

## Übungsblatt 3

**Übung 3.1** Wenden Sie auf das bereitgestellte Greyscale-Bild *grey\_a\_poor.png* verschiedene lokale Binarisation-Methoden an (z.B. *sauvola\_threshold* oder *white\_rohrer\_threshold*, siehe Dokumentation “Plugins/Binarization”) und stellen Sie die Unterschiede zum Otsu-Thresholding wie folgt grafisch dar:

- Pixel, die bei Otsu schwarz sind, aber beim anderen nicht werden rot markiert
- Pixel, die bei Otsu weiß sind, aber beim anderen schwarz werden grün markiert

Die Markierung erfolgt durch die *highlight* Methode, z.B.

```
rgb = img.to_rgb()
rgb.highlight(other, RGBPixel(255,0,0))
```

Die Pixel, die nur bei einem Bild schwarz sind, erhalten Sie mittels Differenzbildung (*subtract\_images*).

**Übung 3.2** Bestimmen Sie in den bereitgestellten Bildern die häufigste schwarze und weiße vertikale Lauflänge (siehe Dokumentation Abschnitt “Plugins/Runlength”) und markieren Sie alle Lauflängen der jeweils häufigsten Lauflänge rot bzw. grün.

Anleitung: Alle schwarzen vertikalen Lauflängen der Länge  $n$  erhalten Sie, indem Sie von einem Bild alle Lauflängen größer  $n$  (*filter\_tall\_runs*) und alle kleiner  $n$  (*filter\_short\_runs*) abziehen. Achtung: diese Funktionen arbeiten direkt auf dem Bild. Sie müssen dies also auf einer Kopie des Bildes durchführen, die Sie mit *image\_copy* erzeugen können.

Die weißen Lauflängen der Länge  $n$  erhalten Sie, indem Sie dieselben Operationen auf einem invertierten Bild (*invert*) durchführen.

**Übung 3.3** Führen Sie eine *cc\_analysis* auf einem Bild durch und geben Sie alle verwendeten Labels in sortierter Reihenfolge aus. Alle Labels können Sie elegant über eine “List-Comprehension” auslesen:

```
labels = [c.label for c in ccs]
```

Sind die Labels konsekutiv oder gibt es Lücken?