Otsu Thresholding and Canny Edge Detection

This document demonstrates the use of built-in MATLAB functions and corresponding step-by-step implementations of two common image processing techniques: Otsu Thresholding and Canny Edge Detection. Each method is explained briefly, followed by the MATLAB code using built-in functions and a manual version that achieves similar results.

1. Otsu Thresholding

Otsu's method automatically determines an optimal threshold to separate the foreground from the background in a grayscale image by maximizing the between-class variance.

Built-in MATLAB Code	**Step-by-step Implementation**

`I = imread('coins.png');`
`level = graythresh(I);`
`BW1 = imbinarize(I, level);`
`imshow(BW1); title('Built-in Otsu Thresholding');`
	`counts = imhist(I);`
	`p = counts / sum(counts);`
	
	`maxVar = 0; bestThresh = 0;`
	`for t = 1:255`
	`  w0 = sum(p(1:t));`
	`  w1 = sum(p(t+1:end));`
	`  if w0 == 0 || w1 == 0, continue; end`
	
	`  mu0 = sum((0:t-1) .* p(1:t)') / w0;`
	`  mu1 = sum((t:255) .* p(t+1:end)') / w1;`
	
	`  sigma_b = w0 * w1 * (mu0 - mu1)^2;`
	
	`  if sigma_b > maxVar`
	`  maxVar = sigma_b;`
	`  bestThresh = t;`
	`  end`
	`end`
	
	`BW2 = I > bestThresh;`
	`imshow(BW2); title(['Manual Otsu, T = ', num2str(bestThresh)]);`

2. Canny Edge Detection

Canny Edge Detection identifies the edges in an image using a multi-stage process: Gaussian smoothing, gradient calculation, non-maximum suppression, and hysteresis thresholding. It’s effective at detecting clean, thin edges.

Built-in MATLAB Code	**Step-by-step Implementation**

`BW_canny = edge(I, 'Canny');`
`imshow(BW_canny); title('Built-in Canny Edge Detection');`
	`I_smooth = imgaussfilt(I, 1);`
	`[Gx, Gy] = imgradientxy(I_smooth);`
	`[Gmag, Gdir] = imgradient(Gx, Gy);`
	
	`edge_simple = Gmag > 50;`
	`imshow(edge_simple); title('Manual Edge (Thresholded Gradient)');`