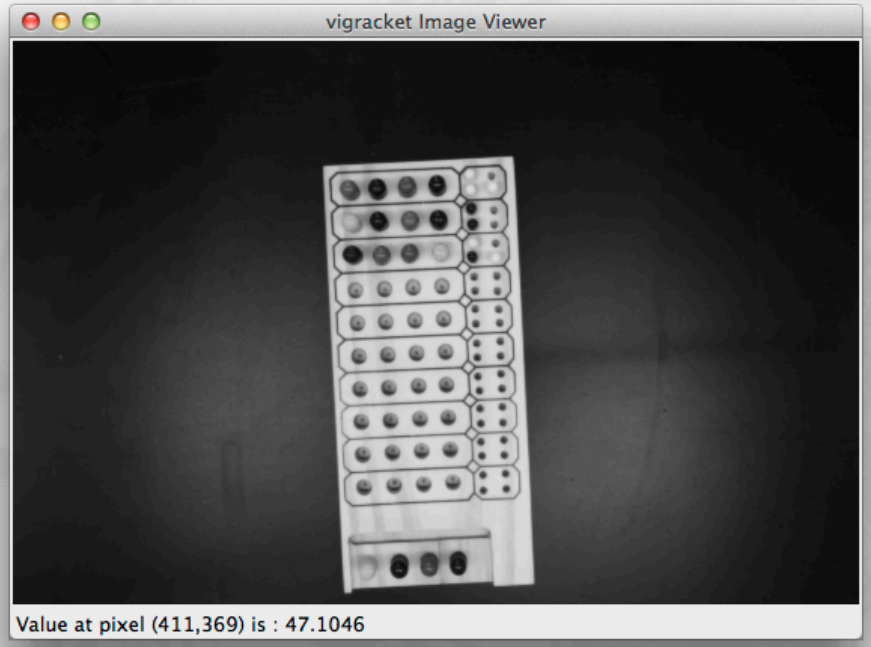

```
(define work_img (image->green resized_img))
(show-image (image-map (lambda (x) (if (> x 100.0) 255.0 0.0)) work_img))
(show-image (image-map (lambda (x) (if (> x 150.0) 255.0 0.0)) work_img))
(show-image (image-map (lambda (x) (if (> x 200.0) 255.0 0.0)) work_img))

(define threshold_img (image-map (lambda (x) (if (> x 150.0) 255.0 0.0)))
```

Bildhistogramme

- Idee: Automatische Bestimmung des Schwellenwerts anhand der Bildstatistik
- Histogramm: Haufigkeitsdiagramm aller Helligkeitswerte (meist im Intervall [0..255]) des Bildes
- Beispiel:

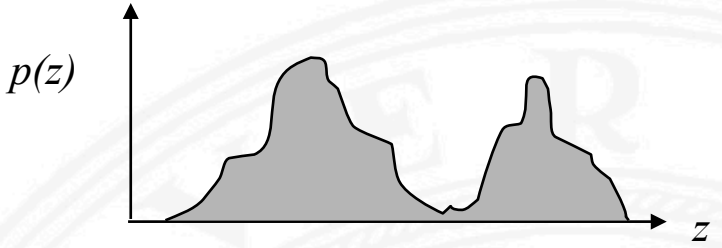




Schwellenwertbestimmung aufgrund der Grauwertverteilung

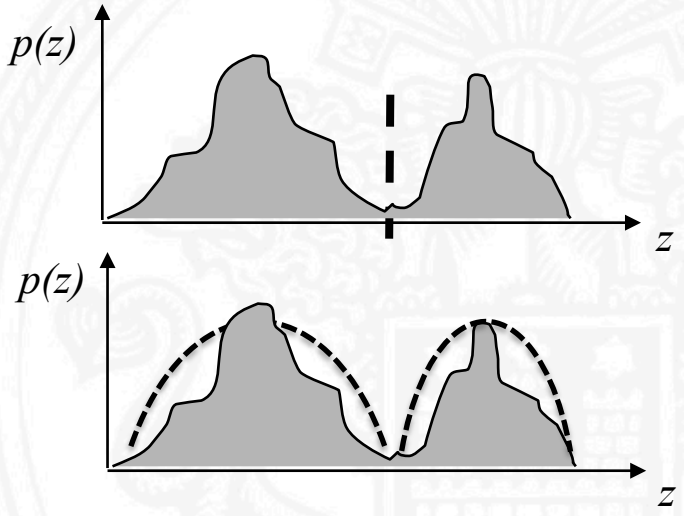
Eine Grauwertverteilung (Wahrscheinlichkeitsdichte oder Histogramm) kann bimodal sein:

Was ist ein plausibler Schwellenwert?



Zwei übliche Methoden zur Bestimmung eines plausiblen Schwellenwertes:

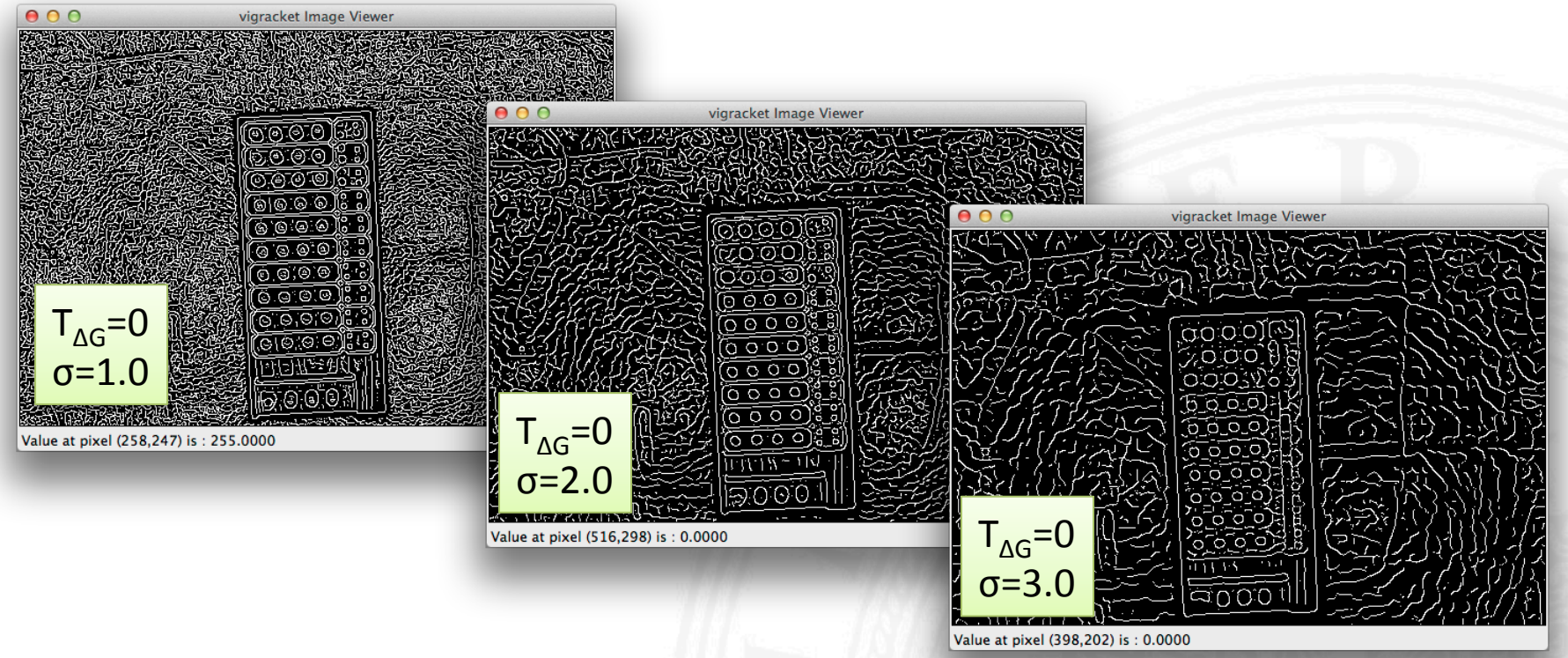
1. Suche das „Tal“ zwischen zwei „Hügeln“
2. Passe Hügelschablonen an und bestimme (deren) Schnittpunkt.



Kantenfinder

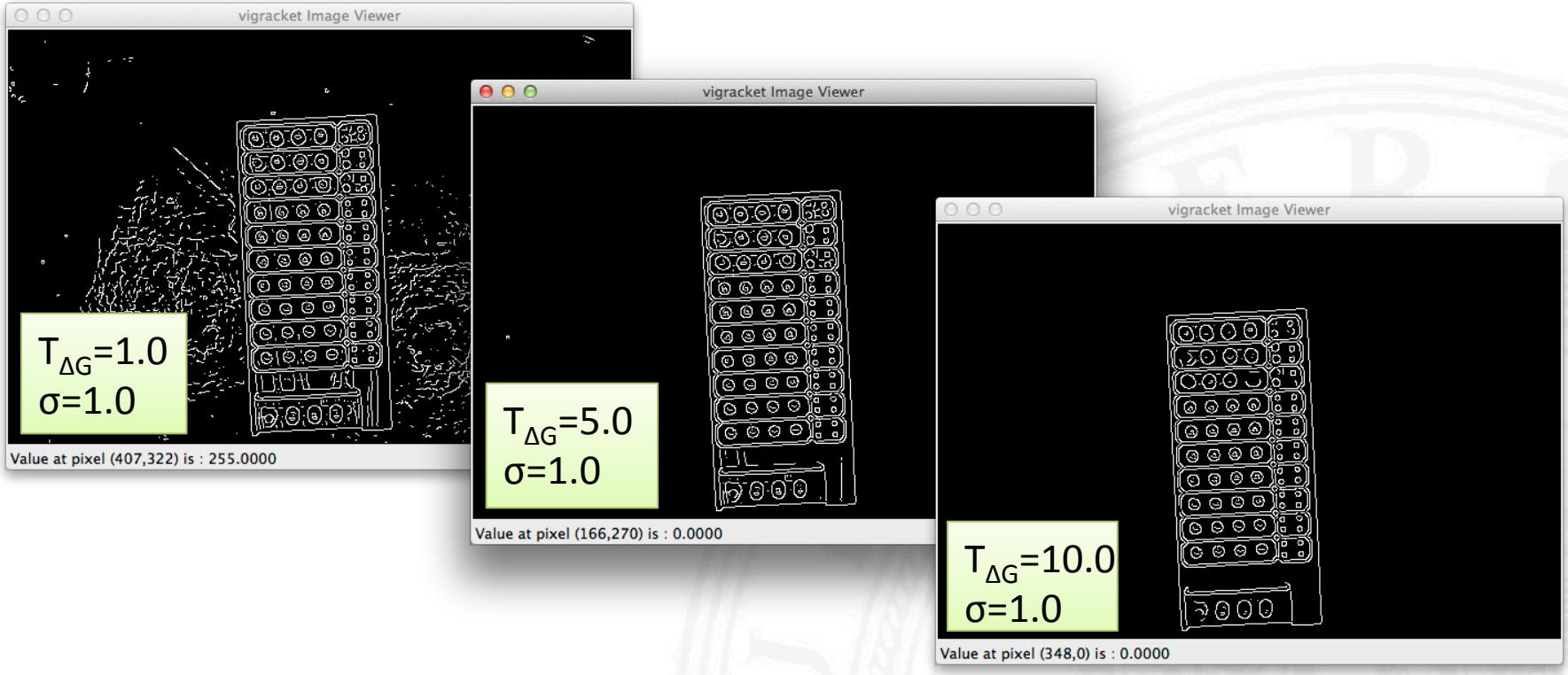
- Duale Sichtweise auf das Problem: Finde das Spielfeld anhand der begrenzenden Kanten
- Verfahren:
  - Maximum der ersten Ableitung der Bildfunktion
  - Nullstellen der zweiten Ableitung der Bildfunktion
- Beispiel: Canny-Kantendetektor
  - Parameter:
    - Unterer Schwellenwert der Kantenstärke
    - Skala auf der die Kanten ermittelt werden
  - Vorgehensweise:
    - Auffinden und Ablaufen von lokalen Maxima
    - Hysterese Verfahren

Beispiele: Anwendung des Canny-Kantendetektors




```
(show-image (cannyedgeimage work_img 1.0 0.0 255.0))
(show-image (cannyedgeimage work_img 2.0 0.0 255.0))
(show-image (cannyedgeimage work_img 3.0 0.0 255.0))
```

Beispiele: Anwendung des Canny-Kantendetektors




```
(show-image (cannyedgeimage work_img 1.0  1.0 255.0))
(show-image (cannyedgeimage work_img 1.0  5.0 255.0))
(show-image (cannyedgeimage work_img 1.0 10.0 255.0))

(define canny_img (cannyedgeimage work_img 1.0 10.0 255.0))
```

Festlegen des freizustellenden Bereichs

- Idee: Suche nach dem begrenzenden Rechteck um das (gedrehte) Spielfeld herum:

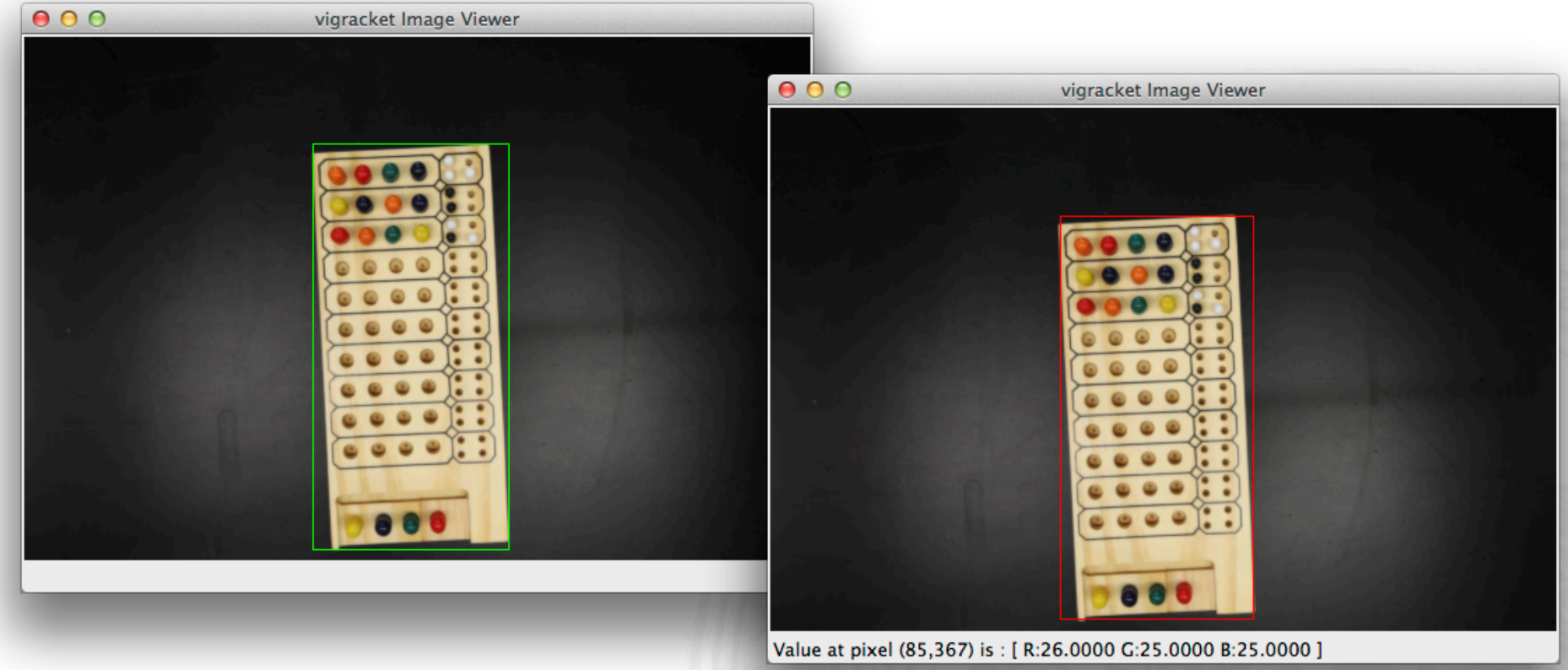
```
;; BBox: 0 - left, 1 - upper, 2 - right, 3 - lower
(define (findBBox x y pixel bbox)
  (when (>  (car pixel) 0.0)
    (begin
      (when (> x (vector-ref bbox 2))    (vector-set! bbox 2 x))
      (when (< x (vector-ref bbox 0))    (vector-set! bbox 0 x))
      (when (> y (vector-ref bbox 3))    (vector-set! bbox 3 y))
      (when (< y (vector-ref bbox 1))    (vector-set! bbox 1 y)))))
```

- Anwendung mit *image-for-each-pixel*:

```
(define bbox1 (vector (image-width resized_img) (image-height resized_img) 0 0))
(define bbox2 (vector (image-width resized_img) (image-height resized_img) 0 0))

(image-for-each-pixel (curryr findBBox bbox1)  canny_img)
(image-for-each-pixel (curryr findBBox bbox2)  threshold_img)
```

Beispiel: Erkennung des begrenzenden Rechtecks




```
(show-image (racket-image->image (overlay-bboxes resized_img (list bbox1) '(green))))
(show-image (racket-image->image (overlay-bboxes resized_img (list bbox2) '(red))))
```

Ausschneiden des Bildes anhand des begrenzenden Rechtecks

- Funktion aus 2htdp/image verwenden:

```
(define (cropimage img ul_x ul_y lr_x lr_y)
  (let ((w (- lr_x ul_x))
        (h (- lr_y ul_y)))
    (racket-image->image (crop ul_x ul_y w h (image->racket-image img)))))
```

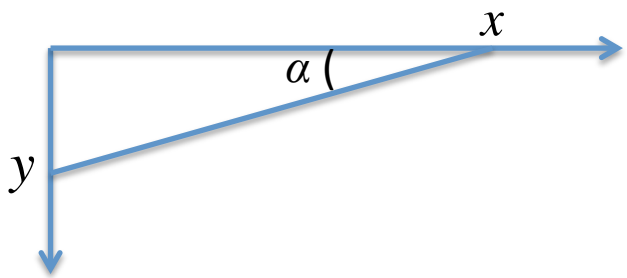
- Anwenden auf das (mit Canny) erzielte Ergebnis:

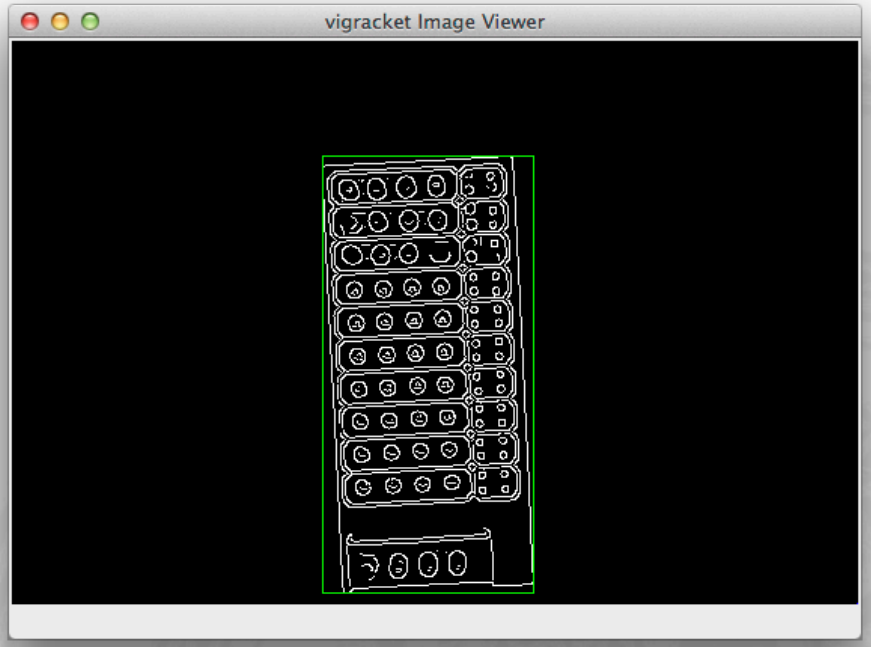
```
(define crop1_img (cropimage  resized_img
                              (vector-ref bbox1 0)
                              (vector-ref bbox1 1)
                              (vector-ref bbox1 2)
                              (vector-ref bbox1 3)))
```

- **Achtung:** Leichte Drehung nach wie vor vorhanden..

Erkennung der Rotation als Nachverarbeitung

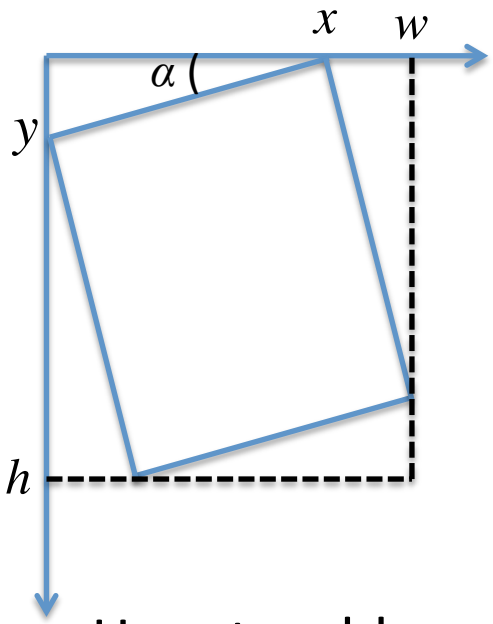
- **Idee**: Betrachten der Schnittpunkte zwischen begrenzendem Rechteck und Maske/Canny Ergebnisbild
- Geometrische Approx. der Drehung durch Schnittpunktposition





$$\alpha = \arctan\left(\frac{y}{x}\right)$$

Alternative Rotationserkennung



$$\alpha = \arctan\left(\frac{y}{x}\right)$$

$$= \arctan\left(\frac{w-x}{x}\right)$$

$$= \arctan\left(\frac{y}{h-y}\right)$$

$$\ldots$$

- Hauptprobleme: x,y unscharf definiert
- Außerdem möglich:
  Begrenzende Rechtecke durch Polygone ersetzen
  Vorteile:
  - Informationen zum Ausschneiden direkt vorhanden
  - Transformation (inkl. Rotation) direkt aus Eckpunkten ablesbar

Ende der Präsentation

Vielen Dank für die Aufmerksamkeit!

Die hier vorgestellten Folien sowie der Quelltext zum Ausprobieren sind ab sofort auch der Veranstaltungs-Seite zu finden!

Viel Spaß beim Bilder Aufnehmen und Auswerten!