

Genauso kann natürlich über den durch das ICC-Profil definierten Farbraum ein sRGB-Pixel in den Farbraum des Scanners oder des Monitors umgerechnet werden.

12.4 Statistiken von Farbbildern

12.4.1 Wie viele Farben enthält ein Bild?

Ein kleines aber häufiges Teilproblem im Zusammenhang mit Farbbildern besteht darin, zu ermitteln, wie viele unterschiedliche Farben in einem Bild überhaupt enthalten sind. Natürlich könnte man dafür ein Histogramm-Array mit einem Integer-Element für jede Farbe anlegen, dieses befüllen und anschließend abzählen, wie viele Histogrammzellen mindestens den Wert 1 enthalten. Da ein 8-Bit-RGB-Farbbild potenziell $2^{24} = 16.777.216$ Farbwerte enthalten kann, wäre ein solches Histogramm-Array (mit immerhin 64 MBytes) in den meisten Fällen aber wesentlich größer als das ursprüngliche Bild selbst!

Eine einfachere Lösung besteht darin, die Farbwerte im Pixel-Array des Bilds zu *sortieren*, sodass alle gleichen Farbwerte beisammen liegen. Die Sortierreihenfolge ist dabei natürlich unwesentlich. Die Zahl der zusammenhängenden Farblöcke entspricht der Anzahl der Farben im Bild. Diese kann, wie in Prog. 12.12 gezeigt, einfach durch Abzählen der Übergänge zwischen den Farblöcken berechnet werden.

Natürlich wird in diesem Fall nicht das ursprüngliche Pixel-Array sortiert (das würde das Bild verändern), sondern eine Kopie des Pixel-Arrays.²² Diese Kopie wird mit der eigenen Methode `duplicateArray()` erzeugt, mit der beliebige, eindimensionale Java-Arrays dupliziert werden können.²³ Das Sortieren erfolgt in Prog. 12.12 (Zeile 4) mithilfe der Java-Systemmethode `Arrays.sort()`, die sehr effizient implementiert ist.

12.4.2 Histogramme

Histogramme von Farbbildern waren bereits in Abschn. 4.5 ein Thema, wobei wir uns auf die eindimensionalen Verteilungen der einzelnen Farbkanaäle bzw. der Intensitätswerte beschränkt haben. Auch die ImageJ-Methode `getHistogram()` berechnet bei Anwendung auf Objekte der Klasse `ColorProcessor` in der Form

```
ColorProcessor cp;
int[] H = cp.getHistogram();
```

²² Alternativ könnte man auch mit der Methode `duplicate()` eine Kopie des `ImageProcessor`-Objekts anlegen.

²³ Das Duplizieren von Java-Arrays ist auch mit der Standardmethode `clone()` möglich, da die `Array`-Klasse das `Cloneable`-Interface implementiert.