Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

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

Ordnung und Verfahren

BV-Praktikum im Sommersemester 2017

Leonie Dreschler-Fischer, David Mosteller
und Benjamin Seppke

Agenda

- Ordnung:
  - Beispielhafte Verzeichnisstruktur
  - Absolute und relative Pfade in Racket/vigracket
- Verfahren:
  - Festlegen des freizustellenden Bereichs
  - Freistellung
  - Auslesen des Spielzustands

Ordnung

„Ordnung ist das halbe Leben, ich lebe in der anderen Hälfte“ (unbekannt)

Aber:

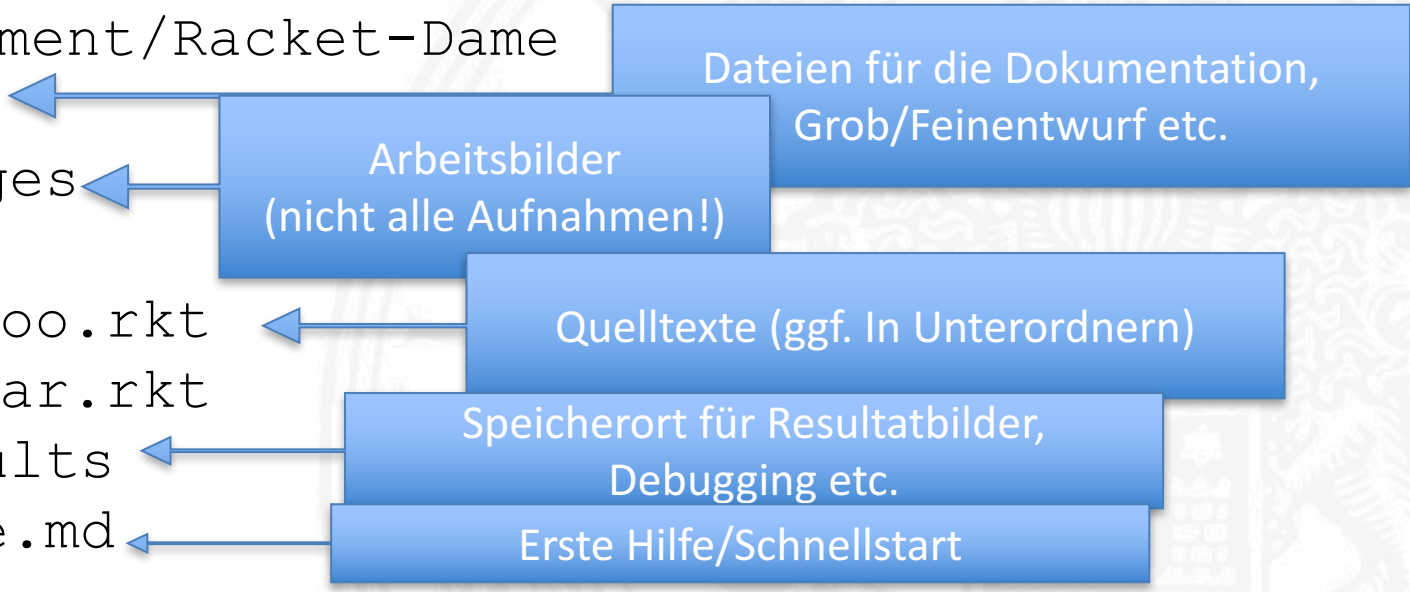
- Ordnung ist wichtig!
- Sie sollte uns Zeit sparen!
- Aufwand:
  - 1x Struktur festlegen,
  - dann „nur noch“ daran halten!

- Wir versuchen es mal...

Ordnung auf Verzeichnisebene

- Einfach zu bewerkstelligen!
- Versionsverwaltung übernimmt Backups (Einführung in 2 Wochen)
- Beispiel:

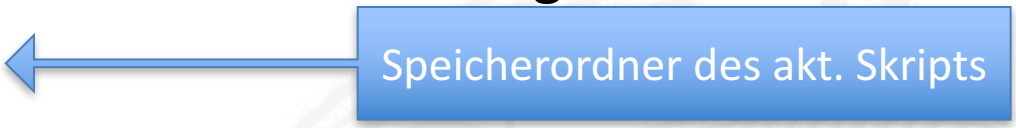
```
~/development/Racket-Dame
    ./doc
    ./images
    ./src
        ./foo.rkt
        ./bar.rkt
    ./results
    Readme.md
```



Absolute und relative Pfade

- Oft werden in Racket/vigracket absolute Pfade benötigt.
- Zum Glück gibt es Hilfe bei der Umwandlung:

```
(current-directory)
```



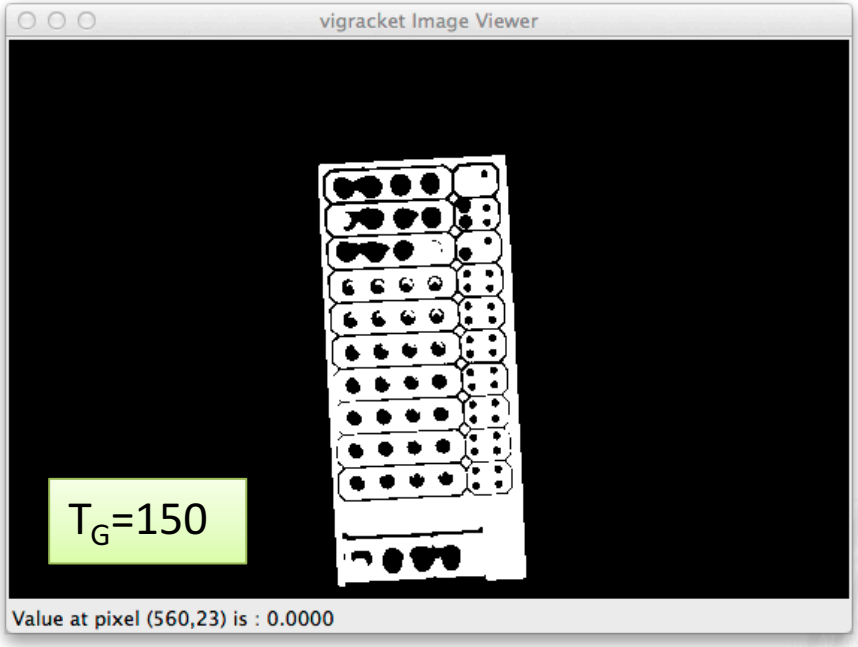
- Nach der Ordnung der vorigen Folie bedeutet das:

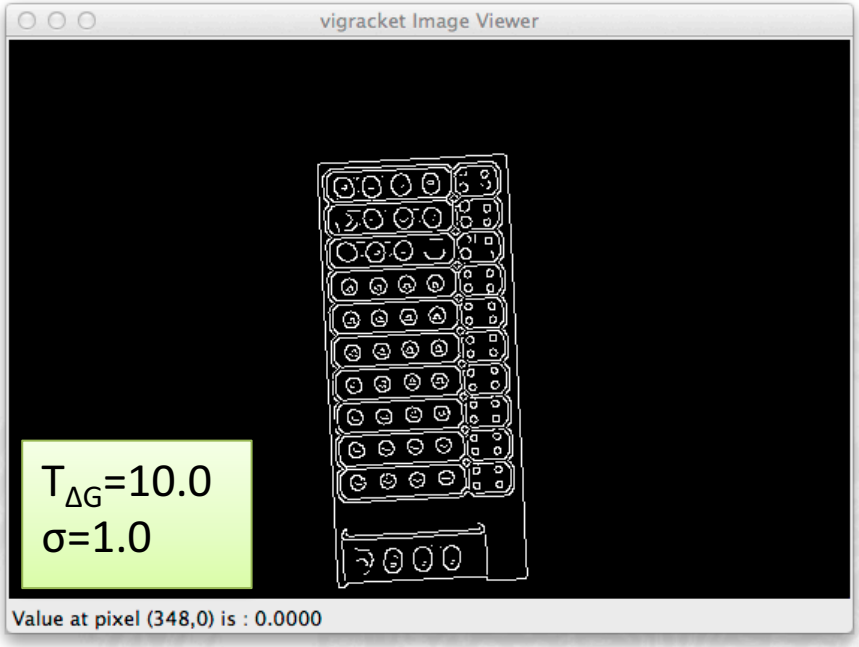
```
foo.rkt:

(require vigracket)

(define src_dir  (current-directory))
(define base_dir (build-path src_dir 'up))
(define img_dir  (build-path base_dir "images"))

(define (load-image-rel rel-path) ;rel-path: filename in "images" folder
    (load-image  (build-path img_dir rel-path)))

;Test:
(define img (load-image-rel "IMG_0998.JPG"))
```

Bildverarbeitungsverfahren “reloaded”

- Annahme:
  Spielfeld-Aufnahme ist durch Verfahren in Hintergrund und Vordergrund (Spielfeldstrukturen) unterschieden
- Heute:
  - Wiederholung (für einige): Von der Segmentierung zum Spielfeld
  - Neu (für alle): Alternative Vorgehensweise
  - Neu (für alle): Auslesen eines Spielzustands

Beispiele: Schwellenwertbildung und Canny-Kantendetektor






```
(define threshold_img (image-map (lambda (x) (if (> x 150.0) 255.0 0.0)))
(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
- Unterteilung des Bildes in Regionen durch Labelling:

```
(define canny_labels (labelimage  canny_img))
```

- Für jede Region können Statistiken/Features abgerufen werden:

```
(define canny_features (extractfeatures canny_img canny_labels))
```

- Wir suchen die größte Region, die nicht Hintergrund ist!

Verwendung von Regionsfeatures

- Beispiel:

```
(image->list canny_features)

'(((0.0 +inf.0 +inf.0 -inf.0 -inf.0 0.0 0.0 3.4028234663852886e+38 -3.4028234663
(203636.0 0.0 0.0 561.0 373.0 280.66784667968875 185.9539794921875 0.0 0.0 0.0 0.0
(822.0 206.0 76.0 346.0 366.0 275.6910095214844 221.77859497070312 255.0 255.0 25
(386.0 209.0 81.0 334.0 234.0 284.9844665527344 127.30052185058594 255.0 255.0 25
(83.0 297.0 84.0 326.0 104.0 310.4337463378906 93.67469787597656 255.0 255.0 255.
(201.0 212.0 85.0 295.0 108.0 254.05970764160156 96.3631820678711 255.0 255.0 255
(17.0 315.0 87.0 320.0 92.0 317.23529052734375 89.35294342041016 255.0 255.0 255.
(51.0 255.0 89.0 268.0 104.0 261.3529357910156 96.60784149169922 255.0 255.0 255.
(47.0 276.0 89.0 287.0 103.0 281.3404235839844 95.68085479736328 255.0 255.0 255.
(50.0 236.0 91.0 248.0 105.0 241.82000732421875 98.18000030517578 255.0 255.0 255
(50.0 217.0 92.0 230.0 105.0 223.24000549316406 98.72000122070312 255.0 255.0 255
(2.0 326.0 97.0 326.0 98.0 326.0 97.5 255.0 255.0 255.0 0.0)
(15.0 294.0 103.0 300.0 108.0 297.1333312988281 105.93333435058594 255.0 255.0 25
(26.0 309.0 106.0 326.0 114.0 319.3461608886719 107.61538696289062 255.0 255.0 25
(175.0 213.0 107.0 296.0 130.0 252.23428344726562 117.17142486572266 255.0 255.0
(23.0 300.0 107.0 307.0 115.0 303.9565124511719 111.26087188720703 255.0 255.0 25
...
```

Hintergrund

Bestimmung des umschließenden Rechtecks

```
;;Find the region with the most pixels assigned to
(define (largestRegionID features [idx 0] [max_idx 0])
  (if (= idx (image-height features))
      max_idx
      (largestRegionID features
                       (+ idx 1)
                       (if (> (image-ref features 0 idx 0)
                              (image-ref features 0 max_idx 0))
                           idx
                           max_idx))))

;;Uses the above function to find the bounding box of the largest region
(define (largestRegionBBox image)
  (let* ((labels (labelimage image))
         (features (extractfeatures image labels))
         (max_region_id (largestRegionID features 2 2)))
    (vector (inexact->exact (image-ref features 1 max_region_id 0)) ;left
            (inexact->exact (image-ref features 2 max_region_id 0)) ;upper
            (inexact->exact (image-ref features 3 max_region_id 0)) ;right
            (inexact->exact (image-ref features 4 max_region_id 0)) ;lower
           )))
```

Alternative: Festlegen des freizustellenden Bereichs

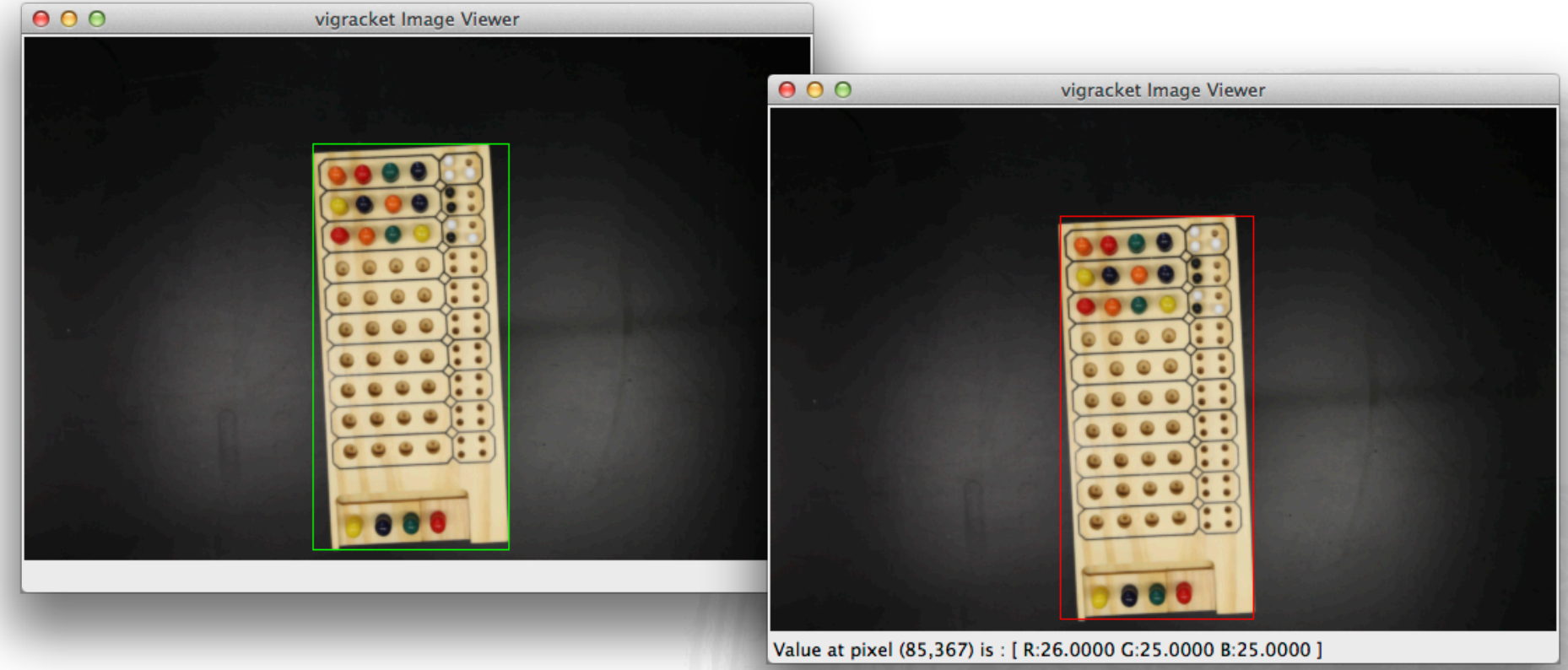
- Idee: Suche nach dem begrenzenden Rechteck um das (gedrehte) Spielfeld herum:

```
;; 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

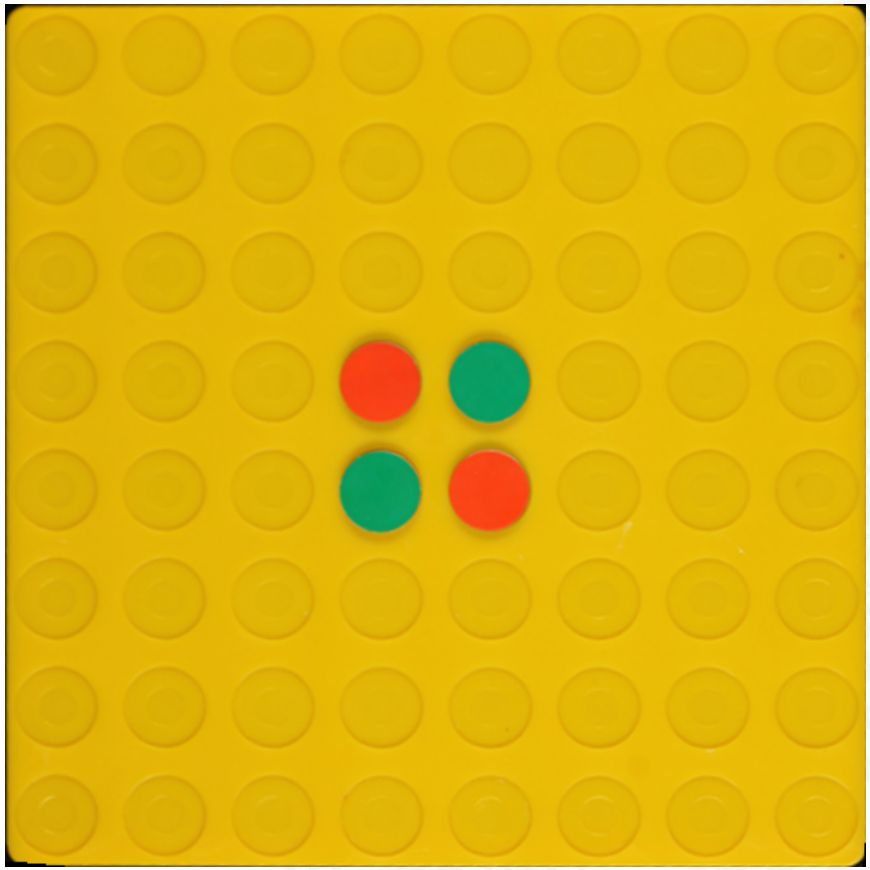
- Funktion subimage verwenden!
- Anwenden auf das (z.B. mit Canny) erzielte Ergebnis:

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

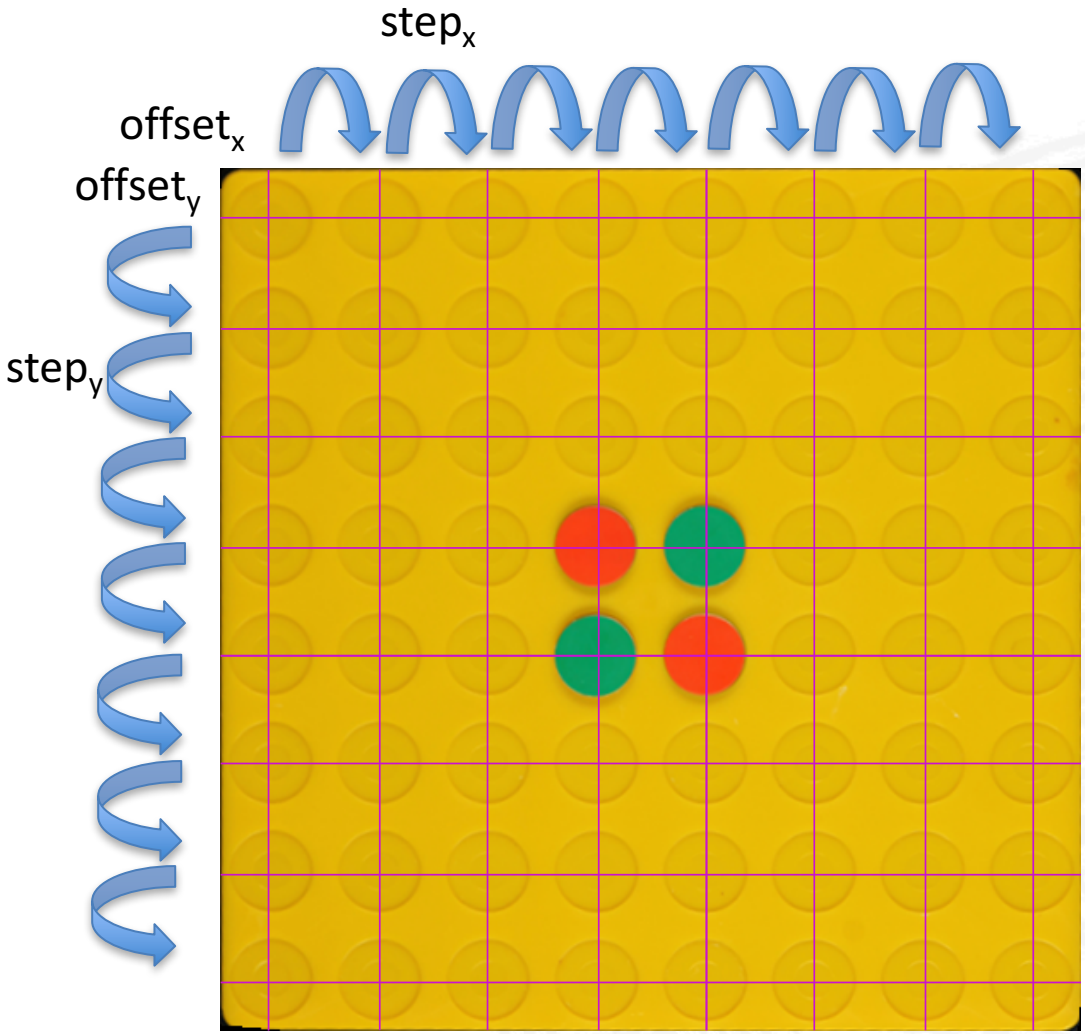
- **Achtung:** Leichte Drehung nach wie vor vorhanden..

Beispiel zur Spielstandsextraktion: Reversi

Ausgangsbild (nach Zuschnitt)

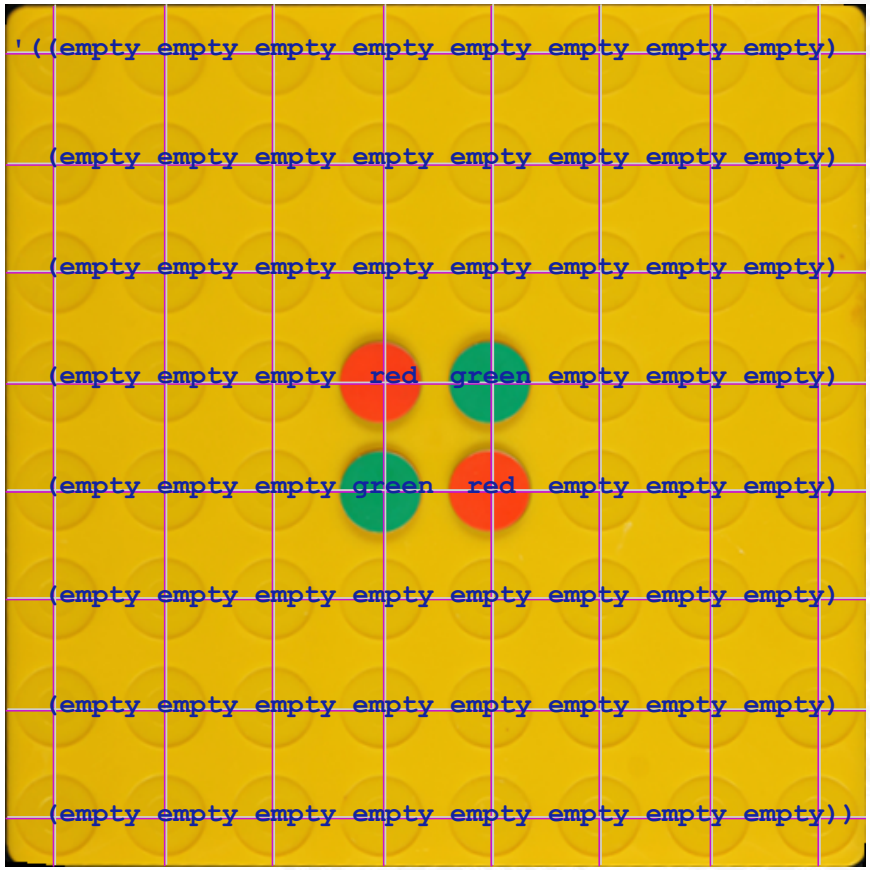


Bestimmung der Spielsteinpositionen



Transfer vom Bild zum Spielzustand

Zum Beispiel durch Farb-Klassifikation:



Ab hier benötigen wir das Bild nicht mehr…

```
'((empty empty empty empty empty empty empty empty)
  (empty empty empty empty empty empty empty empty)
  (empty empty empty empty empty empty empty empty)
  (empty empty empty  red  green empty empty empty)
  (empty empty empty green  red  empty empty empty)
  (empty empty empty empty empty empty empty empty)
  (empty empty empty empty empty empty empty empty)
  (empty empty empty empty empty empty empty empty))
```

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 Auswerten der Bilder!