HW6: Image Processing (EE604)

Chirag Garg (210288)

October 17, 2024

Code

```
import numpy as np
import cv2
from scipy.spatial.distance import cdist
import matplotlib.pyplot as plt

def generate_saliency_map(image):
    ## Generate a saliency map using S(i) = sum(||I_i - I_j||_2) for all j
    if image is None:
        return None
    img = image.astype(float) / 255.0
    pixels = img.reshape(-1, 3)
    distances = cdist(pixels, pixels, metric='euclidean')
    saliency = np.sum(distances, axis=1)
    saliency_map = saliency.reshape(image.shape[:2])
    saliency_map = (saliency_map - np.min(saliency_map)) / (np.max(saliency_map) -
    ↪  np.min(saliency_map))
    return saliency_map

def enhance_saliency_map(saliency_map):
    ##Enhance the saliency map using histogram equalization
    if saliency_map is None:
        return None
    enhanced_map = cv2.equalizeHist((saliency_map * 255).astype(np.uint8))
    enhanced_map = enhanced_map.astype(float) / 255.0
    return enhanced_map

def generate_binary_mask(saliency_map, threshold):
    if saliency_map is None:
        return None
    return (saliency_map > threshold).astype(np.uint8)

def calculate_iou(mask1, mask2):
    if mask1 is None or mask2 is None:
        return 0
    intersection = np.logical_and(mask1, mask2)
    union = np.logical_or(mask1, mask2)
    iou = np.sum(intersection) / np.sum(union)
    return iou

# Load image and ground truth
image = cv2.imread('horse.png')
ground_truth = cv2.imread('horse_gt.png', 0)  # Read as grayscale

# Generate saliency map
saliency_map = generate_saliency_map(image)
```

```
# Enhance saliency map
enhanced_map = enhance_saliency_map(saliency_map)

# Threshold range reduced based on repeated trials to get better results
thresholds = np.linspace(0.85, 0.95, 100)
best_iou = 0
best_threshold = 0
best_mask = None

# Find the best threshold
for threshold in thresholds:
    binary_mask = generate_binary_mask(enhanced_map, threshold)
    iou = calculate_iou(binary_mask, ground_truth)

    if iou > best_iou:
        best_iou = iou
        best_threshold = threshold
        best_mask = binary_mask

print(f"Best IoU: {best_iou}")
print(f"Best Threshold: {best_threshold}")

# Visualize results
fig, axs = plt.subplots(2, 3, figsize=(15, 10))
fig.suptitle('Saliency Map Generation and Comparison', fontsize=16)

axs[0, 0].imshow(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))
axs[0, 0].set_title('Original Image')
axs[0, 0].axis('off')

axs[0, 1].imshow(saliency_map, cmap='gray')
axs[0, 1].set_title('Saliency Map')
axs[0, 1].axis('off')

axs[0, 2].imshow(enhanced_map, cmap='gray')
axs[0, 2].set_title('Enhanced Saliency Map')
axs[0, 2].axis('off')

axs[1, 0].imshow(best_mask, cmap='gray')
axs[1, 0].set_title('Best Binary Mask')
axs[1, 0].axis('off')

axs[1, 1].imshow(ground_truth, cmap='gray')
axs[1, 1].set_title('Ground Truth')
axs[1, 1].axis('off')

axs[1, 2].axis('off')  # This subplot is left empty

plt.tight_layout()
plt.show()
```

Result

Best IoU: 0.8408839779005525
Best Threshold: 0.8631313131313131

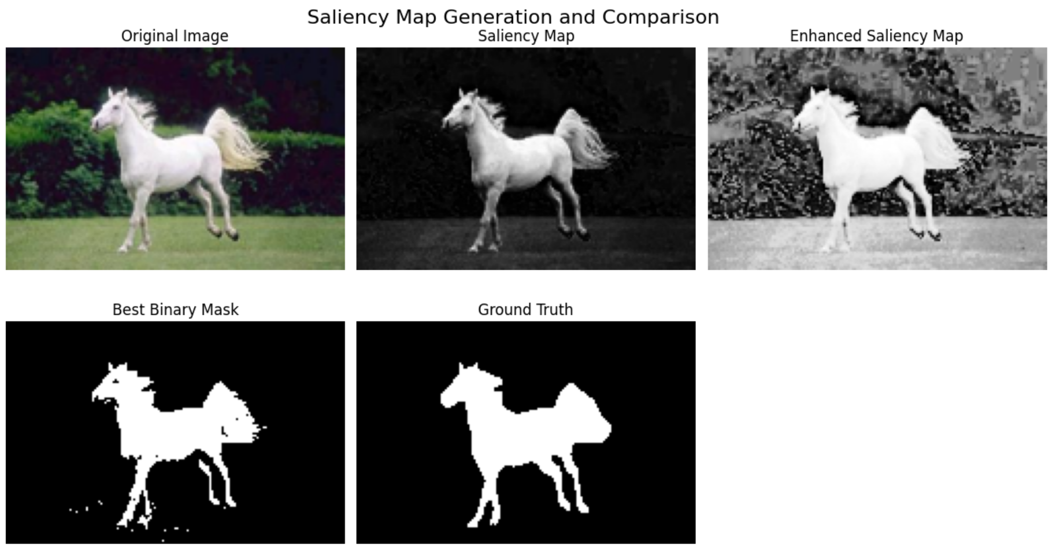

Figure 1: Output image