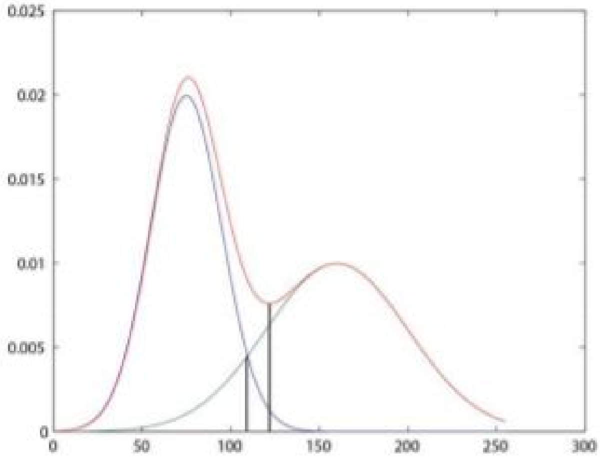
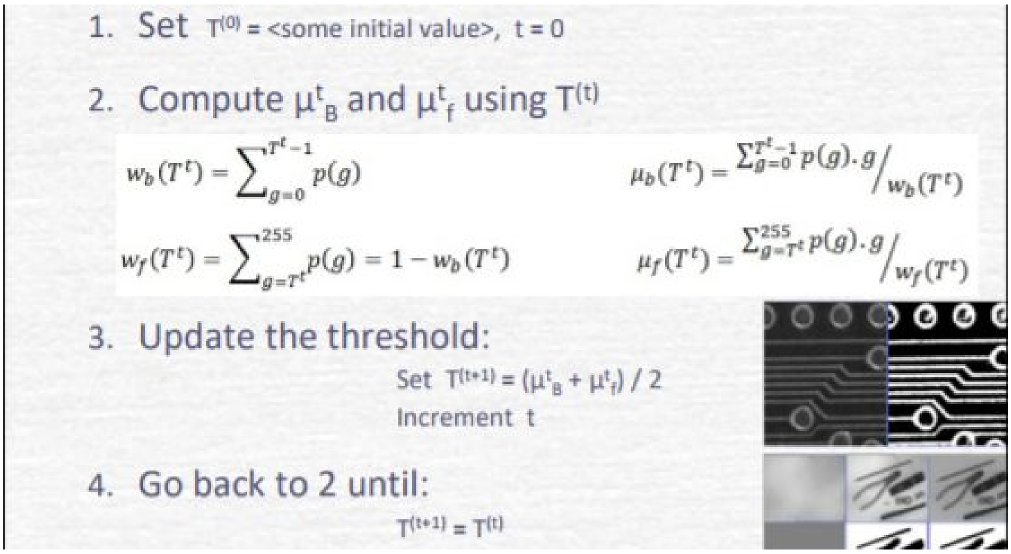
Binary

- **Thresholding**
    - Created using a grey scale image
        - Have 8 bits per pixel, theory is to reduce this down to 1 per pixel (however it is common for 255 to be used to facilitate viewing and processing of the image)
    - Binary thresholding - some threshold T
        - ```
          for all pixels
             g(i,j) = 1 for f(i,j) ≥ T
                    = 0 for f(i,j) < T
          ```
    - **Simplest approach**
        - **Look Up Table**
        - ```
          for all grey levels
             LUT(k) = 1 for k ≥ T
                    = 0 for k < T
          for all pixels
             g(i,j) = LUT( f(i,j) )
          ```
    - **Short coming:** foreground and background need to be clearly separated in the image, without this it will be near impossible to accurately segment it
    - How do we determine the best threshold...
- **Threshold Detection**
    - Manual setting (*magic numbers*)
        - What if the lighting changes? Or we have a sample set bigger than a single image?
    - **Determine automatically**
        - Notation
            - Image: f(i,j)
            - Histogram: h(g)
            - Probability Distribution: p(g) = h(g)/ $\sum_g h(g)$
        - Automatic threshold detection is vital - even in controlled settings, lighting declines consistently over time
        - **Bi-modal histogram analysis**
            - We can assume that the foreground and background are centred around 2 greyscale values
            - So we can select the antimode between these two peaks as the threshold
            - However histograms are generally very noisy - many local min and max
            - Smoothing? Variable step?
                - **These skew the anti-mode however** reducing the performance of the thresholding

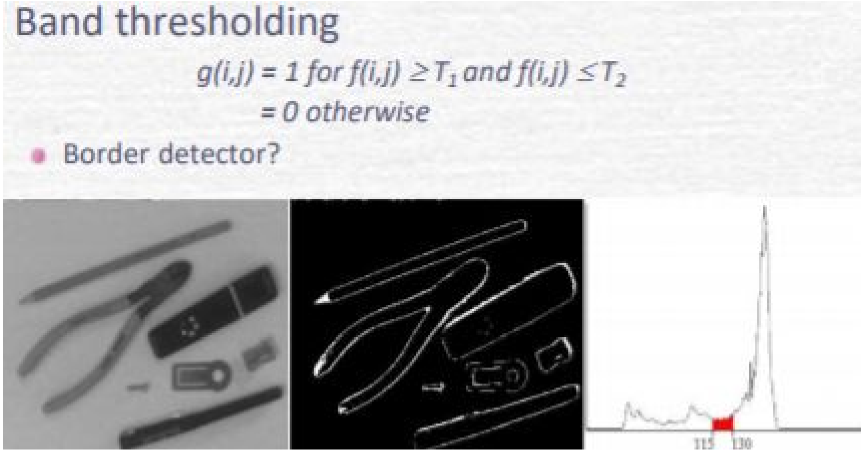
- **Optimal Thresholding**
    - Bi-modal analysis fails when the modes move closer together or there is significant noise
    - Consider the two normal distributions and their summation below..
    - 

        - **The optimal threshold is where they intersect (~110)**
        - The anti mode is to the right of that (~125)
    - **If we can model the histogram as the sum of two normal distributions, we can use optimal thresholding**
    - 

- **Otsu Thresholding**
    - What if the distributions aren't normal?
    - Minimize pixel spread on either side of the threshold
    - Consider all thresholds - Select the threshold which minimizes within class variance

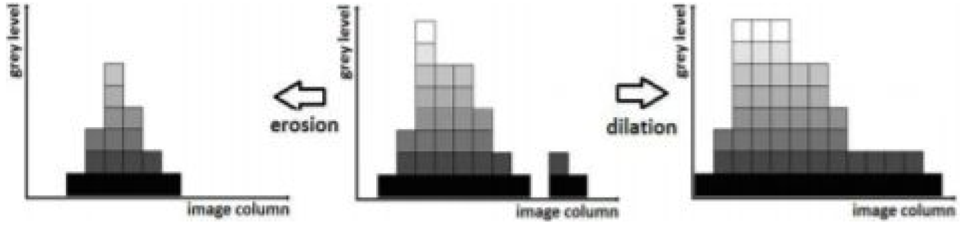
- $$\sigma_W^2(T) = w_f(T)\sigma_f^2(T) + w_b(T)\sigma_b^2(T)$$
  $$w_f(T) = \sum_{g=T}^{255} p(g) \qquad \sigma_f^2(T) = \sum_{g=T}^{255} p(g).(g-\mu_f(T))^2 \Big/ w_f(T)$$
  $$w_b(T) = \sum_{g=0}^{T-1} p(g) \qquad \sigma_b^2(T) = \sum_{g=0}^{T-1} p(g).(g-\mu_b(T))^2 \Big/ w_b(T)$$
  $$\mu_f(T) = \sum_{g=T}^{255} p(g).g \Big/ w_f(T) \qquad \mu_b(T) = \sum_{g=0}^{T-1} p(g).g \Big/ w_b(T)$$
    - Where wf(T) and wb(T) are the portions of points in the foreground/background respectively
    - σ_f^2 *and* σ_b^2 are the variances of the greyscale values within the foreground/background
    - And $\mu_f(T)$ *and* $\mu_b(T)$ are the mean of the foreground/background values
- **The threshold with the smallest within class variance is also the threshold with the largest between class variance**
    - This is easier to calculate
- $$\sigma_B^2(T) = w_f(T)w_b(T)\left(\mu_f(T) - \mu_b(T)\right)^2$$

- **Variations**
    - **Adaptive Thresholding**
    - In general terms
        - Divide image into sub-images
        - Compute thresholds for sub-images
        - Interpolate thresholds for every point using bilinear interpolation
    - This isn't strictly implemented by openCV
        - Compute difference between current pixel and the local average of a block_size*block_size area around the current pixel
        - Subtract an offset from the difference and if it is >0 or <0 set it to 255 or to 0
        - Essentially compute new threshold for every single pixel based on the region
    - **Band Thresholding**
    - In band thresholding 2 thresholds are used, one below and one above object pixels
    - Border detector?

- 

- **Semi Thresholding**
- Not really used for anything other than being human readable
- If greyscale value is greater than threshold, it retains its greyscale value, otherwise it is set to 0
- **Multi-Level Thresholding**
- Threshold all colours separately?
- It is possible to threshold in 3D colour space (defining a 3D threshold region, accepting pixels which exist within this space)
- **Binary images tend to be quite rough/pixelated… need to clean them**

- **Mathematical Morphology**
  - Can't use normal smoothing operations (only 2 values, and require distinct edges)
  - Need to remove the noise on the edges of regions
  - **Use mathematical morphology operations, which treat images as sets**
  - **Erosion**
    - Minkowski set subtraction
    - $X \ominus B = \{ p \in \varepsilon^2 ; p+b \in X \text{ for every } b \in B \}$
    - Removes noise and narrow bridges - used to remove borders
  - **Dilation**
    - Minkowski set addition
    - $X \oplus B = \{ p \in \varepsilon^2 ; p = x+b, x \in X \text{ and } b \in B \}$
    - Fill small holes and gulfs in B.I. - used to add borders
  - **Opening**
    - Erosion followed by dilation
    - $X \circ D = (X \ominus D) \oplus D$
    - Maintains approximate size of objects
  - **Closing**
    - Dilation followed by erosion
    - $X \bullet D = (X \oplus D) \ominus D$
    - Maintains approximate size of objects, but distorts the shape

- **Mathematical morphology can also be used for greyscale/colour images**
    - Each greyscale level (g) is considered to be a set
    - All points >= g undergo the morphology
    - 

    - Can use this to locate local maxima and minima