Image Histograms

- Histograms are used to depict image statistics in an easily interpreted visual format
- Useful during image capturing: now already in digital cameras
- Used to improve the visual appearance of an image
- Can also be used to determine what type of processing has been applied to an image.
- Image histogram: describes the frequency of the intensity values that occur in an image

\[ h(i) = \text{the number of pixels in } I \text{ with the intensity value } i \]

\[ h(i) = \text{card}\{(u, v) \mid I(u, v) = i\} \]
- Histograms don’t encode information about the spatial arrangement of pixels in the image
- We cannot reconstruct an image given only its histogram

Interpreting Histograms

- Histograms depict problems that originate during image acquisition
  - Exposure, contrast, dynamic range
- Histograms can be used to detect a wide range of image defects: saturation, spikes and gaps, impact of image compression
Histograms and Exposure

(a)  
(b)  
(c)  

(d)
Histogram and Contrast

Contrast: the range of intensity values effectively used within an image, low contrast, normal contrast, high contrast

Dynamic Range

Dynamic Range: the number of distinct pixel value in an image
Detecting Image Defects

- There is no ideal or optimal histogram shape. It depends on the image and on the application
- Image Defects:
  - Saturation: the illumination values lying outside of the sensor’s range are mapped to its maximum or minimum values: spike at the tails
  - Spikes and Gaps in manipulated images. Why?
  - Impact of image compression

- Histograms show the impacts of image compression
- Ex: in GIF compression, the dynamic range is reduced to only few intensities (quantization)
- Ex: JPEG compression on a line graphics.

Computing Histograms

```java
public class Compute_Histogram implements PlugInFilter {
  public int setup(String arg, ImagePlus imp) {
    return DOES_BG + NO_CHANGES;
  }

  public void run(ImageProcessor ip) {
    int[] H = new int[256]; // histogram array
    int w = ip.getWidth();
    int h = ip.getHeight();
    for (int v = 0; v < h; v++) {
      for (int u = 0; u < w; u++) {
        int i = ip.getPixel(u, v);
        H[i] = H[i] + 1;
      }
    }
    // Histogram H[] can now be used
  }
}
```
- Histograms of images with more than 8 bits:
  - Binning

  \[ h(j) = \text{card} \{ (u, v) \mid a_j \leq I(u, v) < a_{j+1} \} \text{ for } 0 \leq j < B \]

  \[ a_j = j \cdot \frac{K}{B} = j \cdot k_B \]

  - B: number of bins
  - Ex: B=256 for 14 bit image
  - K=16384, bin width = 64

  \[ \begin{array}{l}
  h(0) \leftarrow 0 \leq I(u, v) < 64 \\
  h(1) \leftarrow 64 \leq I(u, v) < 128 \\
  h(2) \leftarrow 128 \leq I(u, v) < 192 \\
  \vdots \\
  h(j) \leftarrow a_j \leq I(u, v) < a_{j+1} \\
  \vdots \\
  h(255) \leftarrow 16320 \leq I(u, v) < 16384 \\
  \end{array} \]

### Color Image Histograms

(a) ![Image](image1.png)  
(b) ![Histogram](histogram1.png)  
(c) ![Image](image2.png)  
(d) ![Image](image3.png)  
(e) ![Image](image4.png)  
(f) ![Histogram](histogram2.png)  
(g) ![Histogram](histogram3.png)  
(h) ![Histogram](histogram4.png)
Color Image Histograms

- For color images, two kind of histograms:
  - Intensity histogram
  - Individual Color Channel Histograms
- Both provides useful information about lighting, contrast, dynamic range and saturation effects for individual color components
- They provide no information about the actual color distribution!

Cumulative Histograms

\[ H(i) = \sum_{j=0}^{i} h(j) \quad \text{for} \quad 0 \leq i < K \]

\[ H(i) = \begin{cases} 
  h(0) & \text{for} \quad i = 0 \\
  H(i-1) + h(i) & \text{for} \quad 0 < i < K 
\end{cases} \]

\[ H(K-1) = \sum_{j=0}^{K-1} h(j) = M \cdot N \]