Thresholding

Binary thresholding for all pixels:
\[ g(i,j) = \begin{cases} 1 & \text{for } f(i,j) \geq T \\ 0 & \text{for } f(i,j) < T \end{cases} \]

Simple scenes?

Look Up Table for all grey levels:
\[ \text{LUT}(k) = \begin{cases} 1 & \text{for } k \geq T \\ 0 & \text{for } k < T \end{cases} \]

for all pixels:
\[ g(i,j) = \text{LUT}(f(i,j)) \]

Objects of interest vs. background

Threshold Detection

Manual Setting

Issue of changing lighting

Need to determine automatically

For the techniques which follow:

- Image – \( f(i,j) \)
- Histogram – \( h(g) \) of values in \( f(i,j) \)
- Normalised Probability Distribution – \( p(g) = h(g) / \sum g h(g) \)

Threshold Detection – Otsu Thresholding

Minimize the spread of the pixels...

- Smallest within class variance
  \[ \sigma_w^2(T) = w_y(T) \sigma_y^2(T) + w_b(T) \sigma_b^2(T) \]
  \[ w_y(T) = \sum_{g=y} \pi_y(g) \]
  \[ \sigma_y^2(T) = \frac{\sum_{g=y} \pi_y(g) \cdot (g - \mu_y(T))^2}{w_y(T)} \]
  \[ w_b(T) = \sum_{g=b} \pi_b(g) \]
  \[ \sigma_b^2(T) = \frac{\sum_{g=b} \pi_y(g) \cdot (g - \mu_b(T))^2}{w_b(T)} \]
  \[ \mu_y(T) = \frac{\sum_{g=y} \pi_y(g) \cdot g}{w_y(T)} \]
  \[ \mu_b(T) = \frac{\sum_{g=b} \pi_b(g) \cdot g}{w_b(T)} \]

- Largest between class variance
  \[ \sigma_y^2(T) = w_y(T) \sigma_y^2(T) (\mu_y(T) - \mu_b(T))^2 \]

Threshold Detection – Otsu Thresholding

Thresholding

Binary thresholding for all pixels:
\[ g(i,j) = \begin{cases} 1 & \text{for } f(i,j) \geq T \\ 0 & \text{for } f(i,j) < T \end{cases} \]

Simple scenes?

Look Up Table for all grey levels:
\[ \text{LUT}(k) = \begin{cases} 1 & \text{for } k \geq T \\ 0 & \text{for } k < T \end{cases} \]

for all pixels:
\[ g(i,j) = \text{LUT}(f(i,j)) \]

Objects of interest vs. background

Thresholding

Binary thresholding for all pixels:
\[ g(i,j) = \begin{cases} 1 & \text{for } f(i,j) \geq T \\ 0 & \text{for } f(i,j) < T \end{cases} \]

Simple scenes?

Look Up Table for all grey levels:
\[ \text{LUT}(k) = \begin{cases} 1 & \text{for } k \geq T \\ 0 & \text{for } k < T \end{cases} \]

for all pixels:
\[ g(i,j) = \text{LUT}(f(i,j)) \]

Objects of interest vs. background

Threshold Detection

Manual Setting

Issue of changing lighting

Need to determine automatically

For the techniques which follow:

- Image – \( f(i,j) \)
- Histogram – \( h(g) \) of values in \( f(i,j) \)
- Normalised Probability Distribution – \( p(g) = h(g) / \sum g h(g) \)

Threshold Detection – Otsu Thresholding

Minimize the spread of the pixels...

- Smallest within class variance
  \[ \sigma_w^2(T) = w_y(T) \sigma_y^2(T) + w_b(T) \sigma_b^2(T) \]
  \[ w_y(T) = \sum_{g=y} \pi_y(g) \]
  \[ \sigma_y^2(T) = \frac{\sum_{g=y} \pi_y(g) \cdot (g - \mu_y(T))^2}{w_y(T)} \]
  \[ w_b(T) = \sum_{g=b} \pi_b(g) \]
  \[ \sigma_b^2(T) = \frac{\sum_{g=b} \pi_b(g) \cdot (g - \mu_b(T))^2}{w_b(T)} \]
  \[ \mu_y(T) = \frac{\sum_{g=y} \pi_y(g) \cdot g}{w_y(T)} \]
  \[ \mu_b(T) = \frac{\sum_{g=b} \pi_b(g) \cdot g}{w_b(T)} \]

- Largest between class variance
  \[ \sigma_y^2(T) = w_y(T) \sigma_y^2(T) (\mu_y(T) - \mu_b(T))^2 \]
Variations – Adaptive Thresholding

The adaptive thresholding algorithm is:
- Divide the image into sub-images,
- Compute thresholds for all sub-images,
- Interpolate thresholds for every point using bilinear interpolation.

OpenCV version:
\[
f(i,j) - \left( \frac{\sum_{a=-m}^{m} \sum_{b=-m}^{m} f(i+a,j+b)}{2m+1} \right)^2 > offset \\
g(i,j) = 255 \\
g(i,j) = 0
\]

Cleaning Binary Images

- Cannot use normal smoothing operations
- Need to remove noisy points and smooth binary region boundaries.
- Mostly we use operations originally defined as part of “mathematical morphology” (which treats images as sets).
- Most common operations:
  - Erosion
  - Dilation
  - Opening
  - Closing

Mathematical Morphology – Dilation

- Minkowski set addition:
  \[ X \oplus B = \{ p \in \mathbb{R}^2 | p = x + b, x \in X \text{ and } b \in B \} \]
- B is the structuring element
- B is typically isotropic
- Adds pixels around borders

**Effects:**
- Makes 'regions' bigger
- Fills small holes
- Joins close 'regions'

Mathematical Morphology – Erosion

- Minkowski set subtraction:
  \[ X \ominus B = \{ p \in \mathbb{R}^2 | p \ominus b \in X \text{ for every } b \in B \} \]
- Again B is typically isotropic
- Removes border pixels

**Effects:**
- Makes 'regions' smaller
- Removes noise
- Removes narrow bridges
Mathematical Morphology –

Opening: \( X \ominus D = (X \Theta D) \oplus D \)
- Removes noise
- Removes narrow bridges
- Roughly maintains ‘region’ size
- Smooths shape

Closing: \( X \oplus D = (X \ominus D) \Theta D \)
- Fills small holes
- Joins close ‘regions’
- Roughly maintains ‘region’ size
- Smooths shape

Mathematical Morphology – Application

Opening: \( X \ominus D = (X \Theta D) \oplus D \)
- Removes noise
- Removes narrow bridges
- Roughly maintains ‘region’ size
- Smooths shape

Closing: \( X \oplus D = (X \ominus D) \Theta D \)
- Fills small holes
- Joins close ‘regions’
- Roughly maintains ‘region’ size
- Smooths shape

Isotropic structuring element:
- Eliminates small image details
- Properties
  - \( X \ominus D = (X \Theta D) \oplus D \) and \( X \oplus D = (X \ominus D) \Theta D \)

Mathematical Morphology – OpenCV Code

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

Mathematical Morphology – Greyscale / Colour

- Grey scale image
  - One set per grey level (g)
  - All points \( \geq g \)
  - Operations on each set separately
- Colour
  - One set per level per channel
  - E.g. RGB image has \( 255 \times 3 \) sets.

Mathematical Morphology – Local maxima

Can be used to locate local maxima and minima
- Dilate & compare (to find local maxima)
- Threshold the original image
- Logical AND the results together to find ‘high’ local maxima

Mathematical Morphology – Local maxima

Mat dilated, thresholded_input, local_maxima, thresholded_8bit;
dilate( input, dilated, Mat());
compare( input, dilated, local_maxima, CMP_EQ );
threshold( input, thresholded_input, threshold, 255, THRESH_BINARY );
thresholded_input.convertTo( thresholded_8bit, CV_8U );
bitwise_and( local_maxima, thresholded_8bit, local_maxima );