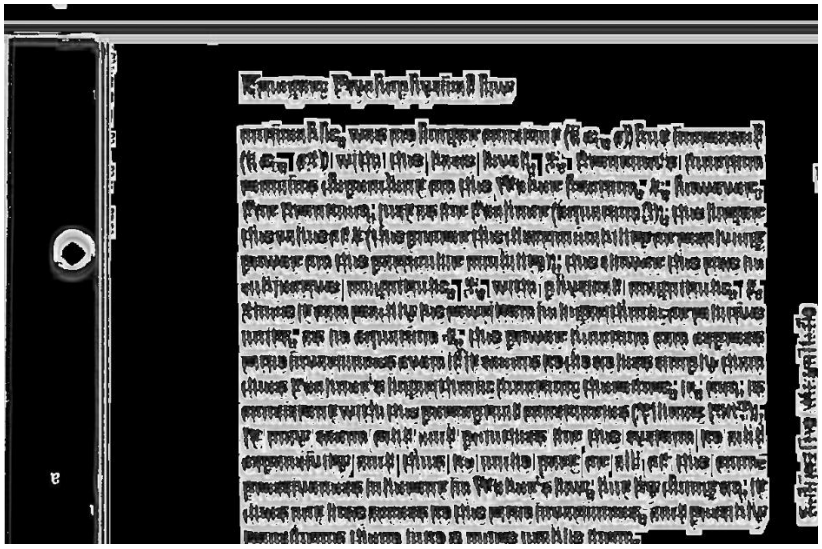
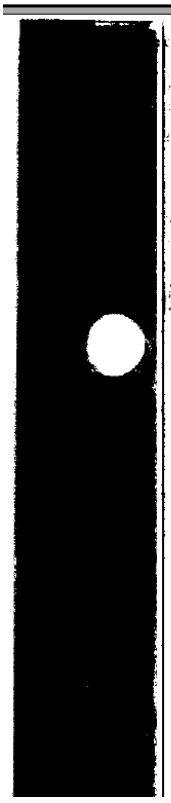
N1

Template Image





Krueger: Psychophysical law

noticeable, was no longer constant (i.e., c) but increased (i.e., cS) with the base level, S. Brentano's function remains dependent on the Weber fraction, k, however. For Brentano, just as for Fechner (equation 3), the larger the value of k (the poorer the discriminability or resolving power on the particular modality), the slower the rise in subjective magnitude, S, with physical magnitude, I. Since it can readily be rewritten in logarithmic or relative units, as in equation 4, the power function can express ratio invariances even if it seems to do so less simply than does Fechner's logarithmic function: therefore, it, too, is consistent with the perceptual constancies (Yilmaz 1967). It may seem odd and pointless for the system to add expansivity and thus to undo part or all of the compressiveness inherent in Weber's law, but by doing so, it does not lose access to the ratio invariances, and possibly transforms them into a more usable form.

Subjective Magnitude

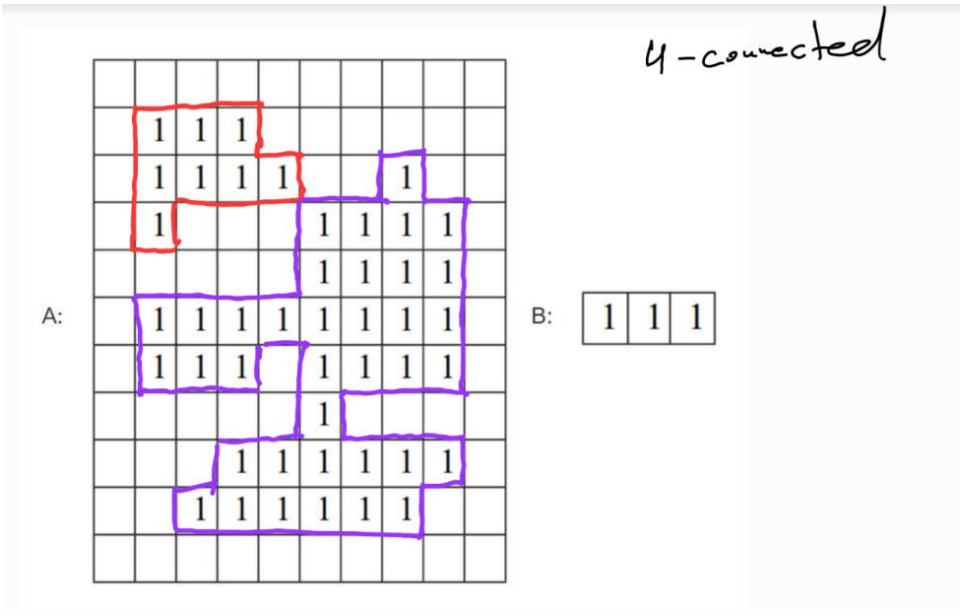
Number of detected "a"s: **53**

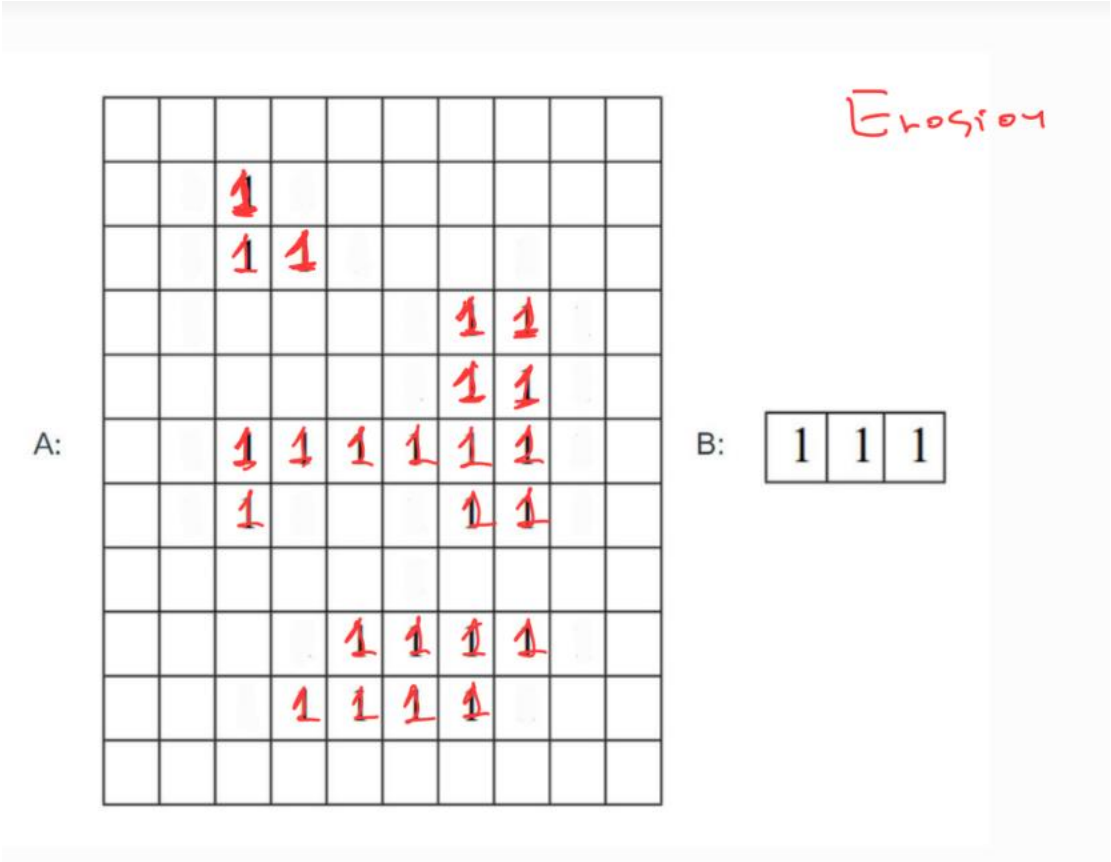
Code:

```
# -*- coding: utf-8 -*-
"""
Created on Mon Jul 22 15:24:06 2024

@author: Zakaria
"""

import cv2
import numpy as np
import matplotlib.pyplot as plt

# Load the image
image_path = "C:\\Users\\Zakaria\\Desktop\\CSCI-507HW#5\\textsample.tif"
image = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)

# Extract the template subimage of 'a' (manually selected region)
x, y, w, h = 442, 198, 16, 25  # Example coordinates for the letter 'a'
template = image[y:y+h, x:x+w]

# Show the template image
plt.imshow(template, cmap='gray')
plt.title('Template Image')
plt.axis('off')
plt.show()

# Match the template
result = cv2.matchTemplate(image, template, cv2.TM_CCOEFF_NORMED)

# Threshold peaks
threshold = 0.8  # Adjust this threshold as necessary
loc = np.where(result >= threshold)

# Create a binary image of the peaks
binary_peaks = np.zeros_like(result)
binary_peaks[loc] = 1

# Use connectedComponentsWithStats to extract the centroids of the peaks
num_labels, labels, stats, centroids =
cv2.connectedComponentsWithStats(binary_peaks.astype(np.uint8), connectivity=8)

# Draw rectangles on the original image at the detected locations
output_image = cv2.cvtColor(image, cv2.COLOR_GRAY2BGR)
count_a = num_labels - 1  # Subtract 1 because background is also labeled
```

```
for pt in zip(*loc[::-1]):
    top_left = (pt[0], pt[1])
    bottom_right = (pt[0] + w, pt[1] + h)
    cv2.rectangle(output_image, top_left, bottom_right, (0, 0, 255), 1)

# Show the template image, the scores image, and the result
plt.figure(figsize=(15, 5))
plt.subplot(1, 3, 1)
plt.title('Template Image')
plt.imshow(template, cmap='gray')
plt.axis('off')

plt.subplot(1, 3, 2)
plt.title('Scores Image')
plt.imshow(result, cmap='gray')
plt.colorbar()
plt.axis('off')

plt.subplot(1, 3, 3)
plt.title('Detected "a"s in Image')
plt.imshow(output_image)
plt.axis('off')
plt.show()

# Save the result image
output_image_path = "C:\\Users\\Zakaria\\Desktop\\CSCI-
507HW#5\\output_image_with_a_markers.png"
cv2.imwrite(output_image_path, output_image)

# Save the scores image
scores_image_path = "C:\\Users\\Zakaria\\Desktop\\CSCI-507HW#5\\scores_image.png"
cv2.imwrite(scores_image_path, (result * 255).astype(np.uint8))

# Print the number of detected 'a's
print(f'Number of detected "a"s: {count_a}')
```

N2





Dillation

A: