210451u-no-q06

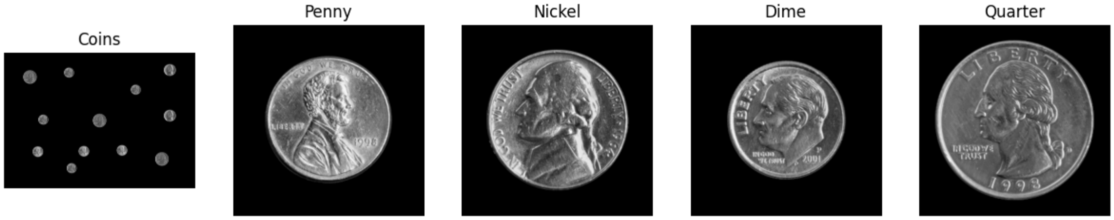
July 12, 2024

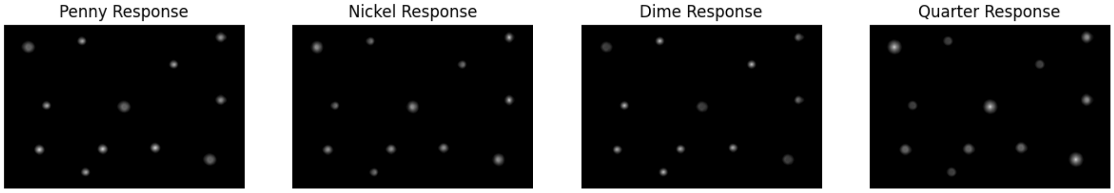
[1]:

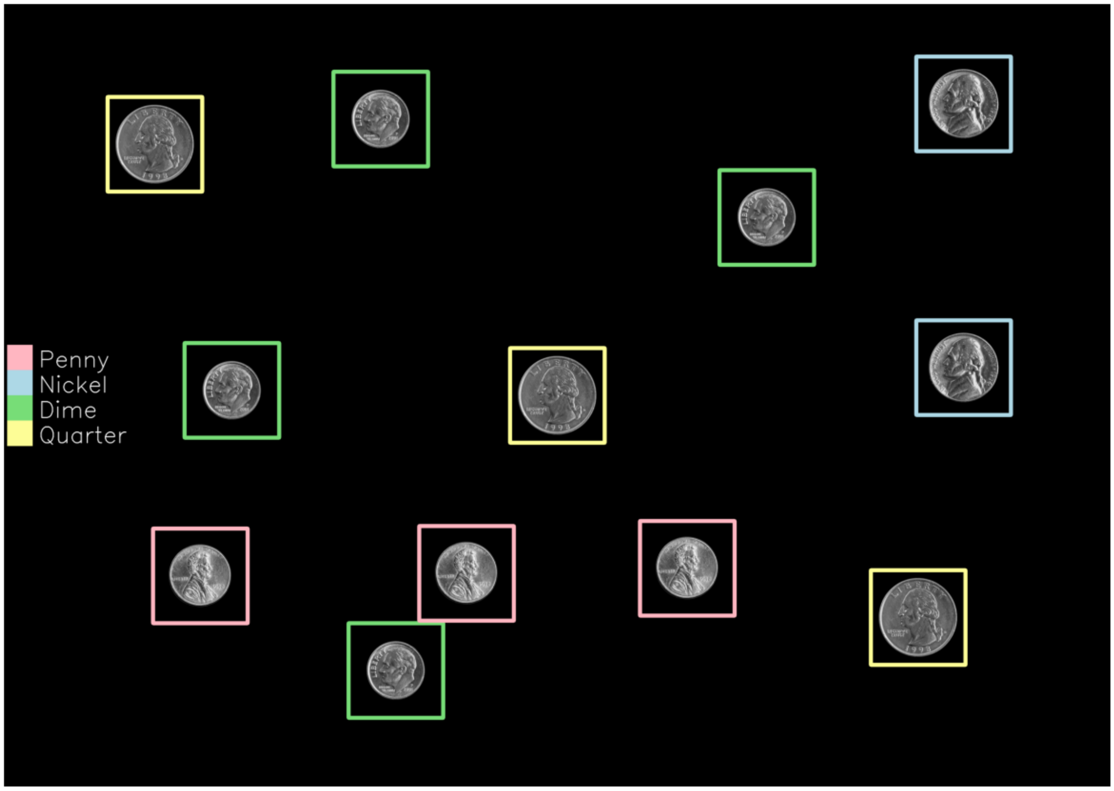
```
import numpy as np
import cv2 as cv
from skimage.feature import peak_local_max
import matplotlib.pyplot as plt

# Read the images
coins = cv.imread("Images/coins/coins.png", cv.IMREAD_GRAYSCALE)
assert coins is not None
p = cv.imread("Images/coins/Penny.png", cv.IMREAD_GRAYSCALE)
assert p is not None
n = cv.imread("Images/coins/Nickel.png", cv.IMREAD_GRAYSCALE)
assert n is not None
d = cv.imread("Images/coins/Dime.png", cv.IMREAD_GRAYSCALE)
assert d is not None
q = cv.imread("Images/coins/Quarter.png", cv.IMREAD_GRAYSCALE)

# Display the images
plt.figure(figsize=(15, 15))
plt.subplot(1, 5, 1)
plt.imshow(X=coins, cmap="gray")
plt.title("Coins")
plt.axis("off")

plt.subplot(1, 5, 2)
plt.imshow(X=p, cmap="gray")
plt.title("Penny")
plt.axis("off")

plt.subplot(1, 5, 3)
plt.imshow(X=n, cmap="gray")
plt.title("Nickel")
plt.axis("off")

plt.subplot(1, 5, 4)
plt.imshow(X=d, cmap="gray")
plt.title("Dime")
plt.axis("off")
plt.subplot(1, 5, 5)
plt.imshow(X=q, cmap="gray")
plt.title("Quarter")
plt.axis("off")

# Get the template matching responses using TM_SQDIFF_NORMED
p_response = cv.matchTemplate(coins, p, cv.TM_SQDIFF_NORMED)
n_response = cv.matchTemplate(coins, n, cv.TM_SQDIFF_NORMED)
d_response = cv.matchTemplate(coins, d, cv.TM_SQDIFF_NORMED)
q_response = cv.matchTemplate(coins, q, cv.TM_SQDIFF_NORMED)

# Invert the responses for consistency with peak detection (lower values mean␣
 ↪better match)
p_response = 1 - p_response
n_response = 1 - n_response
d_response = 1 - d_response
q_response = 1 - q_response

print(f"{p_response.min()=} {p_response.max()=}")
print(f"{n_response.min()=} {n_response.max()=}")
print(f"{d_response.min()=} {d_response.max()=}")
print(f"{q_response.min()=} {q_response.max()=}")

plt.figure(figsize=(15, 15))
plt.subplot(1, 4, 1)
plt.imshow(X=p_response, cmap="gray", vmin=0, vmax=1)
plt.title(label="Penny Response")
plt.axis("off")

plt.subplot(1, 4, 2)
plt.imshow(X=n_response, cmap="gray", vmin=0, vmax=1)
plt.title(label="Nickel Response")
plt.axis("off")

plt.subplot(1, 4, 3)
plt.imshow(X=d_response, cmap="gray", vmin=0, vmax=1)
plt.title(label="Dime Response")
plt.axis("off")

plt.subplot(1, 4, 4)
plt.imshow(X=q_response, cmap="gray", vmin=0, vmax=1)
plt.title(label="Quarter Response")
plt.axis("off")
```

```python
stacked = np.stack(arrays=[p_response, n_response, d_response, q_response], axis=2)

pastel_colors = [
    (255, 182, 193),  # Pastel Pink
    (173, 216, 230),  # Pastel Blue
    (119, 221, 119),  # Pastel Green
    (253, 253, 150)   # Pastel Yellow
]


coins_color = cv.cvtColor(coins, cv.COLOR_GRAY2BGR)


coordinates = peak_local_max(stacked, exclude_border=0, min_distance=5, threshold_abs=0.8)  # Find the local maxima in the stacked responses
print(coordinates)


for coord in coordinates:
    y, x, c = coord
    template_shape = [p.shape, n.shape, d.shape, q.shape][c]
    cv.rectangle(coins_color, (x, y), (x + template_shape[1], y + template_shape[0]), pastel_colors[c], thickness=12)

template_names = ["Penny", "Nickel", "Dime", "Quarter"]

legend_scale = 0.8
rectangle_size = 100

legend_height = int(len(template_names) * rectangle_size * legend_scale)  # Adjusted legend height based on scale


start_y = (coins_color.shape[0] - legend_height) // 2  # Calculate the starting y-coordinate to vertically center the legend

# Print a legend
for i, (name, color) in enumerate(zip(template_names, pastel_colors)):
    scaled_font_scale = legend_scale * 3  # Adjust font scale based on legend scale

    # Calculate rectangle coordinates
    rect_top_left = (10, start_y + int(rectangle_size * legend_scale * i))
    rect_bottom_right = (10 + int(rectangle_size * legend_scale), start_y + int(rectangle_size * legend_scale + rectangle_size * legend_scale * i))
    # Draw rectangle
    cv.rectangle(coins_color, rect_top_left, rect_bottom_right, color=color, thickness=-1)

    # Calculate text position
    text_x = 30 + int(rectangle_size * legend_scale)
    text_y = start_y + int(rectangle_size * legend_scale + 0.5 * rectangle_size * legend_scale * (2 * i + 1)) - int(rectangle_size / 2)

    # Draw text
    cv.putText(coins_color, name, (text_x, text_y), fontFace=cv.FONT_HERSHEY_SIMPLEX, fontScale=scaled_font_scale, color=(255, 255, 255), thickness=2, lineType=cv.LINE_AA)

plt.figure(figsize=(15, 15))
plt.imshow(X=coins_color)
plt.axis("off")
plt.show()

# calculate the total coin value in the image
coin_values = [0.01, 0.05, 0.10, 0.25]
total_value = 0
coin_count = [0, 0, 0, 0]

for coord in coordinates:
    c = coord[2]
    coin_count[c] += 1
    total_value += coin_values[c]

print(f"Total value of the coins: ${total_value:.2f}")
print(f"Coin counts: Pennies: {coin_count[0]}, Nickels: {coin_count[1]}, Dimes: {coin_count[2]}, Quarters: {coin_count[3]}")
```

```
p_response.min()=0.0 p_response.max()=1.0
n_response.min()=0.0 n_response.max()=1.0
d_response.min()=0.0 d_response.max()=1.0
q_response.min()=0.0 q_response.max()=1.0
[[ 166 2892    1]
 [ 214 1044    2]
 [1002 2892    1]
 [1074  572    2]
 [1638 2016    0]
 [1794 2748    3]
 [1962 1092    2]
 [ 294  328    3]
 [1090 1604    3]
[1654 1316    0]
[1662  472    0]
[ 526 2268    2]]
```







```
Total value of the coins: $1.28
Coin counts: Pennies: 3, Nickels: 2, Dimes: 4, Quarters: 3
```

Assumptions

Coin Templates: The provided template images for each coin type (penny, nickel, dime, quarter) are accurate representations of the coins in the main image, and they are placed on a standard 300x300 canvas.

Image Quality and Resolution: The resolution and scale of the coins in the main image are similar to those of the templates, ensuring that the template matching is effective.

Coin Appearance: The coins in the main image are free from significant occlusions, distortions, or variations in appearance that might affect the accuracy of template matching.

Threshold and Peak Detection: The chosen threshold (0.8) for peak detection and the minimum distance between detected peaks (5 pixels) are appropriately set to balance the detection of all instances of each coin type while minimizing false positives and negatives.

Template Matching Method: The use of the TM_SQDIFF_NORMED method for template matching is suitable for this task. The responses are inverted for consistency with peak detection, where lower values indicate a better match.

Pastel Colors for Visualization: The colors chosen for visualizing detected coins (pastel pink, pastel blue, pastel green, pastel yellow) are distinct and adequately represent the different coin types in the overlay on the image.