COMP 478 DD

Piyush Pokharkar

40120654

Assignment 4

# Theoretical Questions

# 1.a)

Repeatedly dilating an image, the ultimate results is that the objects in the image will grow and eventually fill the entire image space. The dilation operation enlarges the boundaries of objects and each iteration expands the objects until filling the image. In the limit, dilation after that will have no effect, because objects have already reached the boundary of the image.

Repeatedly eroding an image, the ultimate result is that the object in the image will shrink and eventually disappear. The erosion operation erodes away the boundaries of objects and each iteration contracts the object until it vanishes. In the limit, further erosion will not have any noticeable effect since the object has already been eroded and disappeared.

# 1.b

For both dilation and erosion, the smallest image form which you can start for the limiting effect, to hold is an image containing a single pixel. Starting with an image that has 1 pixel, repeated dilation will eventually cause that pixel to fill the entire image space and repeated erosion will cause that pixel to disappear, leaving an empty space. This is because dilation enlarges and erosion shrinks objects, so with only 1 pixel, the operation will propagate until the entire image is filled or emptied.

# 2)

Cartesian coordinate system represents lines as y=mx+b where m is the slope. When there are vertical lines, the slope of a line become undefined because the change in x is 0, resulting in division by 0 when calculating the slope. This makes it difficult to represent vertical lines using cartesian representation, and this is a problem because the goal of Hough Transform is to detect lines in an image.

The Hough Transform is defined in polar coordinates to combat the infinite slope issue. It is done by dividing the parameter plane into bins where the rows of the array represent values and columns represent values. Compute all the possible curves in the parameters that pass through all the pixels. For all the curves in the parameter space, increment the corresponding accumulator array. Analyze the accumulator array to detect geometric shapes in lines. Convert the detected peaks in the accumulator array into detected lines. Apply thresholding to filter out weak lines.

# 3)

Use Hough Transform for line detection. Apply thresholding to filter out lines that are not long enough, so only the bigger squares are detected. Iterate through all of them and see how many of them form squares. Count the number of squares.

# Programming Questions

# 1.a)

A close-up of a black and white photo of a black and white photo of a black and white photo of a black and white photo of a black and white photo of a black and white

Description automatically generated

# 1.b)

A close-up of a tool

Description automatically generated

The result, when smoothed has much less noise, almost no noise as compared to the non-smoothed result. However, after smoothing the image, the object looses, some detail and shape, as the weaker pixels aren’t shown. In the original image result, the object is clearer and more prominent.

Main.m

pkg load image

image = imread('tools\_noisy.png');

smoothedImage = imfilter(image, fspecial('average', [5, 5]));

[result] = otsu(image);

[resultSmooth] = otsu(smoothedImage);

figure;

subplot(2, 2, 1);

imshow(image);

title('Original Image');

subplot(2, 2, 2);

imshow(result);

title('Otsu method');

subplot(2, 2, 3);

imshow(smoothedImage);

title('Smoothed Image');

subplot(2, 2, 4);

imshow(resultSmooth);

title('Otsu method smoothed');

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*otsu.m\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

function [result] = otsu(image)

histogram = imhist(image);

totalPixels = numel(image);

maxBetweenClassVariance = 0;

optimalThreshold = 0;

% Iterate through possible thresholds

for t = 1:256

% Class probabilities

w0 = sum(histogram(1:t)) / totalPixels;

w1 = sum(histogram(t+1:end)) / totalPixels;

% Class mean intensities

u0 = sum((0:(t-1))' .\* histogram(1:t)) / (w0 \* totalPixels);

u1 = sum((t:255)' .\* histogram(t+1:end)) / (w1 \* totalPixels);

% Between-class variance

betweenClassVariance = w0 \* w1 \* (u0 - u1)^2;

% Check if the variance is greater than the current maximum

if betweenClassVariance > maxBetweenClassVariance

maxBetweenClassVariance = betweenClassVariance;

optimalThreshold = t - 1; % MATLAB indexing starts from 1

endif

endfor

% Apply the optimal threshold

result = image > optimalThreshold;

# 2.a) Haar wavelet

A close-up of a person's face

Description automatically generated

# 2.b) Daubechies-4

A grey square with text

Description automatically generated

# 2.c)

The result at level 3 using Haar wavelet is closer to the original image than using Daubechies-4 wavelet. The approximation using Haar is very accurate and has more details of the object. Haar has a little bit of the background information, but Daubechies-4 doesn’t have any. Visually Haar wavelet has a much clear result and the level 3 on Haar is more visible, while Daubechies-4 is barely noticeable.

import matplotlib.pyplot as plt

import pywt

from skimage import io

lena\_image = io.imread('lena.tif')

c = pywt.wavedec2(lena\_image, 'haar', level=3)

plt.figure()

plt.subplot(1, 3, 1)

plt.imshow(c[1][0], cmap='gray')

plt.title('Level 1')

plt.axis('off')

plt.subplot(1, 3, 2)

plt.imshow(c[2][0], cmap='gray')

plt.title('Level 2')

plt.axis('off')

plt.subplot(1, 3, 3)

plt.imshow(c[3][0], cmap='gray')

plt.title('Level 3')

plt.axis('off')

plt.show()