BILDVERARBEITUNG MIT RACKET – TEIL I

BILDVERARBEITUNGSPRAKTIKUM, SOMMERSEMESTER 2016
UNIVERSITÄT HAMBURG, FACHBEREICH INFORMATIK
LEONIE DRESCHLER-FISCHER, DAVID MOSTELLER UND BENJAMIN SEPPKE

1. AUFGABE: MIT DER ARBEITSUMGEBUNG VERTRAUT MACHEN

EIN BILD LADEN

Damit man ein Bild laden kann, muss man zunächst die VIGRA-Anbindung einbinden.

```
(require vigracket)
```

Anschließend kann man ein Bild unter Angabe des Dateinamens laden.

Es sind zwei Bilder mitgeliefert, wir beginnen heute mit dem Blox-Bild:

```
(define img (load-image
     (build-path vigracket-path "images/blox.gif")))
```

EIN BILD BETRACHTEN

Oft ist es wichtig, ein Bild mit samt seinen Intensitätswerten zu betrachten. Hierzu bietet die Vigracket einen Viewer, den man starten kann:

```
(show-image img „Das gerade geladene Bild“)
```

Lässt man den String weg, erhält das Anzeigefenster keinen Titel.

Eine weitere Möglichkeit ist es, das Bild direkt in ein Racket-Image zu überführen. Dieses kann dann mit dem 2htdp/image-Paket von Racket weiterverarbeitet werden. Wandelt man ein Vigracket- in ein Racket-Bild um, so wird es direkt im Interpreter angezeigt:

```
(image->bitmap img)
```

Möchte man hingegen weiter mit dem Racket-Bild arbeiten, z.B. um Objekte darauf zu zeichnen, so bietet sich eine Symbol-Zuweisung an:

```
(define racket-img (image->bitmap img))
```

SPEICHERN VON BILDERN

Hierfür steht eine Funktion save-image zur Verfügung. Der Dateiname bestimmt dabei auch den Typ des Bildes. Unterstützt werden z.B. JPG, GIF, TIF, PNG und das BMP-Format.

```
(save-image img „...Dateiname...“)
```

2. AUFGABE: EINFACHE FILTEROPERATIONEN

Wir wollen uns nun zwei Filter anschauen, die aufbauend auf Gauß'schen Filterkernen definiert sind. Beide erwarten als Parameter ein Bild und einen weiteren Parameter `scale`, welcher als Gleitkommazahl angegeben werden muss (z.B. `1.0`):

a) Testet, wie sich das Ergebnisbild in Abhängigkeit von dem Parameter `scale` verändert für die Funktion `(gsmooth img scale)`.

b) Probiert z.B. folgende Werte: `0.5`, `1.0`, `2.0`, `3.0`, `5.0` und `10.0`

c) Analog zu a) allerdings mit der Funktion `(ggradient img scale)`

Tipp: Um die Untersuchung zu beschleunigen, müsst ihr keine Bilder zwischenspeichern. Verschachtelte Funktionsaufrufe, wie `(show-image (gsmooth img scale))` funktionieren selbstverständlich auch...

PAUSE

3. AUFGABE: FUNKTIONALE BILDVERARBEITUNG

In der funktionalen Programmierung sind Listen in Zusammenhang mit Funktionen höherer Ordnung wie z.B. `map` von großer Bedeutung. So ein `map` gibt es mit Vigracket auch für Bilder! Es heißt `image-map` und verhält sich ganz ähnlich wie das normale `map`.
Führt `(map + '(1 2 3) (1 2 3))` zu `'(2 4 6)`, so kann man `image-map` ebenfalls dazu verwendenden, Bilder zu addieren, zu subtrahieren etc..

a) Benutzt `image-map` dazu, um ein geglättetes Bild vom ursprünglichen Bild abzuziehen. Probiert dabei unterschiedliche Skalen wie in Aufgabe 2, beobachtet und beschreibt den Effekt, der sich ergibt!

b) `image-map` kann man auch dazu verwenden, um Bilder zu maskieren. Schreibt zunächst Hilfsfunktion, die dann, mit `image-map` auf alle Pixel angewendet, bei einem gegebenen Grauwert-Bild folgendes tut:

> Falls der Intensitätswert < Schwelle:
> setze ihn auf `0.0`,
> sonst:
> setze ihn auf `1.0`.

c) Nummeriert die Objekte, die durch die Schwellwertbildung entstanden sind, mithilfe der Funktion `labelimage`. Diese weist jeder Zusammenhangskomponente im Bild eine Ziffer zu. Verwendet anschließend `image-reduce`, um die Anzahl der Objekte zu zählen.

ZUSATZAUFGABE: BILDTRANSFORMATIONEN

Bilder können nicht nur durch Filter verändert werden, man kann auch die Geometrie von Bildern verändern. Einfache Beispiele für solche Transformationen sind die Rotation sowie die Vergrößerung von Bildern.

d) In vielen Bildverarbeitungsprogrammen wird man beim Vergrößern von Bildern nach der zu verwendenden Interpolationsmethode gefragt. In der Vigracket ist dies ebenfalls der Fall. Probiert verschiedene Interpolationsgrade von `0` bis `2` aus, indem ihr ein Bild auf das doppelte seiner Größe vergrößert.
Benutzt hierfür die Funktion:

```
(resizeimage img neue_breite neue_hoehe interpolations_grad)
```

e) Auch bei der Rotation muss interpoliert werden! Warum eigentlich? Probiert auch hier unterschiedliche Winkel (z.B. `45.0` und `90.0`) und verschiedene Interpolationsgrade von `0` bis `2` aus. Benutzt hierfür die Funktion:

```
(rotateimage img grad interpolations_grad)
```