Histogrammanpassung ist eine Technik, die ein Bild an eine vorgegebene Verteilungsform oder ein bestehendes Histogramm anzugleichen ermöglicht.

Dies ist beispielsweise besonders hilfreich bei der Vorbereitung einer Reihe von Bildern, die bei unterschiedlichen Aufnahmeverhältnissen oder mit verschiedenen Kameras entstanden sind, um bei der Reproduktion oder Druckproduktion ähnlich auszusehen sollen.

Der Vorgang der Histogrammanpassung basiert auf der Abstimmung der kumulativen Histogramme durch eine homogene Punktoperation. Um von der Bildgröße (Anzahl der Pixel) unabhängig zu sein, wird normalisierte Verteilungen definiert, um anstelle der ursprünglichen Histogramme zu verwenden.

### Die Histogrammanpassung als Plug-In für ImageJ zu implementieren

### (Java implementation)

*HistogramMatchingPlugin_.java | *PlotHistogram.java | *Histogrammanpassung.java ×

```java
public class Histogrammanpassung {

    // compute mapping function f():
    public int[] matchHistograms (int[] hA, int[] hR) {
        int K = hA.length;
        // get (CDF) of histograms target, reference imag
        double[] PA = Cdf(hA);
        double[] PR = Cdf(hR);
        //function f()
        int[] fhs = new int[K];
        for (int a = 0; a < K; a++) {
            int j = K - 1;
            do {
                fhs[a] = j;
                j--;
            } while (j >= 0 && PA[a] <= PR[j]);
        }
        return fhs;
    }

    // returns (cdf) of histogram
    public static double[] Cdf (int[] h) {

        int K = h.length;
        int n = 0;
        for (int i=0; i<K; i++) {
            n = n + h[i];
        }
        double[] P = new double[K];
        int c = h[0];
        P[0] = (double) c / n;
        for (int i = 1; i < K; i++) {
            c = c + h[i];
            P[i] = (double) c / n;
        }
        return P;
    }

```

Klasse Histogrammanpassung
Die Methode matchHistograms () berechnet aus dem Histogramm hA und dem Referenzhistogramm hR die Abbildungsfunktion (fhs)

$$f_{\text{hs}}(a) = \min\{\, j \mid (0 \le j < K) \wedge (\mathrm{P}_A(a) \le \mathrm{P}_R(j)) \,\}$$

Methode Cdf () zur Berechnung der kumulierten Verteilungsfunktion

$$\mathrm{P}(i) = \frac{\mathrm{H}(i)}{\mathrm{H}(K-1)} = \frac{\mathrm{H}(i)}{\mathrm{Sum(h)}} = \sum_{j=0}^{i} \frac{\mathrm{h}(j)}{\mathrm{Sum(h)}}$$

$$= \sum_{j=0}^{i} \mathrm{p}(j), \quad \text{für } 0 \le i < K$$

*HistogramMatchingPlugin_.java × *PlotHistogram.java *Histogrammanpassung.java

```java
import ij.IJ;
import ij.ImagePlus;
import ij.WindowManager;
import ij.gui.GenericDialog;
import ij.plugin.filter.PlugInFilter;
import ij.process.ImageProcessor;

public class HistogramMatchingPlugin_ implements PlugInFilter {

    ImagePlus imR;

    public int setup(String arg0, ImagePlus imA) {
        return DOES_8G;
    }

    public void run(ImageProcessor ipA) {
        if (!dialog())
            return;
        //Refrence image
        ImageProcessor ipR = imR.getProcessor();

        // get histograms of Target, Refrence images
        int[] hA = ipA.getHistogram();
        int[] hR = ipR.getHistogram();
        (new PlotHistogram(hA, "H.Target Image(A)")).show();
        (new PlotHistogram(hR, "H.Referance Image(R)")).show();
        (new PlotHistogram(Histogrammanpassung.Cdf(hA), "Cumulative H(A)")).show();
        (new PlotHistogram(Histogrammanpassung.Cdf(hR), "Cumulative H(R)")).show();


        Histogrammanpassung m = new Histogrammanpassung();
        //mapping function fhs
        int[] F = m.matchHistograms(hA, hR);

        // modify the target image
        ipA.applyTable(F);
        int[] hAm = ipA.getHistogram();
        (new PlotHistogram(hAm, "H.modified Image(Á)")).show();
        (new PlotHistogram(Histogrammanpassung.Cdf(hAm), "Cumulative H(Á)")).show();

    }
```

Java implementation
Klasse HistogramMatchingPlugin_
hier die Methode matchHistograms (), die unter Vorgabe des Ausgangshistogramms hA und eines Referenzhistogramms hR die Abbildung map für das zugehörige Ausgangsbild liefert

Die Modifikation des Ausgangsbilds (ipA) durch die Abbildung fhs (F) erfolgt mit der ImageJ-Methode applyTable ()

für Objekte Typ ImageProcessor, an die eine Lookup-Tabelle F als ein dimensionales int-Array der Größe K übergeben wird .
java.lang.Object.ij.process.ImageProcessor
.public abstract void applyTable(int[ ] lut)

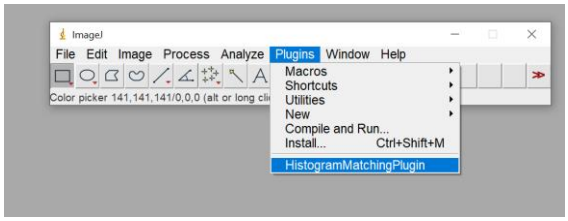
Der Vorteil ist eindeutig - für ein 8-Bit-Grauwertbild z. B. muss in diesem Fall die Abbildungsfunktion (unabhängig von der Bildgröße) nur 256-mal berechnet werden und nicht möglicherweise millionenfach

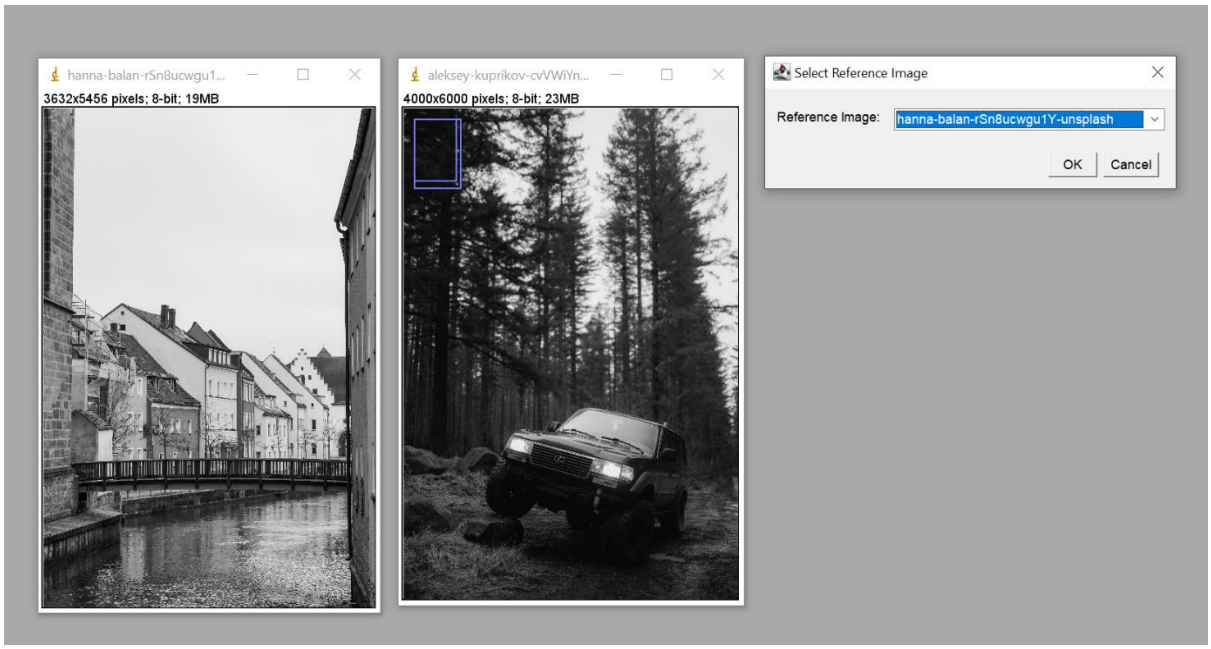
Die Benutzung von Tabellen für Punktoperationen ist also immer dann sinnvoll, wenn die Anzahl der Bildpixel (M7x N) die Anzahl der möglichen Pixelwerte K deutlich übersteigt

**Die Funktionalität verifizieren durch Testbilder zu verifizieren**

Die folgende Beispiele demonstrieren die Anpassung zweier Bilder in Bezug auf ihre Grauwerthistogramme. Eines der Bilder dient als Referenzbild und liefert das Refrenzhistogramm. Das zweite Bild wird modifiziert , dass das resultierende kumulative Histogramm mit dem kumulativen Histogramm des Referenzbildes übereinstimmt.

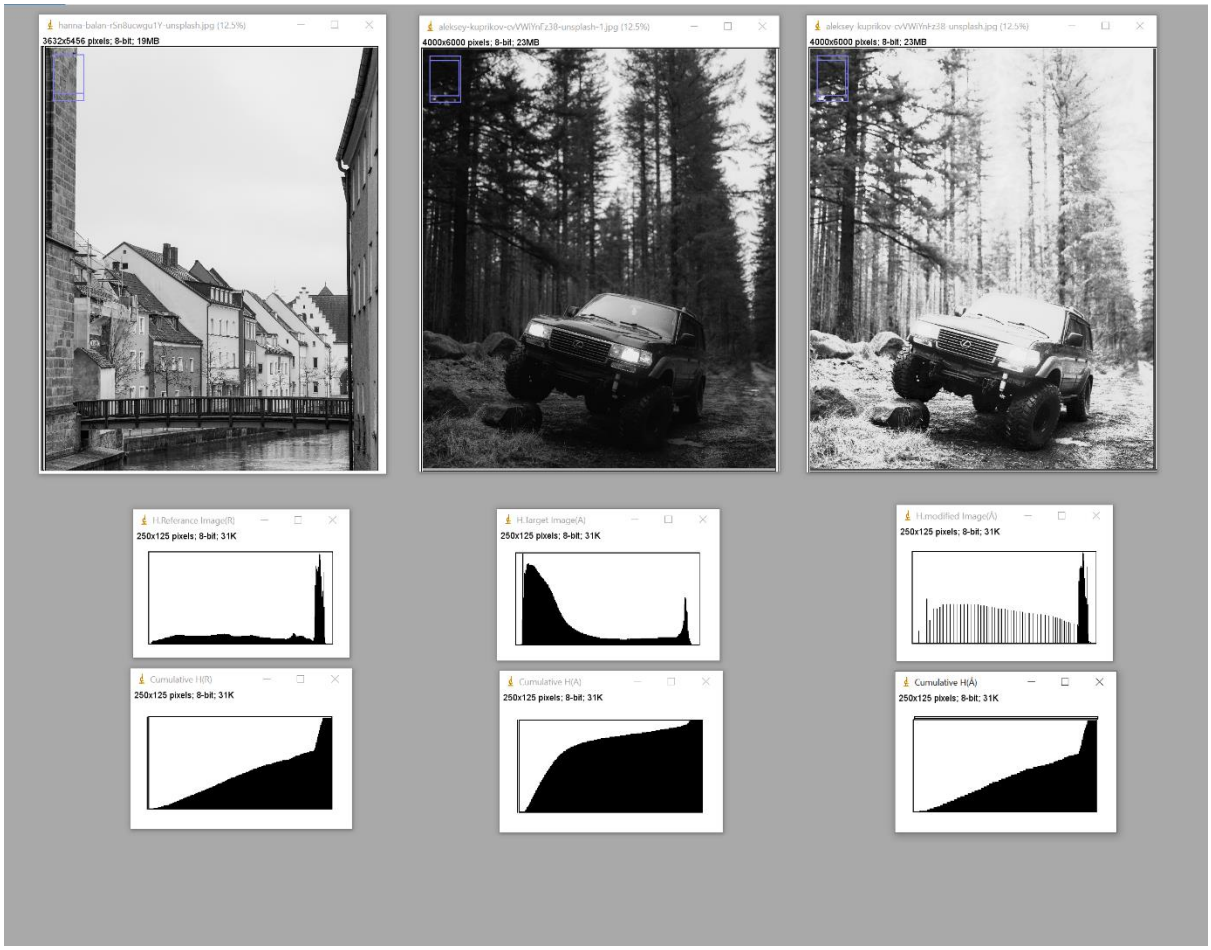
Das endgültige (modifizierte) Bild und das Referenzbild vermitteln einen ähnlichen visuellen Eindruck hinsichtlich Kontrastumfang und Verteilung.

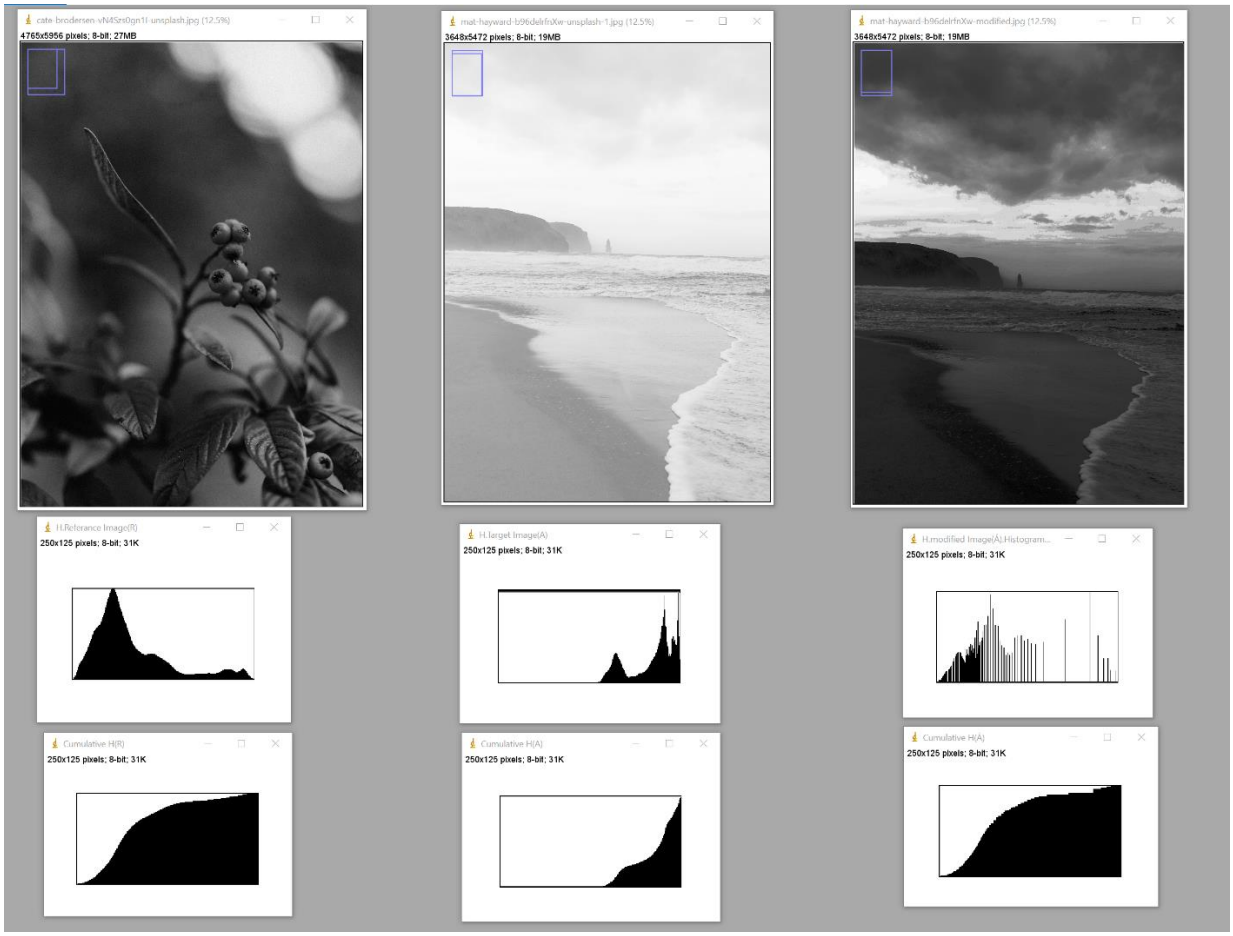


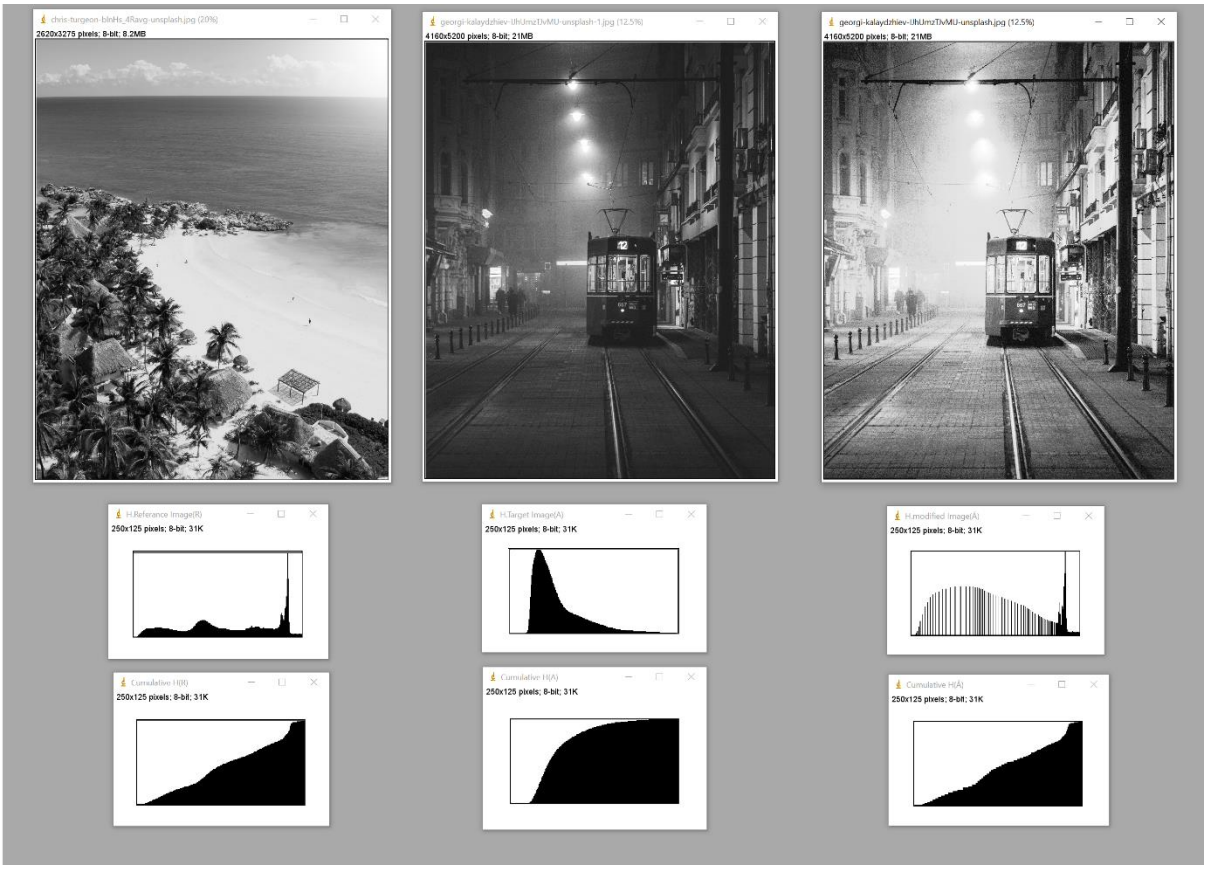



Das Ausgangsbild A ( Mitte) wird durch eine Histogrammanpassung an das Referenzbild R (Links) angeglichen, das Ergebnis ist das modifizierte Bild (Á) ( Rechts). Darunter sind die zugehörigen Histogramme H.R, H.A, H.Á sowie ihre kumulativen Histogramme (bzw. Verteilungsfunktionen) gezeigt.

Die Verteilungsfunktionen des Referenzbilds (R) und des modifizierten Bilds (Á) stimmen offensichtlich weitgehend überein.




cate-brodersen-vN4Snr0qn1I-unsplash.jpg (12.5%)
4765x5956 pixels; 8-bit; 27MB
mat-hayward-b96delrfnXw-unsplash-1.jpg (12.5%)
3648x5472 pixels; 8-bit; 19MB
mat-hayward-b96delrfnXw-modified.jpg (12.5%)
3648x5472 pixels; 8-bit; 19MB
H.Reference Image(R)
250x125 pixels; 8-bit; 31K
H.target Image(A)
250x125 pixels; 8-bit; 31K
H.modified Image(A).Histogram...
250x125 pixels; 8-bit; 31K
Cumulative H(R)
250x125 pixels; 8-bit; 31K
Cumulative H(A)
250x125 pixels; 8-bit; 31K
Cumulative H(A)
250x125 pixels; 8-bit; 31K



**Literatur- und Quellen**


[Burger-2008] Wilhelm Burger, Mark J. Burge, Digital Image Processing – An Algorithmic Introduction using Java. Springer 2008, ISBN 978-1-84628-968-2.

http://imagingbook.com/

imagej documentation and Developer Resources

https://imagej.nih.gov/ij/developer/index.html

JDK 17 Documentation (Java)

https://docs.oracle.com/en/java/javase/17/index.html

Unsplash

https://unsplash.com/