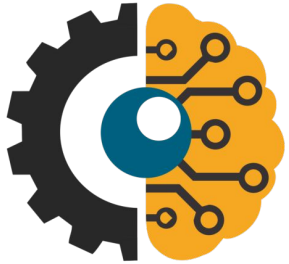
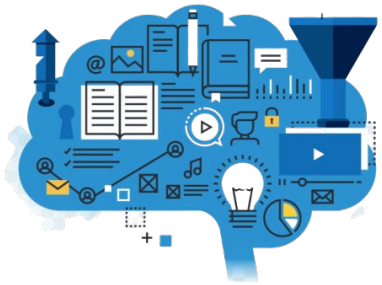


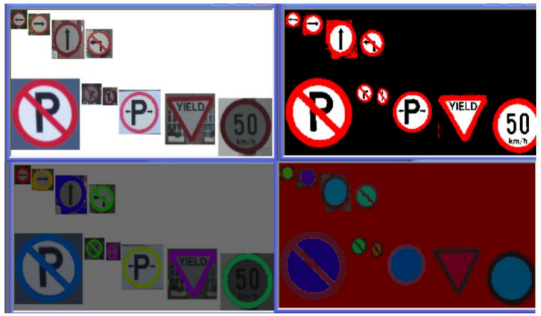


Computer Vision

Binary Vision

Binary Vision

- Thresholding
- Threshold Detection
- Variations
- Mathematical Morphology
- Connectivity
- Objects of interest vs background

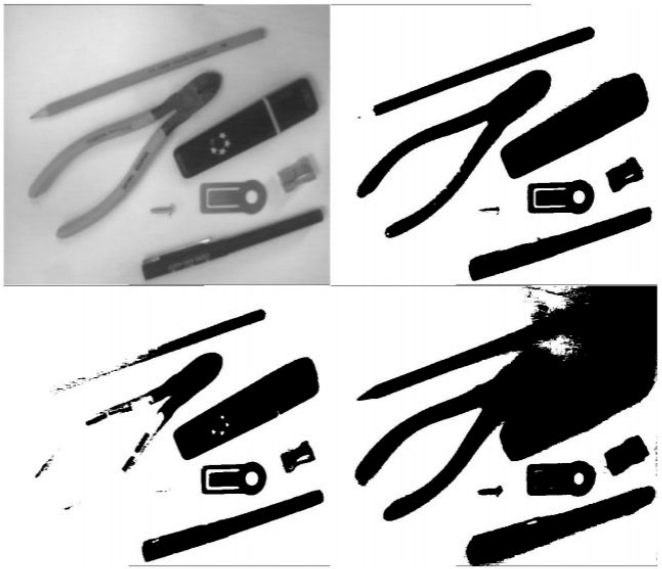


Based on *A Practical Introduction to Computer Vision with OpenCV* by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014

```
threshold(gray_image,binary_image,threshold, 255, THRESH_BINARY);
```

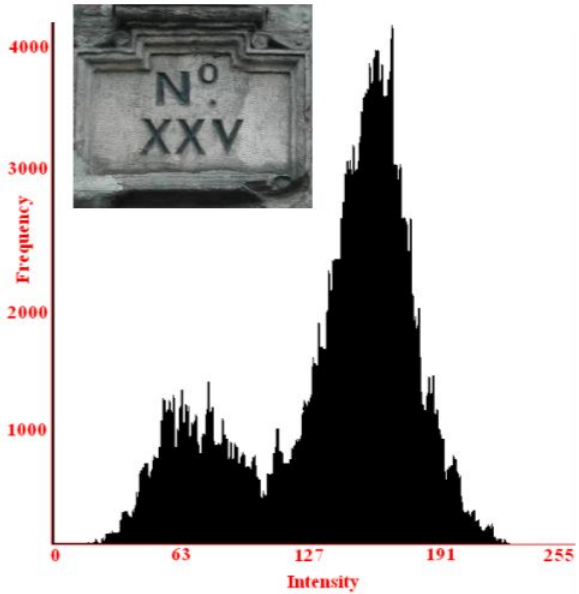
Thresholding

- Distinct foreground and background
- How to determine the best threshold



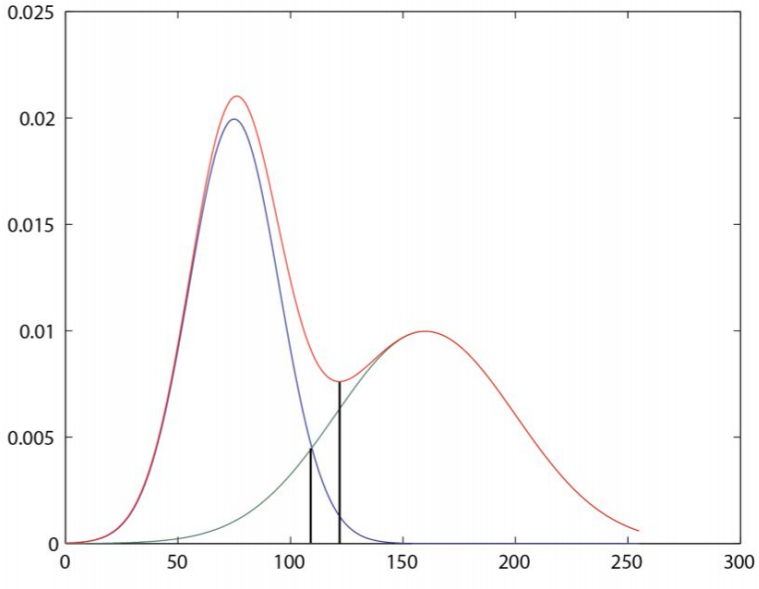
Threshold Detection

- Manual Setting
- Changing lighting
- Need to determine automatically
- Techniques:
  - Image
  - Histogram
  - Probability Distribution



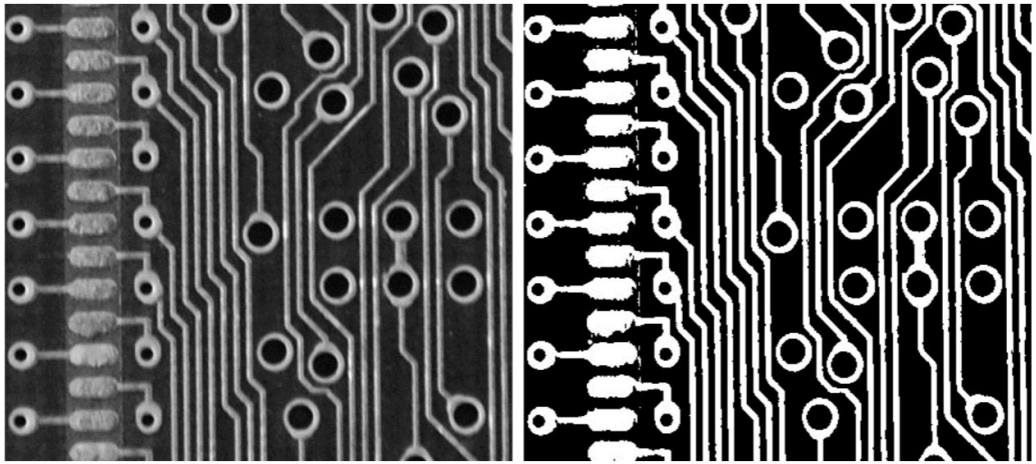
Threshold Detection - Optimal Thresholding

- Model as two normal distributions



Threshold Detection - Otsu Thresholding

- If it's not two normal distributions
- Minimizes the spread of the pixels



```
threshold( gray_image, binary_image, threshold,
                       255, THRESH_BINARY | THRESH_OTSU );
```

Adaptive Thresholding

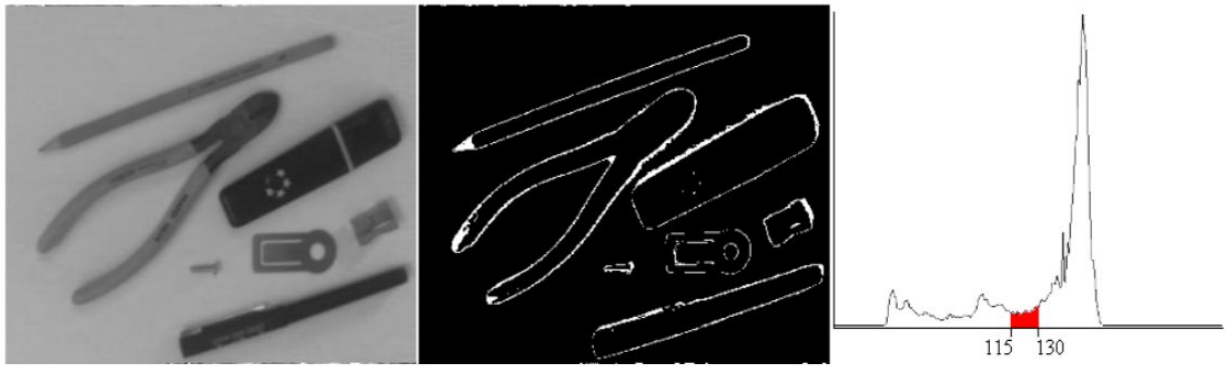
- Divide the image into sub-images
- Compute thresholds for each sub-image
- Interpolate thresholds for every point using bilinear interpolation

Adaptive Thresholding in OpenCV

```
adaptiveThreshold( gray_image,binary_image,output_value,
     ADAPTIVE_THRESH_MEAN_C,THRESH_BINARY,
     block_size,offset );
```

Band Thresholding

- Could be used for edge detection

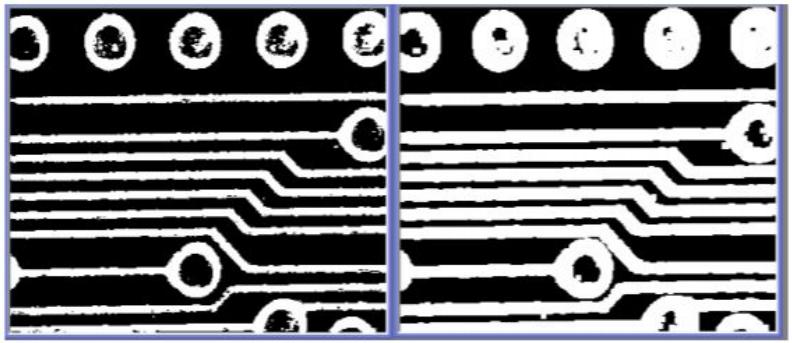


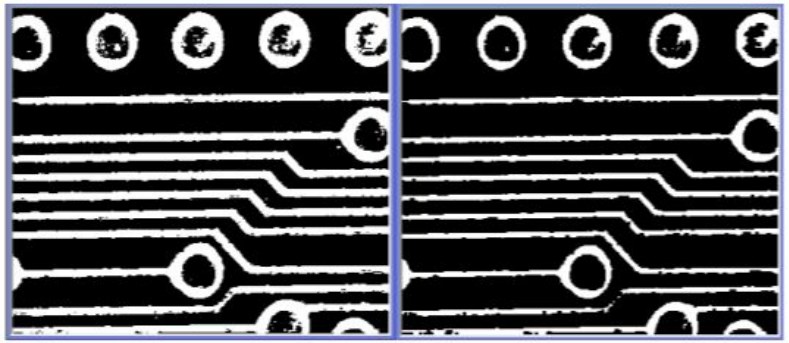

```
threshold( image, binary1, low_threshold, 255, THRESH_BINARY );
threshold( image, binary2, high_threshold, 255, THRESH_BINARY_INV );
bitwise_and( binary_image1, binary_image2,  band_thresholded_image );
```

Mathematical Morphology

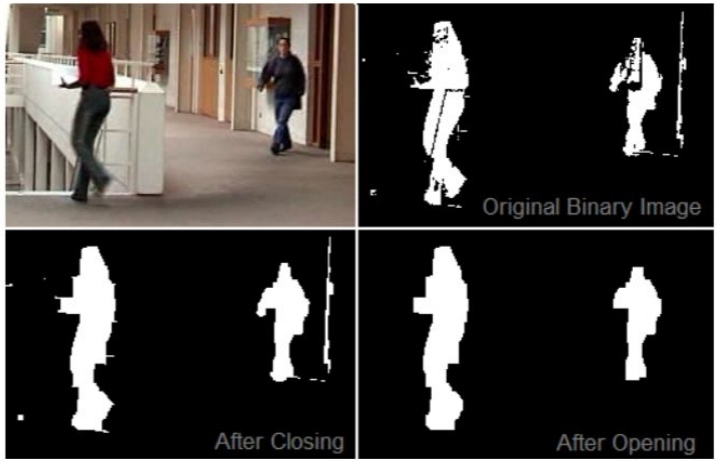
- Based on algebra of non-linear operators operating on object shape
- Performs many tasks better and more quickly than standard approaches
- Seperate part of image analysis
- Main uses:
  - Pre-processing
  - Object structure enhancement
  - Segmentation
  - Description of objects

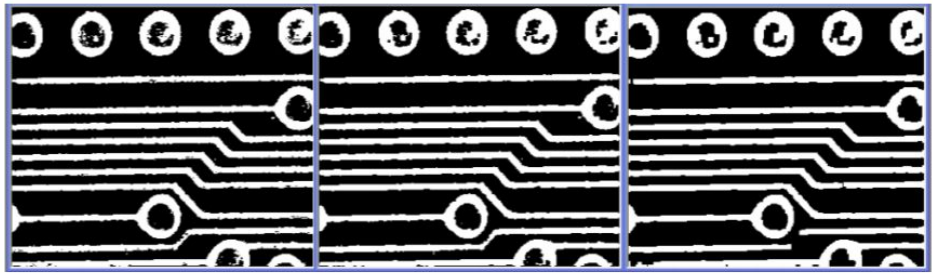
Mathematical Morphology - Dilation and Erosion





Mathematical Morphology - Closing and Opening





Mathematical Morphology in OpenCV

```
dilate( binary_image, dilated_image, Mat());
```

```
Mat structuring_element( 5, 5, CV_8U, Scalar(1) );
dilate( binary_image, dilated_image, structuring_element);
```

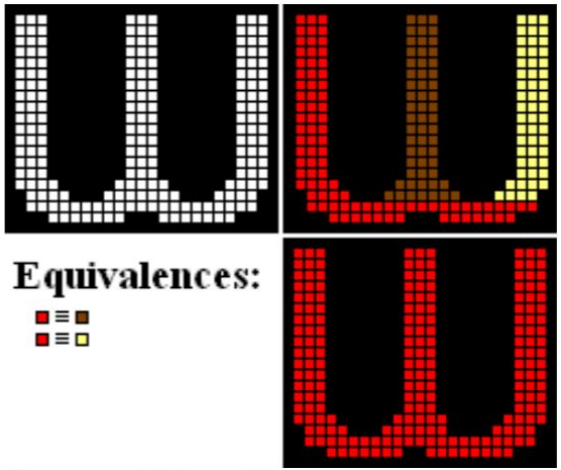
```
erode( binary_image, eroded_image, Mat());
```

```
Mat structuring_element( 5, 5, CV_8U, Scalar(1) );
erode( binary_image, eroded_image, structuring_element);
```

```
Mat five_by_five_element( 5, 5, CV_8U, Scalar(1) );
morphologyEx( binary_image, opened_image,
                          MORPH_OPEN,  five_by_five_element );
  morphologyEx( binary_image, closed_image,
                          MORPH_CLOSE, five_by_five_element );
```

Connectivity

- Search image row by row
  - Label each non-zero pixel
  - Assign new label if previous pixels are background
  - Else pick any label from previous pixels
  - Note equivalence if any of the other pixels have a different label
- Relabel equivalent Labels



Extracting Regions in OpenCV


```
vector<vector<Point>> contours;
vector<Vec4i> hierarchy;
findContours( binary_image, contours, hierarchy,
                  CV_RETR_TREE, CV_CHAIN_APPROX_NONE );
```

Labelling Regions in OpenCV


```
for (int contour=0; (contour < contours.size()); contour++)
  {
    Scalar colour( rand()&0xFF,rand()&0xFF,rand()&0xFF );
    drawContours( contours_image, contours, contour, colour,
                                                CV_FILLED, 8, hierarchy );
  }
```