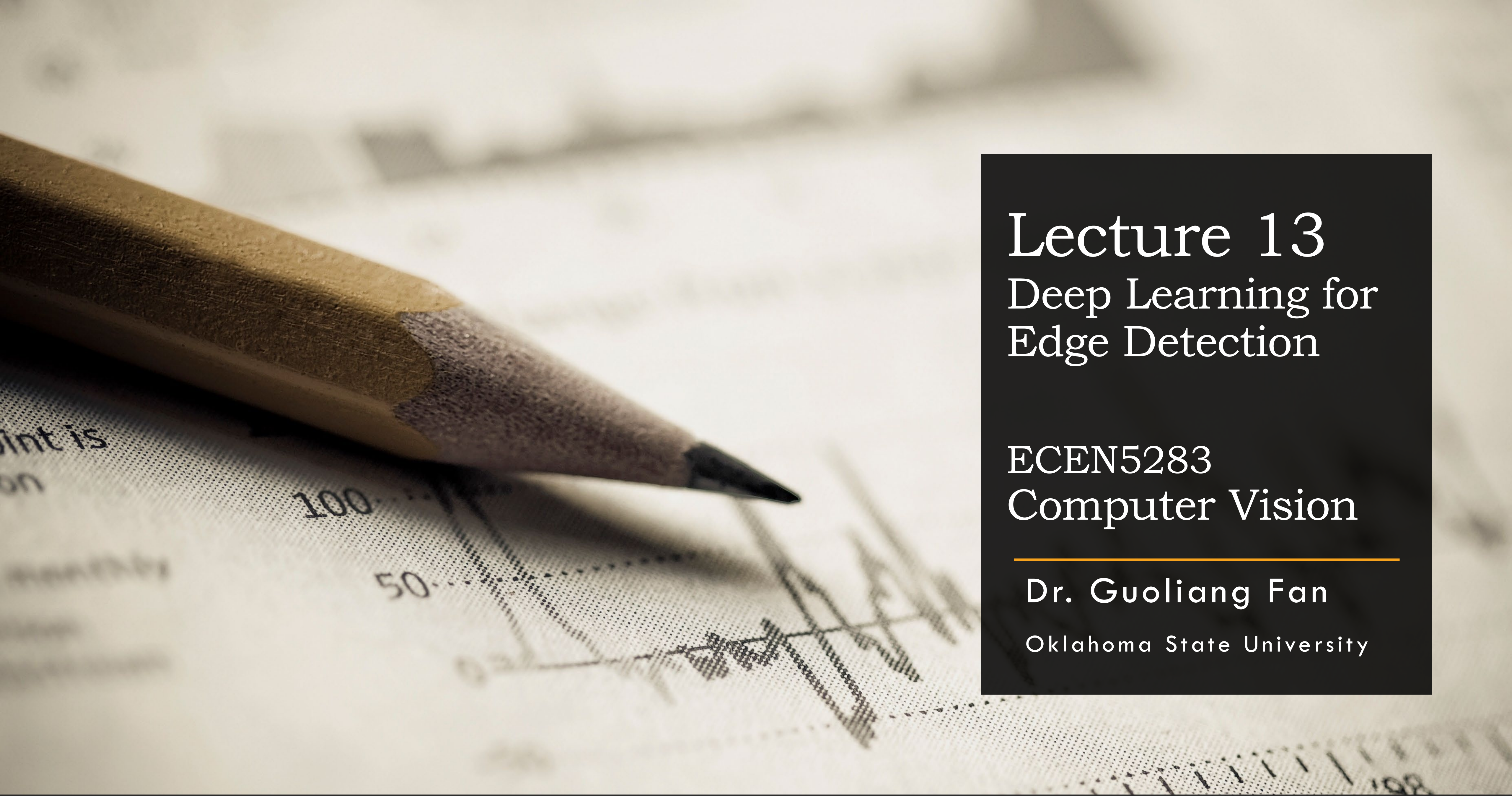


Lecture 13
Deep Learning for Edge Detection

ECEN5283 Computer Vision

Dr. Guoliang Fan

Oklahoma State University

Goals

To review Canny edge detection.

To present two deep learning approaches to edge detection.

To introduce Project 2 that is due on March 2.

Canny Edge Detection

Step 1: We first compute the two first-order derivative maps ($f_x(x,y), f_y(x,y)$).

Step 2: We can compute the magnitude and orientation of the gradient $M(x,y)$ and direction $\alpha(x,y)$ maps for each pixel.

Step 3: Apply non-maximum suppression to the gradient magnitude image and find the local maximum along the gradient direction.

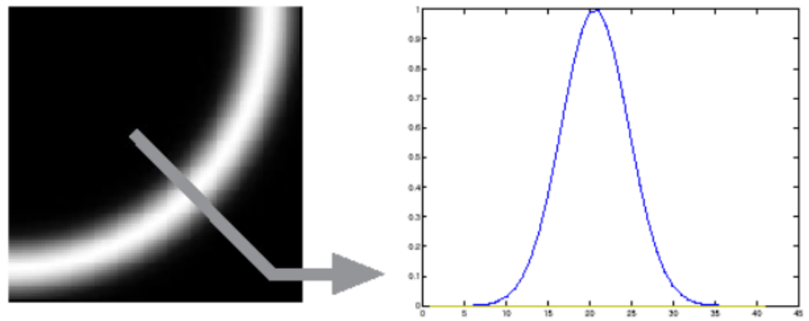
Step 4: Use connectivity analysis to detection and link weak edges.

$$f_x = \nabla_x G_\sigma * f(x,y)$$

$$f_y = \nabla_y G_\sigma * f(x,y)$$

$$M(x,y) = \sqrt{f_x^2 + f_y^2}$$

$$\alpha(x,y) = \tan^{-1}\left(\frac{f_y}{f_x}\right)$$



Edge Linking and Tracking: Double Thresholds and Hysteresis

Edge linking by double thresholds:

- It is essential to filter out edge pixels with a weak gradient value and preserve edge pixels with a high gradient value.
- We use a high-threshold to start edge curves and a low threshold to continue them. The two threshold values are empirically determined

Edge tracking by hysteresis:

- To preserve edge connection, window-based analysis is applied by looking at a weak edge pixel and its 8-connected neighborhood pixels.
- As long as there is one strong edge pixel that is involved in the blob, that weak edge point can be identified as one that should be preserved.

Canny-Net:

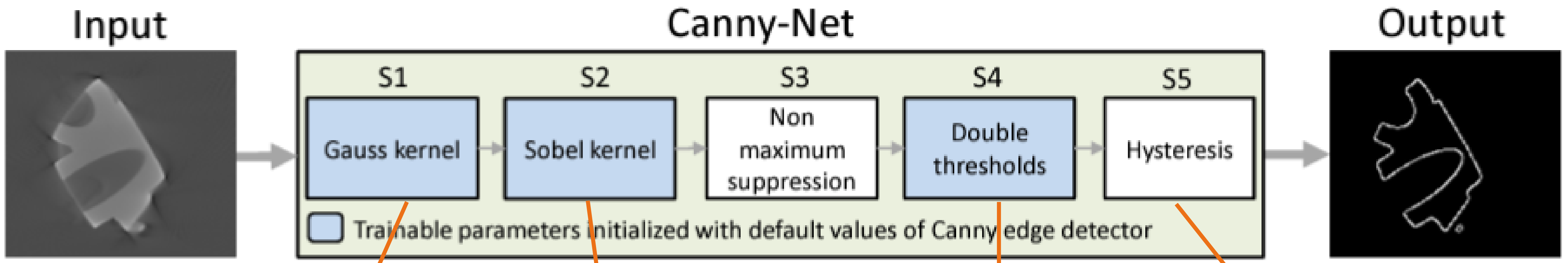


Figure 2: Complete architecture of Canny-Net consisting of five steps (S1-S5). The input image contains noticeable artifacts. Canny-Net's architecture is equivalent to the Canny edge detector and the trainable parameters (blue) of Canny-Net are initialized with the default values of the Canny edge detector. The network yields a binary image with background and edge pixels.

$$\mathbf{B} = \begin{bmatrix} 0.1070 & 0.1131 & 0.1070 \\ 0.1131 & 0.1196 & 0.1131 \\ 0.1070 & 0.1131 & 0.1070 \end{bmatrix}$$

$$\mathbf{G_x} = \begin{bmatrix} -0.5 & 0 & 0.5 \\ -1 & 0 & 1 \\ -0.5 & 0 & 0.5 \end{bmatrix}$$

$$\mathbf{G_y} = \begin{bmatrix} -0.5 & -1 & -0.5 \\ 0 & 0 & 0 \\ 0.5 & 1 & 0.5 \end{bmatrix}$$

$$\mathrm{E}_1 = max(F) \cdot R_1,$$

$$\mathrm{E}_2 = E_1 \cdot R_2.$$

$(F$: the gradient image, $R_1 = 0.2$, $R_2 = 0.1)$

$$\mathbf{C} = \begin{bmatrix} 1.25 & 1.25 & 1.25 \\ 1.25 & 1.25 & 1.25 \\ 1.25 & 1.25 & 1.25 \end{bmatrix}$$

Wittmann J, Herl G. Canny-Net: Known Operator Learning for Edge Detection, in Proc. the 12th Conference on Industrial Computed Tomography, Furth, Germany (iCT 2023), www.ict2023.org 2017.

Kernels: Before and After

$$\mathbf{B} = \begin{bmatrix} 0.1070 & 0.1131 & 0.1070 \\ 0.1131 & 0.1196 & 0.1131 \\ 0.1070 & 0.1131 & 0.1070 \end{bmatrix}$$

$$\Big\downarrow$$

$$\mathbf{B} = \begin{bmatrix} -0.0449 & 0.0990 & 0.0170 \\ 0.0796 & 0.3347 & -0.0800 \\ 0.0350 & -0.0797 & 0.1042 \end{bmatrix}$$

$$\mathbf{G_x} = \begin{bmatrix} -0.5 & 0 & 0.5 \\ -1 & 0 & 1 \\ -0.5 & 0 & 0.5 \end{bmatrix}$$

$$\mathbf{G_y} = \begin{bmatrix} -0.5 & -1 & -0.5 \\ 0 & 0 & 0 \\ 0.5 & 1 & 0.5 \end{bmatrix}$$

$$\Big\downarrow$$

$$\mathbf{G_x} = \begin{bmatrix} -0.5167 & -0.0440 & 0.5046 \\ -0.9993 & -0.0105 & 0.9925 \\ -0.4649 & 0.0057 & 0.5129 \end{bmatrix}$$

$$\mathbf{G_y} = \begin{bmatrix} -0.5327 & -1.0151 & -0.4598 \\ -0.0318 & -0.0048 & 0.0012 \\ 0.4977 & 0.9974 & 0.5197 \end{bmatrix}$$

$$\mathrm{E}_1 = max(F) \cdot R_1,$$

$$\mathrm{E}_2 = E_1 \cdot R_2.$$

(F: the gradient image, $R_1 = 0.2$, $R_2 = 0.1$)

$$\Big\downarrow$$

$R_1 = 0.0832$, $R_2 = 0.0489$

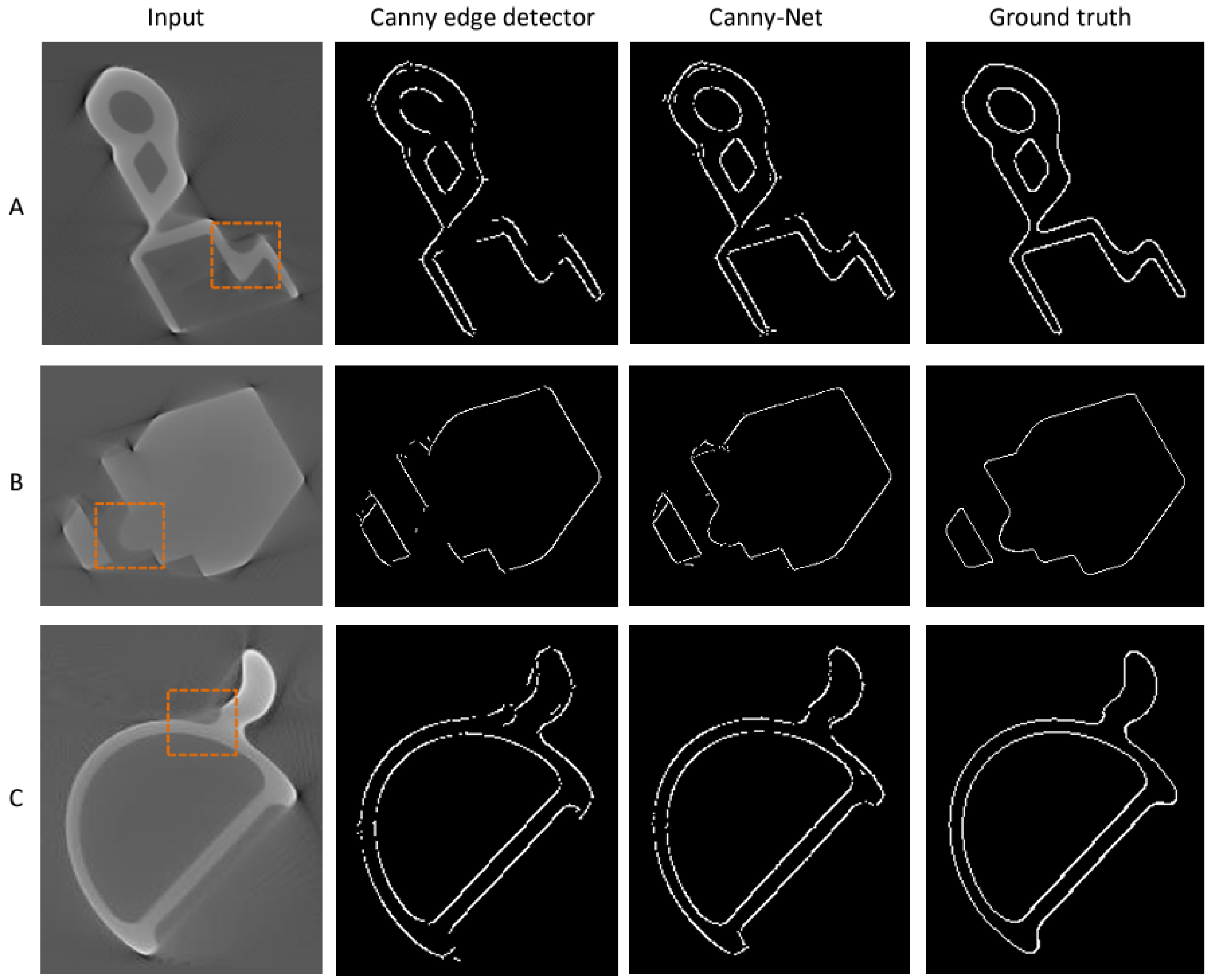


Figure 4: Comparison of the proposed methods on three mono-material metal objects A,B and C. Edge detection is performed on the input image using Canny edge detector and Canny-Net. When compared to the ground truth edges, both approaches yield similar results on sharp edges. In addition, Canny-Net performs better on edges affected by metal-induced artifacts, as illustrated by the areas located inside the orange boxes.

Holistically-Nested Edge Detection

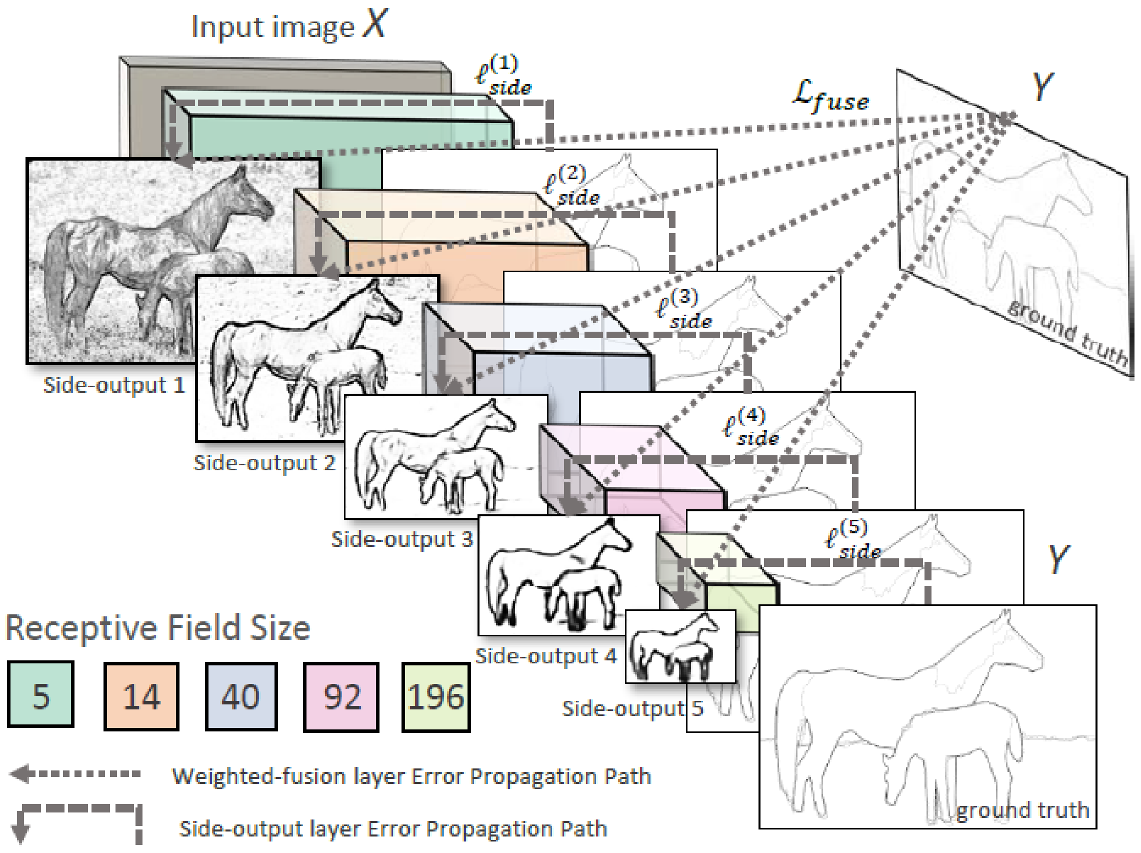
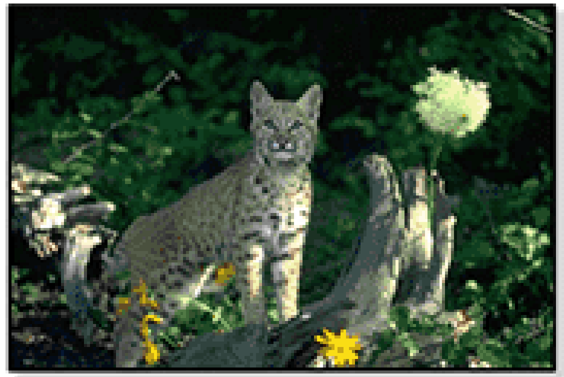
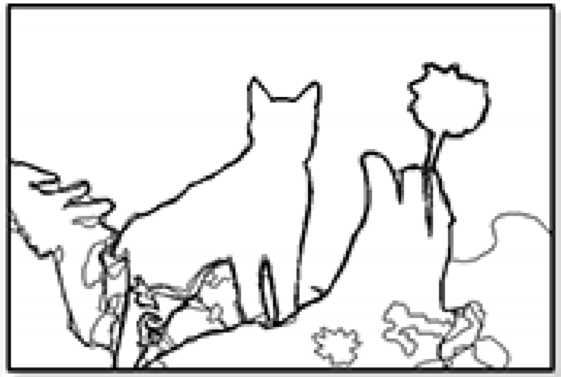


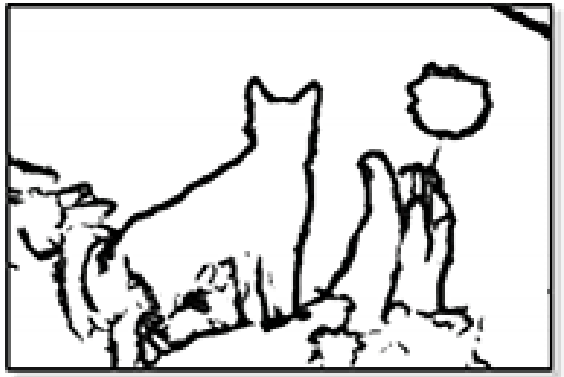
Figure 3. Illustration of our network architecture for edge detection, highlighting the error backpropagation paths. Side-output layers are inserted after convolutional layers. Deep supervision is imposed at each side-output layer, guiding the side-outputs towards edge predictions with the characteristics we desire. The outputs of HED are multi-scale and multi-level, with the side-output-plane size becoming smaller and the receptive field size becoming larger. One weighted-fusion layer is added to automatically learn how to combine outputs from multiple scales. The entire network is trained with multiple error propagation paths (dashed lines).

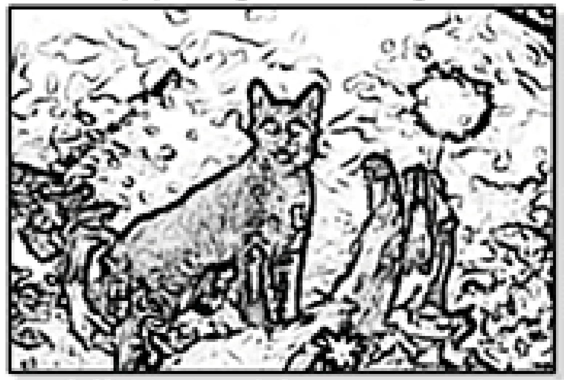
Xie, S., & Tu, Z. (2015). Holistically-nested edge detection. In *Proceedings of the IEEE international conference on computer vision* (pp. 1395-1403).

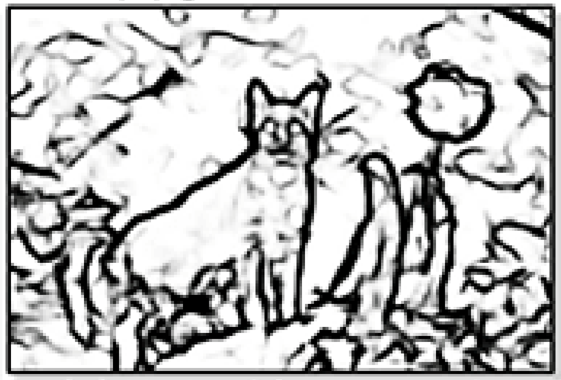
https://github.com/s9xie/hed

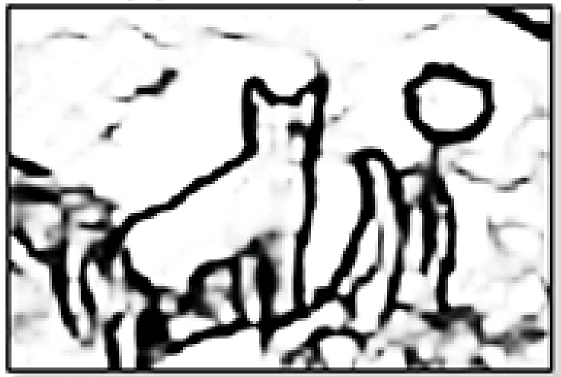

(a) original image

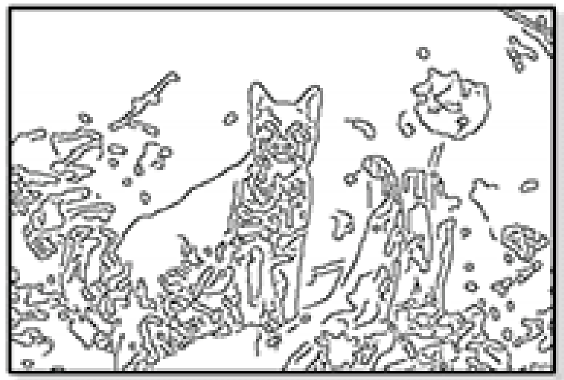

(b) ground truth

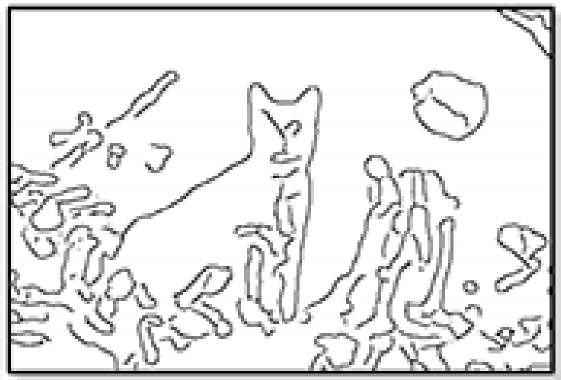

(c) HED: output

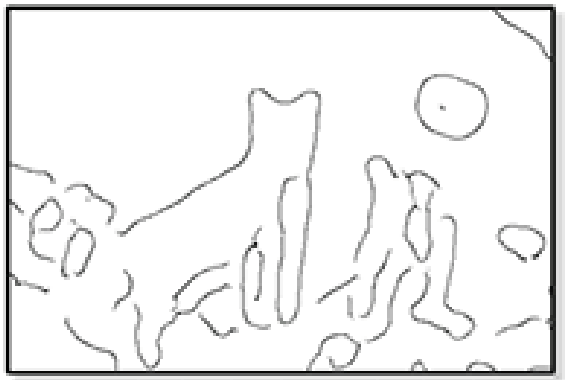

(d) HED: side output 2


(e) HED: side output 3


(f) HED: side output 4

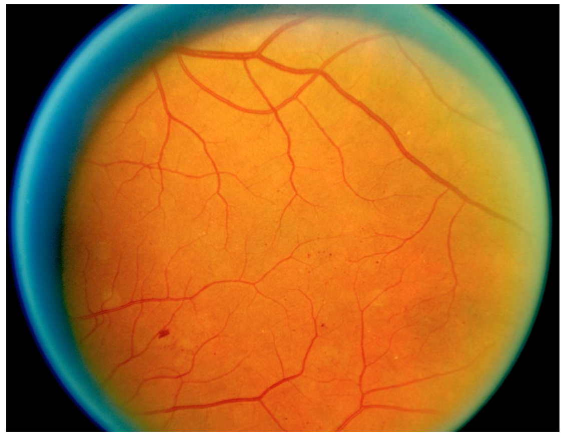

(g) Canny: $\sigma = 2$

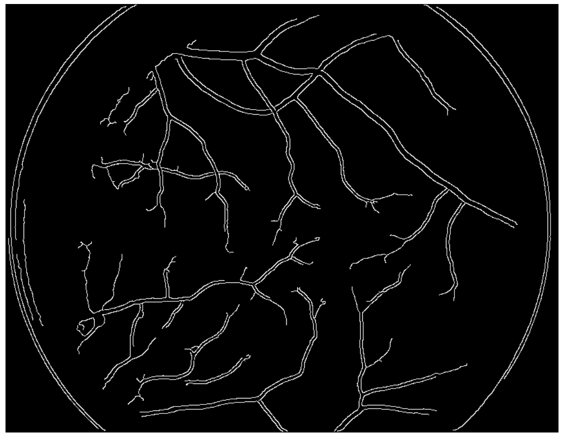

(h) Canny: $\sigma = 4$


(i) Canny: $\sigma = 8$

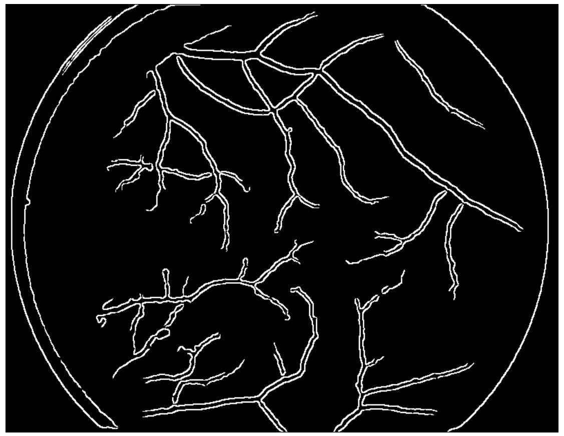
Do you see anything Interesting from Canny results under different Variances?

Project 2 (Due March 2, 2024)

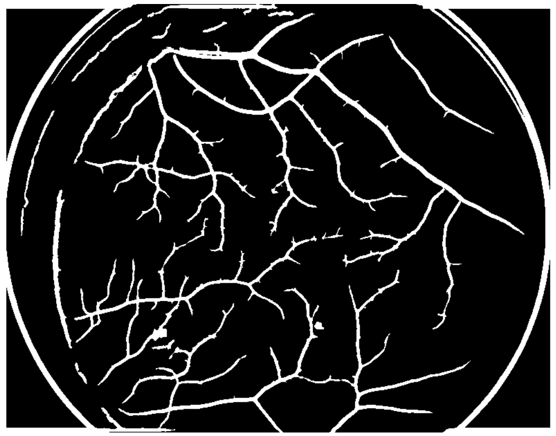




Canny detection



LoG detection



Matched filter

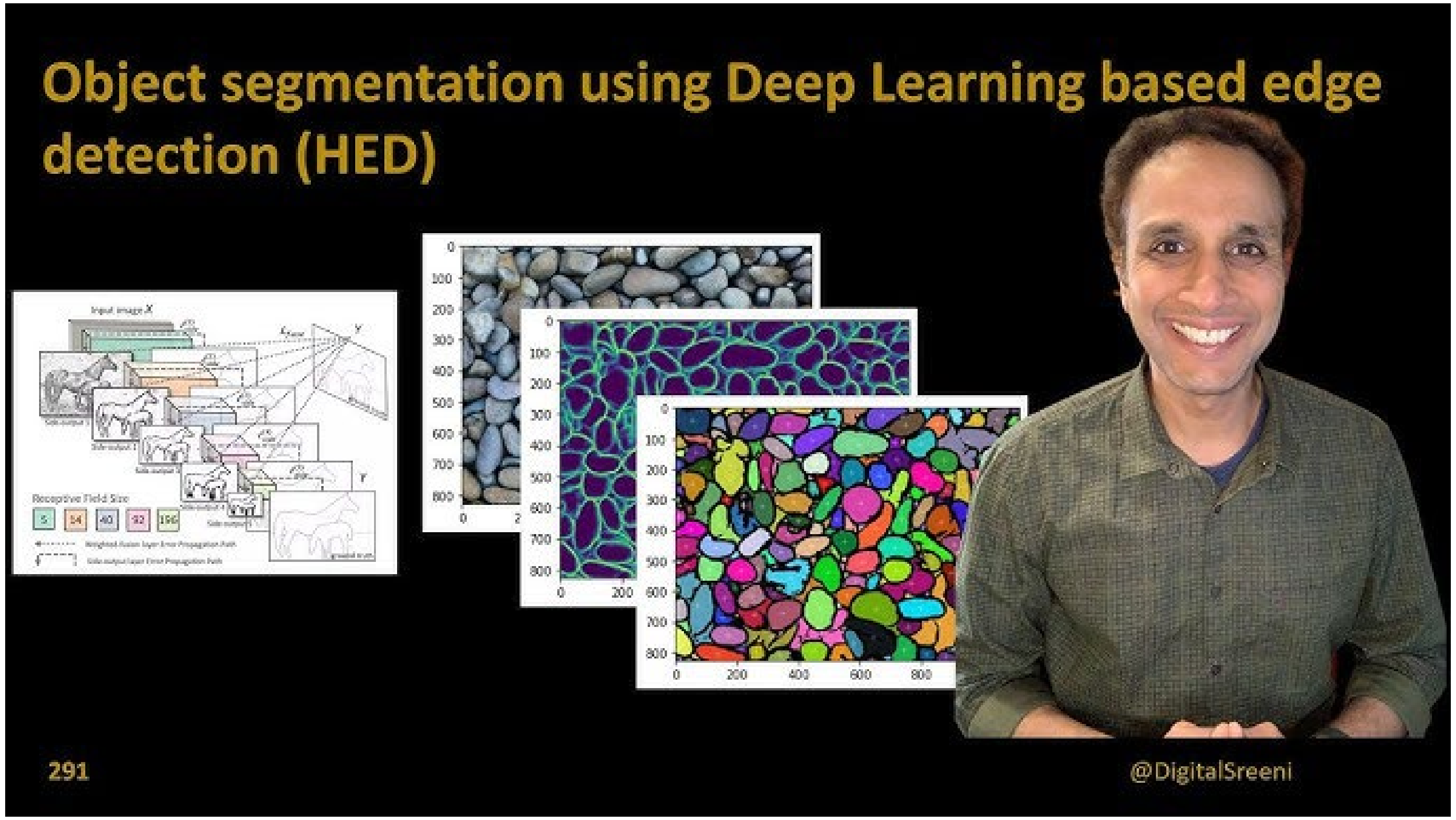
Edge Detection in Matlab

BW = edge(I,method,threshold)

BW = edge(I,method,threshold,sigma)

Method	Description
'Sobel'	Finds edges at those points where the gradient of the image I is maximum, using the Sobel approximation to the derivative.
'Prewitt'	Finds edges at those points where the gradient of I is maximum, using the Prewitt approximation to the derivative.
'Roberts'	Finds edges at those points where the gradient of I is maximum, using the Roberts approximation to the derivative.
'log'	Finds edges by looking for zero-crossings after filtering I with a Laplacian of Gaussian (LoG) filter.
'zerocross'	Finds edges by looking for zero-crossings after filtering I with a filter that you specify.
'Canny'	Finds edges by looking for local maxima of the gradient of I. The edge function calculates the gradient using the derivative of a Gaussian filter. This method uses two thresholds to detect strong and weak edges, including weak edges in the output if they are connected to strong edges. By using two thresholds, the Canny method is less likely than the other methods to be fooled by noise, and more likely to detect true weak edges.
'approxcanny'	Finds edges using an approximate version of the Canny edge detection algorithm that provides faster execution time at the expense of less precise detection. Floating point images are expected to be normalized to the range [0, 1].

Bonus Problem: HED for Edge Detection


Object segmentation using Deep Learning based edge detection (HED)
291
@DigitalSreeni

https://www.youtube.com/watch?app=desktop&v=UIrvEG9Oj1s