

$$H(i, j) = \begin{bmatrix} 0.075 & 0.125 & 0.075 \\ 0.125 & \underline{0.200} & 0.125 \\ 0.075 & 0.125 & 0.075 \end{bmatrix}.$$

In der folgenden `run()`-Methode eines ImageJ-Plugins wird zunächst die Filtermatrix als eindimensionales `float`-Array definiert (man beachte die Form der `float`-Konstanten „0.075f“ usw.), dann wird in Zeile 8 ein neues `Convolver`-Objekt angelegt.

```

1  import ij.plugin.filter.Convolver;
2  ...
3  public void run(ImageProcessor I) {
4      float[] H = {
5          0.075f, 0.125f, 0.075f,
6          0.125f, 0.200f, 0.125f,
7          0.075f, 0.125f, 0.075f };
8      Convolver cv = new Convolver();
9      cv.setNormalize(false); // turn off filter normalization
10     cv.convolve(I, H, 3, 3); // do the filter operation
11 }

```

Die Methode `convolve()` in Zeile 10 benötigt für die Filteroperation neben dem Bild `I` und der Filtermatrix `H` selbst auch deren Breite und Höhe (weil `H` ein eindimensionales Array ist). Das Bild `I` wird durch die Filteroperation modifiziert.

In diesem Fall hätte man auch die nicht normalisierte ganzzahlige Filtermatrix in Gl. 6.10 verwenden können, denn `convolve()` normalisiert das übergebene Filter automatisch (nach `cv.setNormalize(true)`);).

### 6.6.2 Gauß-Filter

In der ImageJ-Klasse `ij.plugin.filter.GaussianBlur` ist ein einfaches Gauß-Filter implementiert, dessen Radius ( $\sigma$ ) frei spezifiziert werden kann. Dieses Gauß-Filter ist natürlich mit separierten Filterkernen implementiert (s. Abschn. 6.3.3).<sup>3</sup> Hier ist ein Beispiel für dessen Anwendung:

```

1  import ij.plugin.filter.GaussianBlur;
2  ...
3  public void run(ImageProcessor I) {
4      GaussianBlur gb = new GaussianBlur();
5      double radius = 2.5;
6      gb.blur(I, radius);
7  }

```

<sup>3</sup> Zur Implementierung in ImageJ ist anzumerken, dass die in der Methode `blur()` generierten Filterkerne relativ zum angegebenen Radius zu klein dimensioniert werden, und es dadurch zu erheblichen Fehlern kommt.