7. Vorlesung Mikroskopische Bildverarbeitung

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

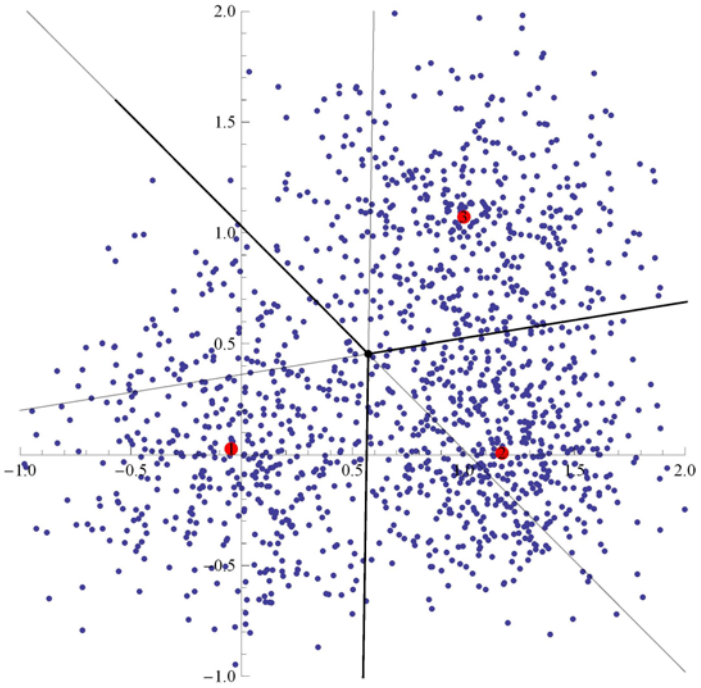
Initialisierung

29. Januar 2016

Aus 20. Segmentierung mit dem k-Means-Verfahren

k-Means Segmentation Algorithm (Steinhaus 1957, MacQueen 1967)

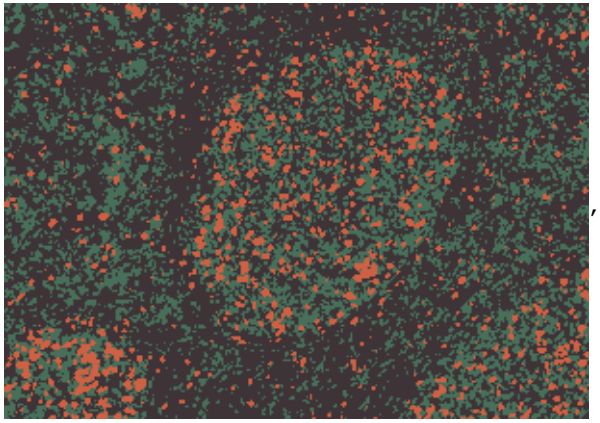
```
SeedRandom[2405];
(*Generierung von multimodaler Verteilung 2D*)
data = Join[RandomReal[NormalDistribution[0, .4], {500, 2}], RandomReal[NormalDistribution[1, .4], {500, 2}],
   Transpose[{RandomReal[NormalDistribution[1.2, .3], 500], RandomReal[NormalDistribution[0, .3], 500]}]];
(*Vorgabe: 3 Klassen, und Einsatz auf Level 1 (vektorielle Daten)*)
cs = ClusteringComponents[data, 3, 1, Method → "KMeans"];
km = Mean[Map[data[[Sequence @@ #]] &, #]] & /@ (Position[cs, #] & /@ Union[Flatten[cs]]);
Show[{ListPlot[data, AspectRatio → 1, PlotRange → {{-1, 2}, {-1, 2}}],
  ListPlot[km, PlotStyle -> Directive[Red, PointSize[Large]]], DiagramPlot[km],
  MyLine[#] & /@ (km[[#]] & /@ Flatten[Table[{i, j}, {i, Length[km] - 1}, {j, i + 1, Length[km]}], 1])
 }]
```



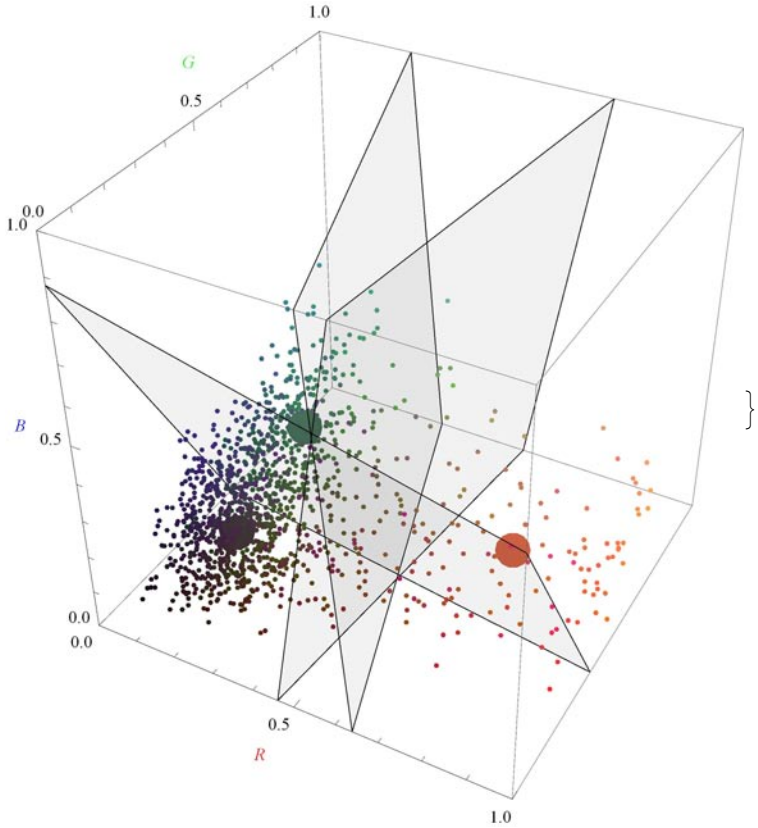

```
Clear[kmeans];
kmeans[bild_Image, klassen_Integer, ebenen_: True] :=
 Module[{c, i, m},
  c = ClusteringComponents[bild, klassen, Method → "KMeans"];
  i = ImageData[bild];
  m = Mean[Map[i[[Sequence @@ #]] &, #]] & /@ (Position[c, #] & /@ Union[Flatten[c]]);
  If[ebenen == True,
   Print[{MatrixForm[m], Show[Colorize[c, ColorFunction → (RGBColor[Sequence @@ m[[#]]] &), ColorFunctionScaling → False]],
      Show[{Graphics3D[({RGBColor[#], PointSize[.0075], Point[#]}) & /@ Flatten[ImageData[bild], 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[#]}) & /@ m],
        MyPlane[#] & /@ (m[[#]] & /@ Flatten[Table[{i, j}, {i, Length[m] - 1}, {j, i + 1, Length[m]}], 1])}]}];
   ,
   Print[{MatrixForm[m], Show[Colorize[c, ColorFunction → (RGBColor[Sequence @@ m[[#]]] &), ColorFunctionScaling → False]],
      Show[{Graphics3D[({RGBColor[#], PointSize[.0075], Point[#]}) & /@ Flatten[ImageData[bild], 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[#]}) & /@ m]}]}];
  ];
  m
 ]
```

```
kmeans[ImageResize[Lym2CD21dreikanalausgleichF1, Scaled[1 / 5]], 3];
```

$$\left\{\begin{pmatrix} 0.276764 & 0.435005 & 0.35404 \\ 0.243732 & 0.202069 & 0.215306 \\ 0.809613 & 0.379581 & 0.266781 \end{pmatrix},\right.$$



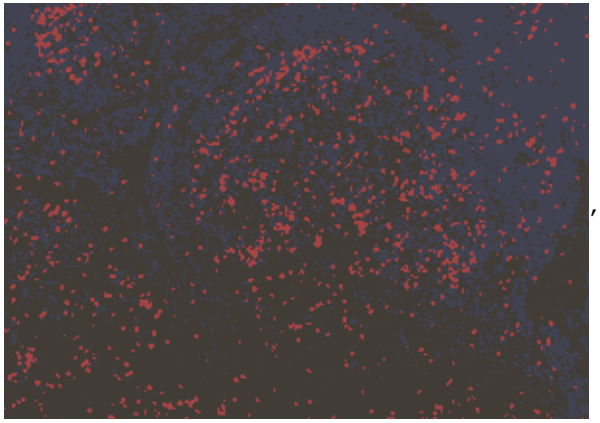
,



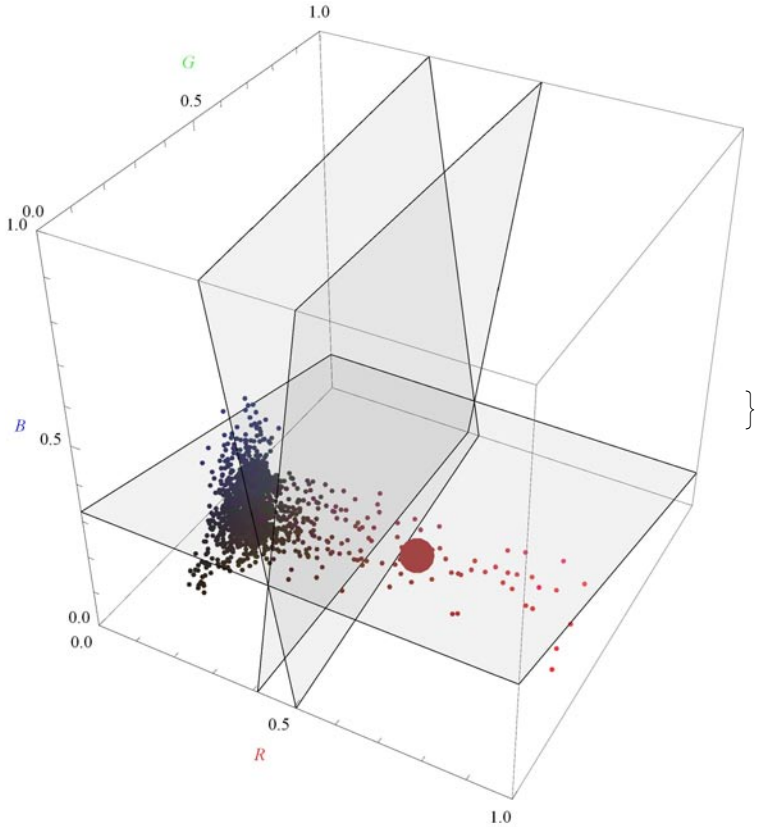
$\left.\right\}$

```
kmeans[ImageResize[Lym3CD21dreikanalausgleichF2, Scaled[1 / 5]], 3];
```

$$\left\{\begin{pmatrix} 0.255548 & 0.261434 & 0.327279 \\ 0.254895 & 0.236316 & 0.213171 \\ 0.646238 & 0.265679 & 0.272093 \end{pmatrix},\right.$$



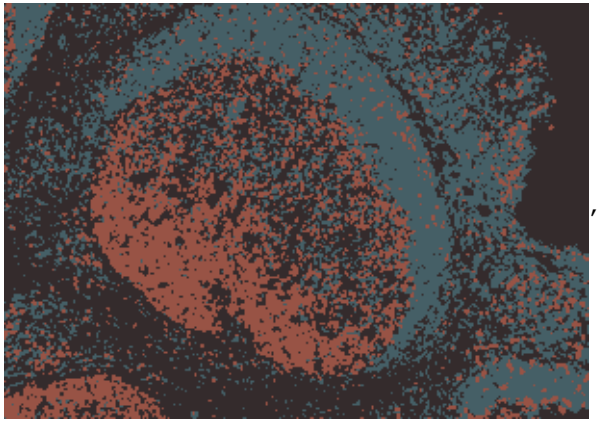
,



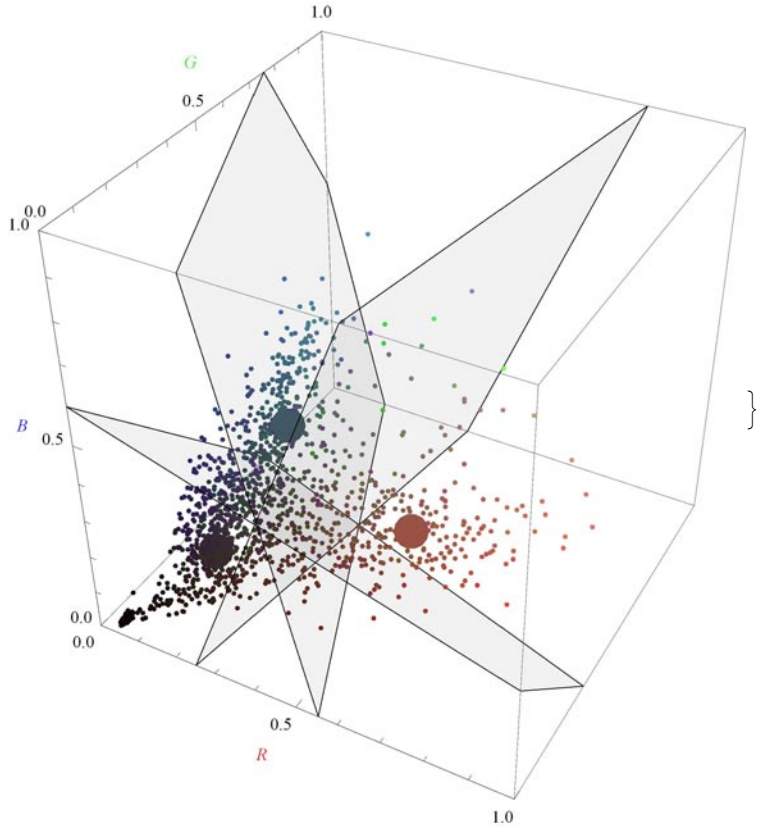
$$\left.\right\}$$

```
kmeans[ImageResize[Ton2CD21dreikanalausgleichF1, Scaled[1 / 5]], 3];
```

$$\left\{\begin{pmatrix} 0.203934 & 0.167473 & 0.172969 \\ 0.596076 & 0.325165 & 0.271018 \\ 0.270848 & 0.371078 & 0.399614 \end{pmatrix},\right.$$



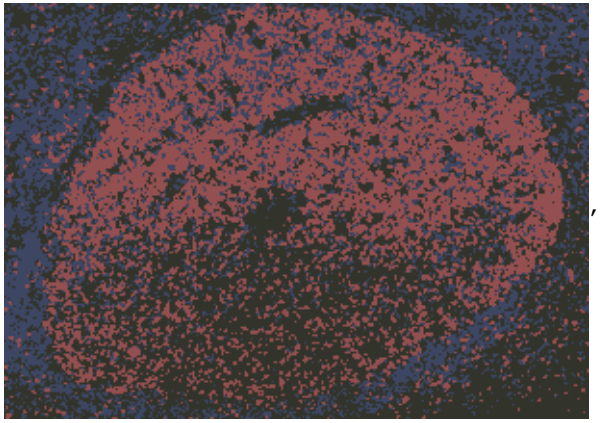
,



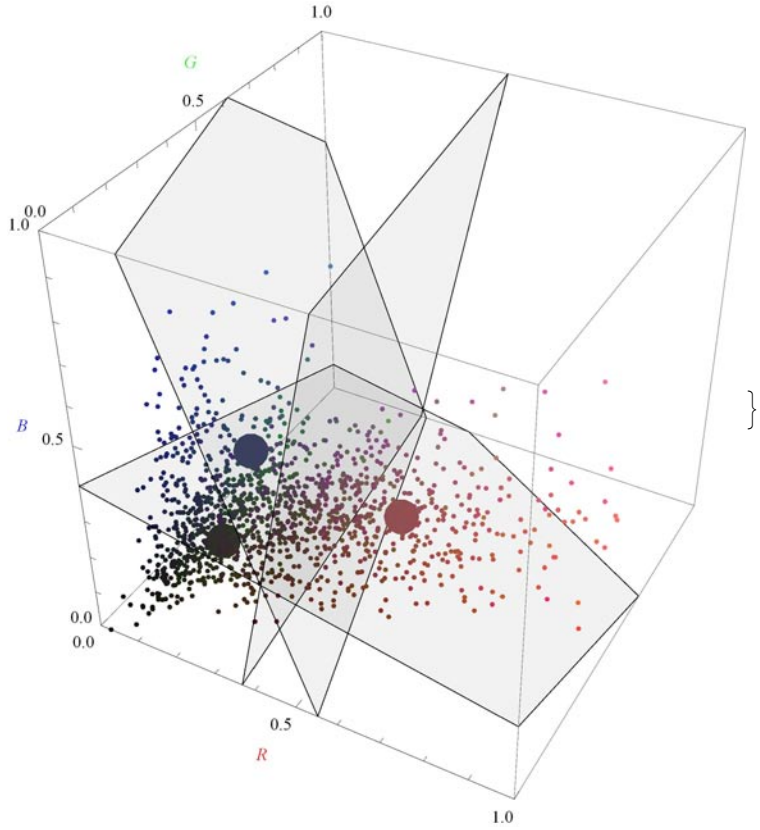
$$\left.\right\}$$

```
kmeans[ImageResize[Ton3CD21dreikanalausgleichF2, Scaled[1 / 5]], 3];
```

$$\left\{\begin{pmatrix} 0.197916 & 0.205108 & 0.169184 \\ 0.238585 & 0.274407 & 0.385438 \\ 0.580251 & 0.311727 & 0.314871 \end{pmatrix},\right.$$



,



$$\left.\right\}$$

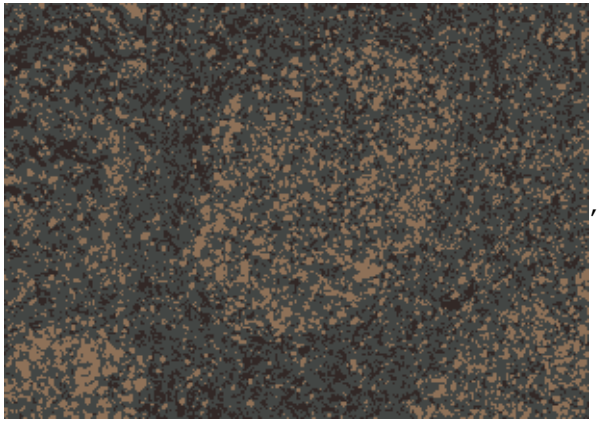
Aus 21. Segmentierung durch hierarchisch aufteilende Partitionierung

Hierarchisch aufteilende Partitionierung

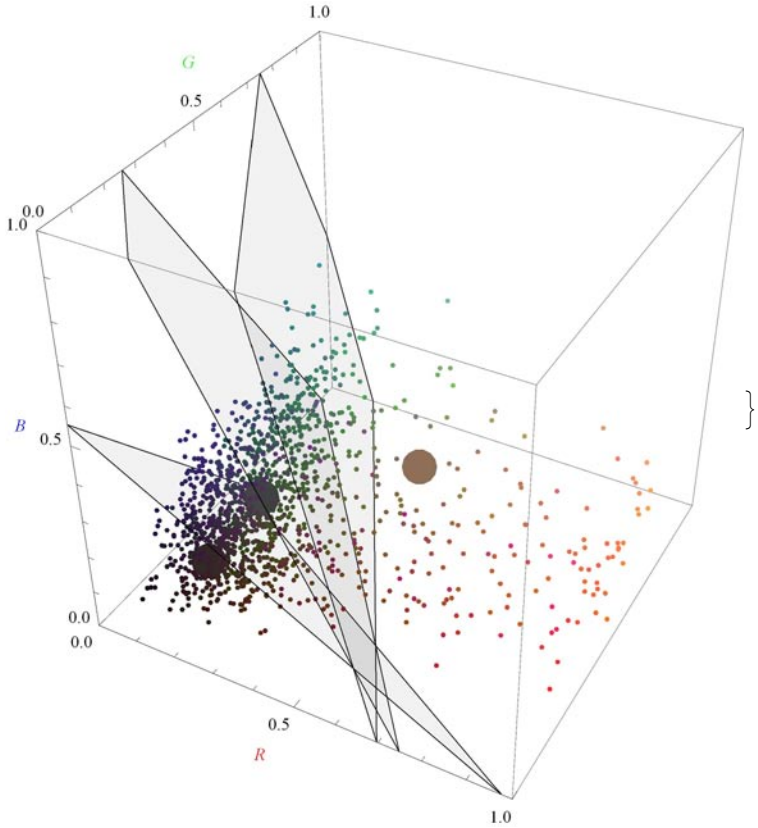
```
Clear[colquant];
colquant[bild_Image, klassen_Integer, ebenen_: True] :=
 Module[{cq, m},
  cq = ColorQuantize[bild, klassen, Dithering → False];
  m = Union[Flatten[ImageData[cq], 1]];
  If[ebenen == True,
   Print[{MatrixForm[m], Show[cq],
      Show[{Graphics3D[({RGBColor[#], PointSize[.0075], Point[#]}) & /@ Flatten[ImageData[bild], 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[#]}) & /@m],
        MyPlane[#] & /@ (m[[#]] & /@ Flatten[Table[{i, j}, {i, Length[m] - 1}, {j, i + 1, Length[m]}], 1])}]}];
   ,
   Print[{MatrixForm[m], Show[cq],
      Show[{Graphics3D[({RGBColor[#], PointSize[.0075], Point[#]}) & /@ Flatten[ImageData[bild], 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[#]}) & /@m]}]}];
  ];
  m
 ]
```

```
colquant[ImageResize[Lym2CD21dreikanalausgleichF1, Scaled[1 / 5]], 3];
```

$$\left\{\begin{pmatrix} 0.196078 & 0.156863 & 0.14902 \\ 0.262745 & 0.27451 & 0.266667 \\ 0.556863 & 0.443137 & 0.345098 \end{pmatrix},\right.$$



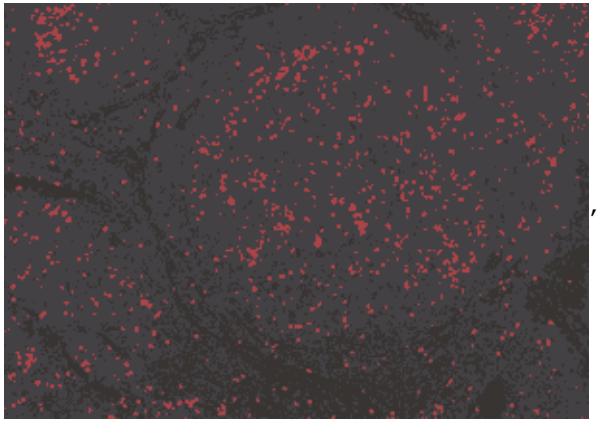
,



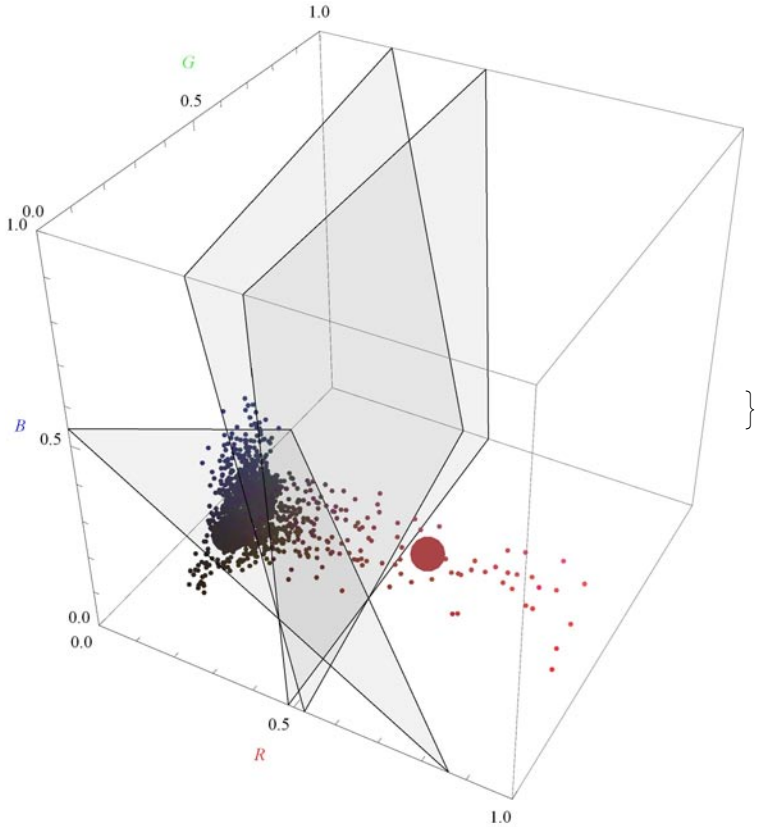
$$\left.\right\}$$

```
colquant[ImageResize[Lym3CD21dreikanalausgleichF2, Scaled[1 / 5]], 3];
```

$$\left\{\begin{pmatrix} 0.219608 & 0.203922 & 0.196078 \\ 0.266667 & 0.254902 & 0.270588 \\ 0.670588 & 0.266667 & 0.286275 \end{pmatrix},\right.$$



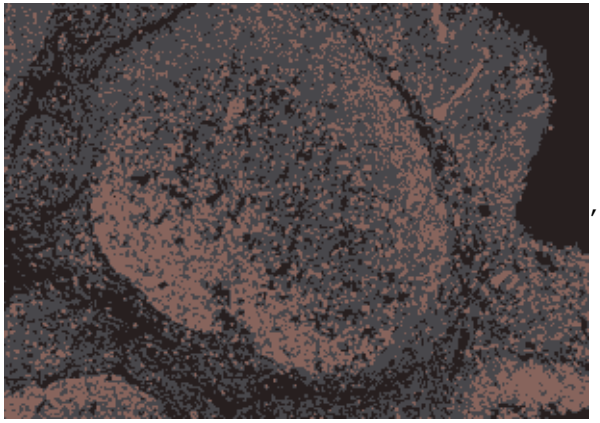
,



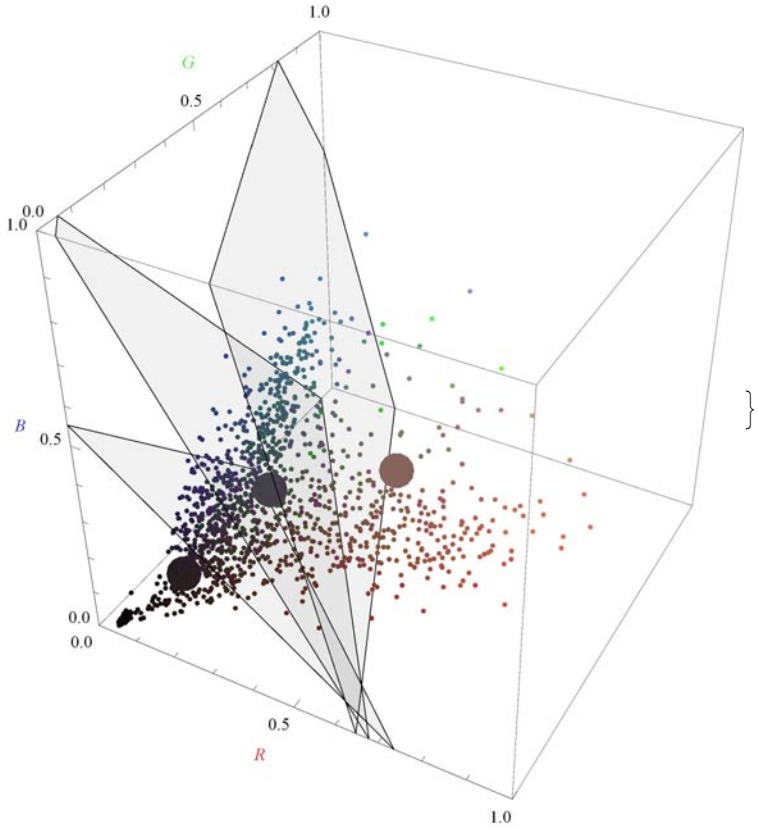
$$\left.\right\}$$

```
colquant[ImageResize[Ton2CD21dreikanalausgleichF1, Scaled[1 / 5]], 3];
```

$$\left\{\begin{pmatrix} 0.152941 & 0.121569 & 0.121569 \\ 0.282353 & 0.278431 & 0.294118 \\ 0.529412 & 0.392157 & 0.360784 \end{pmatrix},\right.$$



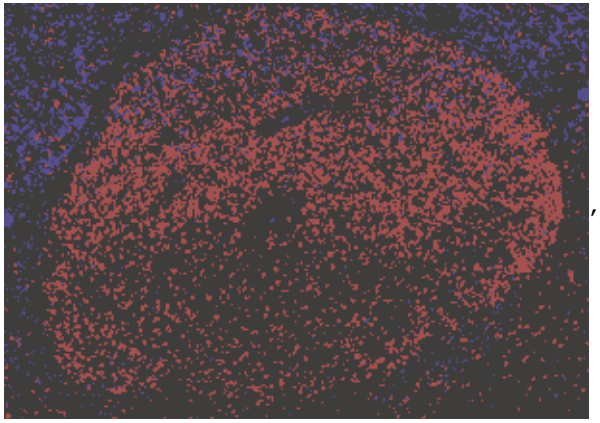
,



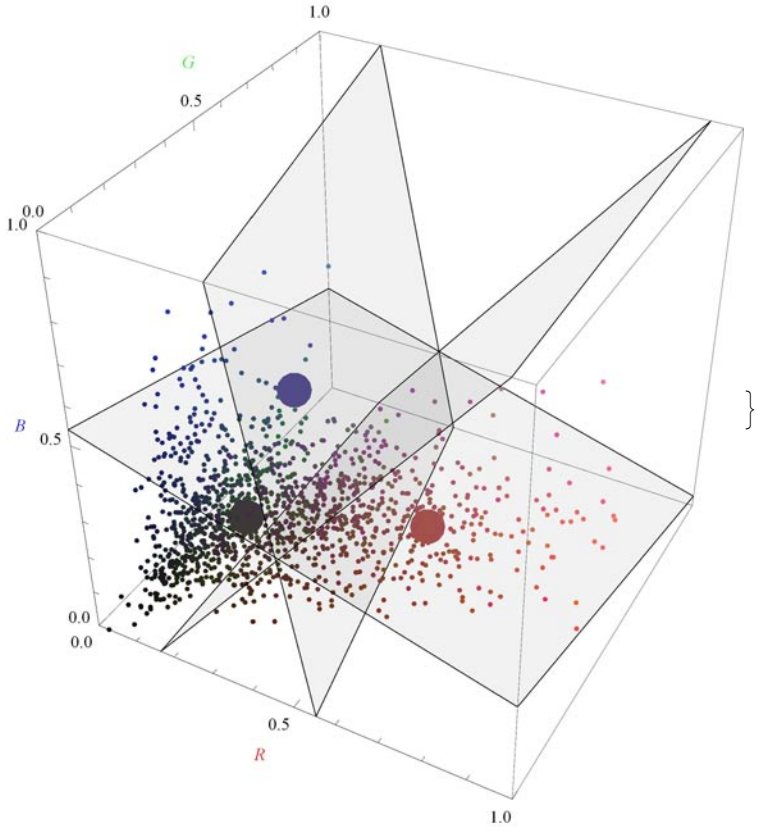
$$\left.\right\}$$

```
colquant[ImageResize[Ton3CD21dreikanalausgleichF2, Scaled[1 / 5]], 3];
```

$$\left\{\begin{pmatrix} 0.247059 & 0.235294 & 0.235294 \\ 0.337255 & 0.309804 & 0.556863 \\ 0.643137 & 0.317647 & 0.309804 \end{pmatrix},\right.$$



,



$$\left.\right\}$$

Aus 22. Segmentierung durch “Mittelwertverschiebung”

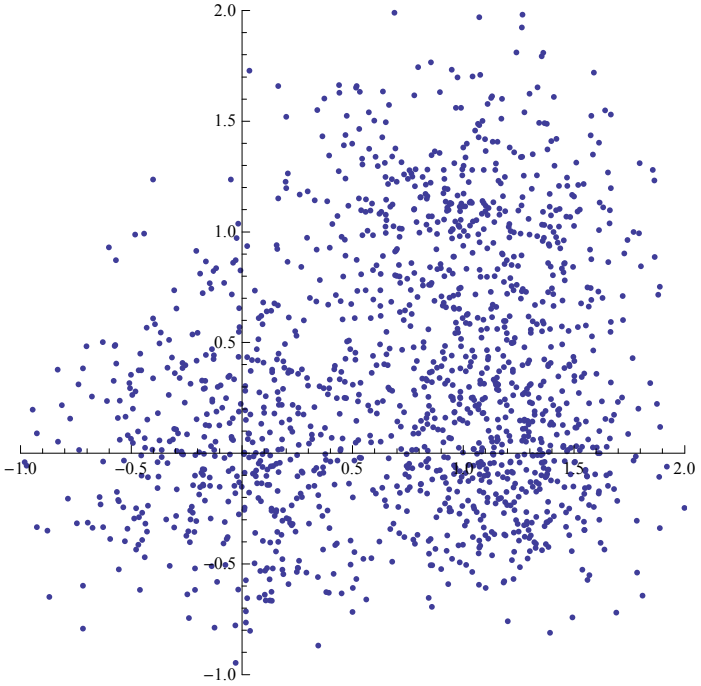
Mean Shift Segmentation Algorithm (Fukunaga und Hostetler 1975)

```
MeanShiftFilter[ {0, 1, 10, 11}, 1, 1.0, MaxIterations → 1]
```

```
{0.5, 0.5, 10.5, 10.5}
```

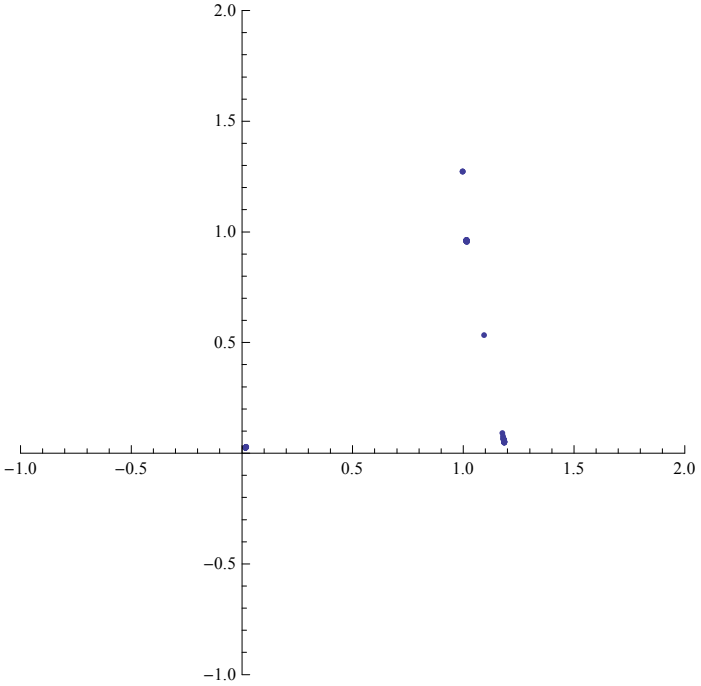
```
(*Auswahlfunktion zur Selection nur bis zu einem bestimmten Betrag abweichender Elemente*)
Clear[MySelect];
MySelect[in_, ref_, d_] := Select[in, EuclideanDistance[#, ref] ≤ d &];

(*Eigentliche Implementierung, es wird eine zyklisch geschlossene Bildfläche (Torus) verarbeitet*)
Clear[MyMeanShiftFilter];
MyMeanShiftFilter[im_, el_, dist_] := Module[{flatelpos, zentralpos},
   (*lineare Positionsindizes für alle Elemente in lokaler Nachbarschaft entsprechend des Strukturelements*)
   flatelpos = Flatten[Position[Flatten[el], 1]];
   (*Index des zentralen Elements innerhalb der Nachbarschaft*)
   zentralpos = Ceiling[Dimensions[Flatten[el]] / 2];
   (*MySelect hat drei Argumente: alle Elemente entspr. Strukturelement,
   Wert des zentralen Pixels als Bezug, maximal zulässiger Werteradius*)
   Developer`PartitionMap[Mean[MySelect[Flatten[#, 1][[flatelpos]], First[Flatten[#, 1][[zentralpos]]], dist]] &,
    N[im], Dimensions[el], 1, Ceiling[Dimensions[el] / 2]]
  ];

(*Test mit selben Daten wie oben*)
MyMeanShiftFilter[{0, 1, 10, 11}, {1, 1, 1}, 1]
{0.5, 0.5, 10.5, 10.5}

SeedRandom[2405];
data = Join[RandomReal[NormalDistribution[0, .4], {500, 2}], RandomReal[NormalDistribution[1, .4], {500, 2}],
   Transpose[{RandomReal[NormalDistribution[1.2, .3], 500], RandomReal[NormalDistribution[0, .3], 500]}]];

data // Dimensions
{1500, 2}
```

```
ListPlot[data, AspectRatio → 1, PlotRange → {{-1, 2}, {-1, 2}}]
```

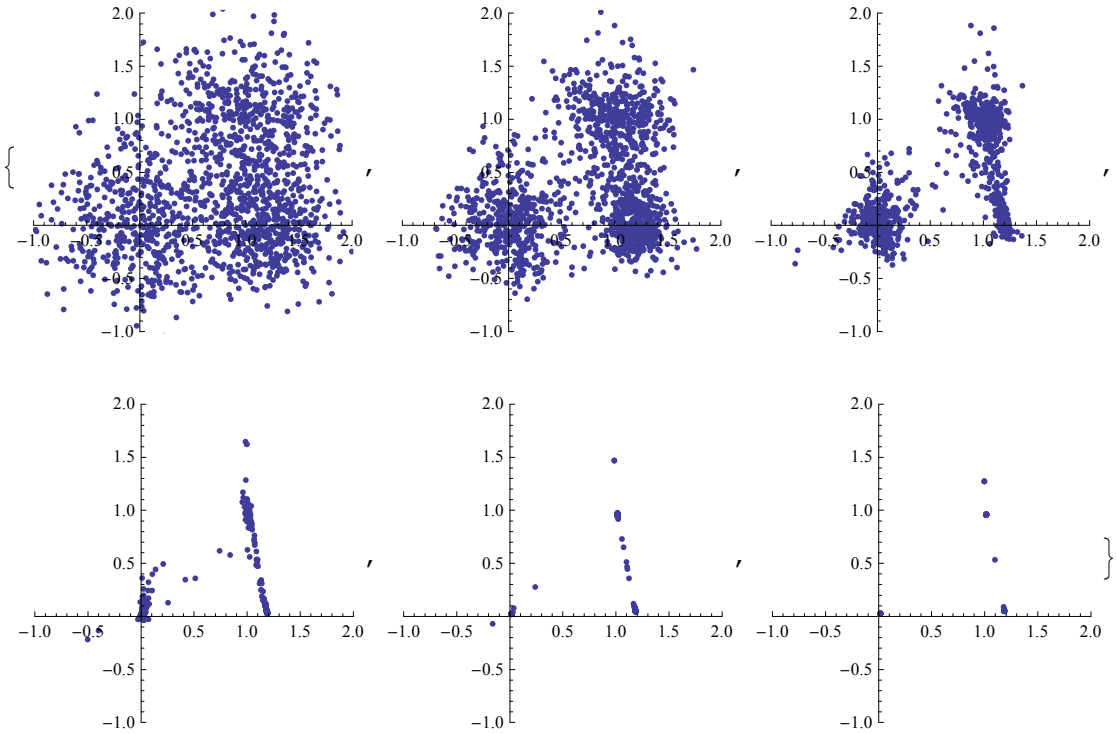



```
ms = NestList[MeanShiftFilter[#, 500, {0.55, 0.5}] &, data, 5];
```

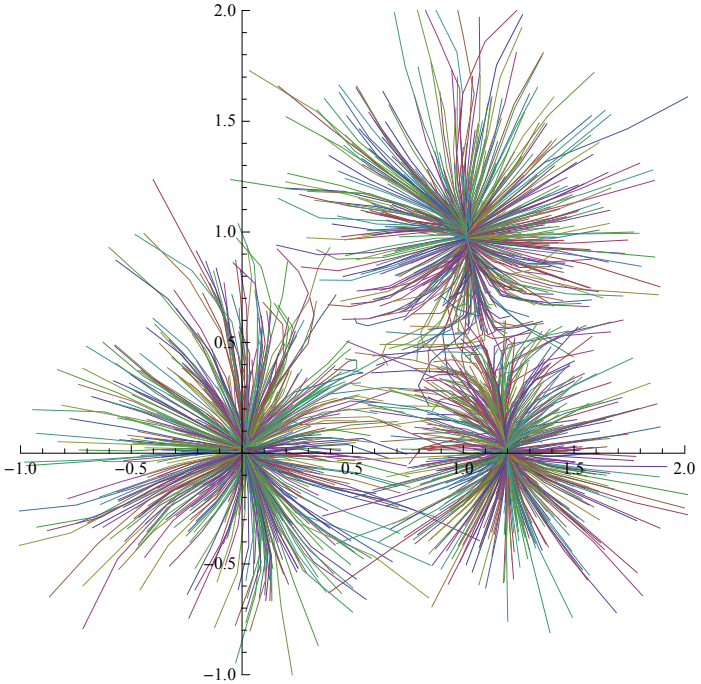
```
ListPlot[Last[ms], AspectRatio → 1, PlotRange → {{-1, 2}, {-1, 2}}]
```



```
ListPlot[#, AspectRatio → 1, PlotRange → {{-1, 2}, {-1, 2}}] & /@ ms
```

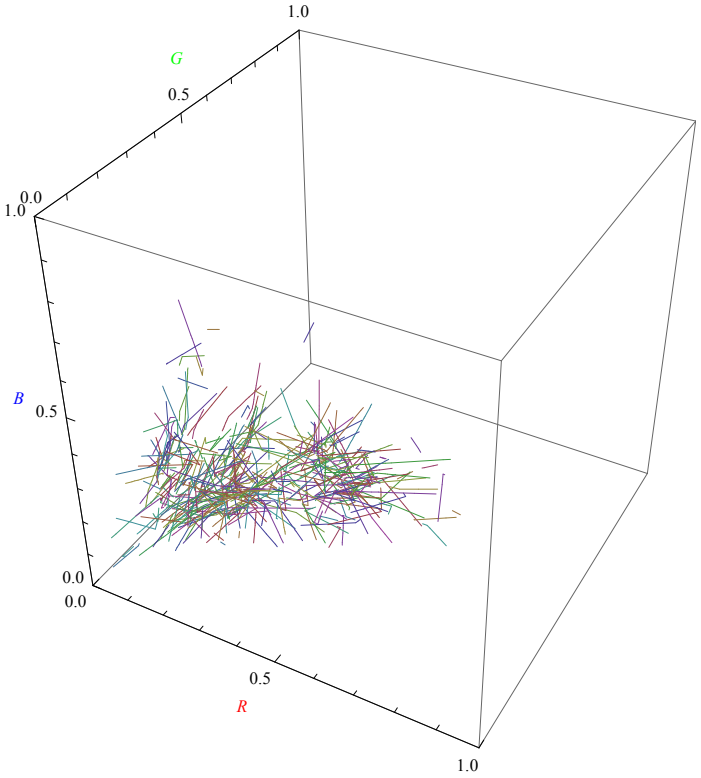


```
ListLinePlot[Transpose[ms], AspectRatio -> 1, PlotRange → {{-1, 2}, {-1, 2}}]
```

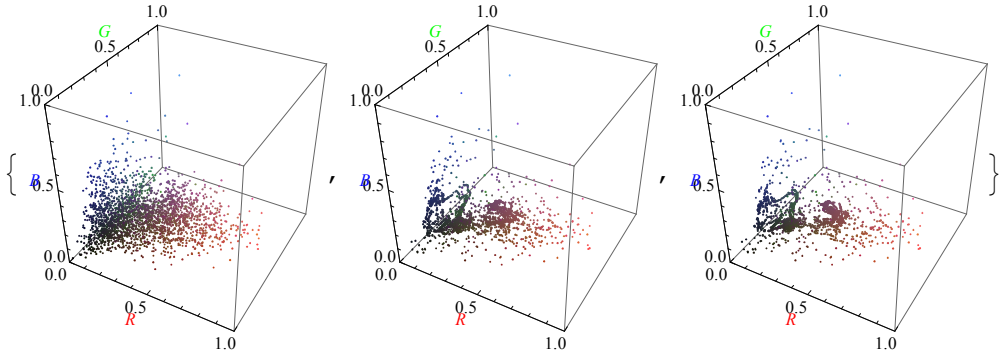



```
mstestbild = NestList[MeanShiftFilter[#, 3, 0.05, MaxIterations → 10] &, ImageResize[Ton3CD21dreikanalausgleichF2, Scaled[1 / 8]], 2];
```

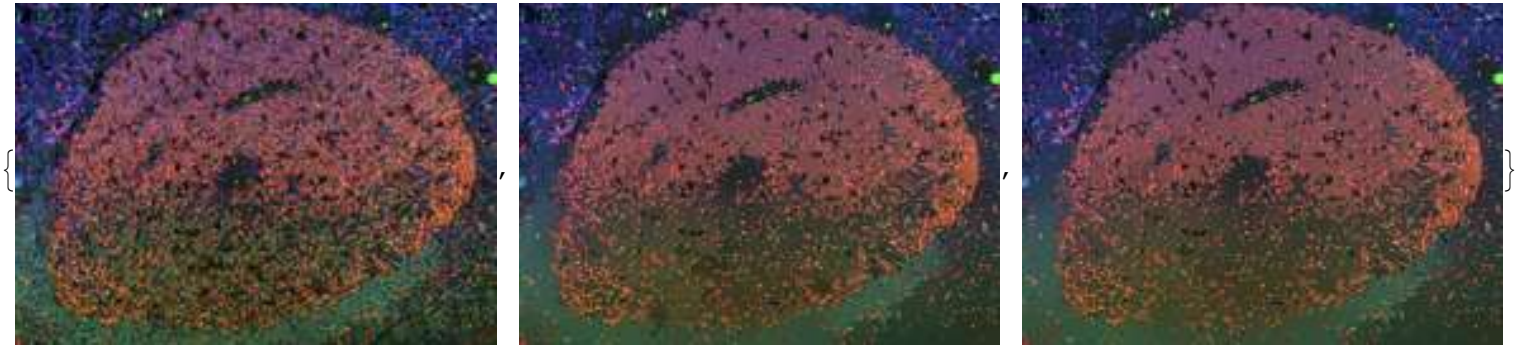
```
Graphics3D[({ColorData[1, RandomInteger[100]], Line[#]}) & /@
  Flatten[Transpose[ImageData[#][[ ;;  ;; 8,  ;;  ;; 8]] & /@mstestbild, {3, 1, 2, 4}], 1], PlotRange → {{0, 1}, {0, 1}, {0, 1}},
 AxesLabel → {Text[Style["R", Red, Italic]], Text[Style["G", Green, Italic]], Text[Style["B", Blue, Italic]]},
 Axes → True, RotationAction → "Clip"]
```



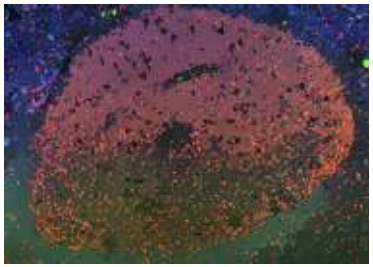

```
Graphics3D[({RGBColor[#], PointSize[.0075], Point[#]}) & /@ Flatten[ImageData[#], 1][[ ;;  ;; 10]], PlotRange → {{0, 1}, {0, 1}, {0, 1}},
   AxesLabel → {Text[Style["R", Red, Italic]], Text[Style["G", Green, Italic]], Text[Style["B", Blue, Italic]]},
   Axes → True, RotationAction → "Clip", ImageSize → 160] & /@mstestbild
```



```
(*Schrittvariation*)
segbildliste = Show[MeanShiftFilter[ ImageResize[Ton3CD21dreikanalausgleichF2, Scaled[1 / 8]], 3, .05, MaxIterations → #], ImageSize → 250] & /@
    (Range[2] * 10);
segbildliste = Prepend[segbildliste, Show[ImageResize[Ton3CD21dreikanalausgleichF2, Scaled[1 / 8]], ImageSize → 250]]
```



```
segbild = MeanShiftFilter[ImageResize[Ton3CD21dreikanalausgleichF2, Scaled[1 / 8]], 3, .05, MaxIterations → 10]
```



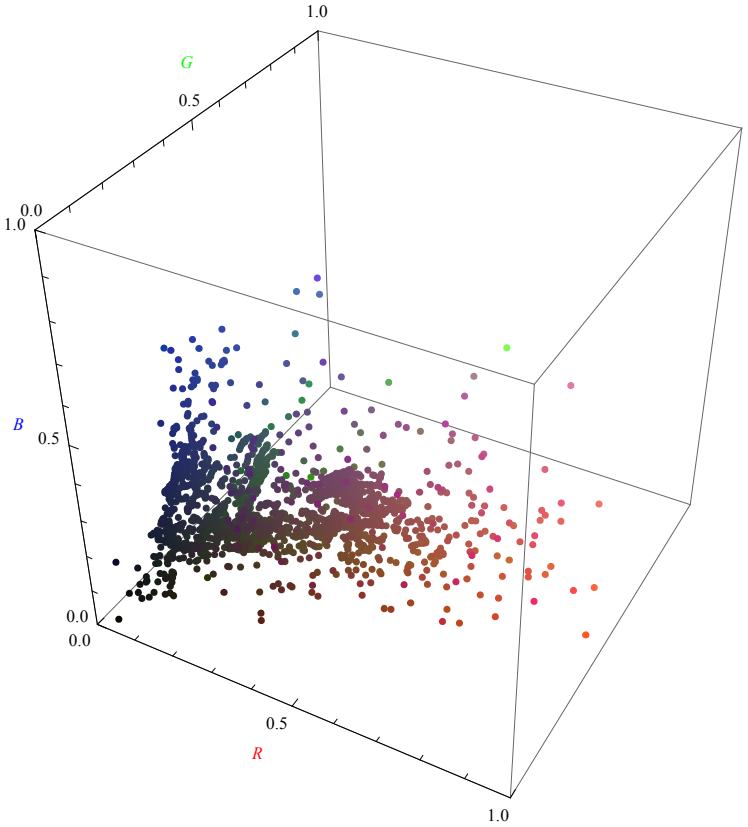
```
Union[Flatten[Round[ImageData[ImageResize[Ton3CD21dreikanalausgleichF2, Scaled[1 / 8]]], 0.1], 1]] // Length
```

```
312
```

```
Union[Flatten[Round[ImageData[segbild], 0.1], 1]] // Length
```

```
285
```

```
Graphics3D[({RGBColor[#], PointSize[.01], Point[#]}) & /@ Flatten[ImageData[segbild][[ ;;  ;; 10]], 1], 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]
```

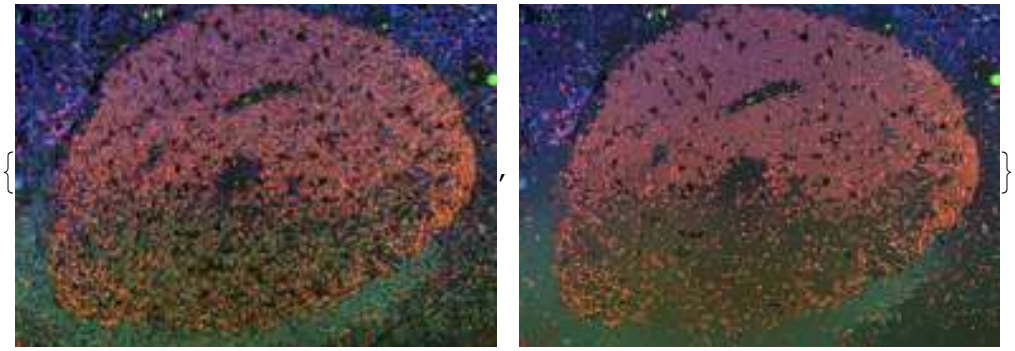


Eigenschaftes des Mean-Shift-Verfahrens:

1. die lokalen Dichtemaxima im Farbraum werden unter Einbeziehung einer lokalen Nachbarschaft bestimmt
2. es erfolgt keine Bestimmung eines Indexbildes mit einer Anzahl vorgegebener Klassen
3. das Verfahren homogenisiert lokal, so daß nachfolgend eingesetzte Verfahren wie k-Means weniger störanfällig für Schattierungen etc. werden
4. das Verfahren ist außerordentlich berechnungsaufwendig

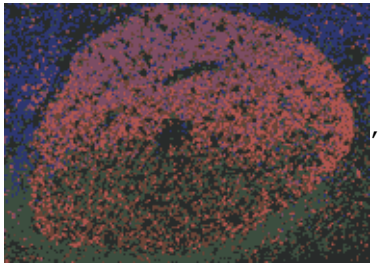
Erweiterung: Kaskadierung von Mean-Shift und k-Means bzw. hierarch. Partitionierung

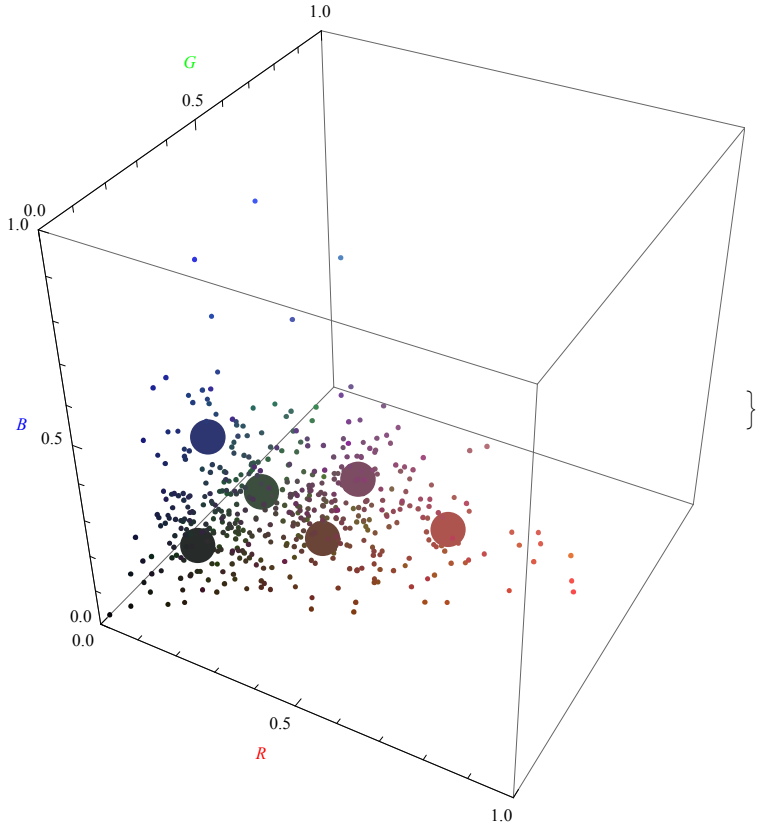
```
Show[#, ImageSize → 250] & /@ {First[mstestbild], Last[mstestbild]}
```



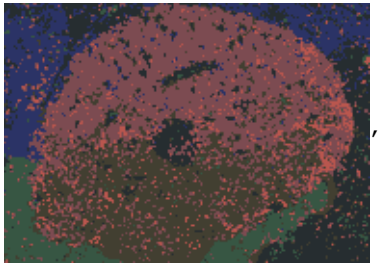
```
kmeans[#, 6, False] & /@ {First[mstestbild], Last[mstestbild]};
```

$$\left\{\begin{pmatrix} 0.155343 & 0.169772 & 0.163164 \\ 0.177938 & 0.211816 & 0.441402 \\ 0.236367 & 0.306961 & 0.245547 \\ 0.487483 & 0.29764 & 0.386917 \\ 0.414598 & 0.26699 & 0.217403 \\ 0.681513 & 0.329525 & 0.305811 \end{pmatrix},\right.$$

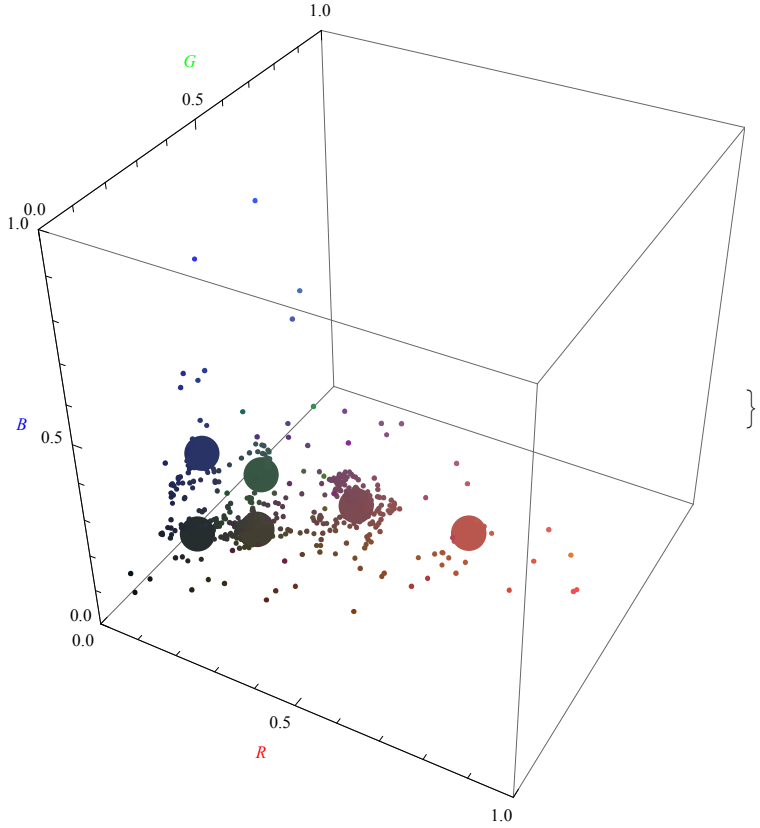




$$\left\{\begin{pmatrix} 0.150586 & 0.178376 & 0.186402 \\ 0.167421 & 0.199868 & 0.401125 \\ 0.260641 & 0.245938 & 0.19353 \\ 0.484659 & 0.292652 & 0.316531 \\ 0.217072 & 0.337592 & 0.263006 \\ 0.72469 & 0.3382 & 0.306157 \end{pmatrix},\right.$$



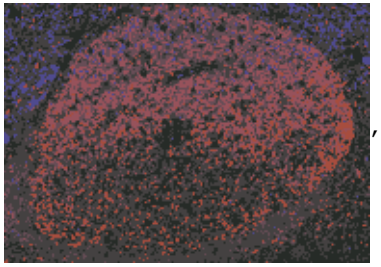
,



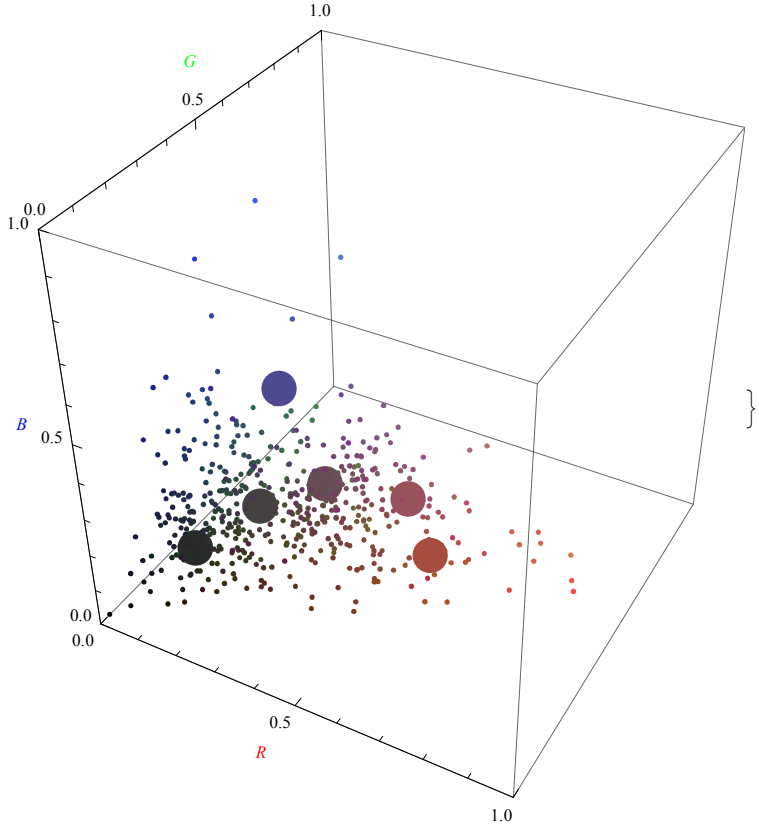
$\left.\right\}$

```
colquant[#, 6, False]; & /@ {First[mstestbild], Last[mstestbild]};
```

$$\left\{\begin{pmatrix} 0.14902 & 0.168627 & 0.152941 \\ 0.266667 & 0.254902 & 0.254902 \\ 0.309804 & 0.290196 & 0.560784 \\ 0.403922 & 0.301961 & 0.337255 \\ 0.592157 & 0.321569 & 0.356863 \\ 0.654902 & 0.301961 & 0.243137 \end{pmatrix},\right.$$

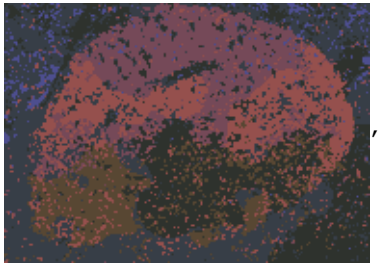


,

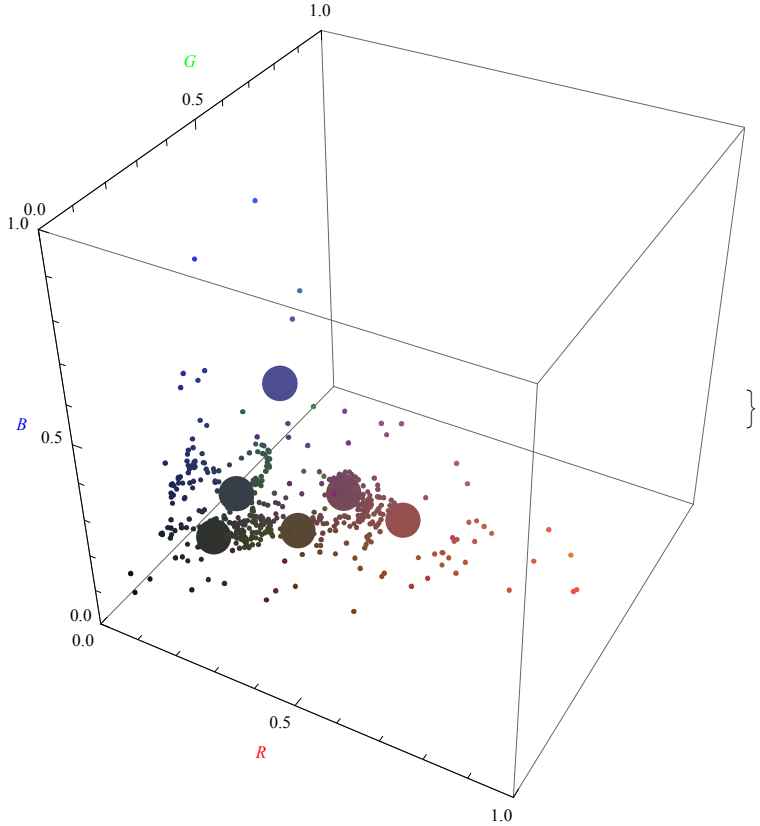


}

$$\left\{\begin{pmatrix} 0.180392 & 0.196078 & 0.176471 \\ 0.215686 & 0.243137 & 0.278431 \\ 0.301961 & 0.305882 & 0.560784 \\ 0.345098 & 0.278431 & 0.2 \\ 0.458824 & 0.286275 & 0.345098 \\ 0.584314 & 0.313725 & 0.301961 \end{pmatrix},\right.$$

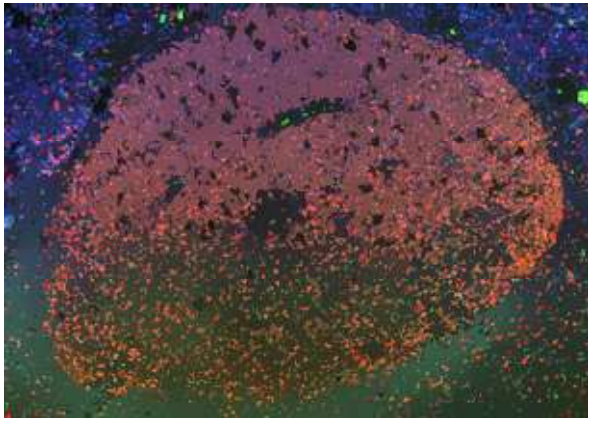


,

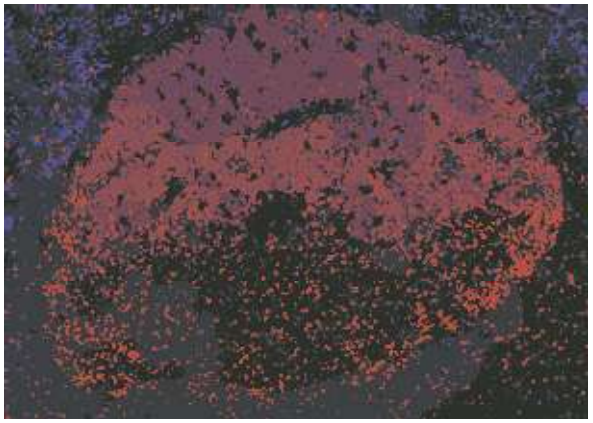


}

```
t = MeanShiftFilter[ImageResize[Ton3CD21dreikanalausgleichF2, Scaled[1 / 5]], 3, .05, MaxIterations → 30]
```



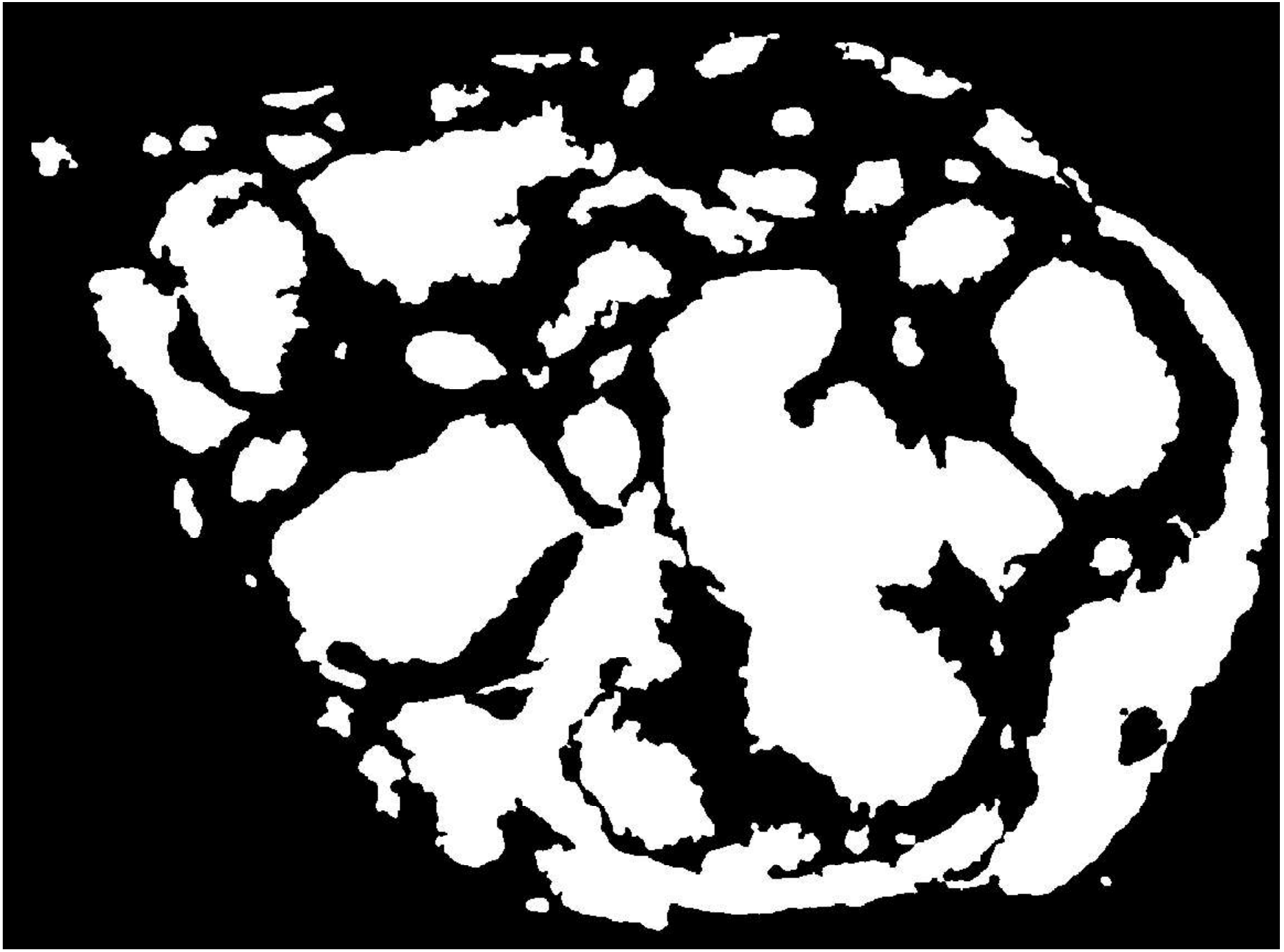
```
(*hierarch. Partitionierung das Mean-Shift-Ergebnisses*)
cq = ColorQuantize[t, 6, Dithering → False]
```



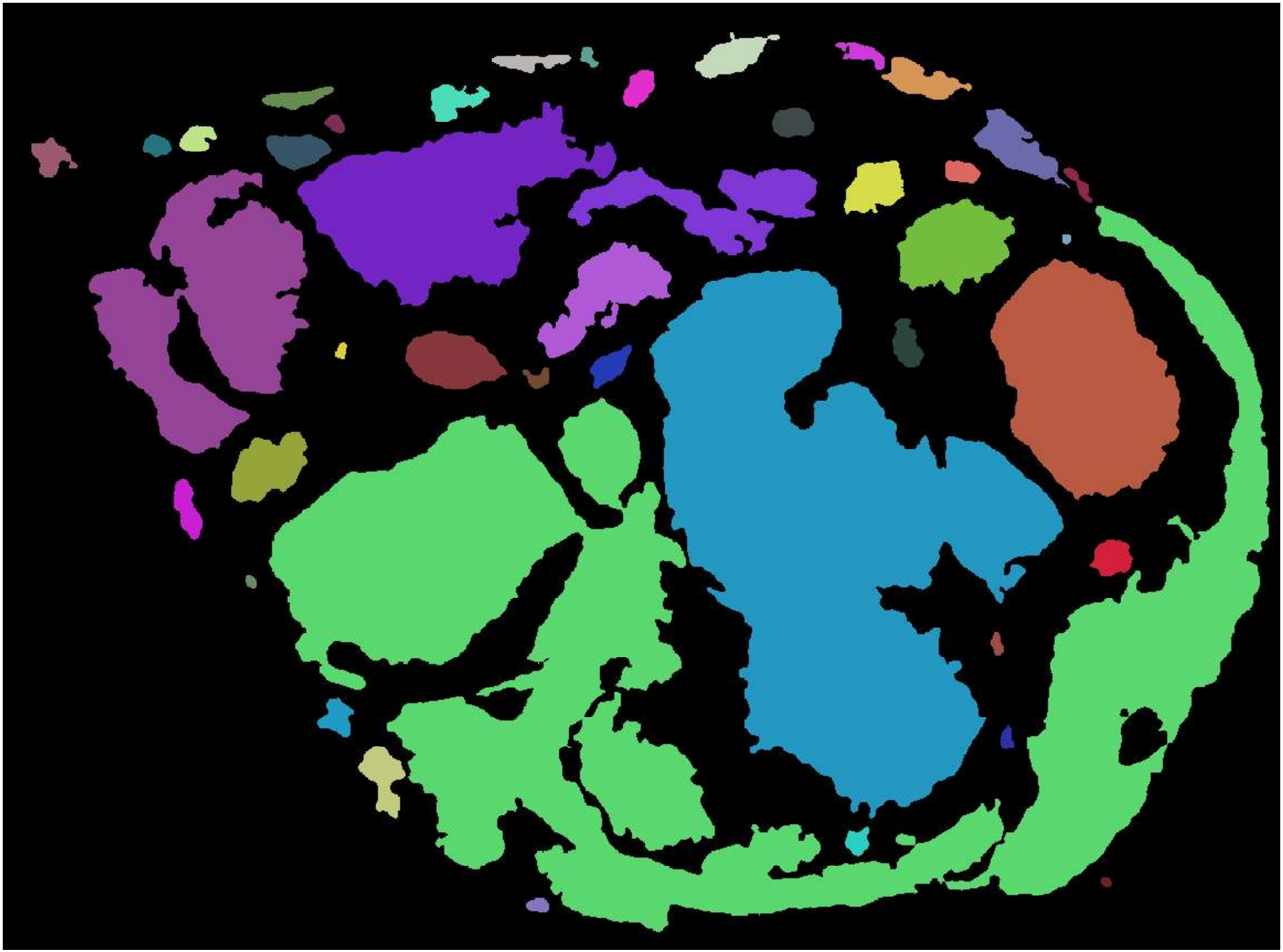
```
(*hierarch. Partitionierung direkt angewendet*)
ColorQuantize[ImageResize[Ton3CD21dreikanalausgleichF2, Scaled[1 / 5]], 6, Dithering → False]
```

Aus 23. Einige Aspekte der Formbeschreibung

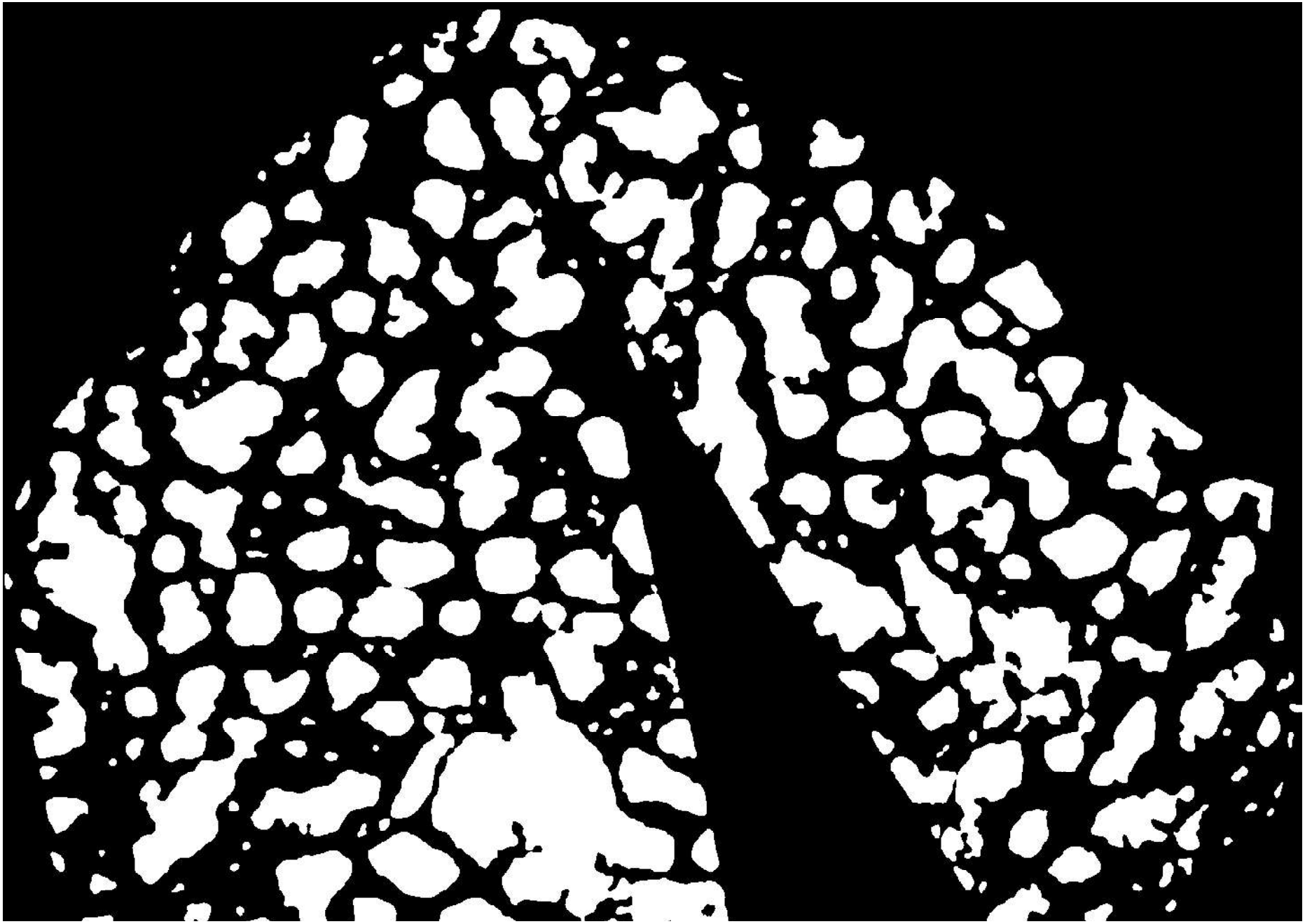
```
testbild1 = Last@ColorSeparate@
```

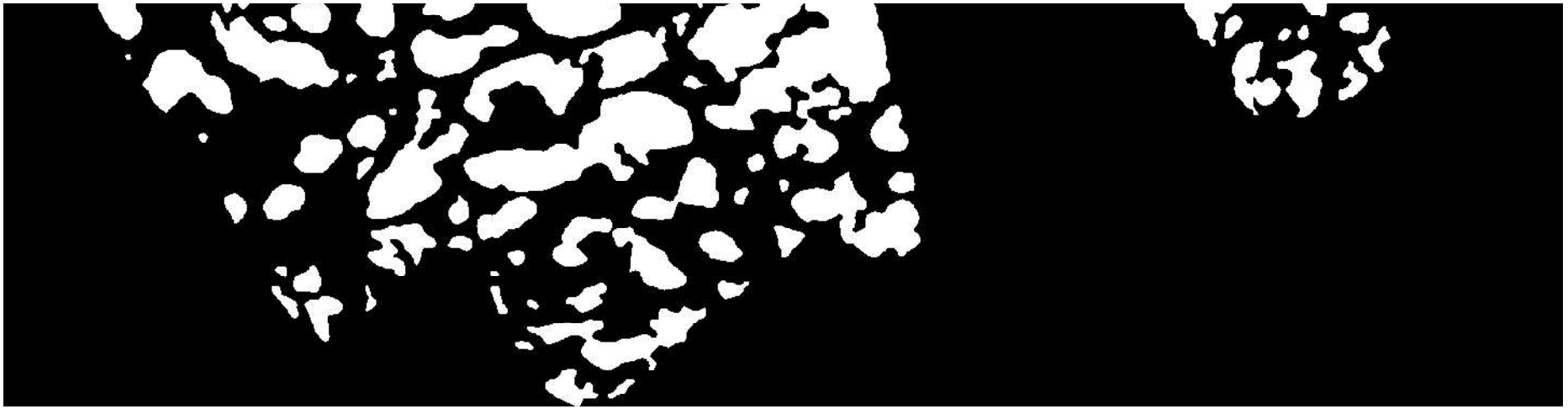


Colorize@MorphologicalComponents@testbild1

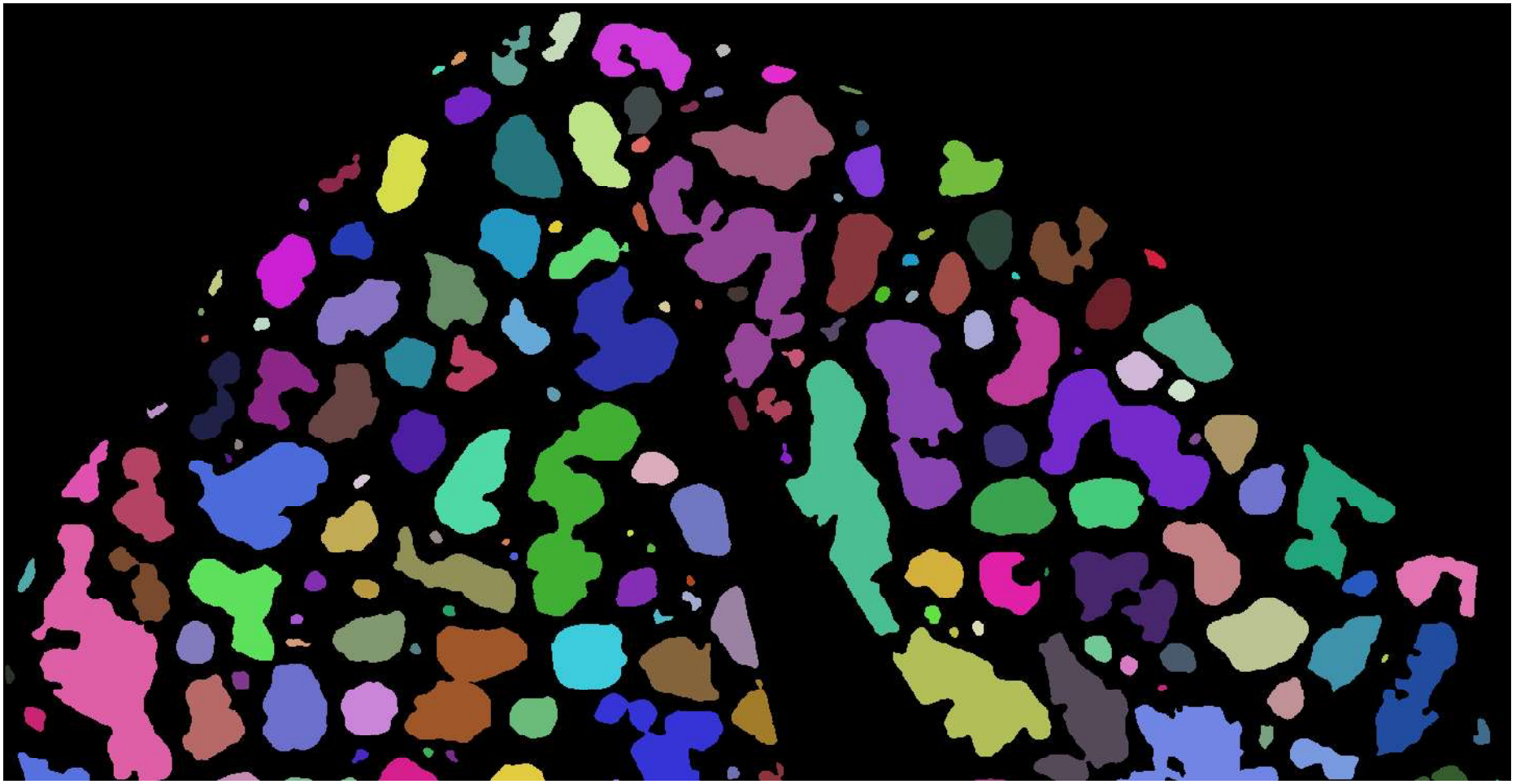


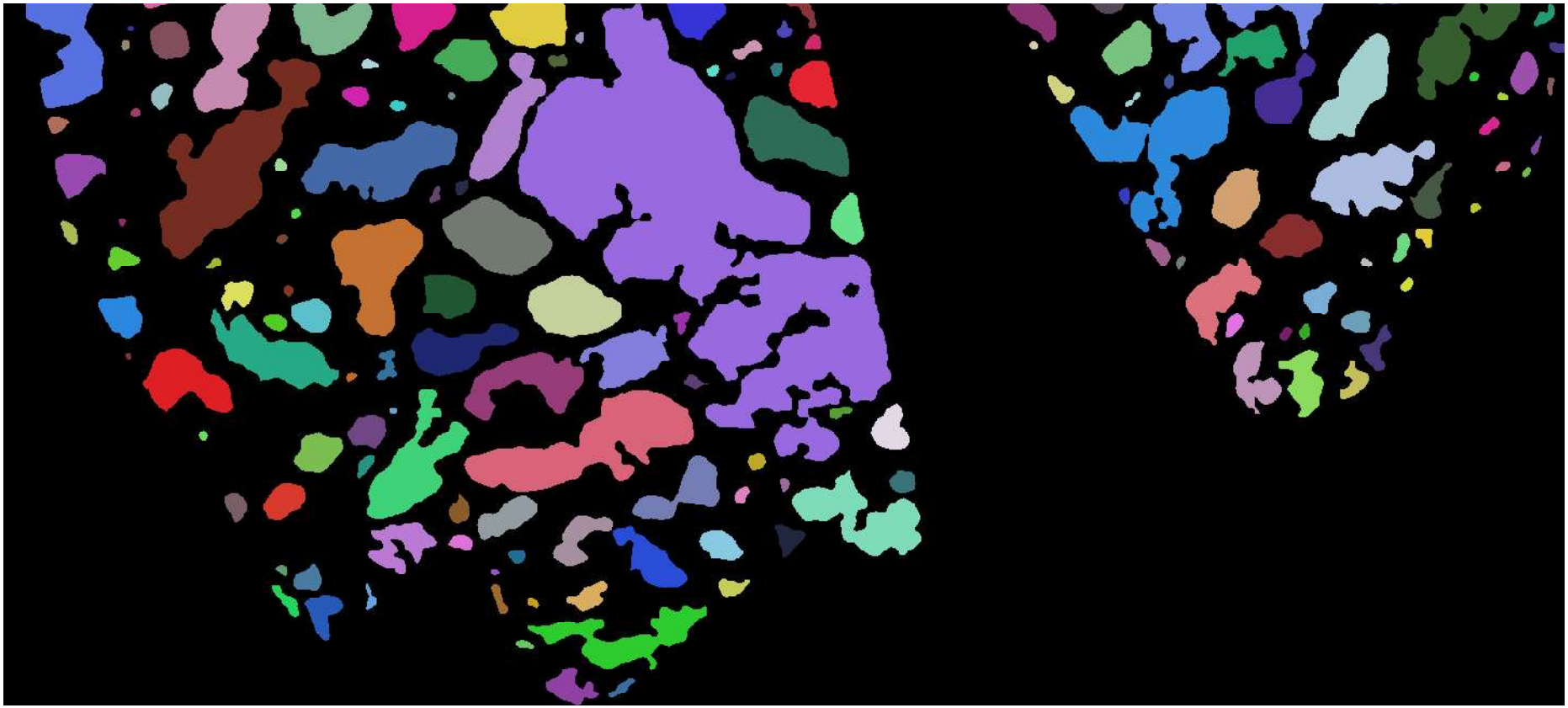
```
testbild2 = Last@ColorSeparate@
```





Colorize@MorphologicalComponents@testbild2





Einfachste Größen: Pixelzahl (Fläche) und Kantenpixelzahl (Umfang)

```
flächen1 = ComponentMeasurements[testbild1, {"Count"}][[All, 2]] // Flatten
```

```
{1368, 461, 158, 658, 1366, 581, 924, 621, 22 544, 699, 2116, 192, 486, 344, 1169, 690, 1672, 459, 6771, 245, 23 312,
 5299, 60, 4719, 21 932, 83 847, 740, 2939, 102, 683, 238, 135 547, 2613, 733, 813, 87, 139, 612, 172, 1273, 354, 56, 195}
```

```
flächen2 = ComponentMeasurements[testbild2, {"Count"}][[All, 2]] // Flatten
```

```
{1194, 4371, 1387, 166, 145, 546, 95, 108, 1441, 1546, 199, 157, 3904, 4668, 185, 8107, 3175, 258, 1745, 715, 14 989, 2756, 72, 96, 327, 3552,
 2344, 4771, 154, 1381, 3512, 2204, 149, 3277, 286, 3697, 1990, 183, 8719, 218, 53, 2061, 3960, 294, 210, 127, 5114, 1702, 72, 108,
 4733, 43, 1089, 195, 310, 11 687, 45, 1837, 2231, 55, 3409, 281, 1478, 2352, 15 742, 3944, 12 315, 500, 175, 490, 780, 176, 13 079, 2903,
 2541, 1674, 5755, 122, 82, 1430, 10 091, 157, 3872, 1351, 63, 2118, 173, 4460, 3541, 3624, 2317, 4296, 18 446, 123, 41, 46, 6991, 63,
 2362, 2323, 2853, 6280, 47, 4891, 3077, 367, 6040, 1486, 40, 740, 391, 78, 468, 2741, 241, 5798, 126, 270, 3225, 138, 3764, 170, 1504,
 161, 6162, 4678, 9172, 11 280, 88, 8722, 646, 4041, 187, 981, 106, 47, 311, 148, 5163, 302, 4048, 1254, 1871, 2758, 54, 1875, 12 382,
 452, 8023, 361, 286, 1868, 86, 175, 112, 3097, 3673, 5206, 148, 6241, 543, 5270, 3149, 1208, 71 604, 247, 179, 1859, 1252, 1869, 38,
 75, 4649, 189, 2047, 62, 322, 46, 141, 860, 4052, 2354, 203, 12 880, 119, 1805, 140, 116, 72, 86, 129, 469, 99, 430, 443, 9946, 53,
 93, 81, 5327, 149, 66, 277, 231, 7394, 113, 5331, 1564, 140, 119, 1043, 68, 2254, 159, 127, 157, 1245, 5963, 84, 74, 2075, 53, 6619,
 285, 235, 351, 96, 397, 515, 99, 112, 81, 3024, 2016, 4394, 136, 710, 752, 94, 1382, 1089, 535, 245, 286, 4947, 294, 3368, 132, 688,
 129, 3303, 3769, 366, 1790, 4364, 31, 599, 72, 200, 1953, 11 162, 1258, 210, 45, 1040, 5568, 76, 1442, 253, 224, 499, 109, 1142, 197,
 486, 437, 1576, 2613, 5505, 1572, 2004, 603, 2211, 979, 292, 198, 571, 95, 38, 388, 701, 158, 259, 353, 986, 84, 4488, 136, 1103, 221}
```

```
umfänge1 = ComponentMeasurements[testbild1, {"PerimeterLength"}][[All, 2]] // Flatten
```

```
{236, 158, 68, 180, 244, 118, 208, 164, 1140, 122, 332, 68, 120, 88, 176, 164, 220, 102, 918, 110,
 1632, 420, 34, 556, 784, 2512, 148, 270, 48, 148, 90, 7874, 274, 154, 140, 42, 62, 136, 64, 226, 100, 34, 64}
```

```
umfänge2 = ComponentMeasurements[testbild2, {"PerimeterLength"}][[All, 2]] // Flatten
```

```
{202, 474, 274, 60, 64, 114, 48, 70, 184, 194, 66, 64, 340, 332, 62, 540, 292, 76, 198, 188, 1440, 288, 38, 46, 98, 286, 226, 364, 58, 176, 436,
 306, 62, 290, 90, 336, 220, 62, 596, 84, 36, 212, 342, 80, 68, 54, 418, 226, 38, 48, 364, 30, 150, 62, 104, 822, 30, 260, 216, 34, 376,
 92, 184, 342, 988, 334, 814, 104, 60, 116, 178, 82, 966, 256, 244, 184, 432, 54, 40, 246, 612, 72, 384, 174, 40, 216, 68, 312, 278, 288,
 238, 358, 1120, 56, 30, 32, 680, 36, 342, 236, 306, 618, 30, 478, 394, 112, 476, 180, 32, 134, 90, 40, 102, 282, 86, 380, 50, 76, 276, 52,
 326, 76, 174, 58, 604, 308, 620, 688, 42, 702, 118, 332, 78, 144, 48, 34, 82, 62, 342, 76, 338, 172, 252, 240, 32, 198, 1106, 100, 722,
 98, 82, 246, 44, 72, 56, 286, 292, 570, 54, 564, 176, 452, 308, 188, 3736, 94, 68, 216, 160, 296, 26, 40, 400, 64, 220, 36, 102, 30, 54,
 144, 460, 288, 68, 904, 56, 202, 54, 52, 40, 40, 52, 116, 52, 100, 98, 1008, 32, 62, 40, 400, 58, 36, 78, 80, 578, 62, 540, 196, 54, 54,
 212, 40, 230, 58, 60, 58, 176, 392, 42, 40, 230, 34, 456, 86, 76, 98, 46, 108, 114, 52, 52, 44, 370, 208, 324, 56, 146, 138, 46, 180, 152,
 108, 86, 84, 486, 80, 352, 56, 172, 54, 388, 362, 134, 268, 488, 26, 158, 42, 80, 340, 842, 182, 76, 30, 156, 626, 42, 184, 74, 80, 102,
 48, 160, 68, 108, 102, 222, 340, 616, 258, 370, 132, 308, 154, 84, 66, 114, 48, 28, 108, 152, 88, 94, 122, 180, 46, 698, 58, 210, 96}
```

Kompaktheit K: Segmentfläche bezogen auf Kreisfläche gleichen Umfangs: $K = \frac{A}{A_c}$

äquiperimetr. Kreisfläche bestimmt über $A_c = \frac{\pi d^2}{4}$ und $P = \pi d$ als $A_c = \frac{P^2}{4\pi}$

```
äquiflächen1 = (umfänge1^2) / (4 π) // N
```

{4432.15, 1986.57, 367.966, 2578.31, 4737.72, 1108.04, 3442.84, 2140.32, 103 419., 1184.43, 8771.35, 367.966, 1145.92, 616.248, 2464.99, 2140.32, 3851.55, 827.924, 67 061.8, 962.887, 211 949., 14 037.5, 91.9916, 24 600.3, 48 912.8, 502 145., 1743.06, 5801.2, 183.346, 1743.06, 644.578, 4.93379×10^6, 5974.36, 1887.26, 1559.72, 140.375, 305.896, 1471.86, 325.949, 4064.5, 795.775, 91.9916, 325.949}

```
äquiflächen2 = (umfänge2^2) / (4 π) // N
```

{3247.08, 17 879.1, 5974.36, 286.479, 325.949, 1034.19, 183.346, 389.93, 2694.17, 2994.98, 346.639, 325.949, 9199.16, 8771.35, 305.896, 23 204.8, 6785.09, 459.639, 3119.76, 2812.59, 165 012., 6600.47, 114.91, 168.386, 764.262, 6509.12, 4064.5, 10 543.7, 267.699, 2464.99, 15 127.4, 7451.32, 305.896, 6692.47, 644.578, 8983.98, 3851.55, 305.896, 28 267.2, 561.499, 103.132, 3576.53, 9307.7, 509.296, 367.966, 232.048, 13 904.1, 4064.5, 114.91, 183.346, 10 543.7, 71.6197, 1790.49, 305.896, 860.71, 53 769.2, 71.6197, 5379.44, 3712.77, 91.9916, 11 250.3, 673.544, 2694.17, 9307.7, 77 679.1, 8877.34, 52 727.7, 860.71, 286.479, 1070.79, 2521.33, 535.079, 74 258.2, 5215.19, 4737.72, 2694.17, 14 851.1, 232.048, 127.324, 4815.71, 29 805.3, 412.53, 11 734.2, 2409.29, 127.324, 3712.77, 367.966, 7746.39, 6150.07, 6600.47, 4507.59, 10 199., 99 822., 249.555, 71.6197, 81.4873, 36 796.6, 103.132, 9307.7, 4432.15, 7451.32, 30 392.5, 71.6197, 18 182.2, 12 353.3, 998.22, 18 030.3, 2578.31, 81.4873, 1428.89, 644.578, 127.324, 827.924, 6328.32, 588.555, 11 491., 198.944, 459.639, 6061.89, 215.177, 8457.18, 459.639, 2409.29, 267.699, 29 031.1, 7549.04, 30 589.6, 37 667.5, 140.375, 39 216.1, 1108.04, 8771.35, 484.149, 1650.12, 183.346, 91.9916, 535.079, 305.896, 9307.7, 459.639, 9091.25, 2354.22, 5053.49, 4583.66, 81.4873, 3119.76, 97 342., 795.775, 41 482.5, 764.262, 535.079, 4815.71, 154.062, 412.53, 249.555, 6509.12, 6785.09, 25 854.7, 232.048, 25 313.3, 2464.99, 16 258., 7549.04, 2812.59, 1.11072×10^6, 703.147, 367.966, 3712.77, 2037.18, 6972.26, 53.7944, 127.324, 12 732.4, 325.949, 3851.55, 103.132, 827.924, 71.6197, 232.048, 1650.12, 16 838.6, 6600.47, 367.966, 65 032., 249.555, 3247.08, 232.048, 215.177, 127.324, 127.324, 215.177, 1070.79, 215.177, 795.775, 764.262, 80 855.8, 81.4873, 305.896, 127.324, 12 732.4, 267.699, 103.132, 484.149, 509.296, 26 585.6, 305.896, 23 204.8, 3057.05, 232.048, 232.048, 3576.53, 127.324, 4209.65, 267.699, 286.479, 267.699, 2464.99, 12 228.2, 140.375, 127.324, 4209.65, 91.9916, 16 547., 588.555, 459.639, 764.262, 168.386, 928.192, 1034.19, 215.177, 215.177, 154.062, 10 894.2, 3442.84, 8353.72, 249.555, 1696.27, 1515.47, 168.386, 2578.31, 1838.56, 928.192, 588.555, 561.499, 18 795.9, 509.296, 9859.97, 249.555, 2354.22, 232.048, 11 979.9, 10 428.2, 1428.89, 5715.57, 18 950.9, 53.7944, 1986.57, 140.375, 509.296, 9199.16, 56 417.6, 2635.92, 459.639, 71.6197, 1936.6, 31 184.5, 140.375, 2694.17, 435.766, 509.296, 827.924, 183.346, 2037.18, 367.966, 928.192, 827.924, 3921.9, 9199.16, 30 196.1, 5296.99, 10 894.2, 1386.56, 7549.04, 1887.26, 561.499, 346.639, 1034.19, 183.346, 62.3887, 928.192, 1838.56, 616.248, 703.147, 1184.43, 2578.31, 168.386, 38 770.5, 267.699, 3509.37, 733.386}

kompaktheiten1 = flächen1 / äquiflächen1

```
{0.308654, 0.232058, 0.429387, 0.255206, 0.288324, 0.524351, 0.268383, 0.290144, 0.217987, 0.590157,
 0.24124, 0.521787, 0.424115, 0.558217, 0.474241, 0.322382, 0.434111, 0.554399, 0.100967, 0.254443, 0.109989,
 0.37749, 0.652234, 0.191827, 0.44839, 0.166978, 0.42454, 0.50662, 0.556324, 0.391839, 0.369234, 0.0274732,
 0.437369, 0.388394, 0.521248, 0.61977, 0.454403, 0.415799, 0.527689, 0.3132, 0.44485, 0.608752, 0.598253}
```

kompaktheiten2 = flächen2 / äquiflächen2

```
{0.367715, 0.244475, 0.232159, 0.579449, 0.444854, 0.52795, 0.518145, 0.276973, 0.534858, 0.516197, 0.574084, 0.48167, 0.424387, 0.532187,
 0.604781, 0.349368, 0.467938, 0.561309, 0.559339, 0.254214, 0.0908359, 0.417546, 0.626578, 0.570119, 0.427864, 0.545696, 0.576701,
 0.452498, 0.575274, 0.560245, 0.232162, 0.295787, 0.487094, 0.489655, 0.443701, 0.41151, 0.516675, 0.598243, 0.308449, 0.388247,
 0.513903, 0.576257, 0.425454, 0.577268, 0.570705, 0.547301, 0.367805, 0.418748, 0.626578, 0.589049, 0.448894, 0.600393, 0.608212,
 0.637472, 0.360168, 0.217355, 0.628319, 0.341486, 0.6009, 0.597881, 0.303013, 0.417196, 0.548591, 0.252694, 0.202654, 0.444277,
 0.233558, 0.580916, 0.610865, 0.457604, 0.30936, 0.328923, 0.176129, 0.556643, 0.536333, 0.621341, 0.387514, 0.525754, 0.644026,
 0.296945, 0.338564, 0.380579, 0.329976, 0.560747, 0.494801, 0.570464, 0.470152, 0.575752, 0.575766, 0.549051, 0.514022, 0.421219,
 0.184789, 0.492877, 0.572468, 0.564505, 0.18999, 0.610865, 0.253768, 0.524125, 0.382885, 0.20663, 0.656244, 0.269, 0.249083,
 0.367654, 0.334991, 0.576347, 0.490874, 0.517883, 0.606599, 0.612611, 0.565269, 0.433132, 0.409477, 0.504569, 0.633345, 0.587417,
 0.532012, 0.641331, 0.445066, 0.369855, 0.624251, 0.601423, 0.212255, 0.619682, 0.299841, 0.299462, 0.626894, 0.222409, 0.583013,
 0.460705, 0.386244, 0.594503, 0.57814, 0.510916, 0.581223, 0.483825, 0.554702, 0.657037, 0.445263, 0.532661, 0.370239, 0.601702,
 0.66268, 0.601009, 0.127201, 0.568, 0.193407, 0.472351, 0.534501, 0.387897, 0.558217, 0.424212, 0.448799, 0.475794, 0.541334,
 0.201356, 0.637799, 0.24655, 0.220285, 0.324148, 0.417139, 0.429498, 0.0644664, 0.351278, 0.486458, 0.500705, 0.614574, 0.268062,
 0.706394, 0.589049, 0.365132, 0.579845, 0.531474, 0.601169, 0.388925, 0.642281, 0.607633, 0.521175, 0.240638, 0.356641, 0.551681,
 0.198056, 0.476849, 0.555884, 0.603324, 0.53909, 0.565487, 0.675442, 0.599505, 0.437993, 0.460085, 0.540354, 0.579644, 0.123009,
 0.650408, 0.304025, 0.636173, 0.418382, 0.556596, 0.639954, 0.572138, 0.453567, 0.278121, 0.369407, 0.229737, 0.511605, 0.603324,
 0.512825, 0.291623, 0.534071, 0.535437, 0.593952, 0.443314, 0.58648, 0.505073, 0.487644, 0.598399, 0.581195, 0.492915, 0.57614,
 0.400012, 0.484237, 0.51127, 0.459267, 0.570119, 0.427713, 0.497975, 0.460085, 0.520501, 0.525762, 0.27758, 0.585563, 0.525993,
 0.54497, 0.418565, 0.496215, 0.558241, 0.53601, 0.592312, 0.57639, 0.416274, 0.509351, 0.263196, 0.577268, 0.341583, 0.528942,
 0.292241, 0.55592, 0.275712, 0.361426, 0.256142, 0.313179, 0.230279, 0.576268, 0.301524, 0.512913, 0.392699, 0.212302, 0.197846,
 0.477252, 0.45688, 0.628319, 0.537024, 0.17855, 0.541408, 0.535229, 0.580587, 0.439823, 0.602712, 0.594503, 0.560578, 0.535375,
 0.523599, 0.527826, 0.401846, 0.284048, 0.182308, 0.296772, 0.183952, 0.43489, 0.292885, 0.518742, 0.520037, 0.571199, 0.552124,
 0.518145, 0.609084, 0.418017, 0.381277, 0.25639, 0.368344, 0.298033, 0.382421, 0.498854, 0.115758, 0.508034, 0.314302, 0.301342}
```

Zirkularität Z: Umfang des zur Segmentfläche flächengleichen Kreisumfangs bezogen auf den Segmentflächenumfang:

$$Z = \frac{P_c}{P}$$

wegen $A_c = \frac{P^2}{4\pi}$ und $A = \frac{P_c^2}{4\pi}$ folgt $Z = \sqrt{\frac{A}{A_c}}$ und damit $Z^2 = \frac{A}{A_c} = K$

```
Flatten@ComponentMeasurements[testbild1, {"Circularity"}][[All, 2]]^2
```

```
{0.308654, 0.232058, 0.429387, 0.255206, 0.288324, 0.524351, 0.268383, 0.290144, 0.217987, 0.590157,
 0.24124, 0.521787, 0.424115, 0.558217, 0.474241, 0.322382, 0.434111, 0.554399, 0.100967, 0.254443, 0.109989,
 0.37749, 0.652234, 0.191827, 0.44839, 0.166978, 0.42454, 0.50662, 0.556324, 0.391839, 0.369234, 0.0274732,
 0.437369, 0.388394, 0.521248, 0.61977, 0.454403, 0.415799, 0.527689, 0.3132, 0.44485, 0.608752, 0.598253}
```

```
Flatten@ComponentMeasurements[testbild2, {"Circularity"}][[All, 2]]^2
```

```
{0.367715, 0.244475, 0.232159, 0.579449, 0.444854, 0.52795, 0.518145, 0.276973, 0.534858, 0.516197, 0.574084, 0.48167, 0.424387, 0.532187,
 0.604781, 0.349368, 0.467938, 0.561309, 0.559339, 0.254214, 0.0908359, 0.417546, 0.626578, 0.570119, 0.427864, 0.545696, 0.576701,
 0.452498, 0.575274, 0.560245, 0.232162, 0.295787, 0.487094, 0.489655, 0.443701, 0.41151, 0.516675, 0.598243, 0.308449, 0.388247,
 0.513903, 0.576257, 0.425454, 0.577268, 0.570705, 0.547301, 0.367805, 0.418748, 0.626578, 0.589049, 0.448894, 0.600393, 0.608212,
 0.637472, 0.360168, 0.217355, 0.628319, 0.341486, 0.6009, 0.597881, 0.303013, 0.417196, 0.548591, 0.252694, 0.202654, 0.444277,
 0.233558, 0.580916, 0.610865, 0.457604, 0.30936, 0.328923, 0.176129, 0.556643, 0.536333, 0.621341, 0.387514, 0.525754, 0.644026,
 0.296945, 0.338564, 0.380579, 0.329976, 0.560747, 0.494801, 0.570464, 0.470152, 0.575752, 0.575766, 0.549051, 0.514022, 0.421219,
 0.184789, 0.492877, 0.572468, 0.564505, 0.18999, 0.610865, 0.253768, 0.524125, 0.382885, 0.20663, 0.656244, 0.269, 0.249083,
 0.367654, 0.334991, 0.576347, 0.490874, 0.517883, 0.606599, 0.612611, 0.565269, 0.433132, 0.409477, 0.504569, 0.633345, 0.587417,
 0.532012, 0.641331, 0.445066, 0.369855, 0.624251, 0.601423, 0.212255, 0.619682, 0.299841, 0.299462, 0.626894, 0.222409, 0.583013,
 0.460705, 0.386244, 0.594503, 0.57814, 0.510916, 0.581223, 0.483825, 0.554702, 0.657037, 0.445263, 0.532661, 0.370239, 0.601702,
 0.66268, 0.601009, 0.127201, 0.568, 0.193407, 0.472351, 0.534501, 0.387897, 0.558217, 0.424212, 0.448799, 0.475794, 0.541334,
 0.201356, 0.637799, 0.24655, 0.220285, 0.324148, 0.417139, 0.429498, 0.0644664, 0.351278, 0.486458, 0.500705, 0.614574, 0.268062,
 0.706394, 0.589049, 0.365132, 0.579845, 0.531474, 0.601169, 0.388925, 0.642281, 0.607633, 0.521175, 0.240638, 0.356641, 0.551681,
 0.198056, 0.476849, 0.555884, 0.603324, 0.53909, 0.565487, 0.675442, 0.599505, 0.437993, 0.460085, 0.540354, 0.579644, 0.123009,
 0.650408, 0.304025, 0.636173, 0.418382, 0.556596, 0.639954, 0.572138, 0.453567, 0.278121, 0.369407, 0.229737, 0.511605, 0.603324,
 0.512825, 0.291623, 0.534071, 0.535437, 0.593952, 0.443314, 0.58648, 0.505073, 0.487644, 0.598399, 0.581195, 0.492915, 0.57614,
 0.400012, 0.484237, 0.51127, 0.459267, 0.570119, 0.427713, 0.497975, 0.460085, 0.520501, 0.525762, 0.27758, 0.585563, 0.525993,
 0.54497, 0.418565, 0.496215, 0.558241, 0.53601, 0.592312, 0.57639, 0.416274, 0.509351, 0.263196, 0.577268, 0.341583, 0.528942,
 0.292241, 0.55592, 0.275712, 0.361426, 0.256142, 0.313179, 0.230279, 0.576268, 0.301524, 0.512913, 0.392699, 0.212302, 0.197846,
 0.477252, 0.45688, 0.628319, 0.537024, 0.17855, 0.541408, 0.535229, 0.580587, 0.439823, 0.602712, 0.594503, 0.560578, 0.535375,
 0.523599, 0.527826, 0.401846, 0.284048, 0.182308, 0.296772, 0.183952, 0.43489, 0.292885, 0.518742, 0.520037, 0.571199, 0.552124,
 0.518145, 0.609084, 0.418017, 0.381277, 0.25639, 0.368344, 0.298033, 0.382421, 0.498854, 0.115758, 0.508034, 0.314302, 0.301342}
```

```
gewichtetegesamtkompaktheit1 = Total[kompaktheiten1 * flächen1] / Total[flächen1]
```

0.148592

```
gewichtetegesamtkompaktheit2 = Total[kompaktheiten2 * flächen2] / Total[flächen2]
```

0.311387

Solidität S: Segmentfläche bezogen auf Fläche der konvexen Hülle: $S = \frac{A}{A_h}$

```
hüllflächen1 = ComponentMeasurements[testbild1, {"ConvexCount"}][[All, 2]] // Flatten // N
```

{1665., 624., 188., 791., 1656., 615., 1139., 739., 28 796., 734., 2509., 198., 532., 365., 1242., 888., 1855., 472., 11 810., 313., 31 140., 5913., 61., 6024., 23 224., 106 893., 829., 3056., 108., 744., 312., 363 010., 2894., 831., 878., 90., 154., 720., 191., 1644., 405., 57., 200.}

```
hüllflächen2 = ComponentMeasurements[testbild2, {"⁝"}][[All, 2]] // Flatten // N
```

{1359., 5475., 1921., 169., 156., 561., 96., 125., 1504., 1667., 208., 168., 4321., 4846., 189., 9714., 3393., 264., 1817., 972., 27 594., 3161., 73., 100., 341., 3721., 2379., 5108., 159., 1448., 4776., 2759., 154., 3439., 308., 4336., 2048., 194., 10 843., 235., 55., 2115., 4407., 300., 217., 130., 6455., 1895., 73., 111., 4954., 44., 1112., 203., 374., 14 861., 46., 2134., 2299., 56., 4459., 324., 1502., 3258., 20 371., 4491., 17 929., 512., 180., 514., 961., 212., 18 646., 2991., 2609., 1714., 6267., 128., 84., 1714., 12 050., 191., 4796., 1388., 68., 2217., 174., 4591., 3670., 3744., 2465., 4943., 25 904., 127., 43., 47., 10 385., 65., 3073., 2465., 3449., 8603., 48., 6506., 4427., 393., 7696., 1529., 42., 769., 398., 79., 482., 2843., 277., 6024., 130., 278., 3344., 141., 4048., 219., 1525., 161., 7785., 4849., 11 449., 13 196., 90., 12 628., 664., 4489., 210., 995., 108., 49., 321., 157., 5355., 316., 4422., 1310., 2090., 2847., 54., 1940., 20 269., 467., 11 102., 411., 300., 2201., 91., 200., 122., 3275., 3891., 6473., 153., 8682., 823., 6350., 3634., 1311., 100 180., 289., 194., 1937., 1293., 2624., 38., 78., 5197., 196., 2128., 64., 385., 46., 147., 883., 5271., 2839., 224., 18 347., 134., 1886., 143., 122., 73., 89., 133., 503., 106., 442., 457., 15 967., 54., 107., 83., 5937., 156., 66., 283., 242., 9249., 122., 6710., 1683., 144., 125., 1328., 70., 2355., 164., 140., 161., 1292., 6153., 84., 76., 2220., 55., 7856., 298., 257., 376., 99., 412., 555., 106., 119., 87., 4023., 2106., 4533., 141., 829., 810., 97., 1467., 1111., 553., 283., 293., 6150., 303., 4193., 137., 947., 138., 4085., 4525., 547., 2285., 6569., 32., 772., 74., 237., 2652., 15 148., 1370., 239., 47., 1095., 7879., 79., 1516., 262., 231., 513., 112., 1177., 207., 508., 468., 1725., 3683., 7781., 2127., 2703., 676., 2682., 1012., 308., 202., 597., 101., 38., 433., 804., 221., 306., 416., 1210., 88., 7618., 142., 1370., 260.}

```
soliditäten1 = flächen1 / hüllflächen1
```

{0.821622, 0.738782, 0.840426, 0.831858, 0.824879, 0.944715, 0.811238, 0.840325, 0.782887, 0.952316, 0.843364, 0.969697, 0.913534, 0.942466, 0.941224, 0.777027, 0.901348, 0.972458, 0.573328, 0.782748, 0.748619, 0.896161, 0.983607, 0.783367, 0.944368, 0.784401, 0.892642, 0.961715, 0.944444, 0.918011, 0.762821, 0.373397, 0.902903, 0.88207, 0.925968, 0.966667, 0.902597, 0.85, 0.900524, 0.774331, 0.874074, 0.982456, 0.975}

```
soliditäten2 = flächen2 / hüllflächen2
```

```
{0.878587, 0.798356, 0.72202, 0.982249, 0.929487, 0.973262, 0.989583, 0.864, 0.958112, 0.927415, 0.956731, 0.934524, 0.903495,
 0.963269, 0.978836, 0.834569, 0.93575, 0.977273, 0.960374, 0.735597, 0.543198, 0.871876, 0.986301, 0.96, 0.958944, 0.954582,
 0.985288, 0.934025, 0.968553, 0.953729, 0.735343, 0.79884, 0.967532, 0.952893, 0.928571, 0.852629, 0.97168, 0.943299, 0.804113,
 0.92766, 0.963636, 0.974468, 0.89857, 0.98, 0.967742, 0.976923, 0.792254, 0.898153, 0.986301, 0.972973, 0.95539, 0.977273,
 0.979317, 0.960591, 0.828877, 0.786421, 0.978261, 0.860825, 0.970422, 0.982143, 0.764521, 0.867284, 0.984021, 0.721915, 0.772765,
 0.878201, 0.686876, 0.976563, 0.972222, 0.953307, 0.811655, 0.830189, 0.701437, 0.970578, 0.973936, 0.976663, 0.918302, 0.953125,
 0.97619, 0.834306, 0.837427, 0.82199, 0.807339, 0.973343, 0.926471, 0.955345, 0.994253, 0.971466, 0.96485, 0.967949, 0.939959,
 0.869108, 0.712091, 0.968504, 0.953488, 0.978723, 0.673182, 0.969231, 0.76863, 0.942394, 0.827196, 0.729978, 0.979167, 0.751768,
 0.695053, 0.933842, 0.784823, 0.971877, 0.952381, 0.962289, 0.982412, 0.987342, 0.970954, 0.964122, 0.870036, 0.962483,
 0.969231, 0.971223, 0.964414, 0.978723, 0.929842, 0.776256, 0.98623, 1., 0.791522, 0.964735, 0.801118, 0.854804, 0.977778,
 0.690687, 0.972892, 0.9002, 0.890476, 0.98593, 0.981481, 0.959184, 0.968847, 0.942675, 0.964146, 0.955696, 0.915423, 0.957252,
 0.895215, 0.968739, 1., 0.966495, 0.610884, 0.96788, 0.722663, 0.878345, 0.953333, 0.848705, 0.945055, 0.875, 0.918033, 0.945649,
 0.943973, 0.804264, 0.96732, 0.718844, 0.659781, 0.829921, 0.866538, 0.921434, 0.714753, 0.854671, 0.92268, 0.959732, 0.968291,
 0.712271, 1., 0.961538, 0.894555, 0.964286, 0.961936, 0.96875, 0.836364, 1., 0.959184, 0.973952, 0.768735, 0.829165, 0.90625,
 0.702022, 0.88806, 0.957052, 0.979021, 0.95082, 0.986301, 0.966292, 0.969925, 0.932406, 0.933962, 0.972851, 0.969365, 0.62291,
 0.981481, 0.869159, 0.975904, 0.897255, 0.955128, 1., 0.978799, 0.954545, 0.799438, 0.92623, 0.794486, 0.929293, 0.972222, 0.952,
 0.785392, 0.971429, 0.957113, 0.969512, 0.907143, 0.975155, 0.963622, 0.969121, 1., 0.973684, 0.934685, 0.963636, 0.842541,
 0.956376, 0.914397, 0.933511, 0.969697, 0.963592, 0.927928, 0.933962, 0.941176, 0.931034, 0.751678, 0.957265, 0.969336,
 0.964539, 0.856454, 0.928395, 0.969072, 0.942059, 0.980198, 0.96745, 0.865724, 0.976109, 0.80439, 0.970297, 0.803244, 0.963504,
 0.726505, 0.934783, 0.808568, 0.832928, 0.669104, 0.78337, 0.664332, 0.96875, 0.775907, 0.972973, 0.843882, 0.736425, 0.736863,
 0.918248, 0.878661, 0.957447, 0.949772, 0.706689, 0.962025, 0.951187, 0.965649, 0.969697, 0.97271, 0.973214, 0.970263, 0.951691,
 0.956693, 0.933761, 0.913623, 0.709476, 0.707493, 0.739069, 0.741398, 0.892012, 0.824385, 0.967391, 0.948052, 0.980198,
 0.956449, 0.940594, 1., 0.896074, 0.871891, 0.714932, 0.846405, 0.848558, 0.814876, 0.954545, 0.589131, 0.957746, 0.805109, 0.85}
```

```
gewichtetegesamtsolidität1 = Total[soliditäten1 * flächen1] / Total[flächen1]
```

```
0.628614
```

```
gewichtetegesamtsolidität2 = Total[soliditäten2 * flächen2] / Total[flächen2]
```

```
0.808572
```

Nochmals Zirkularität Z: Kreisumfang gleicher Fläche bezogen auf Segmentumfang: $Z = \frac{P_c}{P}$

```
gleichflächigekreisumfänge1 = Sqrt[4 flächen1 * Pi] // N
```

```
{131.114, 76.1124, 44.5588, 90.9322, 131.018, 85.4462, 107.756, 88.3386, 532.256, 93.7224, 163.066, 49.1197, 78.1489, 65.7482, 121.203,
 93.1171, 144.952, 75.9471, 291.697, 55.4866, 541.246, 258.049, 27.4587, 243.517, 524.982, 1026.48, 96.4319, 192.178, 35.8018,
 92.6436, 54.6882, 1305.12, 181.207, 95.9747, 101.077, 33.0647, 41.7938, 87.6962, 46.491, 126.479, 66.697, 26.5277, 49.5019}
```

```
gleichflächigekreisumfänge2 = Sqrt[4 flächen2 * Pi] // N
```

```
{122.492, 234.366, 132.021, 45.6729, 42.6863, 82.8326, 34.5515, 36.8398, 134.566, 139.383, 50.0071, 44.4176, 221.493, 242.198, 48.216,
 319.18, 199.745, 56.9396, 148.082, 94.789, 434.002, 186.099, 30.0795, 34.7329, 64.1031, 211.272, 171.626, 244.855, 43.9911, 131.735,
 210.079, 166.422, 43.2711, 202.929, 59.9498, 215.541, 158.136, 47.9546, 331.008, 52.3399, 25.8073, 160.933, 223.076, 60.7825,
 51.3706, 39.9491, 253.504, 146.246, 30.0795, 36.8398, 243.878, 23.2455, 116.982, 49.5019, 62.4145, 383.227, 23.78, 151.936, 167.438,
 26.2897, 206.975, 59.4235, 136.283, 171.919, 444.769, 222.625, 393.389, 79.2665, 46.8947, 78.4699, 99.0039, 47.0285, 405.408,
 190.998, 178.693, 145.038, 268.923, 39.1548, 32.1005, 134.052, 356.1, 44.4176, 220.583, 130.296, 28.1368, 163.143, 46.626, 236.74,
 210.944, 213.402, 170.635, 232.347, 481.455, 39.3149, 22.6985, 24.0427, 296.398, 28.1368, 172.284, 170.856, 189.346, 280.921,
 24.3027, 247.916, 196.639, 67.9107, 275.501, 136.651, 22.42, 96.4319, 70.096, 31.3078, 76.6881, 185.592, 55.0318, 269.926, 39.7915,
 58.2488, 201.312, 41.6432, 217.485, 46.2199, 137.477, 44.9798, 278.27, 242.457, 339.498, 376.495, 33.2542, 331.065, 90.0993,
 225.346, 48.4759, 111.03, 36.4971, 24.3027, 62.5151, 43.1257, 254.716, 61.6039, 225.541, 125.532, 153.335, 186.167, 26.0496,
 153.499, 394.458, 75.3658, 317.522, 67.3532, 59.9498, 153.212, 32.8741, 46.8947, 37.5158, 197.277, 214.84, 255.774, 43.1257,
 280.048, 82.6047, 257.342, 198.926, 123.208, 948.579, 55.7126, 47.4276, 152.843, 125.432, 153.253, 21.8523, 30.6998, 241.704,
 48.7344, 160.385, 27.9126, 63.6111, 24.0427, 42.0934, 103.957, 225.652, 171.992, 50.5072, 402.312, 38.6704, 150.606, 41.9439,
 38.1798, 30.0795, 32.8741, 40.2624, 76.77, 35.2714, 73.5088, 74.6117, 353.532, 25.8073, 34.1859, 31.9042, 258.73, 43.2711, 28.799,
 58.999, 53.8779, 304.821, 37.6829, 258.827, 140.192, 41.9439, 38.6704, 114.485, 29.2321, 168.299, 44.6996, 39.9491, 44.4176,
 125.08, 273.739, 32.4896, 30.4944, 161.478, 25.8073, 288.404, 59.8449, 54.3424, 66.4138, 34.7329, 70.6318, 80.4468, 35.2714,
 37.5158, 31.9042, 194.938, 159.166, 234.982, 41.3404, 94.457, 97.2107, 34.3692, 131.783, 116.982, 81.994, 55.4866, 59.9498,
 249.331, 60.7825, 205.727, 40.7279, 92.9821, 40.2624, 203.732, 217.63, 67.8181, 149.979, 234.179, 19.7372, 86.7598, 30.0795,
 50.1326, 156.659, 374.521, 125.732, 51.3706, 23.78, 114.32, 264.518, 30.9038, 134.613, 56.3852, 53.0553, 79.1872, 37.0099,
 119.795, 49.7552, 78.1489, 74.1047, 140.729, 181.207, 263.017, 140.55, 158.692, 87.049, 166.686, 110.917, 60.5754, 49.8813,
 84.7077, 34.5515, 21.8523, 69.8266, 93.8564, 44.5588, 57.0499, 66.6028, 111.312, 32.4896, 237.482, 41.3404, 117.732, 52.6988}
```

```
zirkularitäten1 = gleichflächigekreisumfänge1 / umfänge1
```

```
{0.555566, 0.481724, 0.655276, 0.505179, 0.536958, 0.724121, 0.518057, 0.53865, 0.466891, 0.768217,
 0.491162, 0.722348, 0.651241, 0.747139, 0.688652, 0.567787, 0.658871, 0.74458, 0.317752, 0.504423, 0.331646,
 0.614402, 0.80761, 0.437981, 0.669619, 0.408629, 0.651567, 0.711772, 0.745871, 0.62597, 0.607646, 0.16575,
 0.661339, 0.623213, 0.721975, 0.787255, 0.674094, 0.644825, 0.726422, 0.559643, 0.66697, 0.780225, 0.773468}
```

```
ComponentMeasurements[testbild1, {"Circularity"}][[All, 2]] // Flatten
```

```
{0.555566, 0.481724, 0.655276, 0.505179, 0.536958, 0.724121, 0.518057, 0.53865, 0.466891, 0.768217,
 0.491162, 0.722348, 0.651241, 0.747139, 0.688652, 0.567787, 0.658871, 0.74458, 0.317752, 0.504423, 0.331646,
 0.614402, 0.80761, 0.437981, 0.669619, 0.408629, 0.651567, 0.711772, 0.745871, 0.62597, 0.607646, 0.16575,
 0.661339, 0.623213, 0.721975, 0.787255, 0.674094, 0.644825, 0.726422, 0.559643, 0.66697, 0.780225, 0.773468}
```

zirkularitäten2 = gleichflächigekreisumfänge2 / umfänge2

{0.606395, 0.494444, 0.481829, 0.761216, 0.666974, 0.726602, 0.719823, 0.526282, 0.73134, 0.718469, 0.757683, 0.694024, 0.65145, 0.729512,
0.777677, 0.591073, 0.68406, 0.749206, 0.74789, 0.504197, 0.30139, 0.646178, 0.791567, 0.755062, 0.654113, 0.738712, 0.759408,
0.67268, 0.758468, 0.748495, 0.481832, 0.543863, 0.697921, 0.699754, 0.666109, 0.641491, 0.718801, 0.773462, 0.555382, 0.623094,
0.71687, 0.759116, 0.652269, 0.759781, 0.75545, 0.739798, 0.60647, 0.647107, 0.791567, 0.767495, 0.669995, 0.77485, 0.77988,
0.798418, 0.60014, 0.466213, 0.792665, 0.584368, 0.775177, 0.773228, 0.550466, 0.645907, 0.740669, 0.502687, 0.450171, 0.666541,
0.483279, 0.762178, 0.781579, 0.676464, 0.556202, 0.573518, 0.419677, 0.746085, 0.732348, 0.788252, 0.622506, 0.725089, 0.802513,
0.544926, 0.581863, 0.616911, 0.574436, 0.74883, 0.703421, 0.755291, 0.685676, 0.758783, 0.758793, 0.74098, 0.716954, 0.649014,
0.429871, 0.702052, 0.756616, 0.751335, 0.435879, 0.781579, 0.503754, 0.723965, 0.618777, 0.454565, 0.810089, 0.518652, 0.499083,
0.606345, 0.578784, 0.759175, 0.700624, 0.719641, 0.778845, 0.782694, 0.751844, 0.658128, 0.639904, 0.71033, 0.79583, 0.766431,
0.729392, 0.800831, 0.667133, 0.608157, 0.790096, 0.775514, 0.460711, 0.787199, 0.547577, 0.547231, 0.791766, 0.471602, 0.763553,
0.678752, 0.621486, 0.77104, 0.760355, 0.714784, 0.76238, 0.695575, 0.744783, 0.810578, 0.667281, 0.729836, 0.608473, 0.775695,
0.814051, 0.775248, 0.356652, 0.753658, 0.439781, 0.687278, 0.731095, 0.622814, 0.747139, 0.651316, 0.669925, 0.689778, 0.735754,
0.448727, 0.798623, 0.496538, 0.469345, 0.56934, 0.645863, 0.655361, 0.253902, 0.592687, 0.697465, 0.707605, 0.783948, 0.517747,
0.840472, 0.767495, 0.604261, 0.761475, 0.729023, 0.775351, 0.623638, 0.801424, 0.779508, 0.721924, 0.490548, 0.597194, 0.742752,
0.445035, 0.690542, 0.745576, 0.776739, 0.734227, 0.751988, 0.821853, 0.774277, 0.66181, 0.678296, 0.735088, 0.761344, 0.350727,
0.806479, 0.551385, 0.797604, 0.646824, 0.746054, 0.799971, 0.756398, 0.673474, 0.527372, 0.607788, 0.479309, 0.715265, 0.776739,
0.716118, 0.540022, 0.730801, 0.731735, 0.770683, 0.665818, 0.76582, 0.710685, 0.698315, 0.773562, 0.762361, 0.702079, 0.759039,
0.632465, 0.695871, 0.715032, 0.677692, 0.755062, 0.653998, 0.705673, 0.678296, 0.721457, 0.725095, 0.526859, 0.765221, 0.725254,
0.738221, 0.646966, 0.704425, 0.747156, 0.732127, 0.769618, 0.759203, 0.645193, 0.713688, 0.513026, 0.759781, 0.584451, 0.727284,
0.540593, 0.7456, 0.525082, 0.601187, 0.506105, 0.559624, 0.479874, 0.759123, 0.549112, 0.71618, 0.626657, 0.460763, 0.444799,
0.690834, 0.675929, 0.792665, 0.732819, 0.422552, 0.735804, 0.731593, 0.761962, 0.663192, 0.776345, 0.77104, 0.748718, 0.731693,
0.723601, 0.726516, 0.633914, 0.532961, 0.426975, 0.544768, 0.428896, 0.659462, 0.541189, 0.720237, 0.721136, 0.755777, 0.74305,
0.719823, 0.780439, 0.646542, 0.617476, 0.50635, 0.606914, 0.545924, 0.618402, 0.706296, 0.340233, 0.712765, 0.560626, 0.548946}

```
ComponentMeasurements[testbild2, {"Circularity"}][[All, 2]] // Flatten
```

```
{0.606395, 0.494444, 0.481829, 0.761216, 0.666974, 0.726602, 0.719823, 0.526282, 0.73134, 0.718469, 0.757683, 0.694024, 0.65145, 0.729512,
 0.777677, 0.591073, 0.68406, 0.749206, 0.74789, 0.504197, 0.30139, 0.646178, 0.791567, 0.755062, 0.654113, 0.738712, 0.759408,
 0.67268, 0.758468, 0.748495, 0.481832, 0.543863, 0.697921, 0.699754, 0.666109, 0.641491, 0.718801, 0.773462, 0.555382, 0.623094,
 0.71687, 0.759116, 0.652269, 0.759781, 0.75545, 0.739798, 0.60647, 0.647107, 0.791567, 0.767495, 0.669995, 0.77485, 0.77988,
 0.798418, 0.60014, 0.466213, 0.792665, 0.584368, 0.775177, 0.773228, 0.550466, 0.645907, 0.740669, 0.502687, 0.450171, 0.666541,
 0.483279, 0.762178, 0.781579, 0.676464, 0.556202, 0.573518, 0.419677, 0.746085, 0.732348, 0.788252, 0.622506, 0.725089, 0.802513,
 0.544926, 0.581863, 0.616911, 0.574436, 0.74883, 0.703421, 0.755291, 0.685676, 0.758783, 0.758793, 0.74098, 0.716954, 0.649014,
 0.429871, 0.702052, 0.756616, 0.751335, 0.435879, 0.781579, 0.503754, 0.723965, 0.618777, 0.454565, 0.810089, 0.518652, 0.499083,
 0.606345, 0.578784, 0.759175, 0.700624, 0.719641, 0.778845, 0.782694, 0.751844, 0.658128, 0.639904, 0.71033, 0.79583, 0.766431,
 0.729392, 0.800831, 0.667133, 0.608157, 0.790096, 0.775514, 0.460711, 0.787199, 0.547577, 0.547231, 0.791766, 0.471602, 0.763553,
 0.678752, 0.621486, 0.77104, 0.760355, 0.714784, 0.76238, 0.695575, 0.744783, 0.810578, 0.667281, 0.729836, 0.608473, 0.775695,
 0.814051, 0.775248, 0.356652, 0.753658, 0.439781, 0.687278, 0.731095, 0.622814, 0.747139, 0.651316, 0.669925, 0.689778, 0.735754,
 0.448727, 0.798623, 0.496538, 0.469345, 0.56934, 0.645863, 0.655361, 0.253902, 0.592687, 0.697465, 0.707605, 0.783948, 0.517747,
 0.840472, 0.767495, 0.604261, 0.761475, 0.729023, 0.775351, 0.623638, 0.801424, 0.779508, 0.721924, 0.490548, 0.597194, 0.742752,
 0.445035, 0.690542, 0.745576, 0.776739, 0.734227, 0.751988, 0.821853, 0.774277, 0.66181, 0.678296, 0.735088, 0.761344, 0.350727,
 0.806479, 0.551385, 0.797604, 0.646824, 0.746054, 0.799971, 0.756398, 0.673474, 0.527372, 0.607788, 0.479309, 0.715265, 0.776739,
 0.716118, 0.540022, 0.730801, 0.731735, 0.770683, 0.665818, 0.76582, 0.710685, 0.698315, 0.773562, 0.762361, 0.702079, 0.759039,
 0.632465, 0.695871, 0.715032, 0.677692, 0.755062, 0.653998, 0.705673, 0.678296, 0.721457, 0.725095, 0.526859, 0.765221, 0.725254,
 0.738221, 0.646966, 0.704425, 0.747156, 0.732127, 0.769618, 0.759203, 0.645193, 0.713688, 0.513026, 0.759781, 0.584451, 0.727284,
 0.540593, 0.7456, 0.525082, 0.601187, 0.506105, 0.559624, 0.479874, 0.759123, 0.549112, 0.71618, 0.626657, 0.460763, 0.444799,
 0.690834, 0.675929, 0.792665, 0.732819, 0.422552, 0.735804, 0.731593, 0.761962, 0.663192, 0.776345, 0.77104, 0.748718, 0.731693,
 0.723601, 0.726516, 0.633914, 0.532961, 0.426975, 0.544768, 0.428896, 0.659462, 0.541189, 0.720237, 0.721136, 0.755777, 0.74305,
 0.719823, 0.780439, 0.646542, 0.617476, 0.50635, 0.606914, 0.545924, 0.618402, 0.706296, 0.340233, 0.712765, 0.560626, 0.548946}
```

```
gewichtetegesamtzirkularität1 = Total[zirkularitäten1 * flächen1] / Total[flächen1]
```

```
0.343804
```

```
gewichtetegesamtzirkularität2 = Total[zirkularitäten2 * flächen2] / Total[flächen2]
```

```
0.53578
```

```
BeispielZirkularitätKompaktheitSolidität
```

24. Histologische Farbseparierung (Color Deconvolution)

Ruifrok & Johnston 2001

Eine Färbung besitzt eigene Absorptionscharakteristik α. Die Lichtintensität nach Durchlaufen von Licht durch eine Probe ist mit dem Lambert-Beerschen Gesetz definiert:

$$I(x) = I_0 \exp(-\alpha \cdot c \cdot x)$$

mit I_0 als der einfallenden Lichtintensität, α der Absorption der Färbung und c als deren Konzentration.

Der Exponent $\alpha \cdot c \cdot x$ stellt die optische Dichte dar: $d = \alpha \cdot c \cdot x = -\ln\left(\frac{I(x)}{I_0}\right)$

Gehen wir davon aus, daß für ein Kamerabildpixel bei gegebenen drei Farbstoffen diese linear beitragen:

$$\begin{pmatrix} d_1 \\ d_2 \\ d_3 \end{pmatrix} = \begin{pmatrix} \alpha_{r1} x & \alpha_{r2} x & \alpha_{r3} x \\ \alpha_{g1} x & \alpha_{g2} x & \alpha_{g3} x \\ \alpha_{b1} x & \alpha_{b2} x & \alpha_{b3} x \end{pmatrix} \cdot \begin{pmatrix} c_1 \\ c_2 \\ c_3 \end{pmatrix} = \begin{pmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{pmatrix} \cdot \begin{pmatrix} c_1 \\ c_2 \\ c_3 \end{pmatrix} = \boldsymbol{M} \cdot \vec{c}$$

mit $\boldsymbol{M}$ als der Mischmatrix. Da $\vec{d} = \boldsymbol{M} \cdot \vec{c}$ kann also umgekehrt mit $\vec{c} = \boldsymbol{M}^{-1} \cdot \vec{d}$ der Konzentrationsvektor bestimmt werden. Im folgenden wird vorausgesetzt, daß $\vec{c}$ und $\vec{d}$ auf die Länge 1 normiert sind.

Bleibt die Frage nach der Bestimmung von $\boldsymbol{M}$: in reinen Bereichen mit nur je einer Konzentration gilt:

$$\begin{pmatrix} d_{r1} \\ d_{g1} \\ d_{b1} \end{pmatrix} = \begin{pmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{pmatrix} \cdot \begin{pmatrix} c_1 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} m_{11} \\ m_{21} \\ m_{31} \end{pmatrix} \text{ sowie analog } \begin{pmatrix} d_{r2} \\ d_{g2} \\ d_{b2} \end{pmatrix} = \begin{pmatrix} m_{12} \\ m_{22} \\ m_{32} \end{pmatrix} \text{ und } \begin{pmatrix} d_{r3} \\ d_{g3} \\ d_{b3} \end{pmatrix} = \begin{pmatrix} m_{13} \\ m_{23} \\ m_{33} \end{pmatrix}$$

$$\boldsymbol{M} = \begin{pmatrix} d_{r1} & d_{r2} & d_{r3} \\ d_{g1} & d_{g2} & d_{g3} \\ d_{b1} & d_{b2} & d_{b3} \end{pmatrix}$$

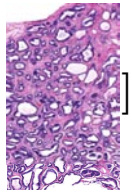
Für eine Anzahl typischer Färbungen sind die $\boldsymbol{M}$ bekannt/publiziert.

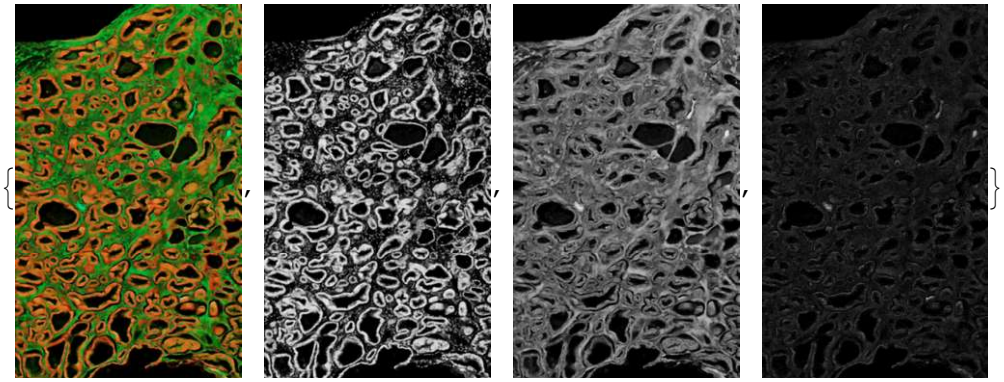
```
Clear[ImageColorDeconvolution];
cDVectors["H&E"] = {{0.644211, 0.716556, 0.266844}, {0.092789, 0.954111, 0.283111}, {0., 0., 0.}};
```

```
cDVectors["H DAB"] = {{0.650, 0.704, 0.286}, {0.268, 0.570, 0.776}, {0., 0., 0.}};
cDVectors["FastRed FastBlue DAB"] = {{0.21393921, 0.85112669, 0.47794022}, {0.74890292, 0.60624161, 0.26731082}, {0.268, 0.570, 0.776}};
cDVectors["Methyl Green DAB"] =
  {(*MG matrix (GL)*){0.98003, 0.144316, 0.133146}, (*DAB matrix*){0.268, 0.570, 0.776}, (*Zero matrix*){0.0, 0.0, 0.0}};
cDVectors["H&E DAB"] = {(*Haem matrix*){0.650, 0.704, 0.286}, (*Eos matrix*)
   {0.072, 0.990, 0.105}, (*DAB matrix*){0.268, 0.570, 0.776}};
cDVectors["H AEC"] = {(*3-amino-9-ethylcarbazole Haem matrix*){0.650, 0.704, 0.286},
   (*AEC matrix*){0.2743, 0.6796, 0.6803}, (*Zero matrix*){0.0, 0.0, 0.0}};
cDVectors["Azan-Mallory"] = {(*Azocarmine and Aniline Blue (AZAN) GL Blue matrix*){0.853033, 0.508733, 0.112656},
   (*GL Red matrix*){0.070933, 0.977311, 0.198067}, (*Orange matrix (not set yet, currently zero)*){0.0, 0.0, 0.0}};
cDVectors["Alcian blue & H"] = {(*GL Alcian Blue matrix*){0.874622, 0.457711, 0.158256},
   (*GL Haematox after PAS matrix*){0.552556, 0.7544, 0.353744}, (*Zero matrix*){0.0, 0.0, 0.0}};
cDVectors["H PAS"] = {(*GL Haem matrix*){0.644211, (*0.650,*)0.716556, (*0.704,*)0.266844},
   (*0.286,*)(*GL PAS matrix*){0.175411, 0.972178, 0.154589}, (*Zero matrix*){0.0, 0.0, 0.0}};
cDVectors["RGB"] = {(*R*){1.0, 0.0, 0.0}, (*G*){0.0, 1.0, 0.0}, (*B*){0.0, 0.0, 1.0}};
cDVectors["CMY"] = {(*C*){0.0, 1.0, 1.0}, (*M*){1.0, 0.0, 1.0}, (*Y*){1.0, 1.0, 0.0}};

ImageColorDeconvolution::usage = "ImageColorDeconvolution[stain,img] applies a color deconvolution of img w.r.t. the staining stain.
    ImageColorDeconvolution[\"Stainings\"] gives a list of all built-in stainings. ImageColorDeconvolution[stainVectors,img]
    uses the vectors form the first argument. The function has two boolean options: LogTransformas and Clip01." ;
ImageColorDeconvolution::invstain =
 "The stain argument of ImageColorDeconvolution[] must be one of `1` instead of `2`.";
ImageColorDeconvolution::invimage =
 "The image argument of ImageColorDeconvolution[] must have three channels.";
ImageColorDeconvolution::invstainvec = "The matrix of staining vectors must be a 3x3 real matrix.";
ImageColorDeconvolution["Stainings"] =
  {"H&E", "H DAB", "FastRed FastBlue DAB", "Methyl Green DAB", "H&E DAB", "Azan-Mallory", "Alcian blue & H", "H PAS", "RGB", "CMY"};

Options[ImageColorDeconvolution] = {LogTransform → True, Clip01 → True};

ImageColorDeconvolution[stain_String, img_?ImageQ, opts : OptionsPattern[ImageColorDeconvolution]] :=
 Module[{dcimg},
 If[! MemberQ[ImageColorDeconvolution["Stainings"], stain],
  Message[ImageColorDeconvolution::invstain,
   ImageColorDeconvolution["Stainings"], stain];
  Return[img];
  ];
 ImageColorDeconvolution[cDVectors[stain], img, opts]
 ];

ImageColorDeconvolution[stainvectors_?MatrixQ, img_?ImageQ, OptionsPattern[ImageColorDeconvolution]] :=
  Module[{dim, cos = Normalize /@ stainvectors, icos, x, imagetype},
   dim = Dimensions[stainvectors];
```

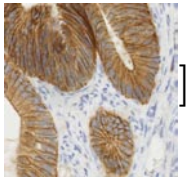
```
    If[! Equal @@ dim || dim[[1]] =!= 3, Message[ImageColorDeconvolution::invstainvec]; Return[img];];
    If[ImageChannels@img =!= 3, Message[ImageColorDeconvolution::invimage]; Return[img];];

    imagetype = ImageType@img;

    If[Norm[cos[[2]], 1] == 0, cos[[2]] = RotateRight@cos[[1]]];
    If[Norm[cos[[3]], 1] == 0, cos[[3]] = Normalize[Sqrt[1.0 - Clip[Plus @@ (# * # & /@ #)]] & /@ Transpose@cos[[{1, 2}, All]]]];
    icos = Inverse@cos;

    If[OptionValue[LogTransform],
     (*mit Log-Hintransformation, Lineartransformation und Wertebereichsbeschraenkung, sowie anschliessender Log-Ruecktransformation*)
     If[OptionValue[Clip01],
       Return@Image[ImageApply[(1. - (256. * Exp[-Log[256.] * #] - 1.) / 255.) &, Image[Clip[ImageData[
              ImageApply[(-Log[(255. * # + 1) / 256.] / Log[256.]).icos &, Image[img, "Real"]], "Real"], {0., 1.}], "Real"]], imagetype];
       ,
       Return@Image[ImageApply[(1. - (256. * Exp[-Log[256.] * #] - 1.) / 255.) &,
            ImageApply[(-Log[(255. * # + 1) / 256.] / Log[256.]).icos &, Image[img, "Real"]]], imagetype];
      ];
     ,
     (*in Anlehnung an die Log-Hin-und-Ruecktransformation umgekehrt linear*)
     If[OptionValue[Clip01],
       Return@Image[Image[Clip[ImageData[ImageApply[(1. - #).icos &, Image[img, "Real"]], "Real"], {0., 1.}], "Real"], imagetype];
       ,
       Return@Image[ImageData[ImageApply[(1. - #).icos &, Image[img, "Real"]], "Real"], imagetype];
      ];
    ];
   ];
```

```
Join[{#}, ColorSeparate@#] &[ImageColorDeconvolution["H&E", ]]
```

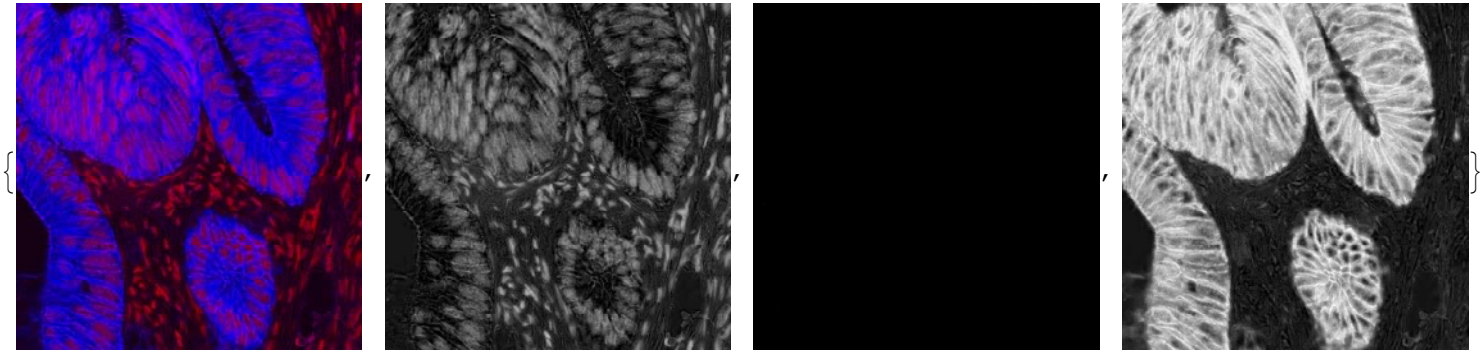




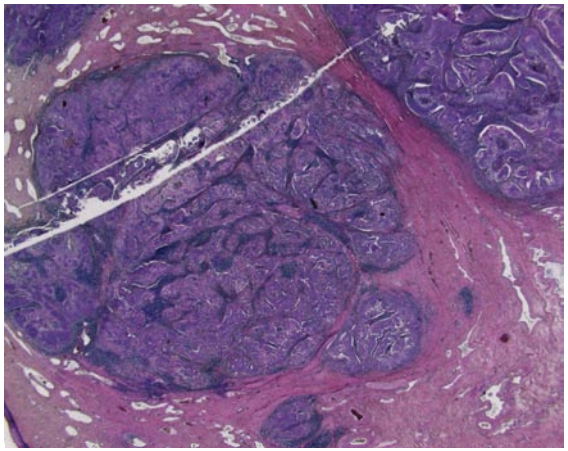
```
Join[{#}, ColorSeparate@#] &[ImageColorDeconvolution["H&E DAB",
```



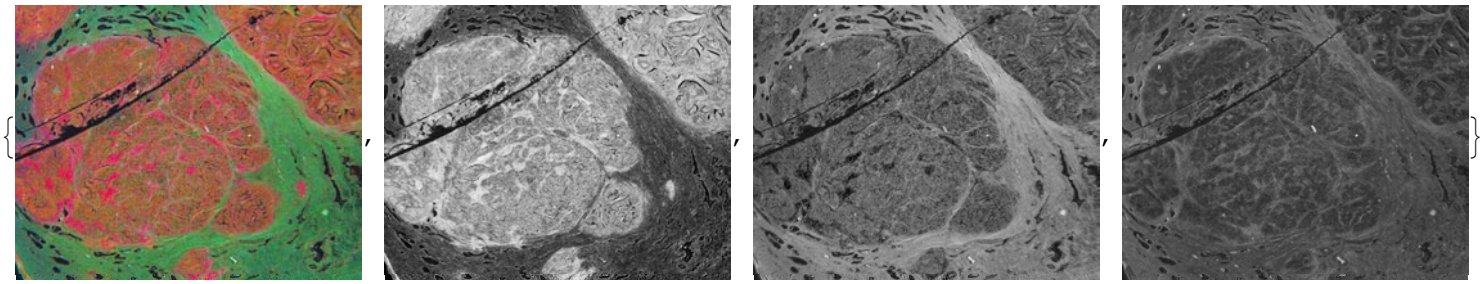
```
]]
```



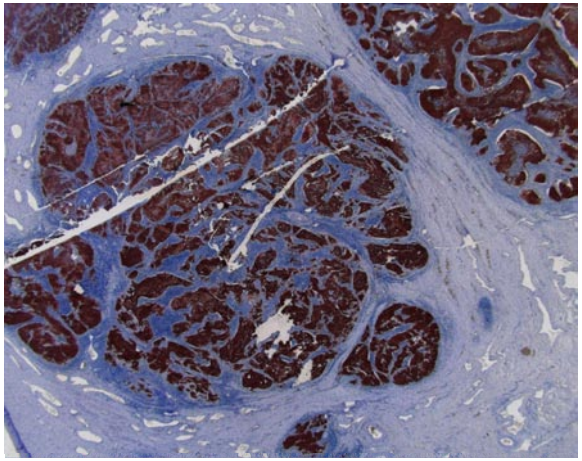
hebild



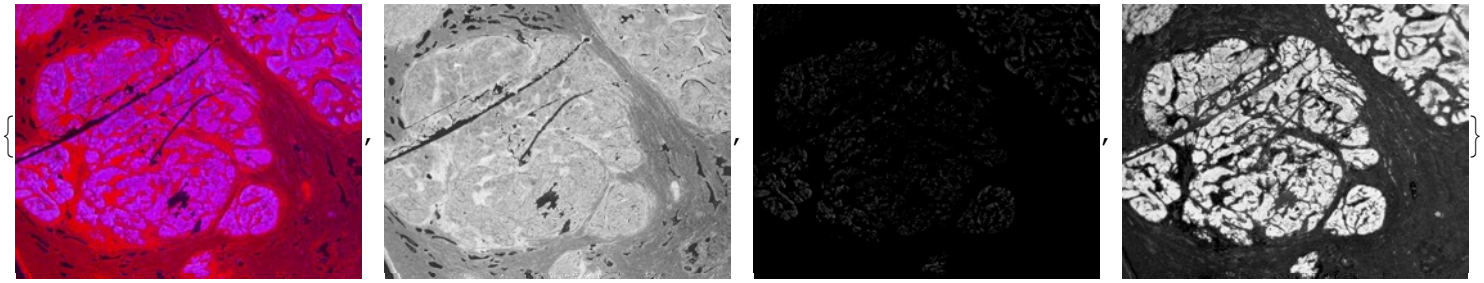
```
Join[{#}, ColorSeparate@#] &[ImageColorDeconvolution["H&E", hebild]]
```



```
p16bild
```



```
Join[{#}, ColorSeparate@#] &[ImageColorDeconvolution["H&E DAB", p16bild]]
```

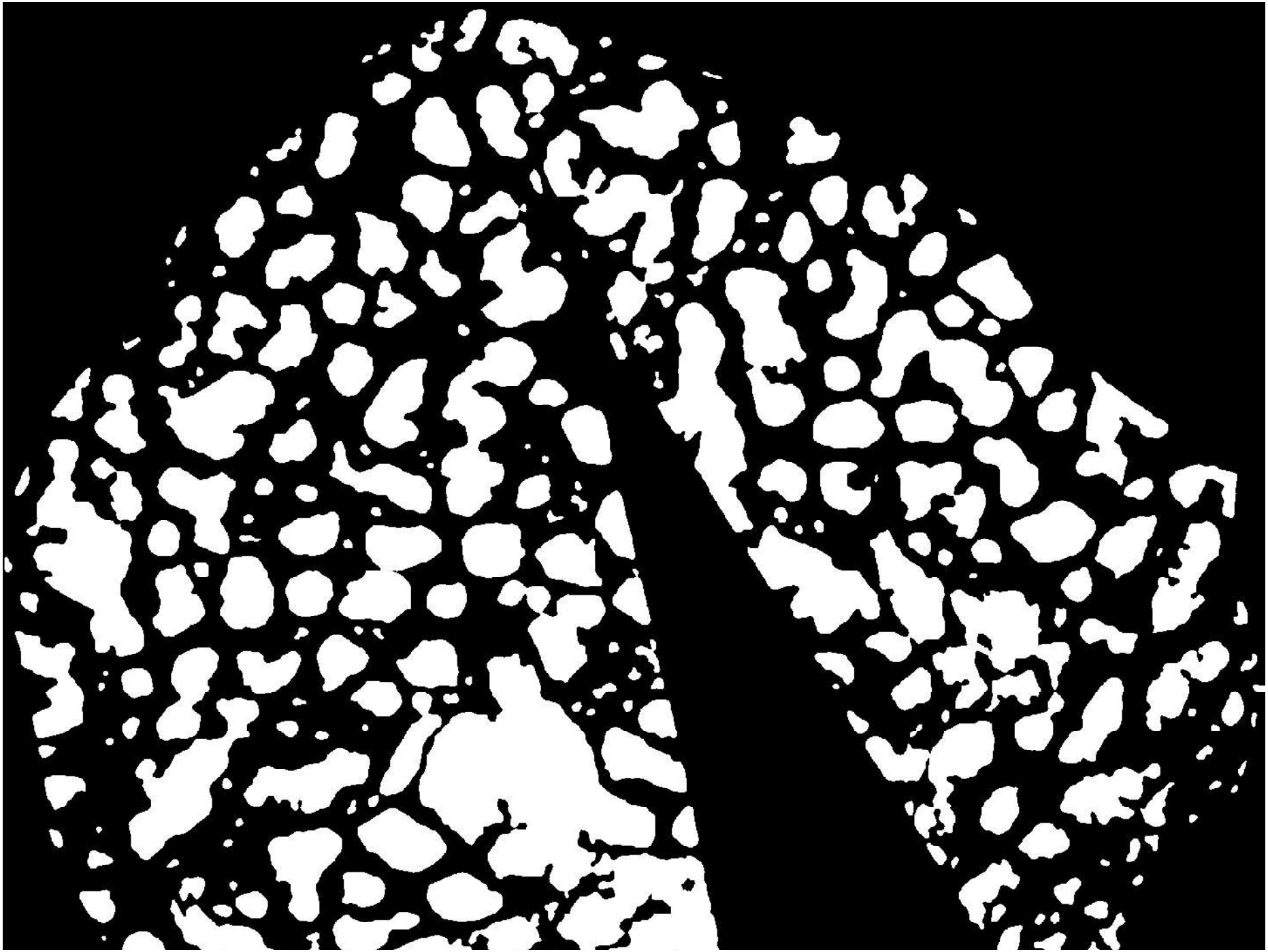


25. Objekttrennung

Eine praktische Anleitung zur Objekttrennung in Binärbildern, zum Teil unter Verwendung von bisher nicht vorgestellten

Funktionen:

testbild2

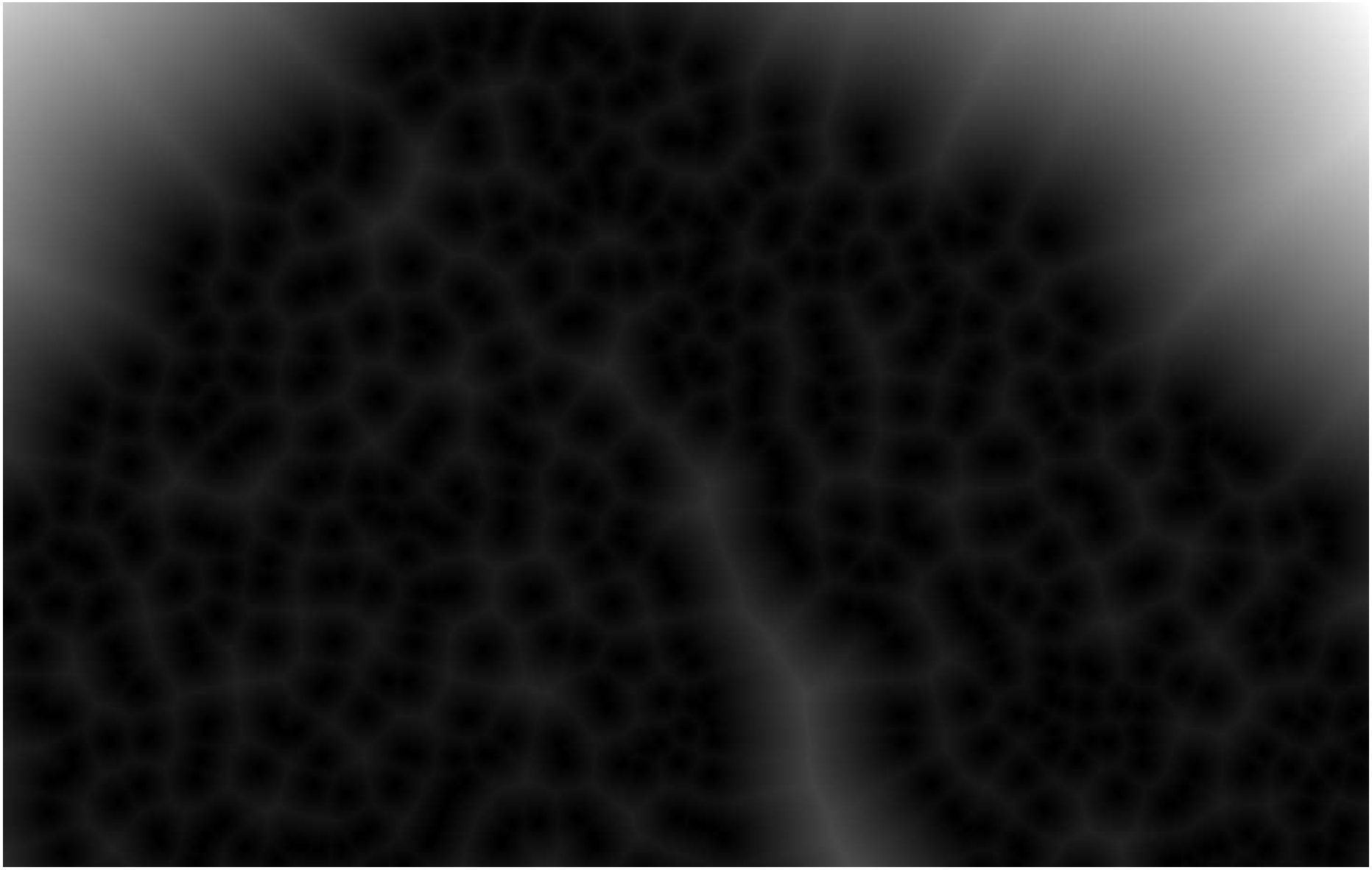


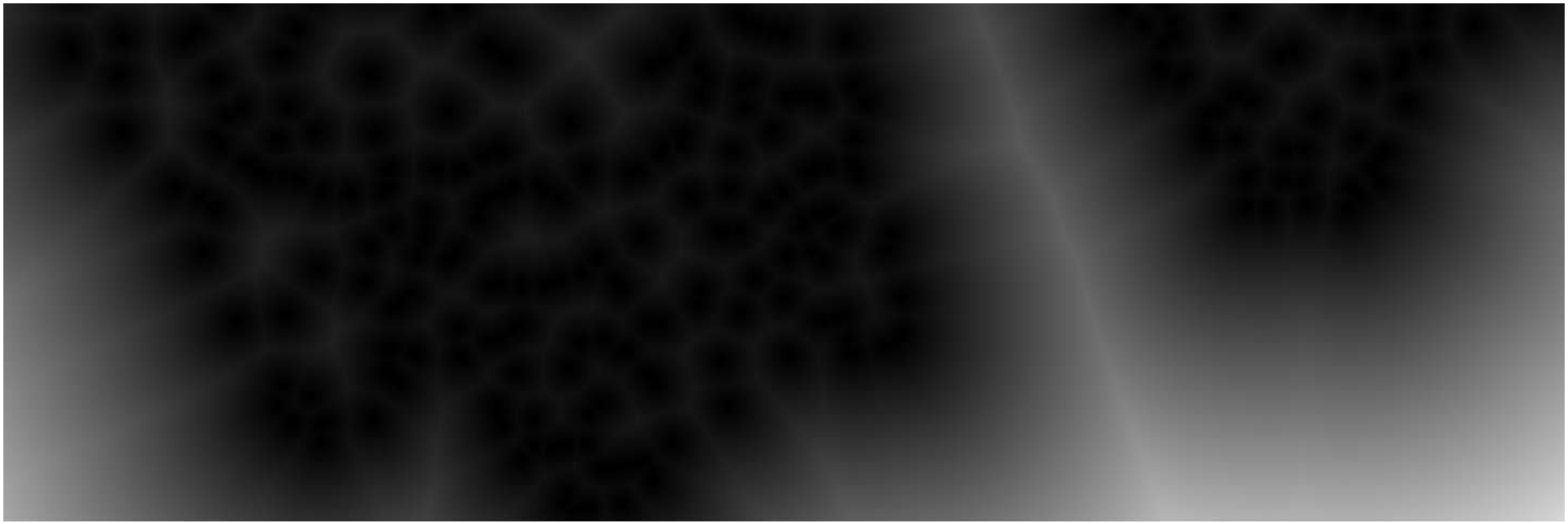
Wir berechnen uns ein Bild, an dem nur kleine schwarze Markierungen dort übrig bleiben wo genügend ausgedehnte Segmente liegen:

```
ColorNegate[
 Binarize[
  ImageMultiply[
     MaxDetect[#], #
    ] &[DistanceTransform[testbild2]], 3(*allow only maxima with distance values > 3*)
 ]
]
```

Wir berechnen ein Distanzbild bezogen auf die schwarzen Markierungen(je heller, desto weiter weg von Schwarz):

```
DistanceTransform[
  ColorNegate[
   Binarize[
    ImageMultiply[
       MaxDetect[#], #
      ] &[DistanceTransform[testbild2]], 3(*allow only maxima with distance values > 3*)
   ]
  ]
 ] // ImageAdjust
```



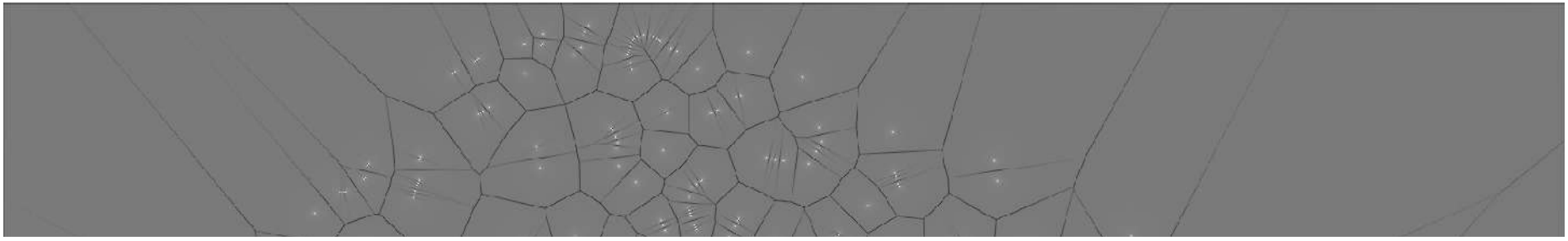


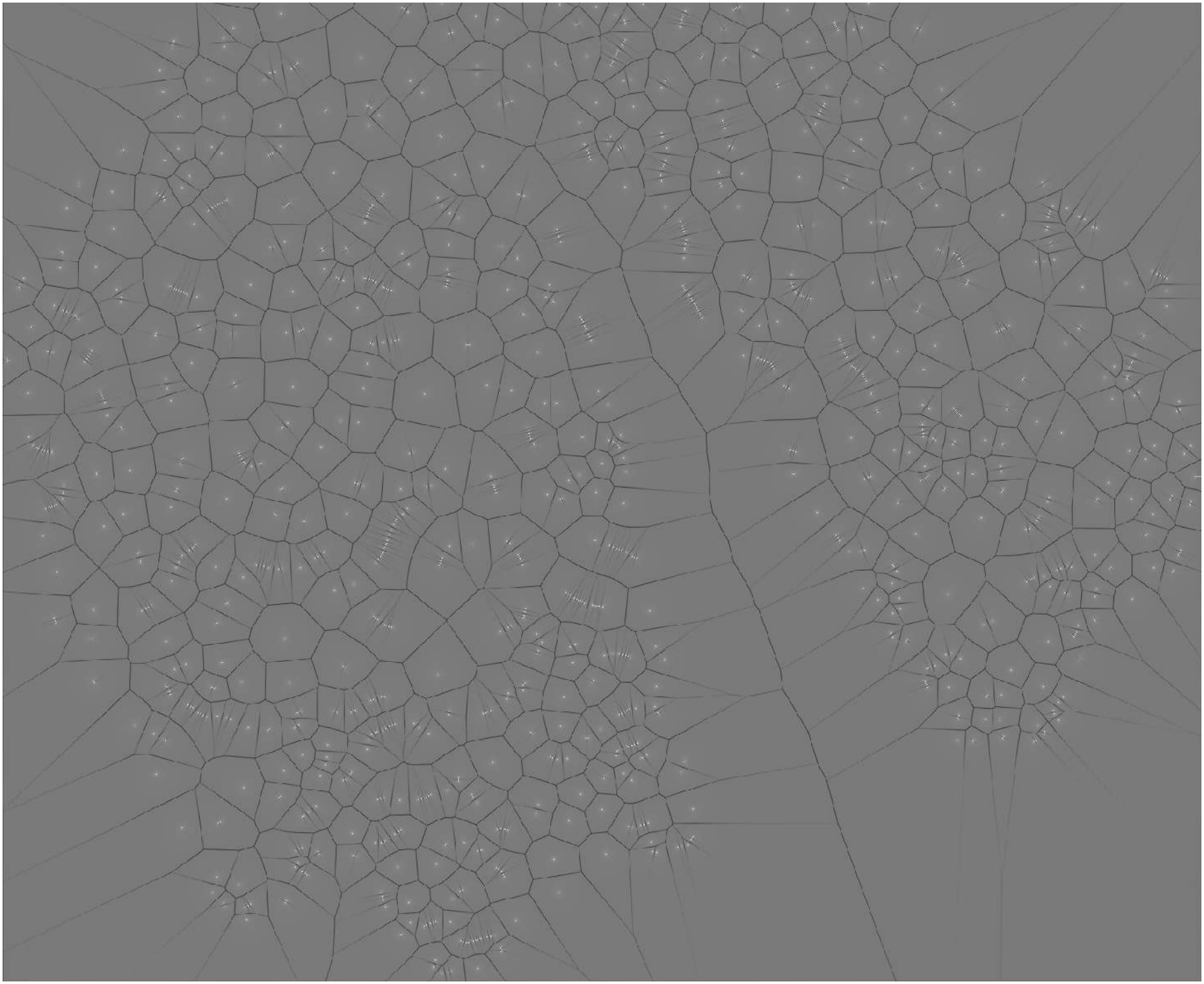
Wir bereiten einen Laplacefilter (2. Ableitung) vor:

```
laplacekern = Transpose[{identität[3]}].{ListConvolve[ableitungsymm, ableitungsymm, 2]} +
    Transpose[{ListConvolve[ableitungsymm, ableitungsymm, 2]}].{identität[3]};
```

Damit können wir die Kammlinien der Distanztransformierten finden, die sich als kleine (d.h. negative) Werte abzeichnen:

```
ImageAdjust@ImageConvolve[DistanceTransform[
    ColorNegate[
     Binarize[
      ImageMultiply[
          MaxDetect[#], #
        ] &[DistanceTransform[testbild2]], 3(*allow only maxima with distance values > 3*)
     ]
    ]
  ], laplacekern]
```



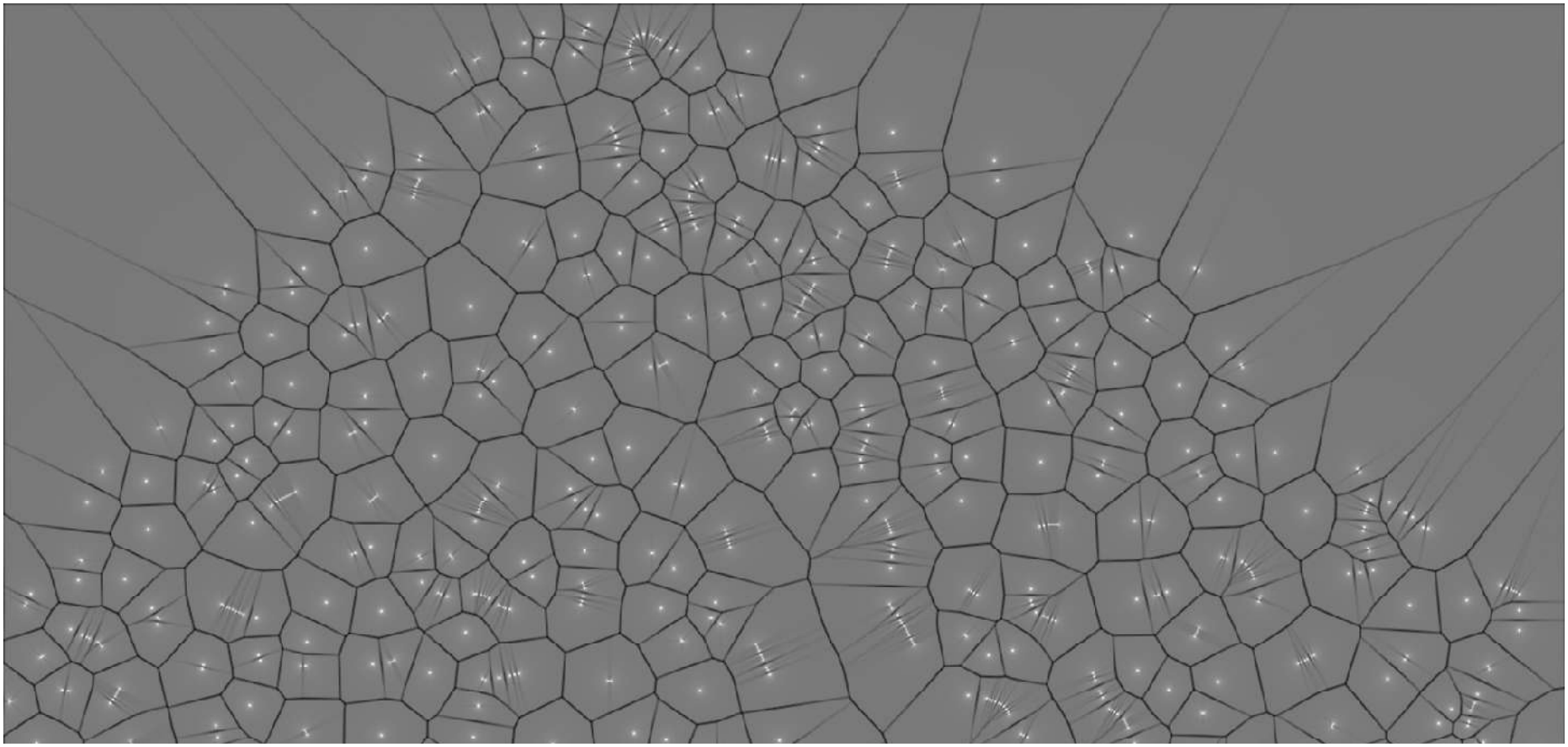


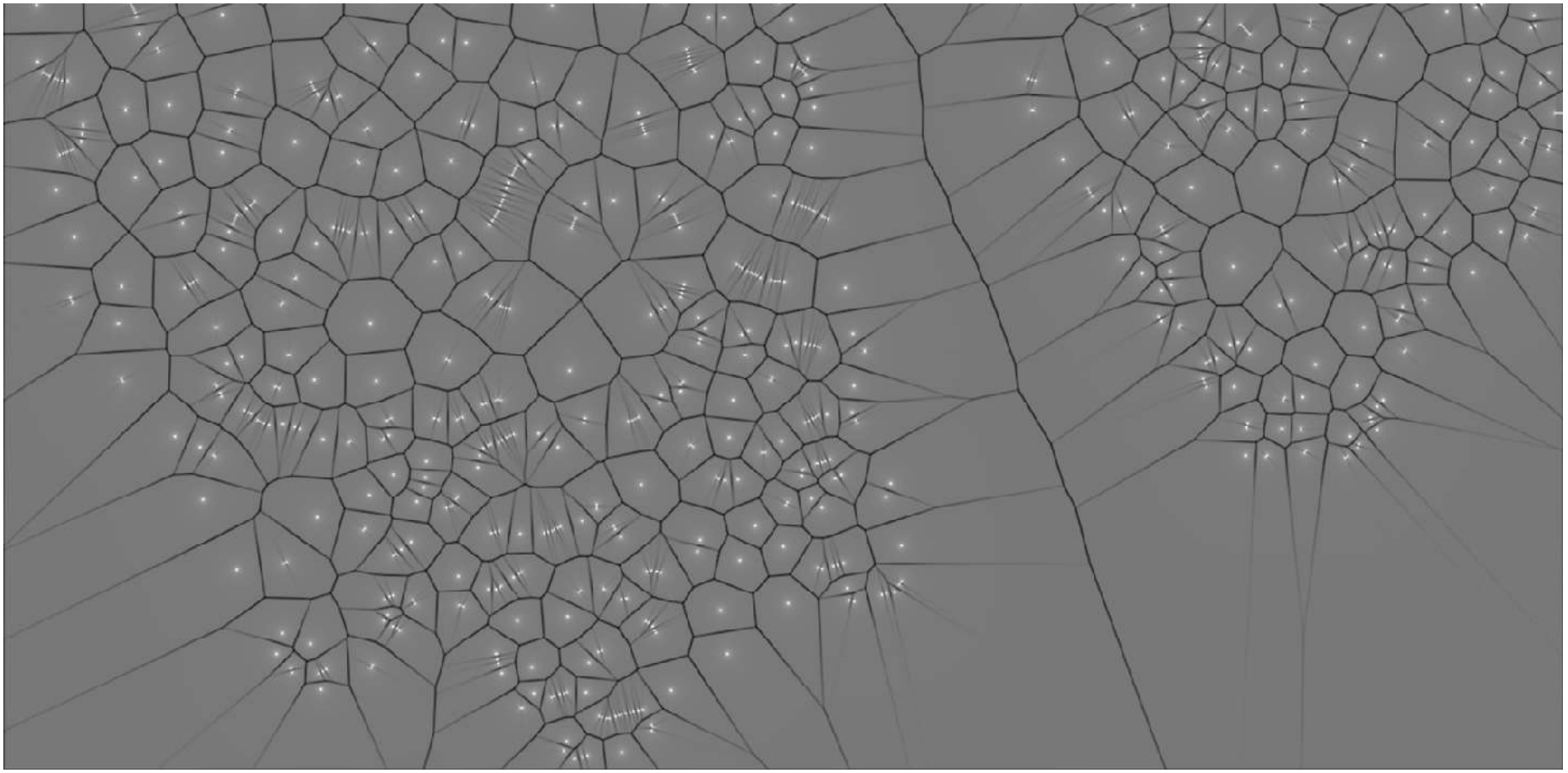
Wir bereiten einen Binomialfilter vor:

```
binomialkern = Transpose[{binom[2]}].{binom[2]};
```

Damit können wir die “Schärfe” der Distanztransformierten vermindern, was sich in breiteren Streifen der Laplacefilterung niederschlägt:

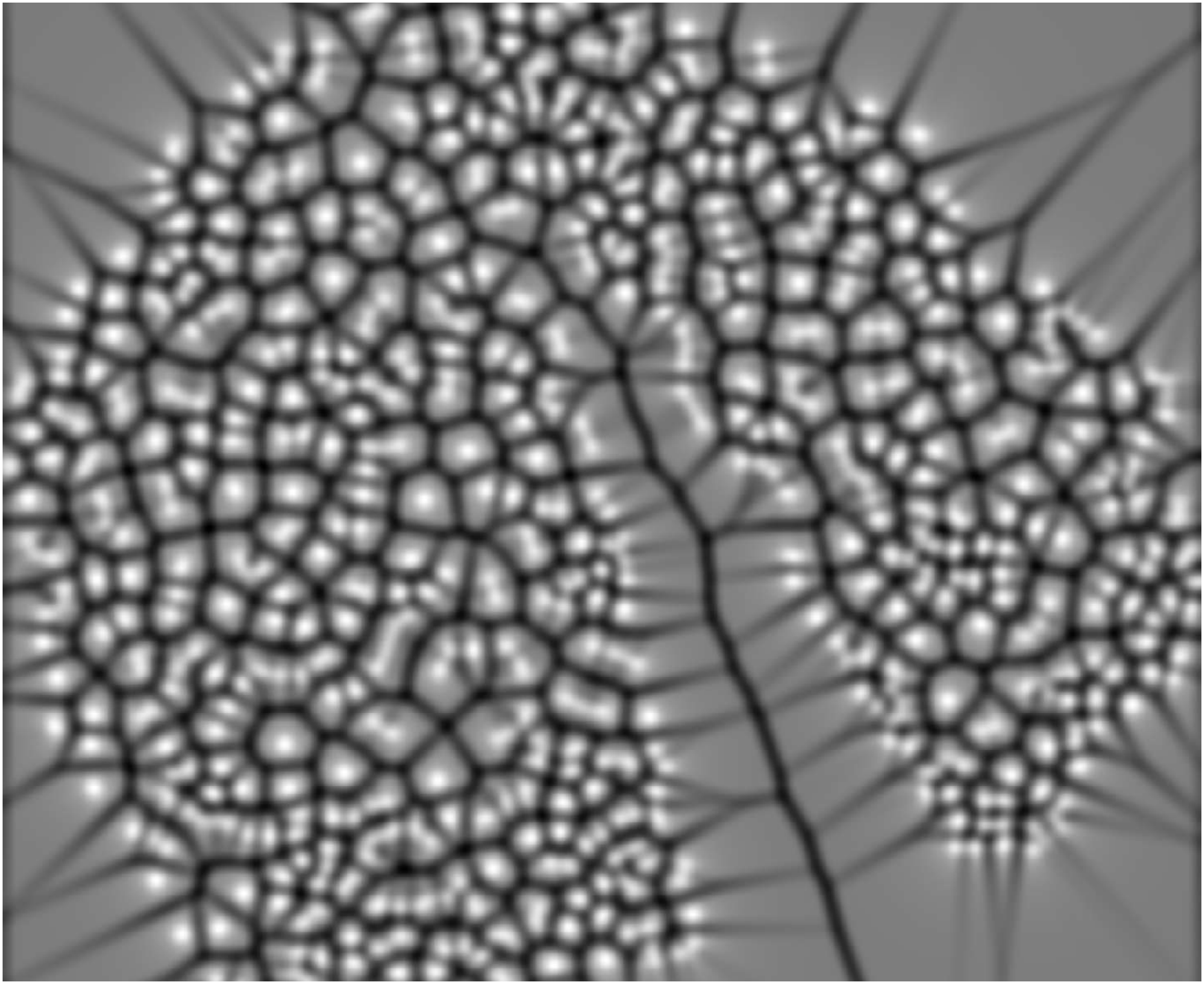
```
ImageAdjust@ImageConvolve[ImageConvolve[DistanceTransform[
    ColorNegate[
     Binarize[
      ImageMultiply[
         MaxDetect[#], #
        ] &[DistanceTransform[testbild2]], 3(*allow only maxima with distance values > 3*)
      ]
     ]
   ], binomialkern], laplacekern]
```

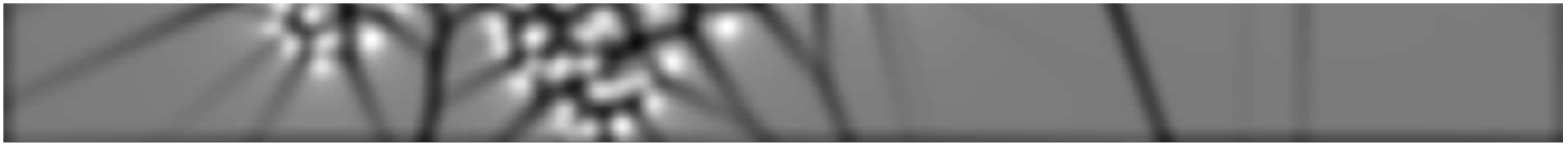




Diese Abfolge von Glättung mit nachfolgender Laplacefilterung wird auch als Laplacian Of Gaussian (LoG) bezeichnet; *Mathematica* hat derlei eingebaut:

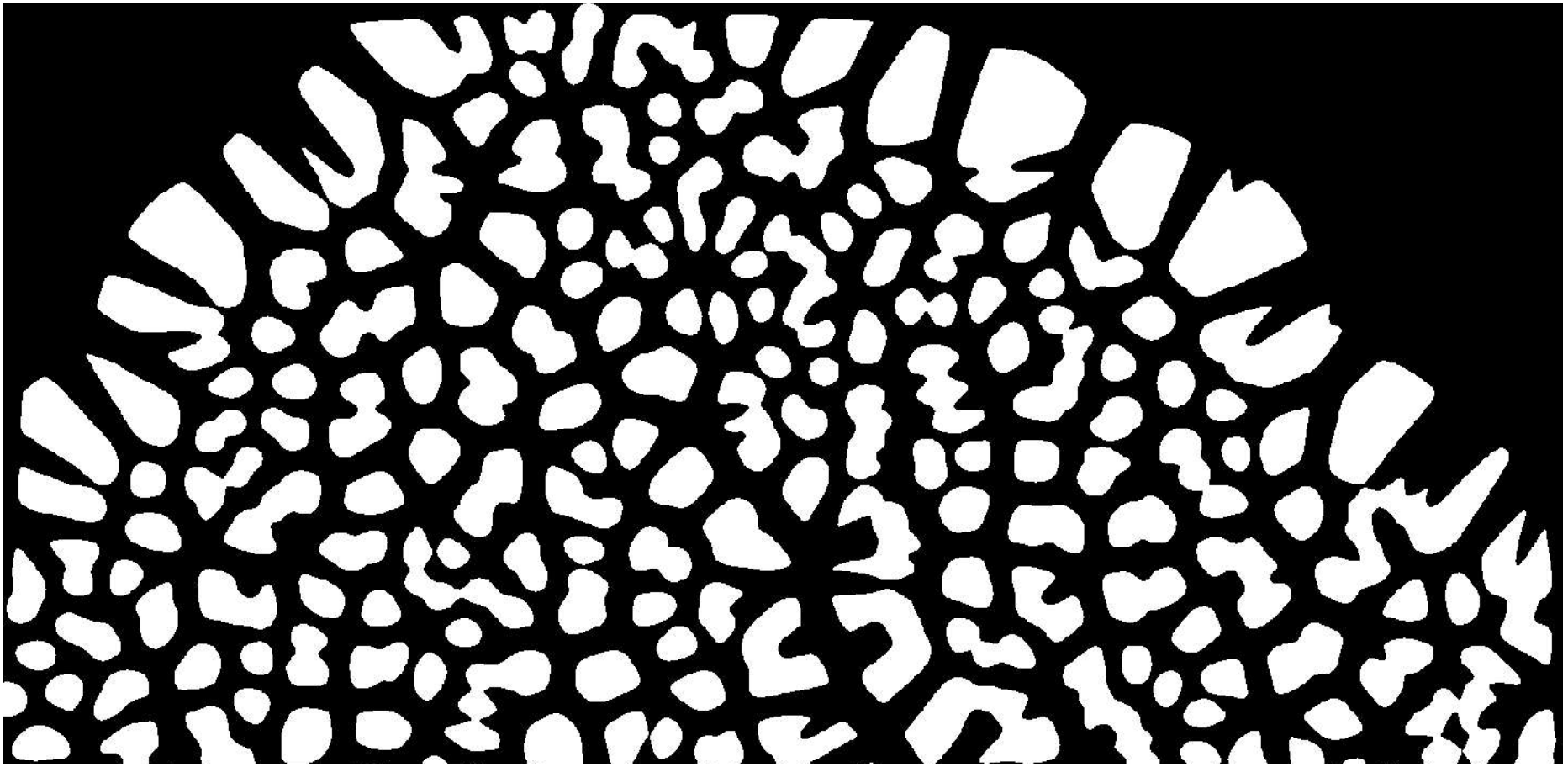
```
LaplacianGaussianFilter[DistanceTransform[
   ColorNegate[
    Binarize[
     ImageMultiply[
        MaxDetect[#], #
       ] &[DistanceTransform[testbild2]], 3(*allow only maxima with distance values > 3*)
     ]
    ]
  ], 20] // ImageAdjust
```

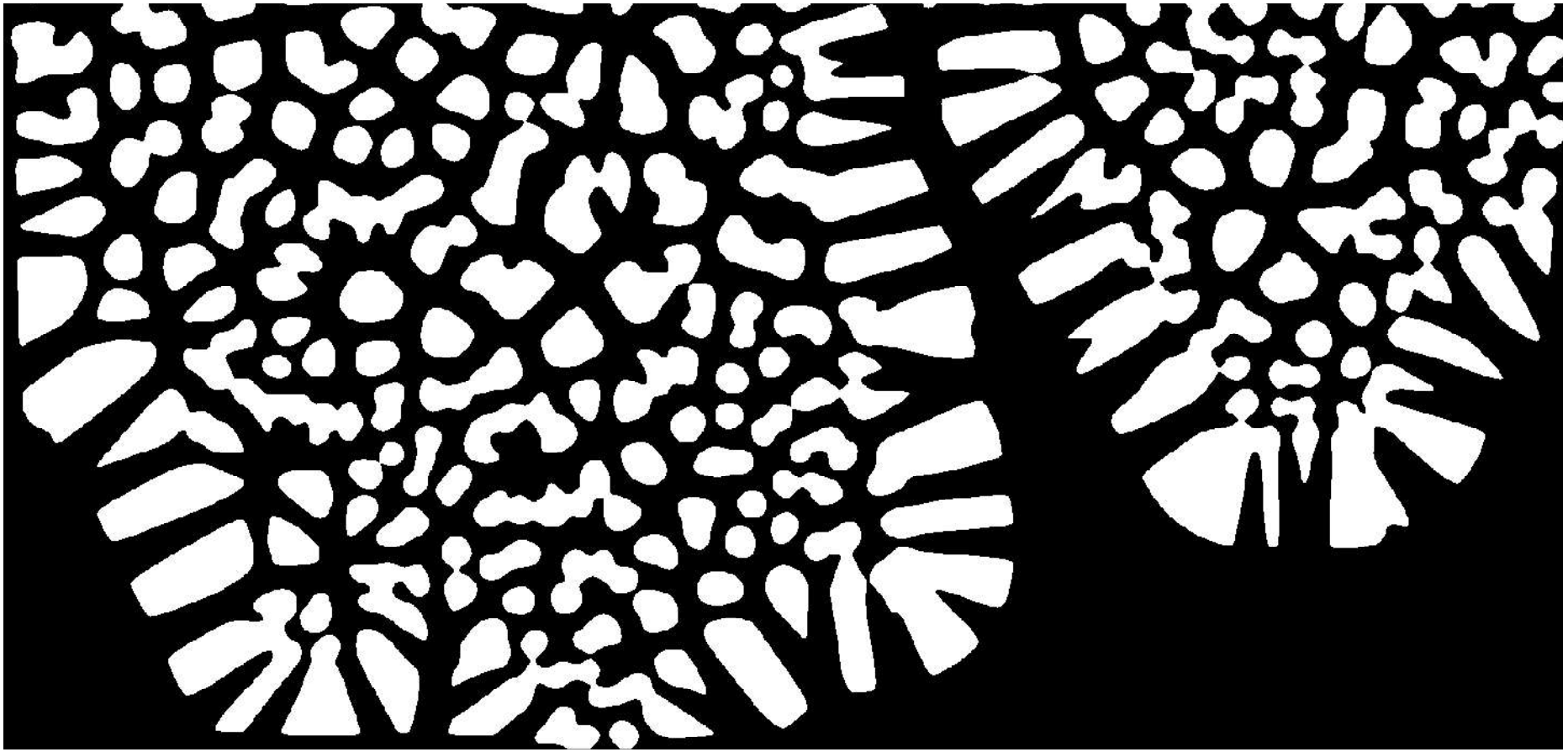




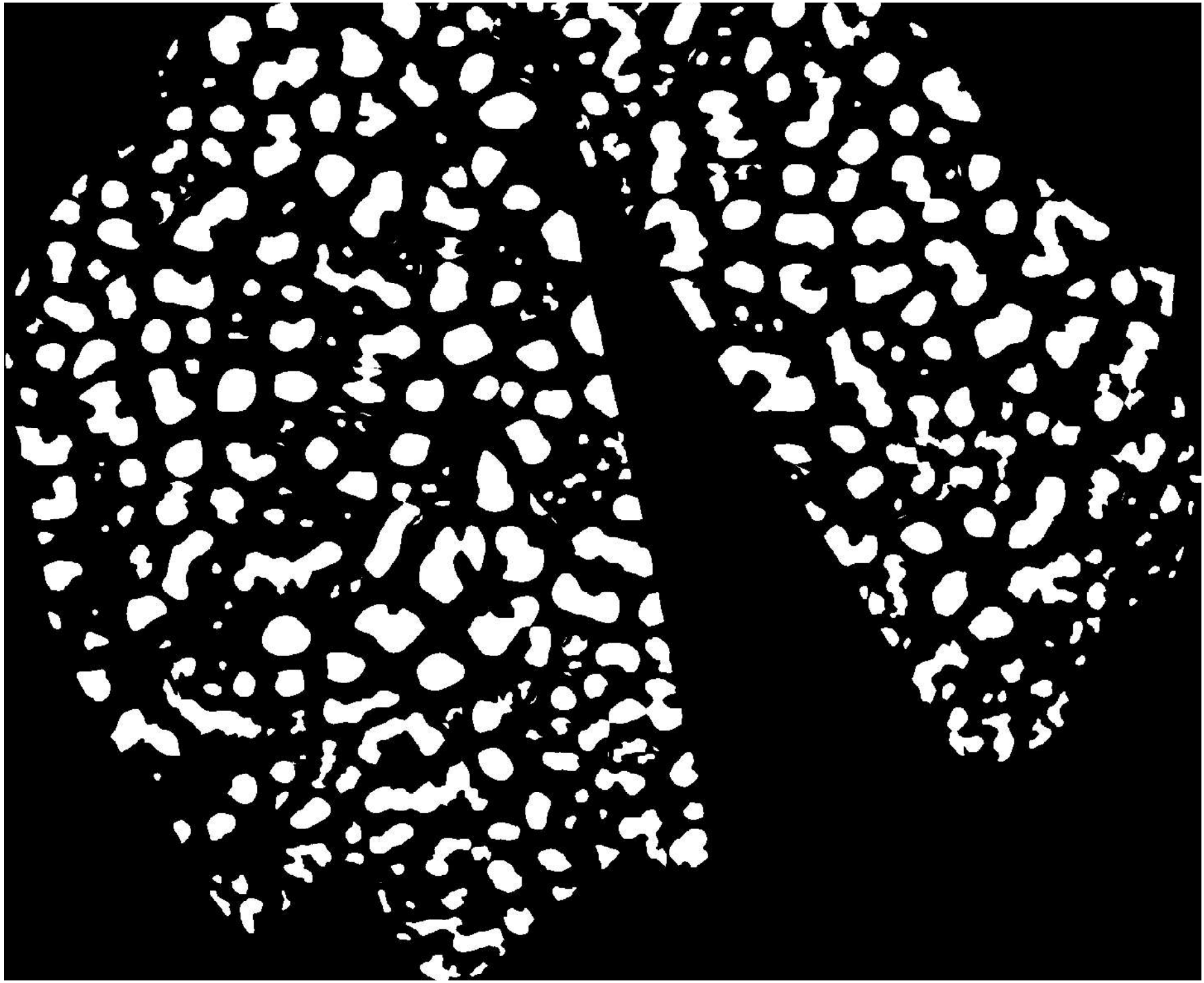
So kann man eine geeignete Maske erstellen, mit der man später das Originalbild zerteilen kann:

```
Binarize[ImageAdjust@LaplacianGaussianFilter[DistanceTransform[
    ColorNegate[
     Binarize[
      ImageMultiply[
         MaxDetect[#], #
        ] &[DistanceTransform[testbild2]], 3(*allow only maxima with distance values > 3*)
      ]
     ]
    ], 20], .5]
```

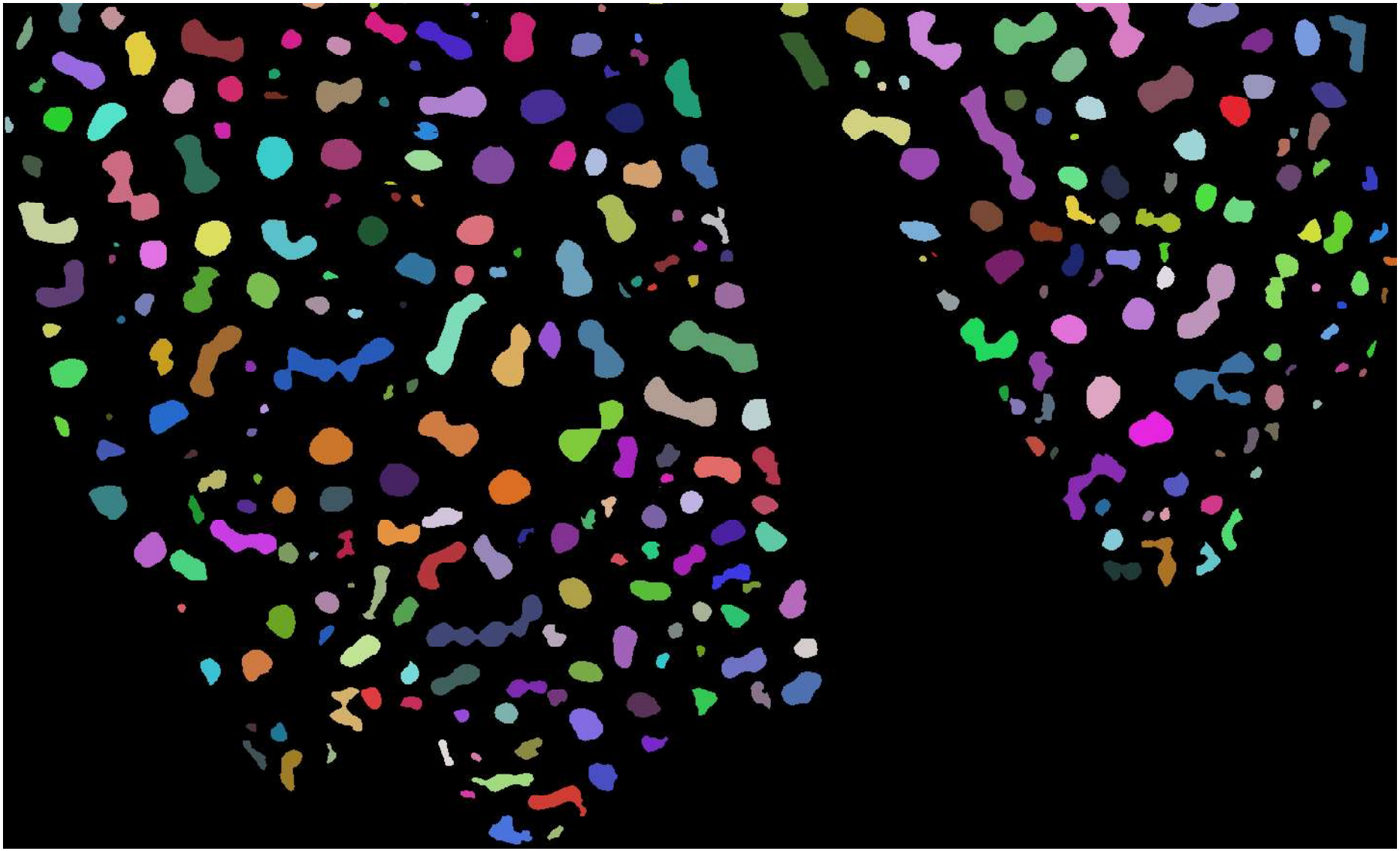




```
ImageMultiply[Binarize[ImageAdjust@LaplacianGaussianFilter[DistanceTransform[
      ColorNegate[
       Binarize[
        ImageMultiply[
           MaxDetect[#], #
          ] &[DistanceTransform[testbild2]], 3(*allow only maxima with distance values > 3*)
       ]
      ]
     ], 20], .5], testbild2]
```



```
Function[{img2d},
  Image[
   Colorize[
    MorphologicalComponents[
     ImageMultiply[
      Binarize[
       LaplacianGaussianFilter[
        DistanceTransform[
         ColorNegate[
          Binarize[
           ImageMultiply[
              MaxDetect[#], #
             ] &[DistanceTransform[img2d]], 3(*allow only maxima with distance values > 3*)
           ]
          ]
         ], 20(*LoG filter kernel size*)
        ]
       ], img2d
      ]
     ]
    ]
   ][testbild2]
```



testbild2