Supplementary Information (SI) for Analyst.

Supplementary Information

for

Quantitative Analysis of Colorimetric Hydrogel/Paper Disk Arrays with a Handheld Wi-Fi Scanner

Lin Qi,[†,⊥] Pitipat Parittothok,[†,‡,⊥] Sophia Sun,[†] Jakrapop Wongwiwat,[‡] Aluck Thipayarat,[§] Wanida Laiwattanapaisal,[#] and Hua-Zhong Yu[†,#,*]

[†]*Department of Chemistry, Simon Fraser University, Burnaby, British Columbia V5A 1S6, Canada*

[‡]*Department of Mechanical Engineering, King Mongkut's University of Technology Thonburi, Bangkok 10140, Thailand*

[§] *Department Food Engineering, King Mongkut's University of Technology Thonburi, Bangkok 10140, Thailand*

[#] *Department of Clinical Chemistry, Chulalongkorn University, Bangkok 10330, Thailand*

[*] *Corresponding author. E-mail: hogan_yu@sfu.ca*

[⊥] *These authors contributed equally to the work.*

Description of the design and function of the color analysis app; the fabrication and detection procedure of hydrogel-coated filter paper; the influence of using scanning box on the imaging results; optimization of the two colorimetric assays performed with the Wi-Fi scanning method and conventional UV/Vis spectrophotometric measurements; additional photos and quantitation results of real samples; the original codes for the ChromaDetect app.

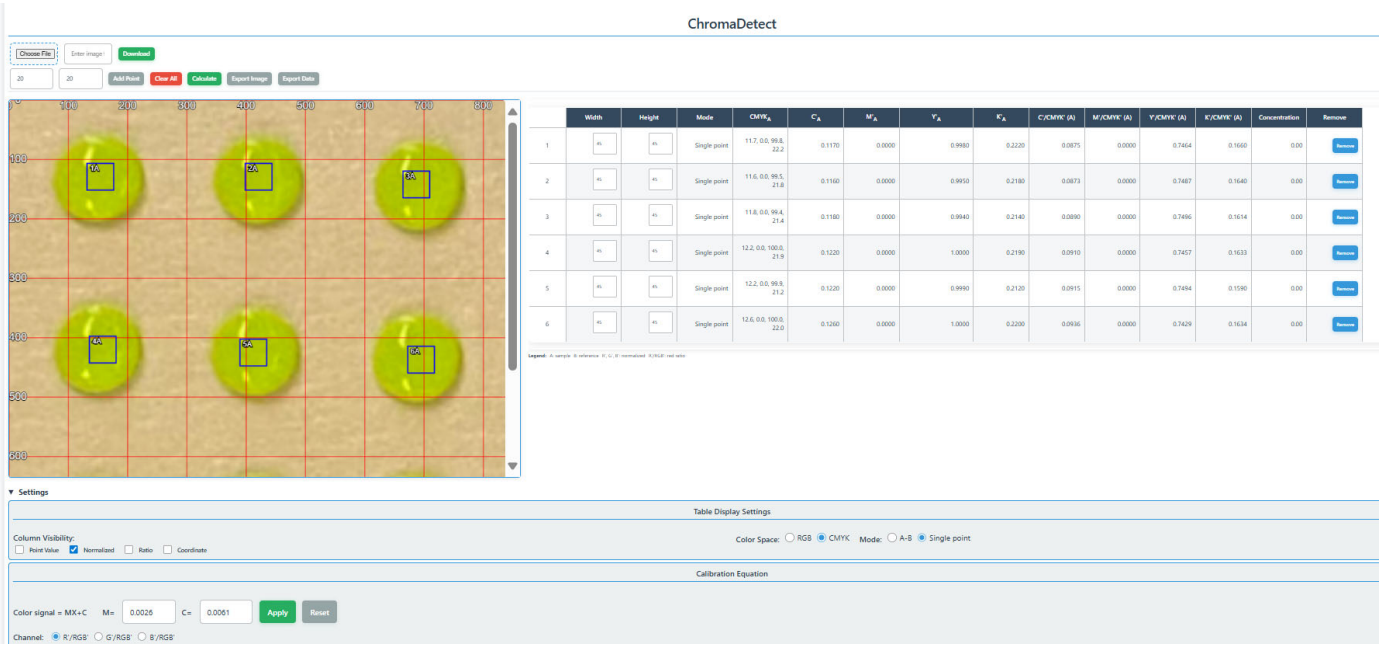

Fig. S1 Screen shot of the interface of the developed *ChromaDetect* app. This app can run on all different mobile devices (smartphones, pads, and laptops) and regular computers with a web browser (no OS restrictions). The image can be uploaded from a local device in either PNG or JPEG format. With one or multiple pre-defined areas (a circle or rectangular region with adjustable coordinate and dimension), with this app we can analyze color information based on either RGB or CMYK color space for each pixel, and provide the average and standard deviation of all pixels in the defined area as the output values. The color analysis can be done with a single point mode (to analyze the color for each pre-defined area) or "A-B" mode (to determine the difference in the color information between two pre-defined areas). In addition, a linear calibration equation (with user-defined slope and intercept values, M and C) can be created, which enables the calculation of the analyte concentration based on the normalized color intensity. The color analysis results can be exported to Excel (xlsx) for further analysis and data processing by clicking "Export Data". The original codes for the ChromaDetect app are presented at the end of this document.

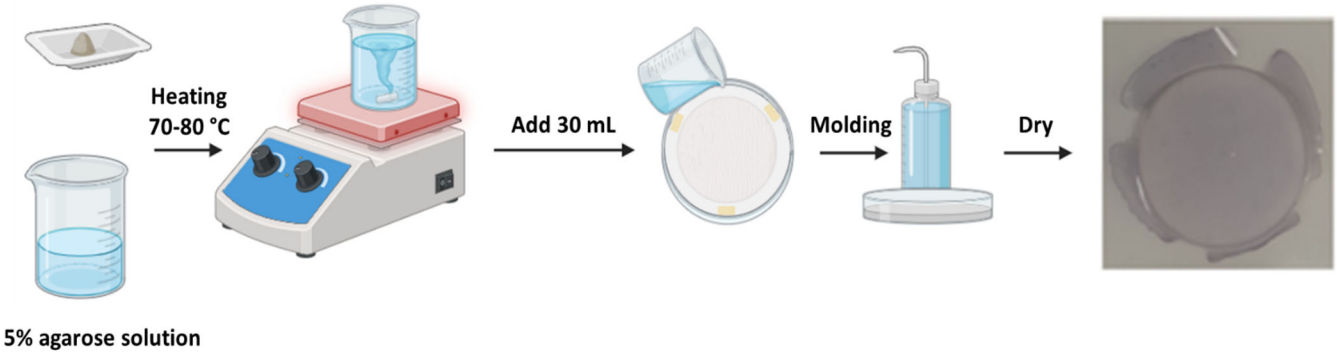

Fig. S2 Preparation of hydrogel coated filter paper. Briefly, the hydrogel-coated filter paper was prepared by dissolving agarose powder in 70 °C water (w/v = 5%) and stirred for at least 15 min to obtain a clear, homogeneous solution (all agarose dissolved). Subsequently, the agarose solution was immediately poured onto a piece of Grade3 filter paper placed inside the lid of a plastic Petri dish, and the bottom of the Petri dish was pressed on top. To control the thickness of agarose film, small pieces of double-sided tape were positioned between the dish bottom and the lid to separate them by ~0.75 mm. After pressing for at least 30 min, the hydrogel-coated filter paper was taken out, dried overnight, and was stored at room temperature before use.

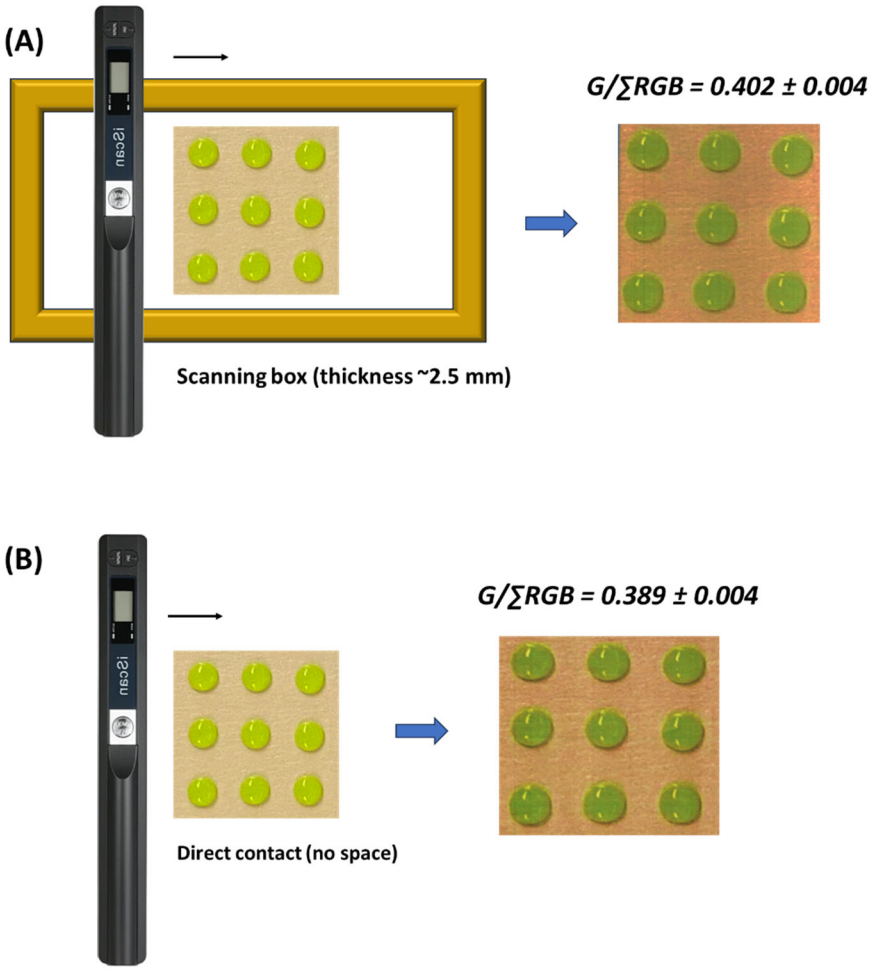

Fig. S3 Image scanning and color analysis of a printed photo of the agarose/paper disk array (loaded with 20 µL of a green food color) with (A) and without (B) using a scanning box (~ 2.5 mm thickness). It demonstrates that the space between the scanner and the object caused by using a scanning box (to avoid contact between the bottom of the scanner and the liquid drop loaded onto the paper disk) has no significant influence on the color information of the scanned images (similar normalized green values, G/ΣRGB).

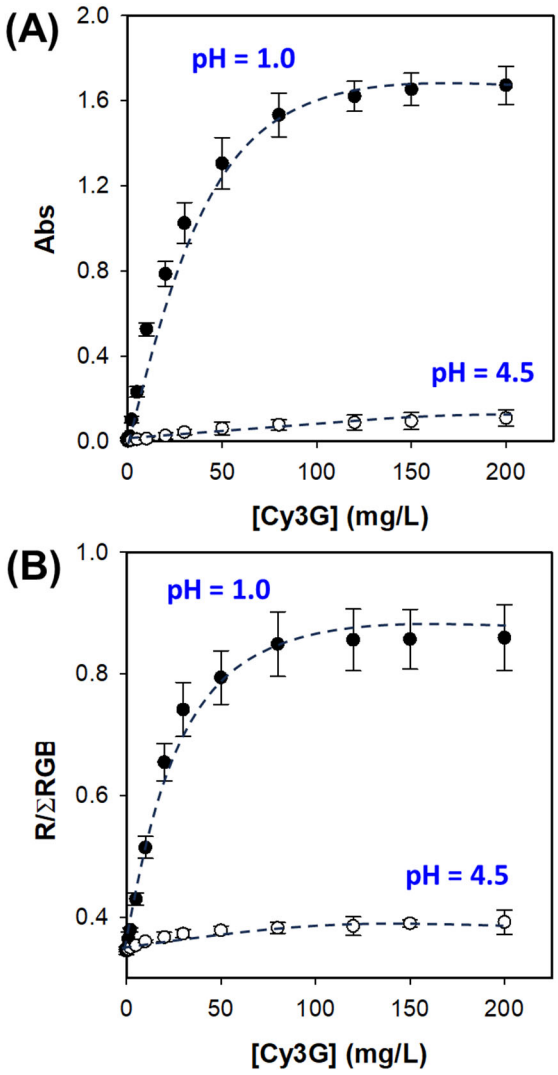

Fig. S4. (A) Absorbance (Abs) at 512 nm of different concentrations of anthocyanin (Cy3G) at pH 1.0 (solid circles) and pH 4.5 (open circles). (B) Normalized red intensity (R/ΣRGB) of different concentrations of anthocyanin (Cy3G) at pH 1.0 (solid circles) and pH 4.5 (open circles) by performing the pH differential colorimetric assay with the Wi-Fi scanning method. The error bars in (A) and (B) are obtained from three repeating experiments. The dash lines in in (A) and (B) are for eye guide only.

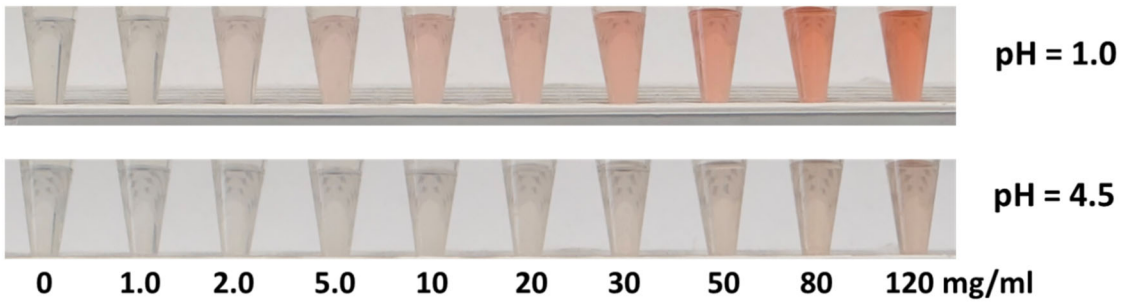

Fig. S5 Anthocyanin (Cy3G) solutions of different concentrations at pH 1.0 and pH 4.5 prepared for the UV-vis absorbance measurements.

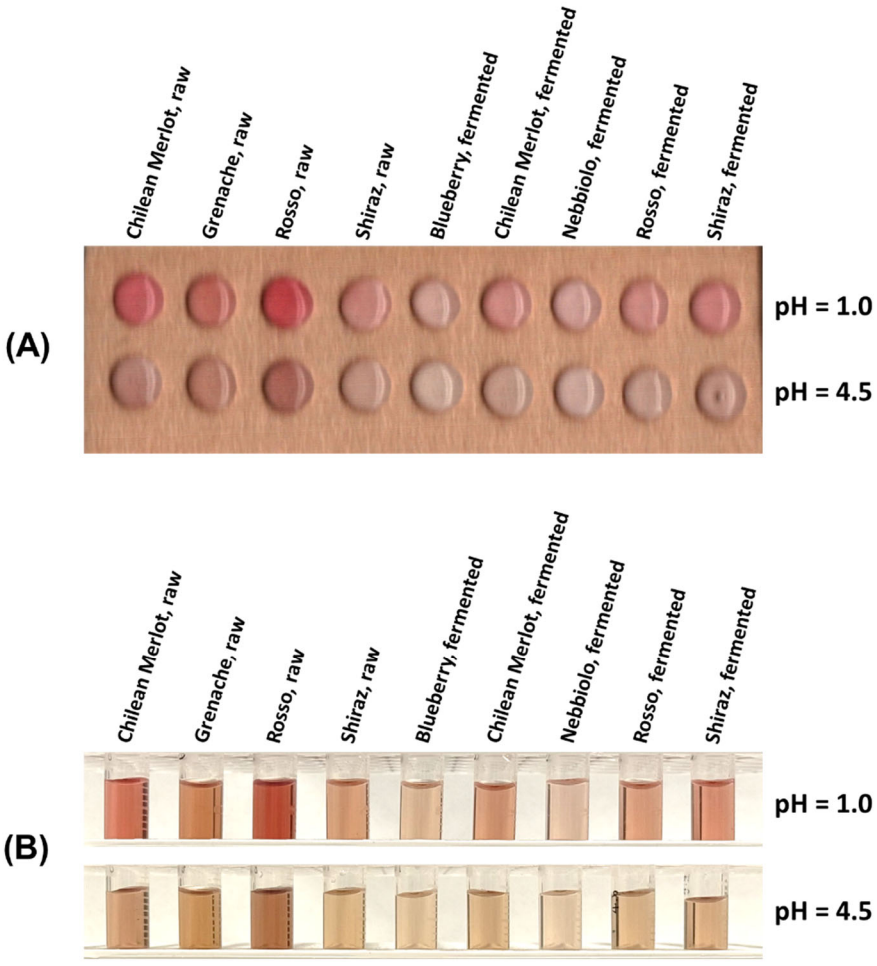

Fig. S6 (A) Scanned image of total anthocyanins detection results in beverages by performing the pH differential colorimetric assay (at pH 1.0 and pH 4.5) with the Wi-Fi scanning method. (B) Anthocyanin solutions of different concentrations at pH 1.0 and pH 4.5 prepared for the UV/Vis spectrophotometric measurements.

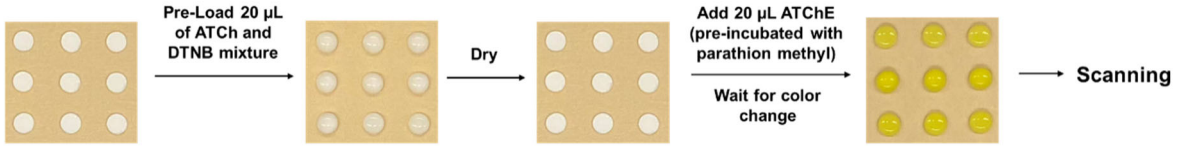

Fig. S7 Performing the Ellman's assay for parathion methyl on a hydrogel/paper disk array.

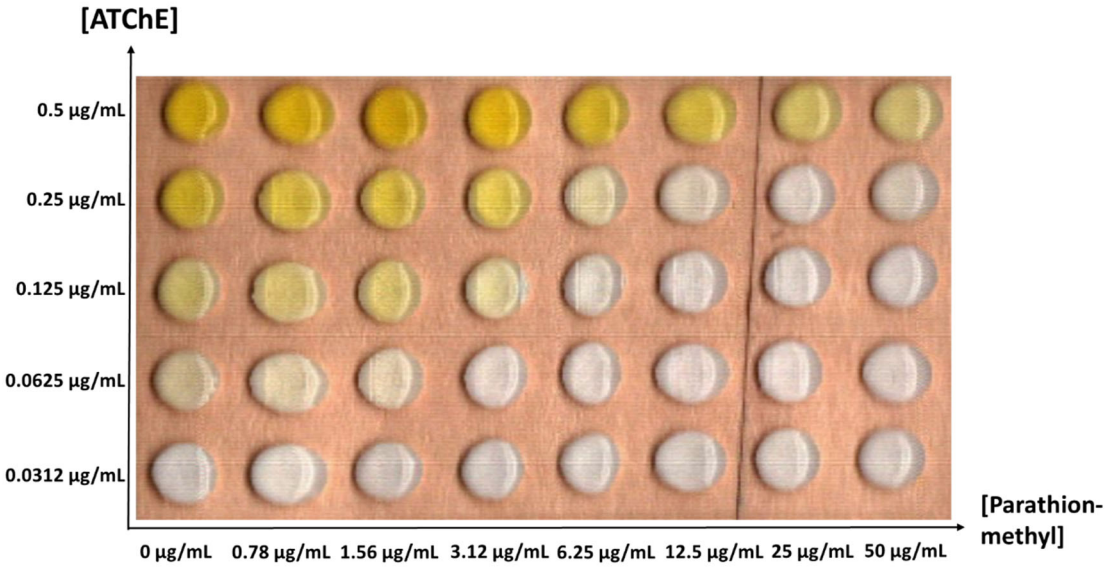

Fig. S8 Scanned image of the Ellman's assay on hydrogel-paper disk array with different concentrations of ATChE and parathion methyl. The data show that for the range of 0 to 50 µg/mL of parathion methyl, 0.25 µg/mL of ATChE would be optimal for performing the quantitation.

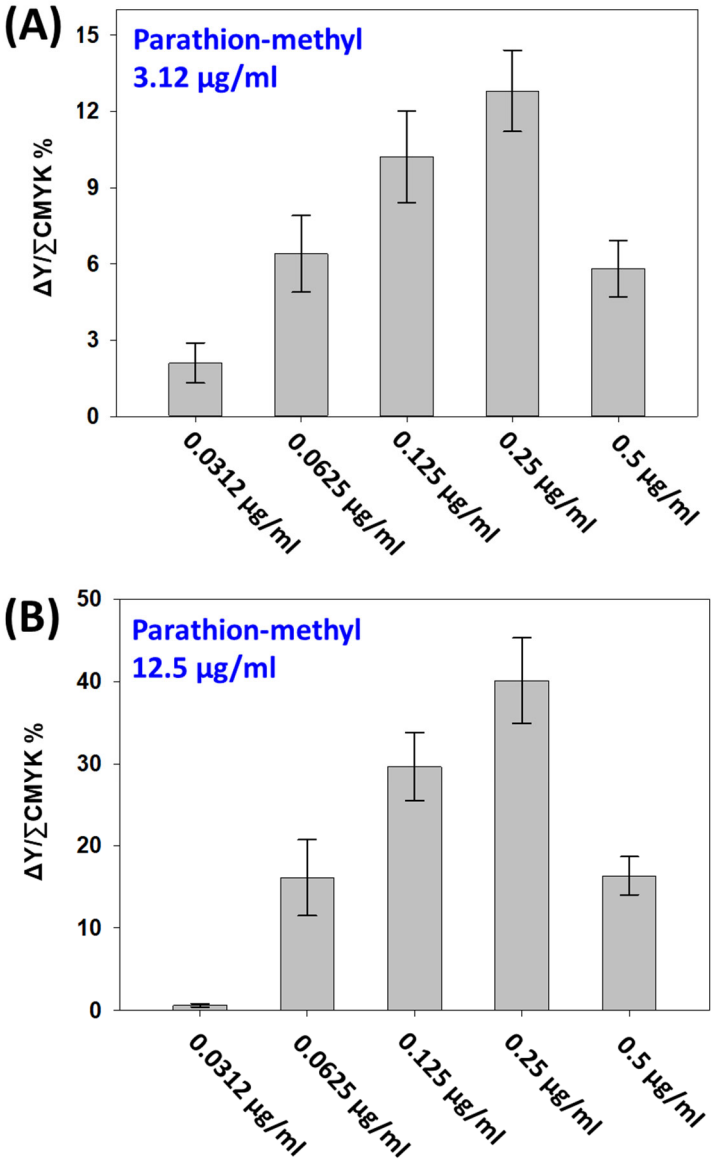

Fig. S9 Influence of the ATChE concentration on the sensing response of (A) 3.12 µg/mL and (B) 12.5 µg/mL of parathion methyl by performing the Ellman's assay with the Wi-Fi scanning method (ΔY/ΣCMYK: difference between the normalized yellow color intensities with and without incubating ATChE with parathion methyl, which are analyzed from the detection result shown in Fig. S5B). It can be observed that 0.25 µg/mL ATChE results in the strongest sensing response for both concentrations of parathion methyl studied.

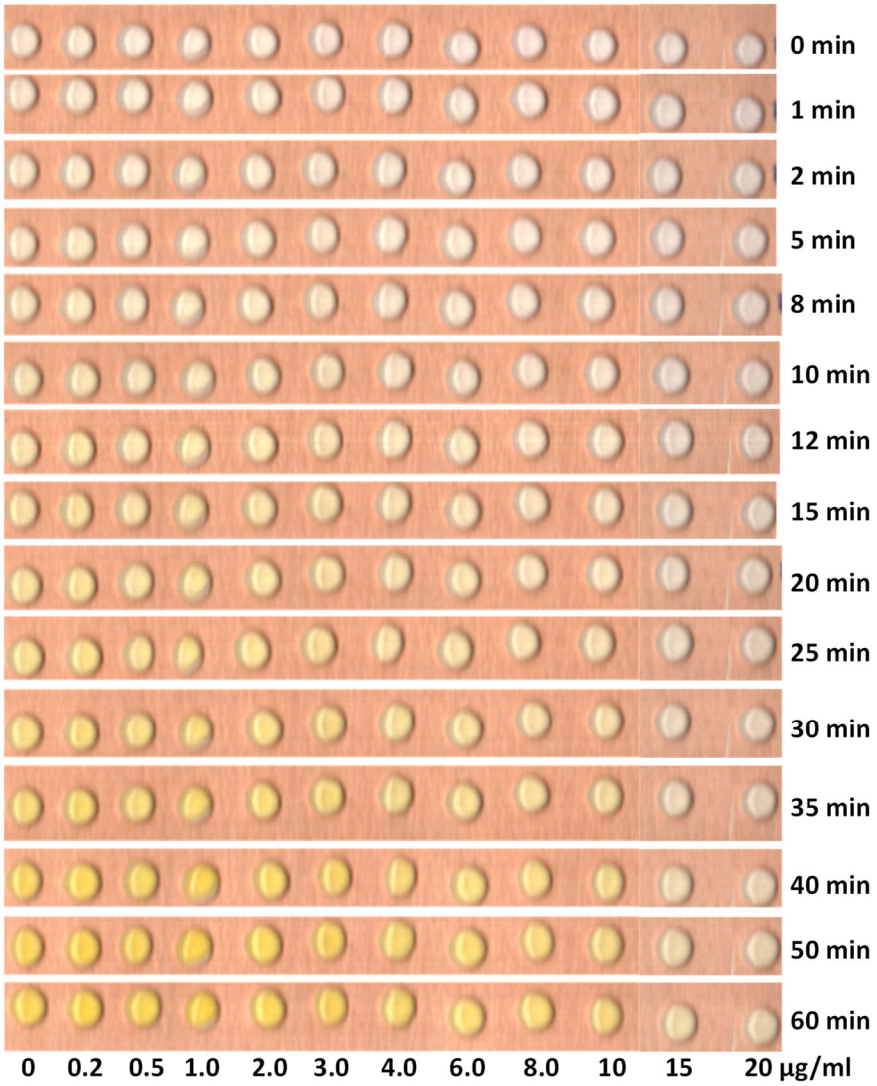

Fig. S10 Scanned image of parathion methyl detection results at different time periods by performing the Ellman's assay on a hydrogel-paper disk array.

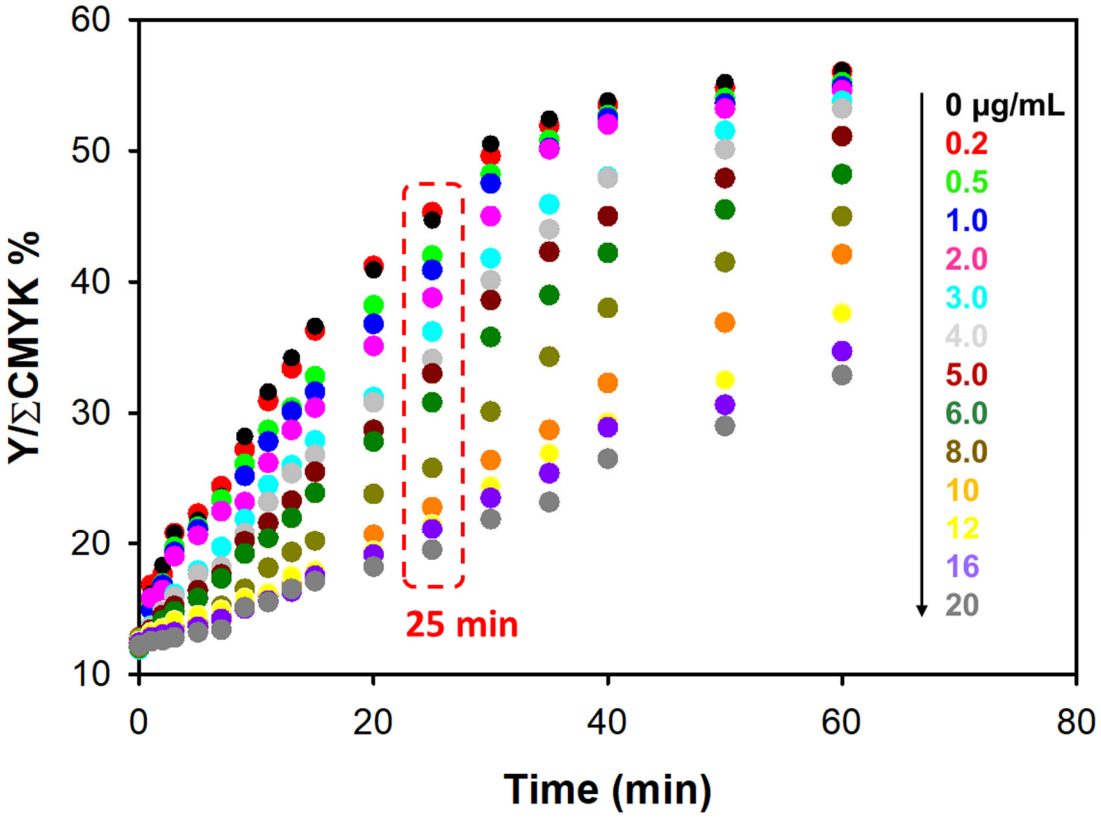

Fig. S11 Real-time sensing response (Y/ΣCMYK: normalized yellow intensity) of different concentrations of parathion methyl (0 to 20 µg/mL) by performing the Ellman's assay with the Wi-Fi scanning method. Based on this result, 25 min was chosen as the assay time as it ensures both strong baseline signal and large difference among sensing responses of different concentrations of parathion methyl.

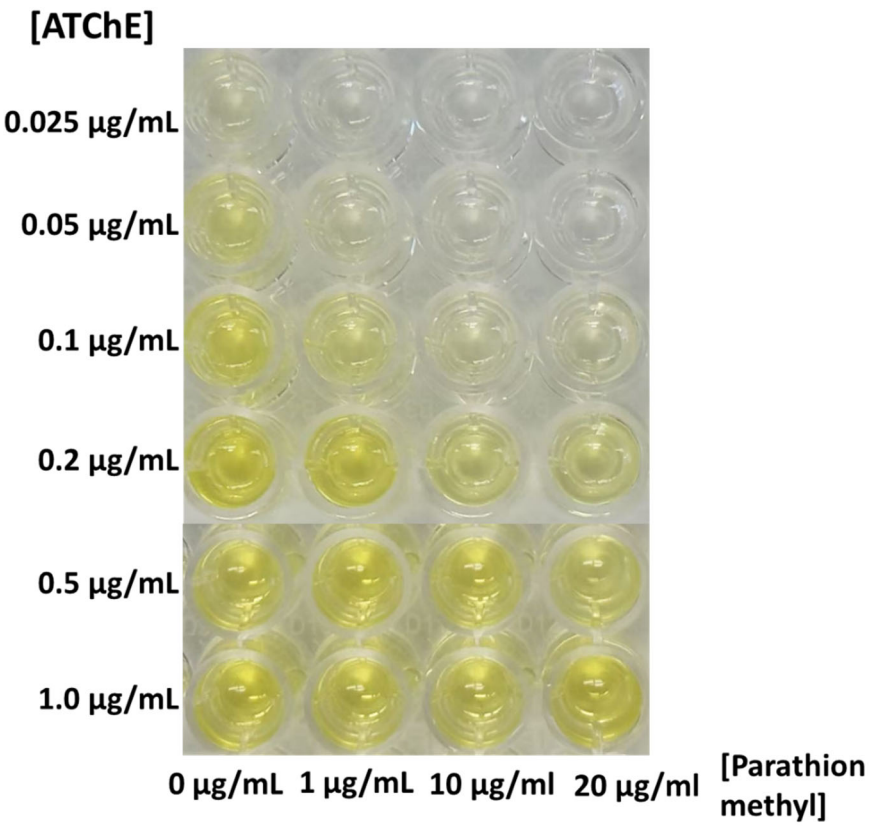

Fig. S12 Smartphone image of the Ellman's assay results for the detection of parathion methyl in microplate wells with different concentrations of ATChE. The reaction time was kept at 30 min.

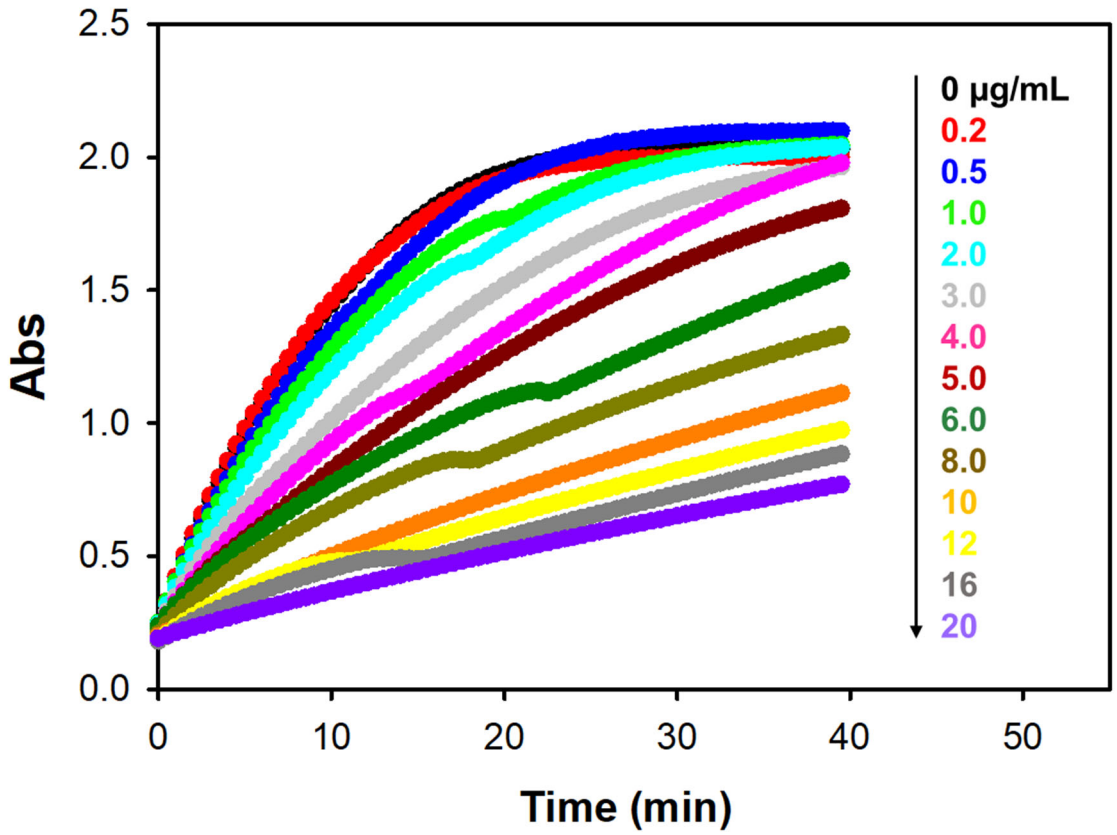

Fig. S13 Real-time sensing response of different concentrations of parathion methyl (0 to 20 µg/mL) by performing the Ellman's assay with conventional UV/Vis spectrophotometric measurements. Based on this data, 10 min was chosen as the assay time as it ensures both strong baseline signal and large difference among the sensing responses of different concentrations of parathion methyl.

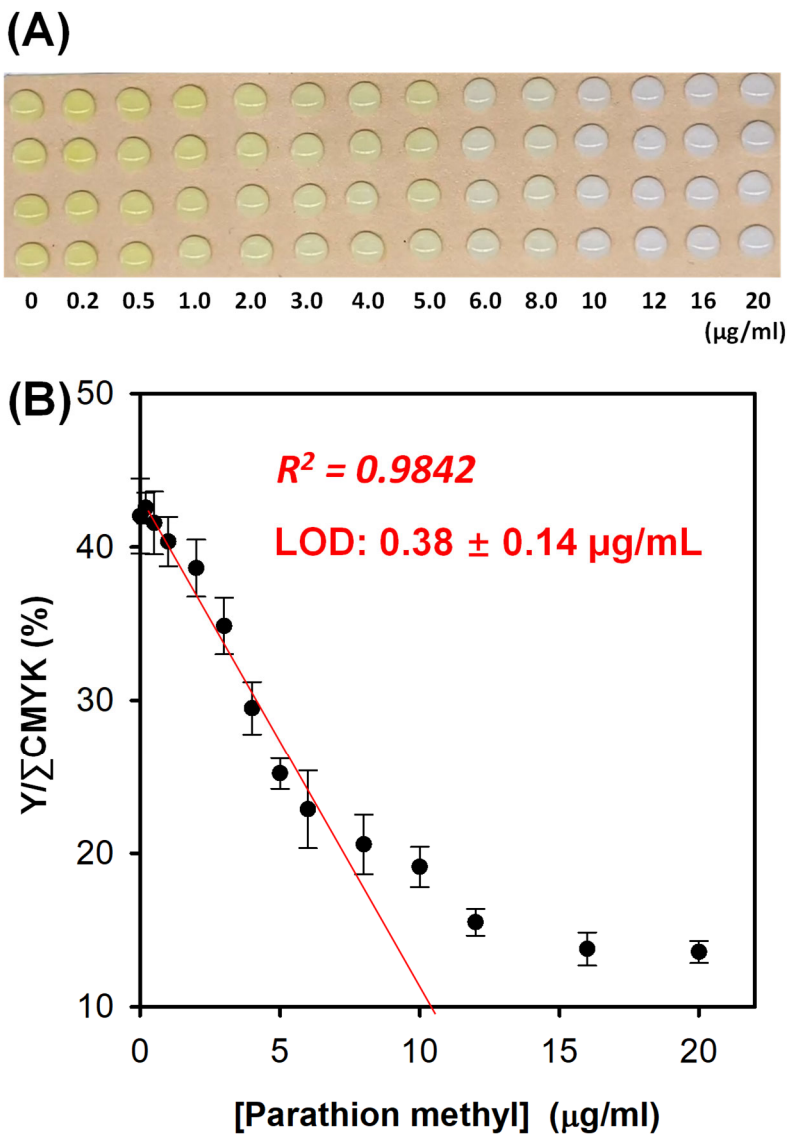

Fig. S14 Smartphone image of performing the Ellman's assay on hydrogel/paper disks (three repeats in 4 × 15 array) for the detection of parathion methyl (0-20 μg/mL). (B) Dependence of normalized Y value (Y/∑CMYK) on the concentration of parathion methyl; the red solid line shows the best linear fit to the experimental data up to 6 μg/mL. The LOD value was determined from the linear fit. The error bars in (B) were obtained from four repeating experiments.

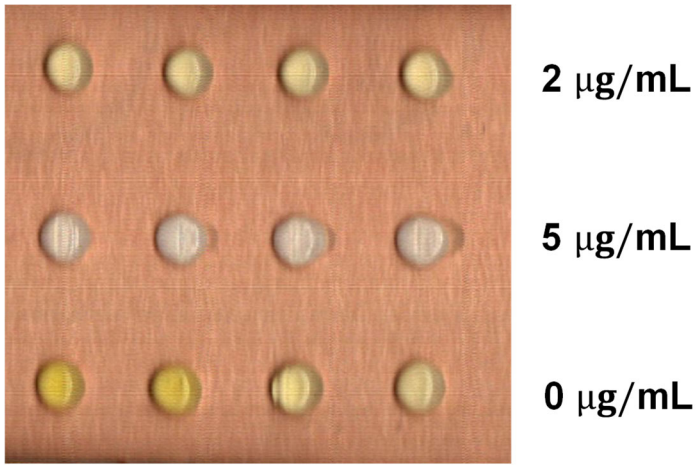

Fig. S15 Scanned image of parathion methyl detection results (0, 2, and 5 μg/mL) in tap water by performing the Ellman's assay with the Wi-Fi scanning method.

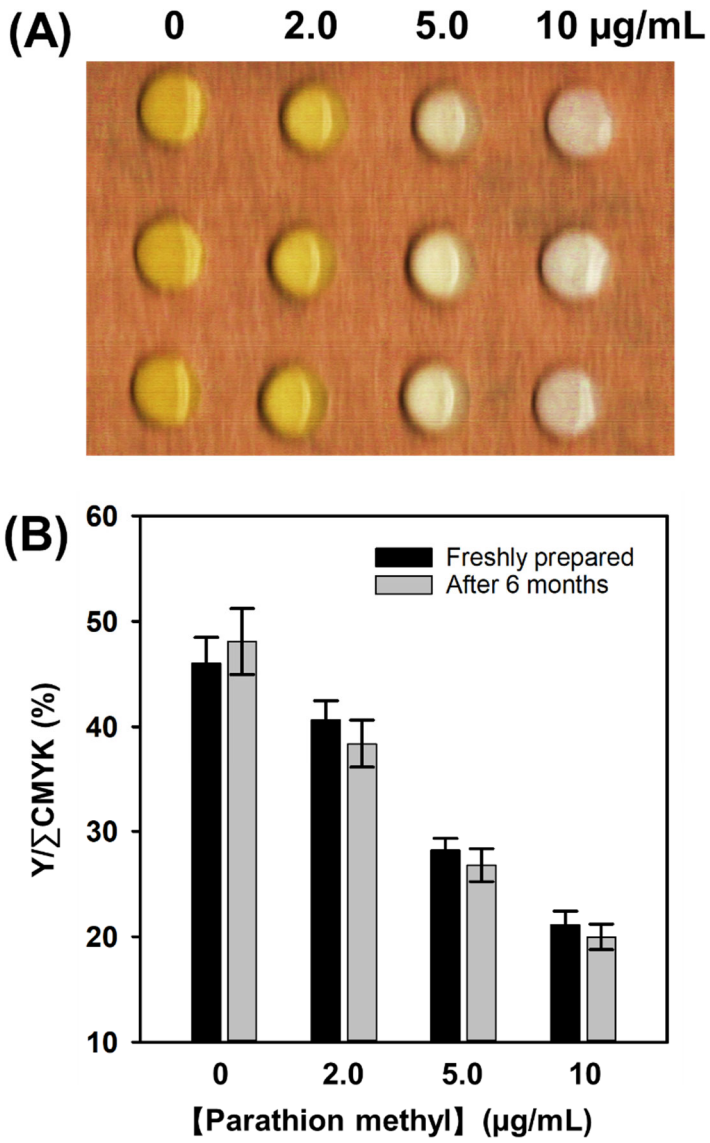

Fig. S16 (A) Scanned image of parathion methyl detection results (0, 2, 5 and 10 μg/mL) by performing the Ellman's assay with the hydrogel-paper disk after 6 months storage. (B) Compared parathion methyl detection results (0, 2, 5 and 10 μg/mL) by performing the Ellman's assay with the hydrogel-paper disk before (black bars) and after 6 months storage (gray bars). There is no discernible degradation of the hydrogen-paper disk arrays upon 6 months of storage, as the detection results are within the experimental uncertainties.

Original codes for the *ChromaDetect* app

```
<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="UTF-8" />
 <meta name="viewport" content="width=device-width, initial-scale=1.0, maximum-scale=1.0, user-scalable=no" />
 <title>ChromaDetect</title>
 <style>
  * {
   box-sizing: border-box;
  }

  body {
   font-family: 'Segoe UI', 'Arial', sans-serif;
   background: #f8f9fa;
   margin: 0;
   padding: 10px;
   color: #2c3e50;
   min-height: 100vh;
   line-height: 1.6;
   font-size: 16px;
  }

  h2 {
   color: #2c3e50;
   text-align: center;
   margin-bottom: 8px;
   font-size: 1.75em;
```

```
  font-weight: 600;
  font-family: 'Segoe UI', 'Arial', sans-serif;
  border-bottom: 2px solid #3498db;
  padding-bottom: 4px;
}

.container {
  max-width: 100%;
  margin: 0 auto;
  background: #ffffff;
  border-radius: 8px;
  padding: 8px;
  box-shadow: 0 2px 10px rgba(0,0,0,0.1);
  border: 1px solid #e9ecef;
}

.instructions {
  background: #ecf0f1;
  border: 2px solid #3498db;
  border-radius: 8px;
  padding: 15px;
  margin-top: 20px;
  color: #2c3e50;
  font-size: 12px;
}

.instructions h3 {
  margin-top: 0;
  color: #2c3e50;
  font-size: 1.2em;
```

```
  margin-bottom: 15px;
  text-align: center;
  font-weight: 600;
  font-family: 'Segoe UI', 'Arial', sans-serif;
}

.instructions ol {
  padding-left: 15px;
}

.instructions li {
  margin-bottom: 8px;
  line-height: 1.4;
  color: #34495e;
}

.instructions code {
  background: #ffffff;
  padding: 2px 4px;
  border-radius: 3px;
  font-family: 'Courier New', monospace;
  color: #e74c3c;
  border: 1px solid #fadbd8;
  font-weight: 600;
  font-size: 11px;
}

input[type="text"],
input[type="number"] {
  width: 100%;
```

```
  padding: 12px 14px;
  border: 2px solid #bdc3c7;
  border-radius: 6px;
  font-size: 16px;
  transition: all 0.3s ease;
  background: #ffffff;
  color: #2c3e50;
  font-family: 'Segoe UI', 'Arial', sans-serif;
  margin-bottom: 8px;
  min-height: 44px;
}

input[type="text"]:focus,
input[type="number"]:focus {
  outline: none;
  border-color: #3498db;
  box-shadow: 0 0 0 3px rgba(52, 152, 219, 0.1);
}

input[type="file"] {
  padding: 12px;
  margin: 8px 0;
  font-size: 14px;
  border: 2px dashed #3498db;
  border-radius: 8px;
  background: #f8f9fa;
  cursor: pointer;
  transition: all 0.3s ease;
  color: #2c3e50;
  font-family: 'Segoe UI', 'Arial', sans-serif;
```

```
  width: 100%;
}

input[type="file"]:hover {
  background: #ecf0f1;
  border-color: #2980b9;
}

button {
  padding: 12px 16px;
  font-size: 16px;
  cursor: pointer;
  margin: 5px 2px;
  background: #3498db;
  color: #ffffff;
  border: none;
  border-radius: 6px;
  transition: all 0.3s ease;
  font-weight: 500;
  box-shadow: 0 2px 8px rgba(52, 152, 219, 0.3);
  font-family: 'Segoe UI', 'Arial', sans-serif;
  min-height: 44px;
  touch-action: manipulation;
}

button:hover {
  background: #2980b9;
  transform: translateY(-1px);
  box-shadow: 0 4px 12px rgba(52, 152, 219, 0.4);
}
```

```
button:active {
  transform: translateY(0);
}

.btn-secondary {
  background: #95a5a6;
  box-shadow: 0 2px 8px rgba(149, 165, 166, 0.3);
}

.btn-secondary:hover {
  background: #7f8c8d;
  box-shadow: 0 4px 12px rgba(149, 165, 166, 0.4);
}

.btn-danger {
  background: #e74c3c;
  box-shadow: 0 2px 8px rgba(231, 76, 60, 0.3);
}

.btn-danger:hover {
  background: #c0392b;
  box-shadow: 0 4px 12px rgba(231, 76, 60, 0.4);
}

.btn-success {
  background: #27ae60;
  box-shadow: 0 2px 8px rgba(39, 174, 96, 0.3);
}
```

```
.btn-success:hover {
  background: #229954;
  box-shadow: 0 4px 12px rgba(39, 174, 96, 0.4);
}

#canvasContainer {
  margin: 8px 0;
  border: 2px solid #3498db;
  border-radius: 6px;
  display: block;
  width: 100%;
  max-height: 55vh;
  overflow: auto;
  background: #ffffff;
  box-shadow: 0 2px 8px rgba(0,0,0,0.1);
  min-height: 150px;
  aspect-ratio: auto;
}

canvas {
  display: block;
  background: #ffffff;
  width: 100%;
  height: auto;
  border-radius: 6px;
  touch-action: manipulation;
  min-height: 180px;
  min-width: 100px;
  object-fit: contain;
}
```

```
table {
  width: 100%;
  margin: 15px 0;
  border-collapse: collapse;
  background: #ffffff;
  border-radius: 8px;
  overflow: hidden;
  box-shadow: 0 4px 16px rgba(0,0,0,0.1);
  border: 1px solid #bdc3c7;
  font-family: 'Segoe UI', 'Arial', sans-serif;
  font-size: 14px;
  position: relative;
}
thead th {
  position: sticky;
  top: 0;
  z-index: 2;
  background: #34495e;
  color: #fff;
}

table, th, td {
  border: 1px solid #bdc3c7;
}

th {
  background: #34495e;
  color: #ffffff;
  padding: 12px 8px;
```

```
  font-weight: 600;
  font-size: 13px;
  font-family: 'Segoe UI', 'Arial', sans-serif;
  white-space: normal;
  line-height: 1.35;
  text-align: center;
}

td {
  padding: 10px 8px;
  font-size: 13px;
  color: #2c3e50;
  font-variant-numeric: tabular-nums;
  text-align: right;
  line-height: 1.35;
}

tr:nth-child(even) {
  background-color: #f4f6f7;
}

tr:hover {
  background-color: #e8f4fd;
}

#pointsTable th:first-child,
#pointsTable td:first-child {
  position: sticky;
  left: 0;
  background: #ffffff;
```

```
  z-index: 1;
  box-shadow: 2px 0 0 rgba(0,0,0,0.04);
  text-align: center;
}

#pointsTable thead th:first-child {
  z-index: 3;
}

.buttons-row {
  margin: 15px 0;
  display: flex;
  flex-direction: row;
  gap: 8px;
  padding: 15px;
  background: #f8f9fa;
  border-radius: 8px;
  border: 1px solid #bdc3c7;
  flex-wrap: wrap;
  align-items: center;
  justify-content: center;
}

.buttons-row p {
  margin: 0 0 8px 0;
  font-weight: 500;
  color: #2c3e50;
  font-family: 'Segoe UI', 'Arial', sans-serif;
  text-align: center;
}
```

```
.button-group {
  display: flex;
  flex-wrap: wrap;
  gap: 5px;
  justify-content: center;
}

.button-group button {
  flex: 1;
  min-width: 120px;
  max-width: 200px;
  min-height: 44px;
  font-size: 16px;
}

.table-responsive {
  width: 100%;
  overflow: auto;
  margin-top: 8px;
  border-radius: 6px;
  box-shadow: 0 2px 8px rgba(0,0,0,0.1);
  -webkit-overflow-scrolling: touch;
  max-height: 50vh;
}

#pointsTable {
  min-width: 800px;
}
```

```
#pointsTable th, #pointsTable td {
  padding: 10px 8px;
  font-size: 13px;
}

#pointsTable input {
  width: 50px;
  padding: 4px 6px;
  border: 1px solid #bdc3c7;
  border-radius: 3px;
  font-size: 10px;
  background: #ffffff;
  color: #2c3e50;
  font-family: 'Segoe UI', 'Arial', sans-serif;
}

#pointsTable button {
  padding: 4px 8px;
  font-size: 10px;
  margin: 0;
  min-height: 30px;
}

.equation-editor {
  margin-top: 15px;
  padding: 15px;
  background: #ecf0f1;
  border-radius: 8px;
  border: 2px solid #bdc3c7;
  box-shadow: 0 2px 8px rgba(0,0,0,0.08);
```

```
}

.equation-editor h4 {
  margin: 0 0 15px 0;
  color: #2c3e50;
  font-size: 1.1em;
  text-align: center;
  font-weight: 600;
  font-family: 'Segoe UI', 'Arial', sans-serif;
}

.equation-inputs {
  display: flex;
  flex-direction: column;
  gap: 10px;
  margin-bottom: 10px;
}

.equation-inputs input {
  width: 100%;
  text-align: center;
  max-width: 200px;
  margin: 0 auto;
}

.equation-inputs label {
  color: #2c3e50;
  font-weight: 500;
  font-family: 'Segoe UI', 'Arial', sans-serif;
  text-align: center;
```

```
  font-size: 12px;
}

.equation-status {
  text-align: center;
  font-weight: 500;
  padding: 10px;
  background: #d5f4e6;
  border-radius: 6px;
  border: 1px solid #a9dfbf;
  color: #1e8449;
  font-family: 'Segoe UI', 'Arial', sans-serif;
  font-size: 11px;
}

.file-status {
  padding: 8px 12px;
  border-radius: 6px;
  font-weight: 500;
  margin: 5px 0;
  font-family: 'Segoe UI', 'Arial', sans-serif;
  font-size: 12px;
  text-align: center;
}

.file-status.success {
  background: #d5f4e6;
  color: #1e8449;
  border: 1px solid #a9dfbf;
}
```

```
.file-status.error {
  background: #fadbd8;
  color: #c0392b;
  border: 1px solid #f1948a;
}

.file-status.info {
  background: #d6eaf8;
  color: #2980b9;
  border: 1px solid #85c1e9;
}

.scientific-note {
  background: #fef9e7;
  border: 1px solid #f7dc6f;
  border-radius: 6px;
  padding: 12px;
  margin: 12px 0;
  color: #7d6608;
  font-style: italic;
  font-family: 'Segoe UI', 'Arial', sans-serif;
  font-size: 11px;
}

.scientific-notes-section {
  margin-top: 30px;
  padding: 15px;
  background: #f8f9fa;
  border-radius: 8px;
```

```
  border: 1px solid #bdc3c7;
}

.scientific-notes-section h3 {
  color: #2c3e50;
  font-size: 1.2em;
  margin-bottom: 15px;
  text-align: center;
  font-weight: 600;
  font-family: 'Segoe UI', 'Arial', sans-serif;
  border-bottom: 2px solid #3498db;
  padding-bottom: 8px;
}

.settings-tab {
  background: #ecf0f1;
  border: 1px solid #3498db;
  border-radius: 6px;
  padding: 8px;
  margin: 4px 0;
  color: #2c3e50;
  display: flex;
  flex-direction: row;
  gap: 16px;
  flex-wrap: wrap;
}
.settings-tab h3 {
  margin-top: 0;
  color: #2c3e50;
  font-size: 1.0em;
```

```
  margin-bottom: 8px;
  text-align: center;
  font-weight: 600;
  font-family: 'Segoe UI', 'Arial', sans-serif;
  border-bottom: 1px solid #3498db;
  padding-bottom: 4px;
  width: 100%;
}
.settings-row {
  flex: 1 1 200px;
  min-width: 160px;
}
@media (max-width: 700px) {
  .settings-tab {
    flex-direction: column;
    gap: 10px;
  }
  .settings-row {
    min-width: unset;
  }
}

.settings-row label {
  font-weight: 600;
  color: #2c3e50;
  text-align: center;
}

.checkbox-group {
  display: flex;
```

```
  flex-direction: row;
  gap: 18px;
  align-items: center;
  flex-wrap: wrap;
}

.checkbox-item {
  display: flex;
  align-items: center;
  gap: 8px;
}

.checkbox-item input[type="checkbox"] {
  width: 18px;
  height: 18px;
  cursor: pointer;
}

.checkbox-item label {
  font-weight: 500;
  color: #2c3e50;
  cursor: pointer;
  font-size: 12px;
}

input[type="radio"] {
  width: 18px;
  height: 18px;
  cursor: pointer;
  accent-color: #3498db;
```

```
  margin: 0;
}

input[type="radio"] + label,
label:has(input[type="radio"]) {
  font-weight: 500;
  color: #2c3e50;
  cursor: pointer;
  user-select: none;
}

.column-hidden {
  display: none !important;
}

.data-label {
  font-weight: 600;
  color: #2c3e50;
  font-family: 'Segoe UI', 'Arial', sans-serif;
}

@media (max-width: 768px) {
  body {
    padding: 5px;
    font-size: 13px;
  }

  .container {
    padding: 10px;
    margin: 5px;
```

```
}

h2 {
  font-size: 1.5em;
  margin-bottom: 15px;
}

.buttons-row {
  padding: 10px;
  flex-direction: column;
  gap: 10px;
}

.button-group {
  flex-direction: column;
  gap: 8px;
  width: 100%;
}

.button-group button {
  width: 100%;
  min-width: unset;
  max-width: unset;
}

input[type="text"],
input[type="number"] {
  width: 100%;
  font-size: 16px;
  padding: 14px;
```

```
  min-height: 44px;
}

#canvasContainer {
  max-height: 50vh;
  min-height: 180px;
}

table {
  font-size: 10px;
}

th, td {
  padding: 4px 2px;
  font-size: 9px;
}

#pointsTable {
  min-width: 600px;
}

#pointsTable input {
  width: 40px;
  font-size: 9px;
  padding: 3px;
  min-height: 32px;
}

.equation-inputs input {
  max-width: 150px;
```

```
  }

  .scientific-note {
    font-size: 10px;
    padding: 10px;
  }
}

@media (max-width: 480px) {
  body {
    padding: 3px;
    font-size: 12px;
  }

  .container {
    padding: 8px;
  }

  h2 {
    font-size: 1.3em;
  }

  #canvasContainer {
    max-height: 40vh;
    min-height: 120px;
  }

  #pointsTable {
    min-width: 500px;
  }
```

```
  th, td {
    padding: 3px 1px;
    font-size: 8px;
  }

  #pointsTable input {
    width: 35px;
    font-size: 8px;
    padding: 2px;
    min-height: 28px;
  }

  button {
    font-size: 14px;
    min-height: 44px;
  }
}

button, input[type="file"], input[type="text"], input[type="number"] {
  -webkit-tap-highlight-color: transparent;
}

input[type="text"], input[type="number"] {
  font-size: 16px;
}

.table-responsive {
  scroll-behavior: smooth;
}
```

```
.equation-inline {
  display: flex;
  flex-direction: row;
  align-items: center;
  flex-wrap: wrap;
  gap: 6px;
}
@media (max-width: 600px) {
  .equation-inline {
    flex-direction: column;
    align-items: stretch;
    gap: 8px;
  }
  .equation-inline label, .equation-inline input, .equation-inline button {
    width: 100% !important;
    max-width: 100%;
  }
}

.control-panel {
  position: sticky;
  top: 0;
  z-index: 10;
  background: #fff;
  padding: 4px 0 2px 0;
  box-shadow: 0 1px 4px rgba(52,152,219,0.08);
  border-bottom: 1px solid #e9ecef;
  display: flex;
  flex-wrap: wrap;
```

```
  gap: 6px;
  align-items: center;
  justify-content: flex-start;
}
.control-panel input[type="text"],
.control-panel input[type="number"],
.control-panel input[type="file"] {
  margin: 0 2px;
  min-width: 70px;
  max-width: 100px;
  font-size: 12px;
  min-height: 28px;
}
.control-panel button {
  margin: 0 2px;
  min-width: 60px;
  min-height: 28px;
  font-size: 12px;
  padding: 6px 10px;
}
.control-panel .equation-inline {
  margin: 0 8px;
}
@media (max-width: 900px) {
  .main-flex {
    flex-direction: column;
  }
  .canvas-col, .table-col {
    width: 100%;
    min-width: unset;
```

```
  }
}
.main-flex {
  display: flex;
  flex-direction: row;
  gap: 16px;
  margin-top: 8px;
  max-height: 60vh;
}
.canvas-col {
  flex: 1 1 300px;
  min-width: 200px;
  max-height: 60vh;
}
.table-col {
  flex: 2 1 400px;
  min-width: 300px;
  max-height: 60vh;
  overflow: hidden;
}
.collapsible-section {
  margin-top: 8px;
  margin-bottom: 4px;
}
.collapsible-section details {
  margin-bottom: 4px;
}
.equation-inline input[type="number"] {
  width: 110px !important;
  min-width: 80px;
```

```
    max-width: 140px;
    margin: 0 4px;
   }
  </style>
</head>
<body>
  <div class="container">
   <h2>ChromaDetect</h2>


   <div class="control-panel" style="flex-direction:column; align-items:stretch; gap:4px;">
     <div style="display:flex; flex-wrap:wrap; align-items:center; gap:8px; margin-bottom:2px;">
     <input type="file" id="fileInput" accept="image/*" />
     <input type="text" id="filename" placeholder="Enter image filename" />
     <button onclick="downloadImageFromScanner()" class="btn-success">Download</button>
     <span id="fileCheckResult" class="file-status"></span>
    </div>
     <div style="display:flex; flex-wrap:wrap; align-items:center; gap:8px; margin-bottom:2px;">
     <input type="number" id="selectionRadius" placeholder="Radius" value="10" style="width:90px;" />
     <button onclick="addPoint()" class="btn-secondary">Add Point</button>
     <button onclick="clearAllPoints()" class="btn-danger">Clear All</button>
     <button onclick="getRGB()" class="btn-success">Calculate</button>
     <button onclick="downloadResultImage()" class="btn-secondary">Export Image</button>
     <button onclick="exportToXLSX()" class="btn-secondary">Export Data</button>
    </div>
   </div>

   <div class="main-flex">
    <div class="canvas-col">
     <div id="canvasContainer">
```

```
    <canvas id="canvas"></canvas>
  </div>
</div>
<div class="table-col">
  <div class="table-responsive">
    <table id="pointsTable">
      <colgroup>
        <col style="width: 4%;">
        <col style="width: 6%;">
        <col style="width: 6%;">
        <col style="width: 6%;">
        <col style="width: 6%;">
        <col style="width: 6%;">
        <col style="width: 8%;">
        <col style="width: 6%;">
        <col style="width: 6%;">
        <col style="width: 6%;">
        <col style="width: 6%;">
        <col style="width: 6%;">
        <col style="width: 6%;">
        <col style="width: 6%;">
        <col style="width: 6%;">
        <col style="width: 6%;">
        <col style="width: 6%;">
        <col style="width: 6%;">
        <col style="width: 6%;">
        <col style="width: 6%;">
        <col style="width: 6%;">
        <col style="width: 6%;">
        <col style="width: 6%;">
      </colgroup>
```

```
<thead>
 <tr>
  <th class="col-point">Point</th>
  <th class="col-xa">XA</th>
  <th class="col-ya">YA</th>
  <th class="col-xb">XB</th>
  <th class="col-yb">YB</th>
 <th class="col-radius">Radius</th>
  <th class="col-mode">Mode</th>
  <th class="col-rgbACombined rgb-combined-header">RGB<sub>A</sub></th>
  <th class="col-rgbBCombined rgb-combined-header">RGB<sub>B</sub></th>
  <th class="col-cmykACombined cmyk-combined-header column-hidden">CMYK<sub>A</sub></th>
  <th class="col-cmykBCombined cmyk-combined-header column-hidden">CMYK<sub>B</sub></th>
  <th class="col-ra raHeader">R<sub>A</sub></th>
  <th class="col-raSD raSDHeader">SD(R<sub>A</sub>)</th>
  <th class="col-ga gaHeader">G<sub>A</sub></th>
  <th class="col-gaSD gaSDHeader">SD(G<sub>A</sub>)</th>
  <th class="col-ba baHeader">B<sub>A</sub></th>
  <th class="col-baSD baSDHeader">SD(B<sub>A</sub>)</th>
  <th class="col-ca caHeader column-hidden">C<sub>A</sub></th>
  <th class="col-caSD caSDHeader column-hidden">SD(C<sub>A</sub>)</th>
  <th class="col-ma maHeader column-hidden">M<sub>A</sub></th>
  <th class="col-maSD maSDHeader column-hidden">SD(M<sub>A</sub>)</th>
  <th class="col-ya2 yaHeader column-hidden">Y<sub>A</sub></th>
  <th class="col-ya2SD yaSDHeader column-hidden">SD(Y<sub>A</sub>)</th>
  <th class="col-ka kaHeader column-hidden">K<sub>A</sub></th>
  <th class="col-kaSD kaSDHeader column-hidden">SD(K<sub>A</sub>)</th>
  <th class="col-rb rbHeader">R<sub>B</sub></th>
  <th class="col-rbSD rbSDHeader">SD(R<sub>B</sub>)</th>
  <th class="col-gb gbHeader">G<sub>B</sub></th>
```

<th class="col-gbSD gbSDHeader">SD(G_B)</th>

<th class="col-bb bbHeader">B_B</th>

<th class="col-bbSD bbSDHeader">SD(B_B)</th>

<th class="col-cb cbHeader column-hidden">C_B</th>

<th class="col-cbSD cbSDHeader column-hidden">SD(C_B)</th>

<th class="col-mb mbHeader column-hidden">M_B</th>

<th class="col-mbSD mbSDHeader column-hidden">SD(M_B)</th>

<th class="col-yb2 ybHeader column-hidden">Y_B</th>

<th class="col-yb2SD ybSDHeader column-hidden">SD(Y_B)</th>

<th class="col-kb kbHeader column-hidden">K_B</th>

<th class="col-kbSD kbSDHeader column-hidden">SD(K_B)</th>

<th class="col-ran">R'_A</th>

<th class="col-gan">G'_A</th>

<th class="col-ban">B'_A</th>

<th class="col-rbn">R'_B</th>

<th class="col-gbn">G'_B</th>

<th class="col-bbn">B'_B</th>

<th class="col-can column-hidden">C'_A</th>

<th class="col-man column-hidden">M'_A</th>

<th class="col-yan column-hidden">Y'_A</th>

<th class="col-kan column-hidden">K'_A</th>

<th class="col-cbn column-hidden">C'_B</th>

<th class="col-mbn column-hidden">M'_B</th>

<th class="col-ybn column-hidden">Y'_B</th>

<th class="col-kbn column-hidden">K'_B</th>

<th class="col-rrgba">R'/RGB' (A)</th>

<th class="col-rrgbb">R'/RGB' (B)</th>

<th class="col-rrgb">R'/RGB' (A-B)</th>

<th class="col-ccmyka column-hidden">C'/CMYK' (A)</th>

<th class="col-ccmykb column-hidden">C'/CMYK' (B)</th>

```
      <th class="col-ccmyk column-hidden">C'/CMYK' (A-B)</th>
      <th class="col-mcmyka column-hidden">M'/CMYK' (A)</th>
      <th class="col-mcmykb column-hidden">M'/CMYK' (B)</th>
      <th class="col-mcmyk column-hidden">M'/CMYK' (A-B)</th>
      <th class="col-ycmyka column-hidden">Y'/CMYK' (A)</th>
      <th class="col-ycmykb column-hidden">Y'/CMYK' (B)</th>
      <th class="col-ycmyk column-hidden">Y'/CMYK' (A-B)</th>
      <th class="col-kcmyka column-hidden">K'/CMYK' (A)</th>
      <th class="col-kcmykb column-hidden">K'/CMYK' (B)</th>
      <th class="col-kcmyk column-hidden">K'/CMYK' (A-B)</th>
      <th class="col-grgba">G'/RGB' (A)</th>
      <th class="col-grgbb">G'/RGB' (B)</th>
      <th class="col-grgb">G'/RGB' (A-B)</th>
      <th class="col-brgba">B'/RGB' (A)</th>
      <th class="col-brgbb">B'/RGB' (B)</th>
      <th class="col-brgb">B'/RGB' (A-B)</th>
      <th class="col-anth">Concentration</th>
      <th class="col-remove">Remove</th>
    </tr>
   </thead>
   <tbody id="pointsBody">

   </tbody>
  </table>
</div>
<div aria-label="Legend" style="margin-top:4px; font-size:10px; color:#4a5568;">
  <strong>Legend:</strong>
  <span style="margin-left:4px;">A: sample</span>
  <span style="margin-left:4px;">B: reference</span>
  <span style="margin-left:4px;">R', G', B': normalized</span>
```

```
    <span style="margin-left:4px;">R'/RGB': red ratio</span>
   </div>
  </div>
</div>

<div class="collapsible-section">
  <details open>
   <summary><b>Settings</b></summary>
   <div class="settings-tab">
    <h3>Table Display Settings</h3>
    <div class="settings-row">
     <label>Column Visibility:</label>
     <div class="checkbox-group">
      <div class="checkbox-item">
       <input type="checkbox" id="showRawRGBColumns" checked>
       <label for="showRawRGBColumns">Point Value</label>
      </div>
      <div class="checkbox-item">
       <input type="checkbox" id="showRGBColumns" checked>
       <label for="showRGBColumns">Normalized</label>
      </div>
      <div class="checkbox-item">
       <input type="checkbox" id="showRGBRatioColumns" checked>
       <label for="showRGBRatioColumns">Ratio</label>
      </div>
      <div class="checkbox-item">
       <input type="checkbox" id="showCoordColumns" checked>
       <label for="showCoordColumns">Coordinate</label>
      </div>
     </div>
```

```
      </div>
      <div class="settings-row">
        <label style="font-weight:600; margin-right:8px;">Color Space:</label>
        <div style="display: inline-flex; gap: 12px; align-items: center;">
          <label style="font-weight: normal; display: inline-flex; align-items: center; cursor: pointer;">
            <input type="radio" name="colorMode" value="rgb" id="colorModeRGB" checked style="margin-right: 6px; cursor: pointer;">
            RGB
          </label>
          <label style="font-weight: normal; display: inline-flex; align-items: center; cursor: pointer;">
            <input type="radio" name="colorMode" value="cmyk" id="colorModeCMYK" style="margin-right: 6px; cursor: pointer;">
            CMYK
          </label>
        </div>
        <label style="font-weight:600; margin-left:16px; margin-right:8px;">Mode:</label>
        <div style="display: inline-flex; gap: 12px; align-items: center;">
          <label style="font-weight: normal; display: inline-flex; align-items: center; cursor: pointer;">
            <input type="radio" name="globalMode" value="ab" id="globalModeAB" checked style="margin-right: 6px; cursor: pointer;">
            A-B
          </label>
          <label style="font-weight: normal; display: inline-flex; align-items: center; cursor: pointer;">
            <input type="radio" name="globalMode" value="a-only" id="globalModeAOnly" style="margin-right: 6px; cursor: pointer;">
            Single point
          </label>
        </div>
      </div>
    </div>
    <div class="settings-tab">
      <h3>Calibration Equation</h3>
```

```
      <div class="settings-row">
        <div class="equation-inline" style="display: flex; align-items: center; gap: 8px; flex-wrap: wrap; margin:10px
0;">
          <label id="equationDescription" style="margin:0; margin-right:8px;">Color signal = MX+C</label>
          <label style="margin:0; margin-left:16px; margin-right:4px;">M=</label>
          <input type="number" id="anthEquationB" placeholder="0.0026" step="0.0001" style="width:100px;" />
          <label style="margin:0; margin-right:4px;">C=</label>
          <input type="number" id="anthEquationA" placeholder="0.0061" step="0.0001" style="width:100px;" />
          <label for="anthEquation" id="equationInputLabel" style="display:none;"></label>
          <button onclick="saveAnthEquation()" class="btn-success" style="margin-right:4px;">Apply</button>
          <button onclick="resetAnthEquation()" class="btn-secondary">Reset</button>
        </div>
        <div class="equation-inline" style="display: flex; align-items: center; gap: 8px; flex-wrap: wrap; margin:10px
0;">
          <label style="font-weight:600; margin-right:8px;">Channel:</label>
          <div id="channelRadioGroup" style="display: inline-flex; gap: 12px; align-items: center;">

          </div>
        </div>
        <div id="equationStatus" class="equation-status" style="margin-top:10px;"></div>
      </div>
    </details>
    <details>
      <summary><b>Instructions</b></summary>
      <div class="instructions">
        <h3>Experimental Protocol</h3>
        <ol>
          <li>
            Enter the spectral image filename (e.g., <code>IMAG0001.JPG</code>) and click
            "Download from Scanner" to retrieve the image from the spectrometer.
```

```
    </li>
    <li>Upload the spectral image using the file input below for RGB spectroscopy analysis.</li>
    <li>
      Tap on the image to establish sampling regions. The sampling radius determines
      the area of spectral analysis for each measurement point.
    </li>
    <li>
      Selected regions are marked with colored indicators (A: blue, B: green) and
      spectral data is tabulated below with normalized RGB values and derived parameters.
    </li>
    <li>
      The data table displays coordinate inputs, normalized RGB spectra (R', G', B'),
      red channel ratios (R'/RGB'), and calculated anthocyanin concentrations.
      Coordinates can be adjusted manually for precise positioning.
    </li>
    <li>
      Individual data points can be removed using the "Remove" button, or all
      experimental data can be cleared for new analysis.
    </li>
  </ol>
 </div>
</details>
<details>
 <summary><b>Scientific Notes</b></summary>
 <div class="scientific-notes-section">
  <h3>Scientific Notes</h3>

  <div class="scientific-note">
   <strong>Scientific Note:</strong> This tool utilizes RGB spectroscopy for anthocyanin concentration analysis.
```

The method is based on the differential absorption of red, green, and blue wavelengths by anthocyanin compounds.

Ensure proper calibration and consistent lighting conditions for accurate measurements.

Calibration Note: These parameters are derived from empirical calibration curves.

Adjust based on your specific experimental conditions and reference standards.

R'/RGB' represents the normalized red channel intensity ratio.

Methodology: This analysis employs RGB spectroscopy for anthocyanin quantification.

The method utilizes differential absorption of visible wavelengths by anthocyanin compounds,

with the red channel (R') providing the primary spectral signature for concentration determination.

Equation Methodology Note: The calculation of concentration begins with the acquisition of RGB color values from selected regions of the spectral image. For each region, the average red (R), green (G), and blue (B) channel intensities are measured and normalized (R', G', B'). The normalized red channel ratio (R'/RGB') is then computed for both sample (A) and reference (B) regions. The difference, R'/RGB' (A-B), serves as the analytical signal. Concentration is determined using the calibration equation:

Concentration = (R'/RGB' (A-B) − A) / B

where A and B are empirically derived calibration parameters. This approach leverages the specific absorbance characteristics of compounds in the visible spectrum, as reflected in the red channel response.

Normalized RGB Calculation Note: To obtain normalized RGB values (R', G', B'), the average red (R), green (G), and blue (B) intensities from the selected region are first measured. Each channel is then divided by the sum of all three channels to normalize the values, as follows:

R' = R / (R + G + B)

```
        <strong>G' = G / (R + G + B)</strong><br>
        <strong>B' = B / (R + G + B)</strong><br>
        This normalization ensures that the sum of R', G', and B' equals 1, allowing for direct comparison of color ratios independent of overall brightness.
      </div>

      <div class="scientific-note">
        <strong>RGB' Definition Note:</strong> In this analysis, RGB' refers to the sum of the normalized red, green, and blue channel values:<br>
        <strong>RGB' = R' + G' + B' = 1</strong><br>
        where R', G', and B' are the normalized values calculated as:<br>
        <strong>R' = R / (R + G + B)</strong><br>
        <strong>G' = G / (R + G + B)</strong><br>
        <strong>B' = B / (R + G + B)</strong><br>
        The denominator (R + G + B) represents the total intensity of the selected region, and normalization ensures that the sum of R', G', and B' is always 1. This allows for direct comparison of color ratios, independent of overall brightness.
      </div>
    </div>
  </details>
 </div>
</div>

<script>

  const elements = {
    canvas: document.getElementById("canvas"),
    pointsBody: document.getElementById("pointsBody"),
    filenameInput: document.getElementById("filename"),
    fileInput: document.getElementById("fileInput"),
    selectionRadiusInput: document.getElementById("selectionRadius"),
    anthEquationAInput: document.getElementById("anthEquationA"),
```

```
  anthEquationBInput: document.getElementById("anthEquationB"),
  equationStatus: document.getElementById("equationStatus"),
  pointsTable: document.getElementById("pointsTable"),
  fileCheckResult: document.getElementById("fileCheckResult"),
  showRGBColumns: document.getElementById("showRGBColumns"),
  showRGBRatioColumns: document.getElementById("showRGBRatioColumns"),
  showRawRGBColumns: document.getElementById("showRawRGBColumns"),
  showCoordColumns: document.getElementById("showCoordColumns"),
};

let canvas = null;
let ctx = null;
let pointsBody = elements.pointsBody;
let currentImage = null;

const pHData = [
  { g_rgb: 0.310, pH: 3.0 },
  { g_rgb: 0.320, pH: 4.05 },
  { g_rgb: 0.325, pH: 5.04 },
  { g_rgb: 0.330, pH: 6.03 },
  { g_rgb: 0.335, pH: 6.99 },
  { g_rgb: 0.340, pH: 8.0 }
];

let anthEquationA = 0.0061;
let anthEquationB = 0.0026;
let anthChannel = 'rrgb';

function getGlobalMode() {
  const checked = document.querySelector('input[name="globalMode"]:checked');
```

```
  return checked ? checked.value : (localStorage.getItem('globalMode') || 'ab');
}
function getGlobalModeLabel() {
  return getGlobalMode() === 'ab' ? 'A-B' : 'Single point';
}

function loadAnthEquation() {
  const savedA = localStorage.getItem('anthEquationA');
  const savedB = localStorage.getItem('anthEquationB');
  const savedChannel = localStorage.getItem('anthChannel');

  if (savedA !== null) anthEquationA = parseFloat(savedA);
  if (savedB !== null) anthEquationB = parseFloat(savedB);
  if (savedChannel !== null) anthChannel = savedChannel;

  elements.anthEquationAInput.value = anthEquationA;
  elements.anthEquationBInput.value = anthEquationB;

  updateChannelOptions();
  updateEquationStatus();
}

function loadColumnVisibilitySettings() {
  const savedRawRGBColumns = localStorage.getItem('showRawRGBColumns');
  const savedRGBColumns = localStorage.getItem('showRGBColumns');
  const savedRGBRatioColumns = localStorage.getItem('showRGBRatioColumns');
  const savedCoordColumns = localStorage.getItem('showCoordColumns');

  if (savedRawRGBColumns !== null) {
    elements.showRawRGBColumns.checked = savedRawRGBColumns === 'true';
```

```
  }
  if (savedRGBColumns !== null) {
    elements.showRGBColumns.checked = savedRGBColumns === 'true';
  }
  if (savedRGBRatioColumns !== null) {
    elements.showRGBRatioColumns.checked = savedRGBRatioColumns === 'true';
  }
  if (savedCoordColumns !== null) {
    elements.showCoordColumns.checked = savedCoordColumns === 'true';
  }

  updateColumnVisibility();
}

function saveColumnVisibilitySettings() {
  localStorage.setItem('showRawRGBColumns', elements.showRawRGBColumns.checked.toString());
  localStorage.setItem('showRGBColumns', elements.showRGBColumns.checked.toString());
  localStorage.setItem('showRGBRatioColumns', elements.showRGBRatioColumns.checked.toString());
  localStorage.setItem('showCoordColumns', elements.showCoordColumns.checked.toString());
}

function updateColumnVisibility() {
  const showRawRGB = elements.showRawRGBColumns.checked;
  const showRGB = elements.showRGBColumns.checked;
  const showRGBRatio = elements.showRGBRatioColumns.checked;
  const showCoord = elements.showCoordColumns.checked;
  const colorMode = document.querySelector('input[name="colorMode"]:checked')?.value || 'rgb';
  const isSinglePoint = (typeof getGlobalMode === 'function') ? (getGlobalMode() === 'a-only') : false;

  document.querySelectorAll('.col-point').forEach(el => el.classList.remove('column-hidden'));
```

```
    ['col-xa','col-ya','col-xb','col-yb'].forEach(cls => {
      document.querySelectorAll('.'+cls).forEach(el => el.classList.toggle('column-hidden', !showCoord));
    });
    ['col-rgbACombined','col-rgbBCombined'].forEach(cls => {
      const hide = colorMode==='cmyk'||!showRawRGB || (cls==='col-rgbBCombined' && isSinglePoint);
      document.querySelectorAll('.'+cls).forEach(el => el.classList.toggle('column-hidden', hide));
    });
    ['col-cmykACombined','col-cmykBCombined'].forEach(cls => {
      const hide = colorMode!=='cmyk' || (cls==='col-cmykBCombined' && isSinglePoint);
      document.querySelectorAll('.'+cls).forEach(el => el.classList.toggle('column-hidden', hide));
    });
  ['col-ra','col-raSD','col-ga','col-gaSD','col-ba','col-baSD','col-rb','col-rbSD','col-gb','col-gbSD','col-bb','col-
bbSD'].forEach(cls => {
    const isB = ['col-rb','col-rbSD','col-gb','col-gbSD','col-bb','col-bbSD'].includes(cls);
    const hide = !showRawRGB||colorMode==='cmyk' || (isB && isSinglePoint);
    document.querySelectorAll('.'+cls).forEach(el => el.classList.toggle('column-hidden', hide));
  });
  ['col-ca','col-caSD','col-ma','col-maSD','col-ya2','col-ya2SD','col-ka','col-kaSD','col-cb','col-cbSD','col-mb','col-
mbSD','col-yb2','col-yb2SD','col-kb','col-kbSD'].forEach(cls => {
    const isB = ['col-cb','col-cbSD','col-mb','col-mbSD','col-yb2','col-yb2SD','col-kb','col-kbSD'].includes(cls);
    const hide = !showRawRGB||colorMode!=='cmyk' || (isB && isSinglePoint);
    document.querySelectorAll('.'+cls).forEach(el => el.classList.toggle('column-hidden', hide));
  });
  ['col-ran','col-gan','col-ban','col-rbn','col-gbn','col-bbn','col-rrgba','col-rrgbb','col-rrgb','col-grgba','col-grgbb','col-
grgb','col-brgba','col-brgbb','col-brgb'].forEach(cls => {
    const isB = ['col-rbn','col-gbn','col-bbn','col-rrgbb','col-grgbb','col-brgbb'].includes(cls);
    const isDiff = ['col-rrgb','col-grgb','col-brgb'].includes(cls);
    const hide = !showRGB||colorMode==='cmyk' || (isB && isSinglePoint) || (isDiff && isSinglePoint);
    document.querySelectorAll('.'+cls).forEach(el => el.classList.toggle('column-hidden', hide));
  });
```

```
  ['col-can','col-man','col-yan','col-kan','col-cbn','col-mbn','col-ybn','col-kbn','col-ccmyka','col-ccmykb','col-
ccmyk','col-mcmyka','col-mcmykb','col-mcmyk','col-ycmyka','col-ycmykb','col-ycmyk','col-kcmyka','col-kcmykb','col-
kcmyk'].forEach(cls => {
    const isB = ['col-cbn','col-mbn','col-ybn','col-kbn','col-ccmykb','col-mcmykb','col-ycmykb','col-
kcmykb'].includes(cls);
    const isDiff = ['col-ccmyk','col-mcmyk','col-ycmyk','col-kcmyk'].includes(cls);
    const hide = !showRGB||colorMode!=='cmyk' || (isB && isSinglePoint) || (isDiff && isSinglePoint);
    document.querySelectorAll('.'+cls).forEach(el => el.classList.toggle('column-hidden', hide));
  });
      document.querySelectorAll('.col-anth').forEach(el => el.classList.remove('column-hidden'));
      document.querySelectorAll('.col-remove').forEach(el => el.classList.remove('column-hidden'));
    }

    function updateChannelOptions() {
      const colorMode = document.querySelector('input[name="colorMode"]:checked')?.value || 'rgb';
      const channelRadioGroup = document.getElementById('channelRadioGroup');

      if (!channelRadioGroup) return;

      channelRadioGroup.innerHTML = '';

      let options = [];

      if (colorMode === 'rgb') {
        options = [
          { value: 'rrgb', text: "R'/RGB'" },
          { value: 'grgb', text: "G'/RGB'" },
          { value: 'brgb', text: "B'/RGB'" }
        ];
      } else {
        options = [
```

```
    { value: 'ccmyk', text: "C'/CMYK'" },
    { value: 'mcmyk', text: "M'/CMYK'" },
    { value: 'ycmyk', text: "Y'/CMYK'" },
    { value: 'kcmyk', text: "K'/CMYK'" }
  ];
}

options.forEach(option => {
  const label = document.createElement('label');
  label.style.cssText = 'font-weight: normal; display: inline-flex; align-items: center; cursor: pointer;';

  const radio = document.createElement('input');
  radio.type = 'radio';
  radio.name = 'anthChannel';
  radio.value = option.value;
  radio.id = `channelRadio_${option.value}`;
  radio.style.cssText = 'margin-right: 6px; cursor: pointer;';

  if (option.value === anthChannel) {
    radio.checked = true;
  }

  radio.addEventListener('change', function() {
    anthChannel = this.value;
    updateEquationStatus();
    updateEquationDescription();
    getRGB();
  });

  label.appendChild(radio);
```

```
    label.appendChild(document.createTextNode(option.text));
    channelRadioGroup.appendChild(label);
  });
}

function saveAnthEquation() {
  const newA = parseFloat(elements.anthEquationAInput.value);
  const newB = parseFloat(elements.anthEquationBInput.value);
  const selectedChannelRadio = document.querySelector('input[name="anthChannel"]:checked');
  const newChannel = selectedChannelRadio ? selectedChannelRadio.value : anthChannel;

  if (isNaN(newA) || isNaN(newB)) {
    alert('Please enter valid numbers for the equation parameters.');
    return;
  }

  anthEquationA = newA;
  anthEquationB = newB;
  anthChannel = newChannel;

  localStorage.setItem('anthEquationA', anthEquationA.toString());
  localStorage.setItem('anthEquationB', anthEquationB.toString());
  localStorage.setItem('anthChannel', anthChannel);

  updateEquationStatus();

  getRGB();
}

function resetAnthEquation() {
```

```
  anthEquationA = 0.0061;
  anthEquationB = 0.0026;
  anthChannel = 'rrgb';

  elements.anthEquationAInput.value = anthEquationA;
  elements.anthEquationBInput.value = anthEquationB;

  localStorage.removeItem('anthEquationA');
  localStorage.removeItem('anthEquationB');
  localStorage.removeItem('anthChannel');

  updateChannelOptions();
  updateEquationStatus();

  getRGB();
}

function updateEquationStatus() {
  const colorMode = document.querySelector('input[name="colorMode"]:checked')?.value || 'rgb';
  const mode = getGlobalMode();
  let channelName = '';

  if (colorMode === 'rgb') {
    switch(anthChannel) {
      case 'rrgb': channelName = "R'/RGB'"; break;
      case 'grgb': channelName = "G'/RGB'"; break;
      case 'brgb': channelName = "B'/RGB'"; break;
    }
  } else {
    switch(anthChannel) {
```

```
      case 'ccmyk': channelName = "C'/CMYK'"; break;
      case 'mcmyk': channelName = "M'/CMYK'"; break;
      case 'ycmyk': channelName = "Y'/CMYK'"; break;
      case 'kcmyk': channelName = "K'/CMYK'"; break;
    }
  }

  const suffix = mode === 'ab' ? ' (A-B)' : ' (A)';
  elements.equationStatus.textContent = `Color signal = ${anthEquationB}X + ${anthEquationA}`;
  elements.equationStatus.style.color = '#27ae60';

  updateEquationDescription();
}

function updateEquationDescription() {
  const mode = getGlobalMode();
  const colorMode = document.querySelector('input[name="colorMode"]:checked')?.value || 'rgb';
  const equationDesc = document.getElementById('equationDescription');
  const equationInputLabel = document.getElementById('equationInputLabel');

  if (equationDesc || equationInputLabel) {
    let channelName = '';

    if (colorMode === 'rgb') {
      switch(anthChannel) {
        case 'rrgb': channelName = "R'/RGB'"; break;
        case 'grgb': channelName = "G'/RGB'"; break;
        case 'brgb': channelName = "B'/RGB'"; break;
      }
    } else {
```

```
    switch(anthChannel) {
      case 'ccmyk': channelName = "C'/CMYK'"; break;
      case 'mcmyk': channelName = "M'/CMYK'"; break;
      case 'ycmyk': channelName = "Y'/CMYK'"; break;
      case 'kcmyk': channelName = "K'/CMYK'"; break;
    }
   }

   const suffix = mode === 'ab' ? ' (A-B)' : ' (A)';

   if (equationDesc) {
    equationDesc.textContent = `Color signal = MX+C`;
   }

   if (equationInputLabel) {
    equationInputLabel.textContent = `(${channelName}${suffix} -`;
   }
  }
}

let abSelectionState = { row: null, next: 'A' };

function downloadImageFromScanner() {
  let fileName = elements.filenameInput.value.trim();
  if (!fileName) {
   alert("Please enter a valid image filename.");
   return;
  }
  let deviceIP = "192.168.10.1";
  let imageUrl = `http://${deviceIP}/${fileName}`;
```

```
  let link = document.createElement("a");
  link.href = imageUrl;
  link.download = fileName;
  document.body.appendChild(link);
  link.click();
  document.body.removeChild(link);
  alert("Download complete. Now upload the image.");
}

elements.fileInput.addEventListener("change", function (event) {
  let file = event.target.files[0];
  if (!file) {
    console.log("No file selected");
    return;
  }

  console.log("File selected:", file.name, file.type, file.size);

  let reader = new FileReader();
  reader.onload = function (e) {
    console.log("FileReader loaded, creating image...");
    let img = new Image();
    img.onload = function () {
      console.log("Image loaded successfully:", img.width, "x", img.height);
      currentImage = img;
      resizeCanvas();
    };
    img.onerror = function (error) {
      console.error("Error loading image:", error);
      alert("Error loading image. Please try a different file format (JPG, PNG, etc.).");
```

```
    };
    img.src = e.target.result;
  };
  reader.onerror = function (error) {
    console.error("Error reading file:", error);
    alert("Error reading file. Please try again.");
  };
  reader.readAsDataURL(file);
});

function createPointRow(xA = "", yA = "", xB = "", yB = "") {
  let tr = document.createElement("tr");
  let tdNumber = document.createElement("td");
  tdNumber.className = "point-number col-point";
  tdNumber.textContent = "";
  tr.appendChild(tdNumber);
  let tdXA = document.createElement("td");
  tdXA.className = "col-xa";
  let inputXA = document.createElement("input");
  inputXA.type = "number";
  inputXA.className = "xaCoord";
  inputXA.value = xA;
  tdXA.appendChild(inputXA);
  tr.appendChild(tdXA);
  let tdYA = document.createElement("td");
  tdYA.className = "col-ya";
  let inputYA = document.createElement("input");
  inputYA.type = "number";
  inputYA.className = "yaCoord";
  inputYA.value = yA;
```

```
    tdYA.appendChild(inputYA);
    tr.appendChild(tdYA);
    let tdXB = document.createElement("td");
    tdXB.className = "col-xb";
    let inputXB = document.createElement("input");
    inputXB.type = "number";
    inputXB.className = "xbCoord";
    inputXB.value = xB;
    tdXB.appendChild(inputXB);
    tr.appendChild(tdXB);
    let tdYB = document.createElement("td");
    tdYB.className = "col-yb";
    let inputYB = document.createElement("input");
    inputYB.type = "number";
    inputYB.className = "ybCoord";
    inputYB.value = yB;
    tdYB.appendChild(inputYB);
    tr.appendChild(tdYB);
    let tdRadius = document.createElement("td");
    tdRadius.className = "col-radius";
    let inputRadius = document.createElement("input");
    inputRadius.type = "number";
    inputRadius.className = "radiusInput";
    inputRadius.min = "1";
    inputRadius.value = (elements.selectionRadiusInput && elements.selectionRadiusInput.value) ?
elements.selectionRadiusInput.value : 10;
    tdRadius.appendChild(inputRadius);
    tr.appendChild(tdRadius);
    let tdMode = document.createElement("td");
    tdMode.className = "col-mode";
```

```
let modeSpan = document.createElement("span");
modeSpan.className = "modeDisplay";
modeSpan.textContent = getGlobalModeLabel();
tdMode.appendChild(modeSpan);
tr.appendChild(tdMode);
let tdRGBa = document.createElement("td");
tdRGBa.className = "rgbACombined col-rgbACombined";
tr.appendChild(tdRGBa);
let tdRGBb = document.createElement("td");
tdRGBb.className = "rgbBCombined col-rgbBCombined";
tr.appendChild(tdRGBb);
let tdCMYKa = document.createElement("td");
tdCMYKa.className = "cmykACombined col-cmykACombined column-hidden";
tr.appendChild(tdCMYKa);
let tdCMYKb = document.createElement("td");
tdCMYKb.className = "cmykBCombined col-cmykBCombined column-hidden";
tr.appendChild(tdCMYKb);
let tdRA = document.createElement("td");
tdRA.className = "raResult raHeader col-ra";
tr.appendChild(tdRA);
let tdRASD = document.createElement("td");
tdRASD.className = "raSDResult raSDHeader col-raSD";
tr.appendChild(tdRASD);
let tdGA = document.createElement("td");
tdGA.className = "gaResult gaHeader col-ga";
tr.appendChild(tdGA);
let tdGASD = document.createElement("td");
tdGASD.className = "gaSDResult gaSDHeader col-gaSD";
tr.appendChild(tdGASD);
let tdBA = document.createElement("td");
```

```
    tdBA.className = "baResult baHeader col-ba";
    tr.appendChild(tdBA);
      let tdBASD = document.createElement("td");
tdBASD.className = "baSDResult baSDHeader col-baSD";
tr.appendChild(tdBASD);
let tdCA = document.createElement("td");
tdCA.className = "caResult caHeader col-ca column-hidden";
tr.appendChild(tdCA);
let tdCASD = document.createElement("td");
tdCASD.className = "caSDResult caSDHeader col-caSD column-hidden";
tr.appendChild(tdCASD);
let tdMA = document.createElement("td");
tdMA.className = "maResult maHeader col-ma column-hidden";
tr.appendChild(tdMA);
let tdMASD = document.createElement("td");
tdMASD.className = "maSDResult maSDHeader col-maSD column-hidden";
tr.appendChild(tdMASD);
let tdYA2 = document.createElement("td");
tdYA2.className = "ya2Result yaHeader col-ya2 column-hidden";
tr.appendChild(tdYA2);
let tdYA2SD = document.createElement("td");
tdYA2SD.className = "ya2SDResult yaSDHeader col-ya2SD column-hidden";
tr.appendChild(tdYA2SD);
let tdKA = document.createElement("td");
tdKA.className = "kaResult kaHeader col-ka column-hidden";
tr.appendChild(tdKA);
let tdKASD = document.createElement("td");
tdKASD.className = "kaSDResult kaSDHeader col-kaSD column-hidden";
tr.appendChild(tdKASD);
let tdRB = document.createElement("td");
```

```
    tdRB.className = "rbResult rbHeader col-rb";
    tr.appendChild(tdRB);
    let tdRBSD = document.createElement("td");
    tdRBSD.className = "rbSDResult rbSDHeader col-rbSD";
    tr.appendChild(tdRBSD);
    let tdGB = document.createElement("td");
    tdGB.className = "gbResult gbHeader col-gb";
    tr.appendChild(tdGB);
    let tdGBSD = document.createElement("td");
    tdGBSD.className = "gbSDResult gbSDHeader col-gbSD";
    tr.appendChild(tdGBSD);
    let tdBB = document.createElement("td");
    tdBB.className = "bbResult bbHeader col-bb";
    tr.appendChild(tdBB);
      let tdBBSD = document.createElement("td");
tdBBSD.className = "bbSDResult bbSDHeader col-bbSD";
tr.appendChild(tdBBSD);
let tdCB = document.createElement("td");
tdCB.className = "cbResult cbHeader col-cb column-hidden";
tr.appendChild(tdCB);
let tdCBSD = document.createElement("td");
tdCBSD.className = "cbSDResult cbSDHeader col-cbSD column-hidden";
tr.appendChild(tdCBSD);
let tdMB = document.createElement("td");
tdMB.className = "mbResult mbHeader col-mb column-hidden";
tr.appendChild(tdMB);
let tdMBSD = document.createElement("td");
tdMBSD.className = "mbSDResult mbSDHeader col-mbSD column-hidden";
tr.appendChild(tdMBSD);
let tdYB2 = document.createElement("td");
```

```
tdYB2.className = "yb2Result ybHeader col-yb2 column-hidden";
tr.appendChild(tdYB2);
let tdYB2SD = document.createElement("td");
tdYB2SD.className = "yb2SDResult ybSDHeader col-yb2SD column-hidden";
tr.appendChild(tdYB2SD);
let tdKB = document.createElement("td");
tdKB.className = "kbResult kbHeader col-kb column-hidden";
tr.appendChild(tdKB);
let tdKBSD = document.createElement("td");
tdKBSD.className = "kbSDResult kbSDHeader col-kbSD column-hidden";
tr.appendChild(tdKBSD);
let tdRAn = document.createElement("td");
tdRAn.className = "ranResult col-ran";
tr.appendChild(tdRAn);
let tdGAn = document.createElement("td");
tdGAn.className = "ganResult col-gan";
tr.appendChild(tdGAn);
let tdBAn = document.createElement("td");
tdBAn.className = "banResult col-ban";
tr.appendChild(tdBAn);
let tdCAn = document.createElement("td");
tdCAn.className = "canResult col-can column-hidden";
tr.appendChild(tdCAn);
let tdMAn = document.createElement("td");
tdMAn.className = "manResult col-man column-hidden";
tr.appendChild(tdMAn);
let tdYAn = document.createElement("td");
tdYAn.className = "yanResult col-yan column-hidden";
tr.appendChild(tdYAn);
let tdKAn = document.createElement("td");
```

```
tdKAn.className = "kanResult col-kan column-hidden";
tr.appendChild(tdKAn);
let tdRBn = document.createElement("td");
tdRBn.className = "rbnResult col-rbn";
tr.appendChild(tdRBn);
let tdGBn = document.createElement("td");
tdGBn.className = "gbnResult col-gbn";
tr.appendChild(tdGBn);
let tdBBn = document.createElement("td");
tdBBn.className = "bbnResult col-bbn";
tr.appendChild(tdBBn);
let tdCBn = document.createElement("td");
tdCBn.className = "cbnResult col-cbn column-hidden";
tr.appendChild(tdCBn);
let tdMBn = document.createElement("td");
tdMBn.className = "mbnResult col-mbn column-hidden";
tr.appendChild(tdMBn);
let tdYBn = document.createElement("td");
tdYBn.className = "ybnResult col-ybn column-hidden";
tr.appendChild(tdYBn);
let tdKBn = document.createElement("td");
tdKBn.className = "kbnResult col-kbn column-hidden";
tr.appendChild(tdKBn);
let tdRRGBa = document.createElement("td");
   tdRRGBa.className = "rrgbaResult col-rrgba";
   tr.appendChild(tdRRGBa);
   let tdRRGBb = document.createElement("td");
   tdRRGBb.className = "rrgbbResult col-rrgbb";
   tr.appendChild(tdRRGBb);
   let tdRRGB = document.createElement("td");
```

```
tdRRGB.className = "rrgbResult col-rrgb";
tr.appendChild(tdRRGB);
let tdCCMYKa = document.createElement("td");
tdCCMYKa.className = "ccmykaResult col-ccmyka column-hidden";
tr.appendChild(tdCCMYKa);
let tdCCMYKb = document.createElement("td");
tdCCMYKb.className = "ccmykbResult col-ccmykb column-hidden";
tr.appendChild(tdCCMYKb);
let tdCCMYK = document.createElement("td");
tdCCMYK.className = "ccmykResult col-ccmyk column-hidden";
tr.appendChild(tdCCMYK);
let tdMCMYKa = document.createElement("td");
tdMCMYKa.className = "mcmykaResult col-mcmyka column-hidden";
tr.appendChild(tdMCMYKa);
let tdMCMYKb = document.createElement("td");
tdMCMYKb.className = "mcmykbResult col-mcmykb column-hidden";
tr.appendChild(tdMCMYKb);
let tdMCMYK = document.createElement("td");
tdMCMYK.className = "mcmykResult col-mcmyk column-hidden";
tr.appendChild(tdMCMYK);
let tdYCMYKa = document.createElement("td");
tdYCMYKa.className = "ycmykaResult col-ycmyka column-hidden";
tr.appendChild(tdYCMYKa);
let tdYCMYKb = document.createElement("td");
tdYCMYKb.className = "ycmykbResult col-ycmykb column-hidden";
tr.appendChild(tdYCMYKb);
let tdYCMYK = document.createElement("td");
tdYCMYK.className = "ycmykResult col-ycmyk column-hidden";
tr.appendChild(tdYCMYK);
let tdKCMYKa = document.createElement("td");
```

```
tdKCMYKa.className = "kcmykaResult col-kcmyka column-hidden";
tr.appendChild(tdKCMYKa);
let tdKCMYKb = document.createElement("td");
tdKCMYKb.className = "kcmykbResult col-kcmykb column-hidden";
tr.appendChild(tdKCMYKb);
let tdKCMYK = document.createElement("td");
tdKCMYK.className = "kcmykResult col-kcmyk column-hidden";
tr.appendChild(tdKCMYK);
let tdGRGBa = document.createElement("td");
tdGRGBa.className = "grgbaResult col-grgba";
tr.appendChild(tdGRGBa);
let tdGRGBb = document.createElement("td");
tdGRGBb.className = "grgbbResult col-grgbb";
tr.appendChild(tdGRGBb);
let tdGRGB = document.createElement("td");
tdGRGB.className = "grgbResult col-grgb";
tr.appendChild(tdGRGB);
let tdBRGBa = document.createElement("td");
tdBRGBa.className = "brgbaResult col-brgba";
tr.appendChild(tdBRGBa);
let tdBRGBb = document.createElement("td");
tdBRGBb.className = "brgbbResult col-brgbb";
tr.appendChild(tdBRGBb);
let tdBRGB = document.createElement("td");
tdBRGB.className = "brgbResult col-brgb";
tr.appendChild(tdBRGB);
let tdAnth = document.createElement("td");
tdAnth.className = "anthResult col-anth";
tr.appendChild(tdAnth);
let tdRemove = document.createElement("td");
```

```
  tdRemove.className = "col-remove";
  let btnRemove = document.createElement("button");
  btnRemove.textContent = "Remove";
  btnRemove.onclick = function () {
    removePointRow(this);
  };
  tdRemove.appendChild(btnRemove);
  tr.appendChild(tdRemove);
  return tr;
}

function updatePointNumbers() {
  let rows = pointsBody.querySelectorAll("tr");
  rows.forEach((row, index) => {
    row.querySelector(".point-number").textContent = index + 1;
  });
}

function addPoint() {
  let newRow = createPointRow();
  pointsBody.appendChild(newRow);
  updatePointNumbers();
  attachInputListeners(newRow);
  updateColumnVisibility(); // Apply current visibility settings to new row
  abSelectionState = { row: newRow, next: 'A' };
}

function addPointFromClick(centerX, centerY) {
  let scaleX = currentImage.width / canvas.width;
  let scaleY = currentImage.height / canvas.height;
```

```
  let originalX = Math.round(centerX * scaleX);
  let originalY = Math.round(centerY * scaleY);

  if (!abSelectionState.row || abSelectionState.next === 'A') {
    let newRow = createPointRow(originalX, originalY);
    pointsBody.appendChild(newRow);
    updatePointNumbers();
    attachInputListeners(newRow);
    updateColumnVisibility(); // Apply current visibility settings to new row
    if (getGlobalMode() === 'ab') {
      abSelectionState = { row: newRow, next: 'B' };
      redrawCanvas();
    } else {
      abSelectionState = { row: null, next: 'A' };
      redrawCanvas();
      getRGB();
    }
  } else if (abSelectionState.next === 'B') {
    abSelectionState.row.querySelector('.xbCoord').value = originalX;
    abSelectionState.row.querySelector('.ybCoord').value = originalY;
    abSelectionState = { row: null, next: 'A' };
    redrawCanvas();
    getRGB();
  }
}

function removeLastPoint() {
  if (pointsBody.lastChild) {
    pointsBody.removeChild(pointsBody.lastChild);
    updatePointNumbers();
```

```
    redrawCanvas();
  }
}

function removePointRow(button) {
  let row = button.parentElement.parentElement;
  pointsBody.removeChild(row);
  updatePointNumbers();
  redrawCanvas();
}

function clearAllPoints() {
  pointsBody.innerHTML = "";
  redrawCanvas();
}

function redrawCanvas() {
  if (!currentImage) {
    console.log("No currentImage available for redrawCanvas");
    return;
  }

  if (!canvas || !ctx) {
    console.error("Canvas or context not initialized");
    return;
  }

  console.log("Redrawing canvas...");
  ctx.clearRect(0, 0, canvas.width, canvas.height);
  ctx.drawImage(currentImage, 0, 0, canvas.width, canvas.height);
```

```
    console.log("Image drawn to canvas");
    drawGrid();

    let rows = pointsBody.querySelectorAll("tr");
    rows.forEach((row, index) => {
      let xa = parseInt(row.querySelector(".xaCoord").value);
      let ya = parseInt(row.querySelector(".yaCoord").value);
      let xb = parseInt(row.querySelector(".xbCoord").value);
      let yb = parseInt(row.querySelector(".ybCoord").value);

      let scaleX = canvas.width / currentImage.width;
      let scaleY = canvas.height / currentImage.height;

    const perRowRadius = Math.max(1, parseInt(row.querySelector('.radiusInput')?.value ||
elements.selectionRadiusInput?.value || 10));
    const canvasRadius = perRowRadius * Math.min(scaleX, scaleY);
      const mode = getGlobalMode();
      if (!isNaN(xa) && !isNaN(ya)) {
        let canvasXa = Math.round(xa * scaleX);
        let canvasYa = Math.round(ya * scaleY);
        drawMarker(canvasXa, canvasYa, (index + 1) + 'A', 'blue', canvasRadius);
      }
      if (mode === 'ab' && !isNaN(xb) && !isNaN(yb)) {
        let canvasXb = Math.round(xb * scaleX);
        let canvasYb = Math.round(yb * scaleY);
        drawMarker(canvasXb, canvasYb, (index + 1) + 'B', 'green', canvasRadius);
      }
    });
  }
```

```
function drawGrid() {
  if (!canvas || !ctx) {
    console.error("Canvas or context not initialized for drawGrid");
    return;
  }

  let gridSize = window.innerWidth < 768 ? 50 : 100;
  ctx.strokeStyle = "rgba(0,0,0,0.5)";
  ctx.lineWidth = 1;
  ctx.font = window.innerWidth < 768 ? "12px Arial" : "18px Arial";
  ctx.textAlign = "center";
  ctx.textBaseline = "middle";

  for (let x = 0; x <= canvas.width; x += gridSize) {
    ctx.lineWidth = 1;
    ctx.strokeStyle = "red";
    ctx.beginPath();
    ctx.moveTo(x, 0);
    ctx.lineTo(x, canvas.height);
    ctx.stroke();

    let posY = window.innerWidth < 768 ? 8 : 10;
    ctx.lineWidth = 2;
    ctx.strokeStyle = "black";
    ctx.fillStyle = "white";
    ctx.strokeText(x, x, posY);
    ctx.fillText(x, x, posY);
  }

  for (let y = 0; y <= canvas.height; y += gridSize) {
```

```
    ctx.lineWidth = 1;
    ctx.strokeStyle = "red";
    ctx.beginPath();
    ctx.moveTo(0, y);
    ctx.lineTo(canvas.width, y);
    ctx.stroke();

    let posX = window.innerWidth < 768 ? 12 : 15;
    ctx.lineWidth = 2;
    ctx.strokeStyle = "black";
    ctx.fillStyle = "white";
    ctx.strokeText(y, posX, y);
    ctx.fillText(y, posX, y);
  }
}

function drawMarker(centerX, centerY, pointLabel, color = 'red', radius = 10) {
  if (!canvas || !ctx) {
    console.error("Canvas or context not initialized for drawMarker");
    return;
  }
  ctx.save();
  ctx.beginPath();
  ctx.arc(centerX, centerY, radius, 0, 2 * Math.PI);
  ctx.strokeStyle = color;
  ctx.lineWidth = window.innerWidth < 768 ? 1.5 : 2;
  ctx.stroke();
  ctx.restore();

  let offsetX = window.innerWidth < 768 ? 3 : 5;
```

```
  let offsetY = window.innerWidth < 768 ? 2 : 3;
  let textX = centerX - radius + offsetX;
  let textY = centerY - radius + offsetY;
  ctx.font = window.innerWidth < 768 ? "10px Arial" : "12px Arial";
  ctx.textAlign = "left";
  ctx.textBaseline = "top";
  ctx.lineWidth = window.innerWidth < 768 ? 2 : 3;
  ctx.strokeStyle = "black";
  ctx.fillStyle = "white";
  ctx.strokeText(pointLabel, textX, textY);
  ctx.fillText(pointLabel, textX, textY);
}

function getRGB() {
  try {
    if (!currentImage) {
      alert("Please upload an image first.");
      return;
    }

    const colorMode = document.querySelector('input[name="colorMode"]:checked')?.value || 'rgb';

    let offscreenCanvas = document.createElement("canvas");
    offscreenCanvas.width = canvas.width;
    offscreenCanvas.height = canvas.height;
    let offscreenCtx = offscreenCanvas.getContext("2d");
    offscreenCtx.drawImage(currentImage, 0, 0, canvas.width, canvas.height);

    let rows = pointsBody.querySelectorAll("tr");
    rows.forEach((row, index) => {
```

```
        let xa = parseInt(row.querySelector(".xaCoord").value);
        let ya = parseInt(row.querySelector(".yaCoord").value);
        let xb = parseInt(row.querySelector(".xbCoord").value);
        let yb = parseInt(row.querySelector(".ybCoord").value);
        const perRowRadius = Math.max(1, parseInt(row.querySelector('.radiusInput')?.value ||
elements.selectionRadiusInput?.value || 10));
        const mode = getGlobalMode();
        if (isNaN(xa) || isNaN(ya)) return;

        let scaleX = canvas.width / currentImage.width;
        let scaleY = canvas.height / currentImage.height;
        let canvasXa = Math.round(xa * scaleX);
        let canvasYa = Math.round(ya * scaleY);
        let canvasXb = !isNaN(xb) ? Math.round(xb * scaleX) : NaN;
        let canvasYb = !isNaN(yb) ? Math.round(yb * scaleY) : NaN;
        const canvasRadiusA = perRowRadius * Math.min(scaleX, scaleY);
        let sumRA = 0, sumGA = 0, sumBA = 0, totalA = 0;
        let arrRA = [], arrGA = [], arrBA = [];
        const radiusSquaredA = canvasRadiusA * canvasRadiusA;
        let minXA = Math.max(0, Math.floor(canvasXa - canvasRadiusA));
        let maxXA = Math.min(canvas.width - 1, Math.ceil(canvasXa + canvasRadiusA));
        let minYA = Math.max(0, Math.floor(canvasYa - canvasRadiusA));
        let maxYA = Math.min(canvas.height - 1, Math.ceil(canvasYa + canvasRadiusA));
        for (let px = minXA; px <= maxXA; px++) {
          for (let py = minYA; py <= maxYA; py++) {
            const dx = px - canvasXa;
            const dy = py - canvasYa;
            const distSquared = dx * dx + dy * dy;
            if (distSquared <= radiusSquaredA) {
              let pixelData = offscreenCtx.getImageData(px, py, 1, 1).data;
```

```
      sumRA += pixelData[0];
      sumGA += pixelData[1];
      sumBA += pixelData[2];
      arrRA.push(pixelData[0]);
      arrGA.push(pixelData[1]);
      arrBA.push(pixelData[2]);
      totalA++;
    }
  }
}
if (totalA === 0) return;
let avgRA = sumRA / totalA;
let avgGA = sumGA / totalA;
let avgBA = sumBA / totalA;
let sdRA = Math.sqrt(arrRA.reduce((acc, v) => acc + Math.pow(v - avgRA, 2), 0) / totalA);
let sdGA = Math.sqrt(arrGA.reduce((acc, v) => acc + Math.pow(v - avgGA, 2), 0) / totalA);
let sdBA = Math.sqrt(arrBA.reduce((acc, v) => acc + Math.pow(v - avgBA, 2), 0) / totalA);
let sumRB = 0, sumGB = 0, sumBB = 0, totalB = 0;
let arrRB = [], arrGB = [], arrBB = [];
let avgRB = 0, avgGB = 0, avgBB = 0, sdRB = 0, sdGB = 0, sdBB = 0;
if (mode === 'ab' && !isNaN(canvasXb) && !isNaN(canvasYb)) {
  const canvasRadiusB = perRowRadius * Math.min(scaleX, scaleY);
  const radiusSquaredB = canvasRadiusB * canvasRadiusB;
  let minXB = Math.max(0, Math.floor(canvasXb - canvasRadiusB));
  let maxXB = Math.min(canvas.width - 1, Math.ceil(canvasXb + canvasRadiusB));
  let minYB = Math.max(0, Math.floor(canvasYb - canvasRadiusB));
  let maxYB = Math.min(canvas.height - 1, Math.ceil(canvasYb + canvasRadiusB));
  for (let px = minXB; px <= maxXB; px++) {
    for (let py = minYB; py <= maxYB; py++) {
      const dx = px - canvasXb;
```

```
      const dy = py - canvasYb;
      const distSquared = dx * dx + dy * dy;
      if (distSquared <= radiusSquaredB) {
        let pixelData = offscreenCtx.getImageData(px, py, 1, 1).data;
        sumRB += pixelData[0];
        sumGB += pixelData[1];
        sumBB += pixelData[2];
        arrRB.push(pixelData[0]);
        arrGB.push(pixelData[1]);
        arrBB.push(pixelData[2]);
        totalB++;
      }
    }
  }
  if (totalB > 0) {
    avgRB = sumRB / totalB;
    avgGB = sumGB / totalB;
    avgBB = sumBB / totalB;
    sdRB = Math.sqrt(arrRB.reduce((acc, v) => acc + Math.pow(v - avgRB, 2), 0) / totalB);
    sdGB = Math.sqrt(arrGB.reduce((acc, v) => acc + Math.pow(v - avgGB, 2), 0) / totalB);
    sdBB = Math.sqrt(arrBB.reduce((acc, v) => acc + Math.pow(v - avgBB, 2), 0) / totalB);
  }
}
let normRA = avgRA / 255;
let normGA = avgGA / 255;
let normBA = avgBA / 255;
let normRB = totalB > 0 ? (avgRB / 255) : 0;
let normGB = totalB > 0 ? (avgGB / 255) : 0;
let normBB = totalB > 0 ? (avgBB / 255) : 0;
let sumNormA = normRA + normGA + normBA;
```

```
        let sumNormB = normRB + normGB + normBB;
        let rrgbA = sumNormA !== 0 ? normRA / sumNormA : 0;
        let rrgbB = (mode === 'ab' && sumNormB !== 0) ? (normRB / sumNormB) : 0;
        let grgbA = sumNormA !== 0 ? normGA / sumNormA : 0;
        let grgbB = (mode === 'ab' && sumNormB !== 0) ? (normGB / sumNormB) : 0;
        let brgbA = sumNormA !== 0 ? normBA / sumNormA : 0;
        let brgbB = (mode === 'ab' && sumNormB !== 0) ? (normBB / sumNormB) : 0;
        let rrgbDiff = (mode === 'ab') ? (rrgbA - rrgbB) : rrgbA;
        let grgbDiff = (mode === 'ab') ? (grgbA - grgbB) : grgbA;
        let brgbDiff = (mode === 'ab') ? (brgbA - brgbB) : brgbA;
        row.querySelector(".raResult").textContent = Math.round(avgRA);
        row.querySelector(".raSDResult").textContent = sdRA.toFixed(2);
        row.querySelector(".gaResult").textContent = Math.round(avgGA);
        row.querySelector(".gaSDResult").textContent = sdGA.toFixed(2);
        row.querySelector(".baResult").textContent = Math.round(avgBA);
        row.querySelector(".baSDResult").textContent = sdBA.toFixed(2);
        row.querySelector(".rbResult").textContent = (mode === 'ab' && totalB > 0) ? Math.round(avgRB) : '';
        row.querySelector(".rbSDResult").textContent = (mode === 'ab' && totalB > 0) ? sdRB.toFixed(2) : '';
        row.querySelector(".gbResult").textContent = (mode === 'ab' && totalB > 0) ? Math.round(avgGB) : '';
        row.querySelector(".gbSDResult").textContent = (mode === 'ab' && totalB > 0) ? sdGB.toFixed(2) : '';
        row.querySelector(".bbResult").textContent = (mode === 'ab' && totalB > 0) ? Math.round(avgBB) : '';
        row.querySelector(".bbSDResult").textContent = (mode === 'ab' && totalB > 0) ? sdBB.toFixed(2) : '';
        row.querySelector(".rgbACombined").textContent = `${Math.round(avgRA)}, ${Math.round(avgGA)},
${Math.round(avgBA)}`;
        row.querySelector(".rgbBCombined").textContent = (mode === 'ab' && totalB > 0) ? `${Math.round(avgRB)},
${Math.round(avgGB)}, ${Math.round(avgBB)}` : '';
        const [cA, mA, yA, kA] = rgbToCmyk(avgRA, avgGA, avgBA);
        const [cB, mB, yB, kB] = totalB > 0 ? rgbToCmyk(avgRB, avgGB, avgBB) : [0,0,0,0];
        row.querySelector(".cmykACombined").textContent = `${cA}, ${mA}, ${yA}, ${kA}`;
        row.querySelector(".cmykBCombined").textContent = (mode === 'ab' && totalB > 0) ? `${cB}, ${mB}, ${yB},
${kB}` : '';
```

```
row.querySelector(".caResult").textContent = cA;
row.querySelector(".maResult").textContent = mA;
row.querySelector(".ya2Result").textContent = yA;
row.querySelector(".kaResult").textContent = kA;
row.querySelector(".cbResult").textContent = (mode === 'ab' && totalB > 0) ? cB : '';
row.querySelector(".mbResult").textContent = (mode === 'ab' && totalB > 0) ? mB : '';
row.querySelector(".yb2Result").textContent = (mode === 'ab' && totalB > 0) ? yB : '';
row.querySelector(".kbResult").textContent = (mode === 'ab' && totalB > 0) ? kB : '';

const normCA = cA / 100;
const normMA = mA / 100;
const normYA = yA / 100;
const normKA = kA / 100;
const normCB = cB / 100;
const normMB = mB / 100;
const normYB = yB / 100;
const normKB = kB / 100;

row.querySelector(".canResult").textContent = normCA.toFixed(4);
row.querySelector(".manResult").textContent = normMA.toFixed(4);
row.querySelector(".yanResult").textContent = normYA.toFixed(4);
row.querySelector(".kanResult").textContent = normKA.toFixed(4);
row.querySelector(".cbnResult").textContent = (mode === 'ab' && totalB > 0) ? normCB.toFixed(4) : '';
row.querySelector(".mbnResult").textContent = (mode === 'ab' && totalB > 0) ? normMB.toFixed(4) : '';
row.querySelector(".ybnResult").textContent = (mode === 'ab' && totalB > 0) ? normYB.toFixed(4) : '';
row.querySelector(".kbnResult").textContent = (mode === 'ab' && totalB > 0) ? normKB.toFixed(4) : '';
const sumNormCmykA = normCA + normMA + normYA + normKA;
const sumNormCmykB = normCB + normMB + normYB + normKB;
const ccmykA = sumNormCmykA !== 0 ? normCA / sumNormCmykA : 0;
```

```
const ccmykB = (mode === 'ab' && sumNormCmykB !== 0) ? normCB / sumNormCmykB : 0;
const ccmykDiff = (mode === 'ab') ? (ccmykA - ccmykB) : ccmykA;
const mcmykA = sumNormCmykA !== 0 ? normMA / sumNormCmykA : 0;
const mcmykB = (mode === 'ab' && sumNormCmykB !== 0) ? normMB / sumNormCmykB : 0;
const mcmykDiff = (mode === 'ab') ? (mcmykA - mcmykB) : mcmykA;
const ycmykA = sumNormCmykA !== 0 ? normYA / sumNormCmykA : 0;
const ycmykB = (mode === 'ab' && sumNormCmykB !== 0) ? normYB / sumNormCmykB : 0;
const ycmykDiff = (mode === 'ab') ? (ycmykA - ycmykB) : ycmykA;
const kcmykA = sumNormCmykA !== 0 ? normKA / sumNormCmykA : 0;
const kcmykB = (mode === 'ab' && sumNormCmykB !== 0) ? normKB / sumNormCmykB : 0;
const kcmykDiff = (mode === 'ab') ? (kcmykA - kcmykB) : kcmykA;
row.querySelector(".ccmykaResult").textContent = ccmykA.toFixed(4);
row.querySelector(".ccmykbResult").textContent = (mode === 'ab' && totalB > 0) ? ccmykB.toFixed(4) : '';
row.querySelector(".ccmykResult").textContent = ccmykDiff.toFixed(4);
row.querySelector(".mcmykaResult").textContent = mcmykA.toFixed(4);
row.querySelector(".mcmykbResult").textContent = (mode === 'ab' && totalB > 0) ? mcmykB.toFixed(4) : '';
row.querySelector(".mcmykResult").textContent = mcmykDiff.toFixed(4);
row.querySelector(".ycmykaResult").textContent = ycmykA.toFixed(4);
row.querySelector(".ycmykbResult").textContent = (mode === 'ab' && totalB > 0) ? ycmykB.toFixed(4) : '';
row.querySelector(".ycmykResult").textContent = ycmykDiff.toFixed(4);
row.querySelector(".kcmykaResult").textContent = kcmykA.toFixed(4);
row.querySelector(".kcmykbResult").textContent = (mode === 'ab' && totalB > 0) ? kcmykB.toFixed(4) : '';
row.querySelector(".kcmykResult").textContent = kcmykDiff.toFixed(4);
row.querySelector(".ranResult").textContent = normRA.toFixed(4);
row.querySelector(".ganResult").textContent = normGA.toFixed(4);
row.querySelector(".banResult").textContent = normBA.toFixed(4);
row.querySelector(".rbnResult").textContent = (mode === 'ab' && totalB > 0) ? normRB.toFixed(4) : '';
row.querySelector(".gbnResult").textContent = (mode === 'ab' && totalB > 0) ? normGB.toFixed(4) : '';
row.querySelector(".bbnResult").textContent = (mode === 'ab' && totalB > 0) ? normBB.toFixed(4) : '';
row.querySelector(".rrgbaResult").textContent = rrgbA.toFixed(4);
```

```
row.querySelector(".rrgbbResult").textContent = (mode === 'ab' && totalB > 0) ? rrgbB.toFixed(4) : '';
row.querySelector(".rrgbResult").textContent = rrgbDiff.toFixed(4);
row.querySelector(".grgbaResult").textContent = grgbA.toFixed(4);
row.querySelector(".grgbbResult").textContent = (mode === 'ab' && totalB > 0) ? grgbB.toFixed(4) : '';
row.querySelector(".grgbResult").textContent = grgbDiff.toFixed(4);
row.querySelector(".brgbaResult").textContent = brgbA.toFixed(4);
row.querySelector(".brgbbResult").textContent = (mode === 'ab' && totalB > 0) ? brgbB.toFixed(4) : '';
row.querySelector(".brgbResult").textContent = brgbDiff.toFixed(4);
row.title = (mode === 'ab' && totalB > 0)
  ? `R'/RGB' (A): ${rrgbA.toFixed(4)}, R'/RGB' (B): ${rrgbB.toFixed(4)}, Diff: ${rrgbDiff.toFixed(4)}`
  : `R'/RGB' (A): ${rrgbA.toFixed(4)}`;
let channelValue = 0;

if (colorMode === 'rgb') {
  switch(anthChannel) {
    case 'rrgb': channelValue = mode === 'ab' ? rrgbDiff : rrgbA; break;
    case 'grgb': channelValue = mode === 'ab' ? grgbDiff : grgbA; break;
    case 'brgb': channelValue = mode === 'ab' ? brgbDiff : brgbA; break;
  }
} else {
  switch(anthChannel) {
    case 'ccmyk': channelValue = mode === 'ab' ? ccmykDiff : ccmykA; break;
    case 'mcmyk': channelValue = mode === 'ab' ? mcmykDiff : mcmykA; break;
    case 'ycmyk': channelValue = mode === 'ab' ? ycmykDiff : ycmykA; break;
    case 'kcmyk': channelValue = mode === 'ab' ? kcmykDiff : kcmykA; break;
  }
}

let anth = (channelValue - anthEquationA) / anthEquationB;
row.querySelector(".anthResult").textContent = anth > 0 ? anth.toFixed(2) : '0.00';
```

```
    });
  } catch (error) {
    alert("Error processing image: " + error.message);
  }
}

function downloadResultImage() {
  alert("Download Result Image functionality not implemented yet.");
}

function handleCanvasInteraction(e) {
  e.preventDefault();
  if (!currentImage) {
    alert('Please upload an image before selecting points.');
    return;
  }
  if (!canvas) {
    console.error("Canvas not initialized for interaction");
    return;
  }
  let rect = canvas.getBoundingClientRect();
  let scaleX = canvas.width / rect.width;
  let scaleY = canvas.height / rect.height;
  let clientX, clientY;
  if (e.type === 'touchstart' || e.type === 'touchend') {
    clientX = e.touches[0] ? e.touches[0].clientX : e.changedTouches[0].clientX;
    clientY = e.touches[0] ? e.touches[0].clientY : e.changedTouches[0].clientY;
  } else {
    clientX = e.clientX;
    clientY = e.clientY;
```

```
  }
  let centerX = (clientX - rect.left) * scaleX;
  let centerY = (clientY - rect.top) * scaleY;
  try {
    addPointFromClick(centerX, centerY);
  } catch (err) {
    alert('Error adding point: ' + err.message);
  }
}


function attachInputListeners(row) {
  let xaInput = row.querySelector('.xaCoord');
  let yaInput = row.querySelector('.yaCoord');
  let xbInput = row.querySelector('.xbCoord');
  let ybInput = row.querySelector('.ybCoord');
  let radiusInput = row.querySelector('.radiusInput');
  [xaInput, yaInput, xbInput, ybInput, radiusInput].forEach(input => {
    input.addEventListener('input', function() {
      redrawCanvas();
      getRGB();
    });
  });
}

document.addEventListener('DOMContentLoaded', function() {
  console.log("DOM Content Loaded");

  canvas = elements.canvas;
  if (canvas) {
```

```
  ctx = canvas.getContext("2d", { willReadFrequently: true });
  console.log("Canvas element:", canvas);
  console.log("Canvas context:", ctx);

  canvas.width = 400;
  canvas.height = 300;
  console.log("Canvas initialized with size:", canvas.width, "x", canvas.height);
} else {
  console.error("Canvas element not found!");
}

console.log("File input element:", elements.fileInput);

if (canvas) {
  canvas.addEventListener("click", handleCanvasInteraction);
  canvas.addEventListener("touchend", handleCanvasInteraction);
  console.log("Canvas event listeners added");
}

let rows = pointsBody.querySelectorAll('tr');
rows.forEach(row => attachInputListeners(row));

let selectionRadiusInput = elements.selectionRadiusInput;
if (selectionRadiusInput) {
  selectionRadiusInput.addEventListener('input', function() {
    redrawCanvas();
    getRGB();
  });
}
```

```
loadAnthEquation();

loadColumnVisibilitySettings();

updateChannelOptions();

elements.showRGBColumns.addEventListener('change', function() {
  updateColumnVisibility();
  saveColumnVisibilitySettings();
});

elements.showRGBRatioColumns.addEventListener('change', function() {
  updateColumnVisibility();
  saveColumnVisibilitySettings();
});

elements.showRawRGBColumns.addEventListener('change', function() {
  updateColumnVisibility();
  saveColumnVisibilitySettings();
});

elements.showCoordColumns.addEventListener('change', function() {
  updateColumnVisibility();
  saveColumnVisibilitySettings();
});

const savedColorMode = localStorage.getItem('colorMode') || 'rgb';
const colorModeRGB = document.getElementById('colorModeRGB');
const colorModeCMYK = document.getElementById('colorModeCMYK');
```

```
if (savedColorMode === 'cmyk') {
  colorModeCMYK.checked = true;
} else {
  colorModeRGB.checked = true;
}

function handleColorModeChange() {
  const selectedColorMode = document.querySelector('input[name="colorMode"]:checked')?.value || 'rgb';
  localStorage.setItem('colorMode', selectedColorMode);
  updateChannelOptions();
  updateColumnVisibility();
  updateEquationDescription();
  getRGB();
}

colorModeRGB.addEventListener('change', handleColorModeChange);
colorModeCMYK.addEventListener('change', handleColorModeChange);

const savedGlobalMode = localStorage.getItem('globalMode') || 'ab';
const globalModeAB = document.getElementById('globalModeAB');
const globalModeAOnly = document.getElementById('globalModeAOnly');

if (savedGlobalMode === 'a-only') {
  globalModeAOnly.checked = true;
} else {
  globalModeAB.checked = true;
}

function handleGlobalModeChange() {
  const selectedGlobalMode = document.querySelector('input[name="globalMode"]:checked')?.value || 'ab';
```

```
  localStorage.setItem('globalMode', selectedGlobalMode);
  document.querySelectorAll('.modeDisplay').forEach(el => {
    el.textContent = getGlobalModeLabel();
  });
  updateEquationStatus();
  updateEquationDescription();
  redrawCanvas();
  getRGB();
}

globalModeAB.addEventListener('change', handleGlobalModeChange);
globalModeAOnly.addEventListener('change', handleGlobalModeChange);

let lastTouchEnd = 0;
document.addEventListener('touchend', function (event) {
  let now = (new Date()).getTime();
  if (now - lastTouchEnd <= 300) {
    event.preventDefault();
  }
  lastTouchEnd = now;
}, false);

let inputs = document.querySelectorAll('input[type="text"], input[type="number"]');
inputs.forEach(input => {
  input.addEventListener('focus', function() {
    setTimeout(() => {
      input.scrollIntoView({ behavior: 'smooth', block: 'center' });
    }, 100);
  });
});
```

```
    canvas.style.touchAction = 'manipulation';

    window.addEventListener('resize', function() {
      if (currentImage) {
        resizeCanvas();
      }
    });
  });

  var script = document.createElement('script');
  script.src = 'https://cdn.jsdelivr.net/npm/xlsx@0.18.5/dist/xlsx.full.min.js';
  document.head.appendChild(script);

  function exportToXLSX() {
    let rows = pointsBody.querySelectorAll('tr');
    let data = [];
    let colorMode = document.getElementById('colorModeSelect').value;

    if (colorMode === 'rgb') {
      data.push([
        'Point', 'XA', 'YA', 'XB', 'YB', 'Radius', 'Mode', 'R_A', 'SD(R_A)', 'G_A', 'SD(G_A)', 'B_A', 'SD(B_A)', 'R_B', 'SD(R_B)', 'G_B', 'SD(G_B)', 'B_B', 'SD(B_B)', "R'_A", "G'_A", "B'_A", "R'_B", "G'_B", "B'_B", "R'/RGB' (A)", "R'/RGB' (B)", "R'/RGB' (A-B)", "G'/RGB' (A)", "G'/RGB' (B)", "G'/RGB' (A-B)", "B'/RGB' (A)", "B'/RGB' (B)", "B'/RGB' (A-B)", 'Concentration'
      ]);
    } else {
      data.push([
        'Point', 'XA', 'YA', 'XB', 'YB', 'Radius', 'Mode', 'C_A', 'SD(C_A)', 'M_A', 'SD(M_A)', 'Y_A', 'SD(Y_A)', 'K_A', 'SD(K_A)', 'C_B', 'SD(C_B)', 'M_B', 'SD(M_B)', 'Y_B', 'SD(Y_B)', 'K_B', 'SD(K_B)', "C'_A", "M'_A", "Y'_A", "K'_A", "C'_B", "M'_B", "Y'_B", "K'_B", "C'/CMYK' (A)", "C'/CMYK' (B)", "C'/CMYK' (A-B)", "M'/CMYK' (A)", "M'/CMYK' (B)", "M'/CMYK' (A-B)", "Y'/CMYK' (A)", "Y'/CMYK' (B)", "Y'/CMYK' (A-B)", "K'/CMYK' (A)", "K'/CMYK' (B)", "K'/CMYK' (A-B)", 'Concentration'
```

```
  ]);
}

rows.forEach(row => {
  if (colorMode === 'rgb') {
    data.push([
      row.querySelector('.point-number')?.textContent || '',
      row.querySelector('.xaCoord')?.value || '',
      row.querySelector('.yaCoord')?.value || '',
      row.querySelector('.xbCoord')?.value || '',
      row.querySelector('.ybCoord')?.value || '',
      row.querySelector('.radiusInput')?.value || '',
      getGlobalModeLabel(),
      row.querySelector('.raResult')?.textContent || '',
      row.querySelector('.raSDResult')?.textContent || '',
      row.querySelector('.gaResult')?.textContent || '',
      row.querySelector('.gaSDResult')?.textContent || '',
      row.querySelector('.baResult')?.textContent || '',
      row.querySelector('.baSDResult')?.textContent || '',
      row.querySelector('.rbResult')?.textContent || '',
      row.querySelector('.rbSDResult')?.textContent || '',
      row.querySelector('.gbResult')?.textContent || '',
      row.querySelector('.gbSDResult')?.textContent || '',
      row.querySelector('.bbResult')?.textContent || '',
      row.querySelector('.bbSDResult')?.textContent || '',
      row.querySelector('.ranResult')?.textContent || '',
      row.querySelector('.ganResult')?.textContent || '',
      row.querySelector('.banResult')?.textContent || '',
      row.querySelector('.rbnResult')?.textContent || '',
      row.querySelector('.gbnResult')?.textContent || '',
```

```
    row.querySelector('.bbnResult')?.textContent || '',
    row.querySelector('.rrgbaResult')?.textContent || '',
    row.querySelector('.rrgbbResult')?.textContent || '',
    row.querySelector('.rrgbResult')?.textContent || '',
    row.querySelector('.grgbaResult')?.textContent || '',
    row.querySelector('.grgbbResult')?.textContent || '',
    row.querySelector('.grgbResult')?.textContent || '',
    row.querySelector('.brgbaResult')?.textContent || '',
    row.querySelector('.brgbbResult')?.textContent || '',
    row.querySelector('.brgbResult')?.textContent || '',
    row.querySelector('.anthResult')?.textContent || ''
  ]);
 } else {
  data.push([
    row.querySelector('.point-number')?.textContent || '',
    row.querySelector('.xaCoord')?.value || '',
    row.querySelector('.yaCoord')?.value || '',
    row.querySelector('.xbCoord')?.value || '',
    row.querySelector('.ybCoord')?.value || '',
    row.querySelector('.radiusInput')?.value || '',
    getGlobalModeLabel(),
    row.querySelector('.caResult')?.textContent || '',
    row.querySelector('.caSDResult')?.textContent || '',
    row.querySelector('.maResult')?.textContent || '',
    row.querySelector('.maSDResult')?.textContent || '',
    row.querySelector('.ya2Result')?.textContent || '',
    row.querySelector('.ya2SDResult')?.textContent || '',
    row.querySelector('.kaResult')?.textContent || '',
    row.querySelector('.kaSDResult')?.textContent || '',
    row.querySelector('.cbResult')?.textContent || '',
```

```
    row.querySelector('.cbSDResult')?.textContent || '',
    row.querySelector('.mbResult')?.textContent || '',
    row.querySelector('.mbSDResult')?.textContent || '',
    row.querySelector('.yb2Result')?.textContent || '',
    row.querySelector('.yb2SDResult')?.textContent || '',
    row.querySelector('.kbResult')?.textContent || '',
    row.querySelector('.kbSDResult')?.textContent || '',
    row.querySelector('.canResult')?.textContent || '',
    row.querySelector('.manResult')?.textContent || '',
    row.querySelector('.yanResult')?.textContent || '',
    row.querySelector('.kanResult')?.textContent || '',
    row.querySelector('.cbnResult')?.textContent || '',
    row.querySelector('.mbnResult')?.textContent || '',
    row.querySelector('.ybnResult')?.textContent || '',
    row.querySelector('.kbnResult')?.textContent || '',
    row.querySelector('.ccmykaResult')?.textContent || '',
    row.querySelector('.ccmykbResult')?.textContent || '',
    row.querySelector('.ccmykResult')?.textContent || '',
    row.querySelector('.mcmykaResult')?.textContent || '',
    row.querySelector('.mcmykbResult')?.textContent || '',
    row.querySelector('.mcmykResult')?.textContent || '',
    row.querySelector('.ycmykaResult')?.textContent || '',
    row.querySelector('.ycmykbResult')?.textContent || '',
    row.querySelector('.ycmykResult')?.textContent || '',
    row.querySelector('.kcmykaResult')?.textContent || '',
    row.querySelector('.kcmykbResult')?.textContent || '',
    row.querySelector('.kcmykResult')?.textContent || '',
    row.querySelector('.anthResult')?.textContent || ''
  ]);
}
```

```
  });

  let ws = XLSX.utils.aoa_to_sheet(data);
  let wb = XLSX.utils.book_new();
  XLSX.utils.book_append_sheet(wb, ws, 'Results');
  XLSX.writeFile(wb, 'results.xlsx');
}

function checkFileOnScanner() {
  let fileName = elements.filenameInput.value.trim();
  let resultSpan = elements.fileCheckResult;
  if (!fileName) {
    resultSpan.textContent = "Please enter a filename.";
    resultSpan.className = "file-status error";
    return;
  }
  let deviceIP = "192.168.10.1";
  let url = `http://${deviceIP}/`;

  resultSpan.textContent = "Checking...";
  resultSpan.className = "file-status info";

  fetch(url)
    .then(response => {
      if (!response.ok) throw new Error("Cannot access scanner.");
      return response.text();
    })
    .then(html => {
      let found = false;
      let date = "";
```

```
      let regex = new RegExp(`<a[^>]*>${fileName}</a>.*?(\\d{4}-\\d{2}-\\d{2} \\d{2}:\\d{2})`, "i");
      let match = html.match(regex);
      if (match) {
        found = true;
        date = match[1];
      } else if (html.includes(fileName)) {
        found = true;
        date = "(date not found)";
      }
      if (found) {
        resultSpan.textContent = `File found. Date: ${date}`;
        resultSpan.className = "file-status success";
      } else {
        resultSpan.textContent = "File not found on scanner.";
        resultSpan.className = "file-status error";
      }
    })
    .catch(err => {
      resultSpan.textContent = "Error accessing scanner or file list.";
      resultSpan.className = "file-status error";
    });
}

function resizeCanvas() {
  if (!currentImage) {
    console.log("No currentImage available for resizeCanvas");
    return;
  }

  if (!canvas || !ctx) {
```

```
    console.error("Canvas or context not initialized for resizeCanvas");
    return;
  }

  console.log("Resizing canvas for image:", currentImage.width, "x", currentImage.height);

  let container = document.getElementById('canvasContainer');
  if (!container) {
    console.error("Canvas container not found");
    return;
  }

  let containerRect = container.getBoundingClientRect();
  let maxWidth = containerRect.width - 20; // Account for padding/borders
  let maxHeight = window.innerHeight * 0.6;

  console.log("Container dimensions:", containerRect.width, "x", containerRect.height);
  console.log("Max dimensions:", maxWidth, "x", maxHeight);

  let scale = Math.min(maxWidth / currentImage.width, maxHeight / currentImage.height);
  canvas.width = currentImage.width * scale;
  canvas.height = currentImage.height * scale;

  console.log("Canvas resized to:", canvas.width, "x", canvas.height, "scale:", scale);

  redrawCanvas();
}

function rgbToCmyk(r, g, b) {
  r = r / 255; g = g / 255; b = b / 255;
```

```
  let k = 1 - Math.max(r, g, b);
  let c = k === 1 ? 0 : (1 - r - k) / (1 - k);
  let m = k === 1 ? 0 : (1 - g - k) / (1 - k);
  let y = k === 1 ? 0 : (1 - b - k) / (1 - k);
  return [c, m, y, k].map(x => (x * 100).toFixed(1));
}

const anthAInputs = document.querySelectorAll('input#anthEquationA');
const anthBInputs = document.querySelectorAll('input#anthEquationB');

function syncAnthEquationInputs() {
  anthAInputs.forEach(input => input.value = elements.anthEquationAInput.value);
  anthBInputs.forEach(input => input.value = elements.anthEquationBInput.value);
  const channelRadio = document.querySelector(`input[name="anthChannel"][value="${anthChannel}"]`);
  if (channelRadio) channelRadio.checked = true;
}

anthAInputs.forEach(input => {
  input.addEventListener('input', function() {
    elements.anthEquationAInput.value = input.value;
    syncAnthEquationInputs();
    saveAnthEquation();
  });
});

anthBInputs.forEach(input => {
  input.addEventListener('input', function() {
    elements.anthEquationBInput.value = input.value;
    syncAnthEquationInputs();
    saveAnthEquation();
```

```
    });
  });
 </script>
</body>
</html>
```