

Supporting information for

Colorimetric Absorbance Mapping and Quantitation on Paper-Based Analytical Devices

Yoshiki Soda, Kye J. Robinson, Thomas J. Cherubini, Eric Bakker*

Department of Inorganic, Analytical Chemistry, University of Geneva, Quai Ernest-Ansermet 30, CH-1211 Geneva, Switzerland.

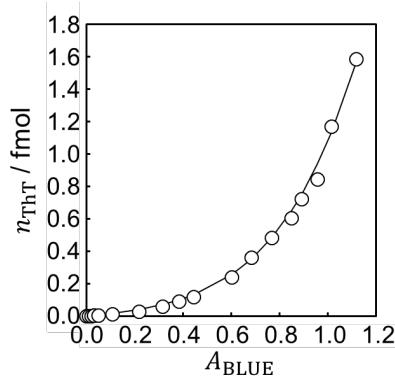


Figure S1. Fitting curve (black line) obtained by fitting experimental data (white circles) to calculate ThT quantity in a pixel based on colorimetric absorbance on the paper substrate. The equation used in this study is shown below.

$$n_{Dye} \cdot a^{-1} = \frac{-8.0819 \times 10^{-6} + 8.1389 \times 10^{-6} e^{7.2770 A_{\text{BLUE}}}}{240.14 + e^{7.2770 A_{\text{BLUE}}}} \quad (\text{S1})$$

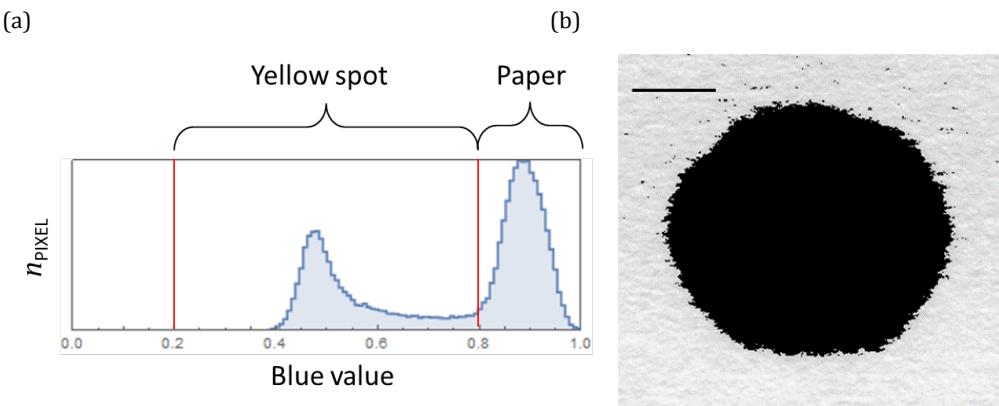


Figure S2. (a) Threshold used in identifying ThT colored part of an image data in blue channel and (b) reconstructed figure after transforming the values of yellow pixel into zero shown with 5 mm scale bar. Computation was carried out using the computing software, Mathematica. Image is prepared by dropping 10 μL of 700 mg/L ThT solution.

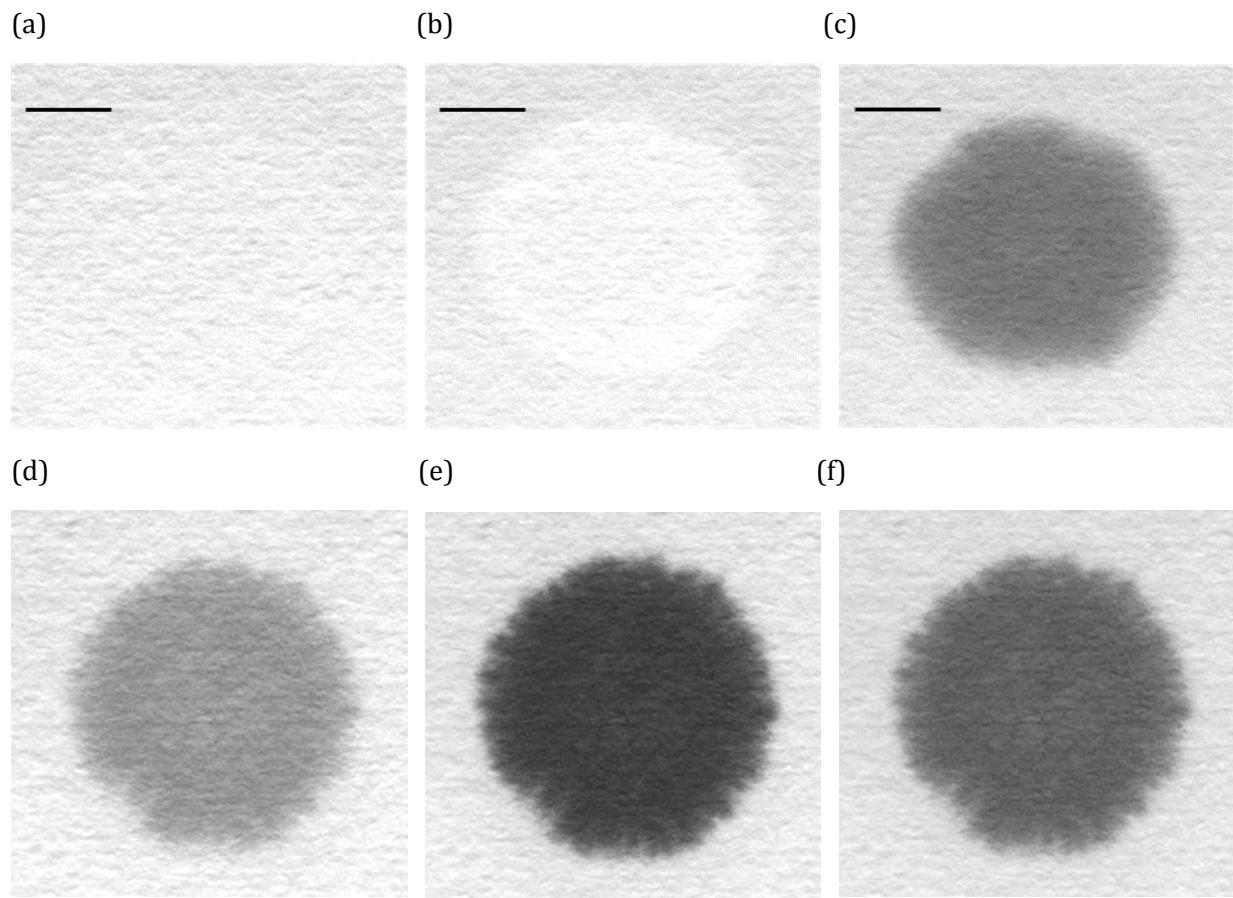


Figure S3. Split image prepared by dropping 10 µL of 700 mg/L ThT solution into (a) red, (b) green, (c) blue channel expressed in gray scale, shown with 5 mm scale bar as in Figure 1 in the main text. Also shown are images prepared with 10 µL of 600 mg/L NR solution into (d) red, (e) green and (f) blue channels.

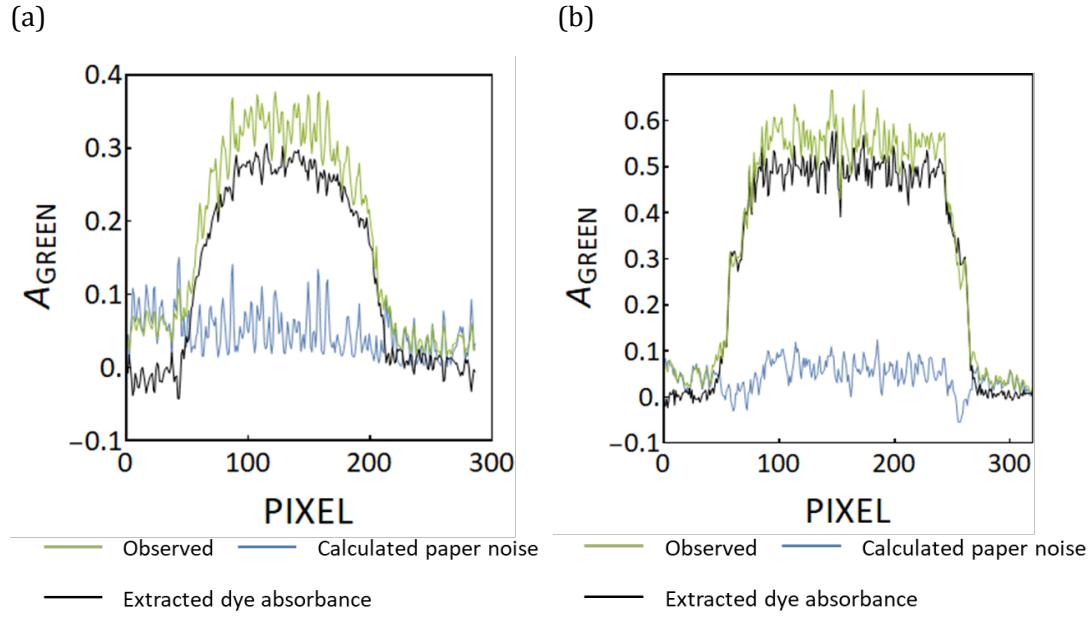


Figure S4. Example of noise extraction taking the 200th horizontal line from the top of picture of Figure S3 prepared by (a) ThT and (b) NR. (a) shows absorbance data of blue channel which is most responsive in the case of ThT while (b) the data of green channel which is the most responsive in the case of NR.

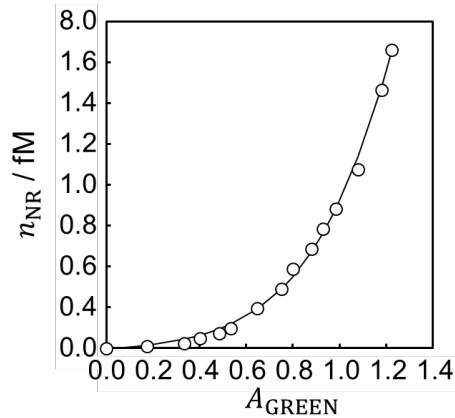


Figure S5. Fitting curve (black line) obtained by spot test (white circle) to calculate NR quantity in a pixel based on colorimetric absorbance on the paper substrate. The equation used in this study is shown as eq S2. Error bars are smaller than the plot symbols.

$$n_{Dye} \cdot a^{-1} = -3.0276 \times 10^{-8} + \frac{8.0355 \times 10^{-6}}{1 + 298.30 \times e^{-3.8252 \times A_{GREEN}}} \quad (S2)$$

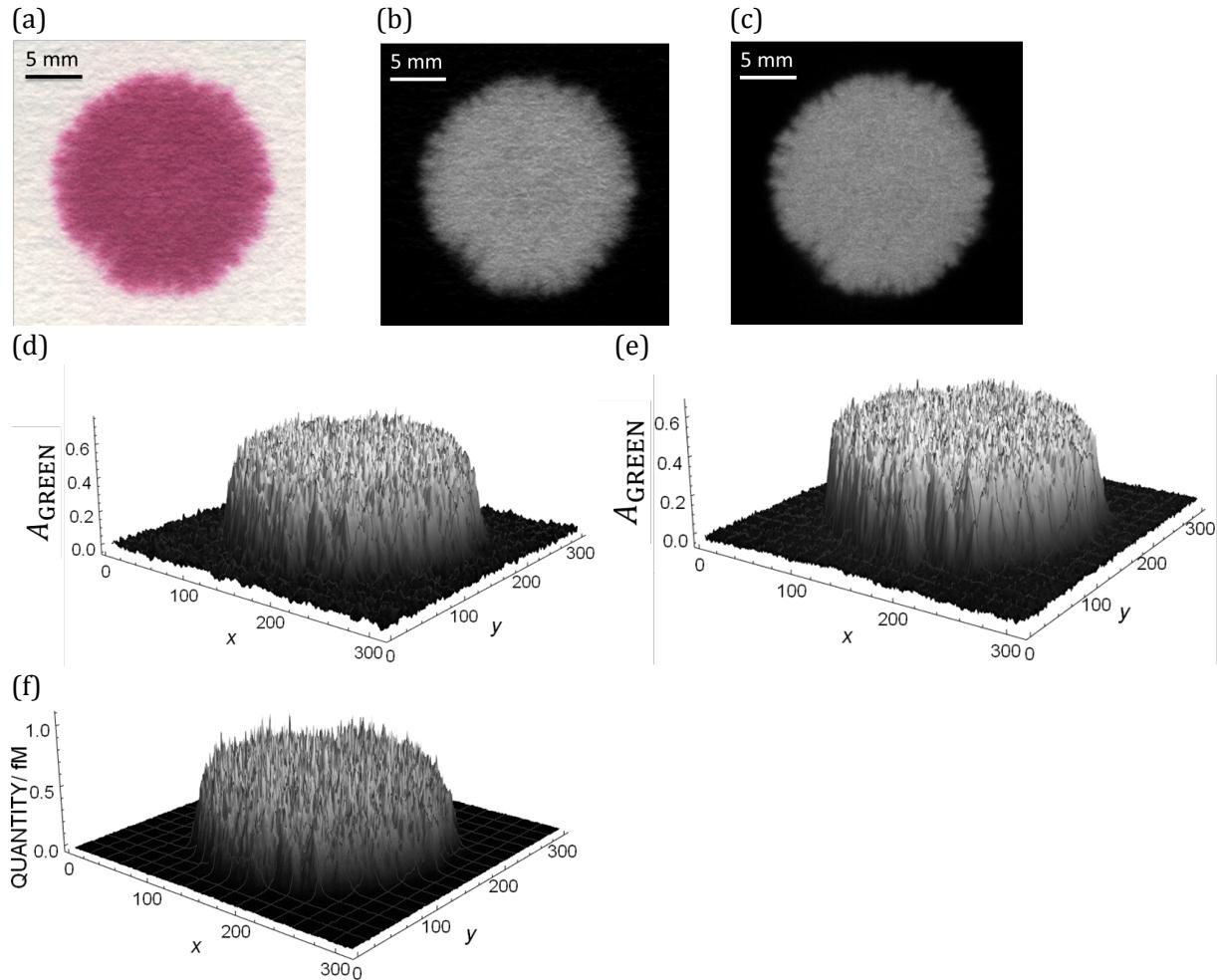
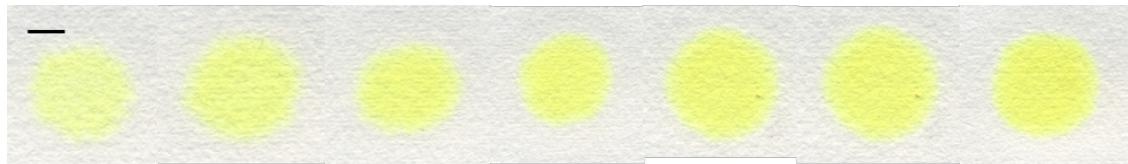


Figure S6. Example of an image data conversion into absorbance mapping and quantification of dye using NR as example dye. (a) shows the original image data prepared by dropping 10 μL of 600 mg/L NR solution on a bare paper that is bottom laminated; (b) represents absorbance mapping figure calculated by the extracted blue value and eq 1 (see eq 6 for the calculation of background blue value) e.g. containing noise from the paper; (c) is also an absorbance mapping figure from the same image data but after elimination of the noise from the paper structure; (d) and (e) are the 3D plots of absorbance values before and after noise cancelling, and (f) is the 3D plot showing the NR quantity profile on the paper substrate calculated based on eq S1 and the noise canceled absorbance data in (e).

(a)



(b)

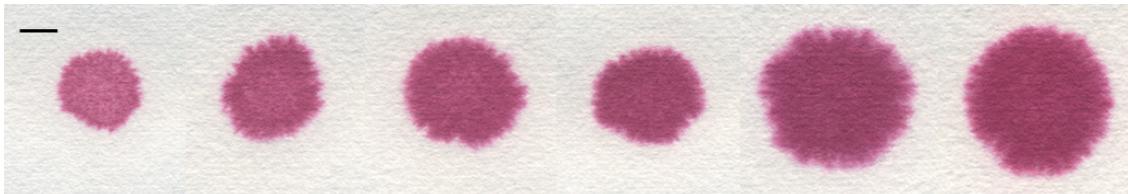


Figure S7. Image data of freely dropped (a) 10 μ L of 100, 200, 300, 400, 500, 600, 700 mg/L ThT (from left to right) and (b) 10 μ L of 100-600 mg/L NR on bare filter paper without hydrophobic barrier. Images were taken by Canoscano 9000f Mark II flatbed scanner, shown with 5 mm scale bar.

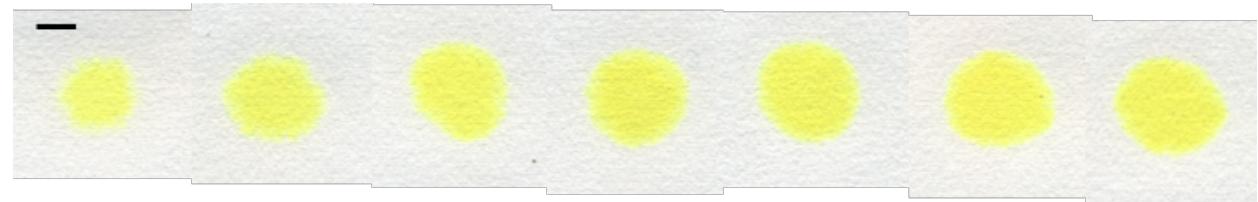


Figure S8. Image data of ThT spots for capillary-based sensor experiment with 1, 2, 3, 4, 5, 6, 10 mM K^+ samples. After 3 minutes incubation for ion exchange between ThT and K^+ , sample solution was discharged on bare filter paper, shown with 5 mm scale bar. Images were taken by Canoscano 9000f Mark II flatbed scanner.



Figure S9. Image data of the same sample as Figure S6 but taken by iPhone XR with the aid of Adobe® Photoshop Lightroom, shown with 5 mm scale bar. See Experimental section for camera settings used.

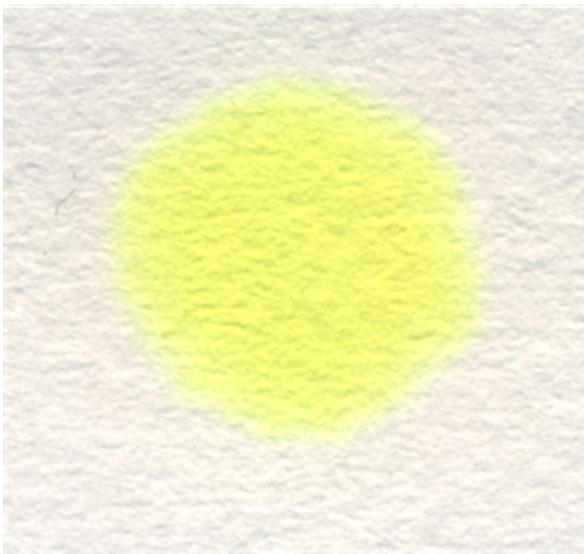
Link to download Mathematica file: <https://gitlab.unige.ch/Kye.Robinson/dye-quantification-on-paper>

Or, see below for the actual code in Mathematica;

```
Clear["Global`*"]
```

To import a picture and split into R,G and B channels

```
imageRGB =  
Import["C:\\\\Users\\\\yoshi\\\\switchdrive\\\\Documents\\\\Bakker's group\\\\Distance-based  
K+ detection\\\\Writing articles\\\\For the third  
paper\\\\Manuscript for LOC\\\\Revision\\\\ThT sample.tif"]
```



```
10 = ImageDimensions[imageRGB]  
{305, 286}
```

```
11 = ColorSeparate[imageRGB]
```



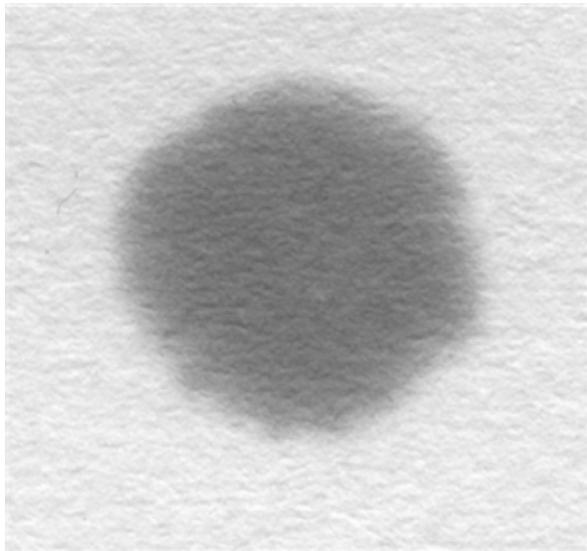
```
imageR = l1[[1]]
```



```
imageG = l1[[2]]
```



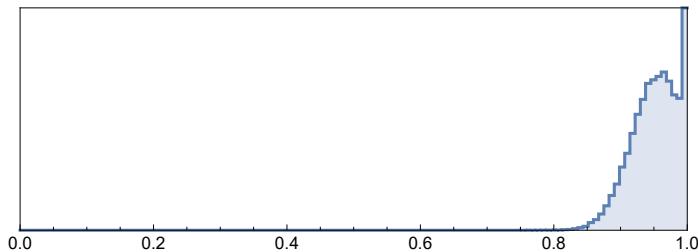
```
imageB = l1[[3]]
```



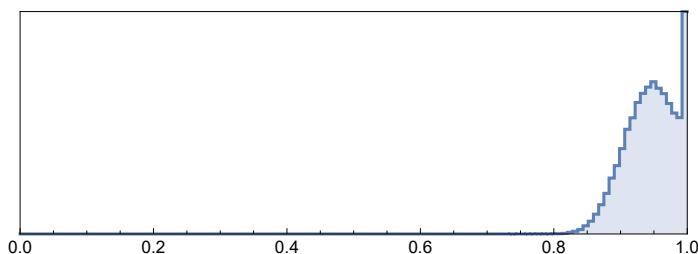
Background calculation

Plot Histogram to decide appropriate threshold

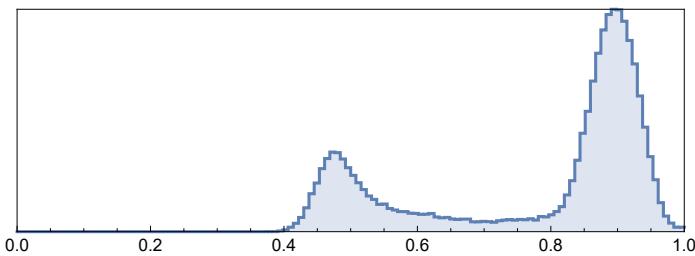
```
ImageHistogram[imageR]
```



```
ImageHistogram[imageG]
```



```
ImageHistogram[imageB]
```



Get positions of colored pixels which are within threshold

```
12 = PixelValuePositions[imageR, 0.45, 0.35]
```

```
{ {164, 279}, {165, 279}, {14, 203}, {15, 203}, {16, 203}, {17, 202},  
{18, 202}, {38, 197}, {38, 196}, {37, 189}, {36, 188}, {33, 185}, {31, 184},  
{32, 184}, {30, 183}, {31, 183}, {29, 182}, {30, 182}, {29, 181} }
```

```
13 = PixelValuePositions[imageG, 0.45, 0.35]
```

```
{ {165, 279}, {95, 261}, {54, 254}, {84, 248}, {14, 203}, {15, 203},  
{16, 203}, {17, 203}, {17, 202}, {18, 202}, {38, 197}, {38, 196}, {37, 189},  
{36, 188}, {34, 186}, {33, 185}, {31, 184}, {32, 184}, {30, 183}, {31, 183},  
{29, 182}, {30, 182}, {29, 181}, {28, 178}, {22, 157}, {303, 122} }
```

```
14 = PixelValuePositions[imageB, 0.45, 0.35]
```

```
{ {164, 279}, {165, 279}, {166, 279}, {182, 273}, {184, 273}, {182, 272}, {234, 272},  
{267, 265}, {158, 264}, {159, 264}, {160, 264}, {161, 264}, {162, 264}, {95, 262},  
{96, 262}, {95, 261}, {96, 261}, {52, 255}, {66, 255}, {111, 255}, {112, 255},  
{54, 254}, {198, 253}, {155, 251}, {156, 251}, {141, 250}, {142, 250}, {143, 250},  
{144, 250}, {147, 250}, {150, 250}, {151, 250}, {152, 250}, {153, 250}, {158, 250},  
{136, 249}, {137, 249}, {138, 249}, {139, 249}, {140, 249}, {141, 249}, {142, 249},  
{144, 249}, {145, 249}, {146, 249}, {147, 249}, {148, 249}, {149, 249}, {150, 249},  
{151, 249}, {152, 249}, {160, 249}, {83, 248}, {84, 248}, {85, 248}, {86, 248},  
{87, 248}, {134, 248}, {135, 248}, {136, 248}, {137, 248}, {138, 248}, {141, 248},  
{142, 248}, {143, 248}, {144, 248}, {145, 248}, {146, 248}, {147, 248}, {148, 248},  
... 28627 ..., {175, 62}, {176, 62}, {177, 62}, {178, 62}, {179, 62}, {135, 61},  
{136, 61}, {137, 61}, {138, 61}, {139, 61}, {140, 61}, {141, 61}, {142, 61}, {143, 61},  
{144, 61}, {145, 61}, {146, 61}, {147, 61}, {148, 61}, {149, 61}, {150, 61},  
{151, 61}, {153, 61}, {154, 61}, {155, 61}, {160, 61}, {161, 61}, {162, 61},  
{163, 61}, {173, 61}, {174, 61}, {175, 61}, {176, 61}, {177, 61}, {178, 61},  
{134, 60}, {135, 60}, {139, 60}, {140, 60}, {141, 60}, {142, 60}, {143, 60},  
{144, 60}, {145, 60}, {146, 60}, {147, 60}, {148, 60}, {149, 60}, {150, 60},  
{151, 60}, {152, 60}, {153, 60}, {154, 60}, {161, 60}, {166, 60}, {169, 60},  
{173, 60}, {174, 60}, {175, 60}, {176, 60}, {141, 59}, {142, 59}, {143, 59},  
{165, 59}, {169, 59}, {170, 59}, {169, 58}, {170, 58}, {171, 58}, {172, 58} }
```

[large output](#)

[show less](#)

[show more](#)

[show all](#)

[set size limit...](#)

Convert pixel values in threshold to 0

```
PixelValue[imageR, 12]
```

```
{0.796078, 0.792157, 0.788235, 0.776471, 0.788235,
0.8, 0.780392, 0.796078, 0.796078, 0.788235, 0.784314, 0.780392,
0.772549, 0.756863, 0.796078, 0.784314, 0.796078, 0.788235, 0.792157}
```

```
PixelValue[imageG, 13]
```

```
{0.8, 0.796078, 0.792157, 0.8, 0.776471, 0.768627, 0.776471, 0.792157, 0.776471, 0.772549,
0.772549, 0.788235, 0.784314, 0.764706, 0.8, 0.768627, 0.760784, 0.741176, 0.780392,
0.784314, 0.780392, 0.788235, 0.780392, 0.792157, 0.796078, 0.796078}
```

```
PixelValue[imageB, 14]
```

```
{0.780392, 0.776471, 0.8, 0.8, 0.8, 0.796078, 0.8, 0.8, 0.792157, 0.788235, 0.788235,
0.788235, 0.792157, 0.796078, 0.8, 0.772549, 0.792157, 0.8, 0.8, 0.788235, 0.788235,
0.796078, 0.8, 0.796078, 0.8, 0.796078, 0.788235, 0.784314, 0.796078, 0.788235, 0.8,
0.784314, 0.784314, 0.788235, 0.792157, 0.8, 0.796078, 0.8, 0.8, 0.788235, 0.772549,
0.792157, ... 28.683 ..., 0.788235, 0.796078, 0.772549, 0.776471, 0.772549, 0.772549,
0.792157, 0.8, 0.792157, 0.788235, 0.764706, 0.721569, 0.713725, 0.74902, 0.788235,
0.768627, 0.760784, 0.784314, 0.780392, 0.756863, 0.760784, 0.780392, 0.788235, 0.780392,
0.8, 0.8, 0.792157, 0.8, 0.796078, 0.8, 0.780392, 0.780392, 0.788235, 0.780392,
0.756863, 0.768627, 0.8, 0.776471, 0.772549, 0.784314, 0.780392, 0.792157, 0.8}
```

large output

[show less](#)

[show more](#)

[show all](#)

[set size limit...](#)

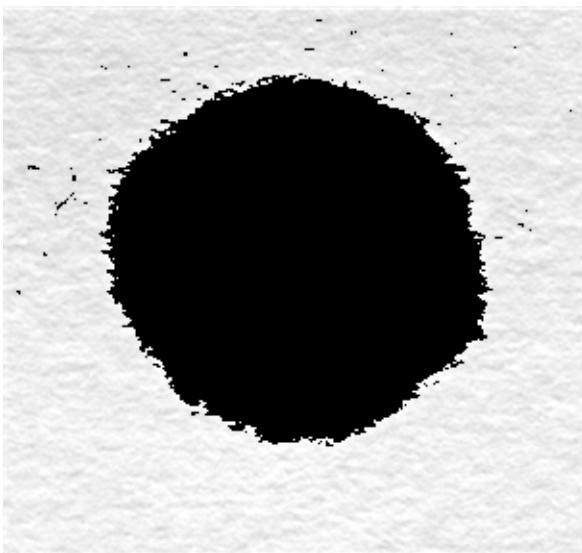
```
image1 = ReplacePixelValue[imageR, 12 → 0]
```



```
image2 = ReplacePixelValue[imageG, 13 → 0]
```



```
image3 = ReplacePixelValue[imageG, 14 → 0]
```



15 = ImageData[image1]

```
{ {0.976471, 0.996078, 0.972549, 0.94902, 0.952941, 0.964706, 0.952941, 0.937255,
 0.945098, 0.956863, 0.964706, 0.972549, 0.980392, 0.968627, 0.968627,
 0.960784, 0.960784, 0.960784, 0.960784, 0.952941, 0.94902, 0.945098, 0.933333,
 0.92549, 0.929412, 0.92549, 0.933333, 0.964706, 0.960784, 0.929412, 0.941176,
 0.945098, 0.933333, ..., 239 ..., 0.92549, 0.94902, 0.952941, 0.960784, 0.964706,
 0.968627, 0.996078, 0.992157, 0.980392, 0.960784, 0.960784, 0.952941,
 0.960784, 0.952941, 0.94902, 0.956863, 0.984314, 0.980392, 0.956863, 0.94902,
 0.956863, 0.972549, 0.964706, 0.964706, 0.976471, 0.968627, 0.964706,
 0.937255, 0.92549, 0.956863, 0.992157, 1., 0.996078}, ..., 284 ..., { ..., 1 ...} }
```

[large output](#)

[show less](#)

[show more](#)

[show all](#)

[set size limit...](#)

16 = ImageData[image2]

```
{ {0.968627, 0.996078, 0.964706, 0.94902, 0.945098, 0.952941, 0.94902,
 0.941176, 0.941176, 0.94902, 0.956863, 0.964706, 0.964706, 0.960784, 0.960784,
 0.952941, 0.952941, 0.94902, 0.945098, 0.945098, 0.945098, 0.933333, 0.929412,
 0.929412, 0.92549, 0.92549, 0.929412, 0.956863, 0.952941, 0.929412, 0.933333,
 0.933333, 0.933333, ..., 239 ..., 0.921569, 0.933333, 0.945098, 0.94902,
 0.94902, 0.956863, 0.984314, 0.980392, 0.964706, 0.94902, 0.94902, 0.952941,
 0.960784, 0.94902, 0.941176, 0.952941, 0.976471, 0.968627, 0.956863, 0.937255,
 0.952941, 0.972549, 0.956863, 0.956863, 0.964706, 0.964706, 0.952941,
 0.937255, 0.929412, 0.956863, 0.984314, 1., 0.984314}, ..., 284 ..., { ..., 1 ...} }
```

[large output](#)

[show less](#)

[show more](#)

[show all](#)

[set size limit...](#)

17 = ImageData[image3]

```
{ {0.968627, 0.996078, 0.964706, 0.94902, 0.945098, 0.952941, 0.94902,
 0.941176, 0.941176, 0.94902, 0.956863, 0.964706, 0.964706, 0.960784, 0.960784,
 0.952941, 0.952941, 0.94902, 0.945098, 0.945098, 0.945098, 0.933333, 0.929412,
 0.929412, 0.92549, 0.92549, 0.929412, 0.956863, 0.952941, 0.929412, 0.933333,
 0.933333, 0.933333, ..., 239 ..., 0.921569, 0.933333, 0.945098, 0.94902,
 0.94902, 0.956863, 0.984314, 0.980392, 0.964706, 0.94902, 0.94902, 0.952941,
 0.960784, 0.94902, 0.941176, 0.952941, 0.976471, 0.968627, 0.956863, 0.937255,
 0.952941, 0.972549, 0.956863, 0.956863, 0.964706, 0.964706, 0.952941,
 0.937255, 0.929412, 0.956863, 0.984314, 1., 0.984314}, ..., 284 ..., { ..., 1 ...} }
```

[large output](#)

[show less](#)

[show more](#)

[show all](#)

[set size limit...](#)

Background Calculation based on Eq6

```
18 = Total[15, {2}]  
{292.518, 302.043, 285.514, 285.471, 284.271, 281.384, 279.616, 277.039, 279.361, 280.58,  
281.945, 283.914, 282.212, 279.592, 278.431, 279.671, 282.11, 283.596, 283.675, 283.91,  
282.525, 280.914, 278.686, 277.902, 279.808, 282., 281.71, 281.416, 281.278, 280.353,  
279.725, 280.918, 284.522, 287.149, 287.576, 286.878, 286.196, 284.463, 282.059,  
281.224, 282.847, 285.906, 286.294, 283.82, 281.439, 281.427, 281.686, 282.745, 285.106,  
288.408, 289.988, 288.051, 284.584, 282.169, 282.161, 283.243, 283.576, 284.251,  
284.816, 284.922, 283.494, 282.412, 283.529, 286.22, 289.039, 290.518, 290.553,  
287.953, 285.627, 285.204, 285.533, 287.463, 290.063, 291.302, 289.894, 289.106,  
288.055, 286.843, 286.529, 288.149, 290.553, 292.094, 290.216, 283.71, 280.98, 282.353,  
285.765, 289.027, 290.078, 289.165, 288.949, 288.271, 286.549, 287.298, 289.361,  
290.392, 291.941, 291.208, 289.965, 287.62, 284.345, 283.016, 284.514, 286.898,  
288.267, 288.843, 288.733, 287.941, 287.424, 288.514, 291.984, 293.882, 293.82, 293.42,  
291.498, 288.643, 287.494, 288.298, 289.035, 288.624, 289.133, 290.482, 290.306,  
289.4, 289.553, 291.176, 292.561, 293.055, 294.325, 294.298, 293.396, 291.89, 290.247,  
289.373, 289.063, 290.125, 290.125, 288.902, 287.769, 287.933, 290.02, 290.518, 291.239,  
292.949, 294.78, 295.251, 292.102, 288.094, 286.529, 289.718, 293.302, 295.157, 295.09,  
293.016, 290.396, 288.761, 287.565, 289.141, 291.675, 293.137, 294.204, 293.824,  
290.816, 288.894, 289.157, 290.176, 291.776, 294.384, 294.275, 292.592, 290.741,  
288.071, 287.471, 288.961, 292.733, 296.58, 297.118, 296.373, 294.69, 293.392, 293.392,  
293.443, 295.141, 297.38, 298.035, 295.082, 290.894, 287.506, 287.471, 291.086,  
293.592, 295.247, 295.616, 296.122, 295.933, 294.682, 294.522, 295.757, 295.733,  
294.722, 292.851, 291.09, 290.831, 291.525, 293.141, 295.416, 297.188, 297.894, 296.773,  
293.996, 292.247, 292.714, 293.784, 295.247, 295.949, 295.431, 295.71, 295.086, 293.71,  
292.698, 293.314, 295.949, 298.573, 299.443, 297.984, 296.259, 295.161, 295.722,  
295.894, 295.196, 295.169, 296.2, 295.918, 294.98, 293.651, 293.02, 295.286, 297.671,  
297.475, 297.373, 296.29, 294.729, 294.184, 294.686, 294.537, 295.878, 297.855, 298.243,  
296.078, 293.537, 293.125, 293.941, 295.094, 296.565, 298.976, 299.384, 297.286,  
295.129, 292.6, 291.047, 292.129, 295.129, 297.278, 297.039, 294.686, 293.055, 293.51,  
294.471, 295.706, 297.929, 298.416, 295.961, 293.443, 292.404, 292.137, 292.247, 293.231,  
295.102, 295.984, 294.294, 290.953, 289.851, 290.675, 291.882, 299.529, 296.094}
```

```
19 = Total[16, {2}]  
{289.737, 299.757, 280.161, 280.129, 279.271, 276.769, 275.525, 273.667, 274.831, 275.843,  
277.318, 278.902, 277.522, 275.208, 274.047, 275.027, 277.416, 278.71, 278.655, 278.824,  
277.8, 276.098, 274.239, 273.588, 275.255, 276.329, 276.722, 276.784, 276.659, 275.918,  
275.318, 276.208, 278.482, 281.878, 282.361, 281.694, 281.18, 279.898, 277.533,  
277.553, 279.467, 282.635, 283.349, 281.49, 279.639, 279.941, 280.404, 281.369, 283.4,  
286.29, 287.533, 285.643, 283.373, 282.137, 282., 282.941, 283.388, 284.455, 285.047,  
285.392, 283.835, 283.106, 284.341, 286.714, 288.945, 290.255, 290.329, 288.282,  
286.514, 286.008, 286.318, 287.741, 289.89, 290.812, 289.8, 289.565, 288.831, 287.686,  
287.584, 288.922, 290.882, 291.957, 290.424, 284.173, 282.643, 284.071, 287.208,  
289.749, 290.431, 289.659, 289.647, 289.278, 287.82, 288.294, 290.063, 290.984,  
292.078, 291.051, 290.039, 288.549, 284.937, 284.463, 285.965, 288.016, 289.133,  
289.514, 289.416, 289.043, 287.875, 289.004, 291.8, 293.616, 293.714, 293.412, 291.69,  
289.133, 288.243, 289.545, 290.212, 289.757, 290.118, 291.176, 291.071, 290.125,  
290.235, 291.663, 292.737, 292.969, 294.051, 292.71, 293.498, 292.569, 291.396,  
290.796, 290.141, 291.114, 291.4, 289.984, 289.043, 289.322, 291.125, 291.447, 291.898,  
293.141, 294.961, 295.769, 292.984, 289.616, 288.122, 290.482, 293.337, 295.165,  
295.161, 293.22, 291.031, 289.859, 288.624, 289.914, 292.094, 293.137, 294.086, 294.02,  
291.439, 289.914, 289.082, 290.839, 292.408, 294.404, 294.392, 292.71, 291.29, 289.094,  
288.412, 289.631, 292.576, 295.882, 296.612, 295.569, 294.365, 293.702, 293.604,  
293.486, 294.729, 296.349, 296.6, 293.875, 290.635, 287.776, 288.18, 291.251, 293.157,  
294.867, 295.388, 295.463, 294.486, 293.208, 293.11, 294.255, 294.365, 293.486, 292.129,  
290.482, 289.859, 290.024, 291.596, 293.525, 294.808, 295.761, 294.576, 291.827,  
290.294, 290.784, 291.898, 293.153, 293.129, 292.588, 292.859, 292.075, 290.282,  
289.408, 290.212, 292.451, 294.843, 295.478, 294.118, 291.78, 290.49, 290.812, 290.78,  
289.62, 290.09, 290.69, 290.035, 289.455, 288.024, 287.255, 289.58, 291.631, 291.494,  
291.498, 290.588, 288.98, 288.18, 289.024, 289.125, 290.722, 292.412, 292.471, 290.361,  
288.086, 287.545, 288.765, 290.118, 291.098, 294.055, 294.133, 291.675, 289.627,  
287.141, 286.035, 287.475, 290.627, 292.376, 291.8, 289.769, 288.471, 288.718, 289.431,  
290.545, 292.894, 293.42, 290.553, 288.373, 287.522, 287.298, 287.631, 288.702,  
290.365, 291.137, 289.502, 286.416, 285.502, 286.357, 287.608, 295.976, 291.925}
```

```

l10 = Total[17, {2}]

{289.737, 299.757, 280.161, 280.129, 279.271, 276.769, 275.525, 272.039, 274.831, 275.843,
277.318, 278.902, 277.522, 273.557, 272.424, 275.027, 277.416, 278.71, 278.655, 278.824,
277.8, 275.275, 270.118, 273.588, 273.612, 275.514, 276.722, 276.784, 276.659, 275.918,
275.318, 272.937, 278.482, 281.043, 282.361, 279.882, 271.976, 264.071, 247.067,
243.839, 238.251, 236.667, 231.435, 222.808, 208.898, 207.161, 204.784, 199.482,
195.251, 198.275, 198.251, 191.337, 187.651, 179.8, 176.196, 176.345, 171.251, 168.62,
167.18, 157.137, 151.533, 151.749, 155.459, 156.145, 152.773, 155.094, 156.094,
149.996, 146.886, 140.702, 141.514, 140.227, 142.561, 141.075, 137.843, 135.878,
133.306, 128.192, 128.973, 131.063, 130.678, 129.922, 123.647, 113.094, 105.643,
109.349, 115.514, 119.059, 118.796, 115.565, 110.816, 112.067, 112.486, 111.478,
114.808, 115.11, 114.706, 111.827, 110.322, 107.612, 104.533, 104.427, 104.973,
107.925, 108.384, 107.024, 108.627, 109.788, 106.173, 107.592, 109.647, 110.071,
108.224, 105.522, 107.055, 108.569, 105.008, 101.914, 98.6863, 98.8784, 100.157,
103.008, 103.863, 102.259, 101.929, 104.169, 103.616, 100.027, 101.031, 102.694,
105.227, 106.522, 104.165, 103.455, 103.996, 105.765, 105.463, 99.8314, 101.502,
102.788, 103.408, 104.075, 104.275, 101.859, 105.067, 106.365, 101.004, 97.2902,
95.9098, 100.776, 107.835, 107.639, 107.078, 104.357, 103.706, 106.6, 107.816,
108.243, 111.153, 108.455, 112.149, 112.416, 113.353, 115.341, 114.086, 115.024,
111.769, 117.761, 117.075, 115.988, 115.498, 113.325, 115.016, 116.808, 124.824,
130.055, 130.188, 127.867, 131.482, 130.596, 133.592, 136.388, 139.475, 143.243,
143.435, 141.922, 139.643, 141.922, 144.988, 149.506, 154.996, 158.722, 160.329,
164.196, 165.706, 162.318, 163.62, 166.329, 165.514, 167.851, 170.741, 168.976,
167.855, 168.224, 173.251, 182.698, 185.78, 187.729, 191.792, 192.349, 193.149,
190.255, 195.31, 206.643, 210.157, 211.427, 222.498, 219.941, 217.718, 226.769,
230.282, 234.165, 243.204, 246.706, 253.604, 262.667, 266.416, 285.106, 287.039,
289.62, 290.09, 290.69, 290.035, 289.455, 288.024, 287.255, 289.58, 291.631, 291.494,
291.498, 290.588, 288.98, 288.18, 289.024, 289.125, 290.722, 292.412, 292.471, 290.361,
288.086, 287.545, 288.765, 290.118, 291.098, 294.055, 294.133, 291.675, 289.627,
287.141, 286.035, 287.475, 290.627, 292.376, 291.8, 289.769, 288.471, 288.718, 289.431,
290.545, 292.894, 293.42, 290.553, 288.373, 287.522, 287.298, 287.631, 288.702,
290.365, 291.137, 289.502, 286.416, 285.502, 286.357, 287.608, 295.976, 291.925}

```

$R_0 = \text{Sum}[18[[n]], \{n, 1, \text{Length}[18]\}] / (\text{Total}[1] * \text{Total}[2] - \text{Length}[12])$

0.951998

$G_0 = \text{Sum}[19[[n]], \{n, 1, \text{Length}[19]\}] / (\text{Total}[1] * \text{Total}[2] - \text{Length}[13])$

0.945948

$B_0 = \text{Sum}[l10[[n]], \{n, 1, \text{Length}[l10]\}] / (\text{Total}[1] * \text{Total}[2] - \text{Length}[14])$

0.932757

Calculation of absorbance in all channels to find the channel with highest response and two background

channels

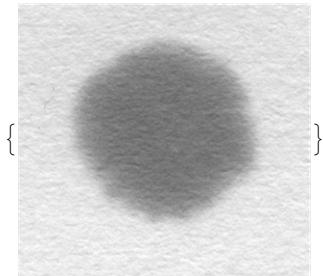
```
l11 = {-Log[10, imageR/R0], -Log[10, imageG/G0], -Log[10, imageB/B0]}
```



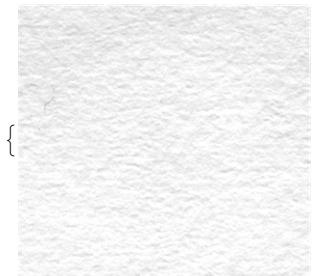
```
l12 = ImageMeasurements[l11, "Mean"]
```

```
{0.000302307, 0.000363094, 0.090365}
```

```
r2 = l1[[Part[Position[l12, Max[l12]], 1]]]
```



```
r5 = l1[[Part[Position[l12, Min[l12]], 1]]]
```



```
r6 = l1[[Part[Position[l12, Median[l12]], 1]]]
```



```
r2D = ImageData[r2[[1]]]
```

{ ... 1 ... }

[large output](#)

[show less](#)

[show more](#)

[show all](#)

[set size limit...](#)

```
bgchannel1 = r5[[1]];
```

```
bgchannel1D = ImageData[bgchannel1]
```

{ {0.976471, 0.996078, 0.972549, 0.94902, 0.952941, 0.964706, 0.952941, 0.937255, 0.945098, 0.956863, 0.964706, 0.972549, 0.980392, 0.968627, 0.968627, 0.960784, 0.960784, 0.960784, 0.952941, 0.94902, 0.945098, 0.933333, 0.92549, 0.929412, 0.92549, 0.933333, 0.964706, 0.960784, 0.929412, 0.941176, 0.945098, 0.933333, ... 239 ..., 0.92549, 0.94902, 0.952941, 0.960784, 0.964706, 0.968627, 0.996078, 0.992157, 0.980392, 0.960784, 0.960784, 0.952941, 0.960784, 0.952941, 0.94902, 0.956863, 0.972549, 0.964706, 0.964706, 0.976471, 0.968627, 0.964706, 0.937255, 0.92549, 0.956863, 0.992157, 1., 0.996078}, ... 284 ..., { ... 1 ... } }

[large output](#)

[show less](#)

[show more](#)

[show all](#)

[set size limit...](#)

```
bgchannel12 = r6[[1]]
```

```
bgchannel12D = ImageData[bgchannel12]
```

{ {0.968627, 0.996078, 0.964706, 0.94902, 0.945098, 0.952941, 0.94902, 0.941176, 0.941176, 0.94902, 0.956863, 0.964706, 0.964706, 0.960784, 0.960784, 0.952941, 0.952941, 0.94902, 0.945098, 0.945098, 0.945098, 0.933333, 0.929412, 0.929412, 0.92549, 0.92549, 0.929412, 0.956863, 0.952941, 0.929412, 0.933333, 0.933333, 0.933333, 0.933333, 0.921569, 0.933333, 0.945098, 0.94902, 0.94902, 0.956863, 0.984314, 0.980392, 0.964706, 0.94902, 0.94902, 0.952941, 0.960784, 0.94902, 0.941176, 0.952941, 0.976471, 0.968627, 0.956863, 0.937255, 0.952941, 0.972549, 0.956863, 0.956863, 0.964706, 0.964706, 0.952941, 0.937255, 0.929412, 0.956863, 0.984314, 1., 0.984314}, ... 284 ..., { ... 1 ... } }

[large output](#)

[show less](#)

[show more](#)

[show all](#)

[set size limit...](#)

```
l2 = ImageData[r2[[1]]]
```

{ ... 1 ... }

[large output](#)

[show less](#)

[show more](#)

[show all](#)

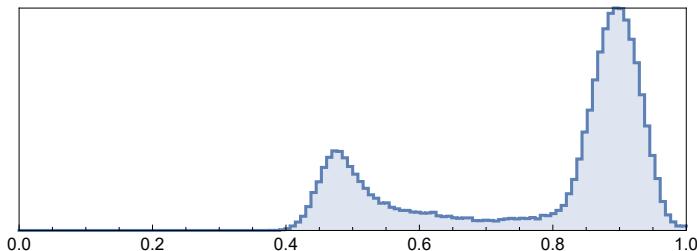
[set size limit...](#)

Background Calculation and Subtraction

Solving Systems of Equations using Simultaneous Equations

Using the same histogram as above

```
ImageHistogram[r2[[1]]]
```



Get positions of colored pixels which are within threshold

```
13 = PixelValuePositions[r2[[1]], 0.45, 0.35]
```

{164, 279}, {165, 279}, {166, 279}, {182, 273}, {184, 273}, {182, 272}, {234, 272},
{267, 265}, {158, 264}, {159, 264}, {160, 264}, {161, 264}, {162, 264}, {95, 262},
{96, 262}, {95, 261}, {96, 261}, {52, 255}, {66, 255}, {111, 255}, {112, 255},
{54, 254}, {198, 253}, {155, 251}, {156, 251}, {141, 250}, {142, 250}, {143, 250},
{144, 250}, {147, 250}, {150, 250}, {151, 250}, {152, 250}, {153, 250}, {158, 250},
{136, 249}, {137, 249}, {138, 249}, {139, 249}, {140, 249}, {141, 249}, {142, 249},
{144, 249}, {145, 249}, {146, 249}, {147, 249}, {148, 249}, {149, 249}, {150, 249},
{151, 249}, {152, 249}, {160, 249}, {83, 248}, {84, 248}, {85, 248}, {86, 248},
{87, 248}, {134, 248}, {135, 248}, {136, 248}, {137, 248}, {138, 248}, {141, 248},
{142, 248}, {143, 248}, {144, 248}, {145, 248}, {146, 248}, {147, 248}, {148, 248},
... 28627 ..., {175, 62}, {176, 62}, {177, 62}, {178, 62}, {179, 62}, {135, 61},
{136, 61}, {137, 61}, {138, 61}, {139, 61}, {140, 61}, {141, 61}, {142, 61}, {143, 61},
{144, 61}, {145, 61}, {146, 61}, {147, 61}, {148, 61}, {149, 61}, {150, 61},
{151, 61}, {153, 61}, {154, 61}, {155, 61}, {160, 61}, {161, 61}, {162, 61},
{163, 61}, {173, 61}, {174, 61}, {175, 61}, {176, 61}, {177, 61}, {178, 61},
{134, 60}, {135, 60}, {139, 60}, {140, 60}, {141, 60}, {142, 60}, {143, 60},
{144, 60}, {145, 60}, {146, 60}, {147, 60}, {148, 60}, {149, 60}, {150, 60},
{151, 60}, {152, 60}, {153, 60}, {154, 60}, {161, 60}, {166, 60}, {169, 60},
{173, 60}, {174, 60}, {175, 60}, {176, 60}, {141, 59}, {142, 59}, {143, 59},
{165, 59}, {169, 59}, {170, 59}, {169, 58}, {170, 58}, {171, 58}, {172, 58} }

large output

[show less](#)

[show more](#)

show all

set size limit...

Get pixel value of colored positions

```
PixelValue[r2[[1]], 13]
```

```
{0.780392, 0.776471, 0.8, 0.8, 0.8, 0.796078, 0.8, 0.8, 0.792157, 0.788235, 0.788235,
0.788235, 0.792157, 0.796078, 0.8, 0.772549, 0.792157, 0.8, 0.8, 0.788235, 0.788235,
0.796078, 0.8, 0.796078, 0.8, 0.796078, 0.788235, 0.784314, 0.796078, 0.788235, 0.8,
0.784314, 0.784314, 0.788235, 0.792157, 0.8, 0.796078, 0.8, 0.8, 0.788235, 0.772549,
0.792157, ... 28683 ..., 0.788235, 0.796078, 0.772549, 0.776471, 0.772549, 0.772549,
0.792157, 0.8, 0.792157, 0.788235, 0.764706, 0.721569, 0.713725, 0.74902, 0.788235,
0.768627, 0.760784, 0.784314, 0.780392, 0.756863, 0.760784, 0.780392, 0.788235,
0.8, 0.8, 0.792157, 0.8, 0.796078, 0.8, 0.780392, 0.780392, 0.788235, 0.780392,
0.756863, 0.768627, 0.8, 0.776471, 0.772549, 0.784314, 0.780392, 0.792157, 0.8}
```

[large output](#)

[show less](#)

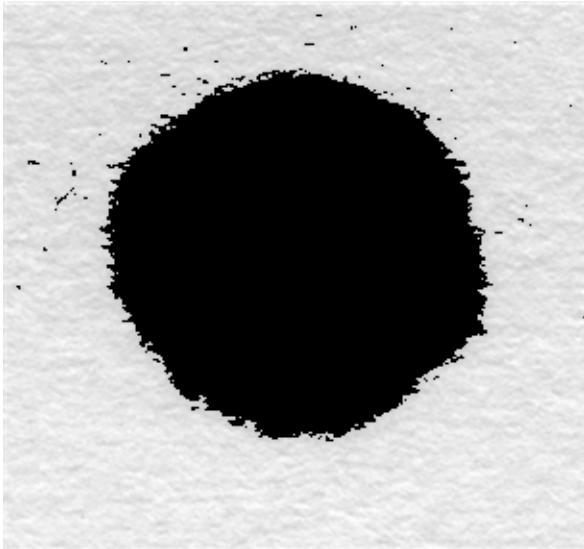
[show more](#)

[show all](#)

[set size limit...](#)

Calculate average paper signal in all channels

```
image2 = ReplacePixelValue[r2[[1]], 13 → 0]
```



```
14 = ImageData[image2]
```

```
{... 1 ...}
```

[large output](#)

[show less](#)

[show more](#)

[show all](#)

[set size limit...](#)

```

15 = Total[14, {2}]

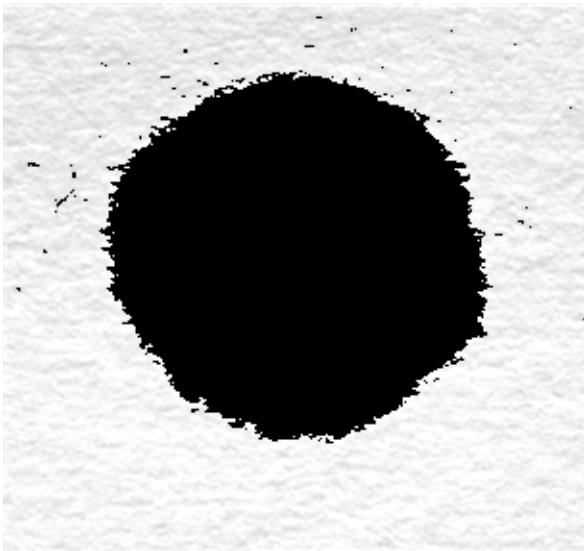
{284.953, 292.843, 272.231, 272.094, 271.094, 269.067, 267.847, 264.396, 267.31, 268.149,
269.075, 270.596, 269.408, 265.533, 264.682, 266.98, 268.714, 269.925, 269.894, 269.741,
268.976, 266.6, 261.808, 265.157, 264.773, 266.376, 267.678, 267.431, 267.173, 266.553,
265.702, 262.973, 267.733, 269.333, 270.259, 267.502, 259.624, 252.027, 236.027,
232.992, 227.596, 225.906, 220.839, 212.835, 200.275, 198.467, 195.957, 191.251,
187.243, 189.718, 189.247, 183.243, 179.529, 172.416, 169.067, 168.875, 164.267,
161.592, 160.09, 150.69, 145.812, 146.071, 149.114, 149.455, 146.098, 147.859, 148.569,
143.137, 140.094, 134.808, 135.227, 134.063, 135.878, 134.353, 131.486, 129.706,
127.608, 122.937, 123.6, 125.02, 124.525, 123.631, 118.275, 108.549, 101.702, 104.922,
110.38, 113.71, 113.204, 110.341, 106.149, 107.38, 107.478, 106.482, 109.231, 109.455,
109.137, 106.702, 105.412, 102.945, 100.024, 100.071, 99.9608, 102.784, 103.016,
101.882, 103.212, 104.302, 101.133, 102.4, 104.227, 104.616, 102.949, 100.157, 101.69,
102.945, 99.6392, 97.1098, 94.3216, 94.3137, 95.0275, 97.9098, 98.8431, 97.2471,
96.9922, 98.9647, 98.5059, 95.2667, 96.0275, 97.451, 99.7843, 101.012, 99.0392, 98.349,
98.8667, 100.361, 100.035, 95.2118, 96.4235, 97.6, 98.1647, 98.7922, 98.9216, 96.7765,
99.5137, 100.933, 96.5137, 93.1961, 91.6078, 95.7922, 102.118, 102.286, 101.749,
99.3804, 98.9137, 101.153, 102.424, 102.961, 105.451, 103.298, 106.502, 106.882,
107.804, 109.329, 108.29, 109.275, 106.733, 112.075, 111.549, 110.506, 110.204, 108.376,
109.898, 111.58, 118.62, 123.298, 123.596, 121.588, 124.659, 124.376, 127.047, 129.549,
132.22, 135.827, 136.196, 135.078, 133.408, 135.443, 138.235, 142.157, 146.973,
150.647, 152.506, 155.631, 157.016, 154.384, 155.596, 158.169, 157.8, 159.886, 162.608,
161.459, 160.6, 161.169, 165.608, 173.635, 176.569, 178.596, 182.259, 183.067, 184.349,
182.012, 186.667, 196.643, 200.055, 201.745, 211.302, 209.329, 207.8, 215.647, 219.247,
223.137, 231.278, 235.086, 240.988, 248.91, 252.463, 268.886, 271.518, 274.796,
276.247, 277.639, 277.706, 277.624, 276.506, 276.251, 278.765, 281.071, 280.941,
280.659, 279.961, 278.706, 278.243, 279.122, 279.969, 281.349, 282.851, 282.62,
280.659, 278.882, 278.573, 279.875, 281.122, 282.392, 284.898, 284.812, 282.647,
280.737, 278.78, 277.882, 279.267, 282.51, 284.451, 283.424, 281.267, 280.545, 280.882,
281.259, 282.424, 284.796, 285.267, 282.361, 280.392, 279.78, 279.824, 280.306, 281.31,
282.773, 283.388, 282.024, 279.235, 278.31, 279.251, 280.353, 289.114, 285.302}

B0 = (Sum[15[[n]], {n, 1, Length[15]}] / (10[[1]] * 10[[2]] - Length[13])))

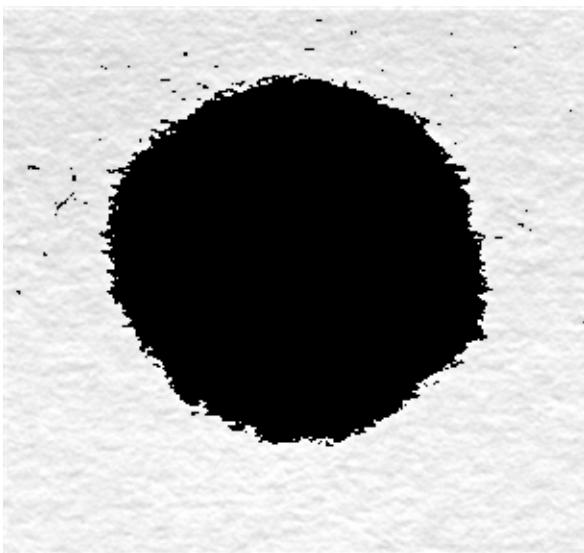
0.896021

```

```
imagepbg1 = ReplacePixelValue[bgchannel1, 13 → 0]
```



```
imagepbg2 = ReplacePixelValue[bgchannel2, 13 → 0]
```



```

bg1m = Total[ImageData[imagepb1], {2}]
{292.518, 302.043, 285.514, 285.471, 284.271, 281.384, 279.616, 276.227, 279.361, 280.58,
281.945, 283.914, 282.212, 277.933, 276.776, 279.671, 282.11, 283.596, 283.675, 283.91,
282.525, 280.078, 274.592, 277.902, 278.165, 280.384, 281.71, 281.416, 281.278, 280.353,
279.725, 277.604, 283.714, 286.306, 287.576, 285.075, 277.102, 268.945, 251.341,
248.192, 242.553, 240.996, 235.733, 226.804, 212.635, 210.788, 208.384, 203.255,
199.098, 202.353, 202.51, 195.6, 191.341, 183.22, 179.647, 179.749, 174.522, 171.961,
170.737, 160.545, 154.824, 154.412, 158.161, 159.176, 155.957, 158.51, 159.596, 153.192,
149.906, 143.69, 144.337, 143.188, 145.706, 144.227, 140.925, 138.953, 136.302, 131.294,
131.906, 133.965, 133.565, 132.741, 126.184, 115.337, 107.729, 111.365, 117.765,
121.643, 121.353, 117.984, 113.188, 114.255, 114.655, 113.831, 117.255, 117.565,
117.247, 114.49, 113.02, 110.078, 106.843, 106.675, 107.129, 110.122, 110.776, 109.6,
111.192, 112.176, 108.439, 110., 112.184, 112.647, 110.62, 107.863, 109.4, 110.831,
107.227, 103.969, 100.839, 101.063, 102.333, 105.443, 106.243, 104.518, 104.212,
106.443, 105.882, 102.361, 103.486, 105.188, 107.725, 108.906, 106.498, 105.651,
106.424, 108.173, 107.608, 101.941, 103.486, 105.016, 105.651, 106.306, 106.647,
104.494, 107.71, 108.682, 103.141, 99.3725, 98.0471, 103.227, 110.447, 110.133, 109.435,
106.871, 106.169, 108.91, 110.498, 111.043, 113.592, 111.055, 114.812, 115.035,
116.067, 117.922, 116.78, 117.71, 114.235, 120.616, 119.816, 118.663, 118.243, 116.067,
117.969, 119.518, 127.976, 133.196, 133.039, 130.882, 134.396, 133.243, 136.494,
139.631, 142.639, 146.384, 146.725, 145.392, 143.086, 145.043, 148.008, 152.761, 158.38,
161.937, 163.514, 167.525, 169.361, 166.129, 167.388, 170.169, 169.204, 171.675,
174.584, 172.859, 171.588, 172.184, 177.263, 186.89, 190.239, 191.949, 196.024,
196.694, 197.431, 194.443, 199.573, 211.035, 214.725, 216.012, 227.047, 224.659,
222.71, 231.557, 234.757, 238.808, 247.831, 251.224, 258.11, 267.745, 271.529, 290.122,
292.188, 295.196, 295.169, 296.2, 295.918, 294.98, 293.651, 293.02, 295.286, 297.671,
297.475, 297.373, 296.29, 294.729, 294.184, 294.686, 294.537, 295.878, 297.855, 298.243,
296.078, 293.537, 293.125, 293.941, 295.094, 296.565, 298.976, 299.384, 297.286,
295.129, 292.6, 291.047, 292.129, 295.129, 297.278, 297.039, 294.686, 293.055, 293.51,
294.471, 295.706, 297.929, 298.416, 295.961, 293.443, 292.404, 292.137, 292.247, 293.231,
295.102, 295.984, 294.294, 290.953, 289.851, 290.675, 291.882, 299.529, 296.094}

bg1 = Sum[bg1m[[n]], {n, 1, Length[bg1m]}] / (10[[1]] * 10[[2]] - Length[13])
0.950872

```

```

bg2m = Total[ImageData[imagepb2], {2}]
{289.737, 299.757, 280.161, 280.129, 279.271, 276.769, 275.525, 272.039, 274.831, 275.843,
277.318, 278.902, 277.522, 273.557, 272.424, 275.027, 277.416, 278.71, 278.655, 278.824,
277.8, 275.275, 270.118, 273.588, 273.612, 275.514, 276.722, 276.784, 276.659, 275.918,
275.318, 272.937, 278.482, 281.043, 282.361, 279.882, 271.976, 264.071, 247.067,
243.839, 238.251, 236.667, 231.435, 222.808, 208.898, 207.161, 204.784, 199.482,
195.251, 198.275, 198.251, 191.337, 187.651, 179.8, 176.196, 176.345, 171.251, 168.62,
167.18, 157.137, 151.533, 151.749, 155.459, 156.145, 152.773, 155.094, 156.094,
149.996, 146.886, 140.702, 141.514, 140.227, 142.561, 141.075, 137.843, 135.878,
133.306, 128.192, 128.973, 131.063, 130.678, 129.922, 123.647, 113.094, 105.643,
109.349, 115.514, 119.059, 118.796, 115.565, 110.816, 112.067, 112.486, 111.478,
114.808, 115.11, 114.706, 111.827, 110.322, 107.612, 104.533, 104.427, 104.973,
107.925, 108.384, 107.024, 108.627, 109.788, 106.173, 107.592, 109.647, 110.071,
108.224, 105.522, 107.055, 108.569, 105.008, 101.914, 98.6863, 98.8784, 100.157,
103.008, 103.863, 102.259, 101.929, 104.169, 103.616, 100.027, 101.031, 102.694,
105.227, 106.522, 104.165, 103.455, 103.996, 105.765, 105.463, 99.8314, 101.502,
102.788, 103.408, 104.075, 104.275, 101.859, 105.067, 106.365, 101.004, 97.2902,
95.9098, 100.776, 107.835, 107.639, 107.078, 104.357, 103.706, 106.6, 107.816,
108.243, 111.153, 108.455, 112.149, 112.416, 113.353, 115.341, 114.086, 115.024,
111.769, 117.761, 117.075, 115.988, 115.498, 113.325, 115.016, 116.808, 124.824,
130.055, 130.188, 127.867, 131.482, 130.596, 133.592, 136.388, 139.475, 143.243,
143.435, 141.922, 139.643, 141.922, 144.988, 149.506, 154.996, 158.722, 160.329,
164.196, 165.706, 162.318, 163.62, 166.329, 165.514, 167.851, 170.741, 168.976,
167.855, 168.224, 173.251, 182.698, 185.78, 187.729, 191.792, 192.349, 193.149,
190.255, 195.31, 206.643, 210.157, 211.427, 222.498, 219.941, 217.718, 226.769,
230.282, 234.165, 243.204, 246.706, 253.604, 262.667, 266.416, 285.106, 287.039,
289.62, 290.09, 290.69, 290.035, 289.455, 288.024, 287.255, 289.58, 291.631, 291.494,
291.498, 290.588, 288.98, 288.18, 289.024, 289.125, 290.722, 292.412, 292.471, 290.361,
288.086, 287.545, 288.765, 290.118, 291.098, 294.055, 294.133, 291.675, 289.627,
287.141, 286.035, 287.475, 290.627, 292.376, 291.8, 289.769, 288.471, 288.718, 289.431,
290.545, 292.894, 293.42, 290.553, 288.373, 287.522, 287.298, 287.631, 288.702,
290.365, 291.137, 289.502, 286.416, 285.502, 286.357, 287.608, 295.976, 291.925}

```

```

bg2 = Sum[bg2m[[n]], {n, 1, Length[bg2m]}] / (10[[1]] * 10[[2]] - Length[13])
0.932757

```

Calculations of absorbance in each channel using a background value of 1 (a saturated pixel)

```

r2A = Evaluate[-Log10[r2[[1]] / 1]];
bgchannel1A = Evaluate[-Log10[bgchannel1 / 1]];
bgchannel2A = Evaluate[-Log10[bgchannel2 / 1]];

```

As to not influence the paper background absorbance with dye coloured pixels we set the values in the threshold (l3) to the average paper value

```

imagersforA = ReplacePixelValue[r2[[1]], 13 → B0];
imagebg1forA = ReplacePixelValue[bgchannel1, 13 → bg1];
imagebg2forA = ReplacePixelValue[bgchannel2, 13 → bg2];
r2bgA = Evaluate[-Log10[imagersforA / 1]];
bg1channelbgA = Evaluate[-Log10[imagebg1forA / 1]];
bg2channelbgA = Evaluate[-Log10[imagebg2forA / 1]];

```

Shown here are ratios of signal from both dye and paper between channels

```
B0A = Mean[r2bgA]
```

```
0.0482145
```

```
bg1A = Mean[bg1channelbgA]
```

```
0.022359
```

```
bg2A = Mean[bg2channelbgA]
```

```
0.0303402
```

```
dyers = Mean[PixelValue[r2A, 13]] - B0A
```

```
0.219874
```

```
dyebg1 = Mean[PixelValue[bgchannel1A, 13]] - bg1A
```

```
-0.00173539
```

```
dyebg2 = Mean[PixelValue[bgchannel2A, 13]] - bg2A
```

```
-0.0180987
```

Applying simultaneous equations we can obtain the dye and paper contribution using either of the background channels

```
dyec1 = ((bgchannel1A * B0A) - (r2A * bg1A)) / ((B0A * dyebg1) - (bg1A * dyers));
```

```
dyec2 = ((bgchannel2A * B0A) - (r2A * bg2A)) / ((B0A * dyebg2) - (bg2A * dyers));
```

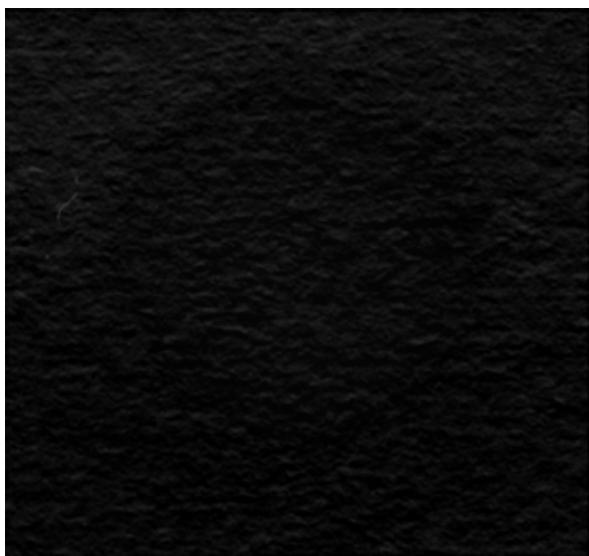
```
paperc1 = ((bgchannel1A * dyers) - (r2A * dyebg1)) / ((dyers * bg1A) - (dyebg1 * B0A));
```

```
paperc2 = ((bgchannel2A * dyers) - (r2A * dyebg2)) / ((dyers * bg2A) - (dyebg2 * B0A));
```

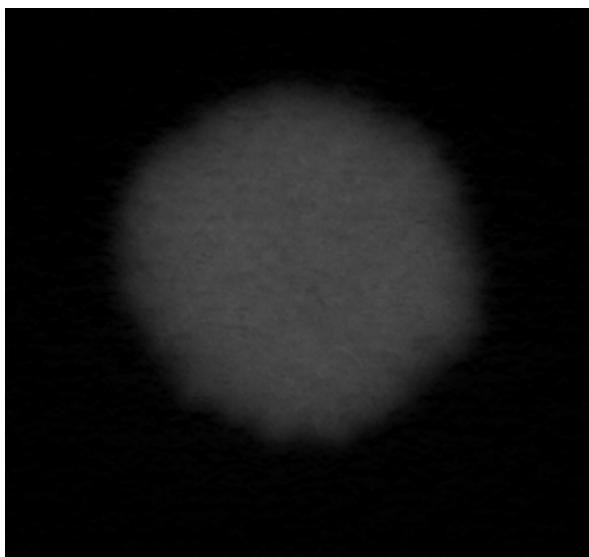
We can then use the average of the solutions

```
dyet = ((dyec1 + dyec2) / 2);
```

```
papert = ((paperc1 + paperc2) / 2) * B0A
```

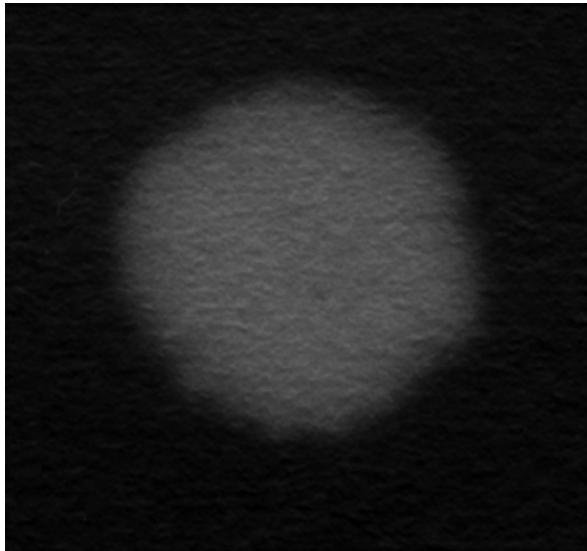


```
dyetwhole = (dyet * dyers)
```



The original absorbance for comparison

```
r2A = Evaluate[-Log10[r2[[1]] / 1]]
```



Line plot illustrating the isolation of the dye and paper component of the original image

```
testline1 = ImageData[ImageRotate[papert]]
```

{ ... 1 ... }

[large output](#)

[show less](#)

[show more](#)

[show all](#)

[set size limit...](#)

```
testline2 = ImageData[ImageRotate[dyetwhole]]
```

{ ... 1 ... }

[large output](#)

[show less](#)

[show more](#)

[show all](#)

[set size limit...](#)

```
testline3 = ImageData[ImageRotate[r2A]]
```

{ {0.0120885, 0.0299632, 0.0299632, 0.0448123, 0.0391843, 0.0173741, 0.0410522, 0.0524317, 0.0373243, 0.0373243, 0.0486053, 0.0391843, 0.0317918, 0.0391843, 0.0336282, 0.0429282, 0.0448123, 0.0373243, 0.0263289, 0.0391843, 0.0562922, 0.0448123, 0.0391843, 0.0373243, 0.0263289, 0.0227248, 0.0317918, 0.0410522, 0.0391843, 0.0336282, ... 226 ..., 0.0299632, 0.0410522, 0.0317918, 0.0391843, 0.0299632, 0.0209339, 0.0354723, 0.0448123, 0.0410522, 0.0156051, 0.00860017, 0.0156051, 0.0281423, 0.0227248, 0.0120885, 0.0227248, 0.0391843, 0.0410522, 0.0448123, 0.0391843, 0.0281423, 0.0227248, 0.0263289, 0.0227248, 0.0209339, 0.0120885, 0.0245231, 0.0317918, 0.0354723, 0.0281423}, ... 303 ..., { ... 1 ... } }

[large output](#)

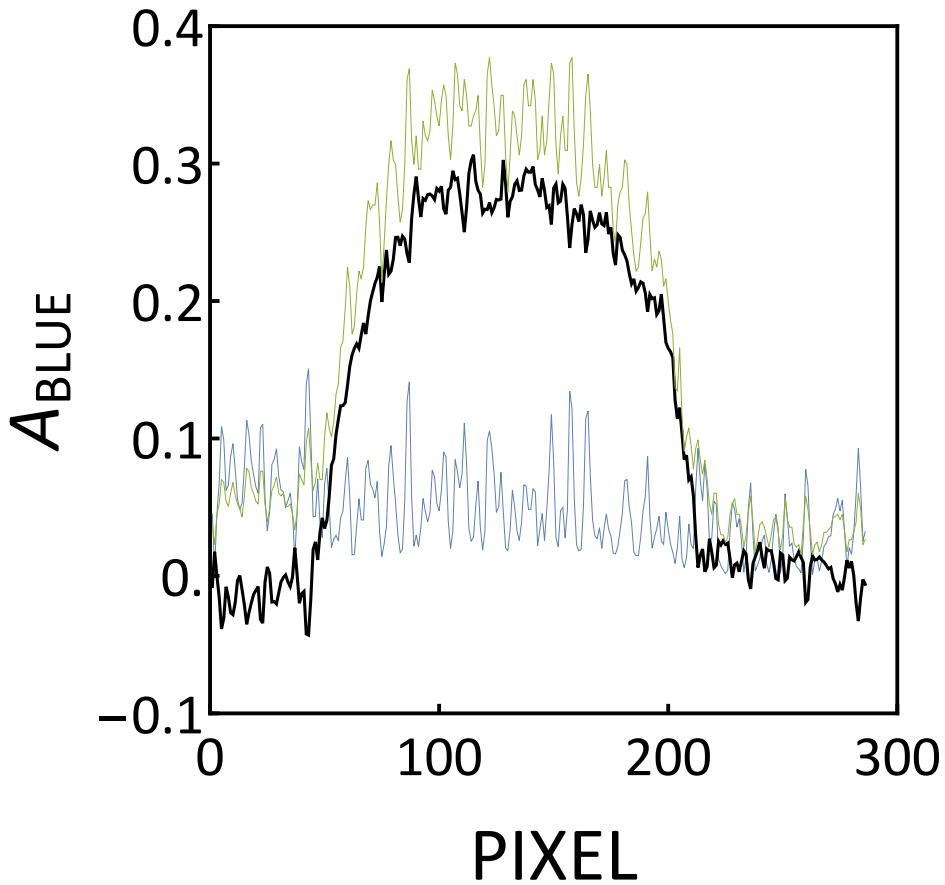
[show less](#)

[show more](#)

[show all](#)

[set size limit...](#)

```
ListLinePlot[{testline1[[200]], testline2[[200]], testline3[[200]]},
  GridLines -> {None, {B0}}, PlotRange -> {{0, 300}, {0.4, -0.1}}, Axes -> False,
  PlotStyle -> {Thin, Black}, Frame -> True, Joined -> True, AspectRatio -> 1,
  FrameLabel -> {Style["PIXEL", 40], Style[Subscript[A, BLUE], 40]},
  FrameStyle -> Directive[Black, Thick],
  BaseStyle -> {30, FontFamily -> "Calibri"}, ImageSize -> 500,
  FrameTicks -> {{Table[u, {u, -0.1, 0.7, 0.1}], {}}, {Table[u, {u, 0, 600, 100}], {}}}]
```



Check for adequate component extraction

```
Check1 = StandardDeviation[papert]
```

```
0.0278961
```

```
Check2 = StandardDeviation[r2A]
```

```
0.11283
```

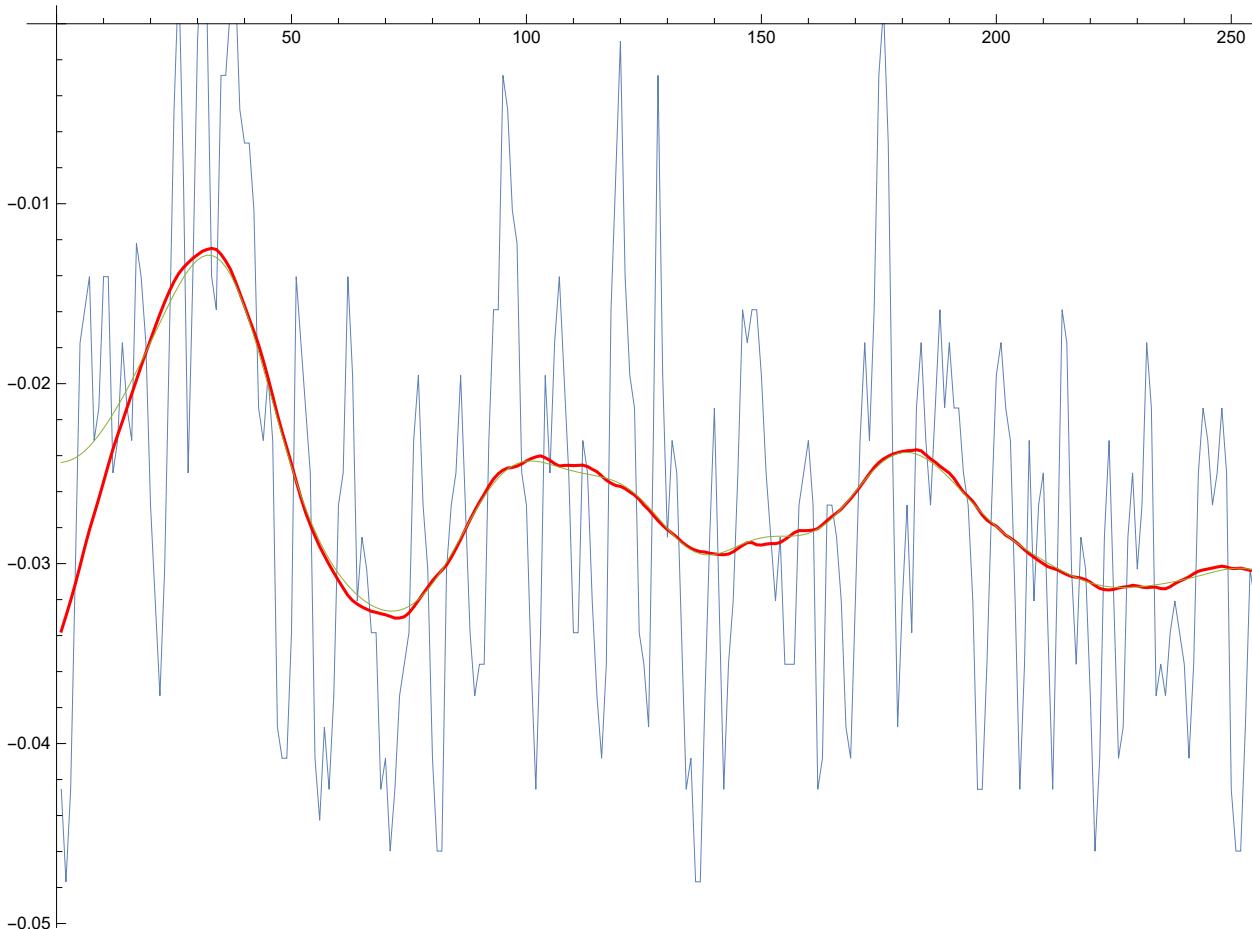
```
If[Check1 < Check2,
 {method1 = "Yes, advanced background was carried out", Imageforconversion = dyetwhole;},
 {method1 = "No, dye signal was insufficient and simplified background
 subtraction was carried out", Imageforconversion = AblueListi;}]
{Yes, advanced background was carried out, Null}
```

Simple Background Calculation in the case of Low Dye Signal

```
bglist = ImageData[r5[[1]]];
Aredlist = Evaluate[-Log10[bglist / B0]];
```

Decide which filtering function is decent. Gaussian filter was chosen in this case

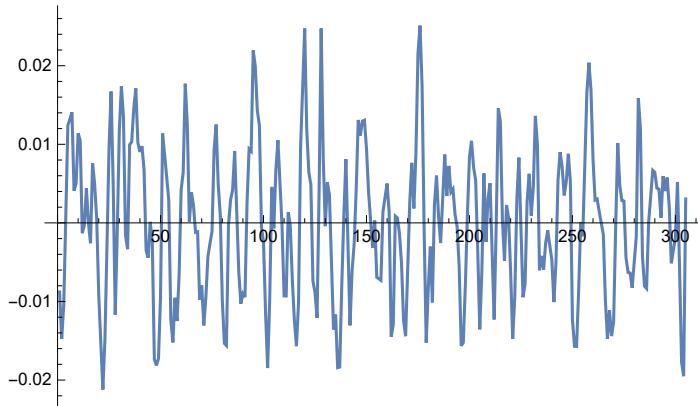
```
AredlistsSmoothLine = ListLinePlot[{Aredlist[[190]], GaussianFilter[Aredlist[[190]], 20],
 BilateralFilter[Aredlist[[190]], 2, .5, MaxIterations -> 25]}, PlotStyle -> {Thin, Red}]
```



Extract the absorbance around the filter in background

channel

```
ListLinePlot[Aredlist[[190]] - GaussianFilter[Aredlist[[190]], 20]]
```



Extract only noise from the background channel image

```
Aredgaussianlist = GaussianFilter[Aredlist, 20];
noise = Aredlist - Aredgaussianlist;
Imagecorrected = Image[Aredlist - Aredgaussianlist];
```

Absorbance in response channel with noise

```
rslist = ImageData[11[[3]]]
```

{ ... 1 ... }

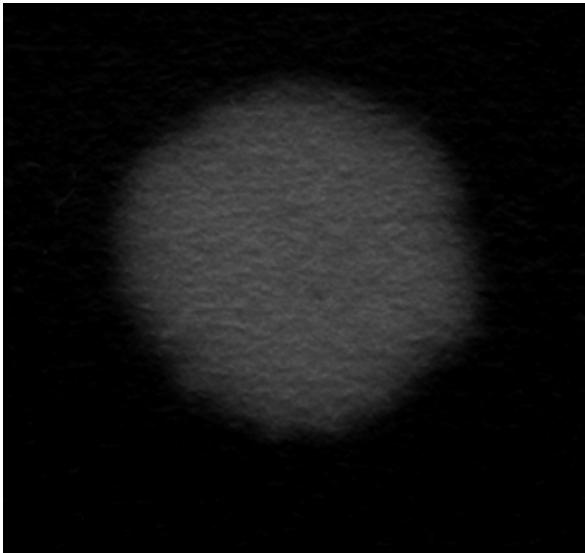
[large output](#) [show less](#) [show more](#) [show all](#) [set size limit...](#)

```
Arslistnoise = Evaluate[-Log10[rslist/B0]]
```

{ ... 1 ... }

[large output](#) [show less](#) [show more](#) [show all](#) [set size limit...](#)

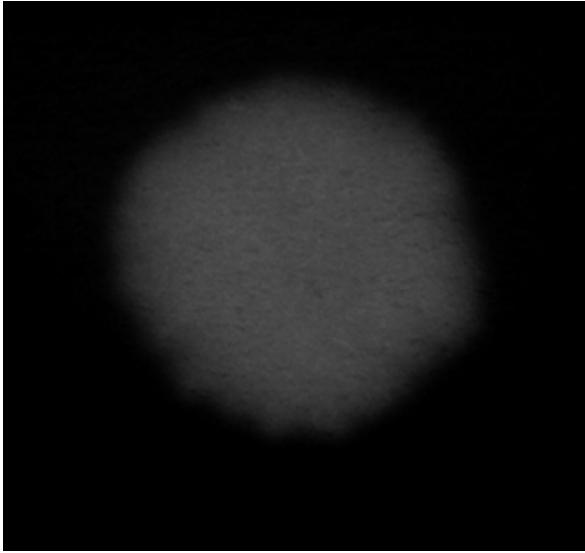
```
Image[Arslistnoise]
```



Cancel the noise

```
Arslist = Arslistnoise - noise;
```

```
Arslisti = Image[Arslist]
```



Conversion of absorbance values into quantity

```
opt = {base → -28.43, max → 6903.8, xhaf → 0.75323, rate -> 0.13742};
```

```

eq1 = 8 * 10^-6 / 6785 * (base + (max / (1 + Exp[(xhaf - A) / rate]))) /. opt
- 28.43 +  $\frac{6903.8}{1 + e^{7.27696(0.75323-A)}}$ 
848 125 000

```

Quantification

```
Qlist = eq1 /. A → Imageforconversion;
```

```
12 = Total[Qlist, {2}]
```

```
0.00430884
```

```
12 / (10 * 10^-6)
```

```
430.884
```

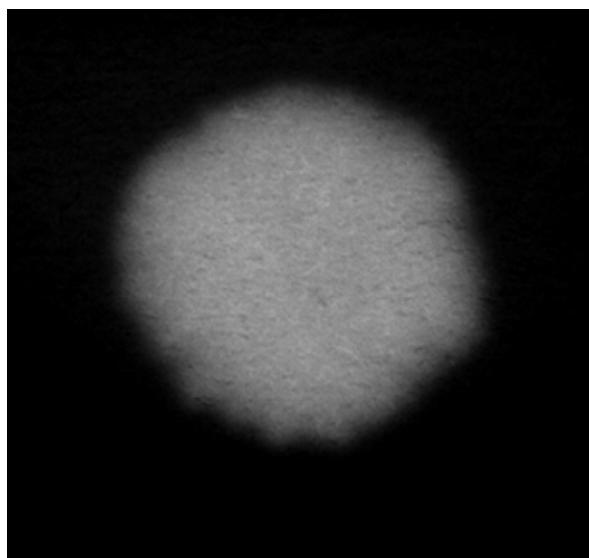
Was the dye concentration suitable for advanced background subtraction?

method1

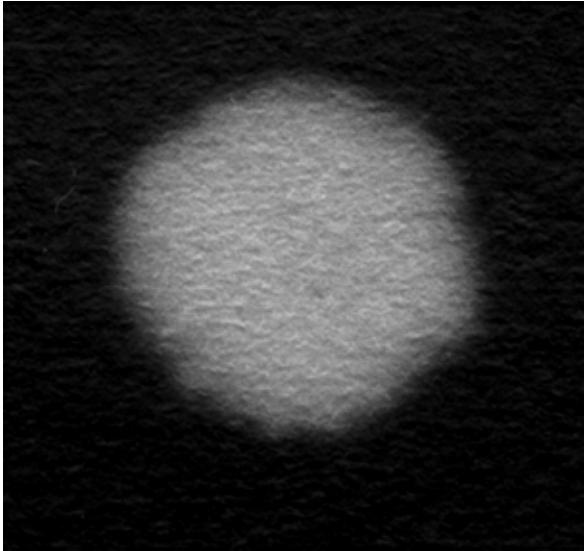
Yes, advanced background was carried out

Figures for paper

```
noiseextracted = Image[Arslist * 2]
```



```
l11[[3]] * 2
```



```
ListPlot3D[Arslist, PlotRange -> All, AspectRatio -> {1, 1, 0.7},
ColorFunction -> GrayLevel, ImageSize -> 800, AxesLabel ->
{Style[x, FontSize -> 30, Black, "Helvetica"], Style[y, FontSize -> 30, Black, "Helvetica"],
Rotate[Style["ARGB", FontSize -> 30, Black, "Helvetica"], Pi/2]}, 
AxesStyle -> Directive[Black, 15, FontFamily -> "Helvetica"], Boxed -> False,
TicksStyle -> Directive[FontSize -> 25], Axes -> True,
AxesEdge -> {{-1, -1}, {1, -1}, {-1, -1}}]

ListPlot3D[ImageData[l11[[3]]], PlotRange -> All,
AspectRatio -> {1, 1, 0.7}, ColorFunction -> GrayLevel, ImageSize -> 800, AxesLabel ->
{Style[x, FontSize -> 30, Black, "Helvetica"], Style[y, FontSize -> 30, Black, "Helvetica"],
Rotate[Style["ARGB", FontSize -> 30, Black, "Helvetica"], Pi/2]}, 
AxesStyle -> Directive[Black, 15, FontFamily -> "Helvetica"], Boxed -> False,
TicksStyle -> Directive[FontSize -> 25], Axes -> True,
AxesEdge -> {{-1, -1}, {1, -1}, {-1, -1}}]

ListPlot3D[ImageData[Qlist / 318.9 * 10^9], PlotRange -> All,
AspectRatio -> {1, 1, 0.7}, ColorFunction -> GrayLevel, ImageSize -> 800, AxesLabel ->
{Style[x, FontSize -> 30, Black, "Helvetica"], Style[y, FontSize -> 30, Black, "Helvetica"],
Rotate[Style["QUANTITY/ fM", FontSize -> 30, Black, "Helvetica"], Pi/2]}, 
AxesStyle -> Directive[Black, 15, FontFamily -> "Helvetica"], Boxed -> False,
TicksStyle -> Directive[FontSize -> 25], Axes -> True,
AxesEdge -> {{-1, -1}, {1, -1}, {-1, -1}}]
```