5. Vorlesung Mikroskopische Bildverarbeitung

Wahlpflichtmodul 9521: EI-M im 1. und 3. Fachsemester

Initialisierung

18. Dezember 2015

Wiederholung aus der 4. Vorlesung

Aus 13. Weitere Einsatzfelder linearer diskreter verschiebungsinvarianter Filter

Automatisierte analytische Berechnung der 1D-DFT

```
Clear[transferfunktion];
transferfunktion[g_List] := Module[{resultat, u, mm, m},
   resultat = (FullSimplify[Sum[g[[Mod[m + (Length[g] - 1) / 2, Length[g]] + 1]] *
         Exp[-2 π i * Join[Table[i, {i, 0, (Length[g] - 1) / 2, 1}], Table[mm - i, {i, (Length[g] - 1) / 2, 1, -1}]][[m + 1]] u / mm],
        {m, 0, Length[g] - 1, 1}], Assumptions → Element[u, Integers]] /. u / mm → k1);
   Return[resultat];
  ];
```

Automatisierte analytische Berechnung der 2D-DFT

```
Clear[transferfunktion2d];
(*schnellere Variante -(M-1)/2 ... (M-1)/2 und -(N-1)/2 ... (N-1)/2*)
transferfunktion2d[g_List] := Module[{resultat, u, v, mm, nn, m, n},
   resultat =
    ((
       FullSimplify[
         FullSimplify[
          Sum[
           Sum[
            g[[m + (Dimensions[g][[1]] + 1) / 2, n + (Dimensions[g][[2]] + 1) / 2]] *
             Exp[-2 π i m * u / mm] *
             Exp[-2 π i n * v / nn],
            {m, -(Dimensions[g][[1]] - 1) / 2, (Dimensions[g][[1]] - 1) / 2, 1}],
           {n, -(Dimensions[g][[2]] - 1) / 2, (Dimensions[g][[2]] - 1) / 2, 1}],
          Assumptions → Element[u, Integers]],
         Assumptions → Element[v, Integers]]
        /. u / mm → k1)
      /. v / nn → k2
    );
   Return[FullSimplify[ComplexExpand[ExpToTrig[resultat]] /. u / mm → k1 /. v / nn → k2]];
  ];
```

Zur Visualisierung von 1D- Filtern

```
Clear[plottransferfunktion];
plottransferfunktion[formel_] := Module[{},
   GraphicsGrid[{{

      Plot[Evaluate[ComplexExpand[Re[formel]]], {k1, 0, kmax},
       AxesLabel -> {k1, None}, PlotRange → {Full, {-1, 1}}, PlotLabel → "Realteil", PlotPoints → r],

      Plot[Evaluate[ComplexExpand[Im[formel]]], {k1, 0, kmax},
       AxesLabel -> {k1, None}, PlotRange → {Full, {-1, 1}}, PlotLabel → "Imaginärteil", PlotPoints → r],

      Plot[Evaluate[ArcTan[ComplexExpand[Im[formel] / (ComplexExpand[Re[formel]] + $MachineEpsilon)]]],
       {k1, 0, kmax}, AxesLabel -> {k1, None}, PlotRange → {Full, {-π / 2, π / 2}}, PlotLabel → "Phasenwinkel",
       PlotPoints → r, Ticks → {Automatic, (Range[#] - (# + 1) / 2) / ((# - 1) / 2) / 2 * π &[5]}]

     }}]
  ];
```

Zur Visualisierung von 2D- Filtern

```
Clear[plottransferfunktion2d];
plottransferfunktion2d[formel_] := Module[{},
   GraphicsGrid[{{

      Plot3D[Evaluate[ComplexExpand[Re[formel]]], {k1, -kmax, kmax}, {k2, -kmax, kmax},
       AxesLabel -> {k1, k2, None}, PlotRange → {Full, Full, {-1, 1}}, PlotLabel → "Realteil", Mesh → 19, RotationAction → "Clip"],

      Plot3D[Evaluate[ComplexExpand[Im[formel]]], {k1, -kmax, kmax}, {k2, -kmax, kmax}, AxesLabel -> {k1, k2, None},
       PlotRange → {Full, Full, {-1, 1}}, PlotLabel → "Imaginärteil", Mesh → 19, RotationAction → "Clip"],

      Plot3D[Evaluate[ArcTan[ComplexExpand[Im[formel] / (ComplexExpand[Re[formel]] + $MachineEpsilon)]]], {k1, -kmax, kmax},
       {k2, -kmax, kmax}, AxesLabel -> {k1, k2, None}, PlotRange → {Full, Full, {-π / 2, π / 2}}, PlotLabel → "Phasenwinkel",
       Ticks → {Automatic, Automatic, (Range[#] - (# + 1) / 2) / ((# - 1) / 2) / 2 * π &[3]}, Mesh → 19, RotationAction → "Clip"]

     }}]
  ];
```

Bandpaß als Linearkombination von Filtern, zum Beispiel als Differenz von Binomialfiltern 2. und 4. Ordnung

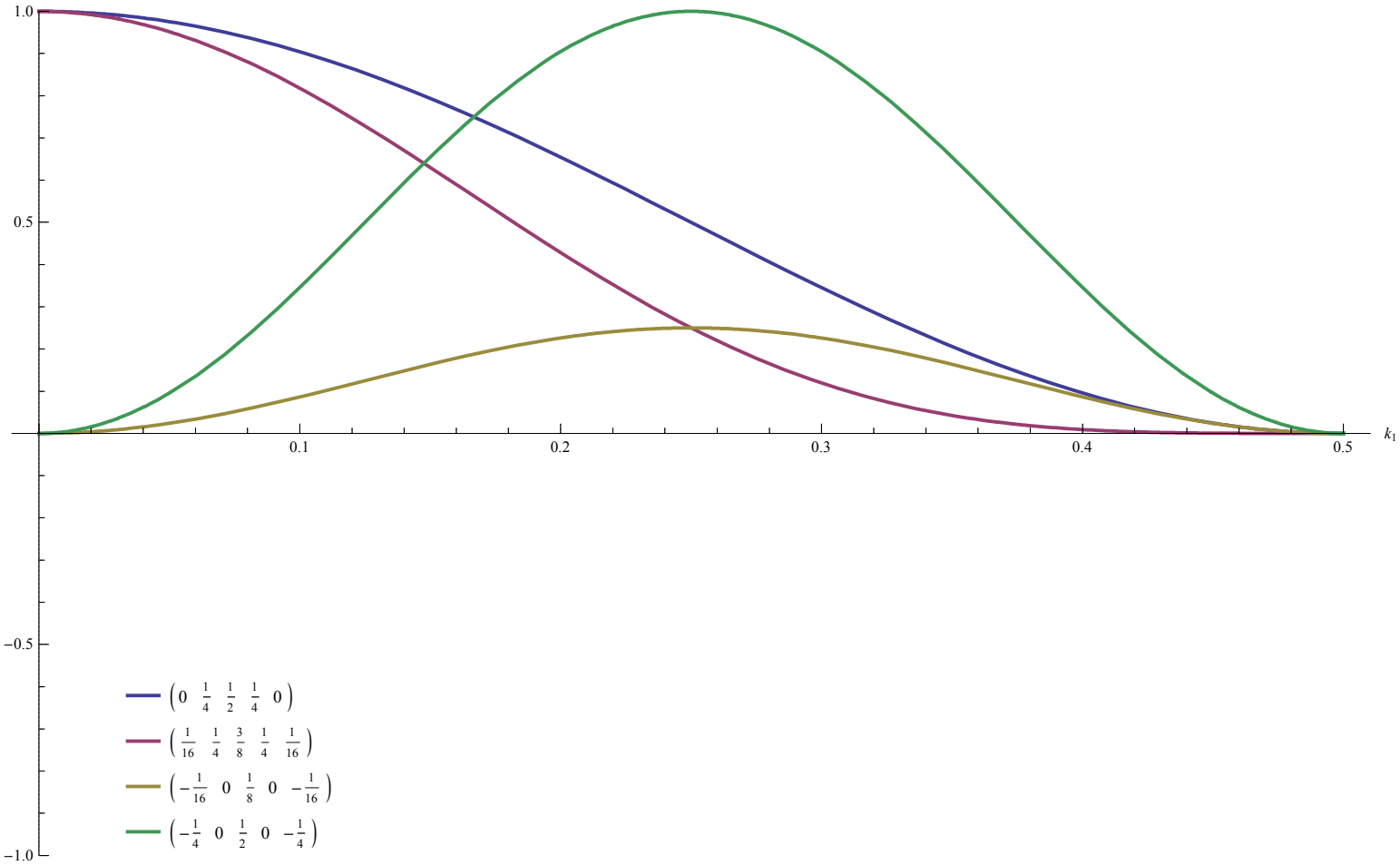
```
Show[(Print[TableForm@TrigReduce[#[[1]]]];
    Plot[Evaluate[#[[1]]], {k1, 0, kmax}, AxesLabel -> {k1, None}, PlotRange → {Full, {-1, 1}},
     PlotStyle → Thick, PlotLegends → Placed[(MatrixForm[#] & /@ #[[2]]), {{0, 0}, {-0.5, 0.0}}]]) &[
  Transpose[{transferfunktion[#], MatrixForm[{#}]} & /@ {ListConvolve[binom[2], identität[5], 2], binom[4],
    ListConvolve[binom[2], identität[5], 2] - binom[4], (ListConvolve[binom[2], identität[5], 2] - binom[4]) /
     First[Maximize[{(transferfunktion[binom[2]] - transferfunktion[binom[4]]), 0 ≤ k1 ≤ 1 / 2}, k1]]}]], ImageSize → pagewidth]
```

$\frac{1}{2}\,(1 + \text{Cos}[2\,\pi\,k_1])$

$\frac{1}{8}\,(3 + 4\,\text{Cos}[2\,\pi\,k_1] + \text{Cos}[4\,\pi\,k_1])$

$\frac{1}{8}\,(1 - \text{Cos}[4\,\pi\,k_1])$

$\frac{1}{2}\,(1 - \text{Cos}[4\,\pi\,k_1])$



transferfunktion[binom[2]] - transferfunktion[binom[4]]

$$\text{Cos}[\pi\, k_1]^2 - \text{Cos}[\pi\, k_1]^4$$

transferfunktion[binom[2]] - transferfunktion[binom[4]] // TrigReduce

$$\frac{1}{8}\,(1 - \text{Cos}[4\,\pi\, k_1])$$

transferfunktion[binom[2]] - transferfunktion[binom[4]] // FullSimplify

$$\frac{1}{4}\,\text{Sin}[2\,\pi\, k_1]^2$$

```
Maximize[{(transferfunktion[binom[2]] - transferfunktion[binom[4]]), 0 ≤ k1 ≤ 1 / 2}, k1]
```

$$\left\{\frac{1}{4}, \left\{k_1 \to \frac{1}{4}\right\}\right\}$$

```
(transferfunktion[binom[2]] - transferfunktion[binom[4]]) /
 First[Maximize[{(transferfunktion[binom[2]] - transferfunktion[binom[4]]), 0 ≤ k1 ≤ 1 / 2}, k1]]
```

$$4\left(\text{Cos}[\pi\, k_1]^2 - \text{Cos}[\pi\, k_1]^4\right)$$

```
(transferfunktion[binom[2]] - transferfunktion[binom[4]]) /
  First[Maximize[{(transferfunktion[binom[2]] - transferfunktion[binom[4]]), 0 ≤ k1 ≤ 1 / 2}, k1]] // TrigReduce
```

$$\frac{1}{2}\left(1 - \text{Cos}[4\,\pi\, k_1]\right)$$

Die Transferfunktion dieses 2D-Bandpasses:

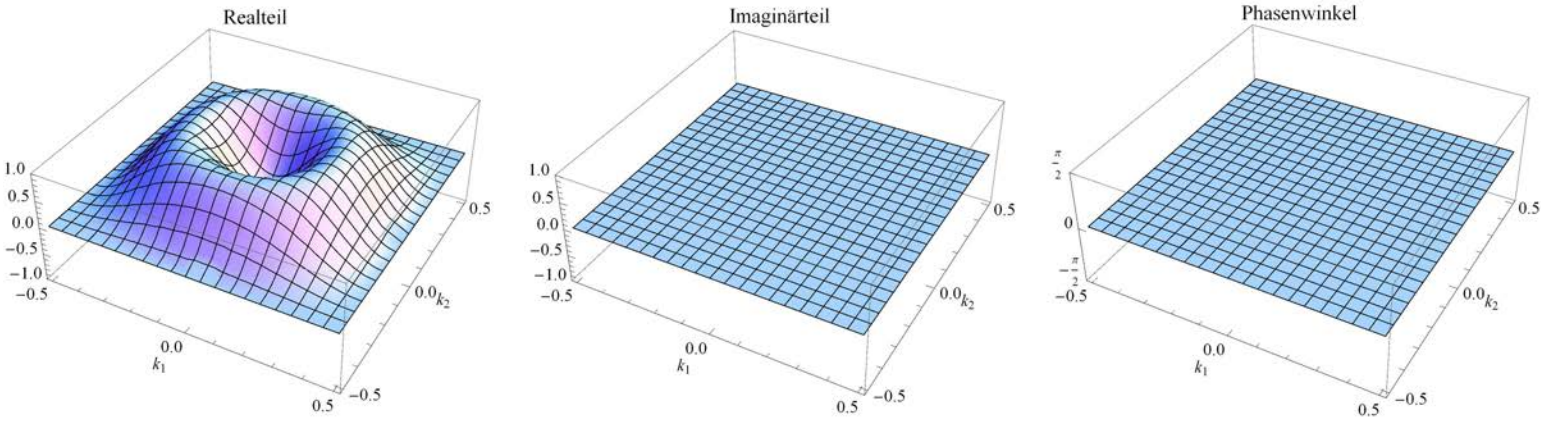
```
bandpaßkern = 4 *
    (ListConvolve[Transpose[{binom[2]}].{binom[2]}, Transpose[{identität[5]}].{identität[5]}, {{2, 2}}] - Transpose[{binom[4]}].{binom[4]});
```

```
bandpaßkern // MatrixForm
```

$$\begin{pmatrix} -\frac{1}{64} & -\frac{1}{16} & -\frac{3}{32} & -\frac{1}{16} & -\frac{1}{64} \\ -\frac{1}{16} & 0 & \frac{1}{8} & 0 & -\frac{1}{16} \\ -\frac{3}{32} & \frac{1}{8} & \frac{7}{16} & \frac{1}{8} & -\frac{3}{32} \\ -\frac{1}{16} & 0 & \frac{1}{8} & 0 & -\frac{1}{16} \\ -\frac{1}{64} & -\frac{1}{16} & -\frac{3}{32} & -\frac{1}{16} & -\frac{1}{64} \end{pmatrix}$$

```
transferfunktion2d[bandpaßkern]
Show[plottransferfunktion2d[transferfunktion2d[bandpaßkern]], ImageSize → pagewidth]
```

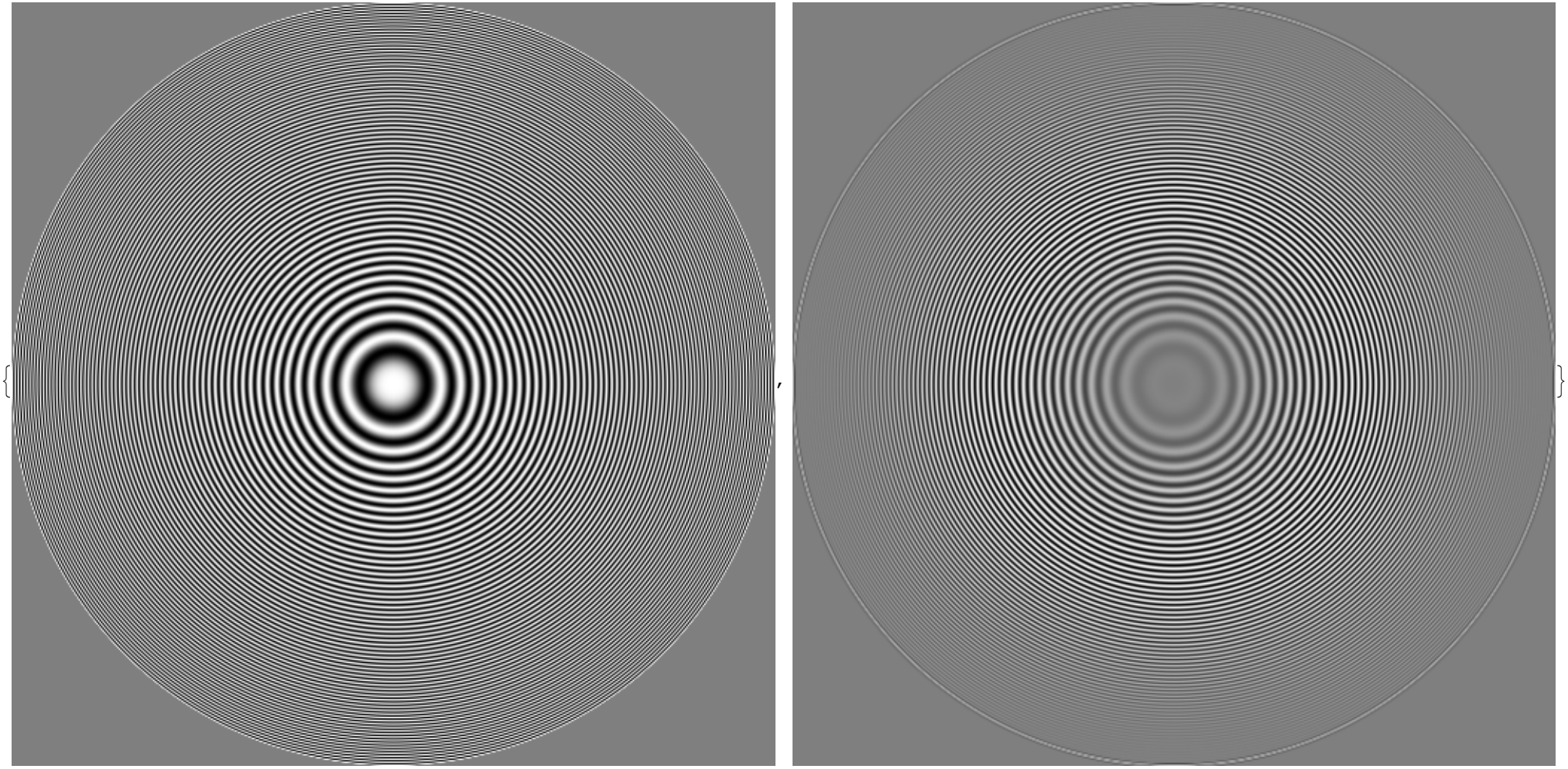
$$-\frac{1}{2}\,\text{Cos}[\pi\,k_1]^2\,\text{Cos}[\pi\,k_2]^2\,(-6+2\,\text{Cos}[2\,\pi\,k_1]+\text{Cos}[2\,\pi\,(k_1-k_2)]+2\,\text{Cos}[2\,\pi\,k_2]+\text{Cos}[2\,\pi\,(k_1+k_2)])$$



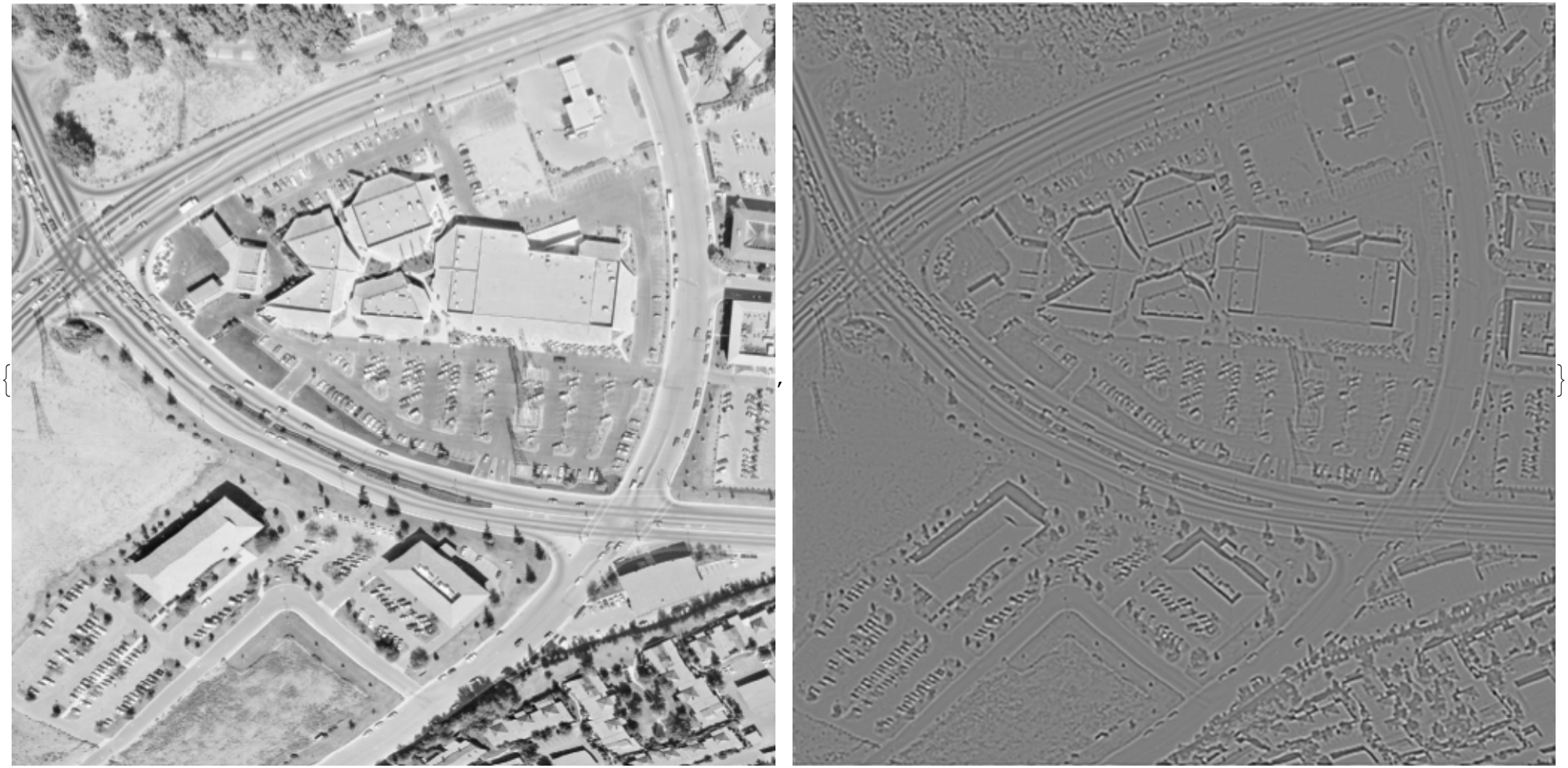

```
FullSimplify[TrigToExp[transferfunktion2d[bandpaßkern]]] ==
 FullSimplify[TrigToExp[4 (transferfunktion2d[Transpose[{binom[2]}].{binom[2]}] - transferfunktion2d[Transpose[{binom[4]}].{binom[4]}])]]
```

```
True
```

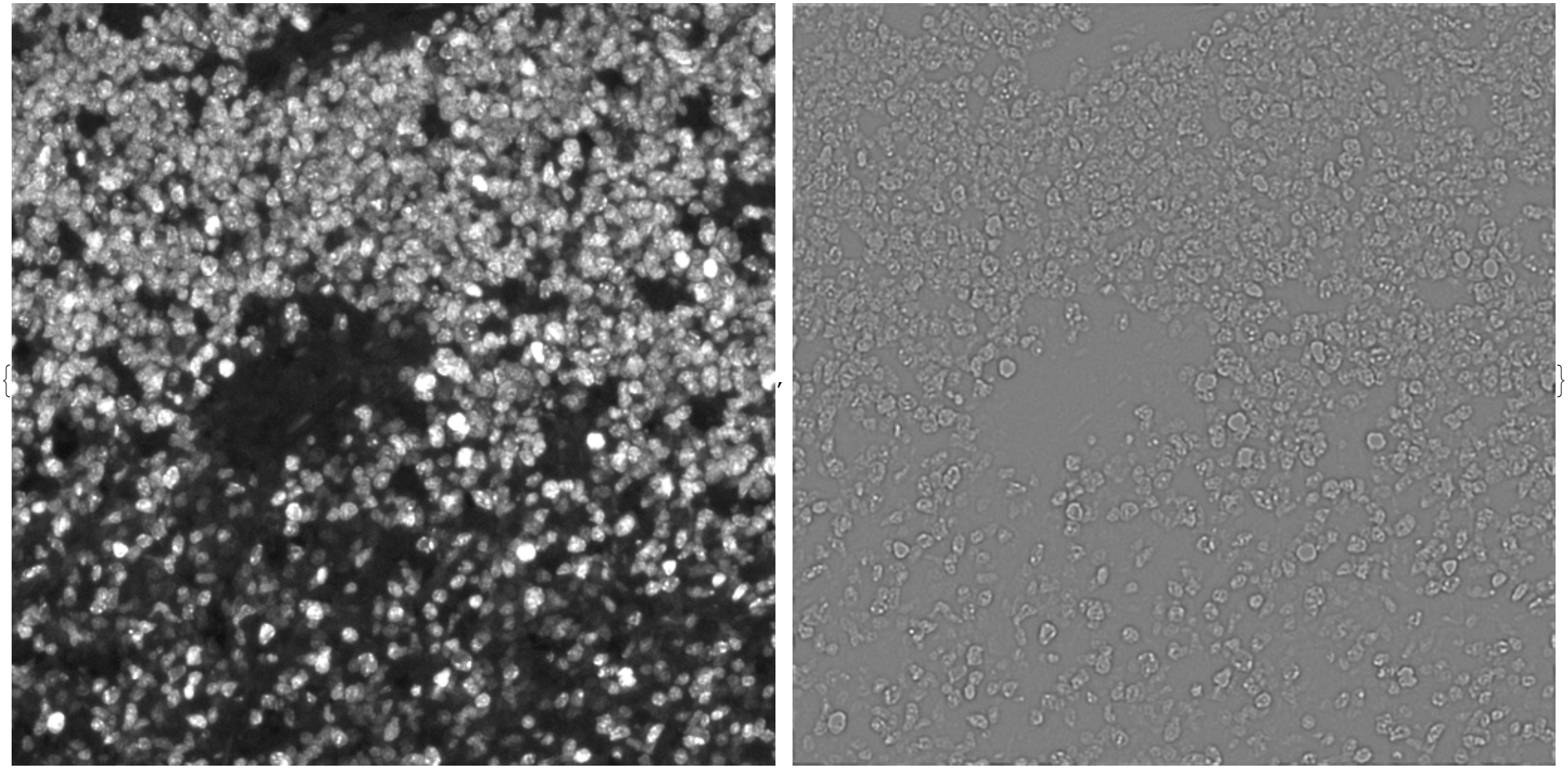
```
{Show[Image[#2]], Show[Image[0.5 + ListConvolve[#1, #2, {(Dimensions[#1] + 1) / 2}]]]} &[bandpaßkern, ImageData[wellenbild]]
```



```
{Show[Image[#2]], Show[Image[0.5 + ListConvolve[#1, #2, {(Dimensions[#1] + 1) / 2}]]]} &[
 bandpaßkern, ImageData[ExampleData[{"TestImage", "Aerial2"}]]]
```



```
{Show[Image[#2]], Show[Image[0.5 + ListConvolve[#1, #2, {(Dimensions[#1] + 1) / 2}]]]} &[
 bandpaßkern, ImageData[First@ColorSeparate[Ton3CD21dreikanalausgleichF2]]]
```



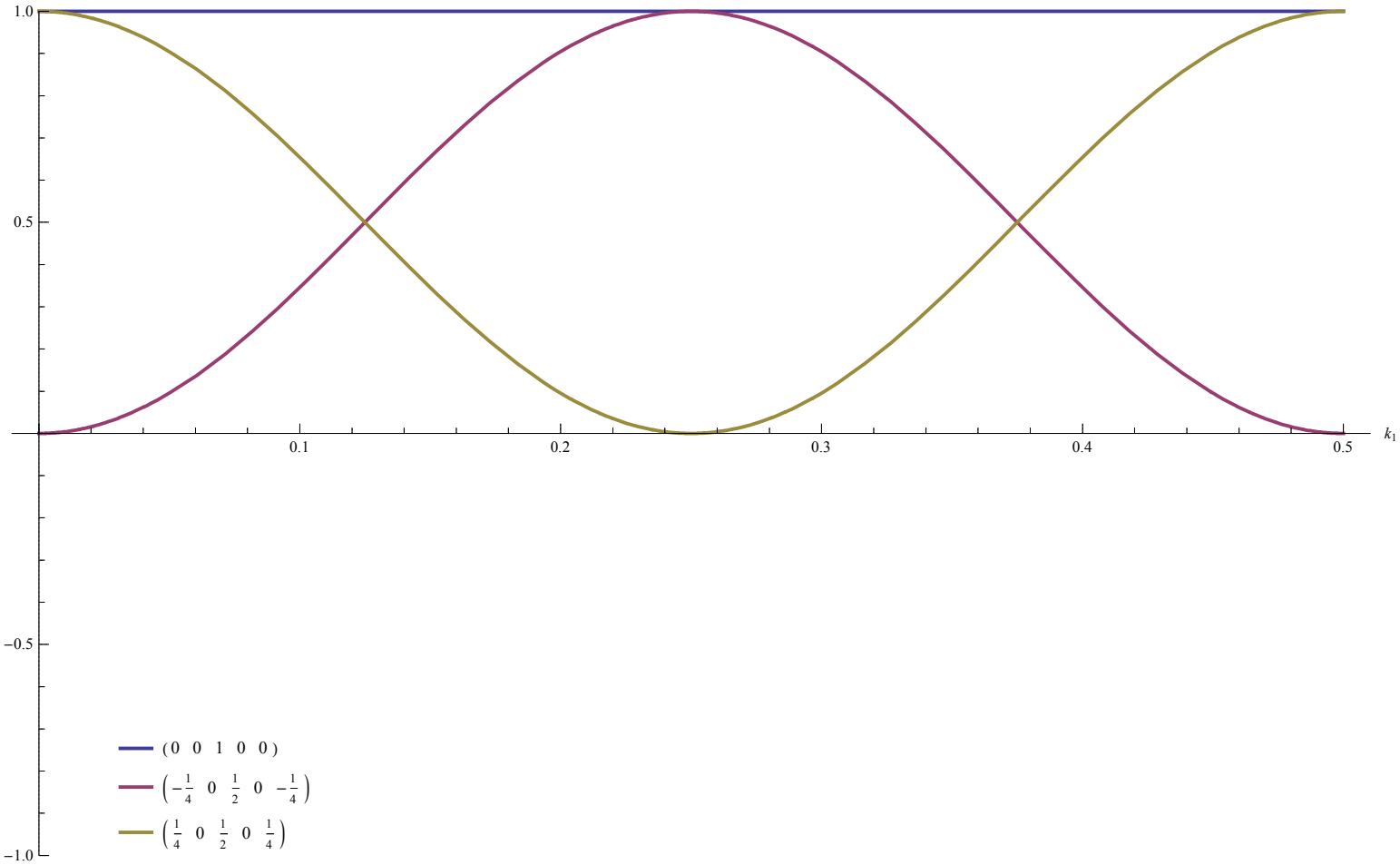
Bandsperre als Linearkombination von Filtern, zum Beispiel als Differenz von Identitätsoperator und Bandpaß

```
Show[(Print[TableForm@TrigReduce[#[[1]]]];
    Plot[Evaluate[#[[1]]], {k1, 0, kmax}, AxesLabel -> {k1, None}, PlotRange → {Full, {-1, 1}}, PlotStyle → Thick,
     PlotLegends → Placed[(MatrixForm[#] & /@ #[[2]]), {{0, 0}, {-0.5, 0.0}}]]) &[
  Transpose[{transferfunktion[#], MatrixForm[{#}]} & /@ {identität[5], (ListConvolve[binom[2], identität[5], 2] - binom[4]) /
       First[Maximize[{(transferfunktion[binom[2]] - transferfunktion[binom[4]]), 0 ≤ k1 ≤ 1 / 2}, k1]],
     identität[5] - (ListConvolve[binom[2], identität[5], 2] - binom[4]) /
       First[Maximize[{(transferfunktion[binom[2]] - transferfunktion[binom[4]]), 0 ≤ k1 ≤ 1 / 2}, k1]]}]], ImageSize → pagewidth]
```

1

$\frac{1}{2}(1 - \text{Cos}[4\,\pi\,k_1])$

$\frac{1}{2}(1 + \text{Cos}[4\,\pi\,k_1])$




```
transferfunktion[identität[3]] - (transferfunktion[binom[2]] - transferfunktion[binom[4]]) /
  First[Maximize[{(transferfunktion[binom[2]] - transferfunktion[binom[4]]), 0 ≤ k1 ≤ 1 / 2}, k1]]
```

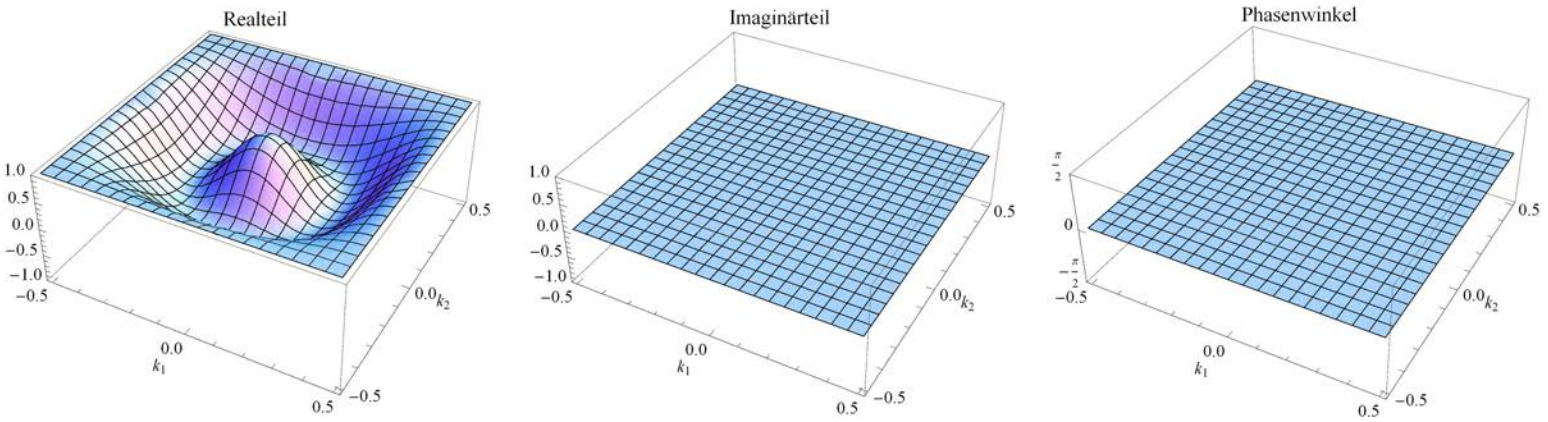
$$1 - 4\left(\text{Cos}[\pi\, k_1]^2 - \text{Cos}[\pi\, k_1]^4\right)$$

```
bandsperrkern = Transpose[{identität[5]}].{identität[5]} - bandpaßkern;
```

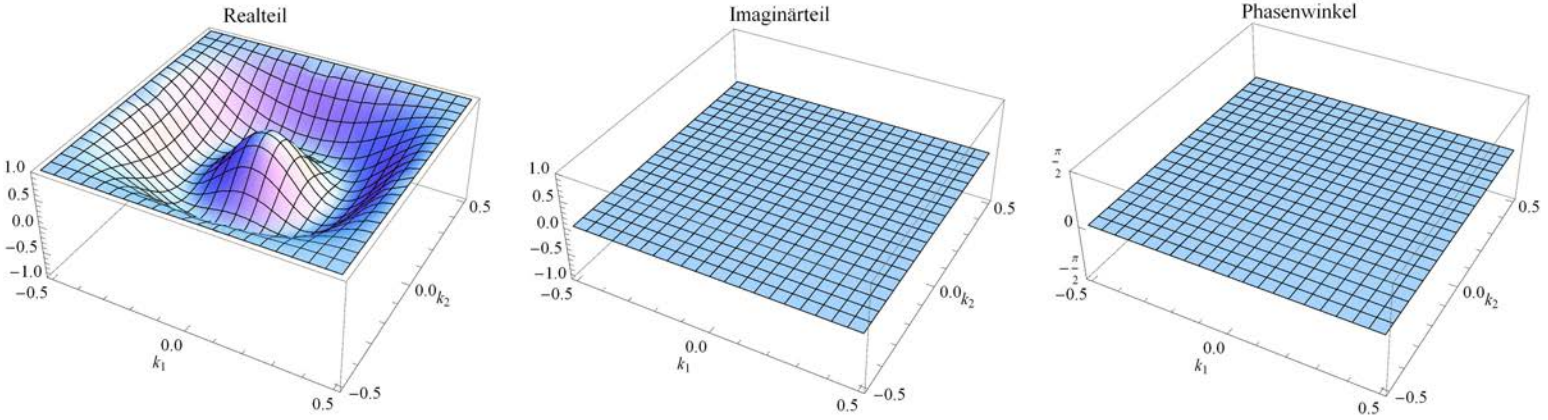
```
bandsperrkern // MatrixForm
```

$$\begin{pmatrix} \frac{1}{64} & \frac{1}{16} & \frac{3}{32} & \frac{1}{16} & \frac{1}{64} \\ \frac{1}{16} & 0 & -\frac{1}{8} & 0 & \frac{1}{16} \\ \frac{3}{32} & -\frac{1}{8} & \frac{9}{16} & -\frac{1}{8} & \frac{3}{32} \\ \frac{1}{16} & 0 & -\frac{1}{8} & 0 & \frac{1}{16} \\ \frac{1}{64} & \frac{1}{16} & \frac{3}{32} & \frac{1}{16} & \frac{1}{64} \end{pmatrix}$$

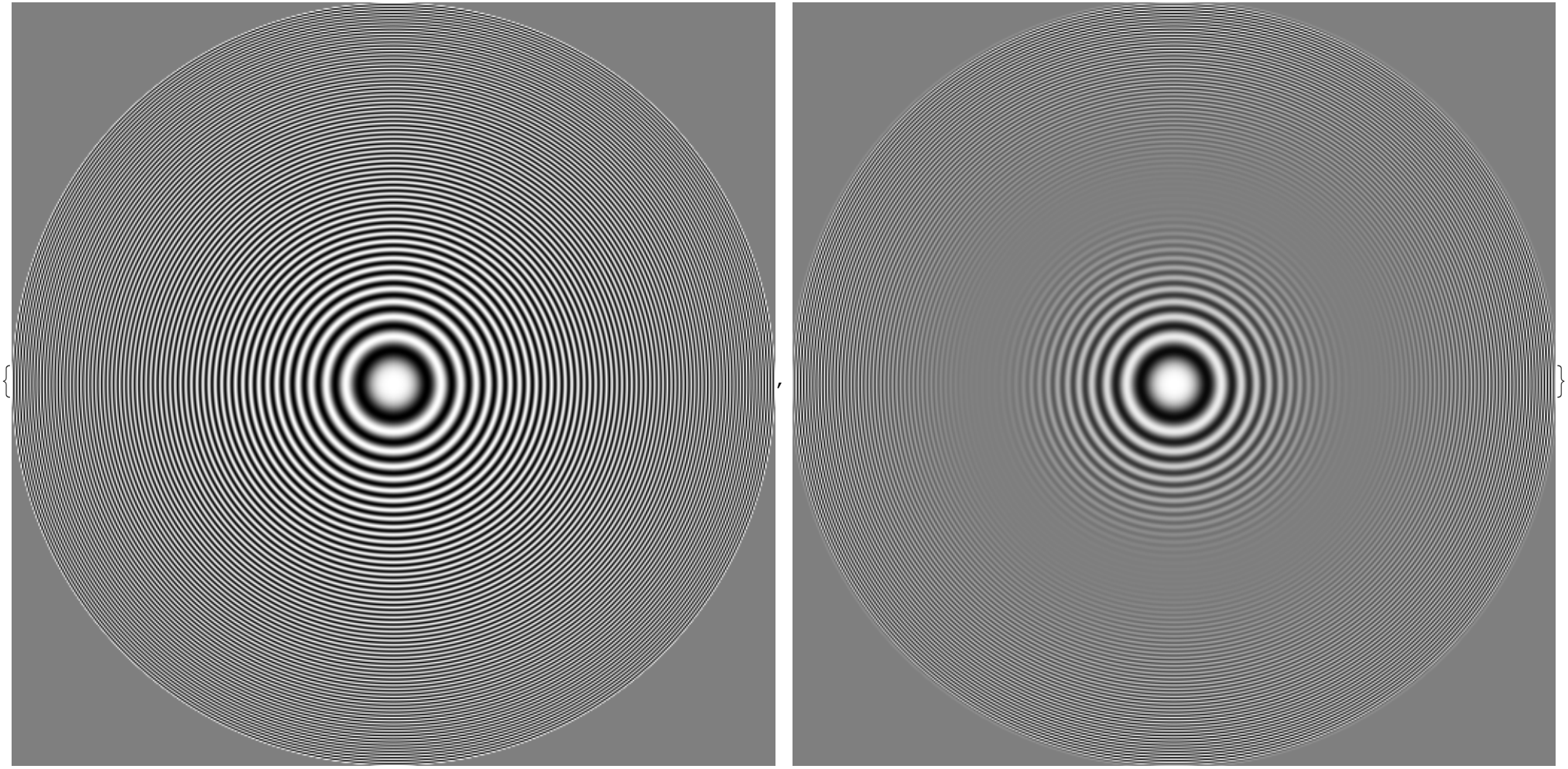
```
Show[plottransferfunktion2d[transferfunktion2d[bandsperrkern]], ImageSize → pagewidth]
```




```
Show[plottransferfunktion2d[transferfunktion2d[Transpose[{identität[5]}].{identität[5]}] - 4 *
    (transferfunktion2d[Transpose[{binom[2]}].{binom[2]}] - transferfunktion2d[Transpose[{binom[4]}].{binom[4]}])], ImageSize → pagewidth]
```

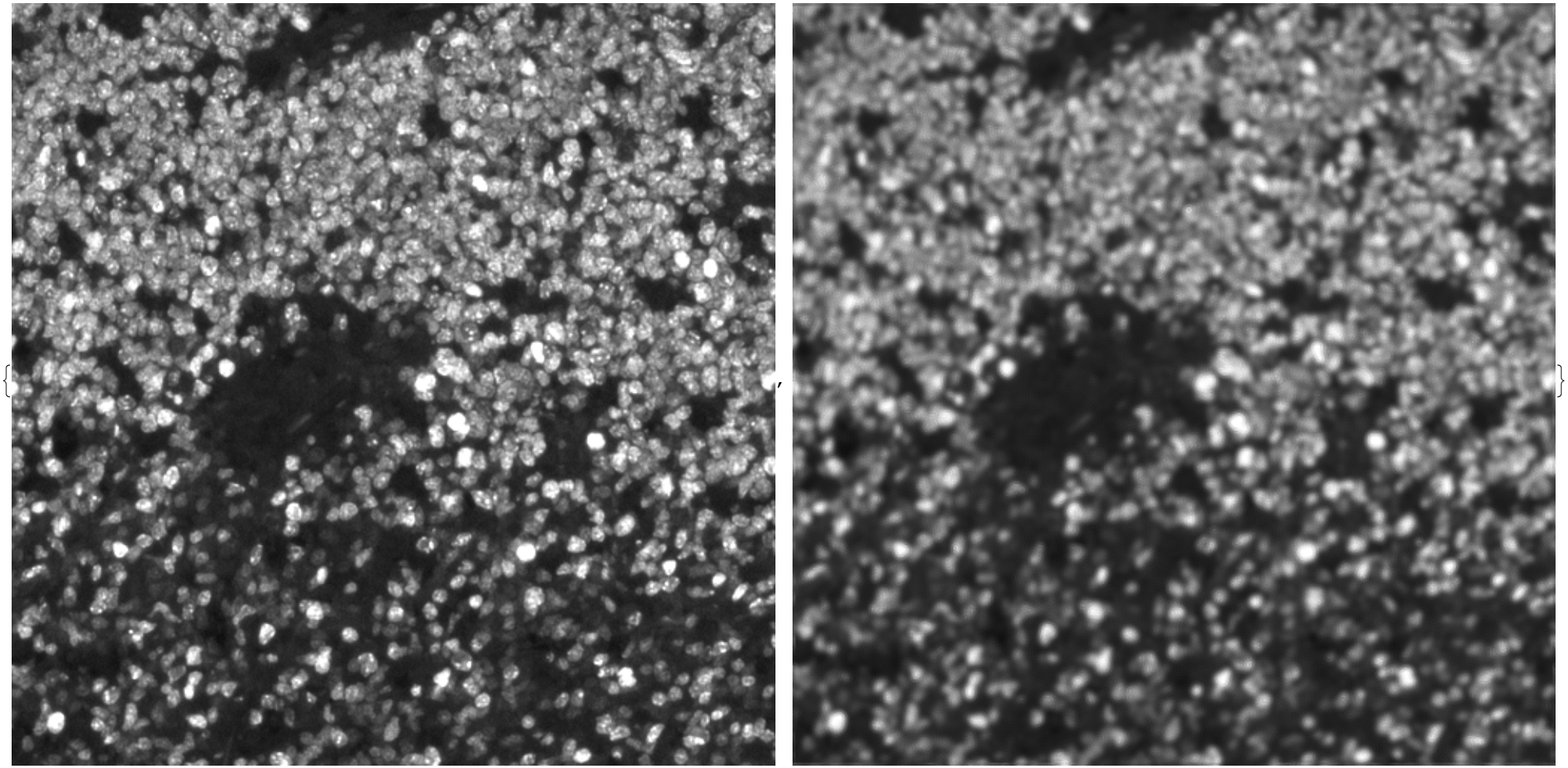


```
{Show[Image[#2]], Show[Image[ListConvolve[#1, #2, {(Dimensions[#1] + 1) / 2}]]]} &[bandsperrkern, ImageData[wellenbild]]
```



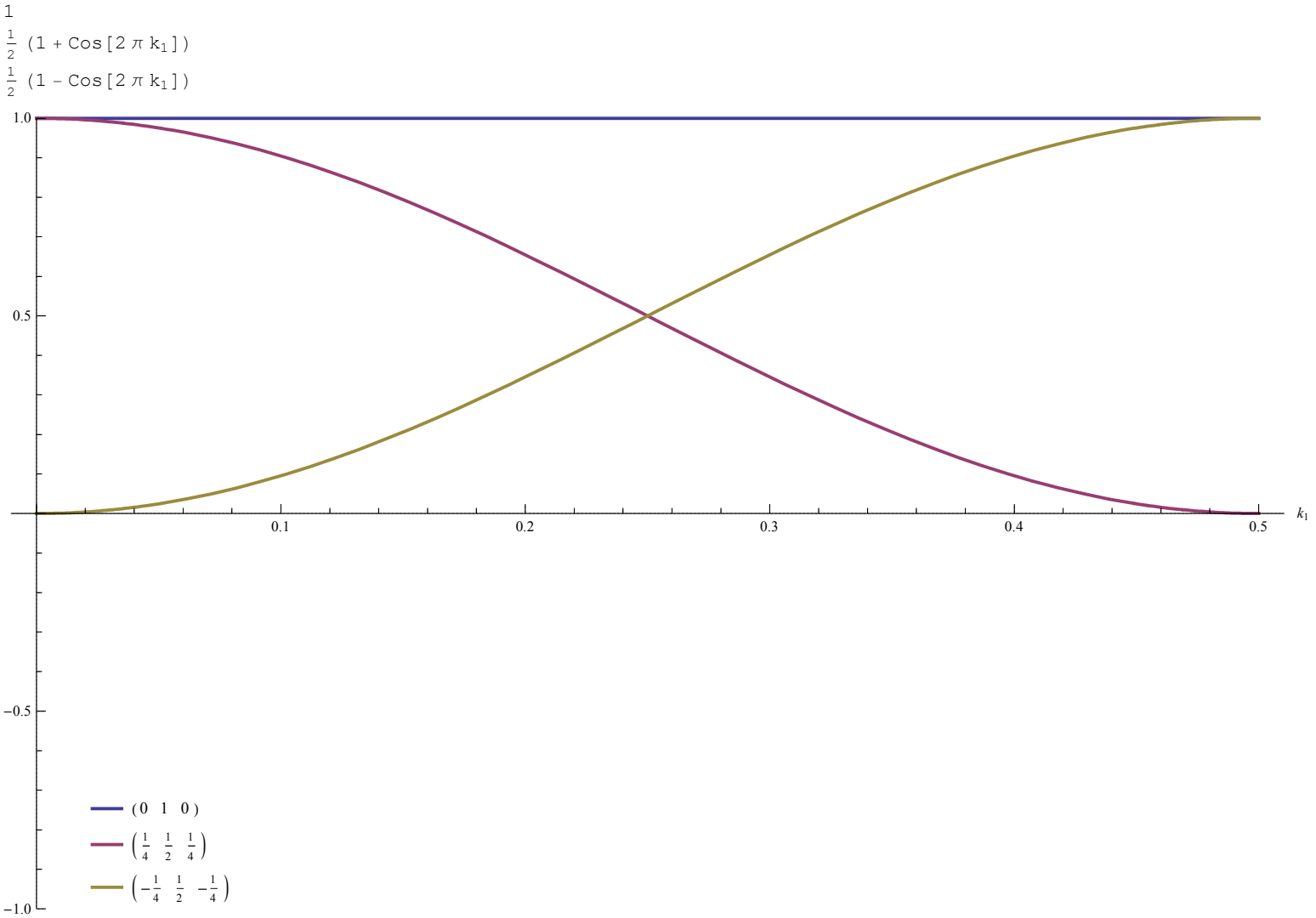
```
{Show[Image[#2]], Show[Image[ListConvolve[#1, #2, {(Dimensions[#1] + 1) / 2}]]]} &[
 bandsperrkern, ImageData[ExampleData[{"TestImage", "Aerial2"}]]]
```

```
{Show[Image[#2]], Show[Image[ListConvolve[#1, #2, {(Dimensions[#1] + 1) / 2}]]]} &[
 bandsperrkern, ImageData[First@ColorSeparate[Ton3CD21dreikanalausgleichF2]]]
```



Hochpaß als Linearkombination von Filtern, zum Beispiel als Differenz von Identitätsoperator und Binomialfilter 4. Ordnung

```
Show[(Print[TableForm@TrigReduce[#[[1]]]]; Plot[Evaluate[#[[1]]], {k1, 0, kmax}, AxesLabel -> {k1, None},
      PlotRange → {Full, {-1, 1}}, PlotStyle → Thick, PlotLegends → Placed[(MatrixForm[#] & /@ #[[2]]), {{0, 0}, {-0.5, 0.0}}]]) &[
  Transpose[{transferfunktion[#], MatrixForm[{#}]} & /@ {identität[3], binom[2], identität[3] - binom[2]}]], ImageSize → pagewidth]
```



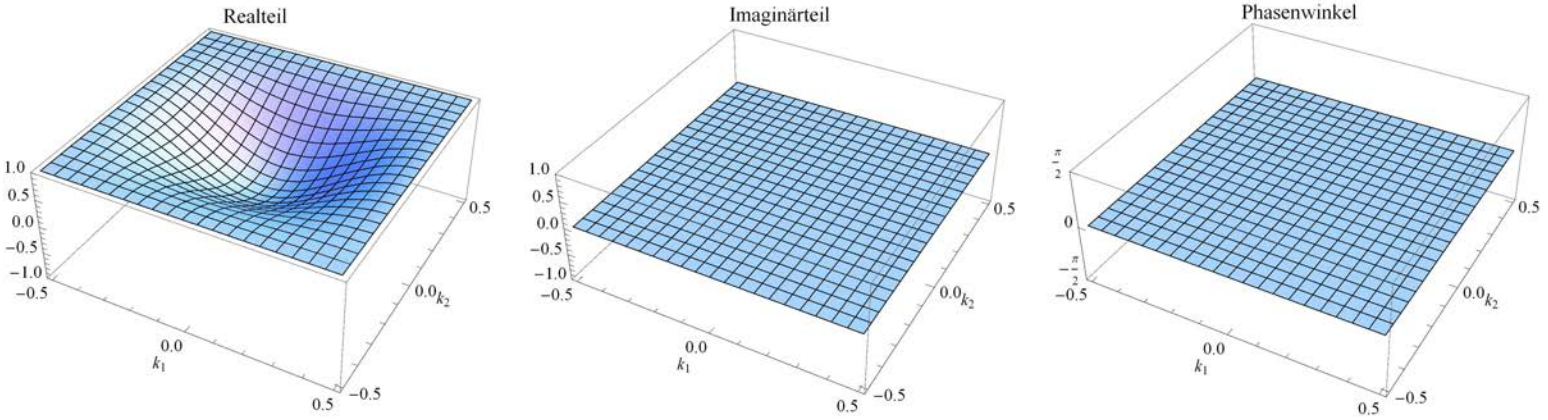
Die Transferfunktion dieses 2D-Hochpasses:

```
hochpaßkern = Transpose[{identität[3]}].{identität[3]} - Transpose[{binom[2]}].{binom[2]};

hochpaßkern // MatrixForm
```

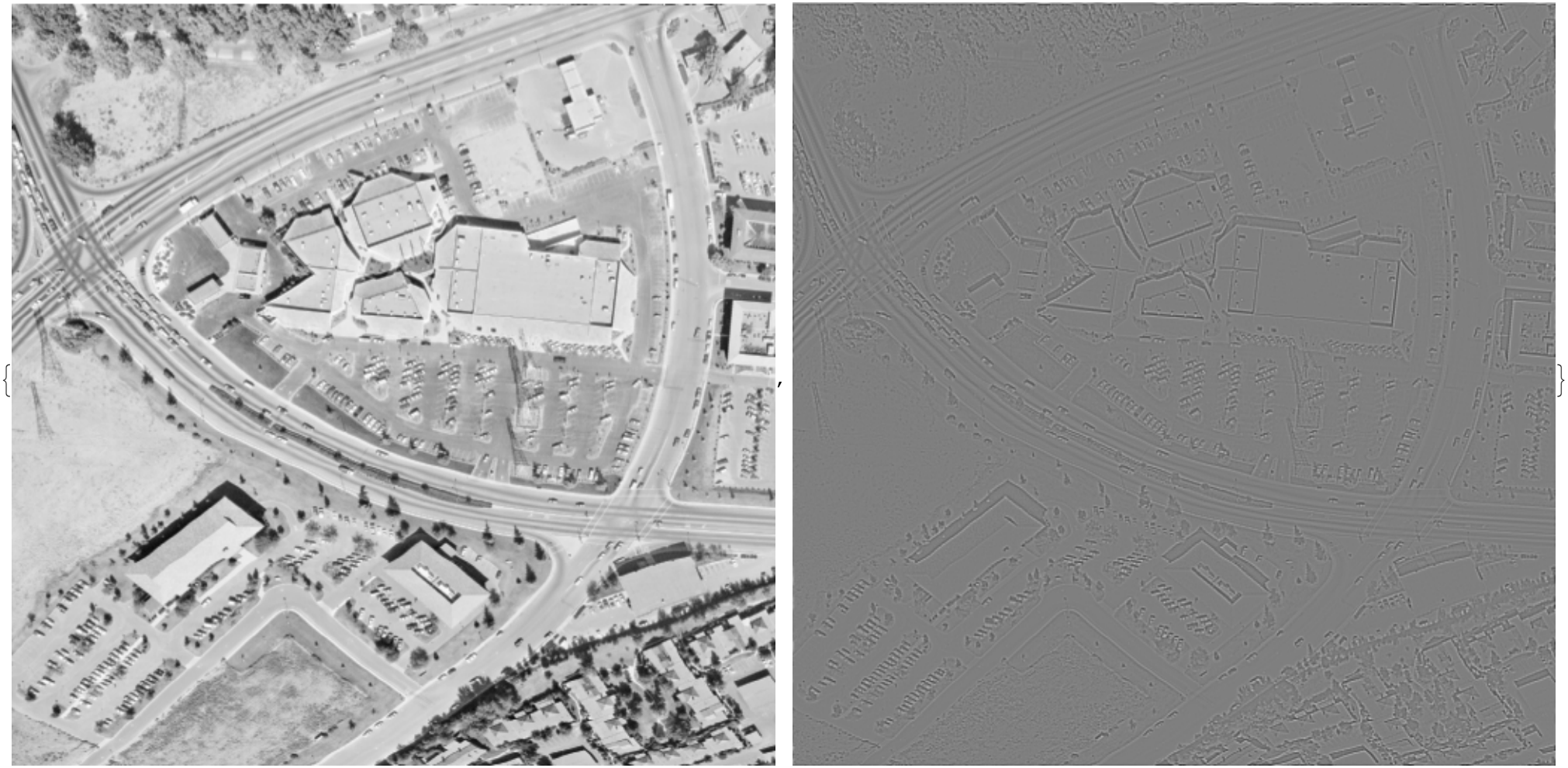
$$\begin{pmatrix} -\frac{1}{16} & -\frac{1}{8} & -\frac{1}{16} \\ -\frac{1}{8} & \frac{3}{4} & -\frac{1}{8} \\ -\frac{1}{16} & -\frac{1}{8} & -\frac{1}{16} \end{pmatrix}$$

```
transferfunktion2d[hochpaßkern]
Show[plottransferfunktion2d[transferfunktion2d[hochpaßkern]], ImageSize → pagewidth]
```

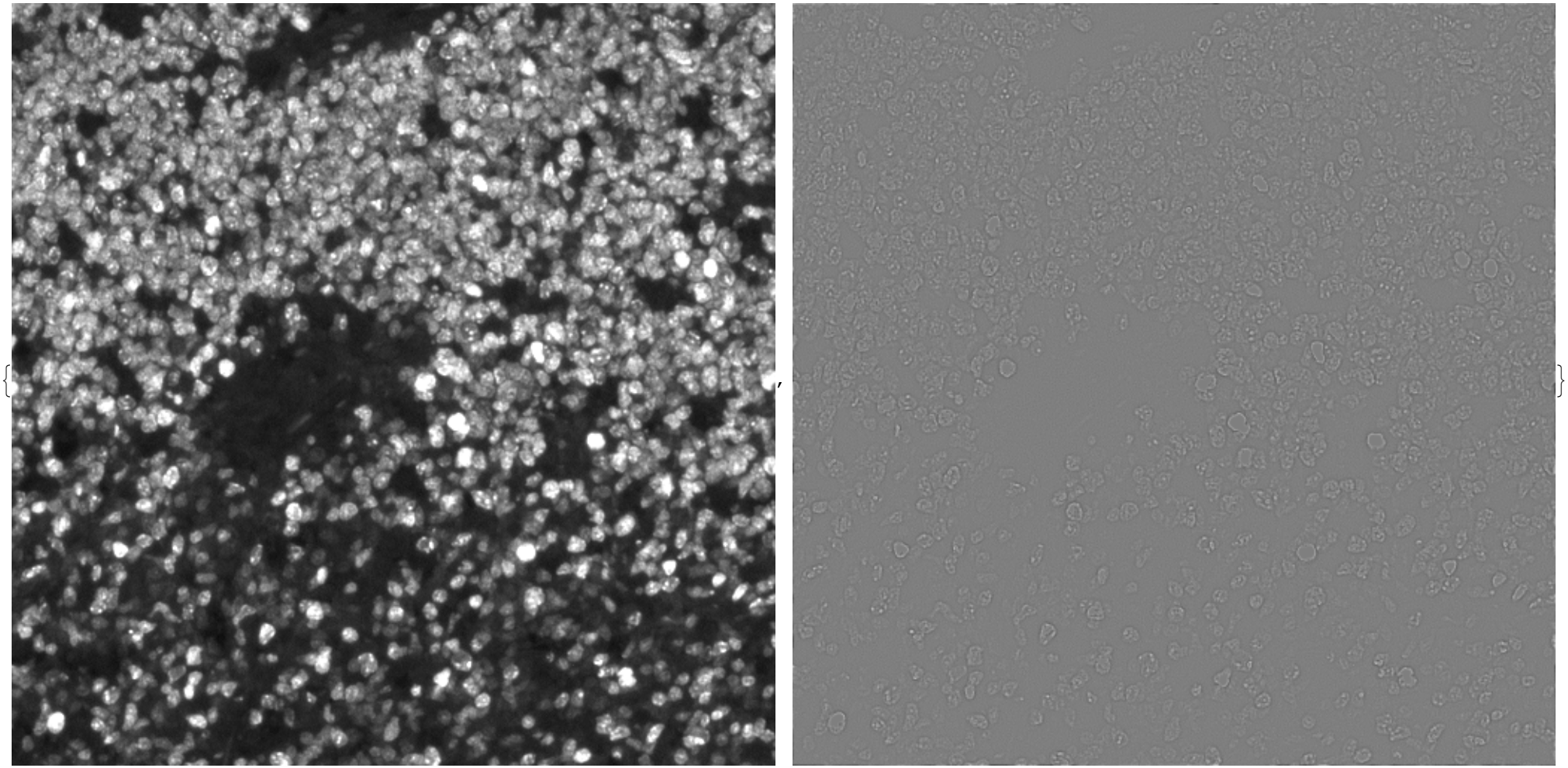
$$\frac{1}{8}\,(6 - 2\,\text{Cos}[2\,\pi\,k_1] - \text{Cos}[2\,\pi\,(k_1 - k_2)] - 2\,\text{Cos}[2\,\pi\,k_2] - \text{Cos}[2\,\pi\,(k_1 + k_2)])$$



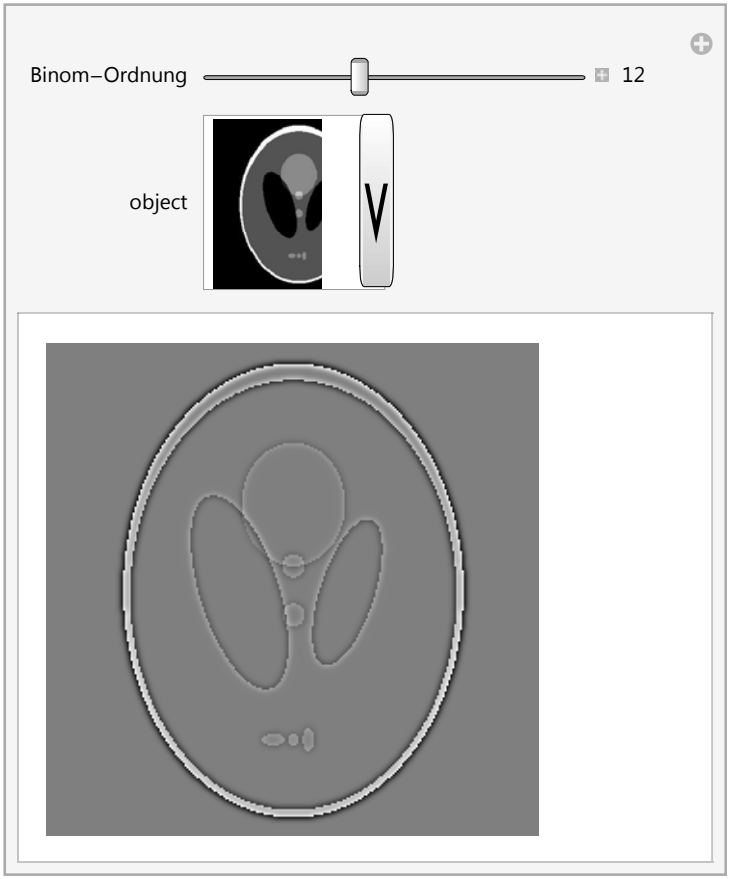
```
{Show[Image[#2]], Show[Image[0.5 + ListConvolve[#1, #2, {(Dimensions[#1] + 1) / 2}]]]} &[
 hochpaßkern, ImageData[ExampleData[{"TestImage", "Aerial2"}]]]
```



```
{Show[Image[#2]], Show[Image[0.5 + ListConvolve[#1, #2, {(Dimensions[#1] + 1) / 2}]]]} &[
 hochpaßkern, ImageData[First@ColorSeparate[Ton3CD21dreikanalausgleichF2]]]
```



```
Manipulate[
 Show[Image[1 / 2 + ListConvolve[Transpose[{identität[ordnung + 1]}].{identität[ordnung + 1]} - Transpose[{binom[ordnung]}].{binom[ordnung]},
     ImageData@bild, ordnung / 2 + 1]], ImageSize → ImageDimensions@bild], {{ordnung, 2, "Binom-Ordnung"}, 0, 30, 2, Appearance → "Labeled"},
 {{bild, wellenbild, "object"}, {wellenbild, First@ColorSeparate[Ton3CD21dreikanalausgleichF2], mandrill, phantomhead, turtle, lenay,
   pollen, randombild, rauschbild, ocelot}, ImageSize → Tiny, ControlType → PopupMenu}, ContinuousAction → False, SaveDefinitions → True]
```



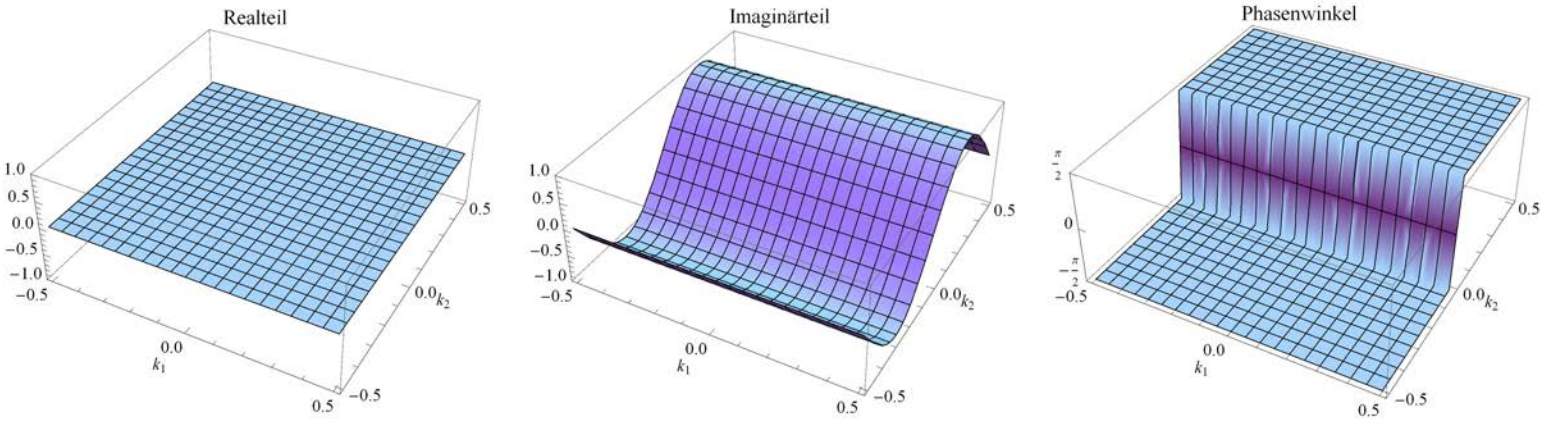
Approximation von Ableitungsoperatoren 1. Ordnung durch lineare Filter

```
Transpose[{identität[3]}].{ableitungsymm} // MatrixForm
```

$$\begin{pmatrix} 0 & 0 & 0 \\ \frac{1}{2} & 0 & -\frac{1}{2} \\ 0 & 0 & 0 \end{pmatrix}$$

```
transferfunktion2d[Transpose[{identität[3]}].{ableitungsymm}]
Show[plottransferfunktion2d[transferfunktion2d[Transpose[{identität[3]}].{ableitungsymm}]], ImageSize → pagewidth]
```

$i \, \mathrm{Sin}[2 \pi k_2]$

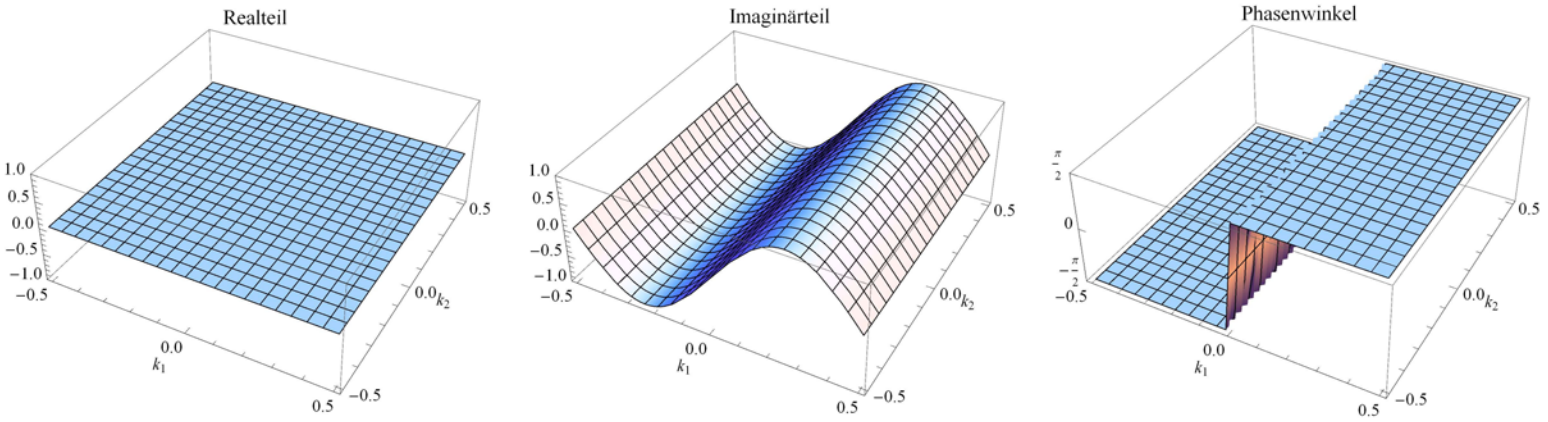



```
Transpose[{ableitungsymm}].{identität[3]} // MatrixForm
```

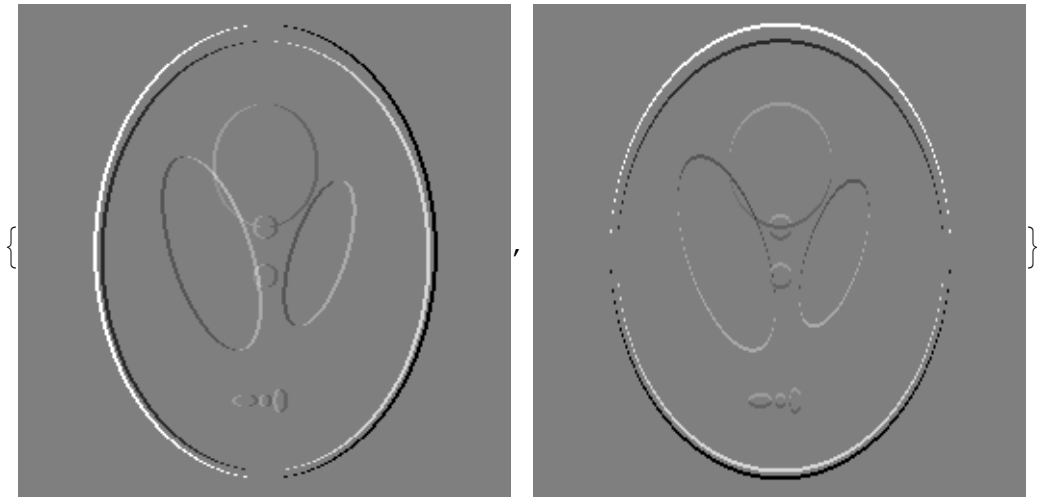
$$\begin{pmatrix} 0 & \frac{1}{2} & 0 \\ 0 & 0 & 0 \\ 0 & -\frac{1}{2} & 0 \end{pmatrix}$$

```
transferfunktion2d[Transpose[{ableitungsymm}].{identität[3]}]
Show[plottransferfunktion2d[transferfunktion2d[Transpose[{ableitungsymm}].{identität[3]}]], ImageSize → pagewidth]
```

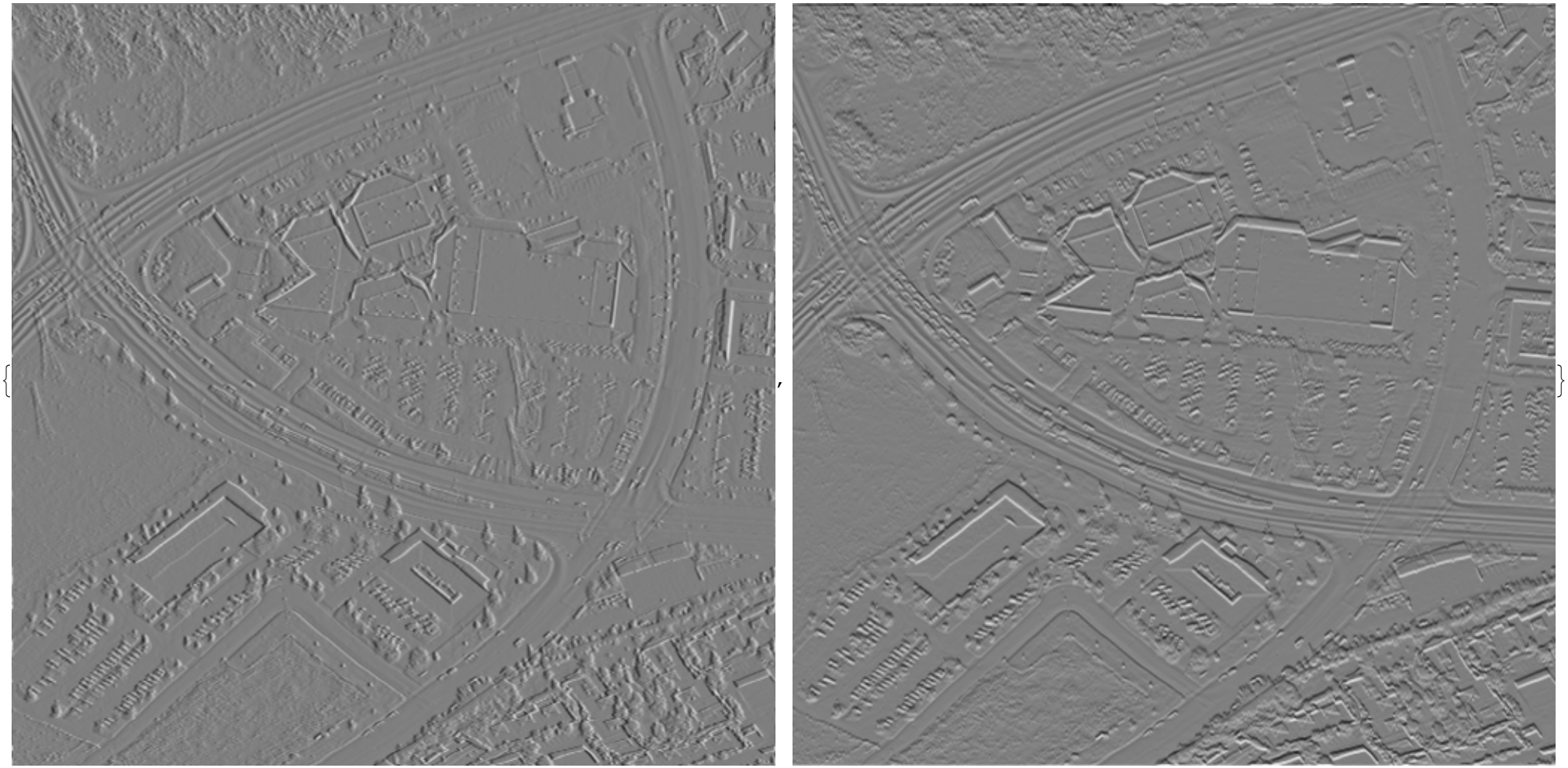
$i \sin[2 \pi k_1]$



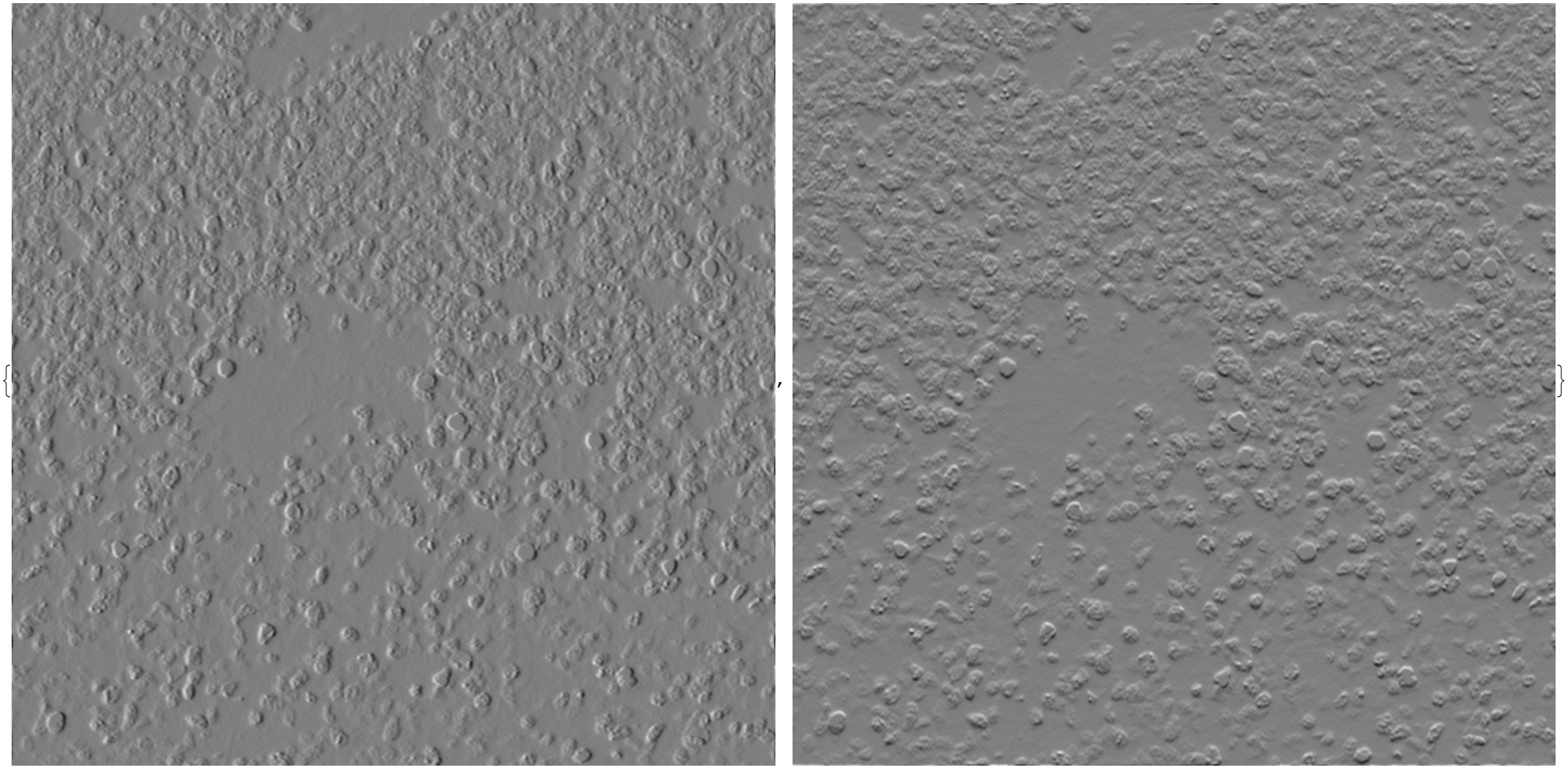

```
{Show[Image[0.5 + ListConvolve[#1, ImageData@phantomhead, {(Dimensions[#1] + 1) / 2}]]],
   Show[Image[0.5 + ListConvolve[#2, ImageData@phantomhead, {(Dimensions[#2] + 1) / 2}]]]} &[
 Transpose[{identität[3]}].{ableitungsymm}, Transpose[{ableitungsymm}].{identität[3]}]
```



```
{Show[Image[0.5 + ListConvolve[#1, ImageData[ExampleData[{"TestImage", "Aerial2"}]], {(Dimensions[#1] + 1) / 2}]]],
   Show[Image[0.5 + ListConvolve[#2, ImageData[ExampleData[{"TestImage", "Aerial2"}]], {(Dimensions[#2] + 1) / 2}]]]} &[
 Transpose[{identität[3]}].{ableitungsymm}, Transpose[{ableitungsymm}].{identität[3]}]
```

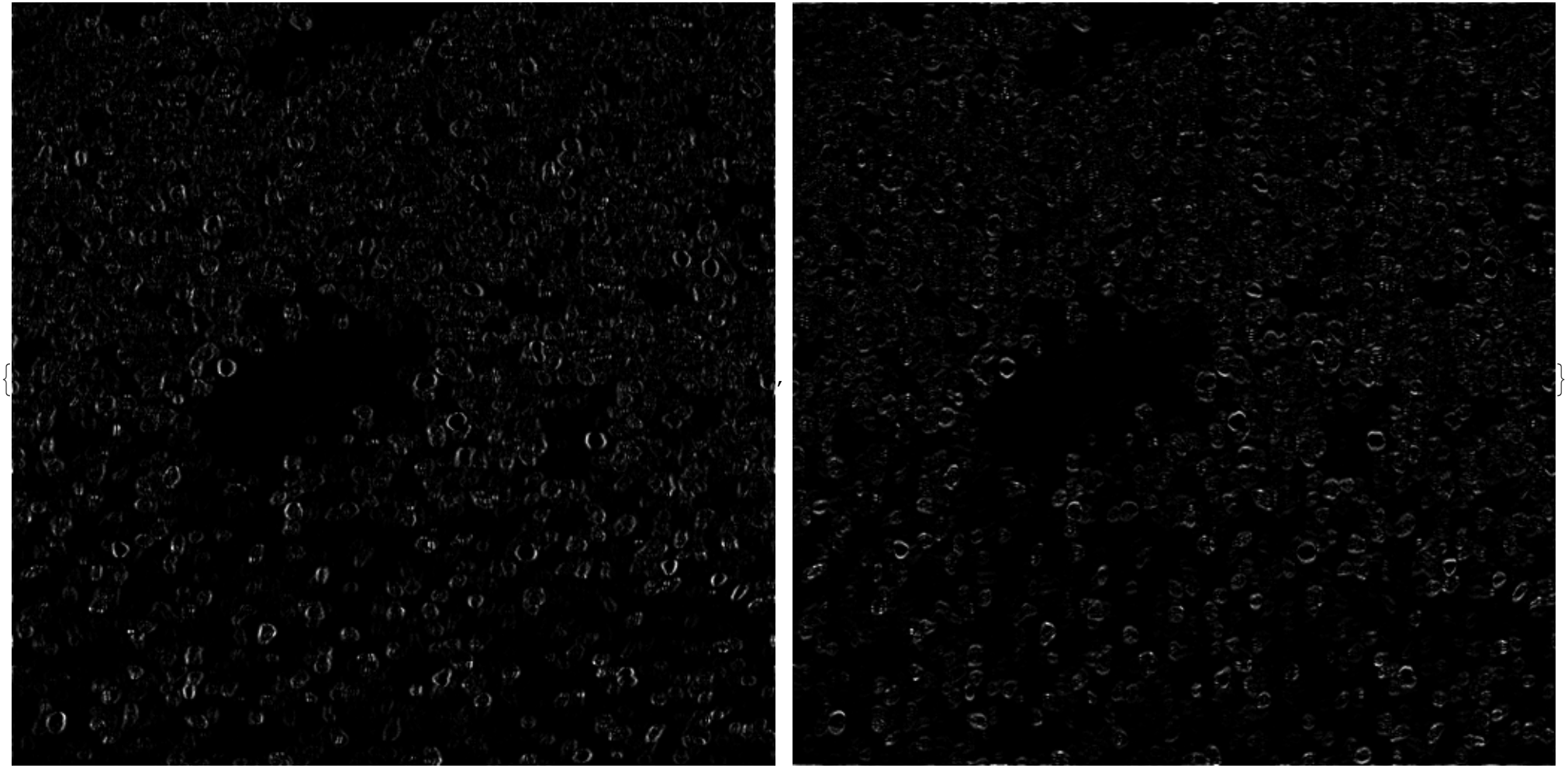


```
{Show[Image[0.5 + ListConvolve[#1, ImageData[First@ColorSeparate[Ton3CD21dreikanalausgleichF2]], {(Dimensions[#1] + 1) / 2}]]],
   Show[Image[0.5 + ListConvolve[#2, ImageData[First@ColorSeparate[Ton3CD21dreikanalausgleichF2]], {(Dimensions[#2] + 1) / 2}]]]} &[
 Transpose[{identität[3]}].{ableitungsymm}, Transpose[{ableitungsymm}].{identität[3]}]
```

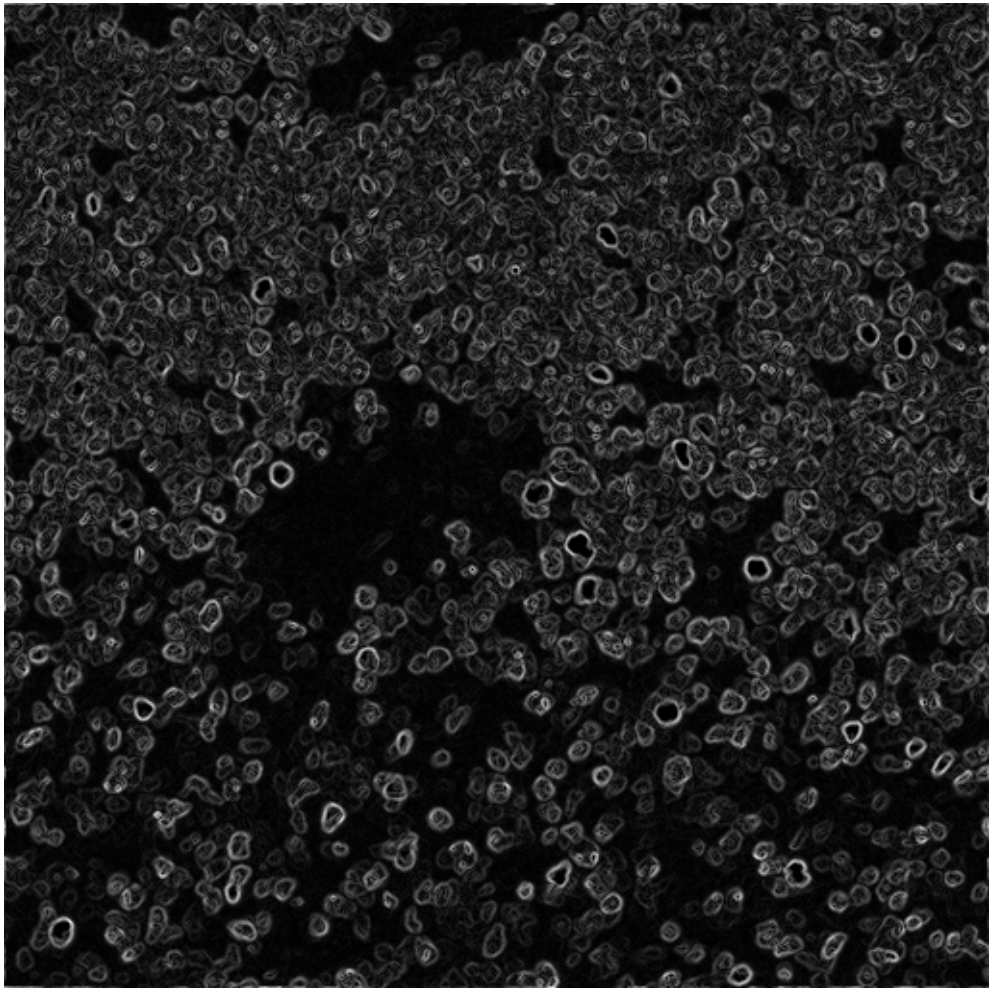


Minderung der Richtungsabhängigkeit: pixelweises Quadrieren der jeweiligen Richtungsableitungen und anschließendes Radizieren

```
{Show[ImageAdjust@Image[ListConvolve[Transpose[{identität[3]}].{ableitungsymm}, #, {{2, 2}}]^2]],
   Show[ImageAdjust@Image[ListConvolve[Transpose[{ableitungsymm}].{identität[3]}, #, {{2, 2}}]^2]]} &[
 ImageData[First@ColorSeparate[Ton3CD21dreikanalausgleichF2]]]
```



```
Show[ImageAdjust@Image[Sqrt[ListConvolve[Transpose[{identität[3]}].{ableitungsymm}, #, {{2, 2}}]^2 +
       ListConvolve[Transpose[{ableitungsymm}].{identität[3]}, #, {{2, 2}}]^2]] &[
 ImageData[First@ColorSeparate[Ton3CD21dreikanalausgleichF2]]]
```



Verfeinerte Gradientenoperatoren mit mehr als zwei Richtungen und Querglättung: Sobelfilter

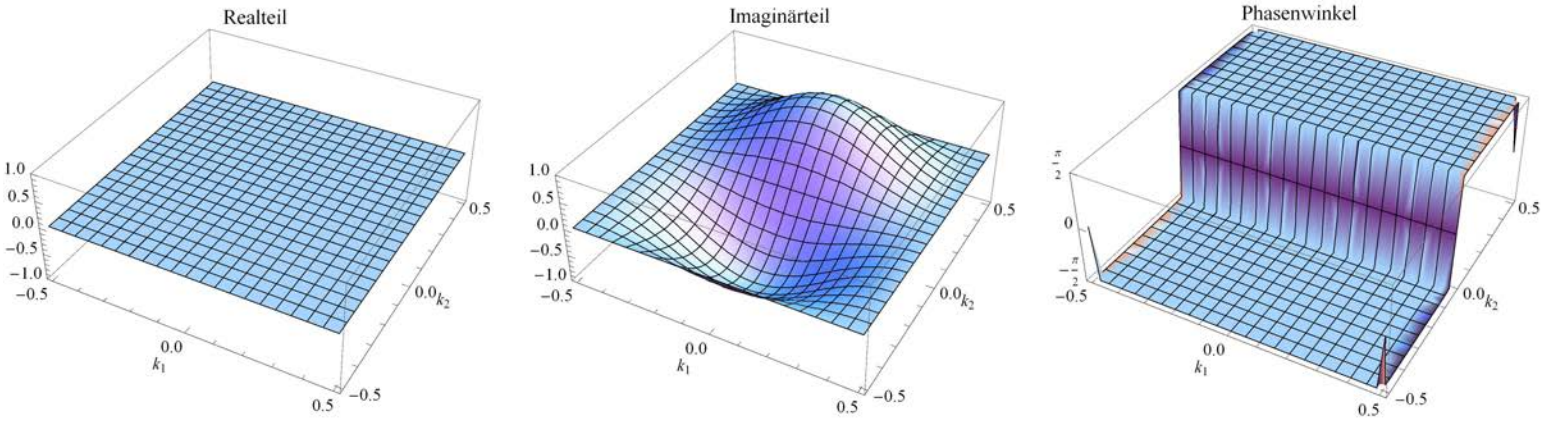
```
Transpose[{binom[2]}].{ableitungsymm} // MatrixForm
```

$$\begin{pmatrix} \frac{1}{8} & 0 & -\frac{1}{8} \\ \frac{1}{4} & 0 & -\frac{1}{4} \\ \frac{1}{8} & 0 & -\frac{1}{8} \end{pmatrix}$$

$$\text{sobel1} = \begin{pmatrix} \frac{1}{8} & 0 & -\frac{1}{8} \\ \frac{1}{4} & 0 & -\frac{1}{4} \\ \frac{1}{8} & 0 & -\frac{1}{8} \end{pmatrix};$$

```
transferfunktion2d[sobel1]
Show[plottransferfunktion2d[transferfunktion2d[sobel1]], ImageSize → pagewidth]
```

$$i \cos[\pi k_1]^2 \sin[2 \pi k_2]$$



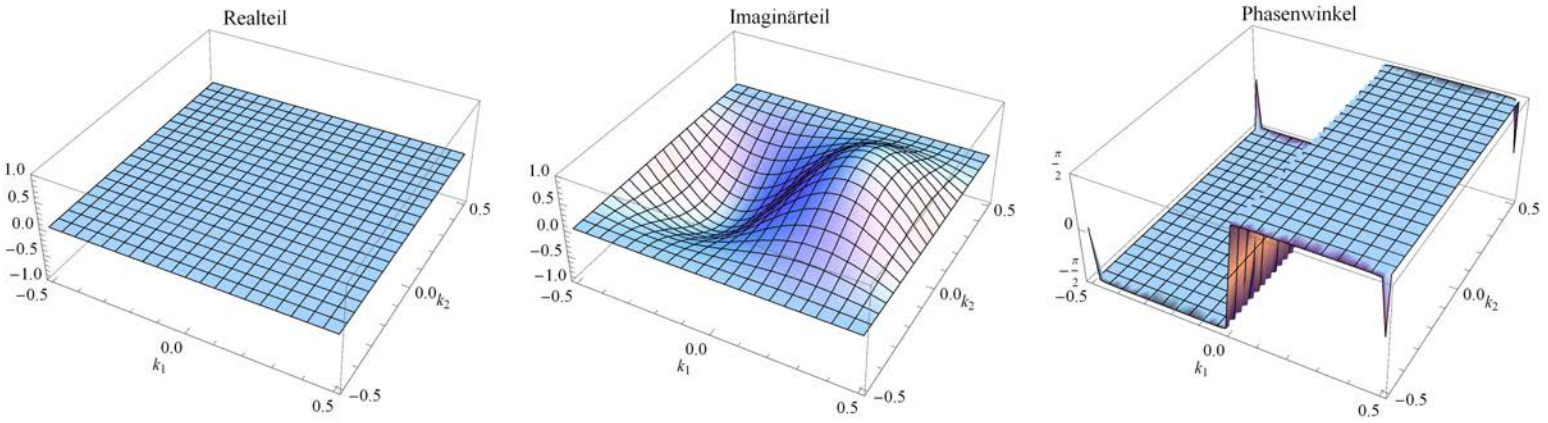

```
Transpose[{ableitungsymm}].{binom[2]} // MatrixForm
```

$$\begin{pmatrix} \frac{1}{8} & \frac{1}{4} & \frac{1}{8} \\ 0 & 0 & 0 \\ -\frac{1}{8} & -\frac{1}{4} & -\frac{1}{8} \end{pmatrix}$$

$$\text{sobel2} = \begin{pmatrix} \frac{1}{8} & \frac{1}{4} & \frac{1}{8} \\ 0 & 0 & 0 \\ -\frac{1}{8} & -\frac{1}{4} & -\frac{1}{8} \end{pmatrix};$$

```
transferfunktion2d[sobel2]
Show[plottransferfunktion2d[transferfunktion2d[sobel2]], ImageSize → pagewidth]
```

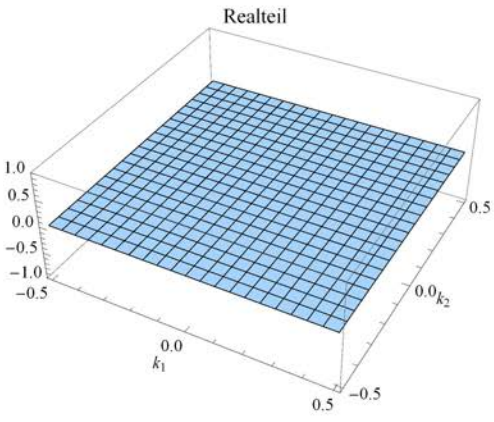
$$i \cos[\pi k_2]^2 \sin[2 \pi k_1]$$

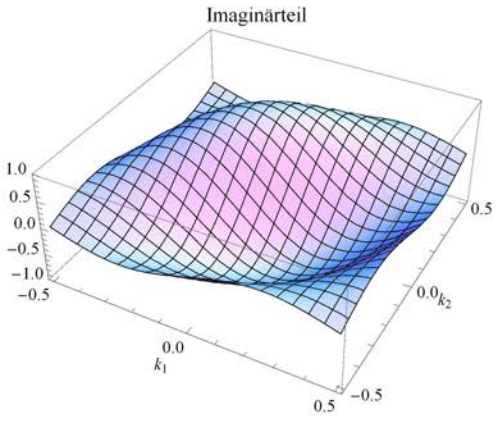


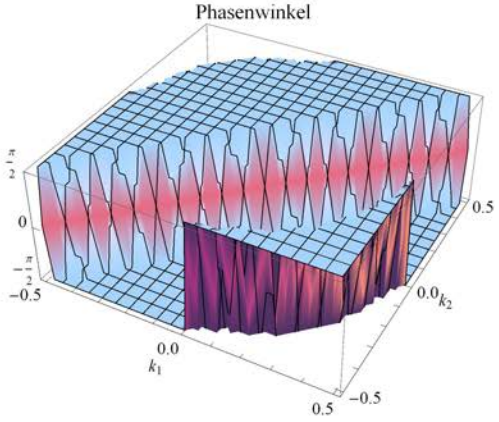
$$\text{sobel3} = \begin{pmatrix} 0 & -\frac{1}{8} & -\frac{1}{4} \\ \frac{1}{8} & 0 & -\frac{1}{8} \\ \frac{1}{4} & \frac{1}{8} & 0 \end{pmatrix};$$

```
transferfunktion2d[sobel3]
Show[plottransferfunktion2d[transferfunktion2d[sobel3]], ImageSize → pagewidth]
```

$$-\frac{1}{4}\,i\,(\text{Sin}[2\,\pi\,k_1]+2\,\text{Sin}[2\,\pi\,(k_1-k_2)]-\text{Sin}[2\,\pi\,k_2])$$



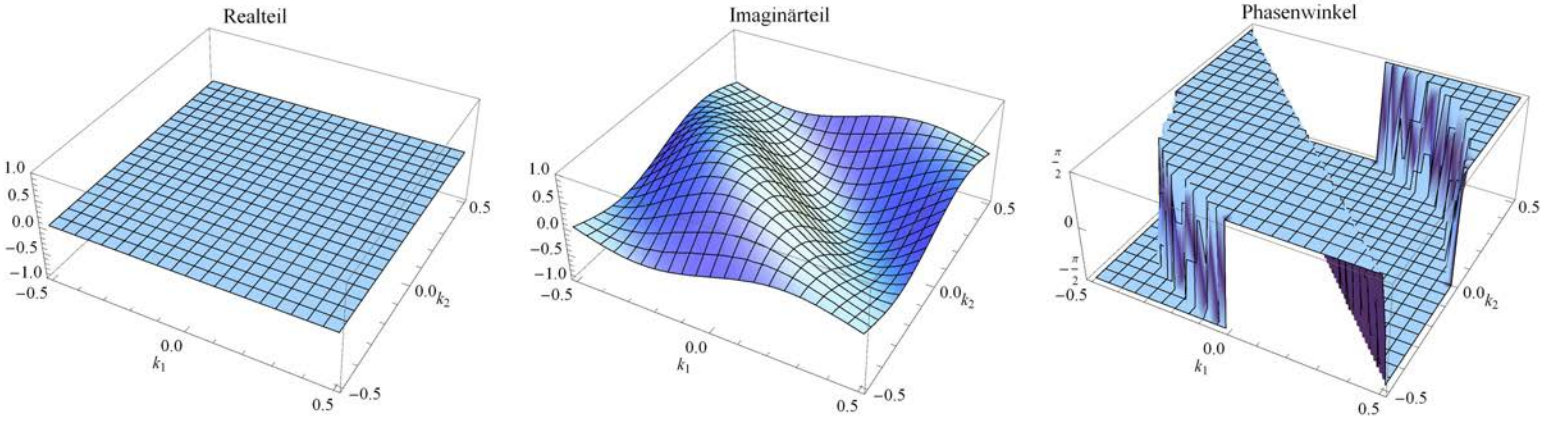




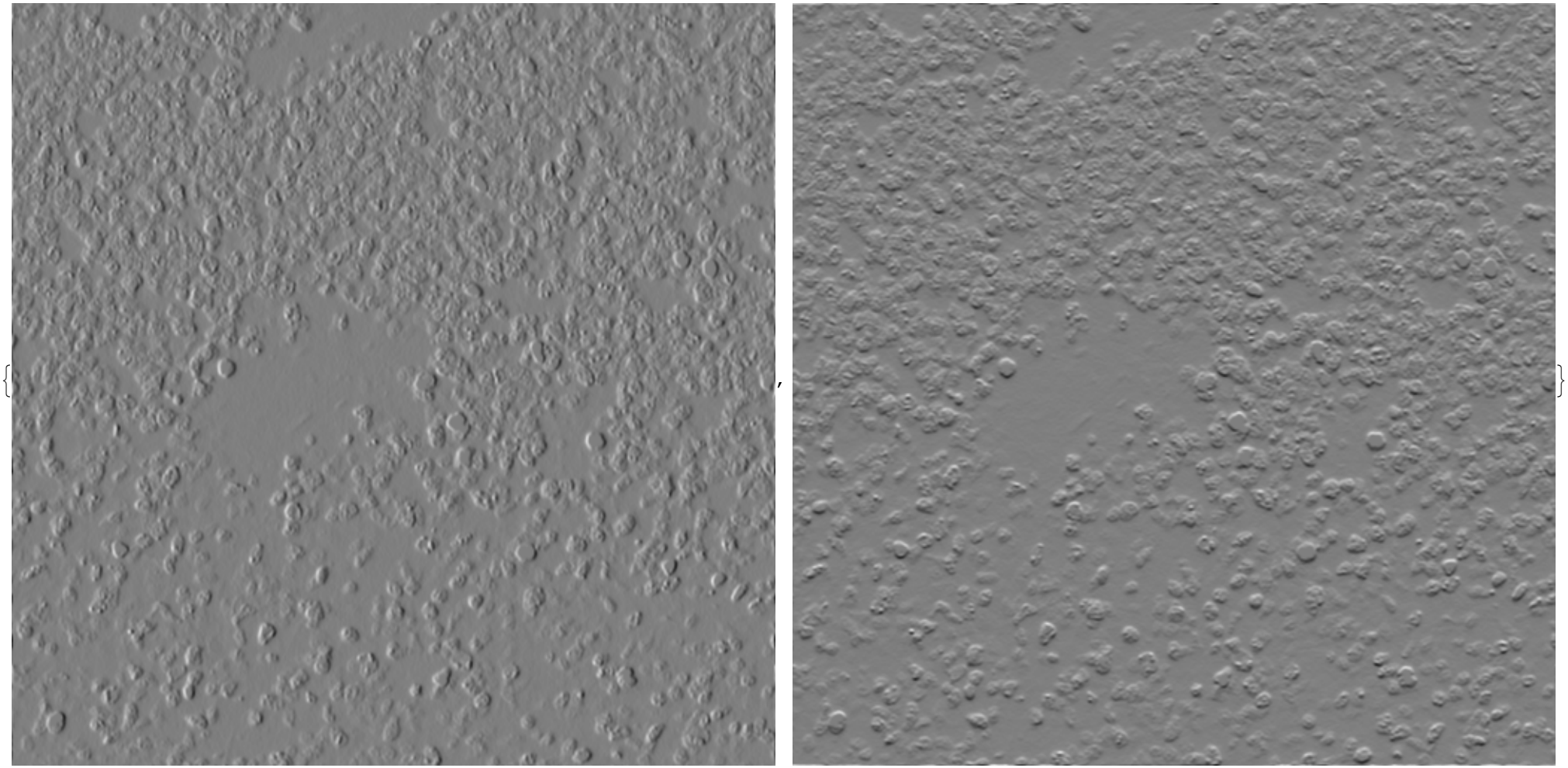
$$\text{sobel4} = \begin{pmatrix} -\frac{1}{4} & -\frac{1}{8} & 0 \\ -\frac{1}{8} & 0 & \frac{1}{8} \\ 0 & \frac{1}{8} & \frac{1}{4} \end{pmatrix};$$

```
transferfunktion2d[sobel4]
Show[plottransferfunktion2d[transferfunktion2d[sobel4]], ImageSize → pagewidth]
```

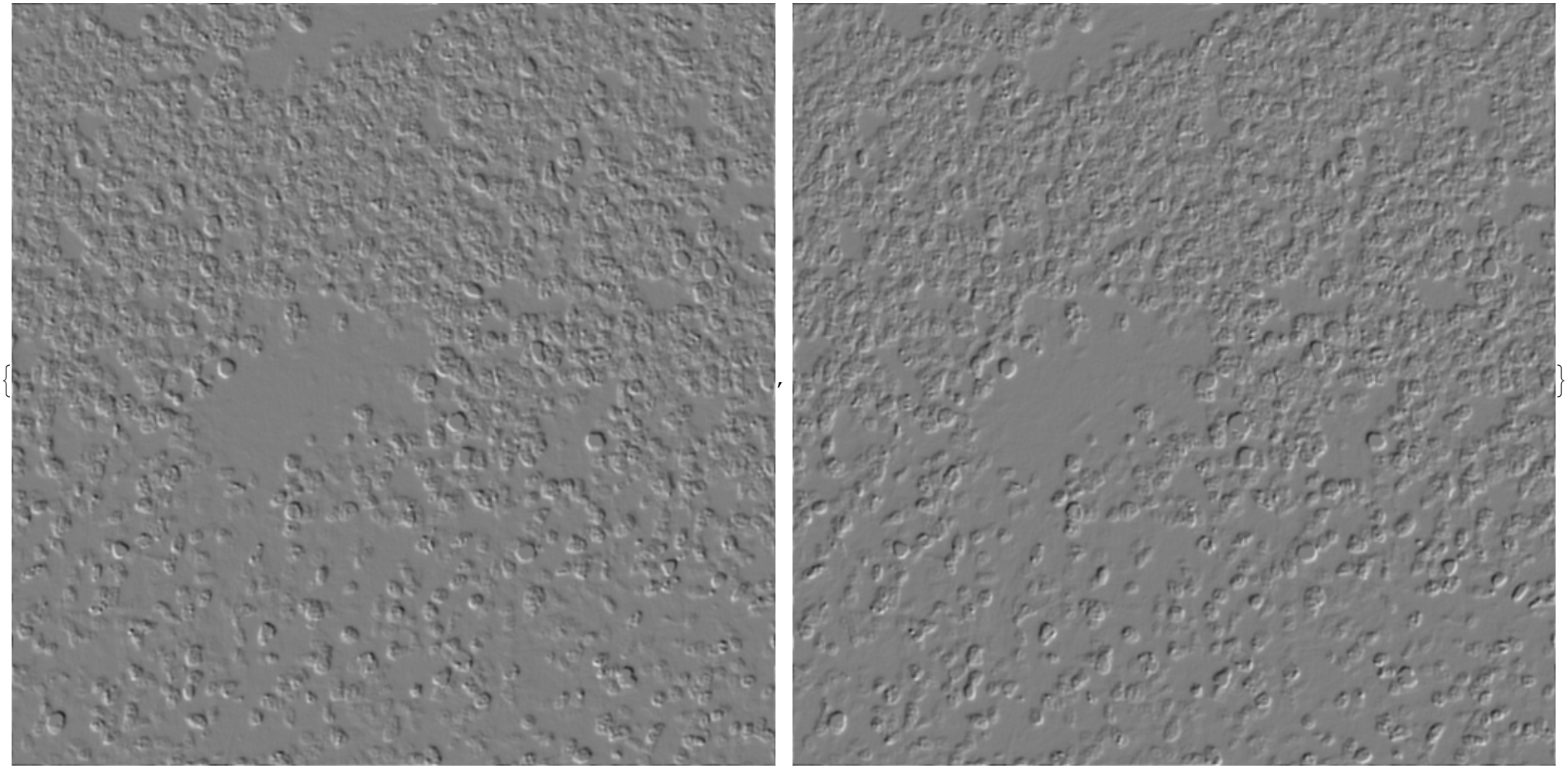
$$-\frac{1}{4}\,i\,(\text{Sin}[2\,\pi\,k_1]+\text{Sin}[2\,\pi\,k_2]+2\,\text{Sin}[2\,\pi\,(k_1+k_2)])$$



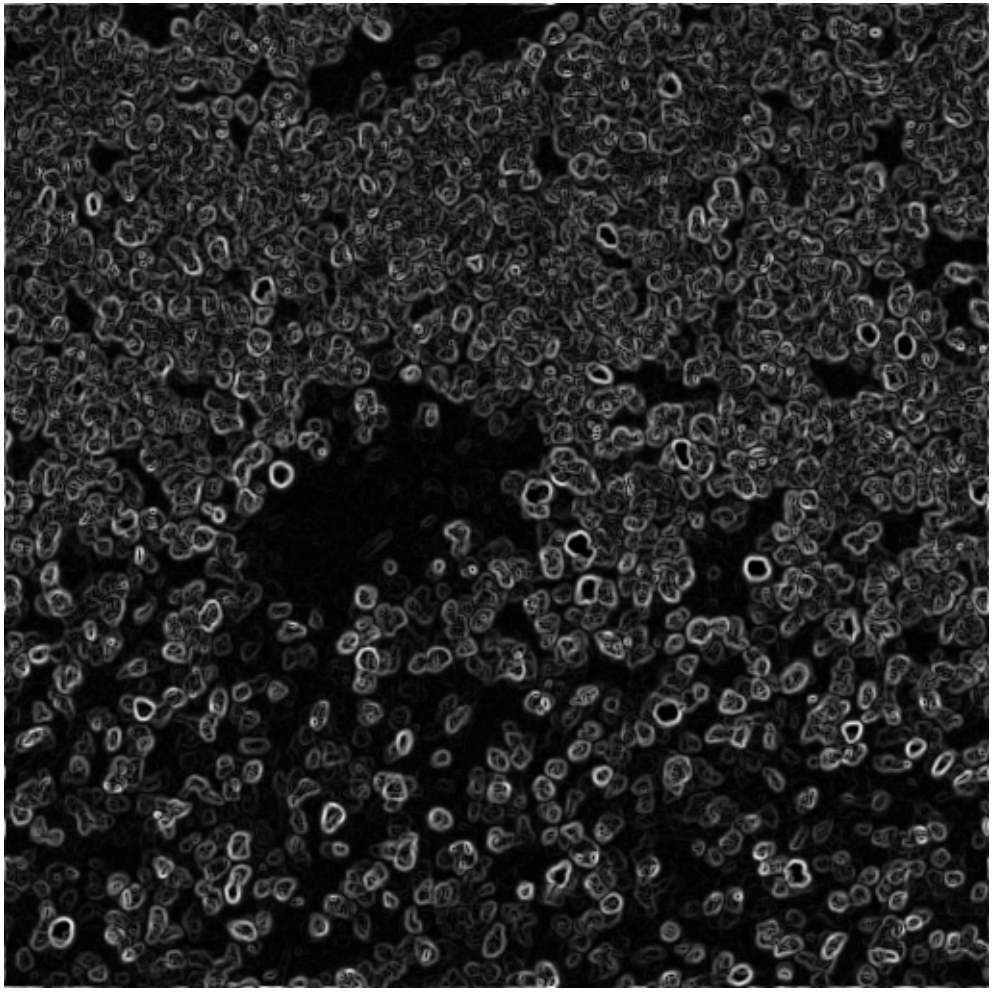
```
{Show[Image[0.5 + ListConvolve[sobel1, #, {{2, 2}}]]], Show[Image[0.5 + ListConvolve[sobel2, #, {{2, 2}}]]]} &[
 ImageData[First@ColorSeparate[Ton3CD21dreikanalausgleichF2]]]
```



```
{Show[Image[0.5 + ListConvolve[sobel3, #, {{2, 2}}]]], Show[Image[0.5 + ListConvolve[sobel4, #, {{2, 2}}]]]} &[
 ImageData[First@ColorSeparate[Ton3CD21dreikanalausgleichF2]]]
```



```
Show[ImageAdjust@Image[Sqrt[ListConvolve[sobel1, #, {{2, 2}}]^2 + ListConvolve[sobel2, #, {{2, 2}}]^2 + ListConvolve[sobel3, #, {{2, 2}}]^2 +
      ListConvolve[sobel4, #, {{2, 2}}]^2]] &[ImageData[First@ColorSeparate[Ton3CD21dreikanalausgleichF2]]]
```



Approximation von Ableitungsoperatoren 2. Ordnung durch lineare Filter

Zweimalige Anwendung von Ableitungsoperatoren, Unabhängigkeit von konstanten Änderungen: Laplacefilter → Addition der beiden Richtungs-Laplacefilter

```
ableitungsymm
```

$$\left\{\frac{1}{2}, 0, -\frac{1}{2}\right\}$$

```
laplacekern = Transpose[{identität[3]}].{ListConvolve[ableitungsymm, ableitungsymm, 2]} +
   Transpose[{ListConvolve[ableitungsymm, ableitungsymm, 2]}].{identität[3]};
(*als Addition für diesen einfachsten Fall!*)
```

```
laplacekern // MatrixForm
```

$$\begin{pmatrix} 0 & \frac{1}{4} & 0 \\ \frac{1}{4} & -1 & \frac{1}{4} \\ 0 & \frac{1}{4} & 0 \end{pmatrix}$$

```
Transpose[{identität[3]}].{ListConvolve[ableitungminus, ableitungplus, 2]} // MatrixForm
```

$$\begin{pmatrix} 0 & 0 & 0 \\ 1 & -2 & 1 \\ 0 & 0 & 0 \end{pmatrix}$$

```
ableitungminus
```

```
{0, 1, -1}
```

```
ableitungplus
```

```
{1, -1, 0}
```

```
Transpose[{ListConvolve[ableitungminus, ableitungplus, 2]}].{identität[3]} // MatrixForm
```

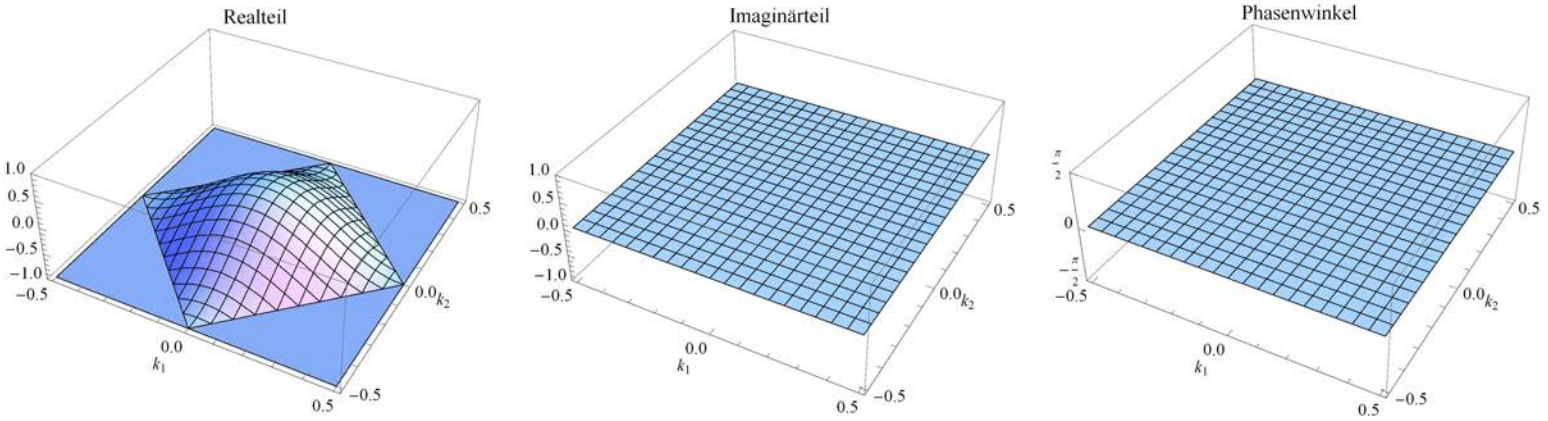
$$\begin{pmatrix} 0 & 1 & 0 \\ 0 & -2 & 0 \\ 0 & 1 & 0 \end{pmatrix}$$

```
(Transpose[{ListConvolve[ableitungminus, ableitungplus, 2]}].{identität[3]} +
    Transpose[{identität[3]}].{ListConvolve[ableitungminus, ableitungplus, 2]}) / 4 // MatrixForm
```

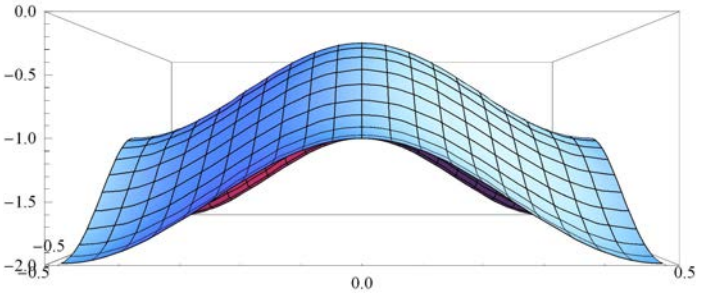
$$\begin{pmatrix} 0 & \frac{1}{4} & 0 \\ \frac{1}{4} & -1 & \frac{1}{4} \\ 0 & \frac{1}{4} & 0 \end{pmatrix}$$

```
transferfunktion2d[laplacekern]
Show[plottransferfunktion2d[transferfunktion2d[laplacekern]], ImageSize → pagewidth]
```

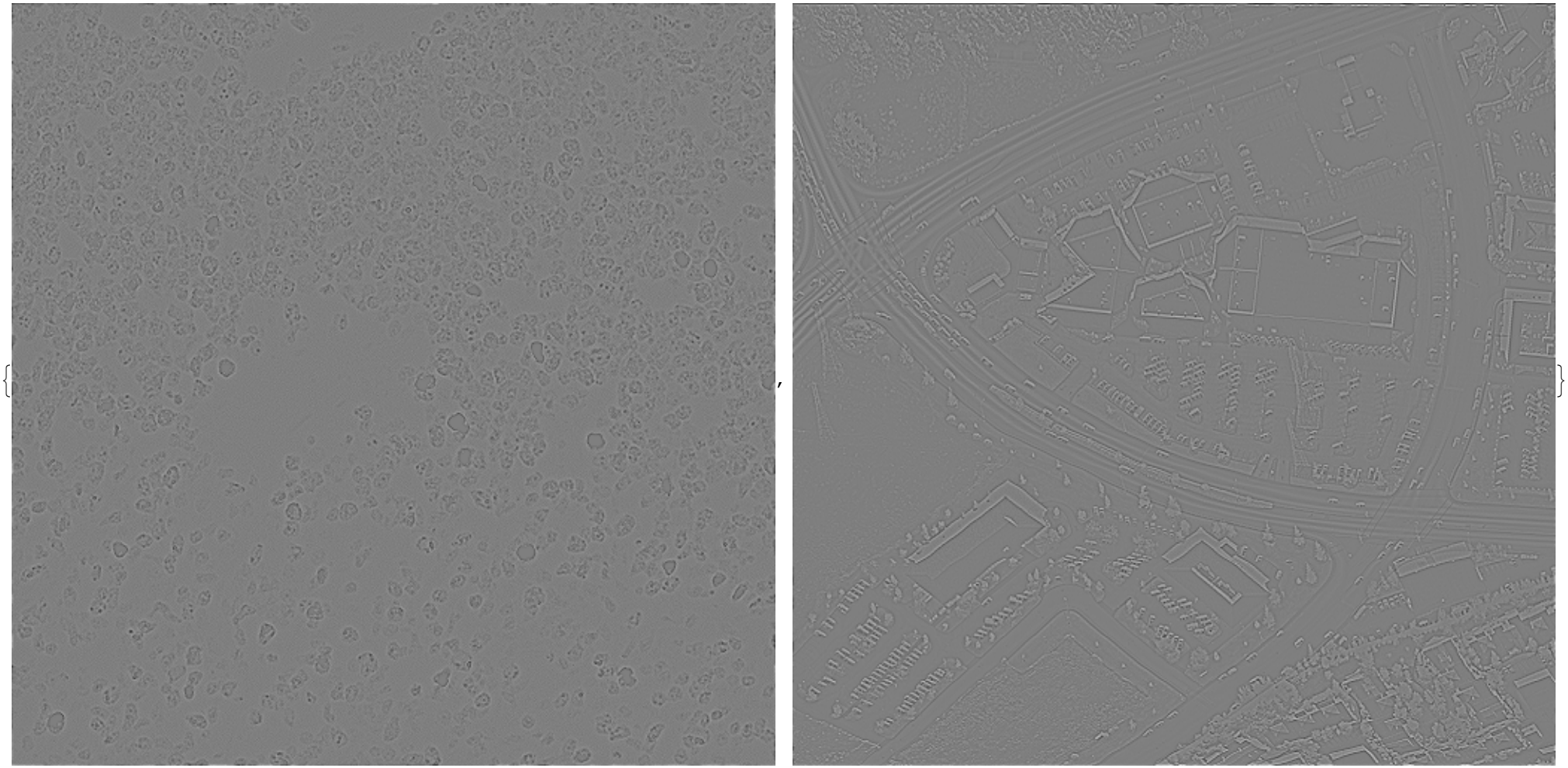
$$\frac{1}{2}\,(-2 + \text{Cos}[2\,\pi\,k_1] + \text{Cos}[2\,\pi\,k_2])$$




```
Plot3D[Evaluate[transferfunktion2d[laplacekern]], {k1, -kmax, kmax}, {k2, -kmax, kmax},
 PlotRange → {Full, Full, {-2, 0}}, Mesh → 19, RotationAction → "Clip", ViewPoint → Front]
```



```
{Show[Image[0.5 + ListConvolve[#, ImageData@First@ColorSeparate[Ton3CD21dreikanalausgleichF2], {(Dimensions[#] + 1) / 2}]]],
   Show[Image[0.5 + ListConvolve[#, ImageData[ExampleData[{"TestImage", "Aerial2"}]], {(Dimensions[#] + 1) / 2}]]]} &[laplacekern]
```



Glättungsfilter 1D mit steilerem Abfall:

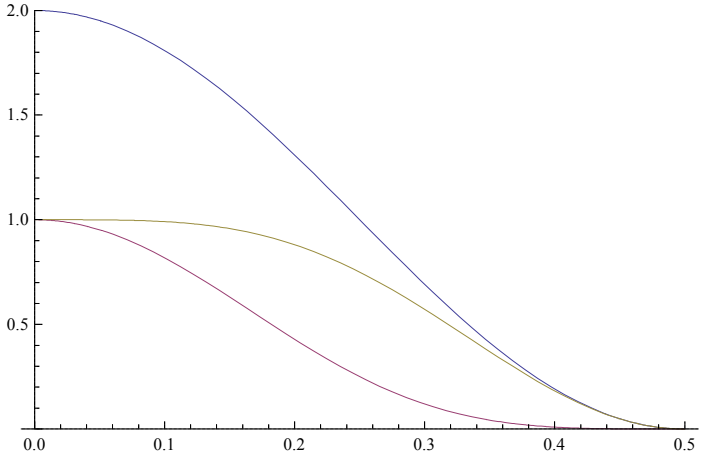
- durch geeignete Kombination des Identitätsoperators $\boldsymbol{I} = (0, 1, 0)$ mit der einfachen Binomialmaske zweiter Ordnung ${}^{2}\boldsymbol{B} = (1/4, 1/2, 1/4)$

- Verallgemeinerter Ansatz: ${}^{(n,l)}\boldsymbol{B} = \left[\boldsymbol{I} - \left(\boldsymbol{I} - {}^{2}\boldsymbol{B}\right)^{n}\right]^{l}$

- Verallgemeinerter Ansatz : $^{(n,l)}\hat{\boldsymbol{B}} = \left\{1 - \frac{1}{2^n}[1 - \cos(2\pi k)]^n\right\}^l$

- Filter zweiter Ordnung: $^{(2,1)}\boldsymbol{B} = 2.^2\boldsymbol{B} -^4\boldsymbol{B}$

```
Plot[Evaluate@{2 transferfunktion[binom[2]], transferfunktion[binom[4]],
    transferfunktion[2 ListConvolve[binom[2], identität[5], 2] - binom[4]]}, {k1, 0, kmax}, PlotRange → {Full, {-0, 2}}]
```



Die Umformung für das obige Beispiel

```
(1 - ((1 - b^2) ^2)) ^1 // Simplify // Expand
```

$2\,b^2 - b^4$

Minuend (“doppelter” Binomialfilter 2. Ordnung)

```
2 ListConvolve[binom[2], identität[5], 2]
```

$\left\{0, \frac{1}{2}, 1, \frac{1}{2}, 0\right\}$

Subtrahend (Binomialfilter 4. Ordnung)

```
binom[4]
```

$\left\{\frac{1}{16}, \frac{1}{4}, \frac{3}{8}, \frac{1}{4}, \frac{1}{16}\right\}$

Ergebnis gemäß $^{(2,1)}\boldsymbol{B} = 2 .^{2} \boldsymbol{B} -^{4} \boldsymbol{B}$

```
2 ListConvolve[binom[2], identität[5], 2] - binom[4]
```

$$\left\{-\frac{1}{16}, \frac{1}{4}, \frac{5}{8}, \frac{1}{4}, -\frac{1}{16}\right\}$$

Dieselben Koeffizienten aus allgemeinem Ansatz: $^{(2,1)}\boldsymbol{B} = \boldsymbol{I} - \left(\boldsymbol{I} -^{2} \boldsymbol{B}\right)^{2}$

```
identität[5] - ListConvolve[identität[5] - ListConvolve[binom[2], identität[5], 2], identität[5] - ListConvolve[binom[2], identität[5], 2], 3]
```

$$\left\{-\frac{1}{16}, \frac{1}{4}, \frac{5}{8}, \frac{1}{4}, -\frac{1}{16}\right\}$$

Transferfunktion $^{(2,1)}\hat{\boldsymbol{B}}$ des Glättungsfilters $^{(2,1)}\boldsymbol{B}$ gebildet aus Filterkoeffizienten gemäß umgeformter Bildungsvorschrift $^{(2,1)}\boldsymbol{B} = 2 .^{2} \boldsymbol{B} -^{4} \boldsymbol{B}$

```
transferfunktion[2 ListConvolve[binom[2], identität[5], 2] - binom[4]]
```

$$-\frac{1}{2} \text{Cos}[\pi k_1]^2 \, (-3 + \text{Cos}[2 \pi k_1])$$

Transferfunktion $^{(2,1)}\hat{\boldsymbol{B}}$ des Glättungsfilters $^{(2,1)}\boldsymbol{B}$ gebildet der gewichteten Differenz der Transferfunktionen $^{(2,1)}\hat{\boldsymbol{B}} = 2 .^{2} \hat{\boldsymbol{B}} -^{4} \hat{\boldsymbol{B}}$

```
2 transferfunktion[binom[2]] - transferfunktion[binom[4]] // TrigFactor
```

$$-\frac{1}{2} \text{Cos}[\pi k_1]^2 \, (-3 + \text{Cos}[2 \pi k_1])$$

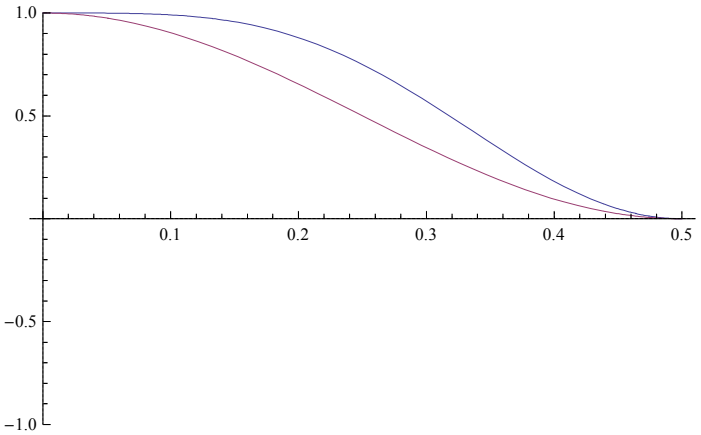
Transferfunktion $^{(2,1)}\hat{\boldsymbol{B}}$ des Glättungsfilters $^{(2,1)}\boldsymbol{B}$ direkt gebildet aus $^{(n,l)}\hat{\boldsymbol{B}} = \left\{1 - \frac{1}{2^n}[1 - \cos(2 \pi k)]^n\right\}^l$

```
(1 - 1 / 2^2 (1 - Cos[2 π k1]) ^2) ^1 // TrigFactor
```

$$-\frac{1}{2} \text{Cos}[\pi k_1]^2 \, (-3 + \text{Cos}[2 \pi k_1])$$

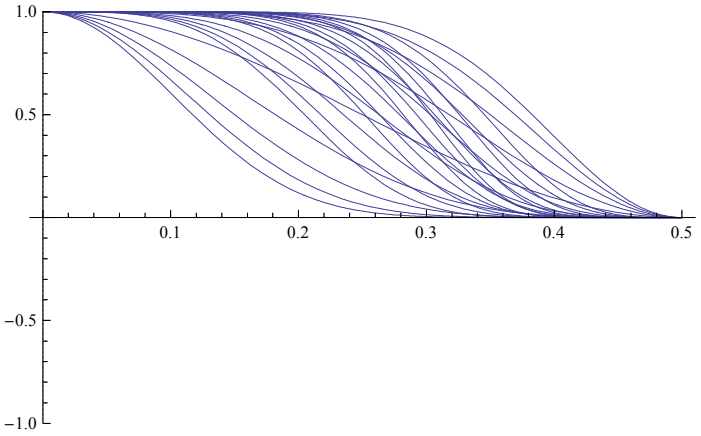
Plot der Transferfunktion $^{(2,1)}\hat{B}$ für $^{(2,1)}B = I - \left(I - {}^{2}B\right)^{2} = 2 \cdot {}^{2}B - {}^{4}B$ im Vergleich zu $^{2}\hat{B}$ des Binomialfilters ^{2}B

```
Plot[Evaluate@{transferfunktion[2 ListConvolve[binom[2], identität[5], 2] - binom[4]], transferfunktion[binom[2]]},
 {k1, 0, kmax}, PlotRange → {Full, {-1, 1}}]
```



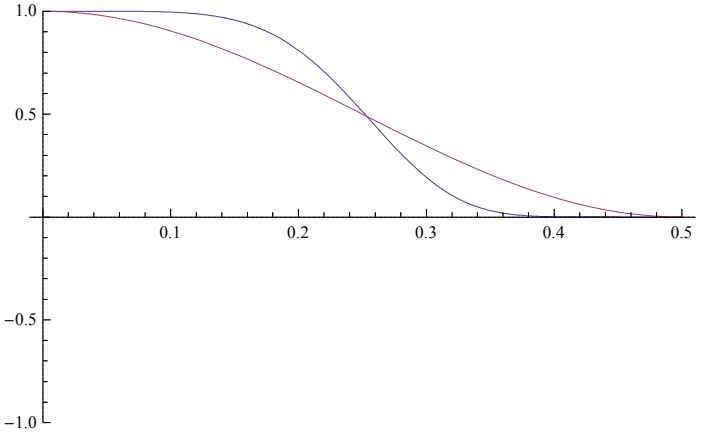
Transferfunktionen $^{(1,1)}\hat{B} = {}^{2}\hat{B}$ bis $^{(5,5)}\hat{B}$

```
Plot[Table[(1 - 1 / 2^n (1 - Cos[2 π k1])^n)^l, {n, 1, 5}, {l, 1, 5}], {k1, 0, kmax}, PlotRange → {Full, {-1, 1}}]
```



Transferfunktionen $^{(1,1)}\hat{B} =^{2} \hat{B}$ und $^{(3,5)}\hat{B}$

```
Plot[{(1 - 1 / 2^3 (1 - Cos[2 π k1])^3)^5, (1 - 1 / 2^1 (1 - Cos[2 π k1])^1)^1}, {k1, 0, kmax}, PlotRange → {Full, {-1, 1}}]
```



Aus 14. Nichtlineare Filter

Medianfilter

Nutzen: Beseitigung singulärer Bildfehler oder störender kleiner Details

“Rangordnungsfilter”: immer das in der Sortierreihenfolge mittlere Element aus Pixelumgebung wird dem Pixel neu zugewiesen

```
Clear[boxmaske];
boxmaske[r_] := Table[1, {2 r + 1}, {2 r + 1}];
```

```
boxmaske[3] // MatrixForm
```

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

```
Clear[kreismaske];
kreismaske[r_] := Ceiling[Rescale[Sign[Table[(x^2 + y^2), {x, -r, r}, {y, -r, r}] - r * (r + 1 / 2)], {1, -1}]];
```

```
kreismaske[3] // MatrixForm
```

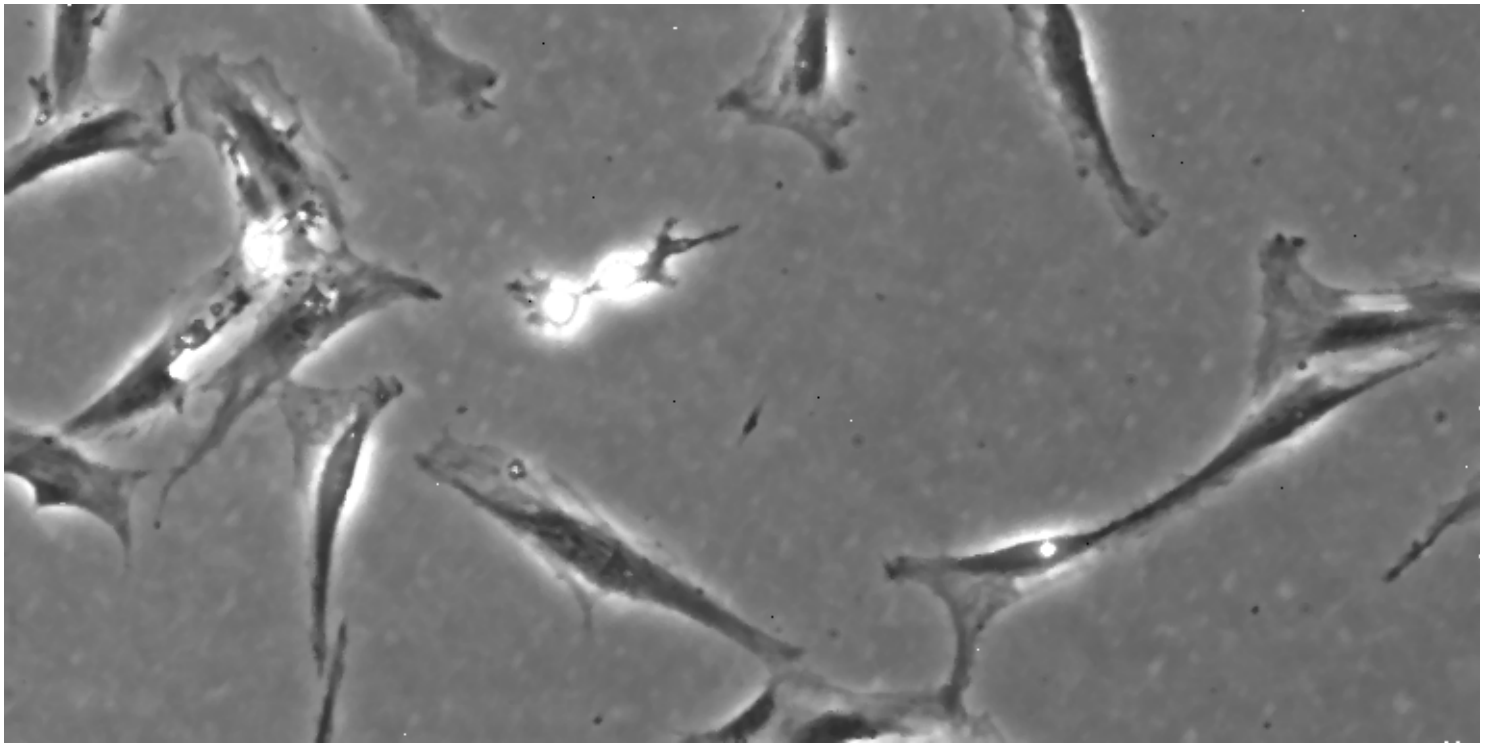
$$\begin{pmatrix} 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 \end{pmatrix}$$

Medianfilter: mit quadratischem Strukturelement (Boxmaske), eingebaute Lösung

```
Manipulate[ControlActive[größe, SeedRandom[2405];
    Show[MedianFilter[Image[Map[If[RandomReal[] < störung, If[RandomReal[] ≤ 0.5, 0, 1], #] &, ImageData[#], {2}]], (größe - 1) / 2],
     ImageSize → ImageDimensions[#]]], {{größe, 3, "Medianfenster"}, 1, 9, 2, Appearance → "Labeled"},
   {{störung, 0.1, "Salz- und Pfeffer-Anteil"}, 0., 1., 0.05, Appearance → "Labeled"}, ContinuousAction → False,
   SaveDefinitions → True] &[ImageCrop[phasenkontrast, {768, 384}]]
```

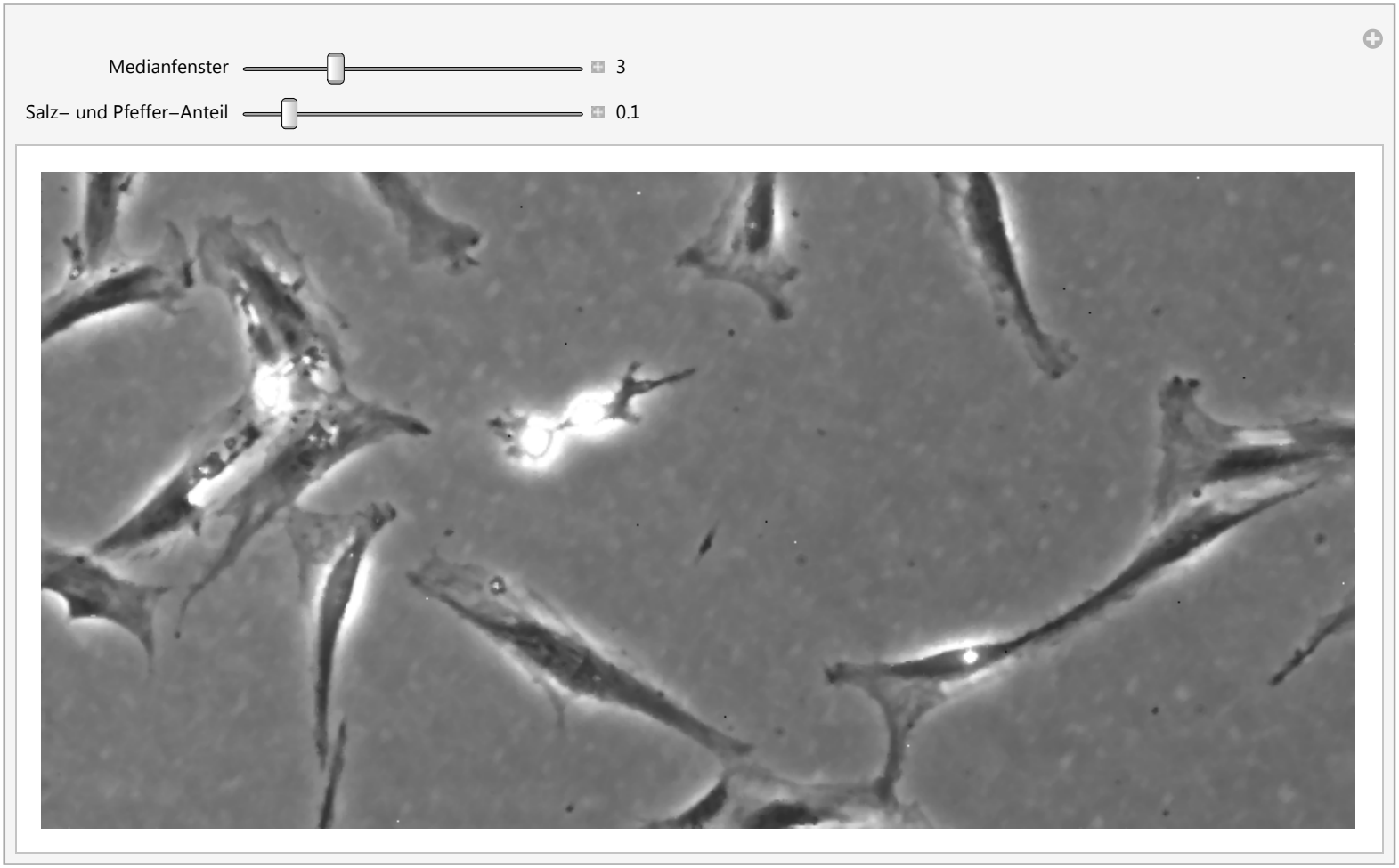
Medianfenster 3

Salz– und Pfeffer–Anteil 0.1



Medianfilter mit quadratischem Strukturelement (Boxmaske), diskrete Lösung 1

```
Manipulate[ControlActive[größe, SeedRandom[2405]; Show[
      Image[Partition[Map[Median[Flatten[#]] &, Flatten[Partition[Map[If[RandomReal[] < störung, If[RandomReal[] ≤ 0.5, 0, 1], #] &, #, {2}],
            {größe, größe}, {1, 1}, {{(größe + 1) / 2, (größe + 1) / 2}, {(größe + 1) / 2, (größe + 1) / 2}}], 1]], Dimensions[#][[2]]]],
      ImageSize → Reverse@Dimensions[#]]], {{größe, 3, "Medianfenster"}, 1, 9, 2, Appearance → "Labeled"},
   {{störung, 0.1, "Salz- und Pfeffer-Anteil"}, 0., 1., 0.05, Appearance → "Labeled"}, ContinuousAction → False,
   SaveDefinitions → True] &[ImageData@ImageCrop[phasenkontrast, {768, 384}]]
```



Medianfilterdefinition mit frei wählbarem Strukturelement (eigene diskrete Lösung)

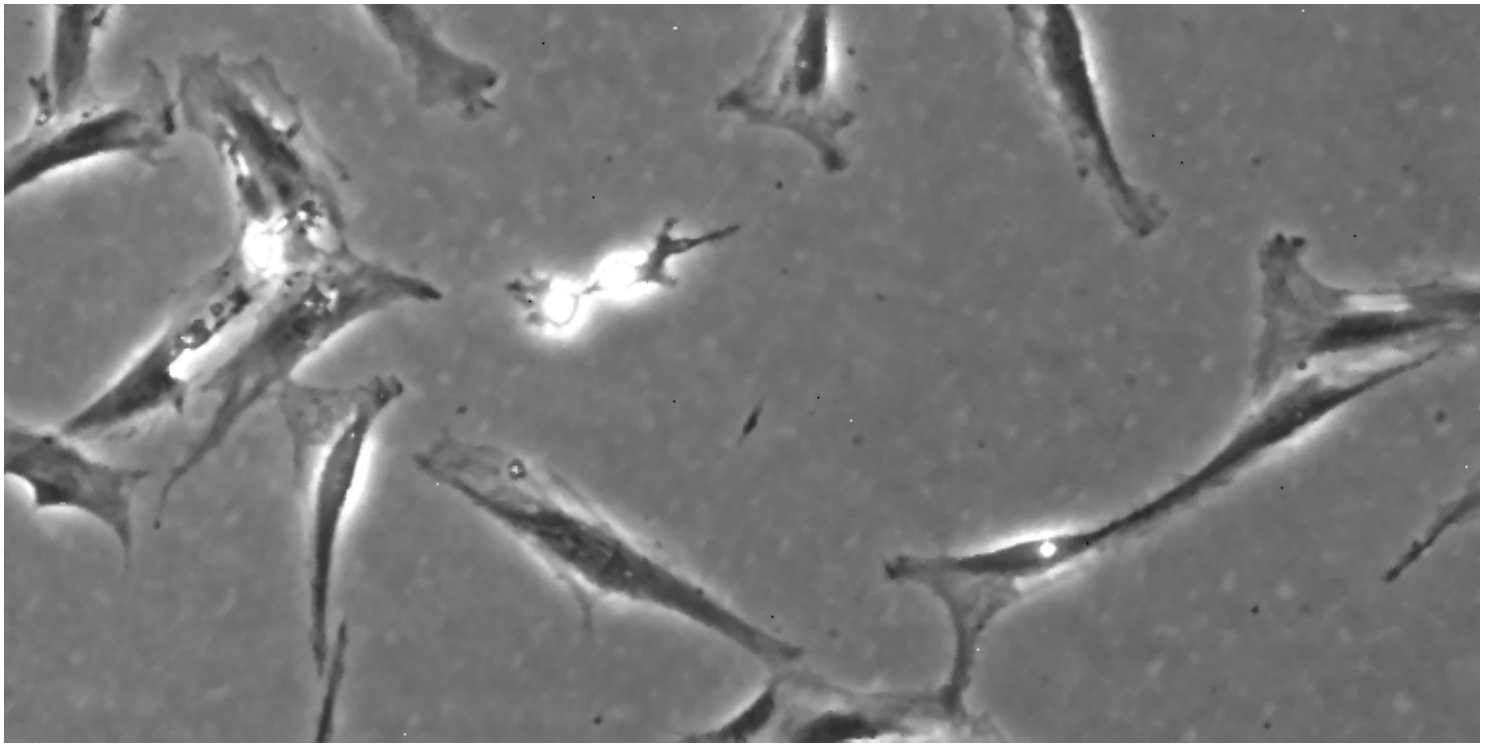
```
Clear[MyMedianFilter];
MyMedianFilter[im_, el_] := Module[{flatelpos},
   flatelpos = Flatten[Position[Flatten[el], 1]];
   Developer`PartitionMap[(Median[Flatten[#, 1][[flatelpos]]]) &, im, Dimensions[el], {1, 1}, Ceiling[Dimensions[el] / 2]]
  ];
```

Medianfilter mit quadratischem Strukturelement (Boxmaske), diskrete Lösung 2

```
Manipulate[ControlActive[größe, SeedRandom[2405];
    Show[Image@MyMedianFilter[Map[If[RandomReal[] < störung, If[RandomReal[] ≤ 0.5, 0, 1], #] &, #, {2}], boxmaske[(größe - 1) / 2]],
     ImageSize → Reverse@Dimensions[#]]], {{größe, 3, "Medianfenster"}, 1, 9, 2, Appearance → "Labeled"},
   {{störung, 0.1, "Salz- und Pfeffer-Anteil"}, 0., 1., 0.05, Appearance → "Labeled"}, ContinuousAction → False,
   SaveDefinitions → True] &[ImageData@ImageCrop[phasenkontrast, {768, 384}]]
```

Medianfenster 3

Salz– und Pfeffer–Anteil 0.1

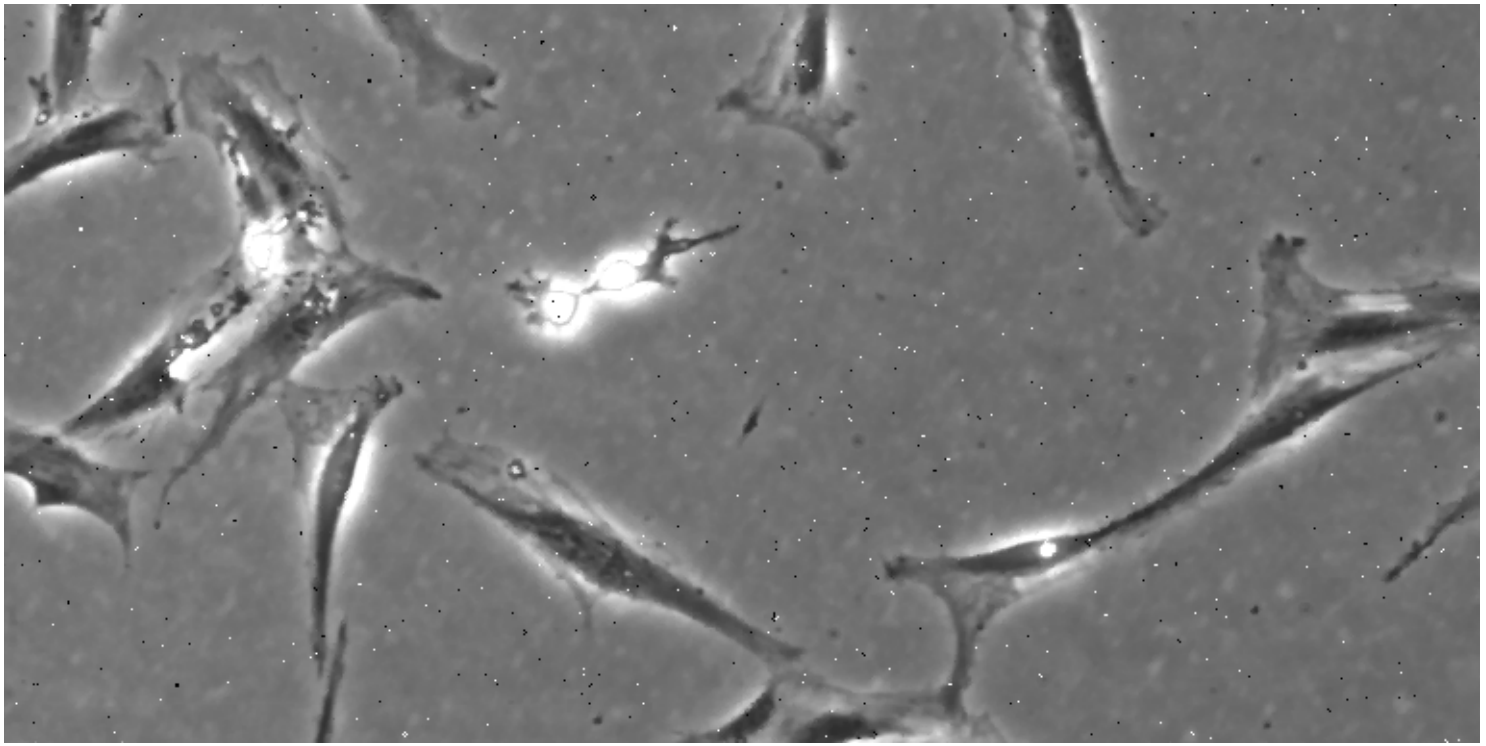


Medianfilter mit kreisförmigem Strukturelement (Kreismaske), diskrete Lösung 2

```
Manipulate[ControlActive[größe, SeedRandom[2405];
    Show[Image@MyMedianFilter[Map[If[RandomReal[] < störung, If[RandomReal[] ≤ 0.5, 0, 1], #] &, #, {2}], kreismaske[(größe - 1) / 2]],
     ImageSize → Reverse@Dimensions[#]]], {{größe, 3, "Medianfenster"}, 1, 9, 2, Appearance → "Labeled"},
   {{störung, 0.1, "Salz- und Pfeffer-Anteil"}, 0., 1., 0.05, Appearance → "Labeled"}, ContinuousAction → False,
   SaveDefinitions → True] &[ImageData@ImageCrop[phasenkontrast, {768, 384}]]
```

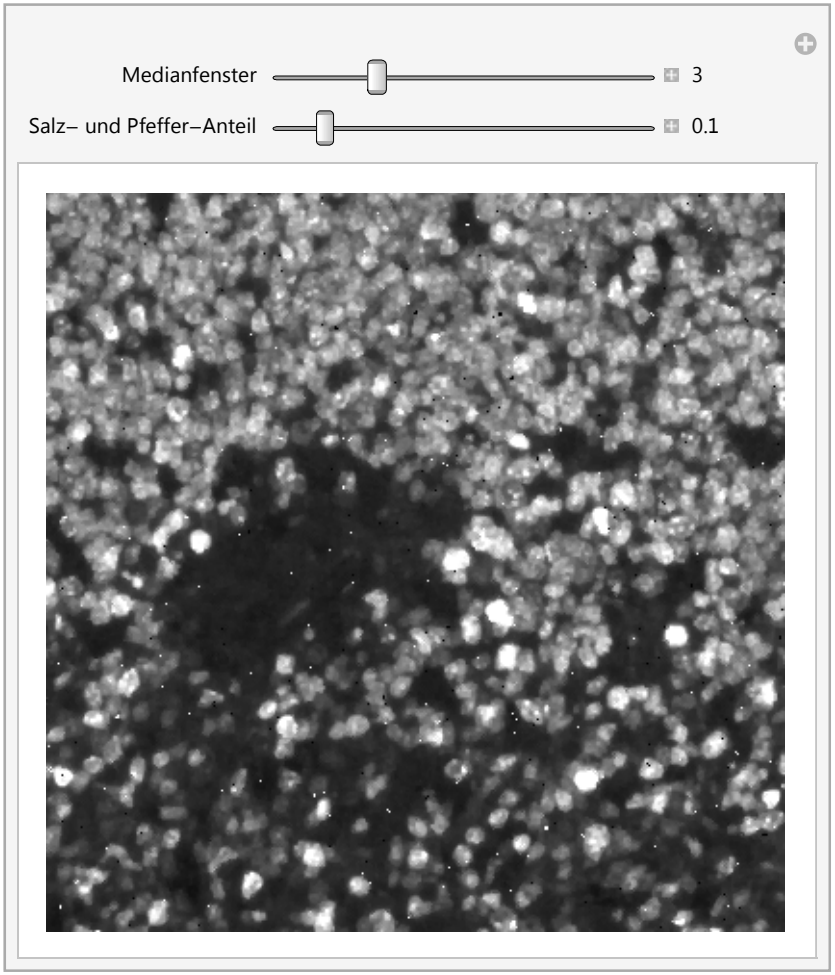
Medianfenster 3

Salz– und Pfeffer–Anteil 0.1



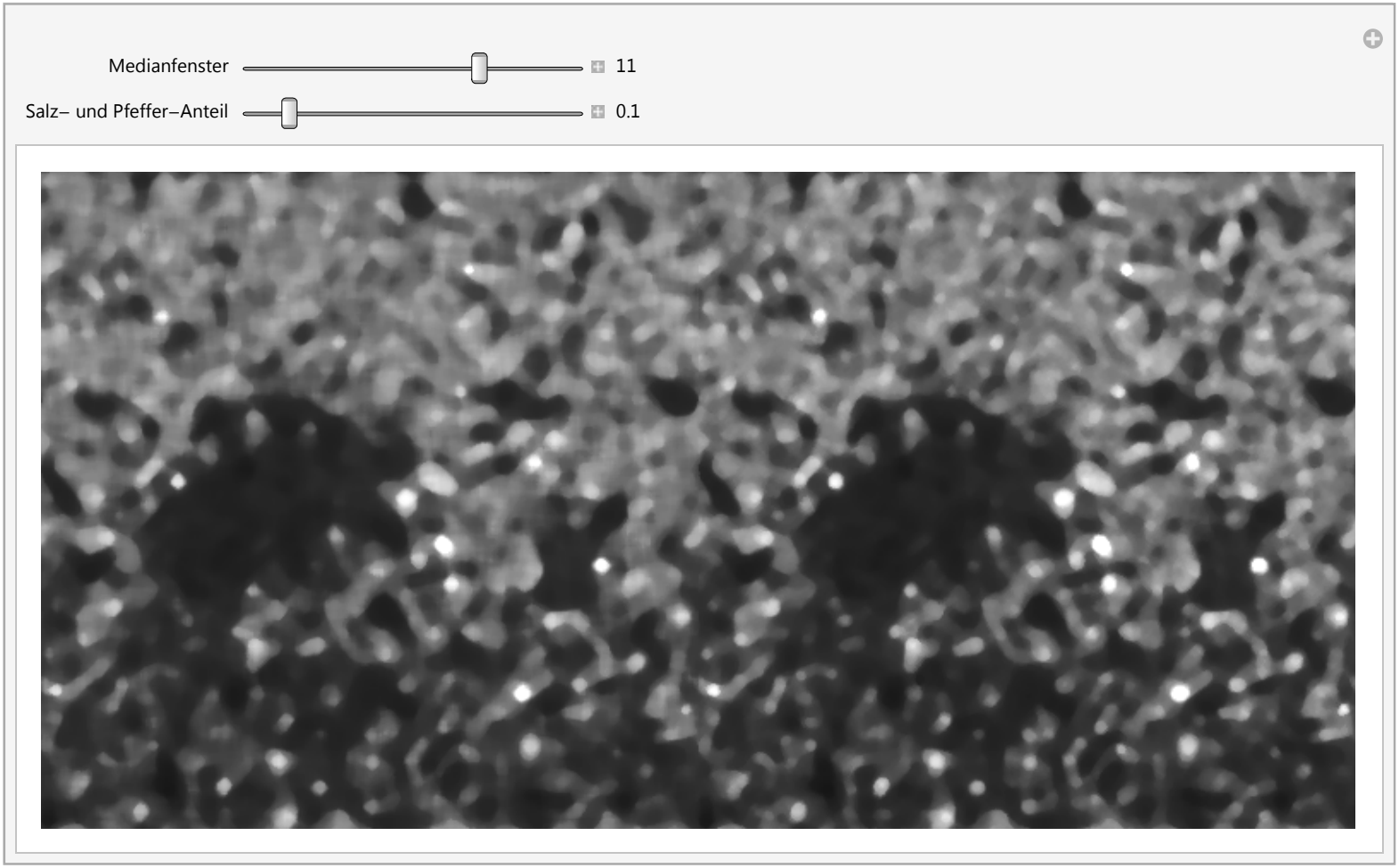
Medianfilter mit kreisförmigem Strukturelement (Kreismaske), diskrete Lösung 2

```
Manipulate[ControlActive[größe, SeedRandom[2405];
    Show[Image@MyMedianFilter[Map[If[RandomReal[] < störung, If[RandomReal[] ≤ 0.5, 0, 1], #] &, #, {2}], kreismaske[(größe - 1) / 2]],
     ImageSize → Reverse@Dimensions[#]]], {{größe, 3, "Medianfenster"}, 1, 9, 2, Appearance → "Labeled"},
   {{störung, 0.1, "Salz- und Pfeffer-Anteil"}, 0., 1., 0.05, Appearance → "Labeled"}, ContinuousAction → False, SaveDefinitions → True] &[
 ImageData@ImageCrop[First@ColorSeparate@Ton3CD21dreikanalausgleichF2, {384, 384}]]
```

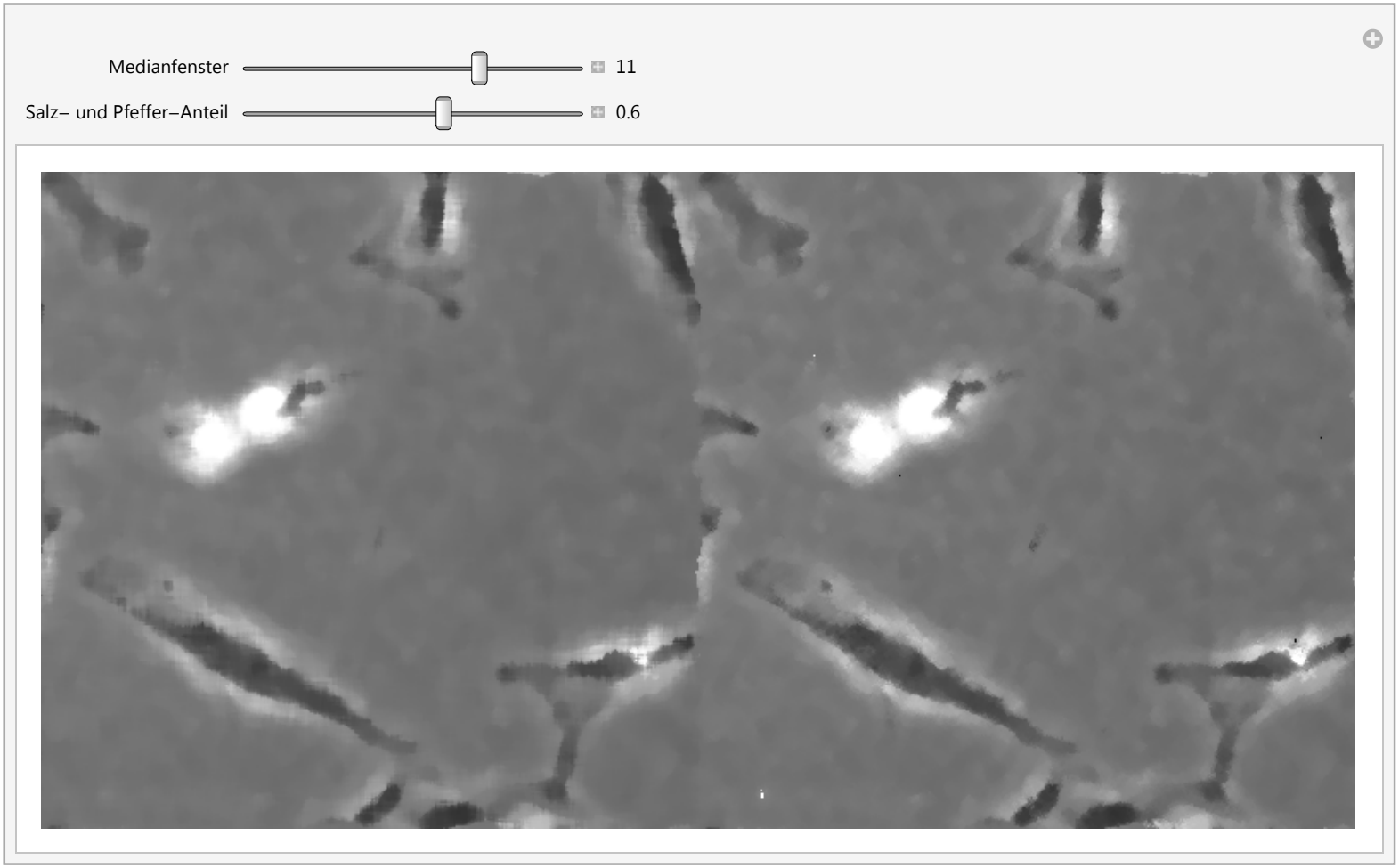


Gegenüberstellung von Medianfilter mit boxförmigem und kreisförmigem Strukturelement

```
Manipulate[ControlActive[größe, Show[ImageAssemble[{SeedRandom[2405]; Image@
         MyMedianFilter[Map[If[RandomReal[] < störung, If[RandomReal[] ≤ 0.5, 0, 1], #] &, #, {2}], boxmaske[(größe - 1) / 2]], SeedRandom[2405];
        Image@MyMedianFilter[Map[If[RandomReal[] < störung, If[RandomReal[] ≤ 0.5, 0, 1], #] &, #, {2}], kreismaske[(größe - 1) / 2]]}],
      ImageSize → {2, 1} * Reverse@Dimensions[#]]], {{größe, 3, "Medianfenster"}, 1, 15, 2, Appearance → "Labeled"},
   {{störung, 0.1, "Salz- und Pfeffer-Anteil"}, 0., 1., 0.05, Appearance → "Labeled"},
   ContinuousAction → False, SaveDefinitions → True] &[
 ImageData@ImageCrop[First@ColorSeparate@Ton3CD21dreikanalausgleichF2, {384, 384}]]
```



```
Manipulate[ControlActive[größe, Show[ImageAssemble[{SeedRandom[2405]; Image@
        MyMedianFilter[Map[If[RandomReal[] < störung, If[RandomReal[] ≤ 0.5, 0, 1], #] &, #, {2}], boxmaske[(größe - 1) / 2]], SeedRandom[2405];
       Image@MyMedianFilter[Map[If[RandomReal[] < störung, If[RandomReal[] ≤ 0.5, 0, 1], #] &, #, {2}], kreismaske[(größe - 1) / 2]]}],
     ImageSize → {2, 1} * Reverse@Dimensions[#]]], {{größe, 3, "Medianfenster"}, 1, 15, 2, Appearance → "Labeled"},
   {{störung, 0.6, "Salz- und Pfeffer-Anteil"}, 0., 1., 0.05, Appearance → "Labeled"}, ContinuousAction → False,
   SaveDefinitions → True] &[ImageData@ImageCrop[phasenkontrast, {384, 384}]]
```



Maximumfilter

“Rangordnungsfilter”: immer das in der aufsteigenden Sortierreihenfolge letzte Element aus Pixelumgebung wird dem Pixel

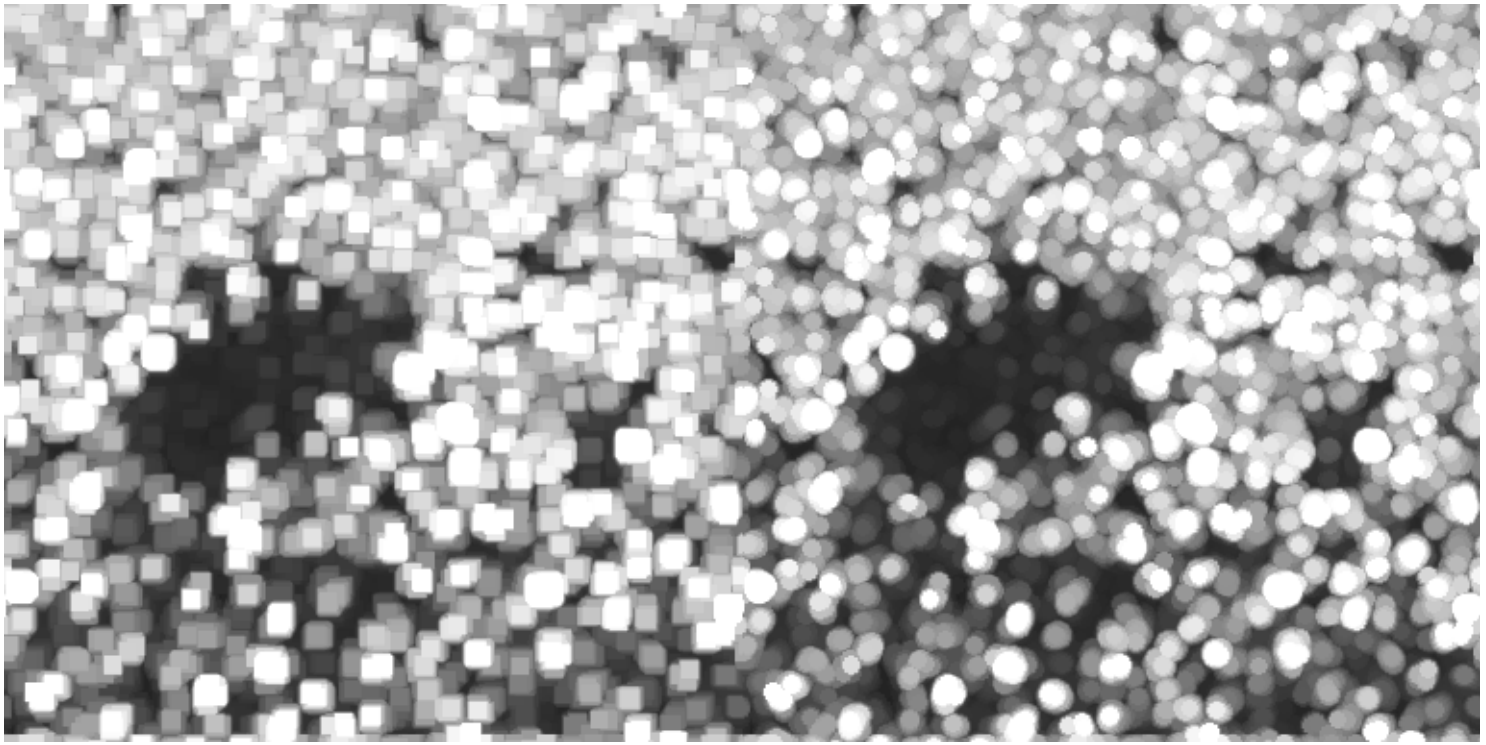
neu zugewiesen

```
Clear[GrayDilate];
GrayDilate = Compile[{{im, _Integer, 2}, {el, _Integer, 2}}, ListConvolve[el, im, {Ceiling[Dimensions[el] / 2]}, im, Times, Max],
   CompilationTarget → "WVM", RuntimeAttributes → {Listable}, Parallelization → False];
```

Maximumfilter mit boxförmigem und kreisförmigem Strukturelement

```
Manipulate[ControlActive[größe,
    Show[ImageAssemble[{Image[GrayDilate[#, boxmaske[(größe - 1) / 2]], "Byte"], Image[GrayDilate[#, kreismaske[(größe - 1) / 2]], "Byte"]}],
     ImageSize → {2, 1} * Reverse@Dimensions[#]]],
   {{größe, 3, "Maximumfenster"}, 1, 15, 2, Appearance → "Labeled"}, ContinuousAction → False, SaveDefinitions → True] &[
 ImageData[ImageCrop[First@ColorSeparate@Ton3CD21dreikanalausgleichF2, {384, 384}], "Byte"]]
```

Maximumfenster 9



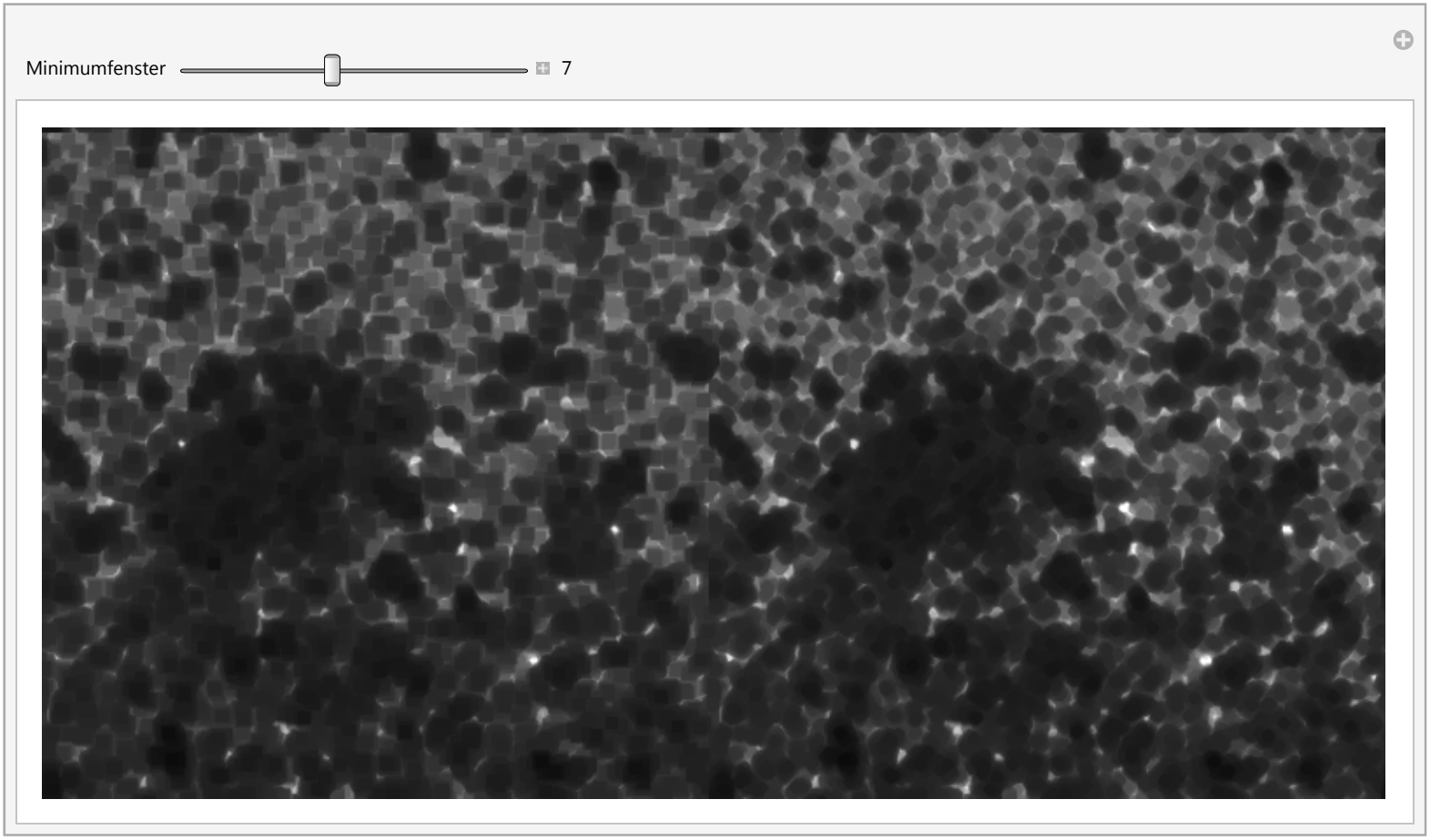
Minimumfilter

“Rangordnungsfilter”: immer das in der aufsteigenden Sortierreihenfolge erste Element aus Pixelumgebung wird dem Pixel neu zugewiesen

```
Clear[GrayErode];
GrayErode = Compile[{{im, _Integer, 2}, {el, _Integer, 2}},
    Module[{negim = 255 - im}, 255 - ListConvolve[el, negim, {Ceiling[Dimensions[el] / 2]}, negim, Times, Max]],
    CompilationTarget → "WVM", RuntimeAttributes → {Listable}, Parallelization → False];
```

Minimumfilter mit boxförmigem und kreisförmigem Strukturelement

```
Manipulate[ControlActive[größe,
    Show[ImageAssemble[{Image[GrayErode[#, boxmaske[(größe - 1) / 2]], "Byte"], Image[GrayErode[#, kreismaske[(größe - 1) / 2]], "Byte"]}],
     ImageSize → {2, 1} * Reverse@Dimensions[#]]],
   {{größe, 3, "Minimumfenster"}, 1, 15, 2, Appearance → "Labeled"}, ContinuousAction → False, SaveDefinitions → True] &[
 ImageData[ImageCrop[First@ColorSeparate@Ton3CD21dreikanalausgleichF2, {384, 384}], "Byte"]]
```



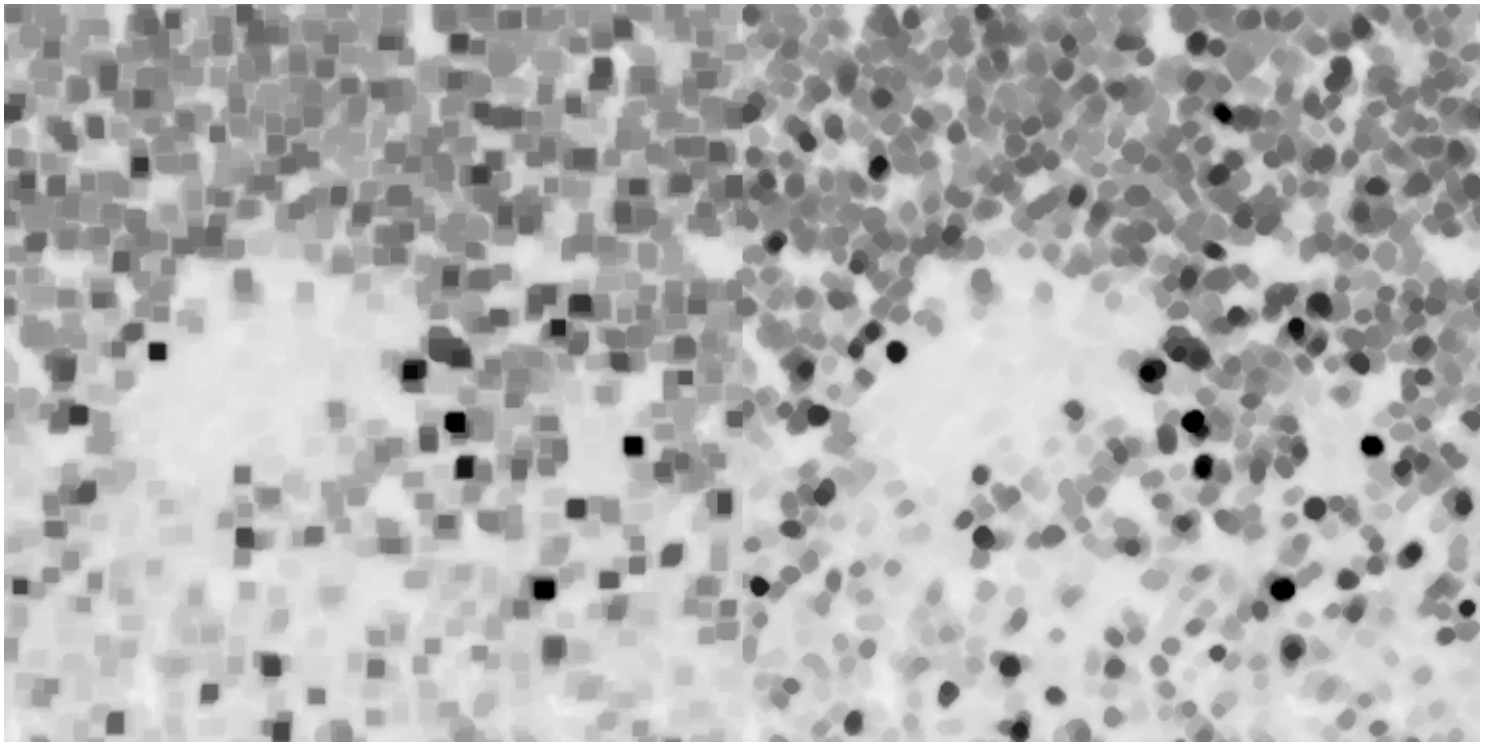
Gray-Scale-Closing: Minimumfilterung gefolgt auf Maximumfilterung

```
Clear[GrayClosing];
GrayClosing[im_, el_] := GrayErode[GrayDilate[im, el], el];
```

Gray-Scale-Closing mit boxförmigem und kreisförmigem Strukturelement (hier auf dem invertierten Bild)

```
Manipulate[ControlActive[größe,
    Show[ImageAssemble[{Image[GrayClosing[#, boxmaske[(größe - 1) / 2]], "Byte"], Image[GrayClosing[#, kreismaske[(größe - 1) / 2]], "Byte"]}],
     ImageSize → {2, 1} * Reverse@Dimensions[#]]],
   {{größe, 3, "Closingfenster"}, 1, 15, 2, Appearance → "Labeled"}, ContinuousAction → False, SaveDefinitions → True] &[
 ImageData[ColorNegate@ImageCrop[First@ColorSeparate@Ton3CD21dreikanalausgleichF2, {384, 384}], "Byte"]]
```

Closingfenster 7



14. Nichtlineare Filter (Fortsetzung)

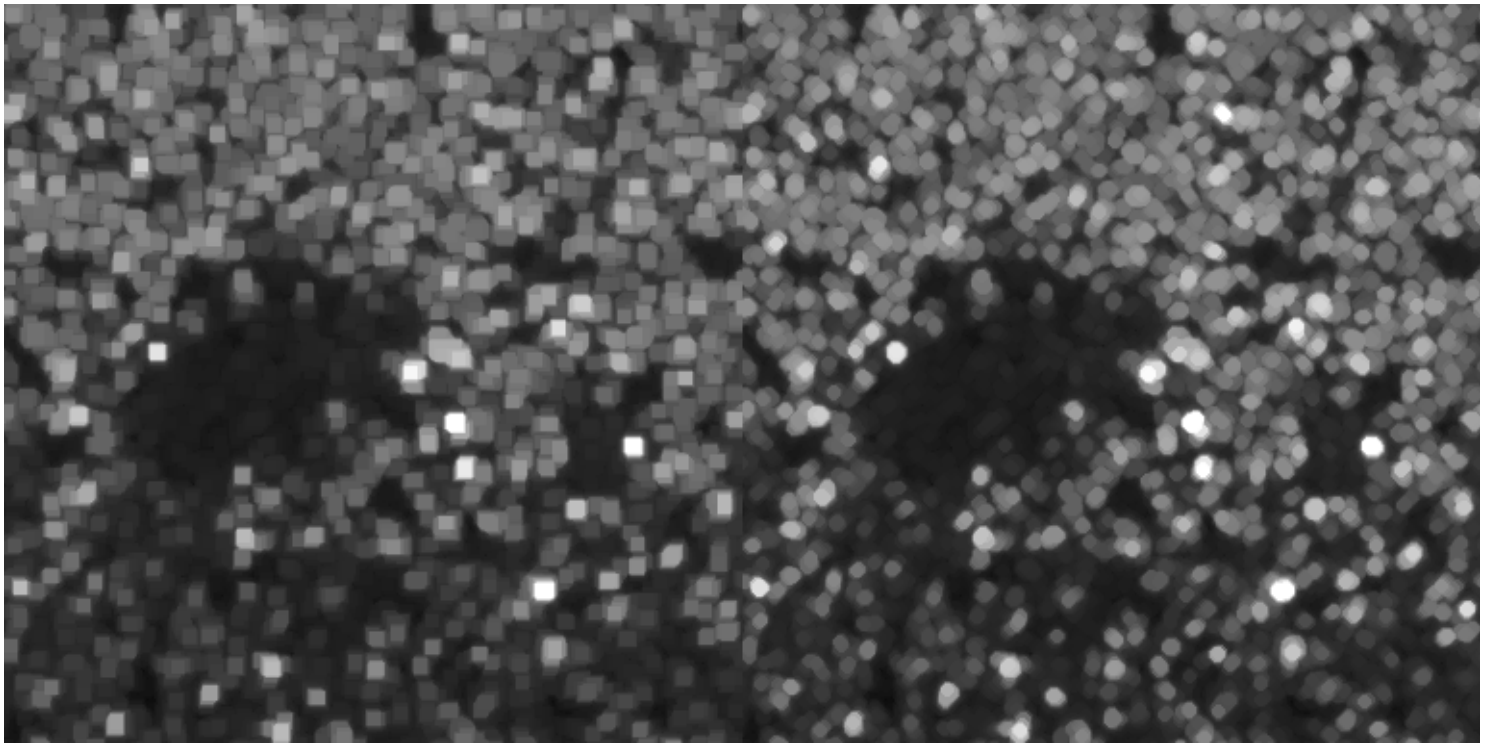
Gray-Scale-Opening: Maximumfilterung gefolgt auf Minimumfilterung

```
Clear[GrayOpening];
GrayOpening[im_, el_] := GrayDilate[GrayErode[im, el], el];
```

Gray-Scale-Opening mit boxförmigem und kreisförmigem Strukturelement

```
Manipulate[ControlActive[größe,
    Show[ImageAssemble[{Image[GrayOpening[#, boxmaske[(größe - 1) / 2]], "Byte"], Image[GrayOpening[#, kreismaske[(größe - 1) / 2]], "Byte"]}],
     ImageSize → {2, 1} * Reverse@Dimensions[#]]],
   {{größe, 3, "Openingfenster"}, 1, 15, 2, Appearance → "Labeled"}, ContinuousAction → False, SaveDefinitions → True] &[
 ImageData[ImageCrop[First@ColorSeparate@Ton3CD21dreikanalausgleichF2, {384, 384}], "Byte"]]
```

Openingfenster 7



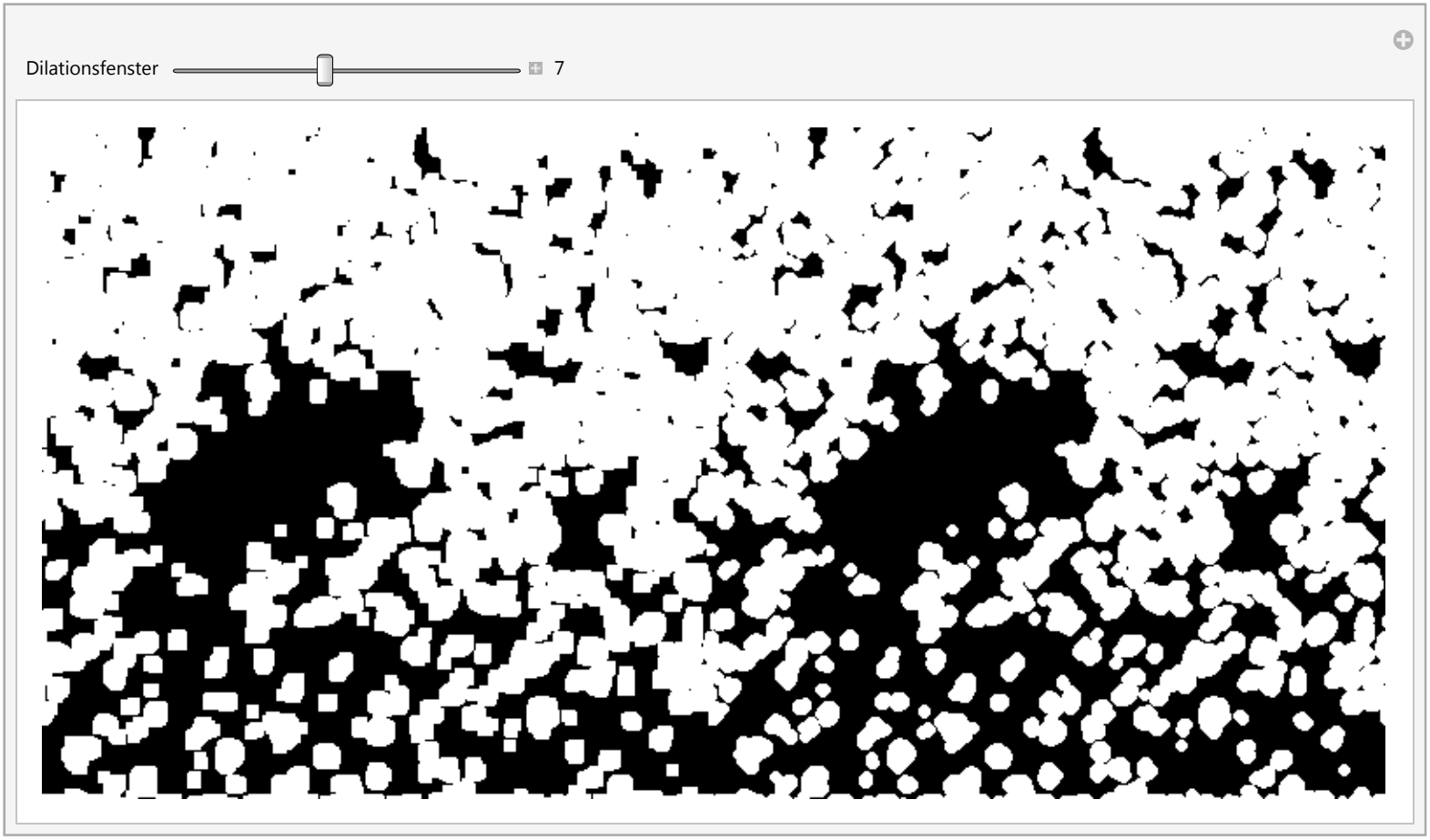
Dilationsfilter, eine vereinfachte Anwendung der Maximumfilter auf Binärbilder

- die Umgebung jedes Binärpixels wird entsprechend des verwendeten Strukturelements erweitert

```
Clear[BinaryDilate];
BinaryDilate = Compile[{{im, _Integer, 2}, {el, _Integer, 2}}, ListConvolve[el, im, {Ceiling[Dimensions[el] / 2]}, im, BitAnd, BitOr],
   CompilationTarget → "WVM", RuntimeAttributes → {Listable}, Parallelization → True];
```

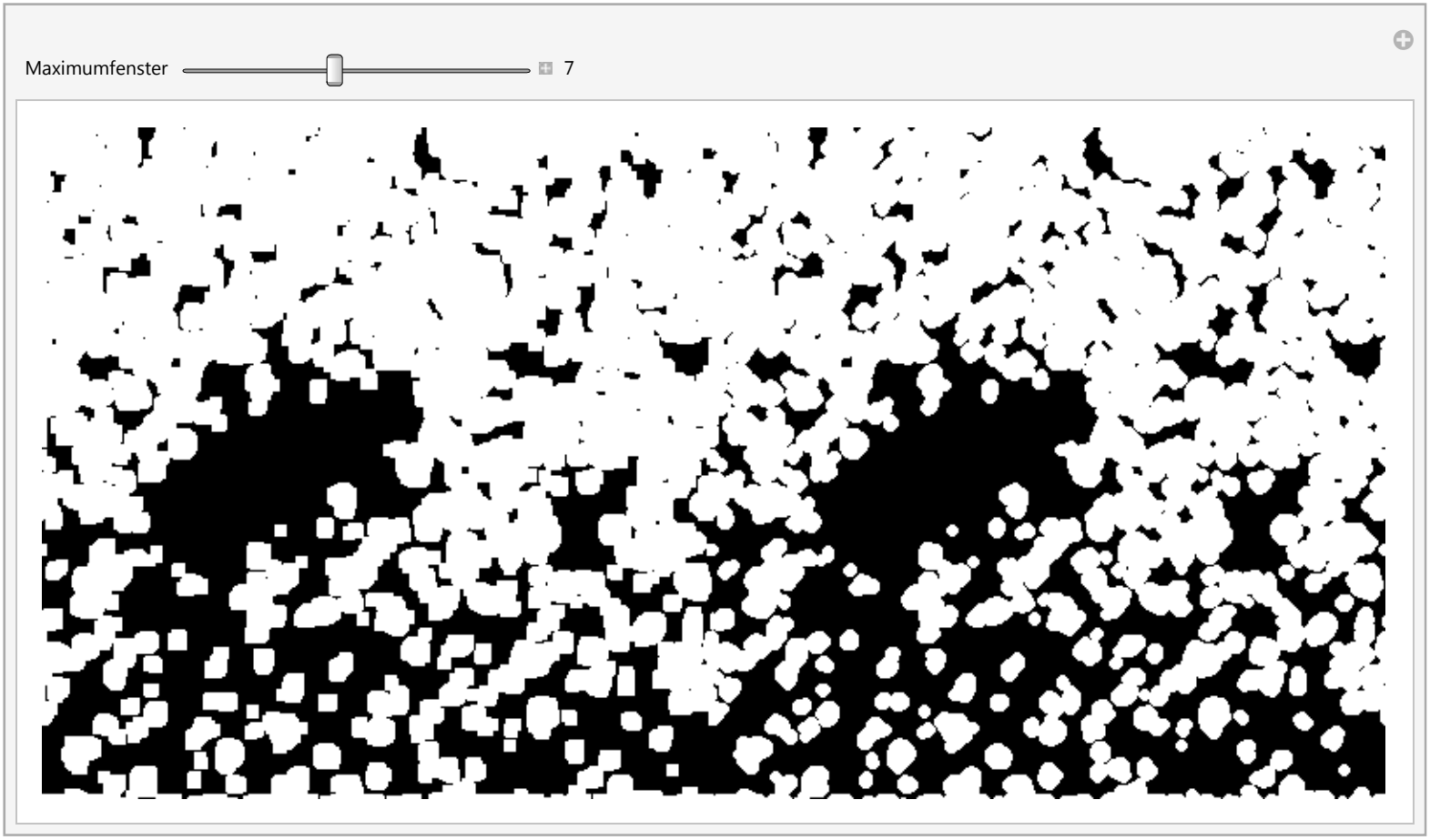
Binärdilation mit boxförmigem und kreisförmigem Strukturelement

```
Manipulate[ControlActive[größe,
    Show[ImageAssemble[{Image[BinaryDilate[#, boxmaske[(größe - 1) / 2]], "Bit"], Image[BinaryDilate[#, kreismaske[(größe - 1) / 2]], "Bit"]}],
     ImageSize → {2, 1} * Reverse@Dimensions[#]]],
   {{größe, 3, "Dilationsfenster"}, 1, 15, 2, Appearance → "Labeled"}, ContinuousAction → False, SaveDefinitions → True] &[
 ImageData[ImageCrop[Binarize[First@ColorSeparate@Ton3CD21dreikanalausgleichF2, 1 / 2], {384, 384}], "Bit"]]
```



Beleg für die Äquivalenz der Binärdilation und der Maximumfilterung von Binärbildern

```
Manipulate[ControlActive[größe,
    Show[ImageAssemble[{Image[GrayDilate[#, boxmaske[(größe - 1) / 2]], "Byte"], Image[GrayDilate[#, kreismaske[(größe - 1) / 2]], "Byte"]}],
     ImageSize → {2, 1} * Reverse@Dimensions[#]]],
   {{größe, 3, "Maximumfenster"}, 1, 15, 2, Appearance → "Labeled"}, ContinuousAction → False, SaveDefinitions → True] &[
 ImageData[ImageCrop[Binarize[First@ColorSeparate@Ton3CD21dreikanalausgleichF2, 1 / 2], {384, 384}], "Byte"]]
```



Erosionsfilter, eine vereinfachte Anwendung der Minimumfilter auf Binärbilder

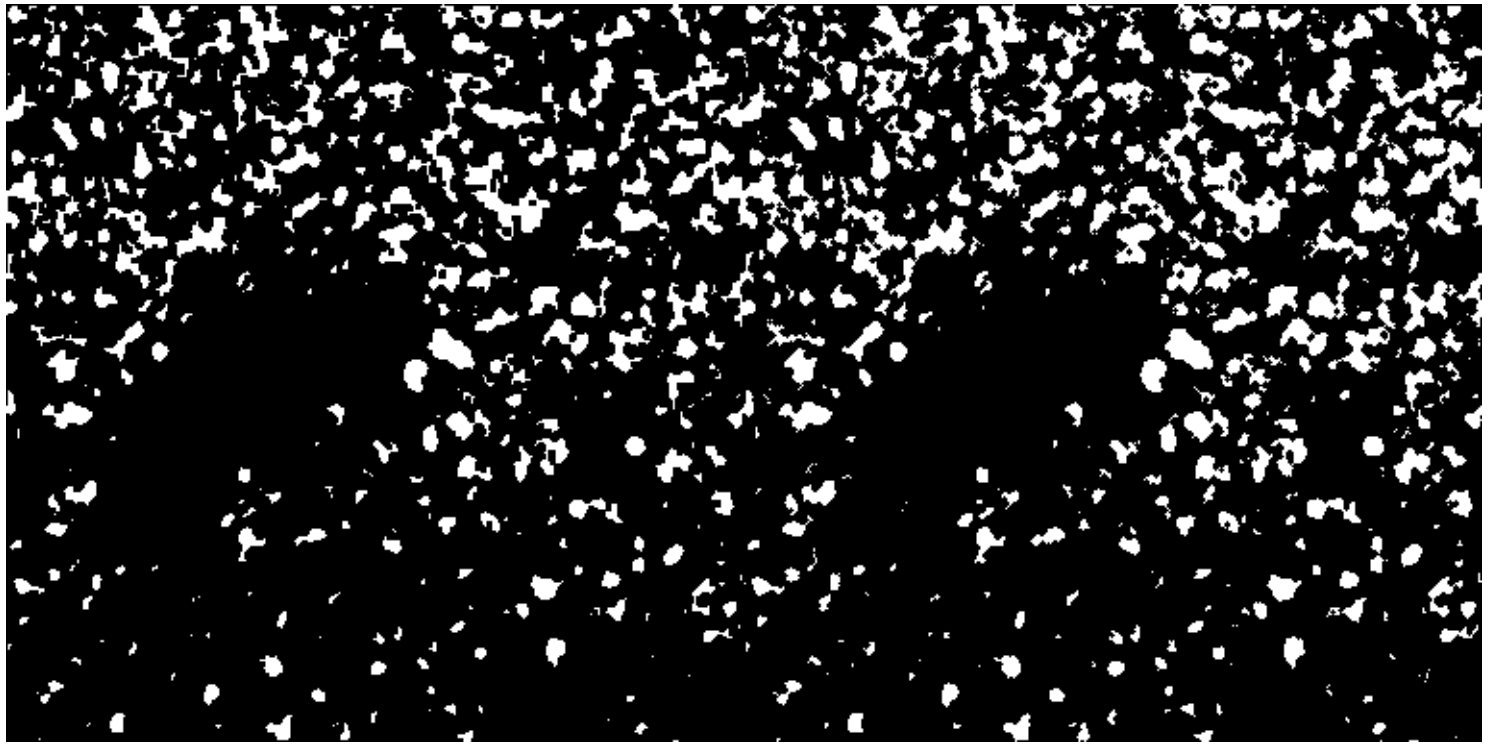
- Binärpixel werden nur belassen, wenn die Umgebung entsprechend des definierten Strukturelements ebenfalls belegt ist

```
Clear[BinaryErode];
BinaryErode = Compile[{{im, _Integer, 2}, {el, _Integer, 2}}, ListConvolve[1 - el, im, {Ceiling[Dimensions[el] / 2]}, im, BitOr, BitAnd],
   CompilationTarget → "WVM", RuntimeAttributes → {Listable}, Parallelization → True];
```

Binärerosion mit boxförmigem und kreisförmigem Strukturelement

```
Manipulate[ControlActive[größe,
    Show[ImageAssemble[{Image[BinaryErode[#, boxmaske[(größe - 1) / 2]], "Bit"], Image[BinaryErode[#, kreismaske[(größe - 1) / 2]], "Bit"]}],
     ImageSize → {2, 1} * Reverse@Dimensions[#]]],
   {{größe, 3, "Erosionsfenster"}, 1, 15, 2, Appearance → "Labeled"}, ContinuousAction → False, SaveDefinitions → True] &[
 ImageData[ImageCrop[Binarize[First@ColorSeparate@Ton3CD21dreikanalausgleichF2, 1 / 2], {384, 384}], "Bit"]]
```

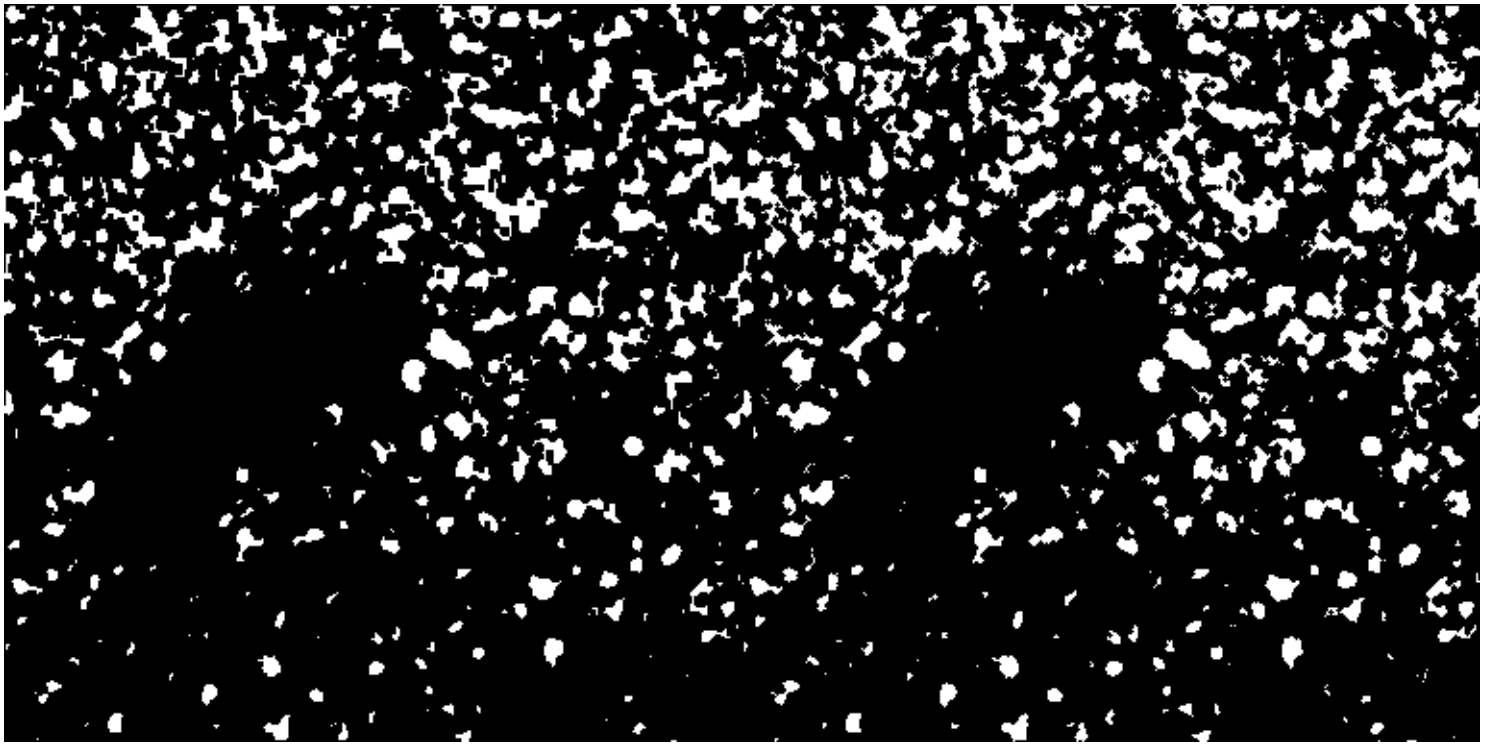
Erosionsfenster 3



Beleg für die Äquivalenz der Binärerosion und der Minimumfilterung von Binärbildern

```
Manipulate[ControlActive[größe,
    Show[ImageAssemble[{Image[GrayErode[#, boxmaske[(größe - 1) / 2]], "Byte"], Image[GrayErode[#, kreismaske[(größe - 1) / 2]], "Byte"]}],
     ImageSize → {2, 1} * Reverse@Dimensions[#]]],
   {{größe, 3, "Minimumfenster"}, 1, 15, 2, Appearance → "Labeled"}, ContinuousAction → False, SaveDefinitions → True] &[
 ImageData[ImageCrop[Binarize[First@ColorSeparate@Ton3CD21dreikanalausgleichF2, 1 / 2], {384, 384}], "Byte"]]
```

Minimumfenster 3

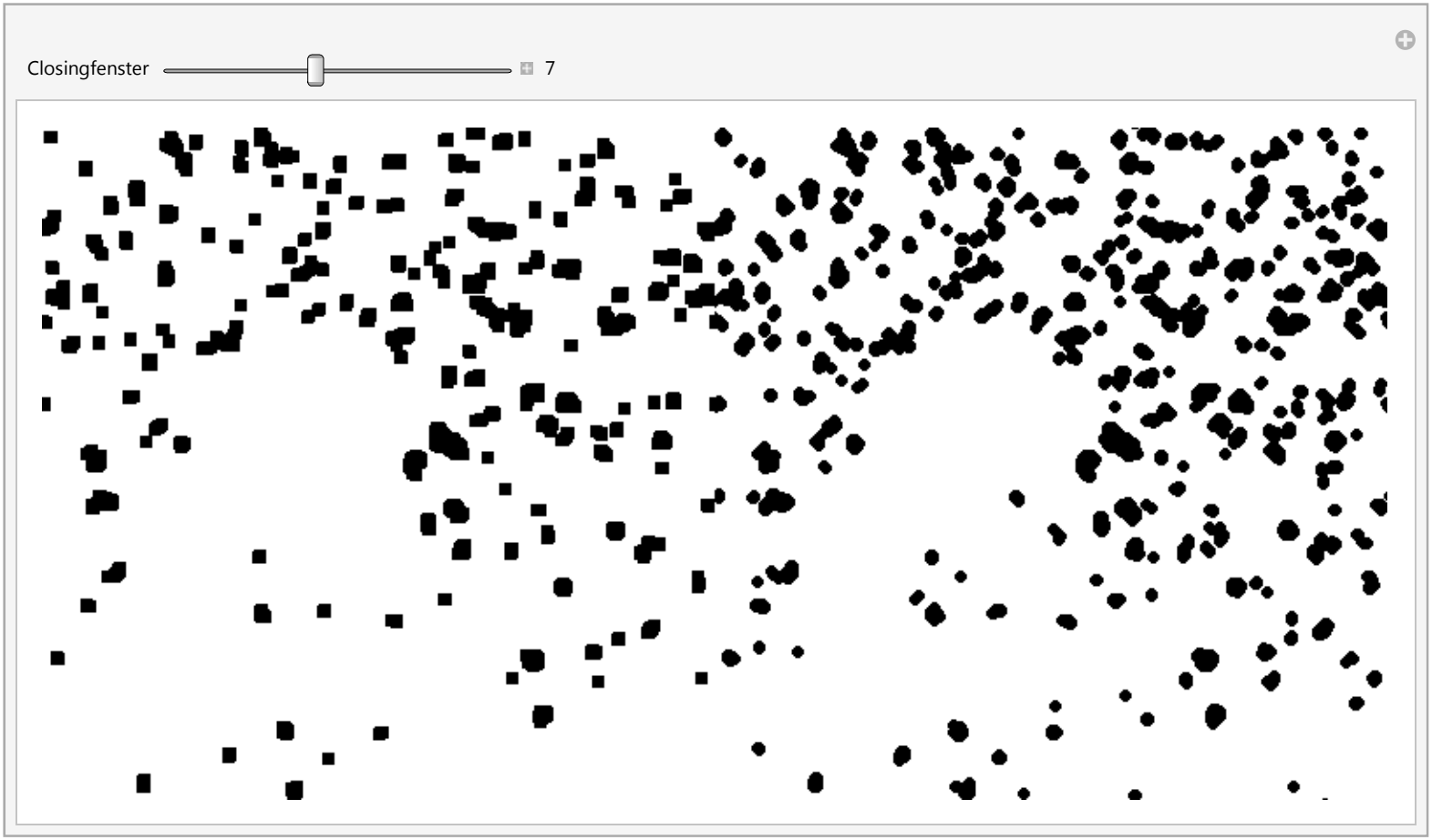


Binär-Closing: Erosion gefolgt auf Dilation

```
Clear[BinaryClosing];
BinaryClosing[im_, el_] := BinaryErode[BinaryDilate[im, el], el]
```

Binär-Closing mit boxförmigem und kreisförmigem Strukturelement (hier invertiert angewandt)

```
Manipulate[ControlActive[größe,
    Show[ImageAssemble[{Image[GrayClosing[#, boxmaske[(größe - 1) / 2]], "Byte"], Image[GrayClosing[#, kreismaske[(größe - 1) / 2]], "Byte"]}],
     ImageSize → {2, 1} * Reverse@Dimensions[#]]],
   {{größe, 3, "Closingfenster"}, 1, 15, 2, Appearance → "Labeled"}, ContinuousAction → False, SaveDefinitions → True] &[
 ImageData[ColorNegate@ImageCrop[Binarize[First@ColorSeparate@Ton3CD21dreikanalausgleichF2, 1 / 2], {384, 384}], "Byte"]]
```



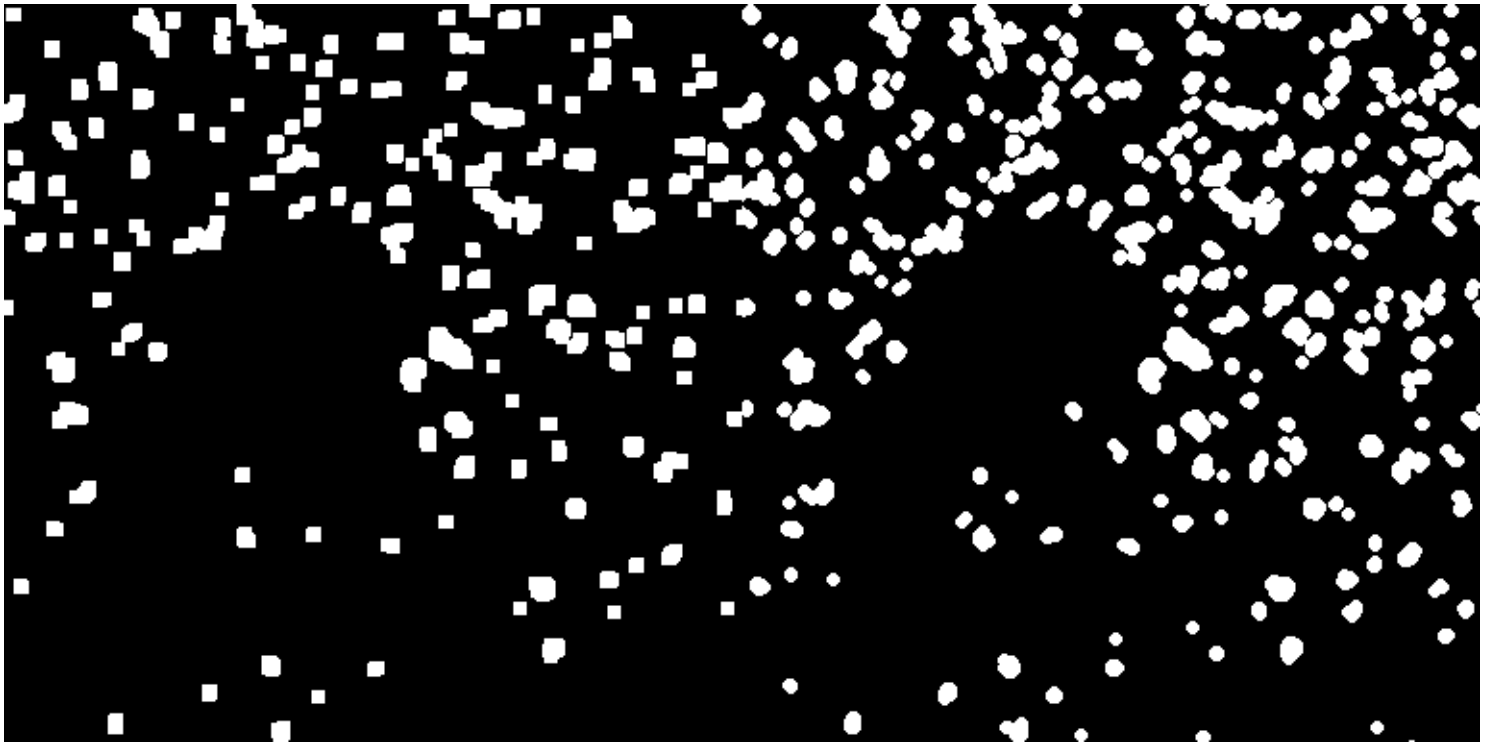
Binär-Opening: Dilation gefolgt auf Erosion

```
Clear[BinaryOpening];
BinaryOpening[im_, el_] := BinaryDilate[BinaryErode[im, el], el]
```

Binär-Opening mit boxförmigem und kreisförmigem Strukturelement

```
Manipulate[ControlActive[größe,
    Show[ImageAssemble[{Image[GrayOpening[#, boxmaske[(größe - 1) / 2]], "Byte"], Image[GrayOpening[#, kreismaske[(größe - 1) / 2]], "Byte"]}],
     ImageSize → {2, 1} * Reverse@Dimensions[#]]],
   {{größe, 3, "Openingfenster"}, 1, 15, 2, Appearance → "Labeled"}, ContinuousAction → False, SaveDefinitions → True] &[
 ImageData[ImageCrop[Binarize[First@ColorSeparate@Ton3CD21dreikanalausgleichF2, 1 / 2], {384, 384}], "Byte"]]
```

Openingfenster 7



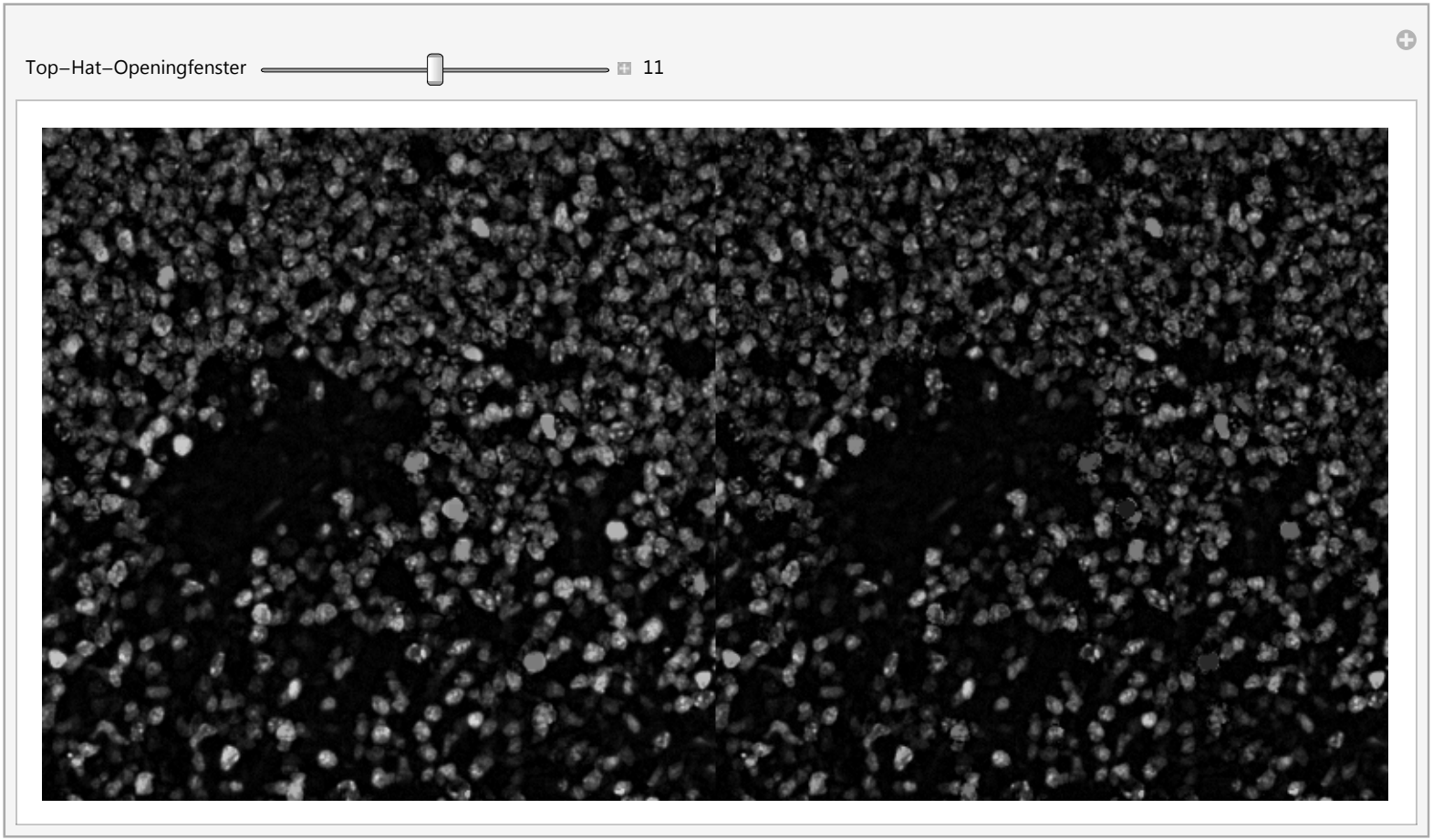
Top-Hat-Transformation als Differenz von Bild und Opening-Bild

Betonung kleiner heller Bilddetails

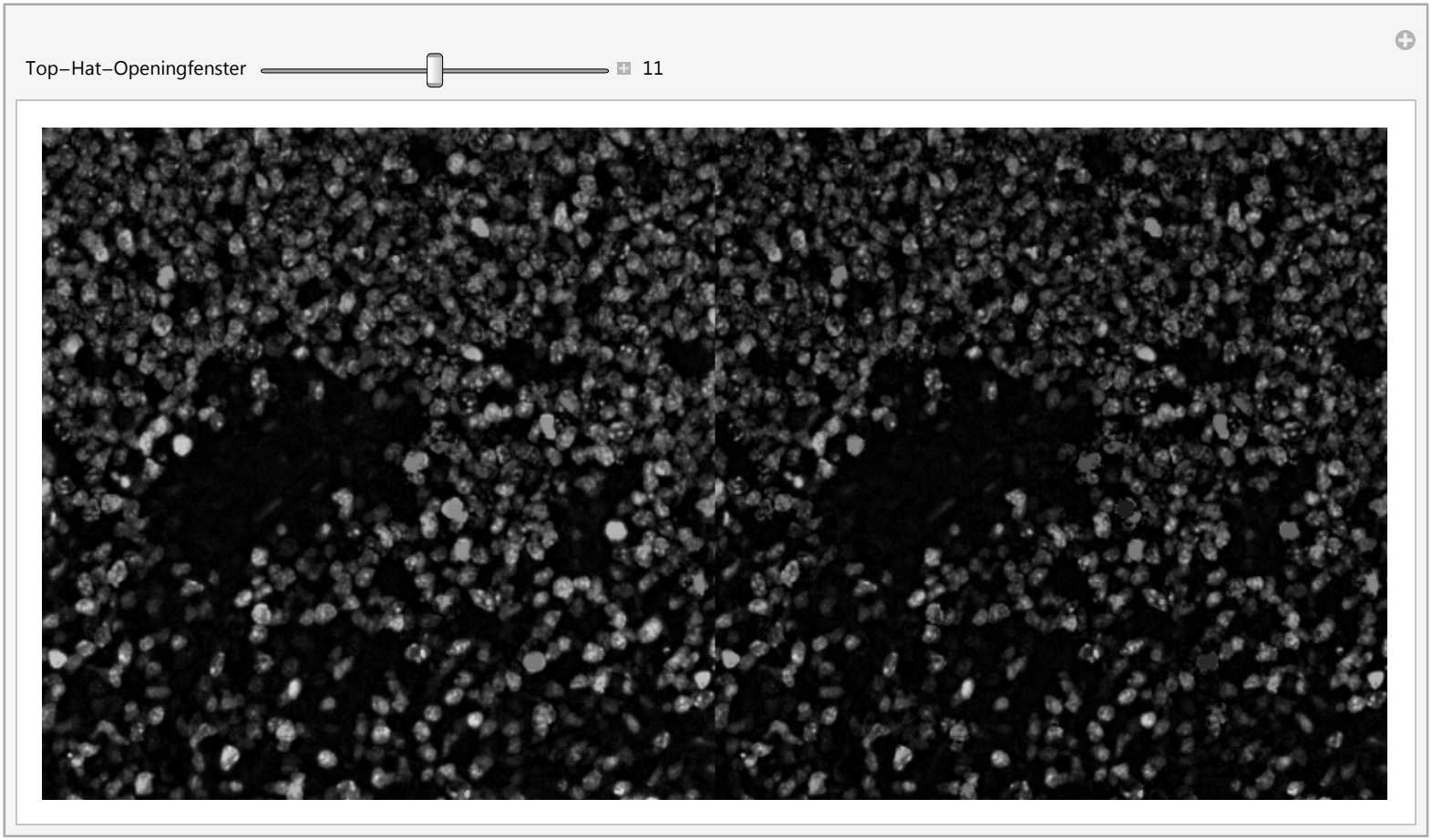
```
Clear[MyTopHatTransform];
MyTopHatTransform[im_, el_] := im - GrayOpening[im, el];
```

Top-Hat-Transformation mit boxförmigem und kreisförmigem Strukturelement

```
Manipulate[ControlActive[größe, Show[ImageAssemble[{Image[MyTopHatTransform[#, boxmaske[(größe - 1) / 2]], "Byte"],
        Image[MyTopHatTransform[#, kreismaske[(größe - 1) / 2]], "Byte"]}], ImageSize → {2, 1} * Reverse@Dimensions[#]]],
   {{größe, 11, "Top-Hat-Openingfenster"}, 1, 21, 2, Appearance → "Labeled"}, ContinuousAction → False, SaveDefinitions → True] &[
 ImageData[ImageCrop[First@ColorSeparate@Ton3CD21dreikanalausgleichF2, {384, 384}], "Byte"]]
```



```
Manipulate[ControlActive[größe, Show[ImageAssemble[{Image[TopHatTransform[#, boxmaske[(größe - 1) / 2]], "Byte"],
        Image[MyTopHatTransform[#, kreismaske[(größe - 1) / 2]], "Byte"]}], ImageSize → {2, 1} * Reverse@Dimensions[#]]],
   {{größe, 11, "Top-Hat-Openingfenster"}, 1, 21, 2, Appearance → "Labeled"}, ContinuousAction → False, SaveDefinitions → True] &[
 ImageData[ImageCrop[First@ColorSeparate@Ton3CD21dreikanalausgleichF2, {384, 384}], "Byte"]]
```



Bottom-Hat-Transformation als Differenz von Closing-Bild und Bild

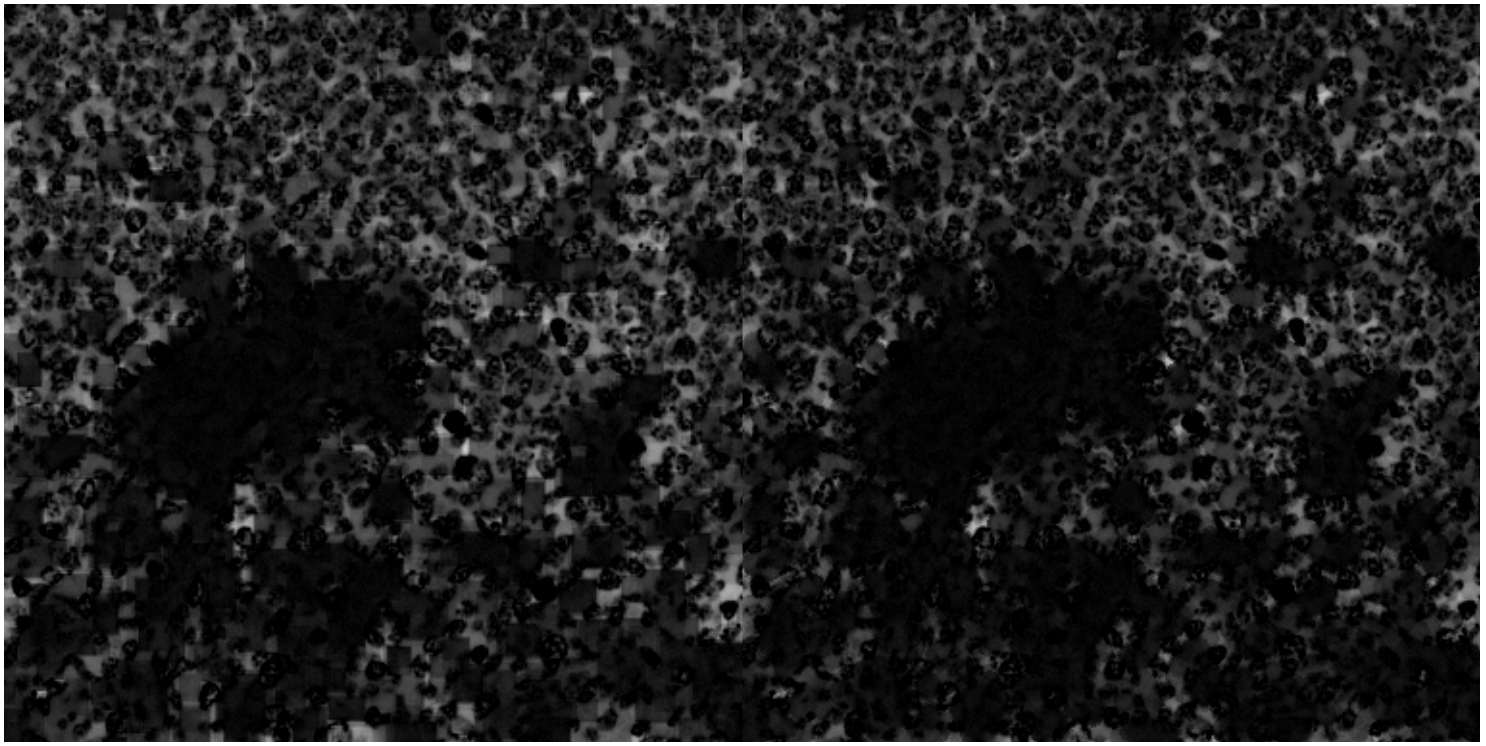
Schwächung kleiner heller Bilddetails

```
Clear[MyBottomHatTransform];
MyBottomHatTransform[im_, el_] := GrayClosing[im, el] - im;
```

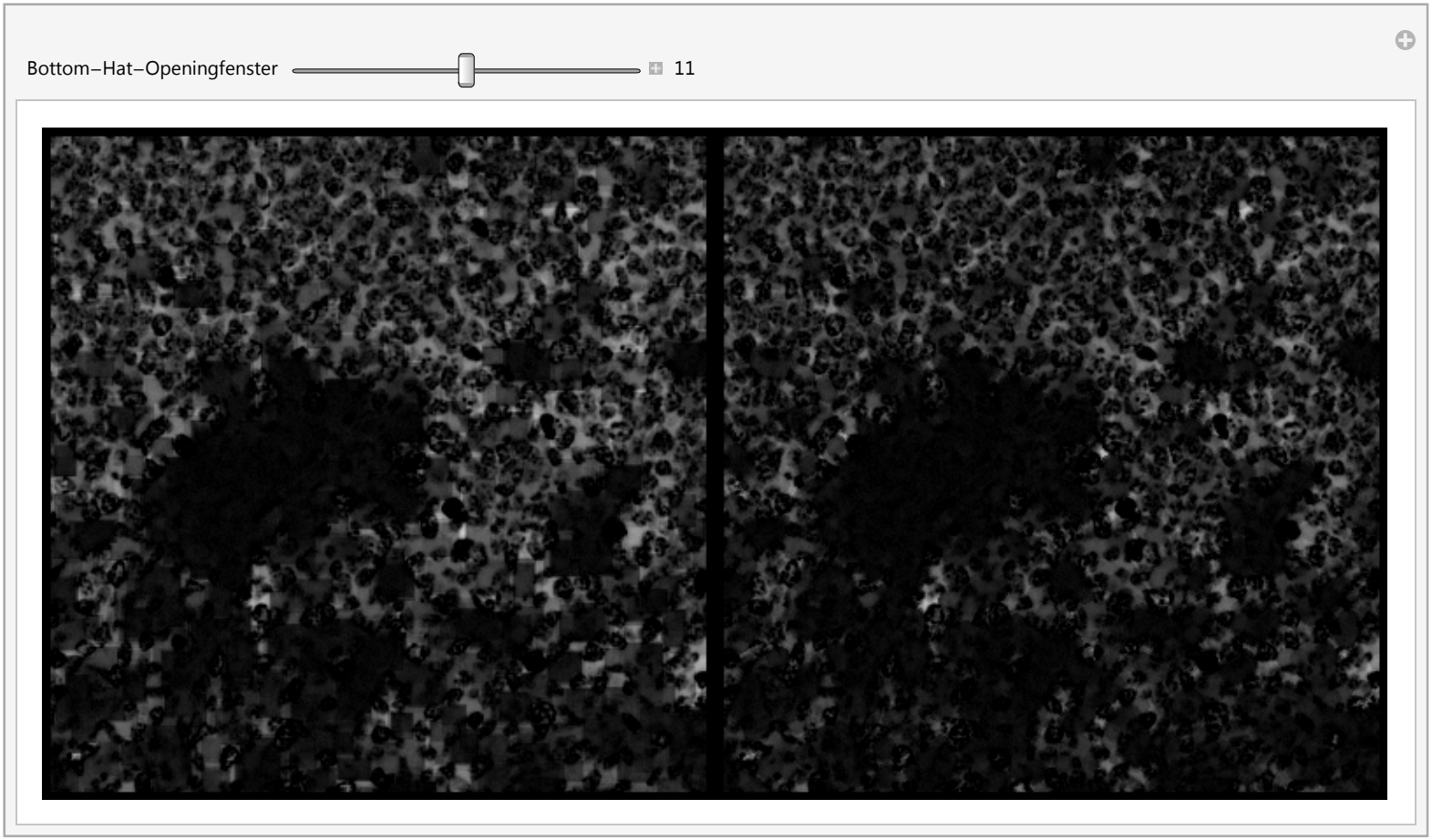
Bottom-Hat-Transformation mit boxförmigem und kreisförmigem Strukturelement

```
Manipulate[ControlActive[größe, Show[ImageAssemble[{Image[MyBottomHatTransform[#, boxmaske[(größe - 1) / 2]], "Byte"],
        Image[MyBottomHatTransform[#, kreismaske[(größe - 1) / 2]], "Byte"]}], ImageSize → {2, 1} * Reverse@Dimensions[#]]],
   {{größe, 11, "Bottom-Hat-Openingfenster"}, 1, 21, 2, Appearance → "Labeled"}, ContinuousAction → False, SaveDefinitions → True] &[
 ImageData[ImageCrop[First@ColorSeparate@Ton3CD21dreikanalausgleichF2, {384, 384}], "Byte"]]
```

Bottom−Hat−Openingfenster 11



```
Manipulate[ControlActive[größe, Show[ImageAssemble[{Image[BottomHatTransform[#, boxmaske[(größe - 1) / 2]], "Byte"],
        Image[BottomHatTransform[#, kreismaske[(größe - 1) / 2]], "Byte"]}], ImageSize → {2, 1} * Reverse@Dimensions[#]]],
   {{größe, 11, "Bottom-Hat-Openingfenster"}, 1, 21, 2, Appearance → "Labeled"}, ContinuousAction → False, SaveDefinitions → True] &[
 ImageData[ImageCrop[First@ColorSeparate@Ton3CD21dreikanalausgleichF2, {384, 384}], "Byte"]]
```



15. Lokale Kontrastanhebung

Lokale Kontrastanhebung in Abhängigkeit von lokaler Standardabweichung

(Narenda und Fitch 1981)

$$f(i,\ j) = m_x(i,\ j) + \frac{\kappa \sum_{i=0}^{M-1} \sum_{j-1}^{N-1} x(i,j)}{\sigma_x(i,j)\, MN} [x(i,\ j) - m_x(i,\ j)]$$

mit

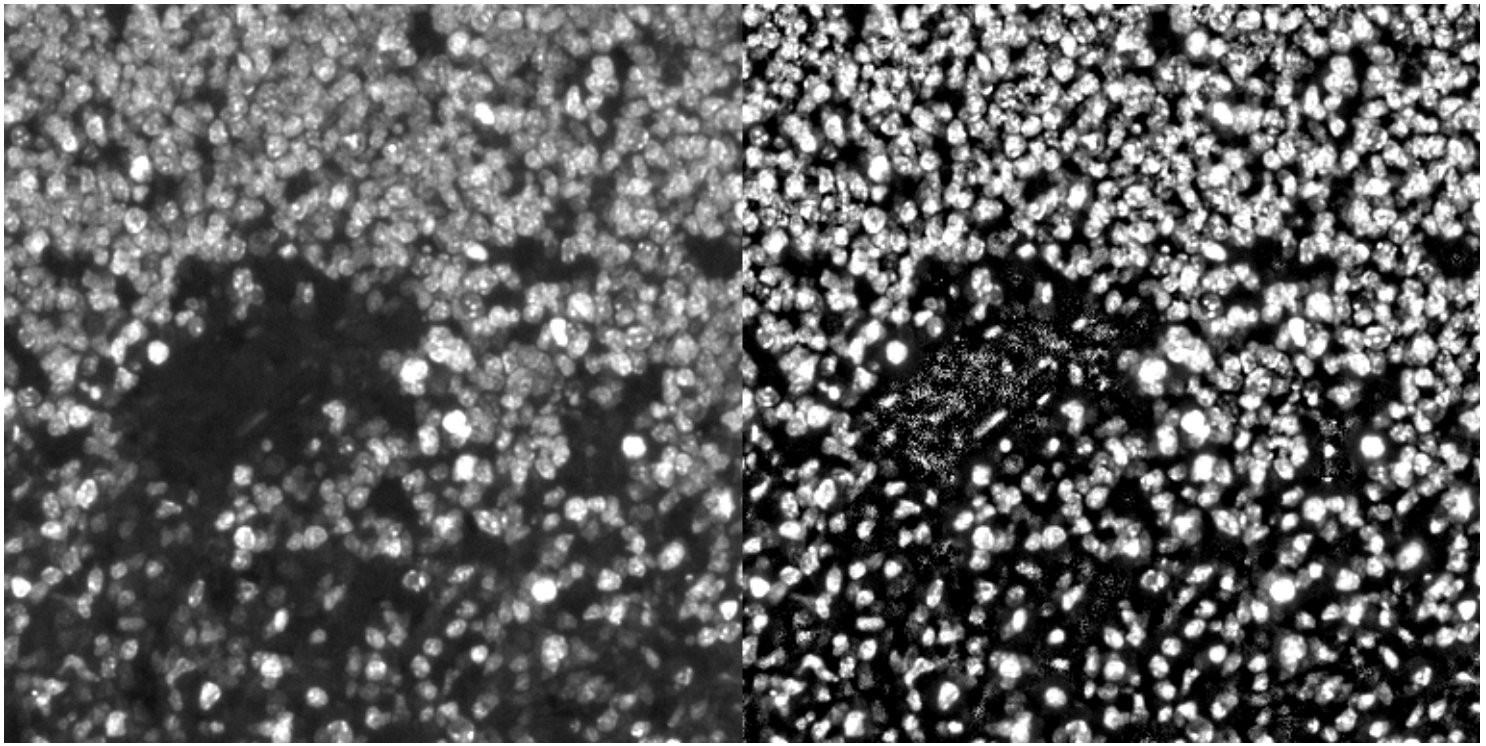
x(i,j): skalarer Bildwert
$\sigma_x(i,\ j)$: lokale Standardabweichung
$m_x(i,\ j)$: lokaler Mittelwert
κ: positiver skalarer Skalierungsparameter
f(i,j): kontrastmodfizierter Bildwert
M,N: Bildabmaße

```
Clear[LocalContrastKern];
LocalContrastKern = Compile[{{list, _Real, 1}, {kappatimesgm, _Real}},
    Module[{mean, stddev},
     mean = Mean[list];
     stddev = $MachineEpsilon;
     If[Length[list] ≠ 1, stddev = StandardDeviation[list] + $MachineEpsilon];
     kappatimesgm / stddev * (list[[Ceiling[Length[list] / 2]]] - mean) + mean
    ], CompilationTarget → "WVM", Parallelization → False];

(*Narenda and Fitch, 1981*)
Clear[MyLocalContrastFilter];
Options[MyLocalContrastFilter] = {"WindowHalfWidth" -> 7, "Scaling" -> Automatic, "Mask" → "Box"};
MyLocalContrastFilter[im_Image, OptionsPattern[]] := Module[{flatelpos, padim, whw, kappa, kappatimesgm, el, imdata},
    whw = Round[OptionValue["WindowHalfWidth"]];
    kappa = OptionValue["Scaling"];
    If[kappa === Automatic, kappa = 1.0, kappa = N[kappa]];
    If[kappa < 0.0 || kappa > 1.0 || ! NumericQ[kappa], kappa = 1.0];
    el = Switch[OptionValue["Mask"],
      "Box", Table[1, {2 whw + 1}, {2 whw + 1}],
      "Circle", Ceiling[Rescale[Sign[Table[(x^2 + y^2), {x, -whw, whw}, {y, -whw, whw}] - whw * (whw + 1 / 2)], {1, -1}]],
      _, Table[1, {2 whw + 1}, {2 whw + 1}]
     ];
    imdata = ImageData[im, "Real"];
    kappatimesgm = kappa * Mean[Flatten[imdata]];
    flatelpos = Flatten[Position[Flatten[el], 1]];
    padim = ArrayPad[imdata, Floor[Dimensions[el] / 2], "Reversed"];
    padim = Developer`PartitionMap[
      (LocalContrastKern[Flatten[#, 1][[flatelpos]], kappatimesgm]) &, padim, Dimensions[el], {1, 1}, Ceiling[Dimensions[el] / 2]];
    Image[ArrayPad[padim, -Floor[Dimensions[el] / 2]], "Real"]
   ];
```

Lokale Kontrastanhebung mit kreisförmigem Strukturelement (prolif. Zellen)

```
Manipulate[ControlActive[größe, Show[ImageAssemble[{#, MyLocalContrastFilter[#, "WindowHalfWidth" → (größe - 1) / 2, "Mask" → "Circle"]}],
      ImageSize → {2, 1} * ImageDimensions[#]]], {{größe, 15, "Fenstergröße"}, 1, 31, 2, Appearance → "Labeled"},
   ContinuousAction → False, SaveDefinitions → True] &[ImageCrop[First@ColorSeparate@Ton3CD21dreikanalausgleichF2, {384, 384}]]
```

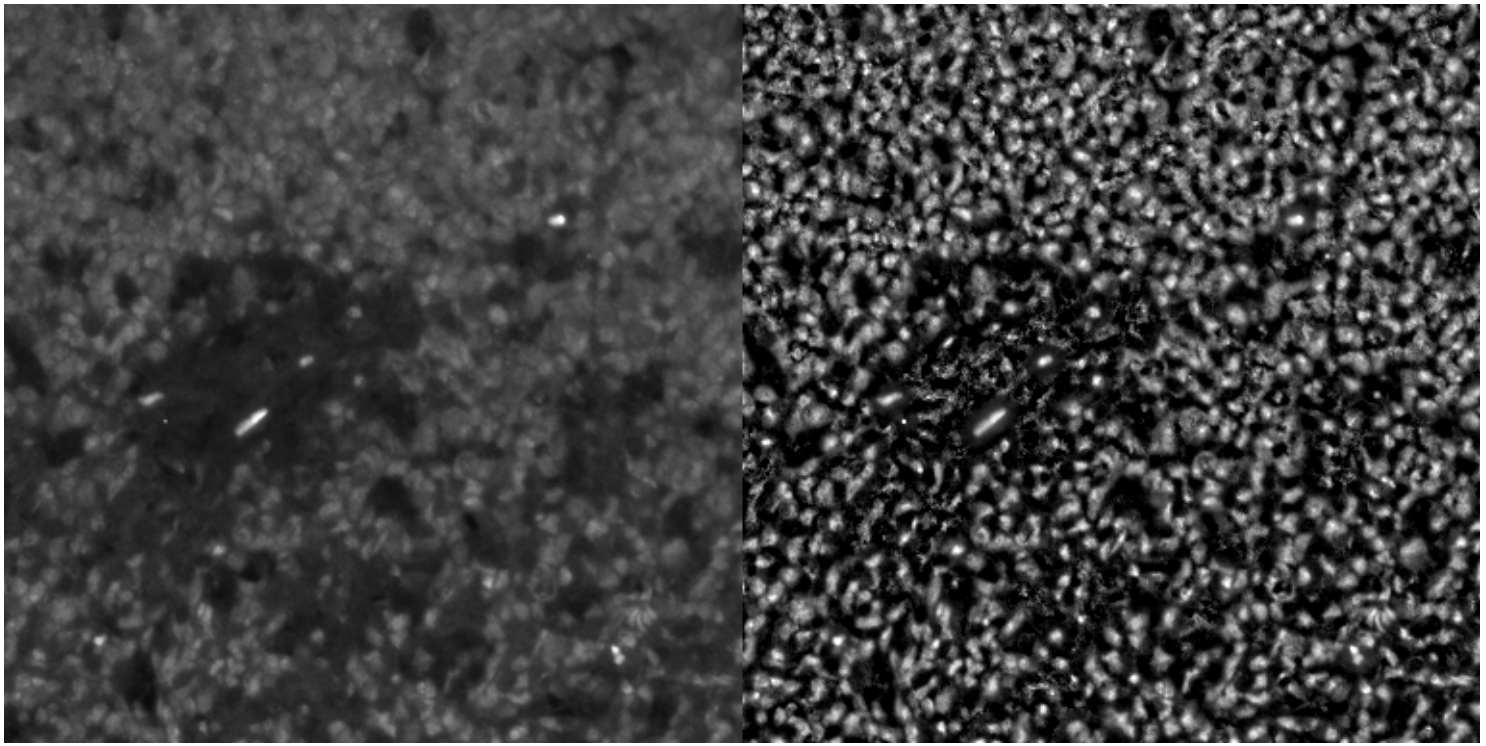
Fenstergröße 19



Lokale Kontrastanhebung mit kreisförmigem Strukturelement (B-Zellkerne)

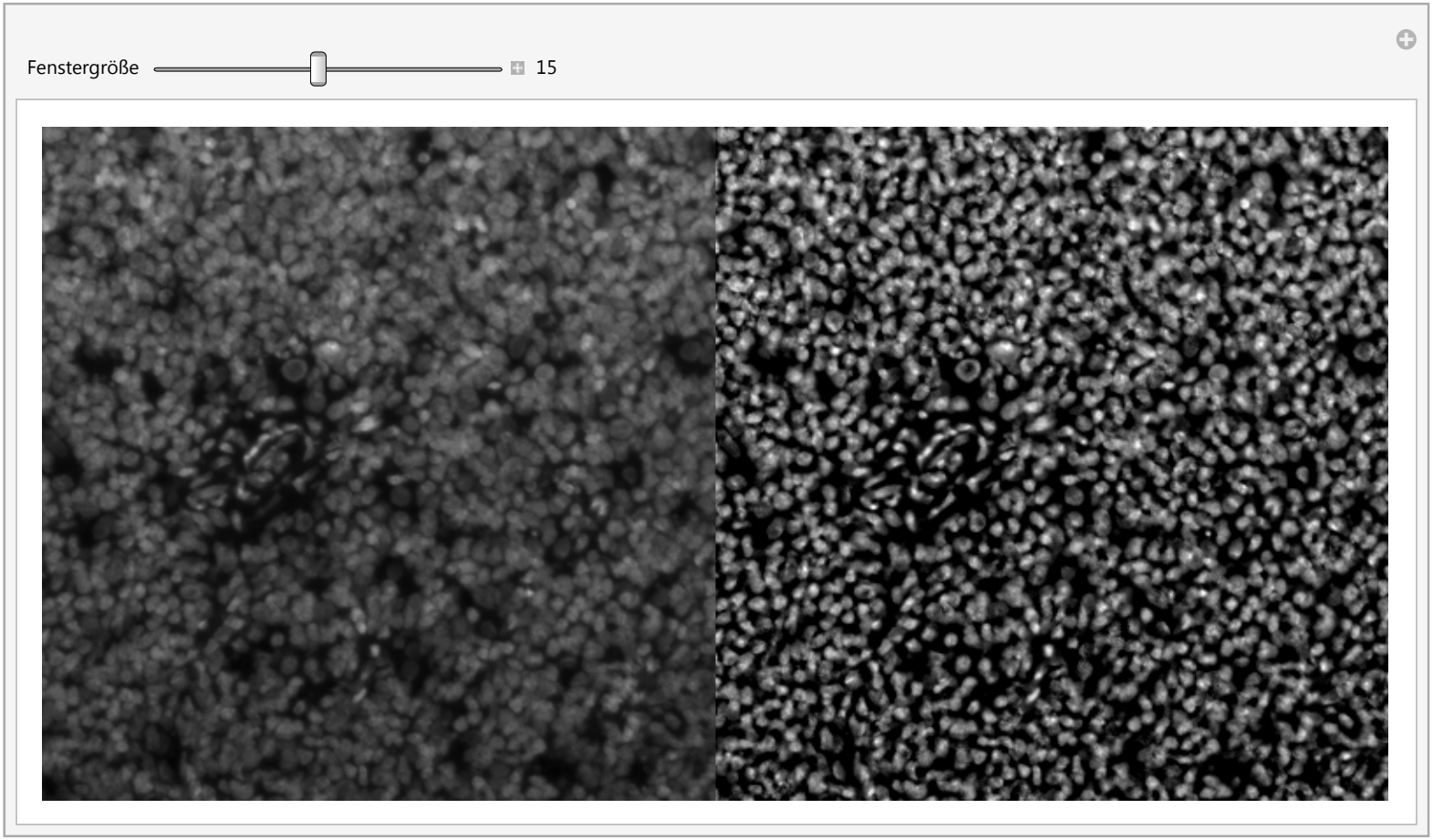
```
Manipulate[ControlActive[größe, Show[ImageAssemble[{#, MyLocalContrastFilter[#, "WindowHalfWidth" → (größe - 1) / 2, "Mask" → "Circle"]}],
    ImageSize → {2, 1} * ImageDimensions[#]]], {{größe, 15, "Fenstergröße"}, 1, 31, 2, Appearance → "Labeled"}, ContinuousAction → False,
  SaveDefinitions → True] &[ImageCrop[First@Rest@ColorSeparate@Ton3CD21dreikanalausgleichF2, {384, 384}]]
```

Fenstergröße 15



Lokale Kontrastanhebung mit kreisförmigem Strukturelement (alle Zellkerne)

```
Manipulate[ControlActive[größe, Show[ImageAssemble[{#, MyLocalContrastFilter[#, "WindowHalfWidth" → (größe - 1) / 2, "Mask" → "Circle"]}],
      ImageSize → {2, 1} * ImageDimensions[#]]], {{größe, 15, "Fenstergröße"}, 1, 31, 2, Appearance → "Labeled"},
   ContinuousAction → False, SaveDefinitions → True] &[ImageCrop[Last@ColorSeparate@Ton3CD21dreikanalausgleichF2, {384, 384}]]
```



16. Binarisierung durch lokale Schwellwertbestimmung

Lokale Schwellwertbildung in Abhängigkeit von Mittel aus lokalem Maximum und Minimum

Bernsen 1986

$$t(i, j) = \frac{max_x + min_x}{2}$$

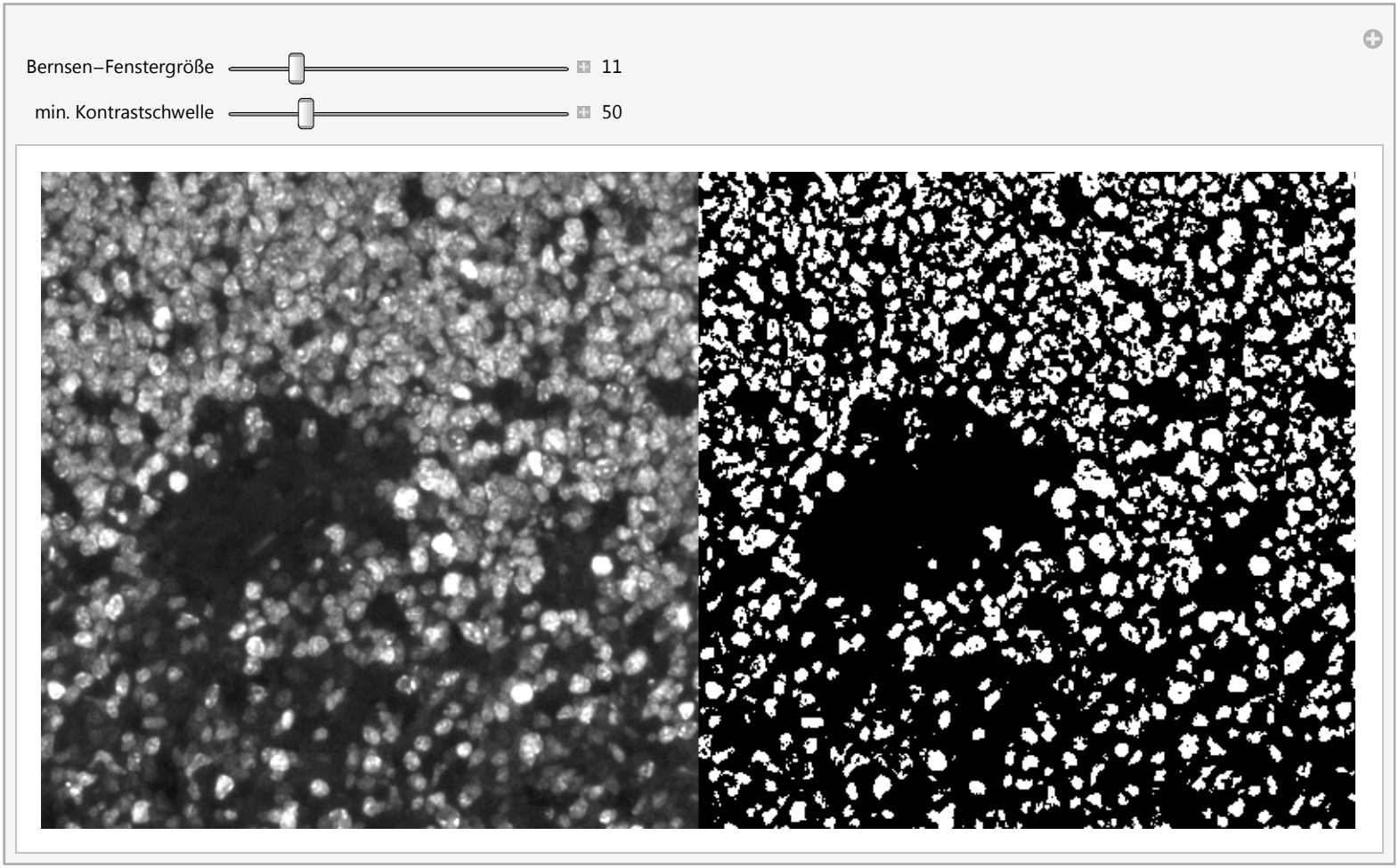
außer falls $(max_x - min_x) < c$, dann Binarisierung mit Ausnahmeregel; c (min. Kontrastschwelle) wählbar, bei 8bit typ. c=15, Umgebungsgröße typ. 15x15

Ausnahmeregel (verbal): falls t(i,j) näher an min_x als an max_x, dann Nullsetzen, sonst Einssetzen.

```
Clear[BernsenKern];
BernsenKern = Compile[{{list, _Real, 1}, {cm, _Real}},
   Module[{min, max, diff, sum, thr},
    max = Max[list];
    min = Min[list];
    diff = (max - min);
    sum = max + min;
    thr = 0.5 * sum;
    If[diff < cm, If[EuclideanDistance[thr, max] < EuclideanDistance[thr, min], thr = min, thr = max]];
    UnitStep[list[[Ceiling[Length[list] / 2]]] - thr]
   ], CompilationTarget → "WVM", Parallelization → False];

(*Bernsen, 1986*)
Clear[MyBernsenFilter];
Options[MyBernsenFilter] = {"ContrastMeasure" → 15, "WindowHalfWidth" → 15, "Mask" → "Box"};
MyBernsenFilter[im_Image, OptionsPattern[]] := Module[{flatelpos, padim, whw, cm, el},
   whw = Round[OptionValue["WindowHalfWidth"]];
   cm = N[OptionValue["ContrastMeasure"] / 255];
   el = Switch[OptionValue["Mask"],
     "Box", Table[1, {2 whw + 1}, {2 whw + 1}],
     "Circle", Ceiling[Rescale[Sign[Table[(x^2 + y^2), {x, -whw, whw}, {y, -whw, whw}] - whw * (whw + 1 / 2)], {1, -1}]],
     _, Table[1, {2 whw + 1}, {2 whw + 1}]
    ];
   flatelpos = Flatten[Position[Flatten[el], 1]];
   padim = ArrayPad[ImageData[im, "Real"], Floor[Dimensions[el] / 2], "Reversed"];
   padim =
    Developer`PartitionMap[(BernsenKern[Flatten[#, 1][[flatelpos]], cm]) &, padim, Dimensions[el], {1, 1}, Ceiling[Dimensions[el] / 2]];
   Image[ArrayPad[padim, -Floor[Dimensions[el] / 2]], "Bit"]
  ];
```

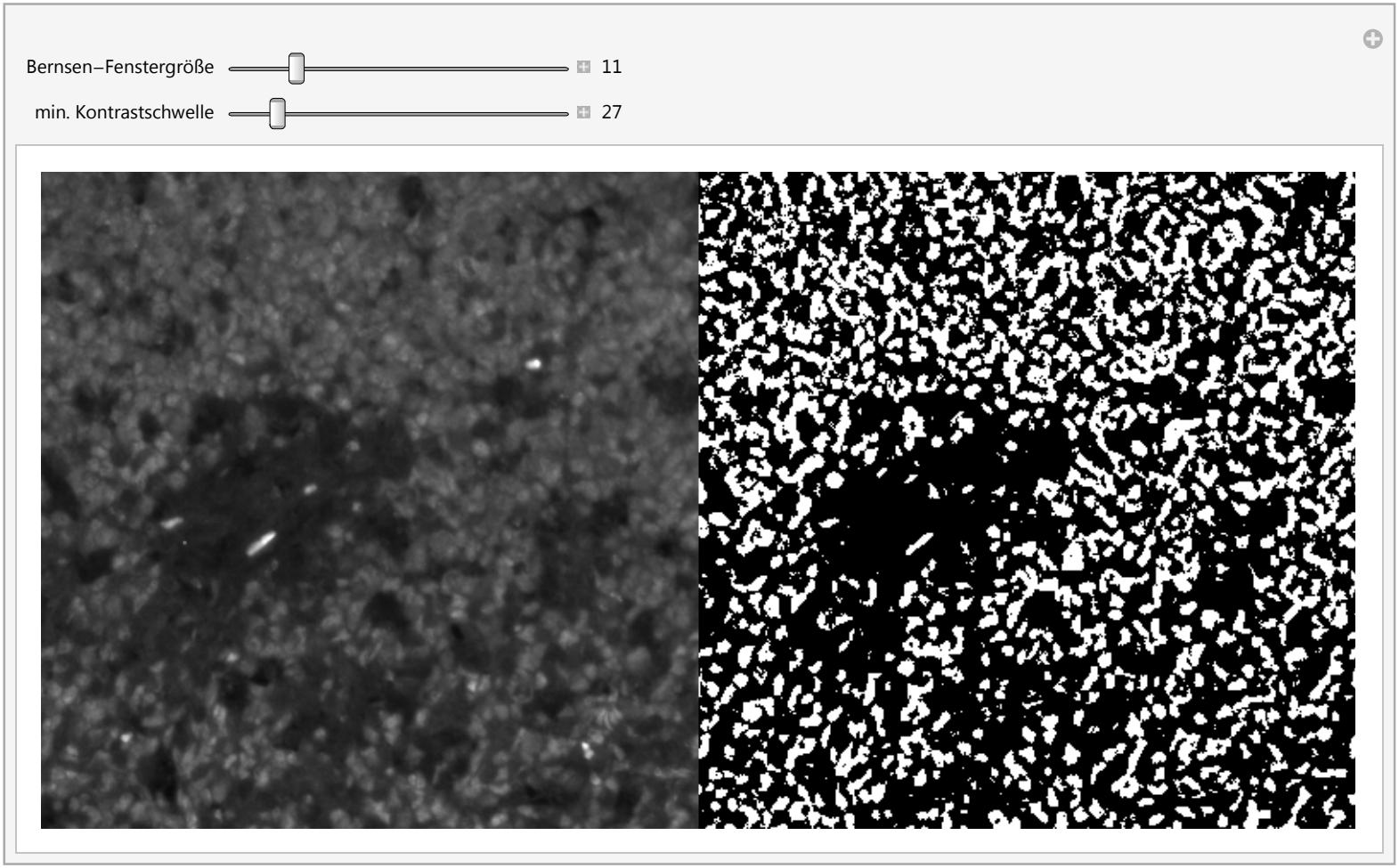
Lokale Schwellwertbildung (Bernsen) mit kreisförmigem Strukturelement (prolif. Zellen)

```
Manipulate[ControlActive[größe, Show[
      ImageAssemble[{#, MyBernsenFilter[#, "ContrastMeasure" → minkontrastschwelle, "WindowHalfWidth" → (größe - 1) / 2, "Mask" → "Circle"]}],
      ImageSize → {2, 1} * ImageDimensions[#]]], {{größe, 11, "Bernsen-Fenstergröße"}, 1, 61, 2, Appearance → "Labeled"},
   {{minkontrastschwelle, 50, "min. Kontrastschwelle"}, 0, 255, 1, Appearance → "Labeled"},
   ContinuousAction → False, SaveDefinitions → True] &[
 ImageCrop[First@ColorSeparate@Ton3CD21dreikanalausgleichF2, {384, 384}]]
```



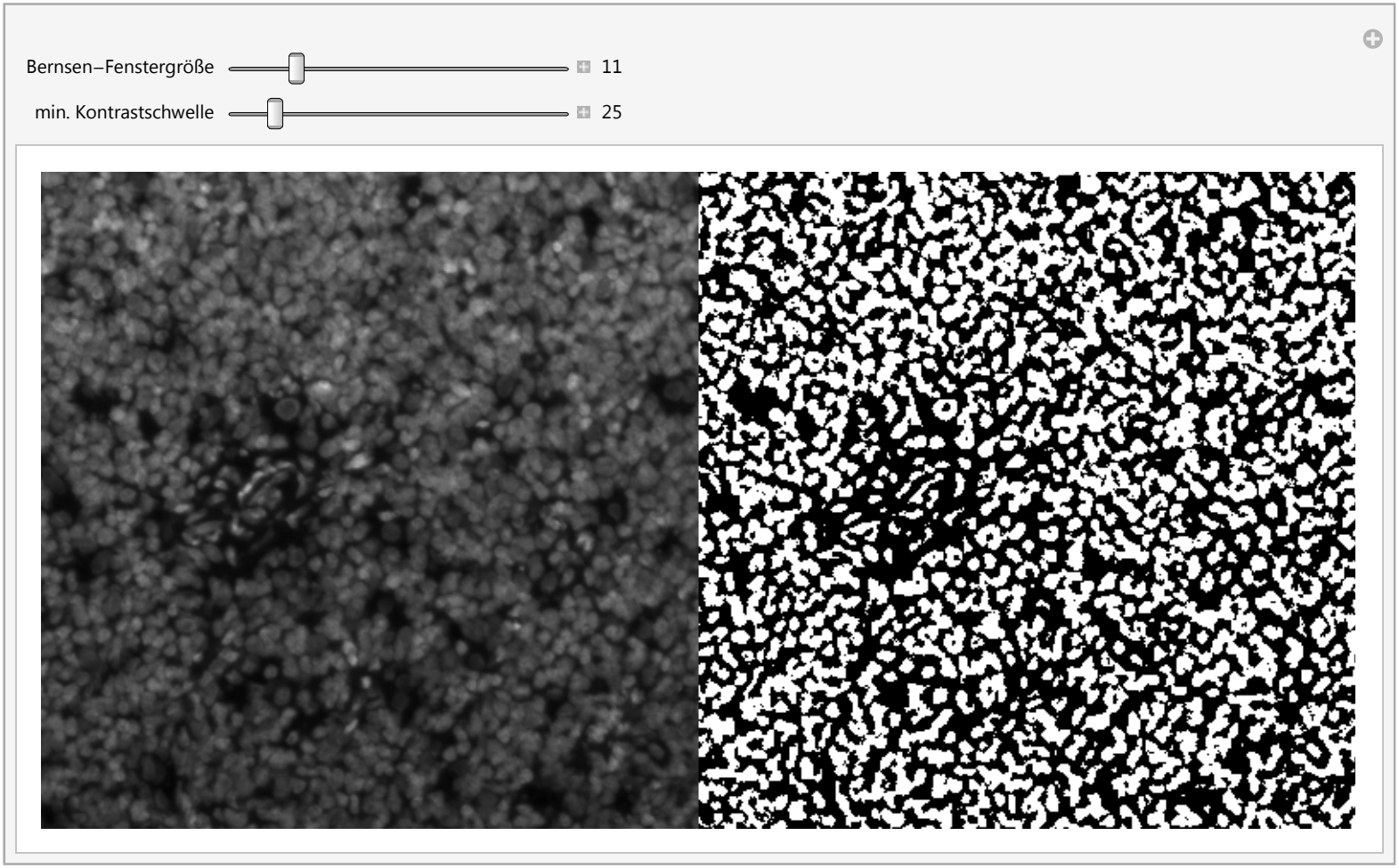
Lokale Schwellwertbildung (Bernsen) mit kreisförmigem Strukturelement (B-Zellkernen)

```
Manipulate[ControlActive[größe, Show[
      ImageAssemble[{#, MyBernsenFilter[#, "ContrastMeasure" → minkontrastschwelle, "WindowHalfWidth" → (größe - 1) / 2, "Mask" → "Circle"]}],
      ImageSize → {2, 1} * ImageDimensions[#]]], {{größe, 11, "Bernsen-Fenstergröße"}, 1, 61, 2, Appearance → "Labeled"},
   {{minkontrastschwelle, 50, "min. Kontrastschwelle"}, 0, 255, 1, Appearance → "Labeled"},
   ContinuousAction → False, SaveDefinitions → True] &[
 ImageCrop[First@Rest@ColorSeparate@Ton3CD21dreikanalausgleichF2, {384, 384}]]
```



Lokale Schwellwertbildung (Bernsen) mit kreisförmigem Strukturelement (alle Zellkerne)

```
Manipulate[ControlActive[größe, Show[
      ImageAssemble[{#, MyBernsenFilter[#, "ContrastMeasure" → minkontrastschwelle, "WindowHalfWidth" → (größe - 1) / 2, "Mask" → "Circle"]}],
      ImageSize → {2, 1} * ImageDimensions[#]]], {{größe, 11, "Bernsen-Fenstergröße"}, 1, 61, 2, Appearance → "Labeled"},
   {{minkontrastschwelle, 25, "min. Kontrastschwelle"}, 0, 255, 1, Appearance → "Labeled"},
   ContinuousAction → False, SaveDefinitions → True] &[
 ImageCrop[Last@ColorSeparate@Ton3CD21dreikanalausgleichF2, {384, 384}]]
```



Lokale Schwellwertbildung in Abhängigkeit von lokalem Mittelwert m_x und Standardabweichung σ_x

Niblack 1986

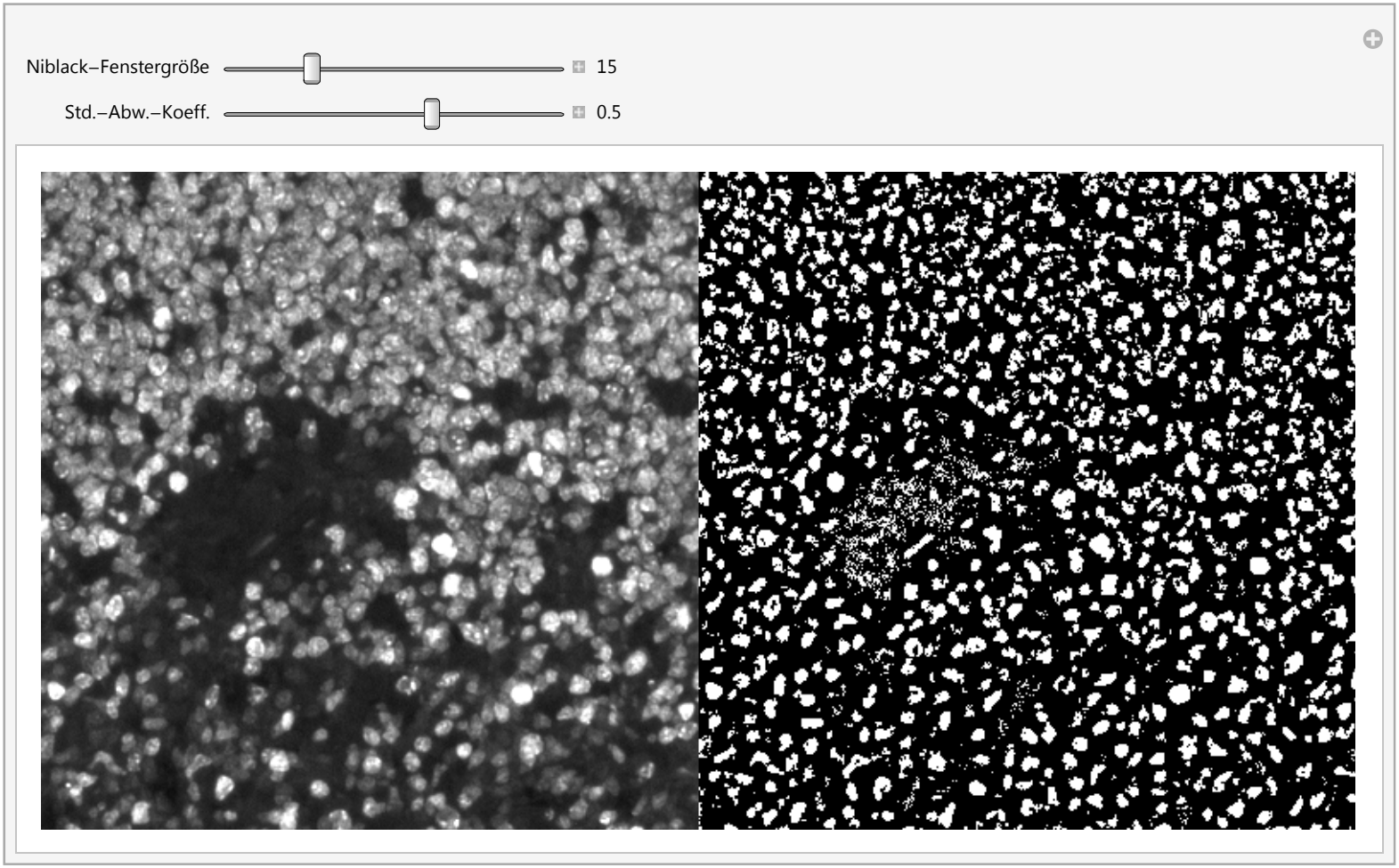
$$t(i, j) = m_x(i, j) + k * \sigma_x(i, j)$$

Umgebungsgröße typ. 15x15, k typ. 0.2

```
Clear[NiblackKern];
NiblackKern = Compile[{{list, _Real, 1}, {sdc, _Real}},
    Module[{mean, stddev, thr},
     mean = Mean[list];
     stddev = StandardDeviation[list];
     thr = mean + sdc * stddev;
     UnitStep[list[[Ceiling[Length[list] / 2]]] - thr]
    ], CompilationTarget → "WVM", Parallelization → False];

(*Niblack, 1986*)
Clear[MyNiblackFilter];
Options[MyNiblackFilter] = {"StdDevCoefficient" → 0.2, "WindowHalfWidth" → 15, "Mask" → "Box"};
MyNiblackFilter[im_Image, OptionsPattern[]] := Module[{flatelpos, padim, whw, sdc, el},
    whw = Round[OptionValue["WindowHalfWidth"]];
    sdc = OptionValue["StdDevCoefficient"];
    el = Switch[OptionValue["Mask"],
      "Box", Table[1, {2 whw + 1}, {2 whw + 1}],
      "Circle", Ceiling[Rescale[Sign[Table[(x^2 + y^2), {x, -whw, whw}, {y, -whw, whw}] - whw * (whw + 1 / 2)], {1, -1}]],
      _, Table[1, {2 whw + 1}, {2 whw + 1}]
     ];
    flatelpos = Flatten[Position[Flatten[el], 1]];
    padim = ArrayPad[ImageData[im, "Real"], Floor[Dimensions[el] / 2], "Reversed"];
    padim =
     Developer`PartitionMap[(NiblackKern[Flatten[#, 1][[flatelpos]], sdc]) &, padim, Dimensions[el], {1, 1}, Ceiling[Dimensions[el] / 2]];
    Image[ArrayPad[padim, -Floor[Dimensions[el] / 2]], "Bit"]
   ];
```

Lokale Schwellwertbildung (Niblack) mit kreisförmigem Strukturelement (prolif. Zellen)

```
Manipulate[ControlActive[größe,
    Show[ImageAssemble[{#, MyNiblackFilter[#, "StdDevCoefficient" → stdabwkoeff, "WindowHalfWidth" → (größe - 1) / 2, "Mask" → "Circle"]}],
     ImageSize → {2, 1} * ImageDimensions[#]]], {{größe, 15, "Niblack-Fenstergröße"}, 1, 61, 2, Appearance → "Labeled"},
   {{stdabwkoeff, 1., "Std.-Abw.-Koeff."}, -2., 2., 0.05, Appearance → "Labeled"}, ContinuousAction → False, SaveDefinitions → True] &[
 ImageCrop[First@ColorSeparate@Ton3CD21dreikanalausgleichF2, {384, 384}]]
```

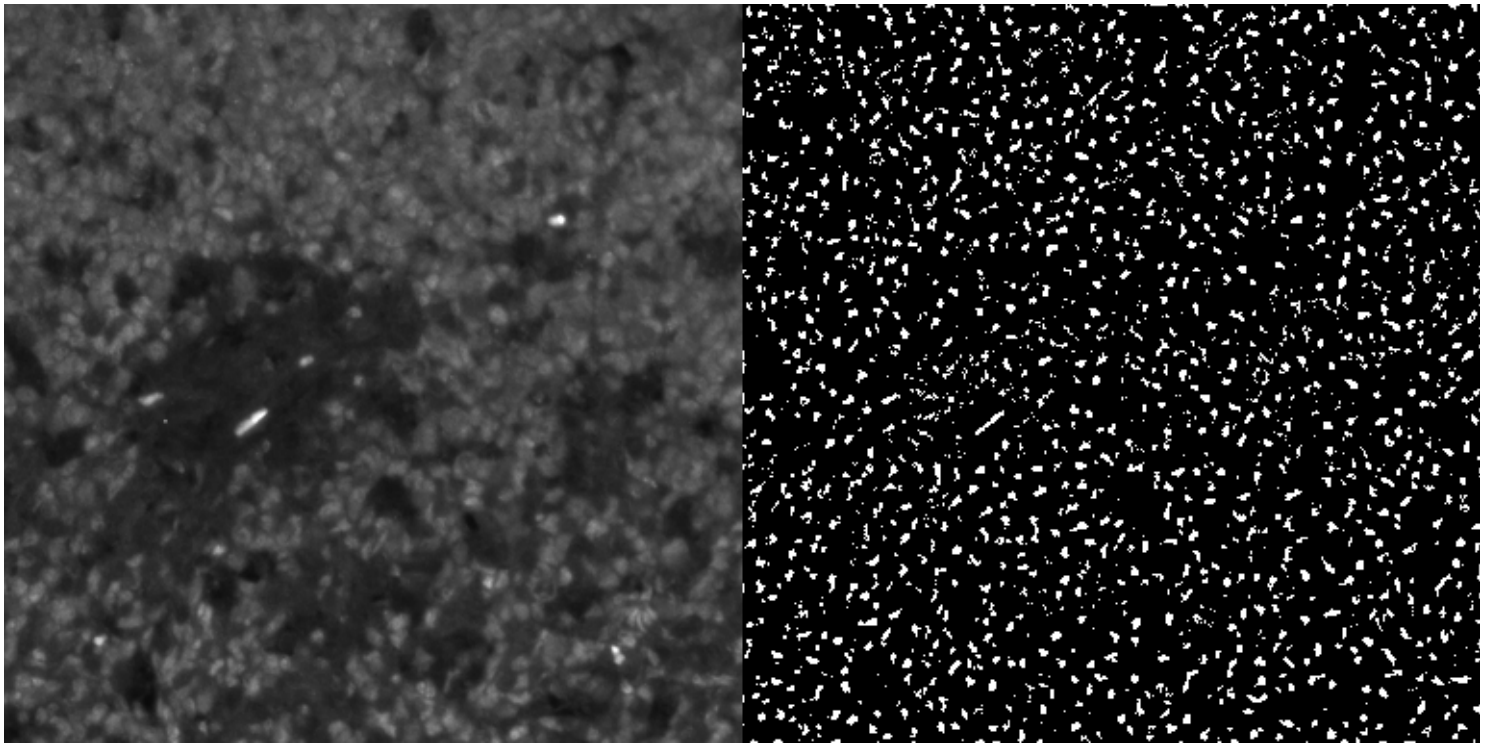


Lokale Schwellwertbildung (Niblack) mit kreisförmigem Strukturelement (B-Zellkerne)

```
Manipulate[ControlActive[größe,
    Show[ImageAssemble[{#, MyNiblackFilter[#, "StdDevCoefficient" → stdabwkoeff, "WindowHalfWidth" → (größe - 1) / 2, "Mask" → "Circle"]}],
     ImageSize → {2, 1} * ImageDimensions[#]]], {{größe, 11, "Niblack-Fenstergröße"}, 1, 61, 2, Appearance → "Labeled"},
   {{stdabwkoeff, 1., "Std.-Abw.-Koeff."}, -2., 2., 0.05, Appearance → "Labeled"}, ContinuousAction → False, SaveDefinitions → True] &[
 ImageCrop[First@Rest@ColorSeparate@Ton3CD21dreikanalausgleichF2, {384, 384}]]
```

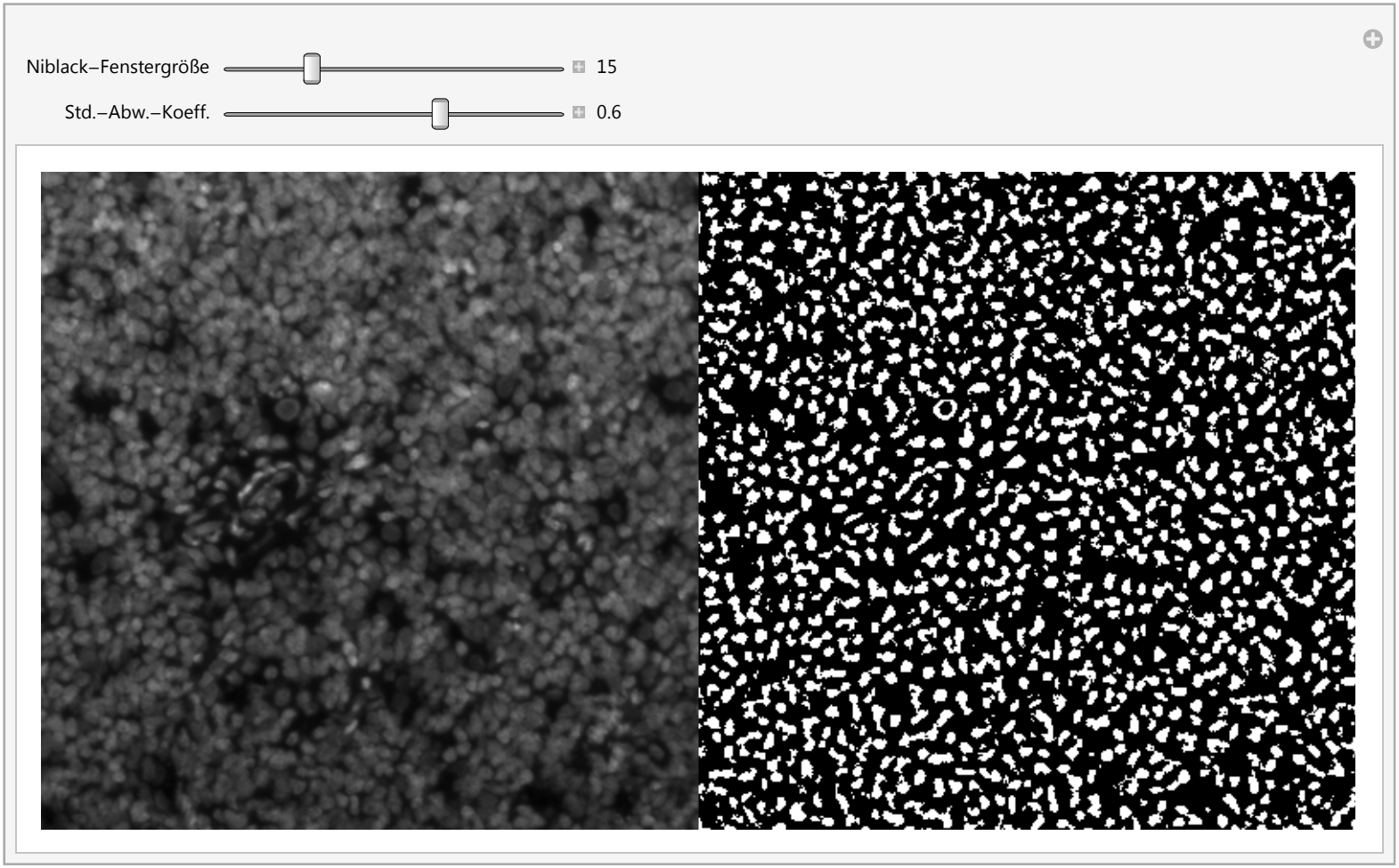
Niblack–Fenstergröße 11

Std.–Abw.–Koeff. 1.



Lokale Schwellwertbildung (Niblack) mit kreisförmigem Strukturelement (alle Zellkerne)

```
Manipulate[ControlActive[größe,
    Show[ImageAssemble[{#, MyNiblackFilter[#, "StdDevCoefficient" → stdabwkoeff, "WindowHalfWidth" → (größe - 1) / 2, "Mask" → "Circle"]}],
     ImageSize → {2, 1} * ImageDimensions[#]]], {{größe, 15, "Niblack-Fenstergröße"}, 1, 61, 2, Appearance → "Labeled"},
   {{stdabwkoeff, 1., "Std.-Abw.-Koeff."}, -2., 2., 0.05, Appearance → "Labeled"}, ContinuousAction → False, SaveDefinitions → True] &[
 ImageCrop[Last@ColorSeparate@Ton3CD21dreikanalausgleichF2, {384, 384}]]
```



Modifizierte lokale Schwellwertbildung in Abhängigkeit von lokalem Mittelwert m_x und Standardabweichung σ_x

Sauvola und Pietikäinen 2000

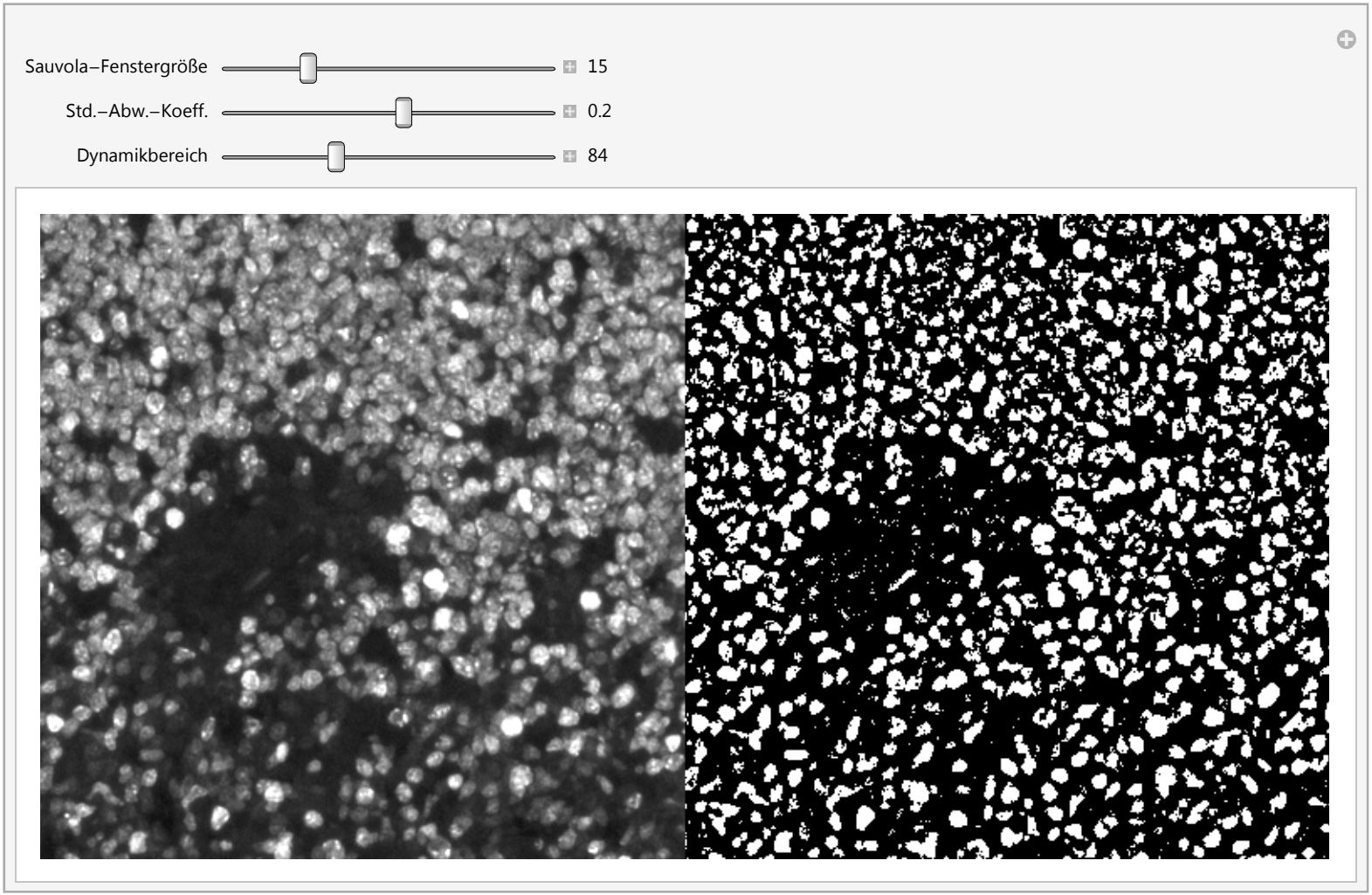
$$t(i,\ j) = m_x(i,\ j)\left[1 + k * \left(1 - \frac{\sigma_x(i,j)}{R}\right)\right]$$

Umgebungsgröße typ. 15x15, k typ. 0.2, Dynamikbereichsparameter R typ. 128 (bei 8bit)

```
Clear[SauvolaKern];
SauvolaKern = Compile[{{list, _Real, 1}, {sdc, _Real}, {dr, _Real}},
    Module[{mean, stddev, thr},
     mean = Mean[list];
     stddev = StandardDeviation[list];
     thr = mean * (1 + sdc * (1 - stddev / dr));
     UnitStep[list[[Ceiling[Length[list] / 2]]] - thr]
    ], CompilationTarget → "WVM", Parallelization → False];

(*Sauvola and Pietikäinen, 2000*)
Clear[MySauvolaFilter];
Options[MySauvolaFilter] = {"StdDevCoefficient" → 0.5, "DynamicRange" → 128, "WindowHalfWidth" → 15, "Mask" → "Box"};
MySauvolaFilter[im_Image, OptionsPattern[]] := Module[{flatelpos, padim, whw, sdc, dr, el},
    whw = Round[OptionValue["WindowHalfWidth"]];
    sdc = OptionValue["StdDevCoefficient"];
    dr = N[OptionValue["DynamicRange"] / 255];
    el = Switch[OptionValue["Mask"],
      "Box", Table[1, {2 whw + 1}, {2 whw + 1}],
      "Circle", Ceiling[Rescale[Sign[Table[(x^2 + y^2), {x, -whw, whw}, {y, -whw, whw}] - whw * (whw + 1 / 2)], {1, -1}]],
      _, Table[1, {2 whw + 1}, {2 whw + 1}]
     ];
    flatelpos = Flatten[Position[Flatten[el], 1]];
    padim = ArrayPad[ImageData[im, "Real"], Floor[Dimensions[el] / 2], "Reversed"];
    padim = Developer`PartitionMap[
      (SauvolaKern[Flatten[#, 1][[flatelpos]], sdc, dr]) &, padim, Dimensions[el], {1, 1}, Ceiling[Dimensions[el] / 2]];
    Image[ArrayPad[padim, -Floor[Dimensions[el] / 2]], "Bit"]
   ];
```

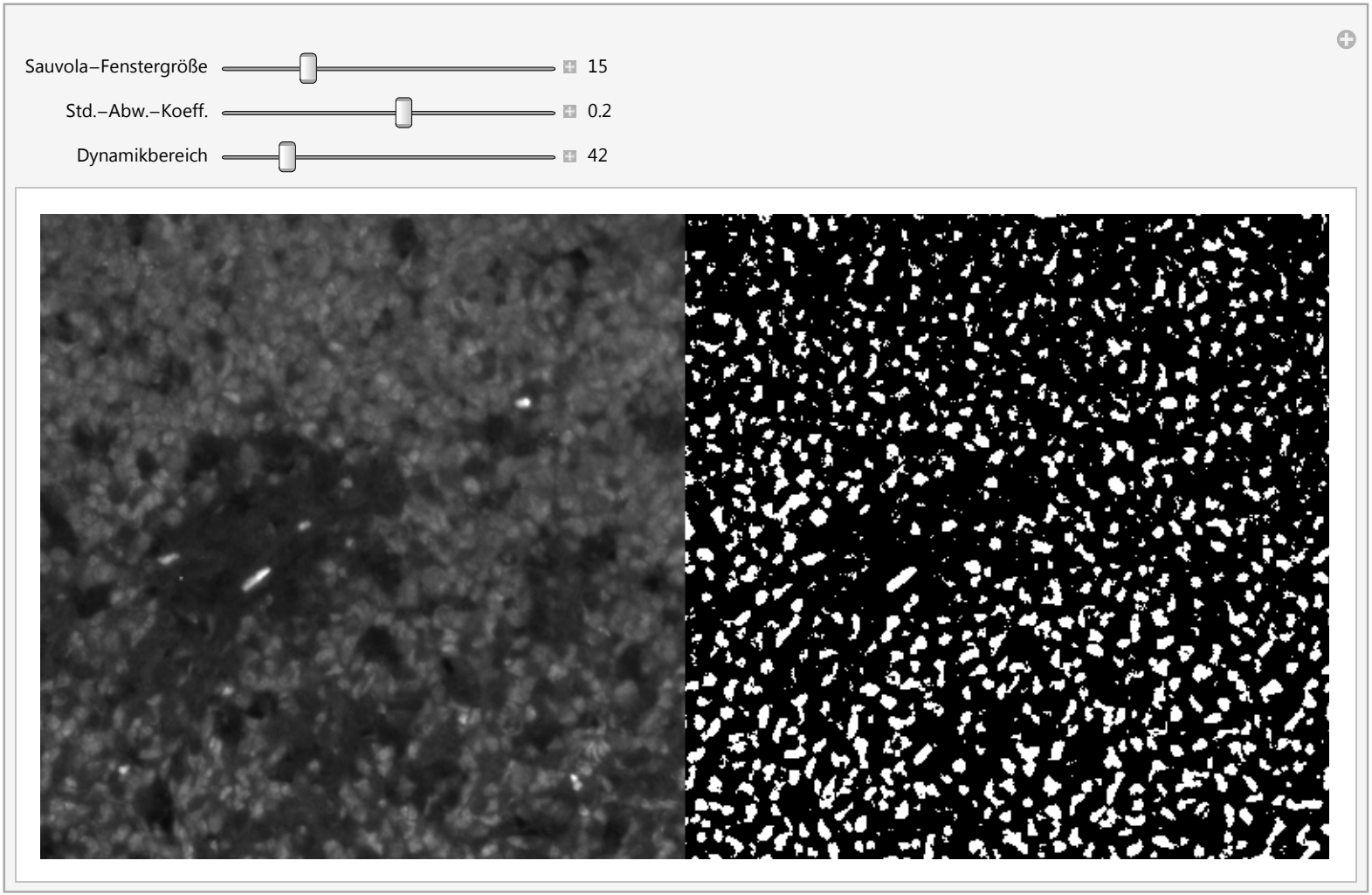
Lokale Schwellwertbildung (Sauvola) mit kreisförmigem Strukturelement (prolif. Zellen)

```
Manipulate[
   ControlActive[größe, Show[ImageAssemble[{#, MySauvolaFilter[#, "StdDevCoefficient" → stdabwkoeff, "DynamicRange" → dynamikbereich,
         "WindowHalfWidth" → (größe - 1) / 2, "Mask" → "Circle"]}], ImageSize → {2, 1} * ImageDimensions[#]]],
   {{größe, 15, "Sauvola-Fenstergröße"}, 1, 61, 2, Appearance → "Labeled"}, {{stdabwkoeff, 0.2, "Std.-Abw.-Koeff."},
    -2., 2., 0.05, Appearance → "Labeled"},
   {{dynamikbereich, 64, "Dynamikbereich"}, 1, 255, 1, Appearance → "Labeled"}, ContinuousAction → False, SaveDefinitions → True] &[
 ImageCrop[First@ColorSeparate@Ton3CD21dreikanalausgleichF2, {384, 384}]]
```



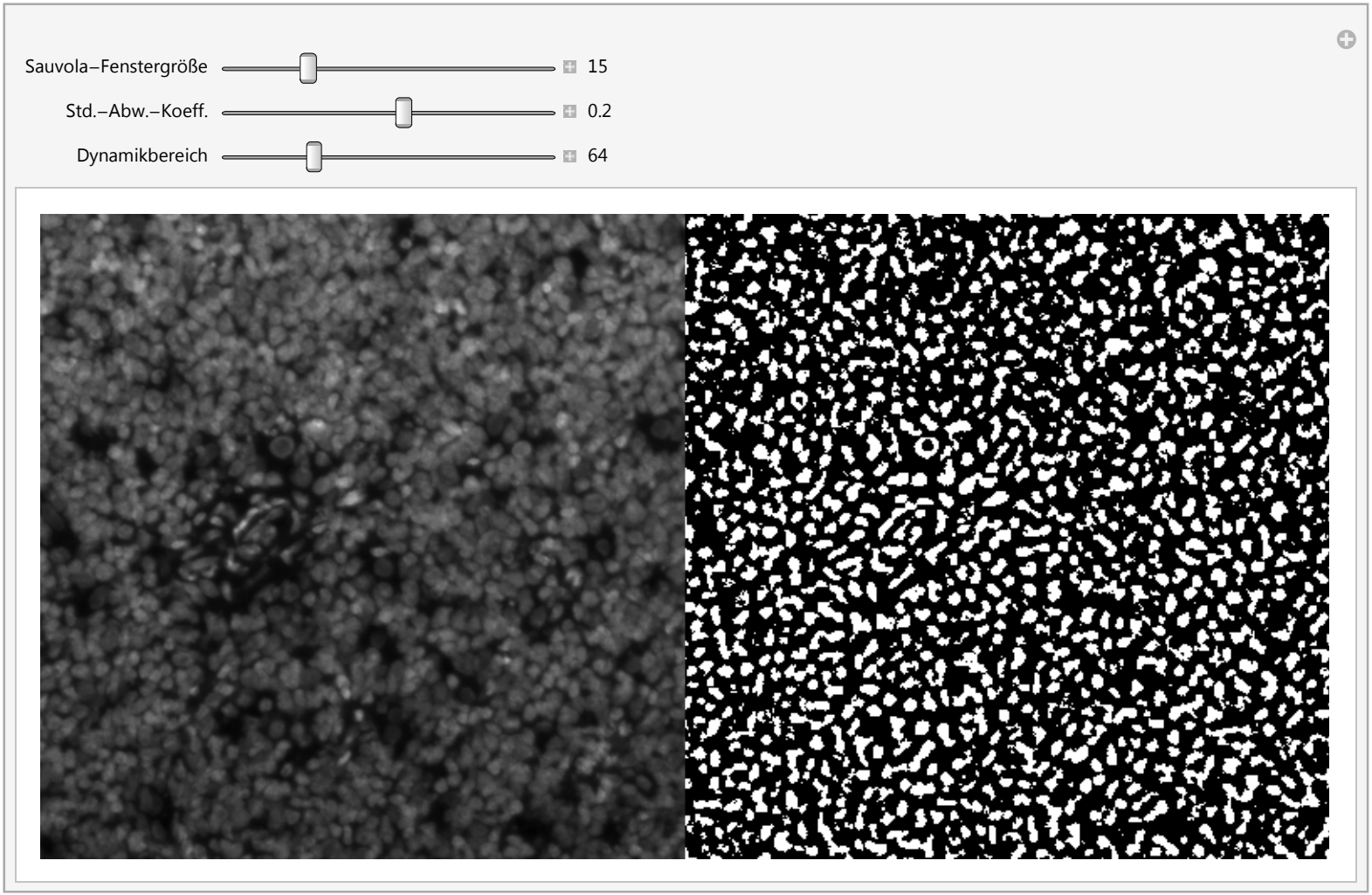
Lokale Schwellwertbildung (Sauvola) mit kreisförmigem Strukturelement (B-Zellkerne)

```
Manipulate[
   ControlActive[größe, Show[ImageAssemble[{#, MySauvolaFilter[#, "StdDevCoefficient" → stdabwkoeff, "DynamicRange" → dynamikbereich,
         "WindowHalfWidth" → (größe - 1) / 2, "Mask" → "Circle"]}], ImageSize → {2, 1} * ImageDimensions[#]]],
   {{größe, 15, "Sauvola-Fenstergröße"}, 1, 61, 2, Appearance → "Labeled"}, {{stdabwkoeff, 0.2, "Std.-Abw.-Koeff."},
    -2., 2., 0.05, Appearance → "Labeled"},
   {{dynamikbereich, 42, "Dynamikbereich"}, 0, 255, 1, Appearance → "Labeled"}, ContinuousAction → False, SaveDefinitions → True] &[
 ImageCrop[First@Rest@ColorSeparate@Ton3CD21dreikanalausgleichF2, {384, 384}]]
```



Lokale Schwellwertbildung (Sauvola) mit kreisförmigem Strukturelement (alle Zellkerne)

```
Manipulate[
   ControlActive[größe, Show[ImageAssemble[{#, MySauvolaFilter[#, "StdDevCoefficient" → stdabwkoeff, "DynamicRange" → dynamikbereich,
         "WindowHalfWidth" → (größe - 1) / 2, "Mask" → "Circle"]}], ImageSize → {2, 1} * ImageDimensions[#]]],
   {{größe, 15, "Sauvola-Fenstergröße"}, 1, 61, 2, Appearance → "Labeled"}, {{stdabwkoeff, 0.2, "Std.-Abw.-Koeff."},
    -2., 2., 0.05, Appearance → "Labeled"},
   {{dynamikbereich, 64, "Dynamikbereich"}, 0, 255, 1, Appearance → "Labeled"}, ContinuousAction → False, SaveDefinitions → True] &[
 ImageCrop[Last@ColorSeparate@Ton3CD21dreikanalausgleichF2, {384, 384}]]
```



17. Labeling in binären Bildern

um Binärbilder in indizierte Bilder zu überführen

→ jedes einzelne Segment verbundener Pixel bekommt eine “Hausnummer” und ist damit adressierbar

1. Überführung eines Bildes in ein Binärbild, falls noch nicht erfolgt
2. Suche mit einer globalen, systematischen Schleife nach unmarkierten Vordergrundpixeln
3. bei Auffinden handelt es sich immer um ein neues separates Segment -> Inkrement und Setzen des Labels; schreibe Koordinaten dieses Pixels in leeren Stack
4. Beginne ab diesen Koordinaten, die vom Stack geholt werden, nach weiteren benachbarten unmarkierten Vordergrundpixeln zu suchen und mit dem aktuellen Label zu markieren. Es wird jedes neu mit diesem Label markierte Pixel in den Stack geschrieben, so daß sich das betrachtete Segment schrittweise komplettiert, da nach jedem Scheitern einer lokalen Suche nach unmarkierten Pixeln wieder vom Stack gelesen wird und zwischenzeitlich aufgeschobene lokale Suchprozesse zu diesem Label fortgesetzt werden, bis Stack leer ist. Dann Fortsetzen der globalen Schleife.
5. der Stack sichert, daß in den Segmenten keine Suchrichtung vorgegeben werden muß und dennoch kein Pixel, das mit dem begonnenen Segment verbunden ist, beim Labeling vergessen wird.

```
Clear[LabelForeground2D];
LabelForeground2D[input_?ArrayQ, threshold_Integer, debug_ : False] :=
  Module[{src = input, n, stack = {}, label = 0, nx, ny, mu, labelimage, i, j, is, js, i1, i2, j1, j2, ii, jj},
   n = ArrayDepth[input];
   If[n == 2,
    {nx, ny} = Dimensions[src];
    mu = nx * ny;
    src = UnitStep[src - $MachineEpsilon - threshold]; (*Schwellwert muß überschritten sein*)
    (*src=UnitStep[src-threshold];(*Schwellwert muß erreicht sein*)*)
    labelimage = Developer`ToPackedArray[Table[0, {nx}, {ny}]];
    For[i = 1, i ≤ nx, i++, (*globale Schleife*)
     For[j = 1, j ≤ ny, j++,
      If[src[[i, j]] > 0, (*ist ein Vordergrund-Pixel?*)
       If[labelimage[[i, j]] == 0, (*Vordergrund-Pixel wurde noch nicht mit Label versehen!*)
        stack = Prepend[stack, {i, j}]; (*Lege die Koordinaten dieses bisher unmarkierten Pixels auf den Stapelspeicher*)
        labelimage[[i, j]] = ++label; (*inkrementiere den Label-Zähler*)
        If[debug == True, Print[Image[Rescale[labelimage]]]];
        While[Length[stack] > 0,
         {is, js} = First[stack]; stack = Rest[stack]; (*entnimm den nächsten Eintrag vom Stapelspeicher*)
         i1 = Max[is - 1, 1]; (*Definition einer Achter-Nachbarschaft um {is,js},
         wobei ein mögliches Verlassen des Bildbereichs abgefangen wird*)
         i2 = Min[is + 1, nx];
```

```
            j1 = Max[js - 1, 1];
            j2 = Min[js + 1, ny];
            For[ii = i1, ii ≤ i2, ii++, (*lokale Schleife innerhalb der Nachbarschaft*)
             For[jj = j1, jj ≤ j2, jj++,
               If[ii != is || jj != js, (*betrachte nur Pixel um ein Zentrum herum, nicht das Zentralpixel selbst*)
                 If[src[[ii, jj]] > 0, (*falls es sich um ein Vordergrund-Pixel handelt*)
                   If[labelimage[[ii, jj]] == 0, (*wurde das Pixel noch nicht mit Label versehen?*)
                     labelimage[[ii, jj]] = label;
                     If[debug == True, Print[Image[Rescale[labelimage]]]];
                     stack = Prepend[stack, {ii, jj}];(*Lege die Koordinaten des eben mit einem
                      Label versehenen Pixels auf den Stack, um später noch dessen Nachbarschaft zu untersuchen*)
                    ];
                  ];
                ];
              ];
            ];
           ];
          ];
         ];
        ];
       ];
      ];
      ,
      Return[$Failed];(*falls es sich nicht um ein Bild (2D-Liste) handelt*)
     ];
     Return[labelimage];
    ];
```

Testbild

```
skalarbild = Image[5 - {{0, 0, 1, 4, 4, 5}, {0, 1, 3, 4, 3, 4}, {1, 3, 4, 2, 1, 3}, {4, 4, 3, 1, 0, 0}, {5, 4, 2, 1, 0, 0}, {5, 5, 4, 3, 1, 0}}, "Byte"];
Image[Rescale[ImageData[skalarbild, "Real"]]]
```

Wertematrix des Testbildes

```
ImageData[skalarbild, "Byte"] // MatrixForm
```

$$\begin{pmatrix} 5 & 5 & 4 & 1 & 1 & 0 \\ 5 & 4 & 2 & 1 & 2 & 1 \\ 4 & 2 & 1 & 3 & 4 & 2 \\ 1 & 1 & 2 & 4 & 5 & 5 \\ 0 & 1 & 3 & 4 & 5 & 5 \\ 0 & 0 & 1 & 2 & 4 & 5 \end{pmatrix}$$

Nach Binarisierung bleiben zwei Segmente übrig

```
Clear[MyBinarize];
MyBinarize[in_, thr_] := Map[(If[# ≤ thr, 0, 1]) &, in, {2}];
```

```
MyBinarize[ImageData[skalarbild, "Byte"], 3 - 1] // MatrixForm
```

$$\begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$

Labeling für beide Segmente

```
LabelForeground2D[ImageData[skalarbild, "Byte"], 3] // MatrixForm
```

$$\begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 2 & 2 & 0 \\ 0 & 0 & 0 & 2 & 2 & 2 \\ 0 & 0 & 2 & 2 & 2 & 2 \\ 0 & 0 & 0 & 0 & 2 & 2 \end{pmatrix}$$

Grauwertzuordnung für die Labels

```
Image[Rescale[LabelForeground2D[ImageData[skalarbild, "Byte"], 3]]]
```

Automatische Farbzuweisung für die Labels

```
LabelForeground2D[ImageData[skalarbild, "Byte"], 3] // Colorize
```

```
LabelForeground2D[ImageData[skalarbild, "Byte"], 3, True];
```

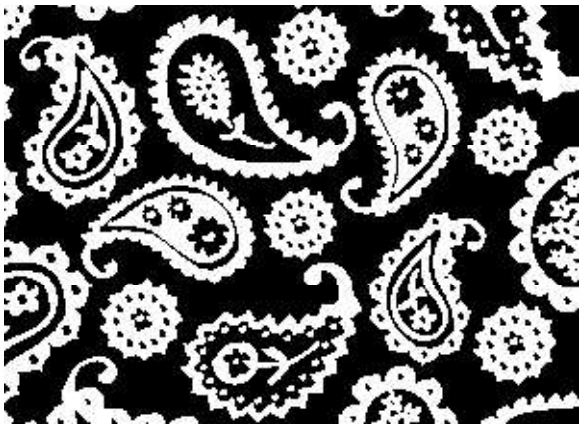
Weiterer Test: Labeling nur der Segmente des Testbild, wo die Zahl 2 steht. Man erkennt, daß erst das zuerst gefundene

Segment komplettiert wird, bevor in Scanline-Order ab der zweiten Hälfte der zweiten Zeile nach einem zweiten Segment weitergesucht wird:

```
Image@Map[If[# == 2, 1, 0] &, ImageData[skalarbild, "Byte"], {2}]
```

```
LabelForeground2D[Map[If[# == 2, 1, 0] &, ImageData[skalarbild, "Byte"], {2}], 0, True];
```

```
labelingtestbild
```



Grauwertzuordnung für die Labels

```
Image[Rescale[LabelForeground2D[ImageData[labelingtestbild, "Byte"], 0]]]
```

Automatische Farbzuweisung für die Labels

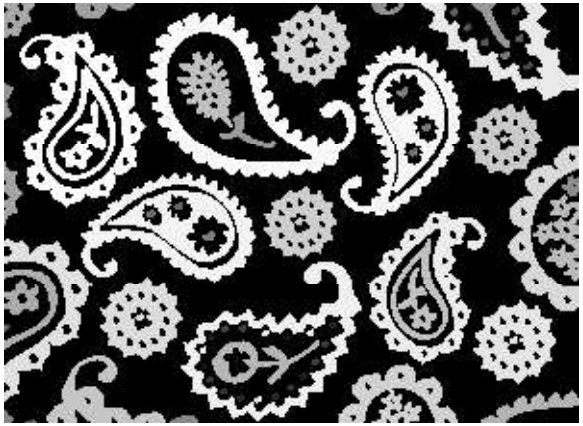
```
LabelForeground2D[ImageData[labelingtestbild, "Byte"], 0] // Colorize
```

Ergebnis des Labelings sind willkürliche Labelzuordnungen, die sich aus der jeweiligen Scanline-Order ergeben. Demgegenüber kann es interessanter sein, die Labels in einer bestimmten Sortierreihenfolge zu vergeben, zum Beispiel anhand der Ordnung der Größe der Segmente:

```
Clear[GrößenrichtigesLabeling];
GrößenrichtigesLabeling[bm_] := Module[{lut},
  lut = Ordering[Join[{0}, Sort[Rest[Sort[Tally[Flatten[bm]]]], #1[[2]] < #2[[2]] &][[All, 1]]]] - 1;
  Map[lut[[# + 1]] &, bm, {2}]
 ]
```

Je größer die Segmente, desto heller:

```
Image[Rescale[GrößenrichtigesLabeling[LabelForeground2D[ImageData[labelingtestbild, "Byte"], 0]]]]
```



Umkehrung: die kleinsten Segmente sind am hellsten:

```
Image[1 - Rescale[GrößenrichtigesLabeling[LabelForeground2D[ImageData[labelingtestbild, "Byte"], 0]]] /. (1 → 0)]
```

Größenrichtige Farbzuweisung entsprechend einer skalierten Farbtabelle

```
ColorData["SunsetColors"]
farbfunktion[wert_] := ColorData["SunsetColors"][wert];
Colorize[#1, ColorFunction → Function[{v}, farbfunktion[v / #2]], ColorFunctionScaling → False] &[GrößenrichtigesLabeling[#], Max[#]] &[
 LabelForeground2D[ImageData[labelingtestbild, "Byte"], 0]]
```

ColorDataFunction[{0, 1},]

Variation der Binarisierungsschwelle, anschl. Labeling, dann automatische Farbzuweisung für die Labels

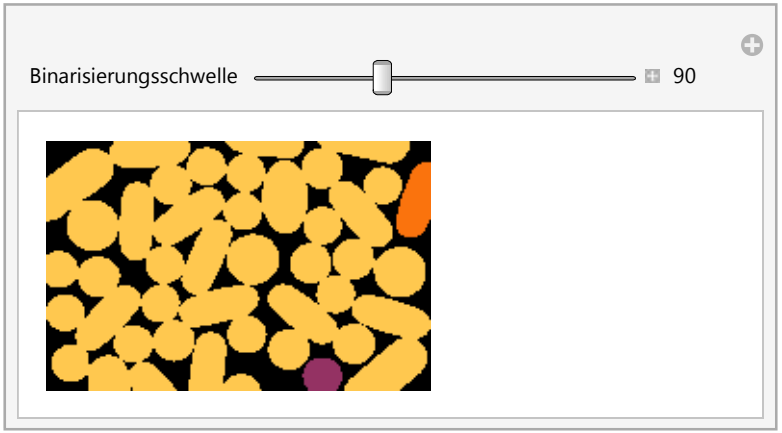
```
tablettenrgb
```

```
Colorize[LabelForeground2D[ImageData[ImageApply[{0.299, 0.587, 0.114}.# &, tablettenrgb], "Byte"], #]] & /@
 {80, 100, 120, 140, 160, 180, 200, 220, 240}
```

Variation der Binarisierungsschwelle, anschl. Labeling, dann größenrichtige Farbzuweisung für die Labels ensprechend einer skalierten Farbtabelle

```
farbfunktion[wert_] := ColorData["SunsetColors"][wert];
Colorize[GrößenrichtigesLabeling[LabelForeground2D[ImageData[ImageApply[{0.299, 0.587, 0.114}.# &, tablettenrgb], "Byte"], #]],
   ColorFunction → Function[{v}, farbfunktion[v]], ColorFunctionScaling → True] & /@ {80, 100, 120, 140, 160, 180, 200, 220, 240}
```

```
Manipulate[
   Block[{farbfunktion},
    farbfunktion[wert_] := ColorData["SunsetColors"][wert];
    ControlActive[schwelle,
     Show[
      Colorize[GrößenrichtigesLabeling[LabelForeground2D[ImageData[#, "Byte"], schwelle]],
       ColorFunction → Function[{v}, farbfunktion[v]], ColorFunctionScaling → True],
      ImageSize → ImageDimensions[#]
     ]
    ]
   ], {{schwelle, 125, "Binarisierungsschwelle"}, 25, 230, 5, Appearance → "Labeled"}, ContinuousAction → False, SaveDefinitions → True] &[
 ImageApply[{0.299, 0.587, 0.114}.# &, tablettenrgb]]
```



In *Mathematica* ist Labeling bereits in Form der Funktion MorphologicalComponents eingebaut:

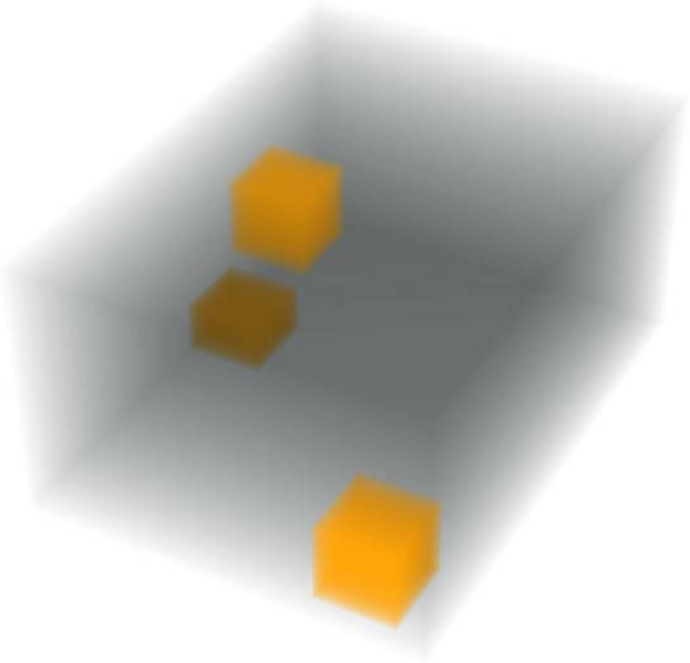
```
labelingtestbild // MorphologicalComponents // Colorize
```

```
farbfunktion[wert_] := ColorData["SunsetColors"][wert];
Colorize[#1, ColorFunction → Function[{v}, farbfunktion[v / #2]], ColorFunctionScaling → False] &[GrößenrichtigesLabeling[#], Max[#]] &[
 MorphologicalComponents@labelingtestbild]
```

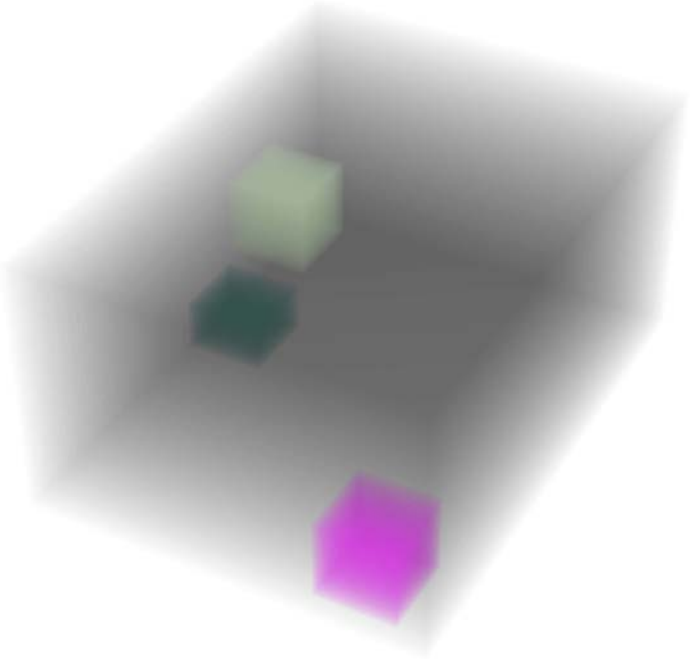
Das Labeling geht auch in 3D:

```
Clear[LabelForeground3D];
LabelForeground3D[input_?ArrayQ, threshold_Integer] :=
  Module[{src = input, n, stack = {}, label = 0, nx, ny, nz, mu, labelimage, i, j, k, is, js, ks, i1, i2, j1, j2, k1, k2, ii, jj, kk},
   n = ArrayDepth[input];
   If[n == 3,
    {nx, ny, nz} = Dimensions[src];
    mu = nx * ny * nz;
    src = UnitStep[src - $MachineEpsilon - threshold]; (*threshold must be exceeded*)
    (*src=UnitStep[src-threshold];(*threshold must be reached*)*)
    labelimage = Developer`ToPackedArray[Table[0, {nx}, {ny}, {nz}]];
    For[i = 1, i ≤ nx, i++,
     For[j = 1, j ≤ ny, j++,
      For[k = 1, k ≤ nz, k++,
       If[src[[i, j, k]] > 0, (*is a foreground voxel*)
        If[labelimage[[i, j, k]] == 0, (*the voxel was not labeled yet*)
         stack = Prepend[stack, {i, j, k}]; (*Push*)
         labelimage[[i, j, k]] = ++label;
         While[Length[stack] > 0,
          {is, js, ks} = First[stack]; stack = Rest[stack]; (*Pop*)
          i1 = Max[is - 1, 1];
          i2 = Min[is + 1, nx];
```

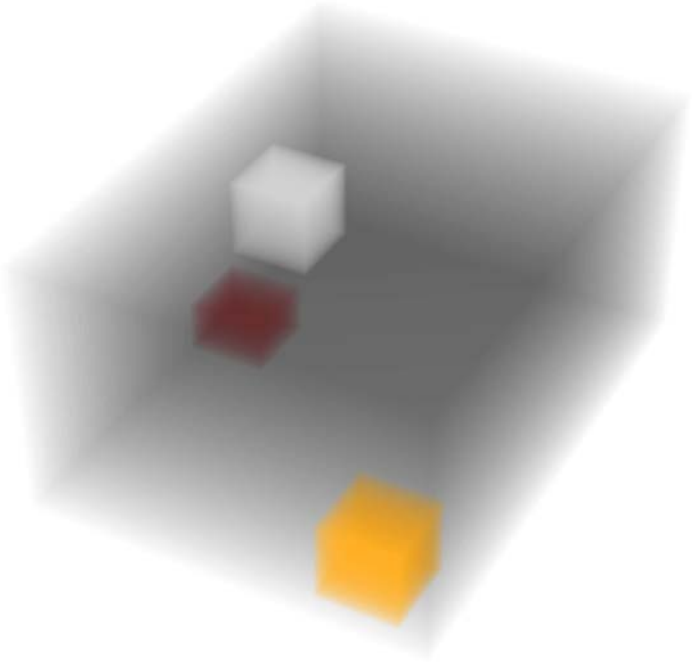
```
                j1 = Max[js - 1, 1];
                j2 = Min[js + 1, ny];
                k1 = Max[ks - 1, 1];
                k2 = Min[ks + 1, nz];
                For[ii = i1, ii ≤ i2, ii++,
                 For[jj = j1, jj ≤ j2, jj++,
                   For[kk = k1, kk ≤ k2, kk++,
                     If[ii != is || jj != js || kk != ks, (*only consider non-center voxels*)
                       If[src[[ii, jj, kk]] > 0, (*is a foreground voxel*)
                         If[labelimage[[ii, jj, kk]] == 0, (*the voxel was not labeled yet*)
                           labelimage[[ii, jj, kk]] = label;
                           stack = Prepend[stack, {ii, jj, kk}];(*Push*)
                          ];
                        ];
                      ];
                    ];
                  ];
                ];
              ];
            ];
          ];
        ];
      ];
     ];
     ,
     Return[$Failed];(*input dimensions do not equal 3*)
    ];
    Return[labelimage];
   ];

SeedRandom[2405];
img3d = Dilation[ColorNegate[Image3D[Table[If[RandomReal[] > N[1 / 1000], 1, 0], {10}, {20}, {15}]]], 1]
```



```
LabelForeground3D[ImageData[img3d, "Bit"], 0] // Colorize
```

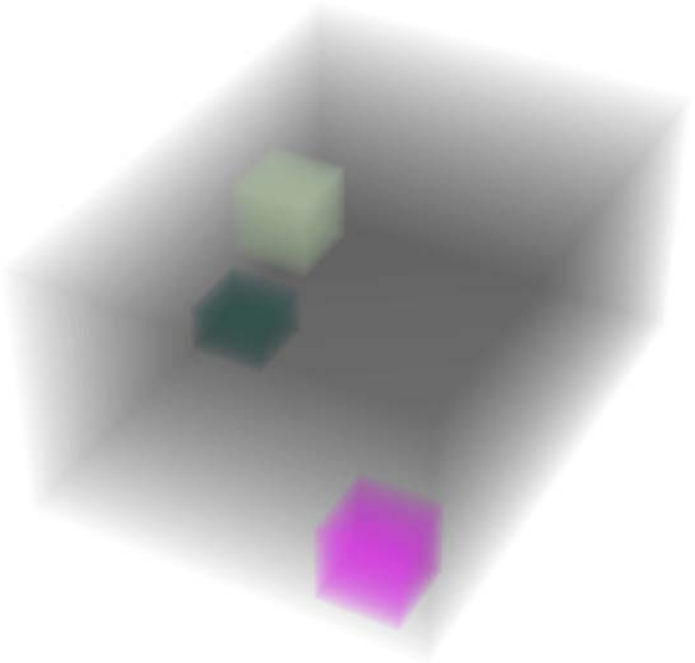


```
farbfunktion[wert_] := ColorData["SunsetColors"][wert];
Colorize[#1, ColorFunction → Function[{v}, farbfunktion[v / #2]], ColorFunctionScaling → False] &[GrößenrichtigesLabeling[#], Max[#]] &[
 LabelForeground3D[ImageData[img3d, "Bit"], 0]]
```

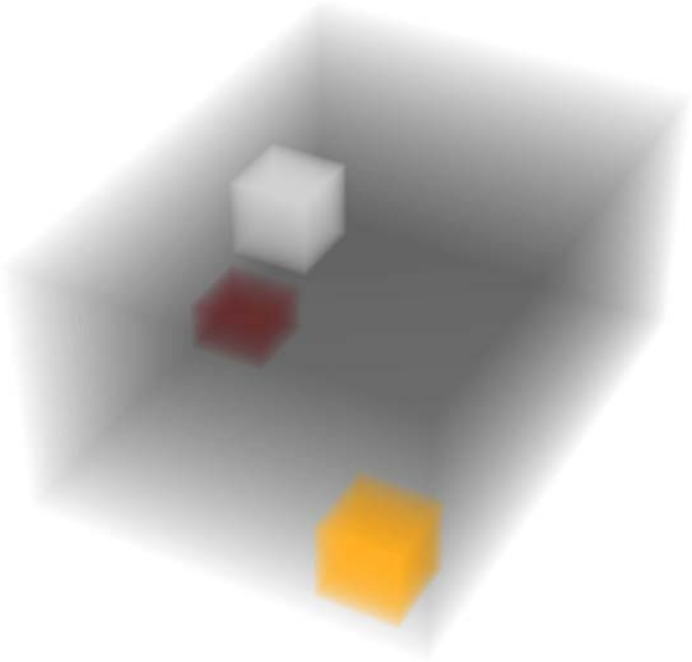


Mathematica unterstützt auch 3D-Labeling direkt:

```
MorphologicalComponents[img3d] // Colorize
```



```
farbfunktion[wert_] := ColorData["SunsetColors"][wert];
Colorize[#1, ColorFunction → Function[{v}, farbfunktion[v / #2]], ColorFunctionScaling → False] &[GrößenrichtigesLabeling[#], Max[#]] &[
 MorphologicalComponents[img3d]]
```



18. Binarisierung mit dem Otsu-Verfahren

nach Nobuyuki Otsu (1979)

1. Ziel des Verfahrens ist es, die Varianz der skalaren Bildwerte innerhalb der zu findenden zwei Klassen zu minimieren
2. Ziel ist es ebenso, die Varianz zwischen den zu findenden zwei Klassen zu maximieren
3. Das Verfahren setzt auf diskretisierten Daten auf bzw. erfordert ein Binning mit einer bestimmten Intervallvorgabe

Formelmäßiger Hintergrund

$H(i)$: absolute Häufigkeit von skalaren Werten im Intervall i, $0 \leq i < L$

$p_i = P(i) = H(i) / (N \cdot M)$: Auftrittswahrscheinlichkeit für Werte im Intervall i

$\sum_{i=0}^{L-1} p_i = 1$

$P_b(k) = \sum_{i=0}^{k} p_i$: Zugehörigkeitswahrscheinlichkeit für die Hintergrundklasse C_b bei gegebenem Schwellwert k

$P_f(k) = \sum_{i=k+1}^{L-1} p_i = 1 - P_b(k)$: Zugehörigkeitswahrscheinlichkeit für die Vordergrundklasse C_f bei gegebenem Schwellwert k

$\mu_T = E(x) = \frac{1}{N \cdot M} \sum_{n=1}^{N} \sum_{m=1}^{M} x_{nm} = \sum_{i=0}^{L-1} i \cdot p_i$: Gesamtmittelwert

$\mu_b(k) = E(x \mid x \leq k) = \sum_{i=0}^{k} i \cdot P(i \mid C_b) =$

$\sum_{i=0}^{k} i \cdot P(C_b \mid i) \cdot P(i) / P(C_b) = \frac{1}{P_b(k)} \sum_{i=0}^{k} i \cdot p_i$: Mittelwert für die Hintergrundklasse C_b bei gegebenem Schwellwert k

mit $P(C_b \mid i) = 1$, da nur i von C_b betrachtet, sowie $P(C_b) = P_b(k)$

$\mu(k) = \sum_{i=0}^{k} i \cdot p_i = \mu_b(k) \cdot P_b(k)$: kumulatives Mittel bis zum Level k

$\mu_f(k) = E(x \mid x > k) = \sum_{i=0}^{k} i \cdot P(i \mid C_f) =$

$\sum_{i=k+1}^{L-1} i \cdot P(C_f \mid i) \cdot P(i) / P(C_f) = \frac{1}{P_f(k)} \sum_{i=k+1}^{L-1} i \cdot p_i$: Mittelwert für die Vordergrundklasse C_f bei gegebenem Schwellwert k

mit $P(C_f \mid i) = 1$, da nur i von C_f betrachtet, sowie $P(C_f) = P_f(k)$

$\sigma^2{}_T = V(x) = E\left((x - \mu_T)^2\right) = \frac{1}{N \cdot M} \sum_{n=1}^{N} \sum_{m=1}^{M} (x_{nm} - \mu_T)^2 = \sum_{i=0}^{L-1} (i - \mu_T)^2 \cdot p_i$: Gesamtvarianz

$\sigma^2{}_b(k) =$

$V(x \mid x \leq k) = E\left((x - \mu_T)^2 \mid x \leq k\right) = \frac{1}{P_b(k)} \sum_{i=0}^{k} (i - \mu_b)^2 \cdot p_i$: Varianz für die Hintergrundklasse C_b bei gegebenem Schwellwert k

$\sigma^2{}_f(k) = V(x \mid x > k) =$

$E\left((x - \mu_T)^2 \mid x > k\right) = \frac{1}{P_f(k)} \sum_{i=k+1}^{L-1} (i - \mu_f)^2 \cdot p_i$: Varianz für die Vordergrundklasse C_f bei gegebenem Schwellwert k

$\sigma^2{}_W(k) = P_b(k) \cdot \sigma^2{}_b(k) + P_f(k) \cdot \sigma^2{}_f(k)$: Varianz innerhalb beider Klassen bei gegebenem Schwellwert k

$\sigma^2{}_B(k) =$

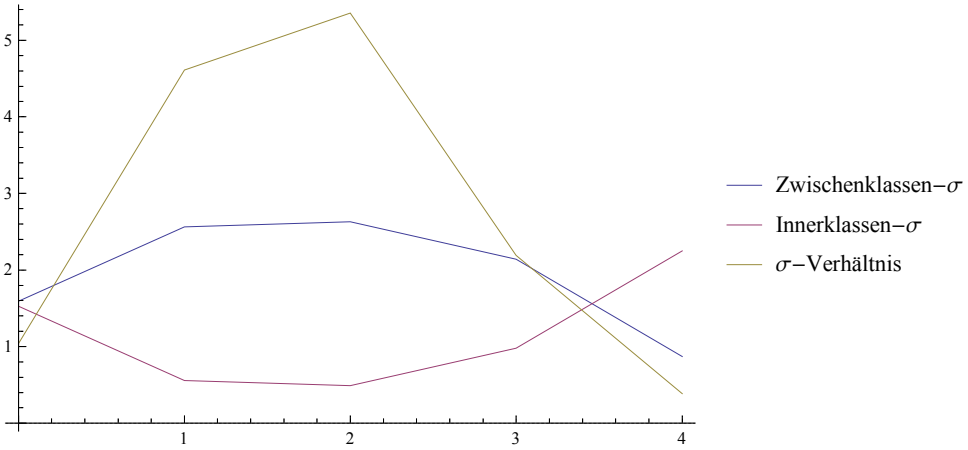
$\sigma^2{}_T - \sigma^2{}_W(k) = P_b(k) \cdot (\mu_b(k) - \mu_T)^2 + P_f(k) \cdot (\mu_f(k) - \mu_T)^2$: Varianz zwischen beiden Klassen bei gegebenem Schwellwert k

$\frac{\sigma^2{}_B(k)}{\sigma^2{}_W(k)} \rightarrow$ Max

Version 1: direkter Ansatz

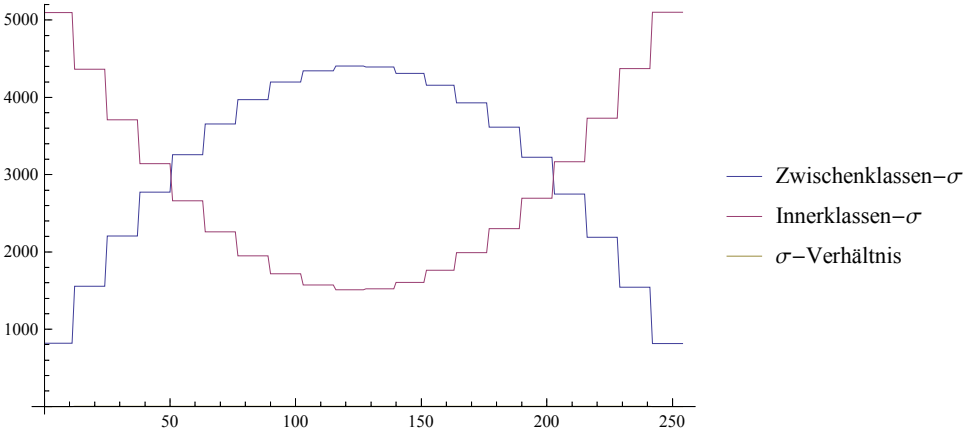
```
Clear[OtsuVersion1];
(*Maximize Ratio (Between-class variance)/(Within-class variance)*)
OtsuVersion1[bm_] := Module[
    {minbm = Min[bm], maxbm = Max[bm], histogramm, k, kopt, μT, ωf, ωb, μb, μf, σqb, σqf, σqW, σqB, σqBmax, Q, Qmax, σqBliste, σqWliste, Qliste},
    histogramm = N[BinCounts[Flatten[bm], {0, maxbm + 1, 1}] / (Times @@ Dimensions[bm])];
    (*normalized overall histogramm, contains all pi*)
    (*μT=N[Mean[Flatten[bm]]];*)
    μT = Total[Range[0, maxbm] * histogramm[[1 ;; maxbm + 1]]];
    (*Total, overall mean*)
    kopt = 0;
    (*kopt is the k-1 for best threshold*)
    Qmax = 0;
    (*Qmax is the maximum Q*)
    σqBliste = Table[0., {Length[histogramm] - 1}];
    σqWliste = Table[0., {Length[histogramm] - 1}];
    Qliste = Table[0., {Length[histogramm] - 1}];
    For[k = minbm + 1, k ≤ maxbm, k++,
     ωb = Total[histogramm[[1 ;; k]]];
     (*ωb is the k-dependent weight for background, i.e. Pb(k)*)
     ωf = (1 - ωb);
     (*ωf is the k-dependent weight for foreground, i.e. Pf(k)*)
     (*μb=N[Mean[Select[Flatten[bm],#<k&]]];*)
     μb = Total[Range[0, k - 1] * histogramm[[1 ;; k]]] / ωb;
     (*μb is the k-dependent mean for background*)
     (*μf=N[Mean[Select[Flatten[bm],#≥k&]]];*)
     μf = Total[Range[k, maxbm] * histogramm[[k + 1 ;; maxbm + 1]]] / ωf;
     (*μf is the k-dependent mean for foreground*)
     (*σqb=N[CentralMoment[Select[Flatten[bm],#<k&]-μb,2]];*)
     σqb = Total[((Range[0, k - 1] - μb) ^ 2) * histogramm[[1 ;; k]]] / ωb;
     (*σqb is the k-dependent variance for background*)
     (*σqf=N[CentralMoment[Select[Flatten[bm],#≥k&]-μf,2]];*)
     σqf = Total[((Range[k, maxbm] - μf) ^ 2) * histogramm[[k + 1 ;; maxbm + 1]]] / ωf;
     (*σqf is the k-dependent variance for foreground*)
     σqW = ωb * σqb + ωf * σqf;
```

```
    (*σqW is the k-dependent Within-class variance*)
    σqWliste[[k]] = σqW;
    σqB  = ωb * (μb - μT) ^ 2 + ωf * (μf - μT) ^ 2;
    (*σqB is the k-dependent Between-class variance*)
    σqBliste[[k]] = σqB;
    Q = σqB / σqW;
    (*Q is the variance ratio*)
    Qliste[[k]] = Q;
    If[Q > Qmax, Qmax = Q; kopt = k - 1];
    (*update kopt according to σqBmax*)
    (*Print[{{k,μb,μf,ωb,ωf,σqb,σqf,σqB,σqW,Q}}];*)
   ];
   Print[ListLinePlot[{σqBliste, σqWliste, Qliste},
     DataRange → {0, Length[histogramm] - 2}, PlotLegends → {"Zwischenklassen-σ", "Innerklassen-σ", "σ-Verhältnis"}]];
   Print[kopt];
   kopt
  ];
skalarbild = Image[{{0, 0, 1, 4, 4, 5}, {0, 1, 3, 4, 3, 4}, {1, 3, 4, 2, 1, 3}, {4, 4, 3, 1, 0, 0}, {5, 4, 2, 1, 0, 0}, {5, 5, 4, 3, 1, 0}}, "Byte"];
{Image[Rescale[#]], Image[MyBinarize[#, OtsuVersion1[#]]]} &[ImageData[skalarbild, "Byte"]]
```

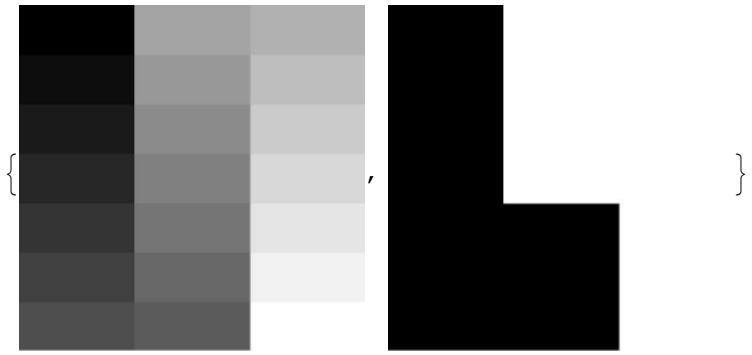


2

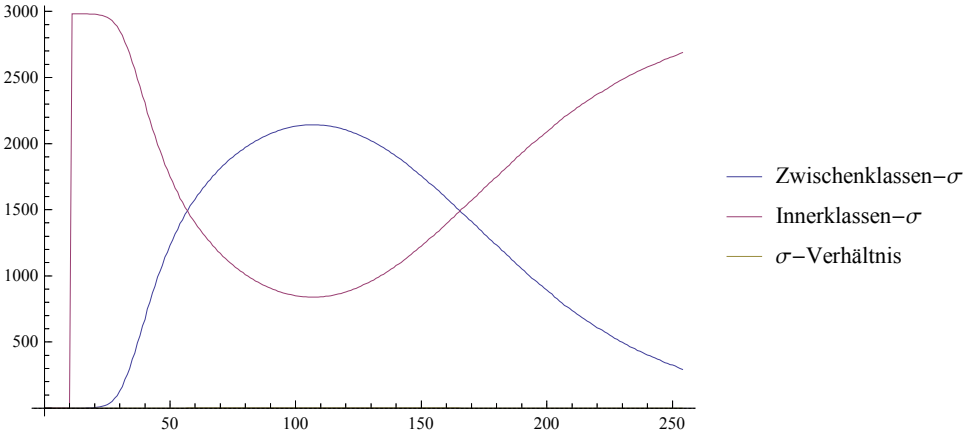
```
{Image[#, "Byte"], Image[MyBinarize[#, OtsuVersion1[#]]]} &[ImageData[ExampleData[{"TestImage", "Gray21"}], "Byte"]]
```



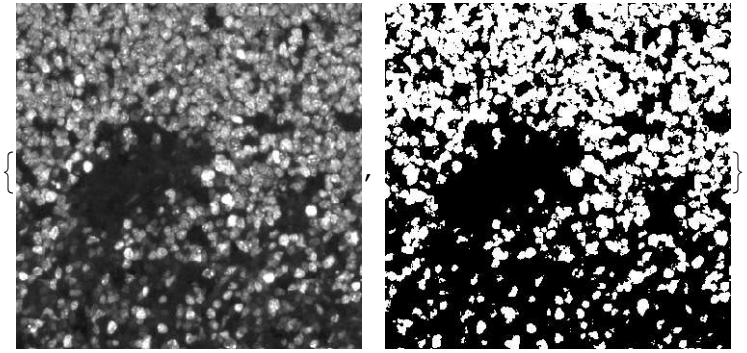
116



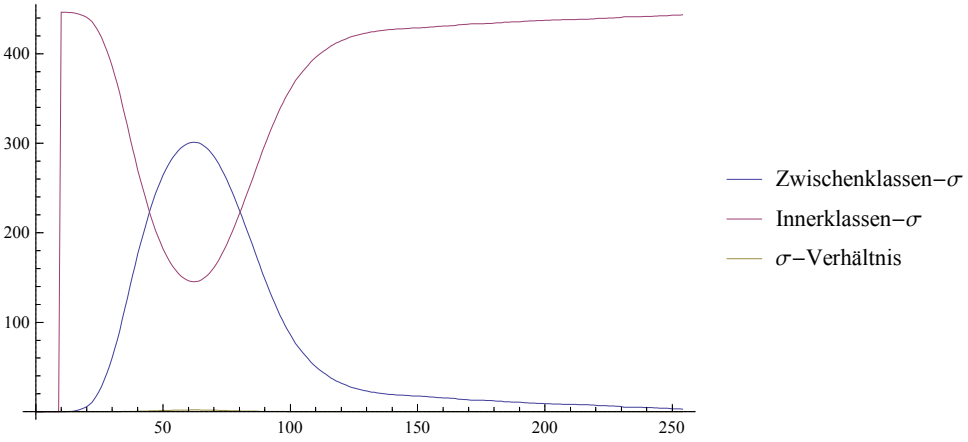
```
{Image[#, "Byte"], Image[MyBinarize[#, OtsuVersion1[#]]]} &[
 ImageData[ImageCrop[First@ColorSeparate@Ton3CD21dreikanalausgleichF2, {384, 384}], "Byte"]]
```



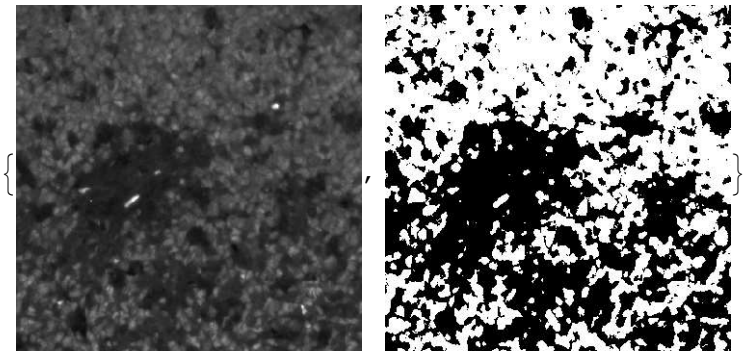
107



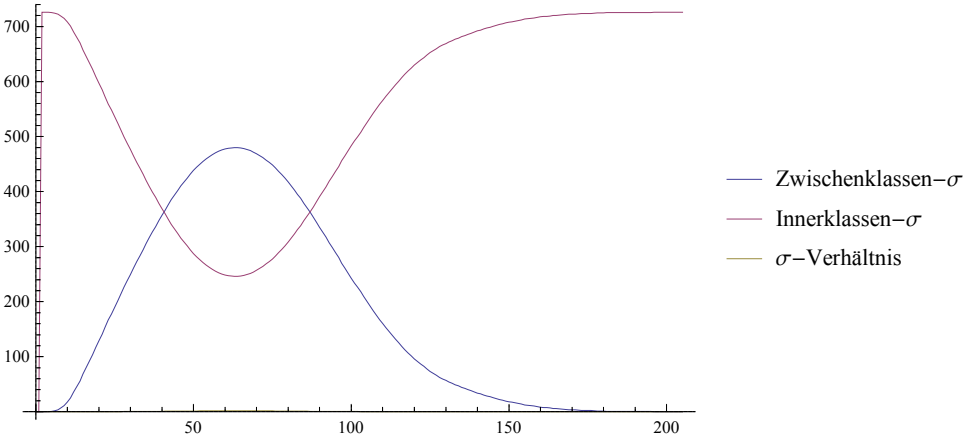
```
{Image[#, "Byte"], Image[MyBinarize[#, OtsuVersion1[#]]]} &[
 ImageData[ImageCrop[First@Rest@ColorSeparate@Ton3CD21dreikanalausgleichF2, {384, 384}], "Byte"]]
```



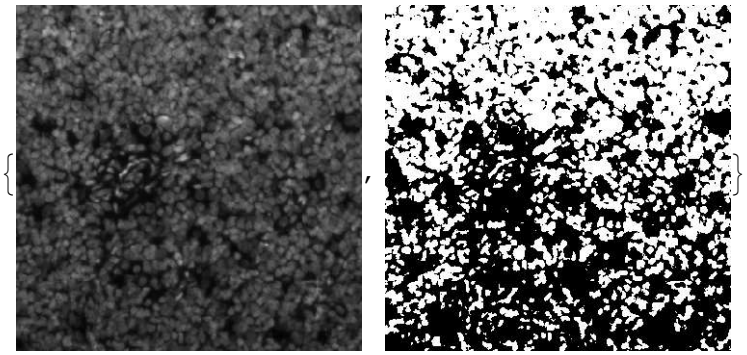
62



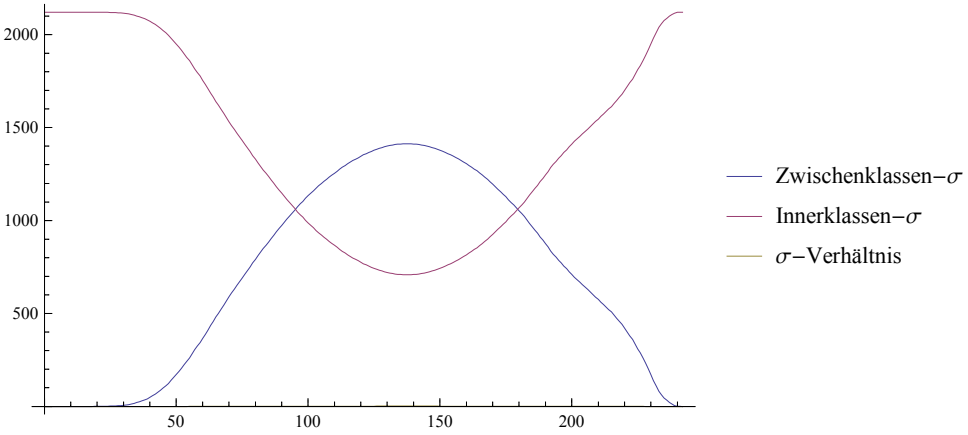
```
{Image[#, "Byte"], Image[MyBinarize[#, OtsuVersion1[#]]]} &[
 ImageData[ImageCrop[Last@ColorSeparate@Ton3CD21dreikanalausgleichF2, {384, 384}], "Byte"]]
```



63



```
{Image[#, "Byte"], Image[MyBinarize[#, OtsuVersion1[#]]]} &[ImageData[ExampleData[{"TestImage", "Elaine"}], "Byte"]]
```


2000
1500
1000
500
50
100
150
200
Zwischenklassen−σ
Innerklassen−σ
σ−Verhältnis

```
137
```

Version 2: Minimierung nur der Varianz innerhalb beider Klassen

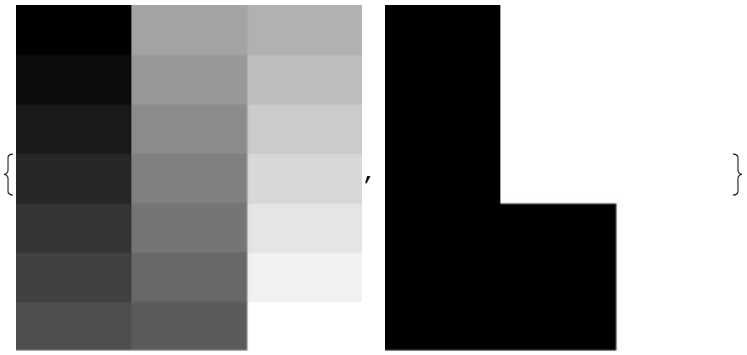
```
Clear[OtsuVersion2];
(*Minimze Within-class variance*)
OtsuVersion2[bm_] := Module[{minbm = Min[bm], maxbm = Max[bm], histogramm, k, kopt, ωf, ωb, μb, μf, σqb, σqf, σqW, σqWmin},
   histogramm = N[BinCounts[Flatten[bm], {0, maxbm + 1, 1}] / (Times @@ Dimensions[bm])];
   (*normalized overall histogramm, contains all p_i*)
   kopt = 0;
   (*kopt is the k-1 for best threshold*)
   σqWmin = $MaxMachineNumber;
   (*σqWmin is the minimum σqW*)
   For[k = minbm + 1, k ≤ maxbm, k++,
    ωb = Total[histogramm[[1 ;; k]]];
    (*ωb is the k-dependent weight for background, i.e. P_b(k)*)
    ωf = (1 - ωb);
    (*ωf is the k-dependent weight for foreground, i.e. P_f(k)*)
    (*μb=N[Mean[Select[Flatten[bm],#<k&]]];*)
    μb = Total[Range[0, k - 1] * histogramm[[1 ;; k]]] / ωb;
    (*μb is the k-dependent mean for background*)
    (*μf=N[Mean[Select[Flatten[bm],#≥k&]]];*)
    μf = Total[Range[k, maxbm] * histogramm[[k + 1 ;; maxbm + 1]]] / ωf;
    (*μf is the k-dependent mean for foreground*)
    (*σqb=N[CentralMoment[Select[Flatten[bm],#<k&]-μb,2]];*)
    σqb = Total[((Range[0, k - 1] - μb) ^ 2) * histogramm[[1 ;; k]]] / ωb;
    (*σqb is the k-dependent variance for background*)
    (*σqf=N[CentralMoment[Select[Flatten[bm],#≥k&]-μf,2]];*)
    σqf = Total[((Range[k, maxbm] - μf) ^ 2) * histogramm[[k + 1 ;; maxbm + 1]]] / ωf;
    (*σqf is the k-dependent variance for foreground*)
    σqW = ωb * σqb + ωf * σqf;
    (*σqW is the k-dependent Within-class variance*)
    If[σqW < σqWmin, σqWmin = σqW; kopt = k - 1];
    (*update kopt according to σqBmax*)
    (*Print[{{k,μb,μf,ωb,ωf,σqb,σqf,σqW}}];*)
   ];
   Print[kopt];
   kopt
  ];
```

```
skalarbild = Image[{{0, 0, 1, 4, 4, 5}, {0, 1, 3, 4, 3, 4}, {1, 3, 4, 2, 1, 3}, {4, 4, 3, 1, 0, 0}, {5, 4, 2, 1, 0, 0}, {5, 5, 4, 3, 1, 0}}, "Byte"];
{Image[Rescale[#]], Image[MyBinarize[#, OtsuVersion2[#]]]} &[ImageData[skalarbild, "Byte"]]
```

2

```
{Image[#, "Byte"], Image[MyBinarize[#, OtsuVersion2[#]]]} &[ImageData[ExampleData[{"TestImage", "Gray21"}], "Byte"]]
```

116



```
{Image[#, "Byte"], Image[MyBinarize[#, OtsuVersion2[#]]]} &[ImageData[ExampleData[{"TestImage", "Elaine"}], "Byte"]]
```

137

Version 3: Maximierung nur der Varianz zwischen beiden Klassen, Variante a

Ansatz über Vereinfachung von $\sigma^2{}_B(k) = P_b(k) \cdot (\mu_b(k) - \mu_T)^2 + P_f(k) \cdot (\mu_f(k) - \mu_T)^2$ mit $\mu_T = P_b(k) \cdot \mu_b(k) + P_f(k) \cdot \mu_f(k)$

$\sigma^2{}_B(k) = P_b(k) \cdot (\mu_b(k) - \mu_T)^2 + P_f(k) \cdot (\mu_f(k) - \mu_T)^2$

$P_b(k) \cdot (\mu_b(k) - P_b(k) \cdot \mu_b(k) - P_f(k) \cdot \mu_f(k))^2 + P_f(k) \cdot (\mu_f(k) - P_b(k) \cdot \mu_b(k) - P_f(k) \cdot \mu_f(k))^2$

$$= P_b(k) \cdot (\mu_b(k) - P_b(k) \cdot \mu_b(k) - P_f(k) \cdot \mu_f(k))^2 + P_f(k) \cdot (\mu_f(k) - P_f(k) \cdot \mu_f(k) - P_b(k) \cdot \mu_b(k))^2$$

$$= P_b(k) \cdot (\mu_b(k) \cdot (1 - P_b(k)) - P_f(k) \cdot \mu_f(k))^2 + P_f(k) \cdot (\mu_f(k) \cdot (1 - P_f(k)) - P_b(k) \cdot \mu_b(k))^2$$

$$= P_b(k) \cdot (\mu_b(k) \cdot P_f(k) - P_f(k) \cdot \mu_f(k))^2 + P_f(k) \cdot (\mu_f(k) \cdot P_b(k) - P_b(k) \cdot \mu_b(k))^2$$

$$= P_b(k) \cdot P^2{}_f(k) \cdot (\mu_b(k) - \mu_f(k))^2 + P_f(k) \cdot P^2{}_b(k) \cdot (\mu_f(k) - \mu_b(k))^2$$

$$= \left(P_b(k) \cdot P^2{}_f(k) + P_f(k) \cdot P^2{}_b(k)\right) \cdot (\mu_b(k) - \mu_f(k))^2$$

$$= (\mu_b(k) - \mu_f(k))^2 \cdot \left(P_b(k) \cdot (1 - P_b(k))^2 + (1 - P_b(k)) \cdot P^2{}_b(k)\right)$$

$$= (\mu_b(k) - \mu_f(k))^2 \cdot \left(P_b(k) \cdot \left(1 - 2\,P_b(k) + P^2{}_b(k)\right) + (1 - P_b(k)) \cdot P^2{}_b(k)\right)$$

$$= (\mu_b(k) - \mu_f(k))^2 \cdot \left(P_b(k) - 2\,P^2{}_b(k) + P^3{}_b(k) + P^2{}_b(k) - P^3{}_b(k)\right)$$

$$= (\mu_b(k) - \mu_f(k))^2 \cdot \left(P_b(k) - 2\,P^2{}_b(k) + P^2{}_b(k)\right)$$

$$= (\mu_b(k) - \mu_f(k))^2 \cdot \left(P_b(k) - P^2{}_b(k)\right)$$

$$= (\mu_b(k) - \mu_f(k))^2 \cdot P_b(k)\,(1 - P_b(k))$$

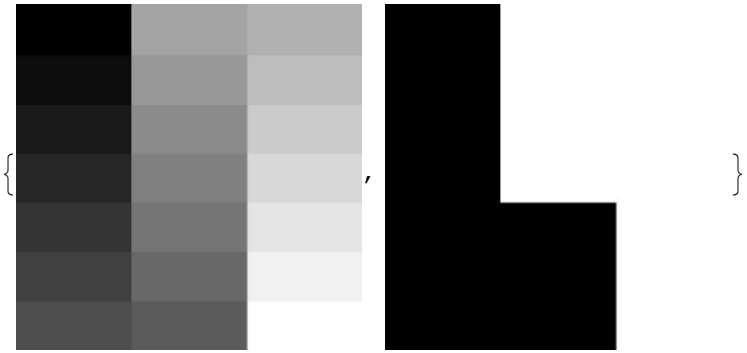
$$= (\mu_b(k) - \mu_f(k))^2 \cdot P_b(k) \cdot P_f(k)$$

```
Clear[OtsuVersion3];
(*Maximize Between-class variance, Variante a*)
OtsuVersion3[bm_] := Module[{minbm = Min[bm], maxbm = Max[bm], histogramm, k, kopt, ωf, ωb, μb, μf, σqb, σqf, σqB, σqBmax},
   histogramm = N[BinCounts[Flatten[bm], {0, maxbm + 1, 1}] / (Times @@ Dimensions[bm])];
   (*normalized overall histogramm, contains all p_i*)
   σqBmax = 0.;
   (*σqBmax is the maximum Between-class variance*)
   kopt = 0;
   (*kopt is the k-1 for best threshold*)
   For[k = minbm + 1, k ≤ maxbm, k++,
    ωb = Total[histogramm[[1 ;; k]]];
    (*ωb is the k-dependent weight for background, i.e. P_b(k)*)
    ωf = (1 - ωb);
    (*ωf is the k-dependent weight for foreground, i.e. P_f(k)*)
    (*μb=N[Mean[Select[Flatten[bm],#<k&]]];*)
    μb = Total[Range[0, k - 1] * histogramm[[1 ;; k]]] / ωb;
    (*μb is the k-dependent mean for background*)
    (*μf=N[Mean[Select[Flatten[bm],#≥k&]]];*)
    μf = Total[Range[k, maxbm] * histogramm[[k + 1 ;; maxbm + 1]]] / ωf;
    (*μf is the k-dependent mean for foreground*)
    σqB = ωb * ωf * (μb - μf)^2;
    (*σqB is the k-dependent Between-class variance*)
    If[σqB > σqBmax, σqBmax = σqB; kopt = k - 1];
    (*update kopt according to σqBmax*)
    (*Print[{{k,μb,μf,ωb,ωf,σqB}}];*)
   ];
   Print[kopt];
   kopt
  ];

skalarbild = Image[{{0, 0, 1, 4, 4, 5}, {0, 1, 3, 4, 3, 4}, {1, 3, 4, 2, 1, 3}, {4, 4, 3, 1, 0, 0}, {5, 4, 2, 1, 0, 0}, {5, 5, 4, 3, 1, 0}}, "Byte"];
{Image[Rescale[#]], Image[MyBinarize[#, OtsuVersion3[#]]]} &[ImageData[skalarbild, "Byte"]]
```

```
2
```

```
{Image[#, "Byte"], Image[MyBinarize[#, OtsuVersion3[#]]]} &[ImageData[ExampleData[{"TestImage", "Gray21"}], "Byte"]]
```

116



```
{Image[#, "Byte"], Image[MyBinarize[#, OtsuVersion3[#]]]} &[ImageData[ExampleData[{"TestImage", "Elaine"}], "Byte"]]
```

137

Version 4: Maximierung nur der Varianz zwischen beiden Klassen, Variante b

Ansatz über weitere Vereinfachung von $\sigma^2{}_B(k) = (\mu_b(k) - \mu_f(k))^2 \cdot P_b(k) \cdot P_f(k)$, so daß auf die wiederkehrende Berechnung von Varianzen für Vorder- und Hintergrundklasse ganz verzichtet werden kann!

$$\sigma^2{}_B(k) = (\mu_b(k) - \mu_f(k))^2 \cdot P_b(k) \cdot P_f(k)$$

mit $\mu_b(k) = \mu(k) / P_b(k)$

und mit $\mu_f(k) = \frac{\mu_T - P_b(k) \cdot \mu_b(k)}{P_f(k)} = \frac{\mu_T - \mu(k)}{P_f(k)}$ wegen $\mu_T = P_b(k) \cdot \mu_b(k) + P_f(k) \cdot \mu_f(k)$

$$\sigma^2{}_B(k) = \left(\frac{\mu(k)}{P_b(k)} - \frac{\mu_T - \mu(k)}{P_f(k)}\right)^2 \cdot P_b(k) \cdot P_f(k)$$

$$= \left(\frac{\mu(k) \cdot P_f(k)}{P_b(k) \cdot P_f(k)} - \frac{(\mu_T - \mu(k)) \cdot P_b(k)}{P_f(k) \cdot P_b(k)}\right)^2 \cdot P_b(k) \cdot P_f(k)$$

$$= \left(\frac{\mu(k) \cdot P_f(k) - (\mu_T - \mu(k)) \cdot P_b(k)}{P_f(k) \cdot P_b(k)}\right)^2 \cdot P_b(k) \cdot P_f(k)$$

$$= (\mu(k) \cdot P_f(k) - (\mu_T - \mu(k)) \cdot P_b(k))^2 \cdot \frac{1}{P_b(k) \cdot P_f(k)}$$

$$= (\mu(k) \cdot P_f(k) - \mu_T \cdot P_b(k) + \mu(k) \cdot P_b(k))^2 \cdot \frac{1}{P_b(k) \cdot P_f(k)}$$

$$= (\mu(k) \cdot (1 - P_b(k)) - \mu_T \cdot P_b(k) + \mu(k) \cdot P_b(k))^2 \cdot \frac{1}{P_b(k) \cdot P_f(k)}$$

$$= (\mu(k) - \mu(k) \cdot P_b(k) - \mu_T \cdot P_b(k) + \mu(k) \cdot P_b(k))^2 \cdot \frac{1}{P_b(k) \cdot P_f(k)}$$

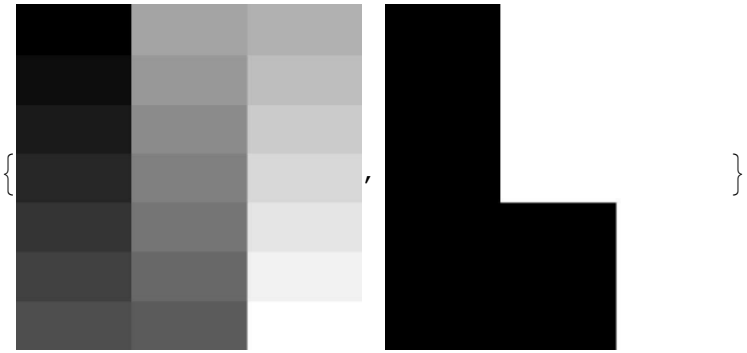
$$= \frac{(\mu(k) - \mu_T \cdot P_b(k))^2}{P_b(k) \cdot (1 - P_b(k))}$$

```
Clear[OtsuVersion4];
(*Maximize Between-class variance, Variante b*)
OtsuVersion4[bm_] := Module[{minbm = Min[bm], maxbm = Max[bm], histogramm, k, kopt, μT, ωf, ωb, μbtωb, μb, μf, σqB, σqBmax},
  histogramm = N[BinCounts[Flatten[bm], {0, maxbm + 1, 1}]] / (Times @@ Dimensions[bm]);
  (*normalized overall histogramm, contains all p_i*)
  (*μT=N[Mean[Flatten[bm]]];*)
  μT = Total[Range[0, maxbm] * histogramm[[1 ;; maxbm + 1]]];
  (*Total, overall mean*)
  σqBmax = 0.;
  (*σqBmax is the maximum Between-class variance*)
  kopt = 0;
  (*kopt is the k-1 for best threshold*)
  For[k = minbm + 1, k ≤ maxbm, k++,
   ωb = Total[histogramm[[1 ;; k]]];
   (*ωb is the k-dependent weight for background, i.e. P_b(k)*)
   ωf = (1 - ωb);
   (*ωf is the k-dependent weight for foreground, i.e. P_f(k)*)
   μbtωb = Total[Range[0, k - 1] * histogramm[[1 ;; k]]];
   (*μbtωb is the k-dependent mean for background times weight for background, i.e. kumulative mean up to level k*)
   If[ωb ≠ 1. && ωb ≠ 0., σqB = (μT * ωb - μbtωb) * (μT * ωb - μbtωb) / (ωb * ωf), σqB = 0.];
   (*σqB is the k-dependent Between-class variance*)
   If[σqB > σqBmax, σqBmax = σqB; kopt = k - 1];
   (*update kopt according to σqBmax*)
   (*Print[{{k,ωb,ωf,σqB}}];*)
  ];
  Print[kopt];
  kopt
 ]

skalarbild = Image[{{0, 0, 1, 4, 4, 5}, {0, 1, 3, 4, 3, 4}, {1, 3, 4, 2, 1, 3}, {4, 4, 3, 1, 0, 0}, {5, 4, 2, 1, 0, 0}, {5, 5, 4, 3, 1, 0}}, "Byte"];
{Image[Rescale[#]], Image[MyBinarize[#, OtsuVersion4[#]]]} &[ImageData[skalarbild, "Byte"]]
```

2

```
{Image[#, "Byte"], Image[MyBinarize[#, OtsuVersion4[#]]]} &[ImageData[ExampleData[{"TestImage", "Gray21"}], "Byte"]]
```

116



```
{Image[#, "Byte"], Image[MyBinarize[#, OtsuVersion4[#]]]} &[ImageData[ExampleData[{"TestImage", "Elaine"}], "Byte"]]
```

137

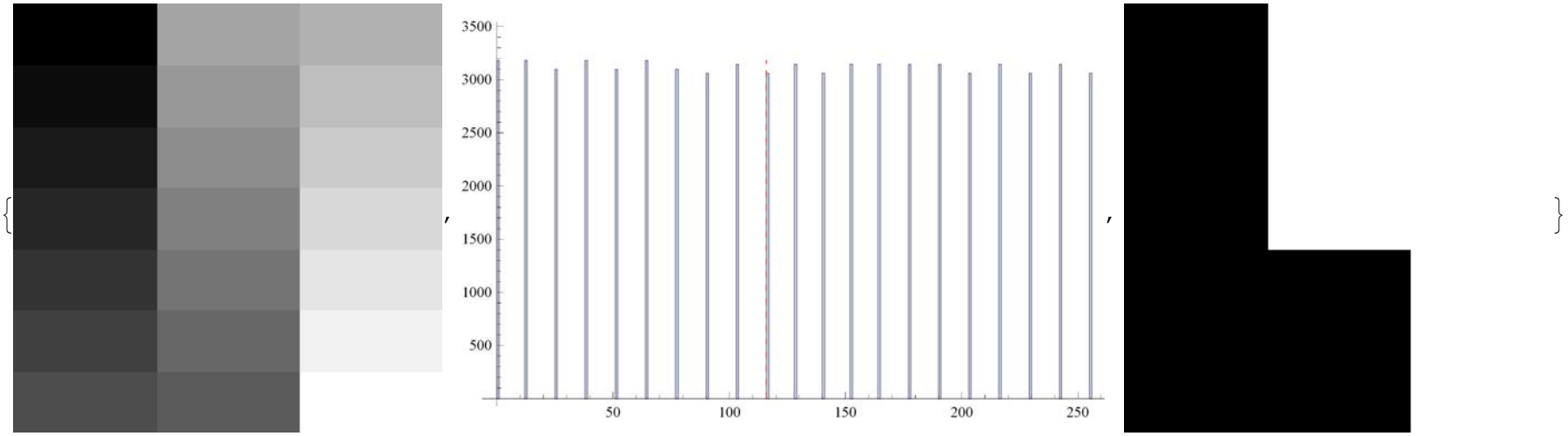
Zusammenfassung von Version 4 des Otsu-Binarisierungsverfahrens:

1. Berechne das normierte Histogramm des skalaren Eingangbildes, also die p_i
2. Berechne die kumulativen Summen $P_b(k)$ für alle $k = 0, \ ..., \ L - 1$
3. Berechne die kumulativen Mittel $\mu(k)$ für alle $k = 0, \ ..., \ L - 1$
4. Berechne das globale Mittel μ_T
5. Berechne daraus die Zwischen-Klassen-Varianz ${\sigma^2}_B(k)$ für alle $k = 0, \ ..., \ L - 1$
6. Bestimme das k^*, für das ${\sigma^2}_B(k^*) = \text{Max}\left({\sigma^2}_B(k)\right)$ für $0 \le k \le L - 1$

Anwendung des schnellen Otsu-Binarisierungsverfahrens

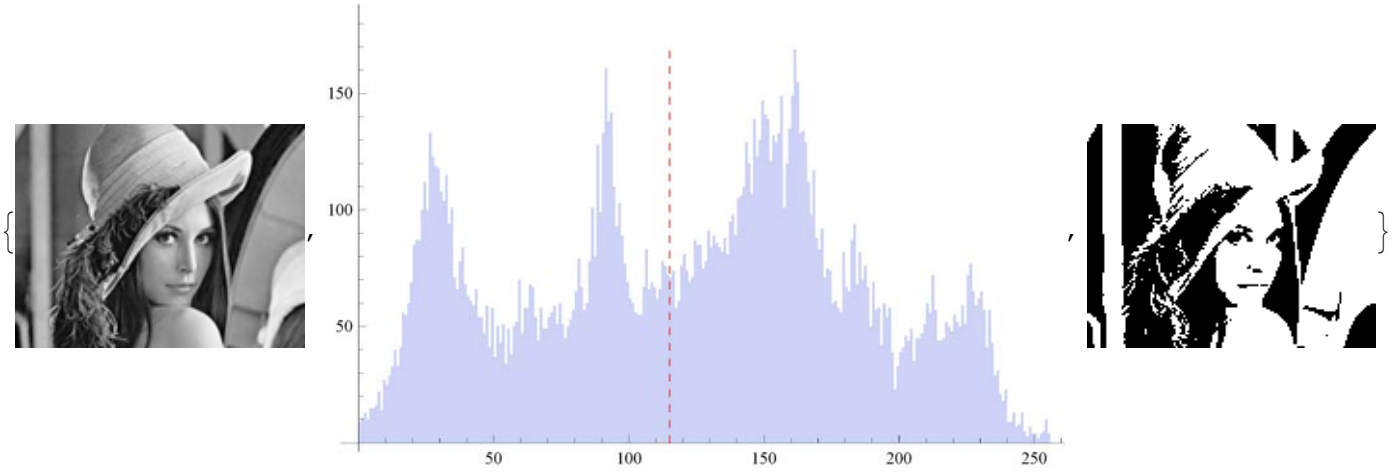
```
{Show[#1], Show[Histogram[Flatten[ImageData[#1, "Byte"]], {1}],
    Graphics[{Red, Dashed, Line[{{#2, 0}, {#2, Last[First[Sort[Tally[Flatten[ImageData[#1, "Byte"]]], #1[[2]] > #2[[2]] &]]]}}]}],
    ImageSize → 384], Show[Image[MyBinarize[ImageData[#1, "Byte"], #2]]]} &[Sequence @@ ({#, OtsuVersion4[ImageData[#, "Byte"]]}) &[
  Image[ImageData[ExampleData[{"TestImage", "Gray21"}], "Real"][[ ;; ;; 2, ;; ;; 2]]], "Real"]]
```

116



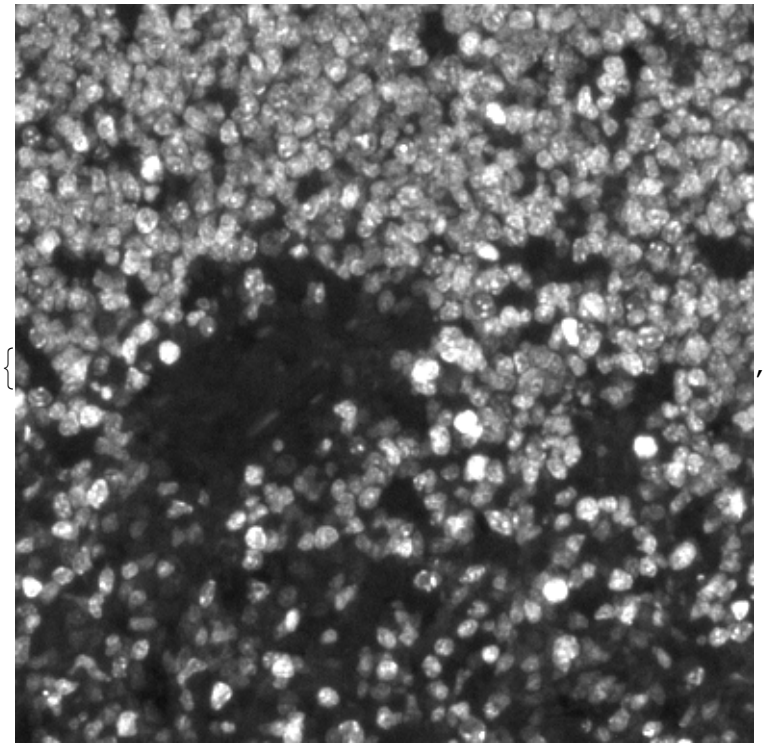

```
{Show[#1], Show[Histogram[Flatten[ImageData[#1, "Byte"]], {1}],
    Graphics[{Red, Dashed, Line[{{#2, 0}, {#2, Last[First[Sort[Tally[Flatten[ImageData[#1, "Byte"]]], #1[[2]] > #2[[2]] &]]]}}]}],
    ImageSize → 384], Show[Image[MyBinarize[ImageData[#1, "Byte"], #2]]]} &[
 Sequence @@ ({#, OtsuVersion4[ImageData[#, "Byte"]]}) &[ImageResize[lenay, Scaled[1 / 1]]]]
```

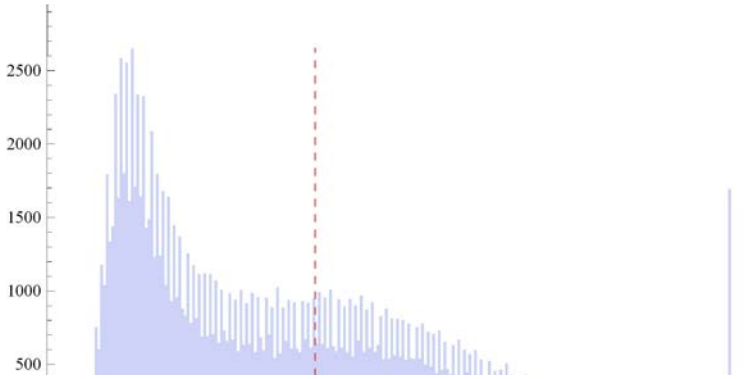
115

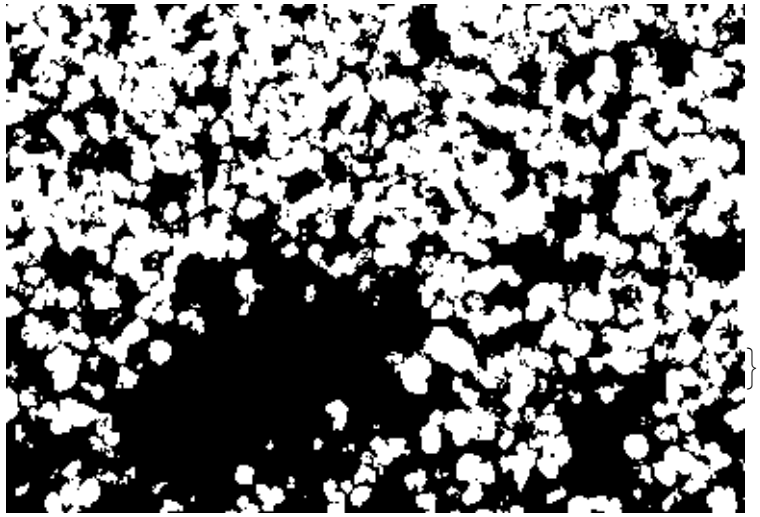


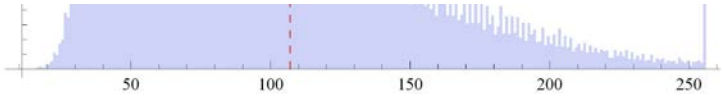
```
{Show[#1], Show[Histogram[Flatten[ImageData[#1, "Byte"]], {1}],
    Graphics[{Red, Dashed, Line[{{#2, 0}, {#2, Last[First[Sort[Tally[Flatten[ImageData[#1, "Byte"]]], #1[[2]] > #2[[2]] &]]]}}]}],
    ImageSize → 384], Show[Image[MyBinarize[ImageData[#1, "Byte"], #2]]]} &[
 Sequence @@ ({#, OtsuVersion4[ImageData[#, "Byte"]]}) &[ImageCrop[First@ColorSeparate@Ton3CD21dreikanalausgleichF2, {384, 384}]]]
```

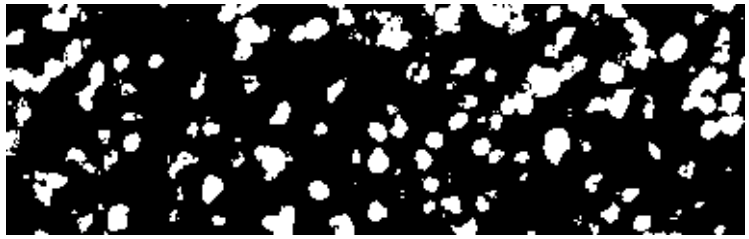
107





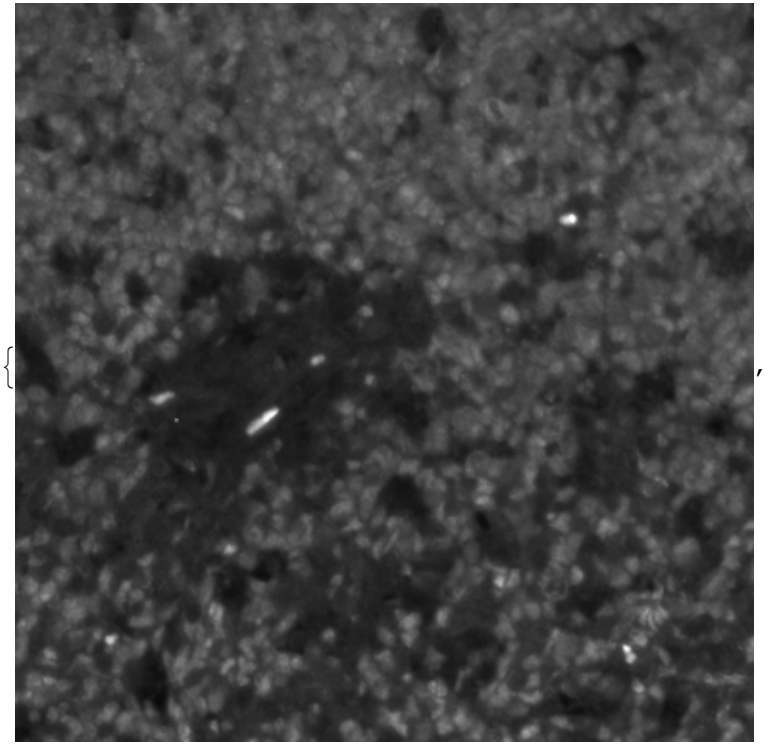


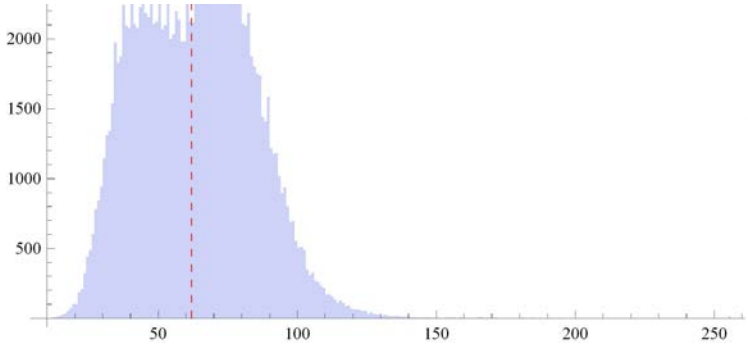




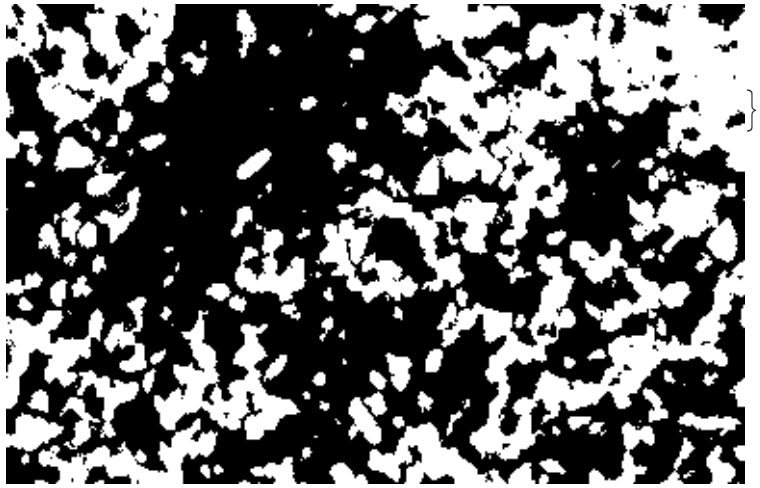
```
{Show[#1], Show[Histogram[Flatten[ImageData[#1, "Byte"]], {1}],
    Graphics[{Red, Dashed, Line[{{#2, 0}, {#2, Last[First[Sort[Tally[Flatten[ImageData[#1, "Byte"]]], #1[[2]] > #2[[2]] &]]]}}]}],
    ImageSize → 384], Show[Image[MyBinarize[ImageData[#1, "Byte"], #2]]]} &[
 Sequence @@ ({#, OtsuVersion4[ImageData[#, "Byte"]]}) &[ImageCrop[First@Rest@ColorSeparate@Ton3CD21dreikanalausgleichF2, {384, 384}]]]
```

62

{  ,



,

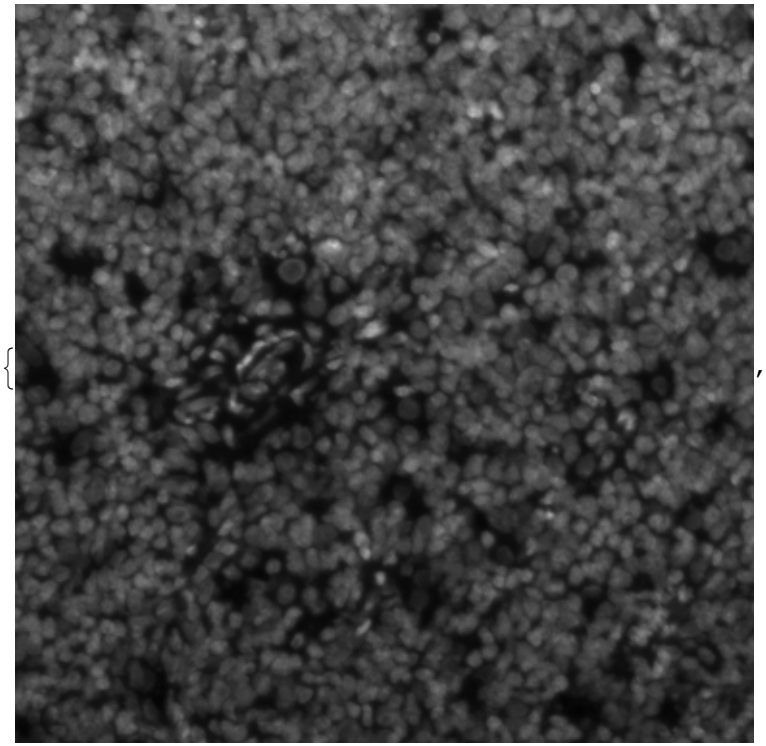


}

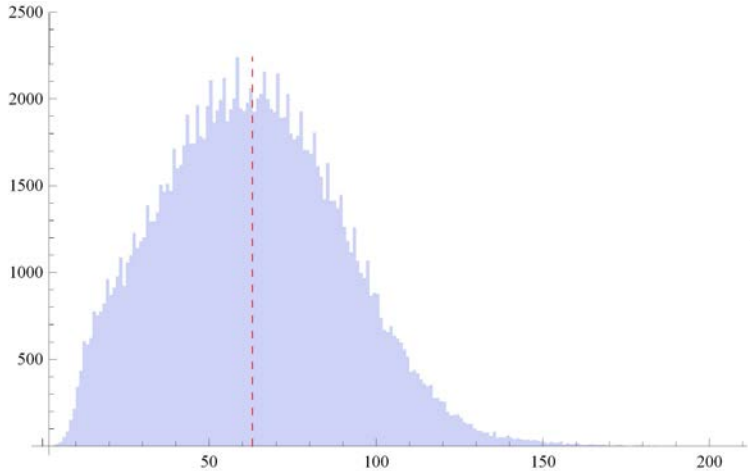
```
{Show[#1], Show[Histogram[Flatten[ImageData[#1, "Byte"]], {1}],
    Graphics[{Red, Dashed, Line[{{#2, 0}, {#2, Last[First[Sort[Tally[Flatten[ImageData[#1, "Byte"]]], #1[[2]] > #2[[2]] &]]]}}]}],
    ImageSize → 384], Show[Image[MyBinarize[ImageData[#1, "Byte"], #2]]]} &[
 Sequence @@ ({#, OtsuVersion4[ImageData[#, "Byte"]]}) &[ImageCrop[Last@ColorSeparate@Ton3CD21dreikanalausgleichF2, {384, 384}]]]
```

63

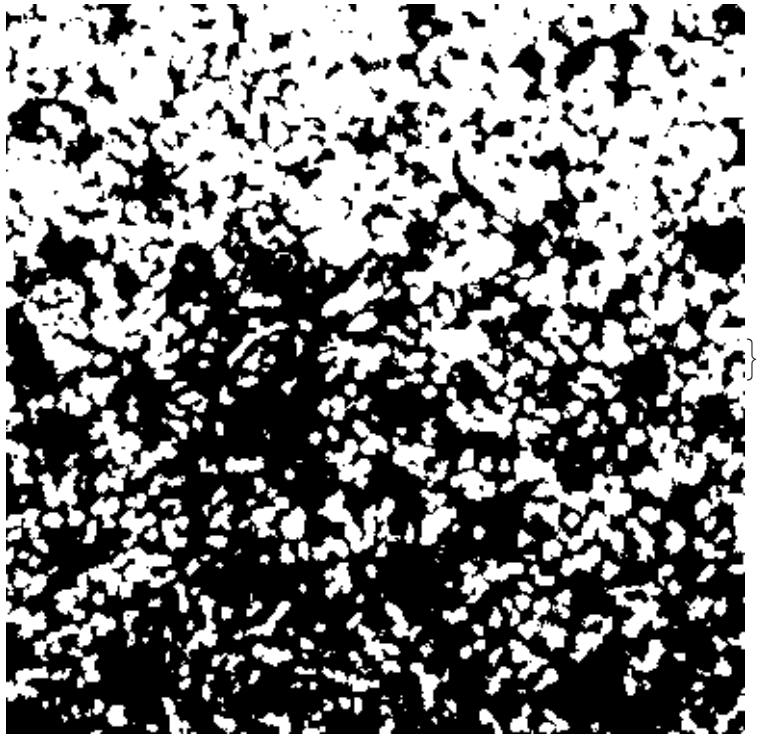
{



,



,

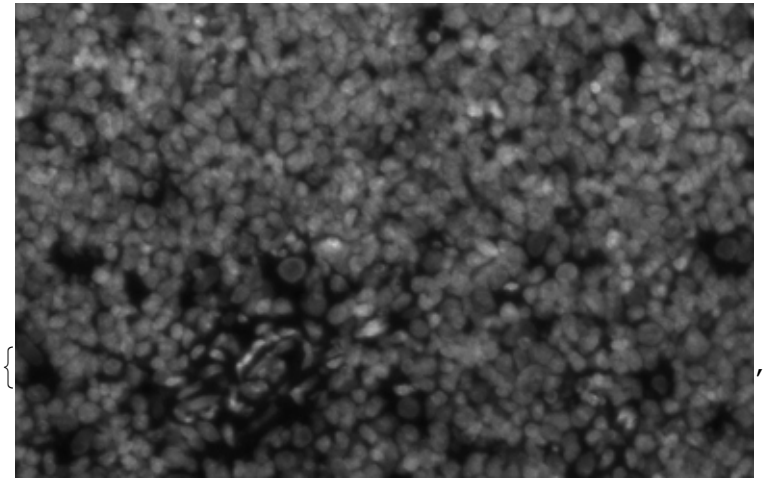


}

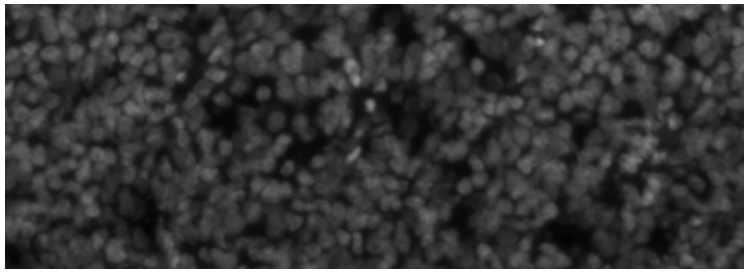
In *Mathematica* ist bereits eine Funktion eingebaut, die das Otsu-Verfahren realisiert:

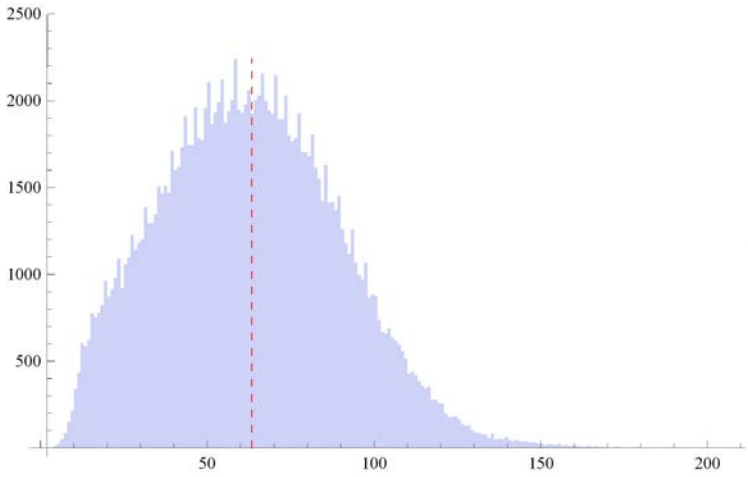
```
{Show[#1], Show[Histogram[Flatten[ImageData[#1, "Byte"]], {1}],
    Graphics[{Red, Dashed, Line[{{#2, 0}, {#2, Last[First[Sort[Tally[Flatten[ImageData[#1, "Byte"]]], #1[[2]] > #2[[2]] &]]]}}]}],
    ImageSize → 384], Show[Image[MyBinarize[ImageData[#1, "Byte"], #2]]]} &[
 Sequence @@ ({#, FindThreshold[ImageData[#, "Real"], Method → "Cluster"] * 255}) &[
  ImageCrop[Last@ColorSeparate@Ton3CD21dreikanalausgleichF2, {384, 384}]]]
```

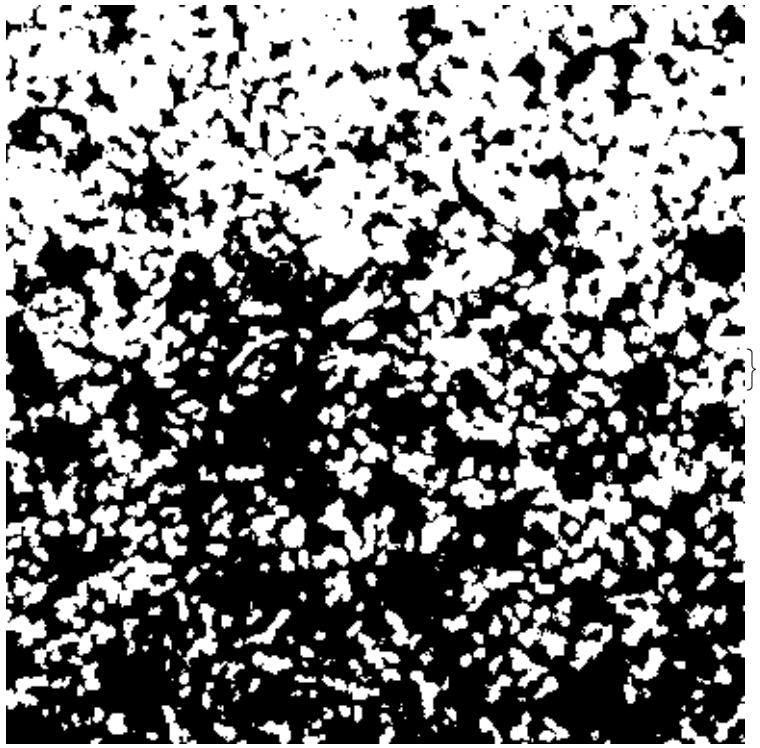
{



,







19. Binarisierung mit dem ISODATA-Verfahren

Iterative Self-Organising Data Analysis Technique (ISODATA, Ridler und Calvard 1978)

→ “Picture thresholding using an iterative selection method”

→ kann als Spezialfall des (nachfolgend dargestellten) k-Means-Verfahren angesehen werden (2-Means, skalare Daten)

1. Bestimmung eines Startschwellwertes
2. Einteilung aller Bildwerte und unter- und überschwellige Mengen
3. Bestimmung jeweils eines Mittelwertes für beide Mengen
4. Bestimmung eines neues Schwellwertes als Mittelwert beider Mittelwerte
5. Wiederholung der Schritte 2 bis 4, bis Mittelwerte konvergieren und die Mengeneinteilung unverändert bleibt

```
Clear[MyISODATA];
MyISODATA[bm_] := Module[{bin, pos, neueschwelle, mean0, mean1, lenpos0, lenpos1, startschwellwert = N[(Max[bm] - Min[bm]) / 2]},
    Print[{{"N.A.", "N.A."},
      N[Mean[Map[bm[[Sequence @@ #]] &, Position[MyBinarize[bm, startschwellwert], #], {1}]] & /@ {0, 1}], startschwellwert}];
    Clear[schritte];
    schritte[alteschwelle_] := Module[{},
      bin = Map[(If[# < alteschwelle, 0, 1]) &, bm, {2}];
      pos = Position[bin, 0];
      lenpos0 = Length[pos];
      If[pos ≠ {}, mean0 = N[Mean[Map[bm[[Sequence @@ #]] &, pos, {1}]]], mean0 = 0.];
      pos = Position[bin, 1];
      lenpos1 = Length[pos];
      If[pos ≠ {}, mean1 = N[Mean[Map[bm[[Sequence @@ #]] &, pos, {1}]]], mean1 = 0.];
      neueschwelle = N[(mean0 + mean1) / 2];
      Print[{{lenpos0, lenpos1}, {mean0, mean1}, neueschwelle}];
      neueschwelle
     ];
    FixedPoint[schritte, N[startschwellwert]]
   ];

daten = {{1, 2, 3}, {2, 3, 4}, {0, 2, 4}, {2, 1, 0}};
MyISODATA[daten]
```

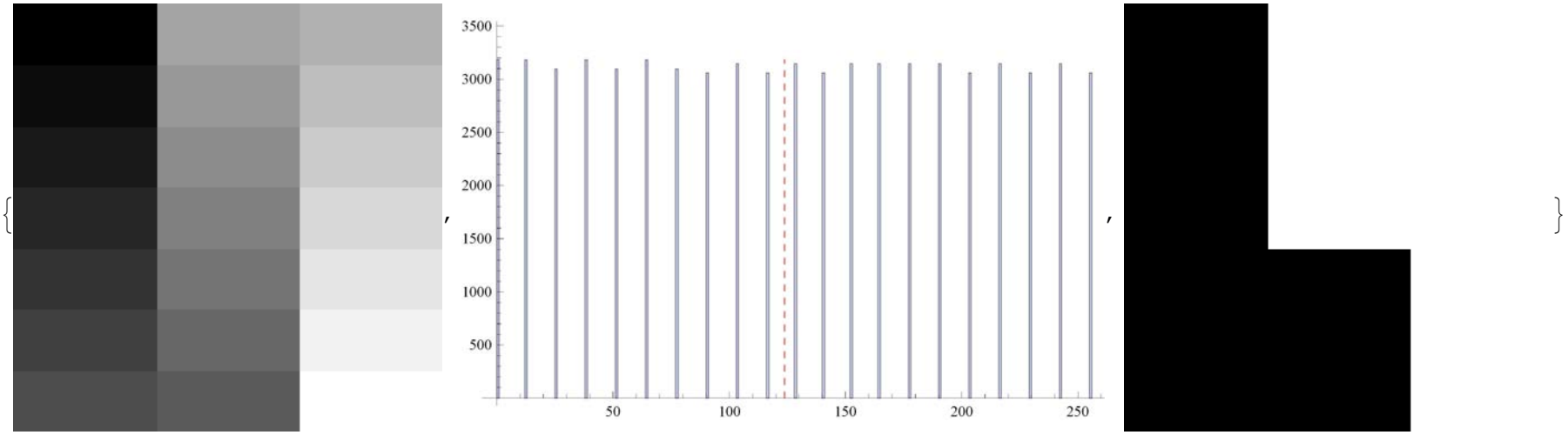
```
{{N.A., N.A.}, {1.25, 3.5}, 2.}

{{4, 8}, {0.5, 2.75}, 1.625}

{{4, 8}, {0.5, 2.75}, 1.625}

1.625
```

```
(*ISODATA anhand eines synthetischen, regelmäßig gestuften Bildes*)
{Show[#1], Show[Histogram[Flatten[ImageData[#1, "Byte"]], {1}],
    Graphics[{Red, Dashed, Line[{{#2, 0}, {#2, Last[First[Sort[Tally[Flatten[ImageData[#1, "Byte"]]], #1[[2]] > #2[[2]] &]]]}}]},
    ImageSize → 384], Show[Image[MyBinarize[ImageData[#1, "Byte"], #2]]]} &[Sequence @@ ({#, MyISODATA[ImageData[#, "Real"] * 255]}) &[
  Image[ImageData[ExampleData[{"TestImage", "Gray21"}], "Real"][[ ;; ;; 2, ;; ;; 2]]], "Real"]]
```

```
{{N.A., N.A.}, {57.2466, 190.385}, 127.5}
```

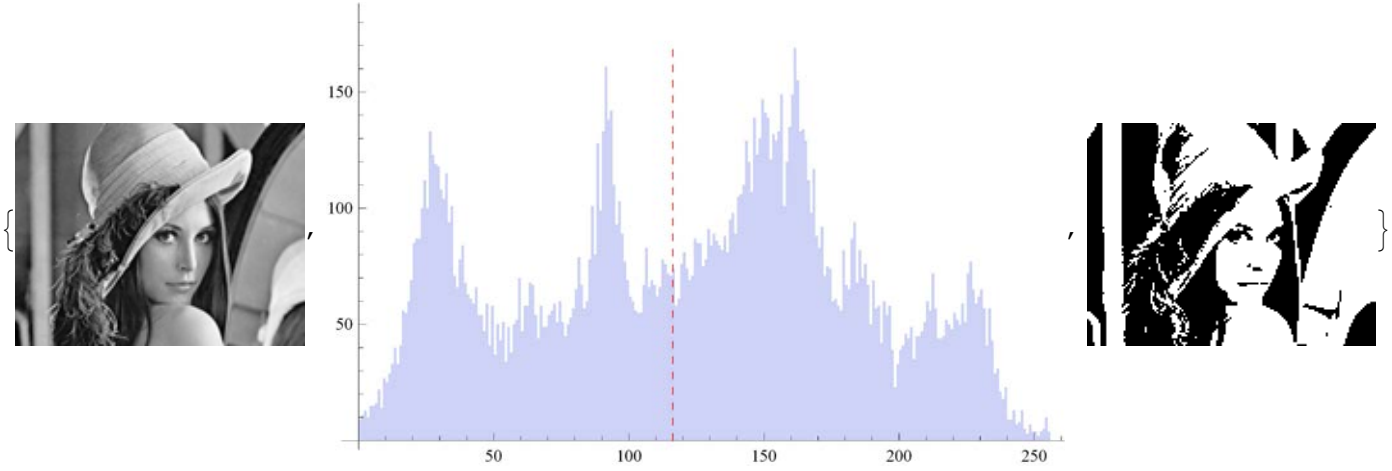
```
{{31 281, 34 255}, {57.2466, 190.385}, 123.816}
```

```
{{31 281, 34 255}, {57.2466, 190.385}, 123.816}
```



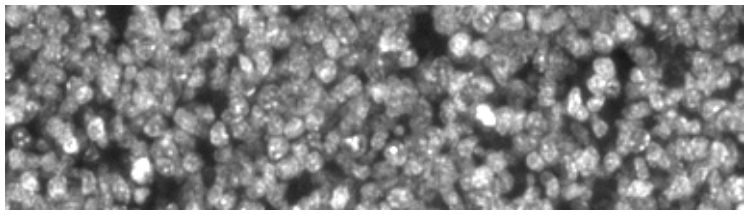

```
(*ISODATA anhand von Lena*)
{Show[#1], Show[Histogram[Flatten[ImageData[#1, "Byte"]], {1}],
    Graphics[{Red, Dashed, Line[{{#2, 0}, {#2, Last[First[Sort[Tally[Flatten[ImageData[#1, "Byte"]]], #1[[2]] > #2[[2]] &]]]}}]},
    ImageSize → 384], Show[Image[MyBinarize[ImageData[#1, "Byte"], #2]]]} &[
 Sequence @@ ({#, MyISODATA[ImageData[#, "Real"] * 255]}) &[Image[ImageData[lenay, "Real"][[ ;; ;; 1, ;; ;; 1]]], "Real"]]
```

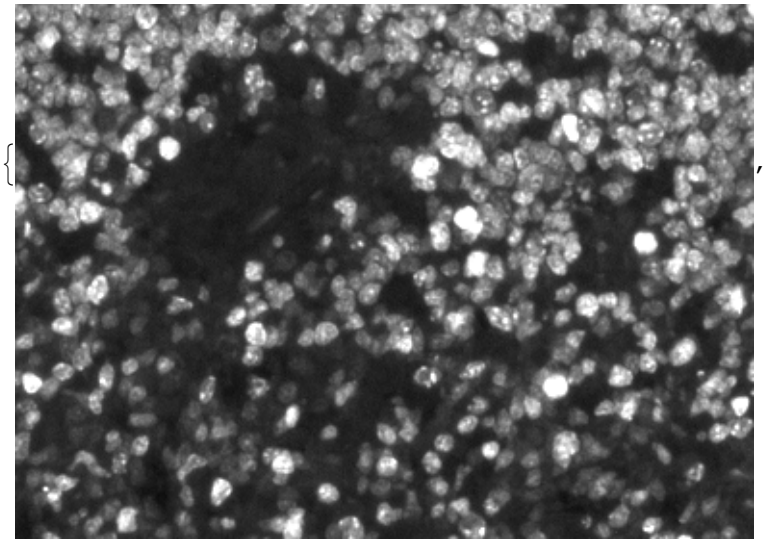
```
{{N.A., N.A.}, {68.7371, 173.697}, 127.5}
{{8560, 8840}, {68.7371, 173.697}, 121.217}
{{8088, 9312}, {65.4786, 171.207}, 118.343}
{{7857, 9543}, {63.8759, 169.967}, 116.922}
{{7738, 9662}, {63.051, 169.321}, 116.186}
{{7738, 9662}, {63.051, 169.321}, 116.186}
```

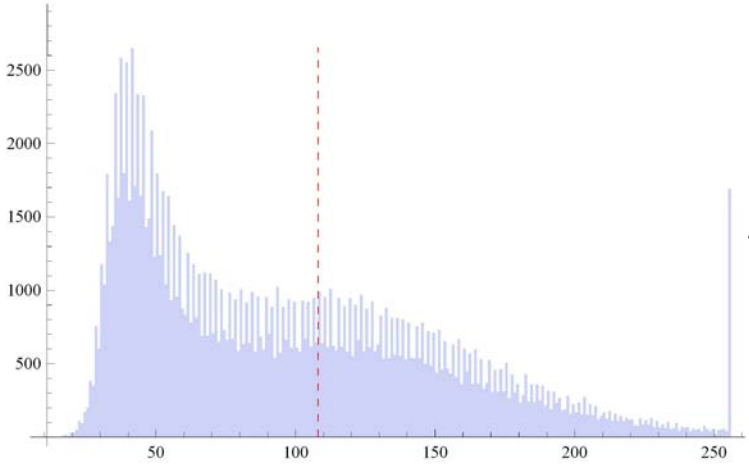


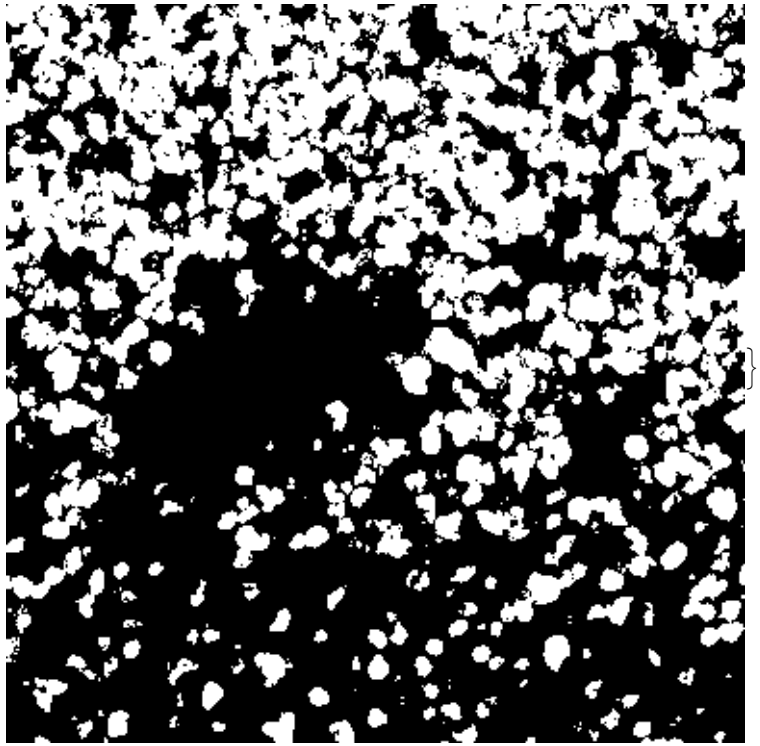

```
(*ISODATA anhand von Elaine*)
{Show[#1], Show[Histogram[Flatten[ImageData[#1, "Byte"]], {1}],
    Graphics[{Red, Dashed, Line[{{#2, 0}, {#2, Last[First[Sort[Tally[Flatten[ImageData[#1, "Byte"]]], #1[[2]] > #2[[2]] &]]]}}]}],
    ImageSize → 384], Show[Image[MyBinarize[ImageData[#1, "Byte"], #2]]]} &[
 Sequence @@ ({#, MyISODATA[ImageData[#, "Real"] * 255]}) &[ImageCrop[First@ColorSeparate@Ton3CD21dreikanalausgleichF2, {384, 384}]]]
```

```
{{N.A., N.A.}, {65.9134, 165.411}, 122.}
{{102 109, 45 347}, {65.5492, 164.777}, 115.163}
{{97 616, 49 840}, {63.108, 160.613}, 111.86}
{{94 450, 53 006}, {61.42, 157.797}, 109.608}
{{92 883, 54 573}, {60.5939, 156.435}, 108.515}
{{92 242, 55 214}, {60.2575, 155.885}, 108.071}
{{92 242, 55 214}, {60.2575, 155.885}, 108.071}
```



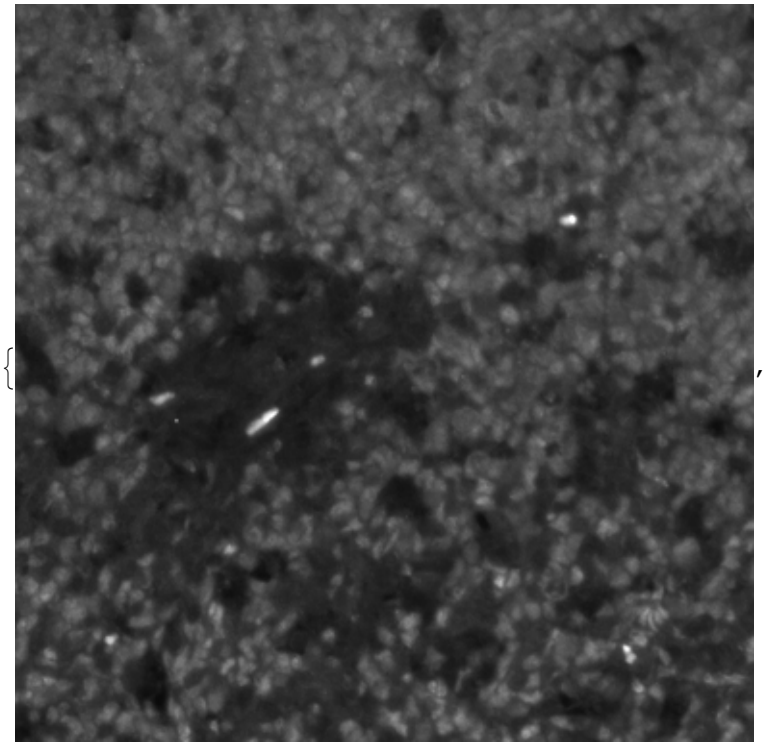


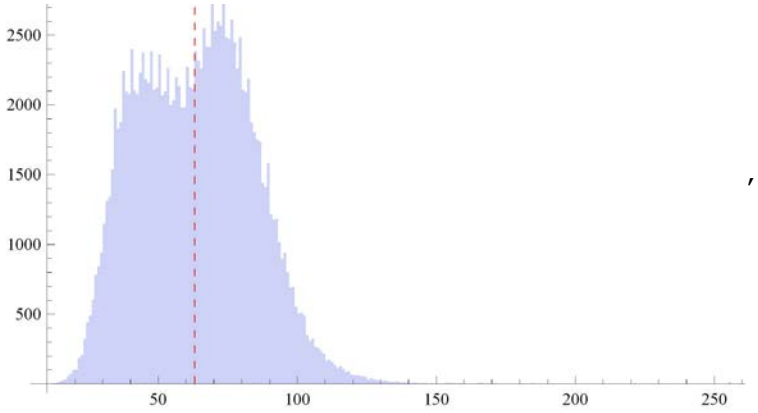


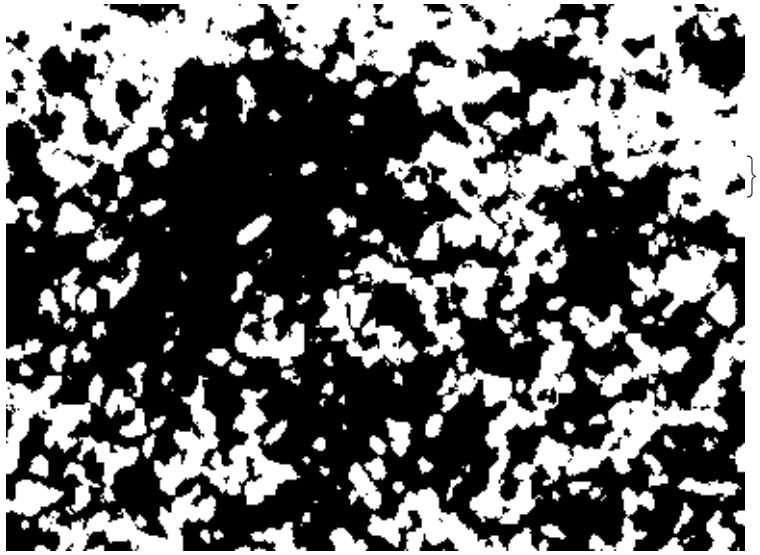


```
{Show[#1], Show[Histogram[Flatten[ImageData[#1, "Byte"]], {1}],
    Graphics[{Red, Dashed, Line[{{#2, 0}, {#2, Last[First[Sort[Tally[Flatten[ImageData[#1, "Byte"]]], #1[[2]] > #2[[2]] &]]]}}]}],
    ImageSize → 384], Show[Image[MyBinarize[ImageData[#1, "Byte"], #2]]]} &[
 Sequence @@ ({#, MyISODATA[ImageData[#, "Real"] * 255]}) &[ImageCrop[First@Rest@ColorSeparate@Ton3CD21dreikanalausgleichF2, {384, 384}]]]
```

```
{{N.A., N.A.}, {62.5171, 145.231}, 122.5}
{{146 815, 641}, {62.5171, 145.231}, 103.874}
{{143 795, 3661}, {61.5161, 116.315}, 88.9154}
{{131 183, 16 273}, {58.3616, 99.2737}, 78.8177}
{{112 292, 35 164}, {54.2103, 90.5514}, 72.3809}
{{97 273, 50 183}, {50.9353, 86.0232}, 68.4793}
{{86 791, 60 665}, {48.5759, 83.3362}, 65.956}
{{79 408, 68 048}, {46.8646, 81.5618}, 64.2132}
{{77 148, 70 308}, {46.3333, 81.0294}, 63.6814}
{{74 831, 72 625}, {45.7863, 80.4861}, 63.1362}
{{74 831, 72 625}, {45.7863, 80.4861}, 63.1362}
```

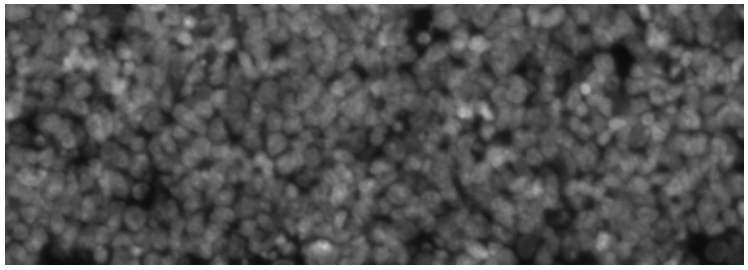


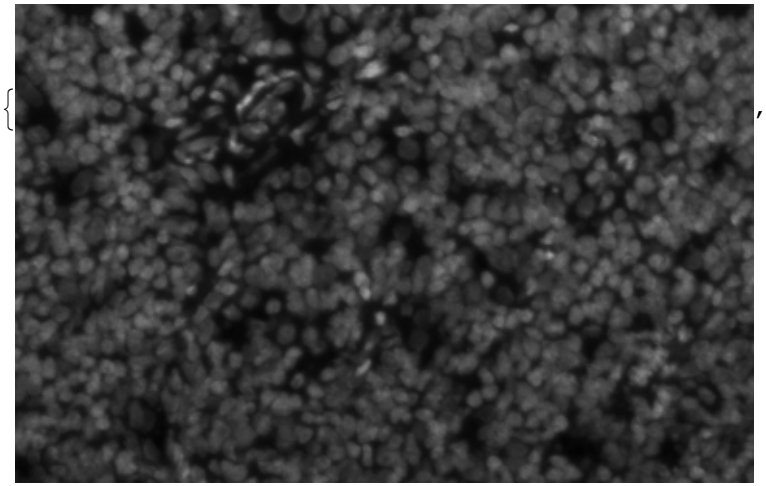


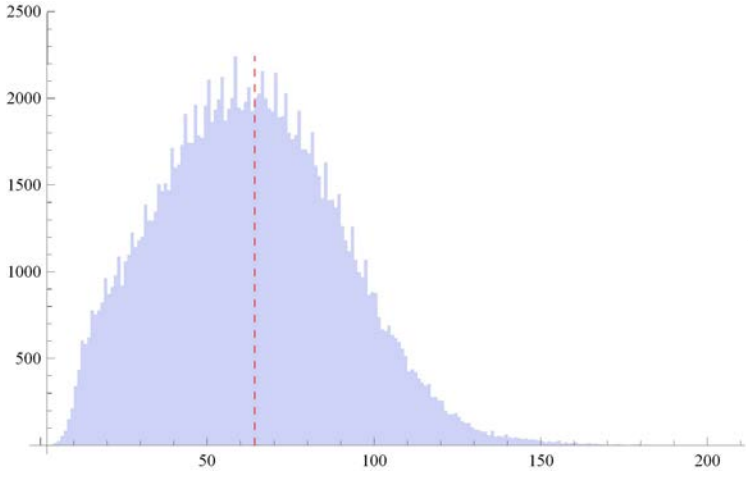


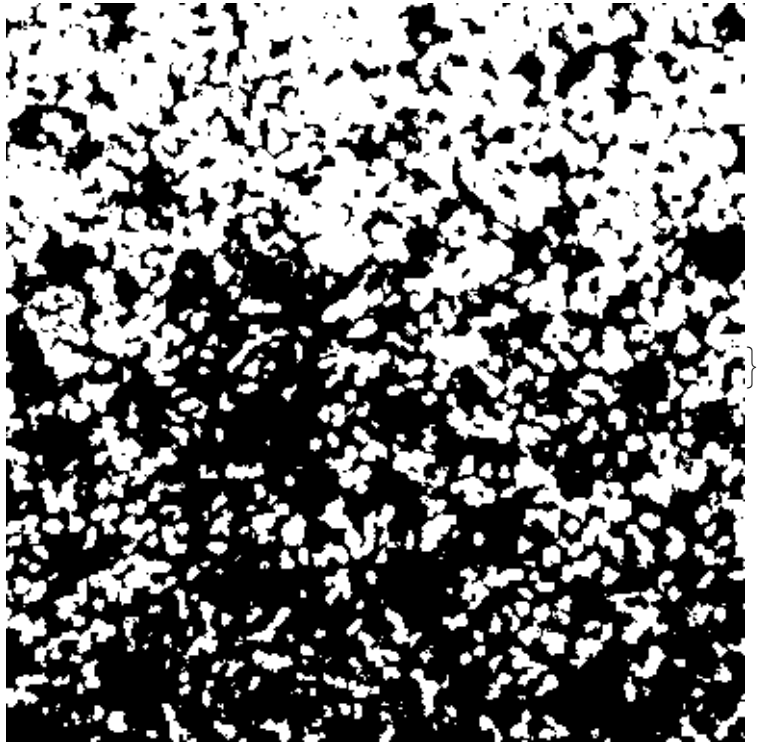
```
{Show[#1], Show[Histogram[Flatten[ImageData[#1, "Byte"]], {1}],
    Graphics[{Red, Dashed, Line[{{#2, 0}, {#2, Last[First[Sort[Tally[Flatten[ImageData[#1, "Byte"]]], #1[[2]] > #2[[2]] &]]]}]}],
    ImageSize → 384], Show[Image[MyBinarize[ImageData[#1, "Byte"], #2]]]} &[
 Sequence @@ ({#, MyISODATA[ImageData[#, "Real"] * 255]}) &[ImageCrop[Last@ColorSeparate@Ton3CD21dreikanalausgleichF2, {384, 384}]]]
```

```
{{N.A., N.A.}, {58.1711, 115.857}, 102.}
{{135 967, 11 489}, {57.9557, 115.051}, 86.5033}
{{119 450, 28 006}, {53.0806, 102.171}, 77.626}
{{104 925, 42 531}, {49.0979, 95.2316}, 72.1647}
{{95 616, 51 840}, {46.5783, 91.5944}, 69.0864}
{{89 683, 57 773}, {44.9655, 89.4751}, 67.2203}
{{85 816, 61 640}, {43.9051, 88.1591}, 66.0321}
{{83 816, 63 640}, {43.3541, 87.4941}, 65.4241}
{{81 659, 65 797}, {42.7559, 86.7895}, 64.7727}
{{79 631, 67 825}, {42.1894, 86.1379}, 64.1637}
{{79 631, 67 825}, {42.1894, 86.1379}, 64.1637}
```





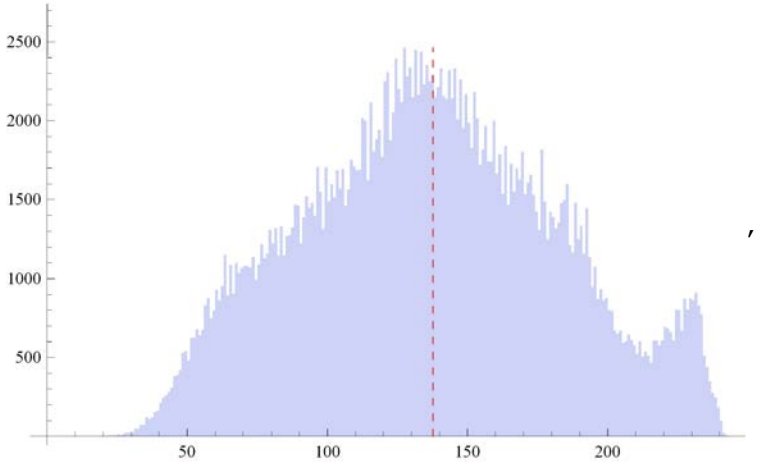




```
{Show[#1], Show[Histogram[Flatten[ImageData[#1, "Byte"]], {1}],
    Graphics[{Red, Dashed, Line[{{#2, 0}, {#2, Last[First[Sort[Tally[Flatten[ImageData[#1, "Byte"]]], #1[[2]] > #2[[2]] &]]]}}]}],
    ImageSize → 384], Show[Image[MyBinarize[ImageData[#1, "Byte"], #2]]]} &[
 Sequence @@ ({#, MyISODATA[ImageData[#, "Real"] * 255]}) &[ExampleData[{"TestImage", "Elaine"}]]]
```

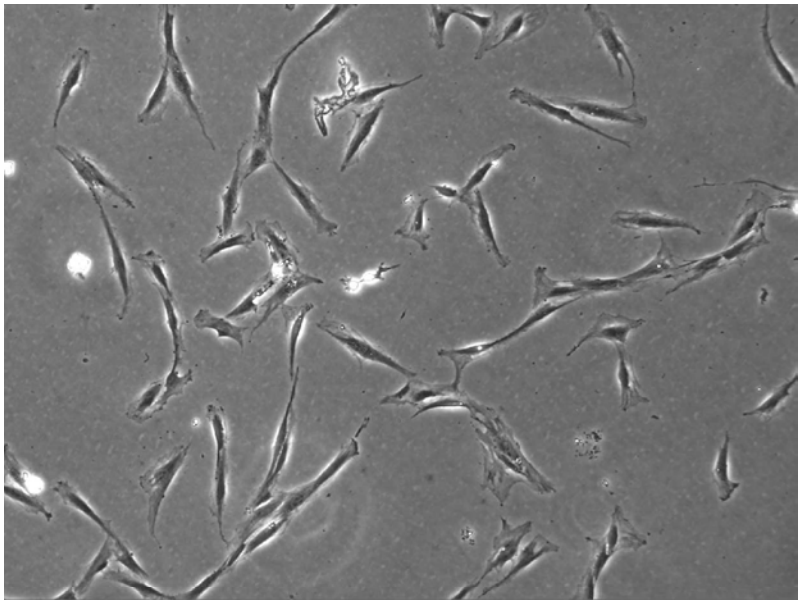
```
{{N.A., N.A.}, {89.379, 165.168}, 121.5}
{{99 655, 162 489}, {89.379, 165.168}, 127.273}
{{112 761, 149 383}, {93.4741, 168.726}, 131.1}
{{121 976, 140 168}, {96.1971, 171.304}, 133.75}
{{126 578, 135 566}, {97.518, 172.62}, 135.069}
{{131 161, 130 983}, {98.8107, 173.953}, 136.382}
{{133 411, 128 733}, {99.4379, 174.617}, 137.027}
{{135 701, 126 443}, {100.072, 175.298}, 137.685}
{{135 701, 126 443}, {100.072, 175.298}, 137.685}
```

{  ,



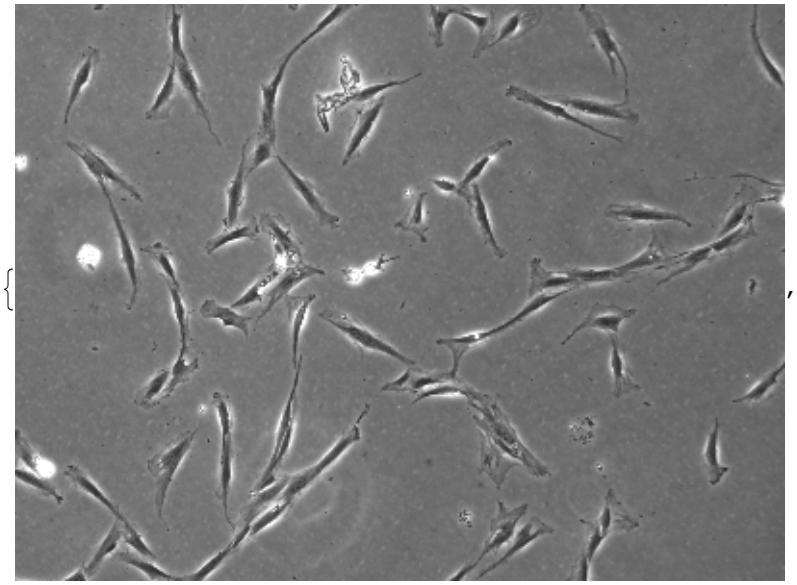
,

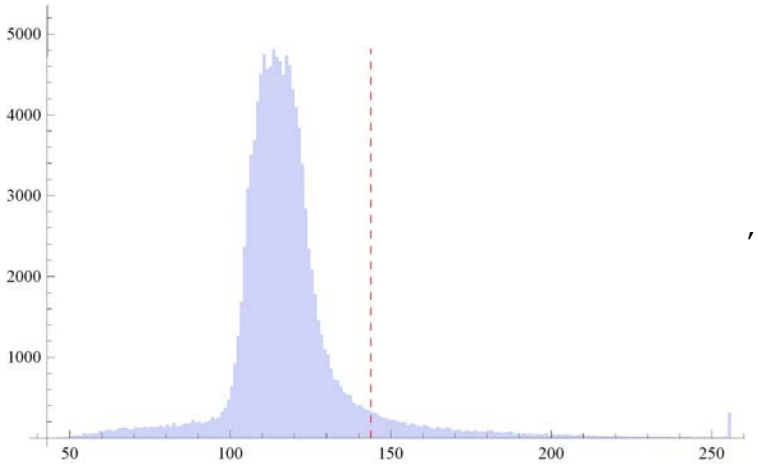
phasenkontrast

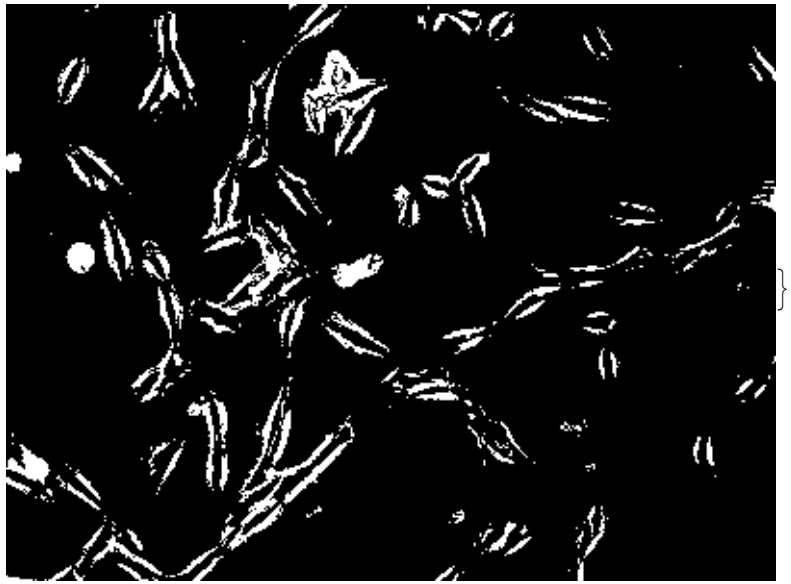


```
{Show[#1], Show[Histogram[Flatten[ImageData[#1, "Byte"]], {1}],
    Graphics[{Red, Dashed, Line[{{#2, 0}, {#2, Last[First[Sort[Tally[Flatten[ImageData[#1, "Byte"]]], #1[[2]] > #2[[2]] &]]]}}]}],
    ImageSize → 384], Show[Image[MyBinarize[ImageData[#1, "Byte"], #2]]]} &[
 Sequence @@ ({#, MyISODATA[ImageData[#, "Real"] * 255]}) &[Image@ImageData[phasenkontrast][[ ;; ;; 4, ;; ;; 4]]]]
```

```
{{N.A., N.A.}, {96.1443, 122.488}, 106.}
{{17 387, 102 613}, {94.1587, 121.925}, 108.042}
{{28 739, 91 261}, {99.2541, 123.775}, 111.514}
{{42 564, 77 436}, {102.746, 126.233}, 114.489}
{{56 695, 63 305}, {105.304, 129.185}, 117.244}
{{70 590, 49 410}, {107.41, 132.891}, 120.151}
{{83 612, 36 388}, {109.209, 137.877}, 123.543}
{{93 687, 26 313}, {110.574, 143.994}, 127.284}
{{101 367, 18 633}, {111.69, 151.696}, 131.693}
{{105 641, 14 359}, {112.405, 158.348}, 135.377}
{{108 283, 11 717}, {112.917, 163.976}, 138.446}
{{109 783, 10 217}, {113.245, 167.946}, 140.595}
{{110 595, 9405}, {113.438, 170.402}, 141.92}
{{110 970, 9030}, {113.531, 171.623}, 142.577}
{{111 319, 8681}, {113.62, 172.814}, 143.217}
{{111 662, 8338}, {113.71, 174.04}, 143.875}
{{111 662, 8338}, {113.71, 174.04}, 143.875}
```

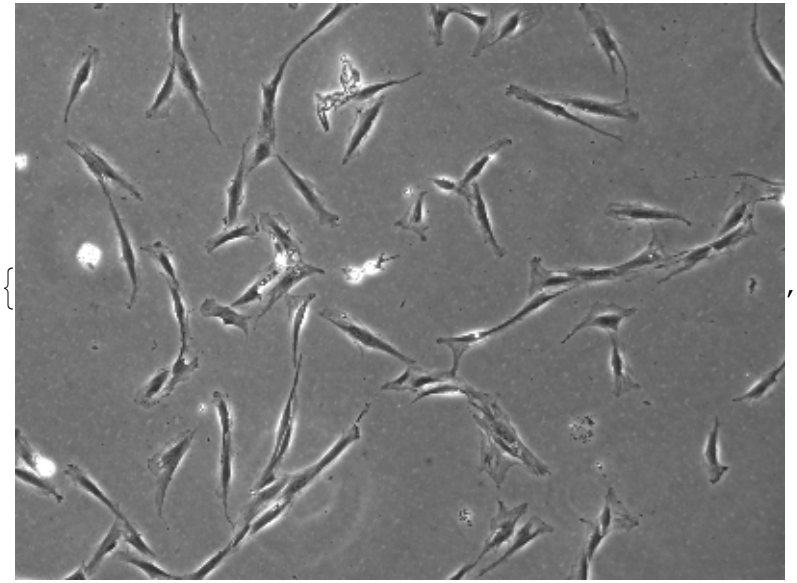
{  ,

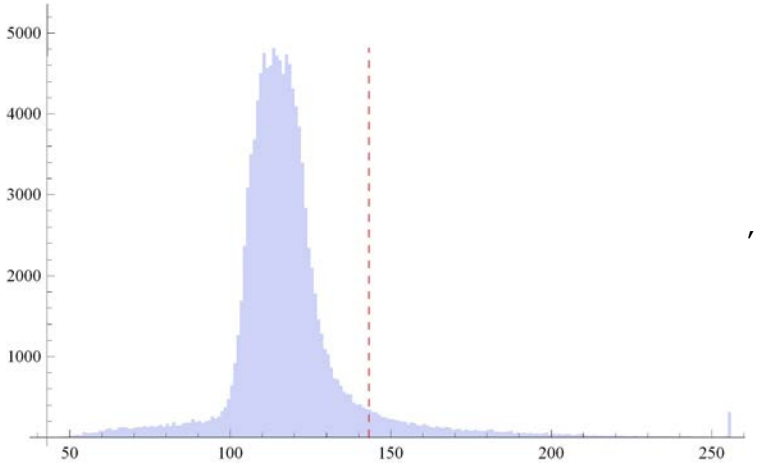

5000
4000
3000
2000
1000
50
100
150
200
250

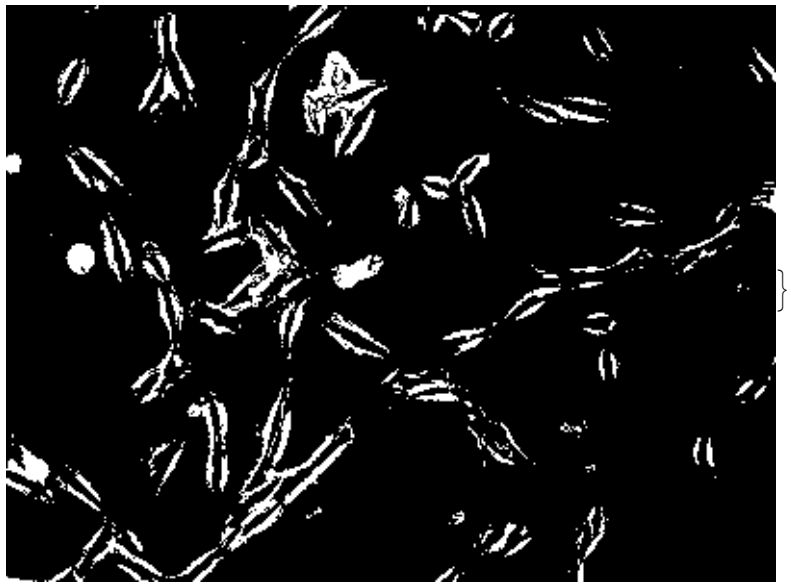
,  }

Zum Vergleich noch die Anwendung des Otsu-Verfahrens mit sehr ähnlichem Ergebnis:

```
{Show[#1], Show[Histogram[Flatten[ImageData[#1, "Byte"]], {1}],
    Graphics[{Red, Dashed, Line[{{#2, 0}, {#2, Last[First[Sort[Tally[Flatten[ImageData[#1, "Byte"]]], #1[[2]] > #2[[2]] &]]]}]}],
    ImageSize → 384], Show[Image[MyBinarize[ImageData[#1, "Byte"], #2]]]} &[
 Sequence @@ ({#, FindThreshold[ImageData[#, "Real"], Method → "Cluster"] * 255}) &[Image@ImageData[phasenkontrast][[ ;; ;; 4, ;; ;; 4]]]]
```







Verallgemeinerung des ISODATA-Verfahrens für vektoriellen Daten (2-Means)

1. Bestimmung zweier Startmittelwertvektoren
2. Einteilung aller Bildwerte in zwei Mengen entsprechend der je Pixel kürzesten Euklidschen Distanz zu einem der Mittelwertvektoren
3. Bestimmung jeweils eines neuen Mittelwertvektors für beide Mengen
4. Wiederholung der Schritte 2 bis 3, bis Mittelwertvektoren konvergieren und die Mengeneinteilung unverändert bleibt

```
Clear[MyISODATARGB];
MyISODATARGB[bm_] :=
  Module[
   {bin, pos, neuesmean0, neuesmean1, lenpos0, lenpos1, altesmean0 = N[Mean[Flatten[bm, 1]] - .1], altesmean1 = N[Mean[Flatten[bm, 1]] + .1]},
   Print[{{"N.A.", "N.A."}, {altesmean0, altesmean1}}];
   Clear[schrittergb];
   schrittergb[{altesmean0_, altesmean1_}] := Module[{},
     bin = Map[(If[EuclideanDistance[altesmean0, #] < EuclideanDistance[altesmean1, #], 0, 1]) &, bm, {2}];
     pos = Position[bin, 0];
     lenpos0 = Length[pos];
     If[pos ≠ {}, neuesmean0 = N[Mean[Map[bm[[Sequence @@ #]] &, pos, {1}]]], neuesmean0 = altesmean1];
     pos = Position[bin, 1];
     lenpos1 = Length[pos];
     If[pos ≠ {}, neuesmean1 = N[Mean[Map[bm[[Sequence @@ #]] &, pos, {1}]]], neuesmean1 = altesmean0];
     Print[{{lenpos0, lenpos1}, {neuesmean0, neuesmean1}}];
     {neuesmean0, neuesmean1}
    ];
   FixedPoint[schrittergb, {altesmean0, altesmean1}]
  ];

datenrgb = {{{1, 2, 3}, {2, 3, 4}, {4, 5, 6}}, {{3, 2, 2}, {4, 3, 2}, {6, 5, 4}}};
MyISODATARGB[datenrgb]
```

```
{{N.A., N.A.}, {{3.23333, 3.23333, 3.4}, {3.43333, 3.43333, 3.6}}}

{{4, 2}, {{2.5, 2.5, 2.75}, {5., 5., 5.}}}

{{4, 2}, {{2.5, 2.5, 2.75}, {5., 5., 5.}}}

{{2.5, 2.5, 2.75}, {5., 5., 5.}}
```

```
{Show[#1],
   Show[{Graphics3D[({RGBColor[#], PointSize[.0075], Point[#]}) & /@ Flatten[ImageData[#1], 1][[ ;; ;; 25]],
        PlotRange → {{0, 1}, {0, 1}, {0, 1}}, AxesLabel → {Text[Style["R", Red, Italic]], Text[Style["G", Green, Italic]],
          Text[Style["B", Blue, Italic]]}, Axes → True, RotationAction → "Clip", ImageSize → 384],
       Graphics3D[({RGBColor[#], PointSize[.05], Point[#]}) & /@ (#2 / 255)], MyPlane[#2 / 255]}] &[#1, #2],
   Show[Image[MyBinarizeRGB[ImageData[#1, "Byte"], #2]]]} &[Sequence @@ ({#, MyISODATARGB[ImageData[#, "Real"] * 255]}) &[
  ExampleData[{"TestImage", "Girl2"}]]]
```

```
{{N.A., N.A.}, {{137.503, 139.858, 143.918}, {137.703, 140.058, 144.118}}}
```

```
{{19179, 46357}, {{100.899, 116.91, 116.457}, {152.788, 149.493, 155.42}}}
```

```
{{15758, 49778}, {{93.2288, 112.754, 111.649}, {151.65, 148.57, 154.265}}}
```

```
{{14058, 51478}, {{90.3117, 109.373, 108.63}, {150.517, 148.31, 153.682}}}
```

```
{{12834, 52702}, {{88.3887, 106.298, 105.961}, {149.587, 148.155, 153.285}}}
```

```
{{11683, 53853}, {{86.4978, 102.923, 103.059}, {148.69, 147.992, 152.904}}}
```

```
{{10469, 55067}, {{84.3062, 98.7337, 99.4834}, {147.735, 147.795, 152.484}}}
```

```
{{9249, 56287}, {{81.7835, 93.6887, 95.2168}, {146.775, 147.561, 152.037}}}
```

```
{{8229, 57307}, {{79.3297, 88.6614, 90.9727}, {145.971, 147.324, 151.635}}}
```

```
{{7485, 58051}, {{77.3678, 84.3854, 87.3906}, {145.369, 147.123, 151.319}}}
```

```
{{7029, 58507}, {{76.0542, 81.5076, 84.9896}, {144.997, 146.98, 151.109}}}
```

```
{{6806, 58730}, {{75.3911, 80.0242, 83.7592}, {144.812, 146.903, 151.001}}}
```

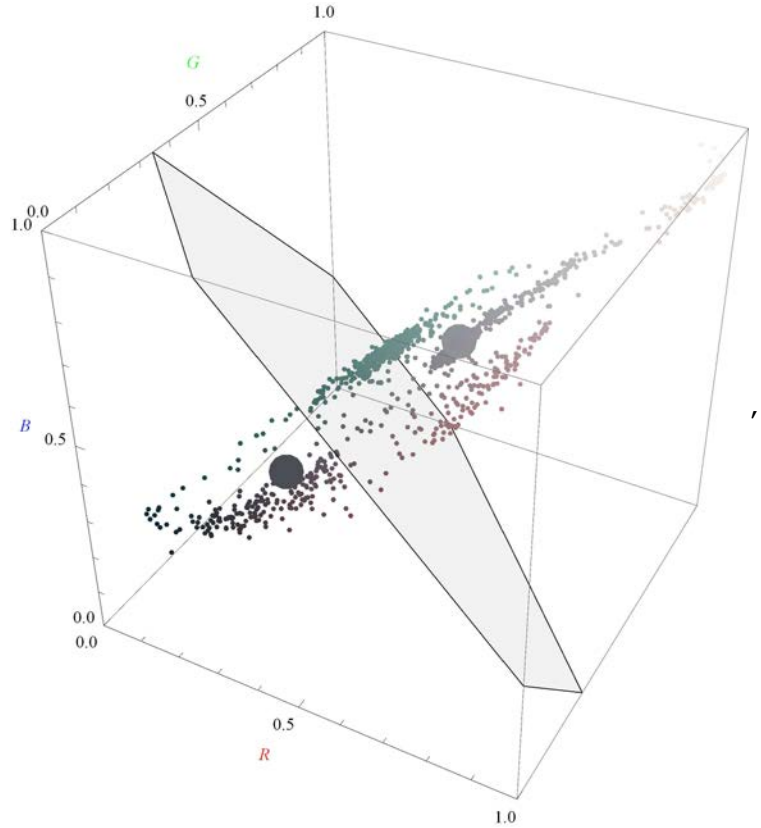
```
{{6713, 58823}, {{75.0897, 79.4095, 83.2473}, {144.737, 146.868, 150.953}}}
```

```
{{6664, 58872}, {{74.9584, 79.063, 82.9664}, {144.694, 146.851, 150.928}}}
```

```
{{6658, 58878}, {{74.9222, 79.033, 82.9399}, {144.691, 146.847, 150.925}}}
```

```
{{6655, 58881}, {{74.9055, 79.0167, 82.9267}, {144.689, 146.846, 150.923}}}
```


```
{{6655, 58881}, {{74.9055, 79.0167, 82.9267}, {144.689, 146.846, 150.923}}}
```



```
{Show[#1],
   Show[{Graphics3D[({RGBColor[#], PointSize[.0075], Point[#]}) & /@ Flatten[ImageData[#1], 1][[ ;; ;; 50]],
         PlotRange → {{0, 1}, {0, 1}, {0, 1}}, AxesLabel → {Text[Style["R", Red, Italic]], Text[Style["G", Green, Italic]],
           Text[Style["B", Blue, Italic]]}, Axes → True, RotationAction → "Clip", ImageSize → 384],
        Graphics3D[({RGBColor[#], PointSize[.05], Point[#]}) & /@ (#2 / 255)], MyPlane[#2 / 255]}] &[#1, #2],
   Show[Image[MyBinarizeRGB[ImageData[#1, "Byte"], #2]]]} &[Sequence @@ ({#, MyISODATARGB[ImageData[#, "Real"] * 255]}) &[
  ImageResize[ExampleData[{"TestImage", "Apples"}], Scaled[1 / 8]]]]
```

{{N.A., N.A.}, {{132.902, 100.159, 47.2799}, {133.102, 100.359, 47.4799}}}

{{41 122, 34 928}, {{85.0307, 38.3013, 17.8529}, {189.48, 173.203, 82.143}}}

{{42 227, 33 823}, {{87.2096, 39.359, 18.4981}, {190.173, 176.29, 83.4379}}}

{{42 705, 33 345}, {{87.9761, 40.0115, 18.7521}, {190.667, 177.417, 84.0435}}}

{{42 948, 33 102}, {{88.3137, 40.3822, 18.8908}, {190.983, 177.945, 84.3428}}}

{{43 051, 32 999}, {{88.4685, 40.5332, 18.9456}, {191.101, 178.177, 84.4757}}}

{{43 093, 32 957}, {{88.5325, 40.5942, 18.968}, {191.148, 178.273, 84.5299}}}

{{43 116, 32 934}, {{88.5629, 40.6298, 18.983}, {191.18, 178.322, 84.5561}}}

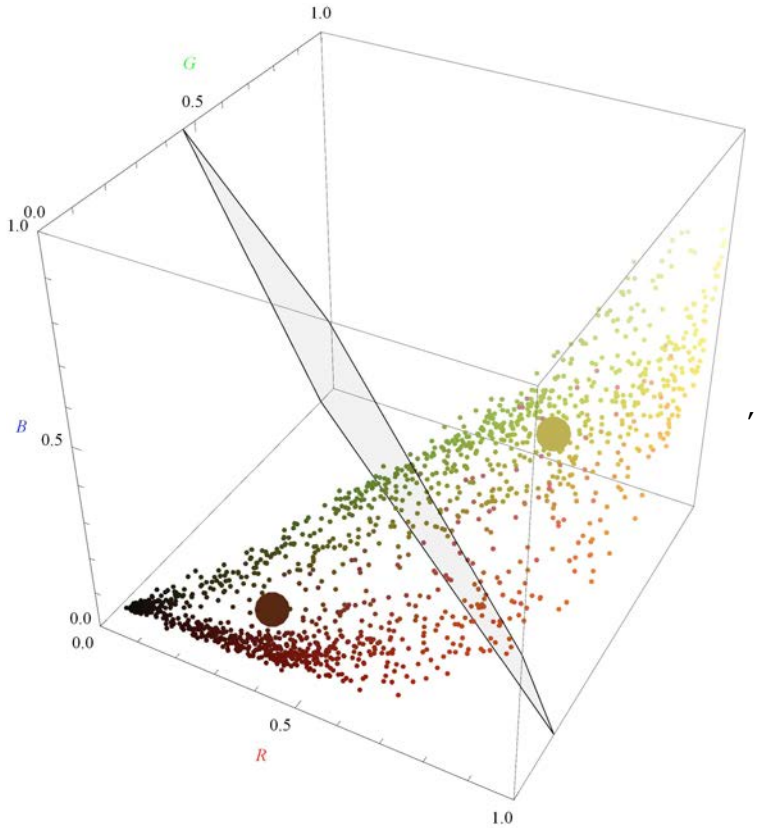
{{43 128, 32 922}, {{88.5802, 40.648, 18.9892}, {191.195, 178.349, 84.5718}}}

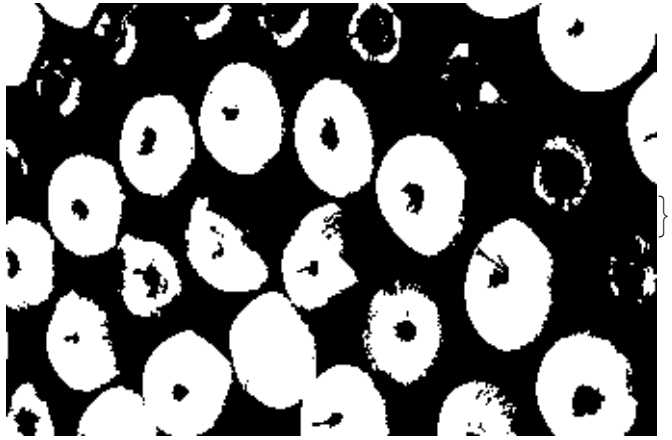
{{43 134, 32 916}, {{88.5849, 40.6598, 18.9931}, {191.207, 178.358, 84.5787}}}

{{43 136, 32 914}, {{88.5856, 40.6643, 18.9943}, {191.213, 178.361, 84.581}}}

{{43 136, 32 914}, {{88.5856, 40.6643, 18.9943}, {191.213, 178.361, 84.581}}}

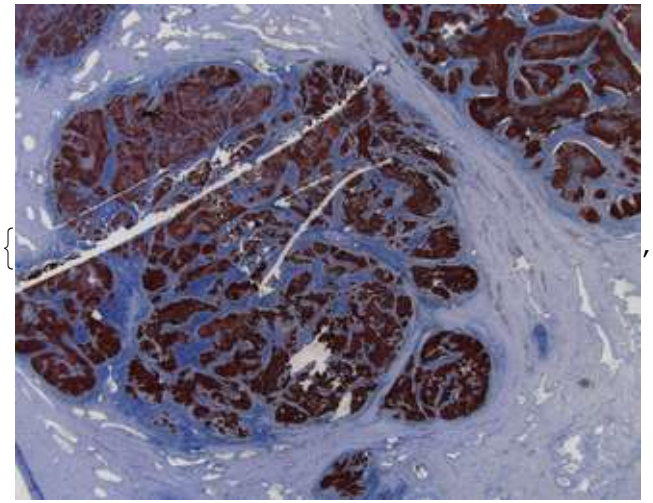
{  ,

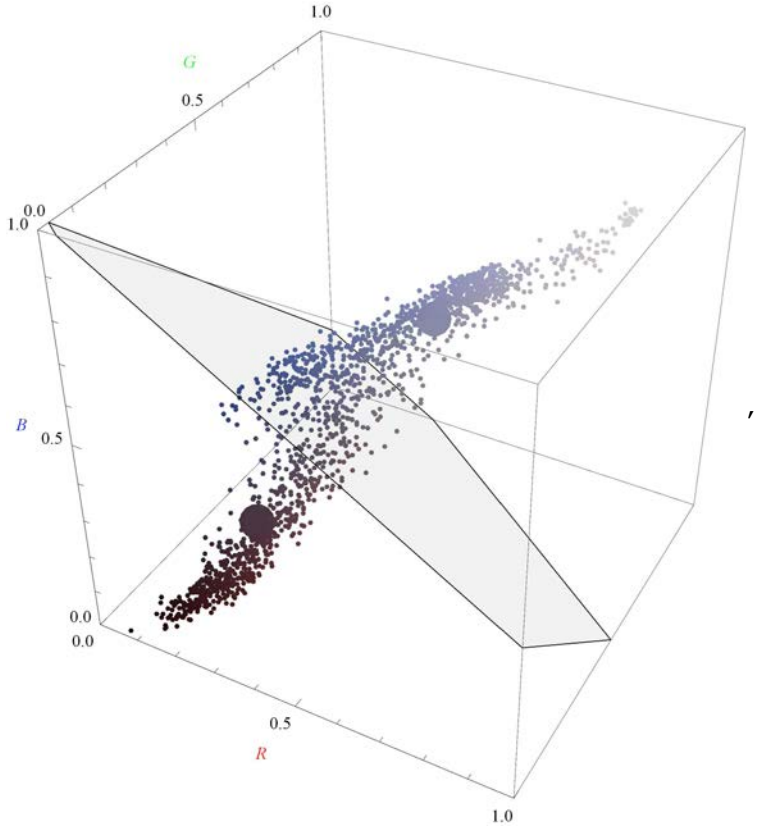

1.0
G
0.5
0.0
1.0
B
0.5
0.0
0.0
0.5
R
1.0

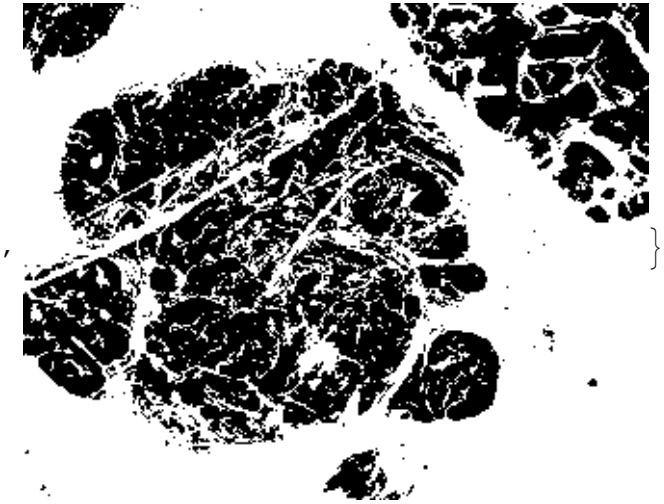
,  }

```
{Show[#1],
   Show[{Graphics3D[({RGBColor[#], PointSize[.0075], Point[#]}) &/@Flatten[ImageData[#1], 1][[ ;; ;; 50]],
        PlotRange → {{0, 1}, {0, 1}, {0, 1}}, AxesLabel → {Text[Style["R", Red, Italic]], Text[Style["G", Green, Italic]],
          Text[Style["B", Blue, Italic]]}, Axes → True, RotationAction → "Clip", ImageSize → 384],
       Graphics3D[({RGBColor[#], PointSize[.05], Point[#]}) &/@ (#2 / 255)], MyPlane[#2 / 255]}] &[#1, #2],
   Show[Image[MyBinarizeRGB[ImageData[#1, "Byte"], #2]]]} &[Sequence @@ ({#, MyISODATARGB[ImageData[#, "Real"] * 255]}) &[
  ImageResize[p16bild, Scaled[1 / 4]]]]
```

```
{{N.A., N.A.}, {{109.698, 102.993, 124.499}, {109.898, 103.193, 124.699}}}
{{40 577, 43 273}, {{76.0578, 59.3715, 75.2426}, {141.436, 144.09, 170.881}}}
{{38 811, 45 039}, {{75.3641, 57.5283, 72.2534}, {139.47, 142.357, 169.707}}}
{{37 636, 46 214}, {{74.8271, 56.2599, 70.3016}, {138.278, 141.233, 168.818}}}
{{36 793, 47 057}, {{74.4579, 55.3435, 68.8555}, {137.43, 140.427, 168.184}}}
{{36 226, 47 624}, {{74.2086, 54.7166, 67.8719}, {136.87, 139.891, 167.75}}}
{{35 832, 48 018}, {{74.0407, 54.2815, 67.1751}, {136.481, 139.517, 167.45}}}
{{35 557, 48 293}, {{73.92, 53.972, 66.6907}, {136.214, 139.259, 167.236}}}
{{35 371, 48 479}, {{73.8376, 53.7625, 66.3614}, {136.035, 139.085, 167.09}}}
{{35 263, 48 587}, {{73.7958, 53.6416, 66.1651}, {135.927, 138.983, 167.009}}}
{{35 197, 48 653}, {{73.7729, 53.5688, 66.0424}, {135.86, 138.92, 166.961}}}
{{35 142, 48 708}, {{73.7519, 53.5071, 65.9419}, {135.805, 138.868, 166.92}}}
{{35 111, 48 739}, {{73.7398, 53.4732, 65.8847}, {135.774, 138.838, 166.897}}}
{{35 094, 48 756}, {{73.7329, 53.4536, 65.8543}, {135.757, 138.823, 166.883}}}
{{35 079, 48 771}, {{73.7266, 53.4365, 65.8275}, {135.743, 138.809, 166.871}}}
{{35 069, 48 781}, {{73.7217, 53.4252, 65.8099}, {135.734, 138.799, 166.863}}}
{{35 057, 48 793}, {{73.7172, 53.4116, 65.788}, {135.721, 138.788, 166.854}}}
{{35 049, 48 801}, {{73.7135, 53.4021, 65.7742}, {135.714, 138.781, 166.848}}}
{{35 039, 48 811}, {{73.7123, 53.3914, 65.7538}, {135.702, 138.771, 166.842}}}
{{35 036, 48 814}, {{73.7115, 53.388, 65.7481}, {135.699, 138.768, 166.839}}}
{{35 034, 48 816}, {{73.7103, 53.3857, 65.7448}, {135.697, 138.766, 166.838}}}
{{35 034, 48 816}, {{73.7103, 53.3857, 65.7448}, {135.697, 138.766, 166.838}}}
```

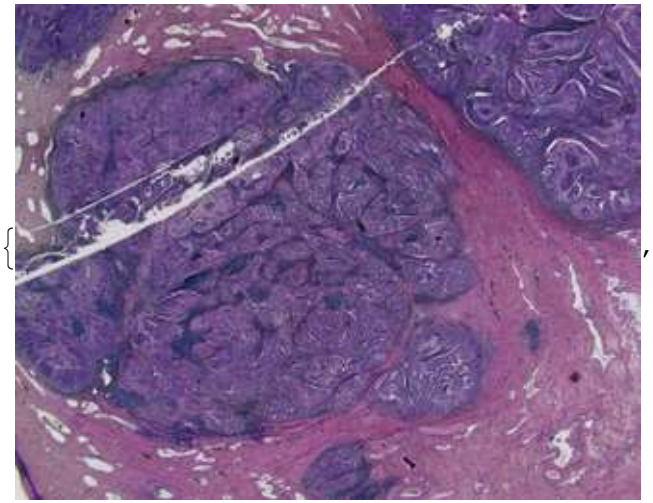
{  ,

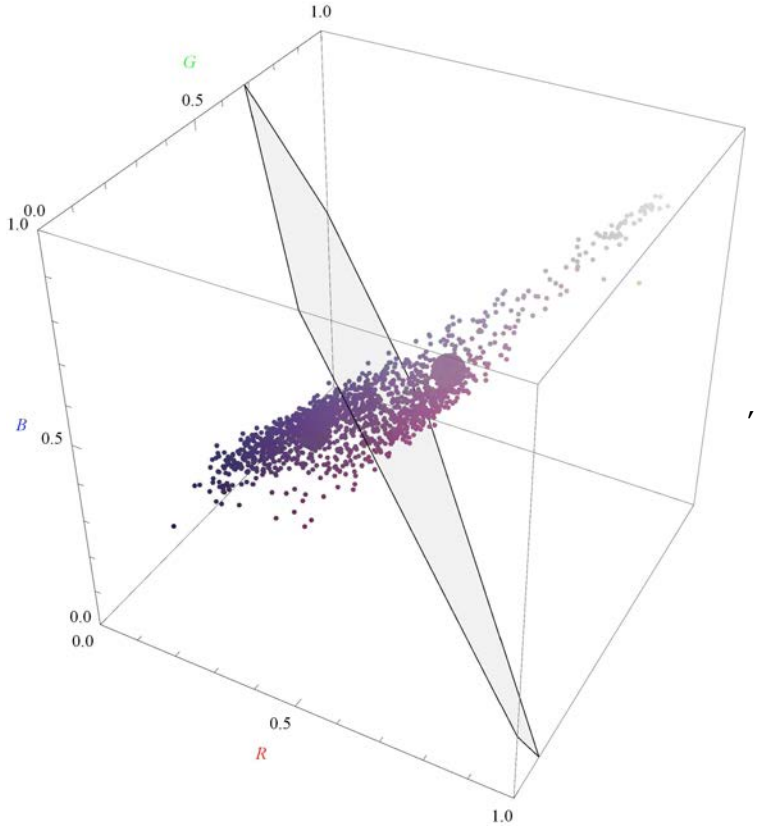

1.0
G
0.5
0.0
1.0
B
0.5
0.0
0.0
0.5
R
1.0

 }

```
{Show[#1],
   Show[{Graphics3D[({RGBColor[#], PointSize[.0075], Point[#]}) & /@ Flatten[ImageData[#1], 1][[ ;; ;; 50]],
        PlotRange → {{0, 1}, {0, 1}, {0, 1}}, AxesLabel → {Text[Style["R", Red, Italic]], Text[Style["G", Green, Italic]],
          Text[Style["B", Blue, Italic]]}, Axes → True, RotationAction → "Clip", ImageSize → 384],
       Graphics3D[({RGBColor[#], PointSize[.05], Point[#]}) & /@ (#2 / 255)], MyPlane[#2 / 255]}] &[#1, #2],
   Show[Image[MyBinarizeRGB[ImageData[#1, "Byte"], #2]]]} &[Sequence @@ ({#, MyISODATARGB[ImageData[#, "Real"] * 255]}) &[
  ImageResize[hebild, Scaled[1 / 4]]]]
```

```
{{N.A., N.A.}, {{119.681, 89.0555, 133.941}, {119.881, 89.2555, 134.141}}}

{{46 423, 37 427}, {{95.7714, 70.063, 120.887}, {149.561, 112.837, 150.357}}}

{{49 157, 34 693}, {{96.9991, 71.1482, 121.938}, {152.061, 114.67, 151.191}}}

{{50 778, 33 072}, {{97.9088, 71.732, 122.405}, {153.363, 115.907, 151.907}}}

{{51 790, 32 060}, {{98.499, 72.0893, 122.674}, {154.16, 116.724, 152.403}}}

{{52 444, 31 406}, {{98.8869, 72.3137, 122.848}, {154.671, 117.279, 152.732}}}

{{52 836, 31 014}, {{99.1204, 72.4475, 122.952}, {154.978, 117.62, 152.932}}}

{{53 073, 30 777}, {{99.2629, 72.5285, 123.014}, {155.163, 117.828, 153.057}}}

{{53 224, 30 626}, {{99.3537, 72.5798, 123.054}, {155.281, 117.962, 153.136}}}

{{53 293, 30 557}, {{99.3943, 72.6043, 123.072}, {155.336, 118.022, 153.172}}}

{{53 337, 30 513}, {{99.4212, 72.62, 123.082}, {155.37, 118.06, 153.198}}}

{{53 371, 30 479}, {{99.441, 72.6321, 123.091}, {155.398, 118.089, 153.215}}}

{{53 393, 30 457}, {{99.4532, 72.6405, 123.098}, {155.417, 118.107, 153.225}}}

{{53 404, 30 446}, {{99.4601, 72.6441, 123.1}, {155.425, 118.117, 153.232}}}

{{53 409, 30 441}, {{99.4626, 72.6461, 123.102}, {155.429, 118.121, 153.234}}}

{{53 414, 30 436}, {{99.4653, 72.648, 123.104}, {155.434, 118.125, 153.236}}}

{{53 414, 30 436}, {{99.4653, 72.648, 123.104}, {155.434, 118.125, 153.236}}}
```

{  ,


1.0
G
0.5
0.0
1.0
B
0.5
0.0
0.0
0.5
R
1.0

,  }