Name: Seshasai.PB Reg_N0: 21MIA1005

Image and Video Analytics

Object Detection and Labelling Script

Step-by-Step Breakdown

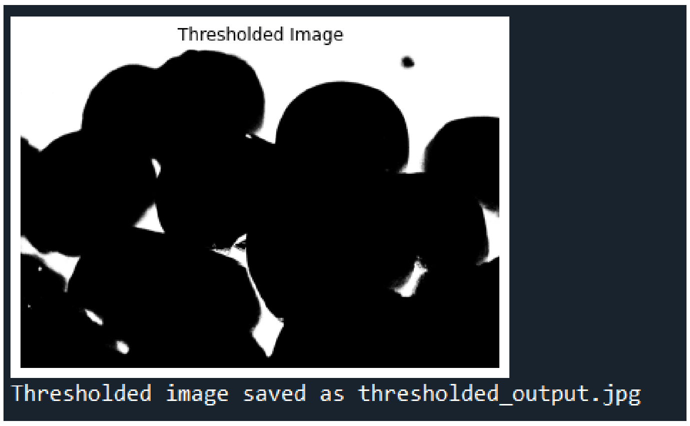
1. Image Segmentation (segment_image function)

Function:

```
def segment_image(image):
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    _, thresholded = cv2.threshold(gray, 0, 255, cv2.THRESH_BINARY_INV
+ cv2.THRESH_OTSU)
    return thresholded
```

Description:

- **Input:** A color image (BGR format).
- **Process:** Converts the image to grayscale and applies Otsu's thresholding to segment the image into foreground and background.
- **Output:** A binary (thresholded) image where the objects of interest are white, and the background is black.



```
Thresholded image saved as thresholded_output.jpg
```

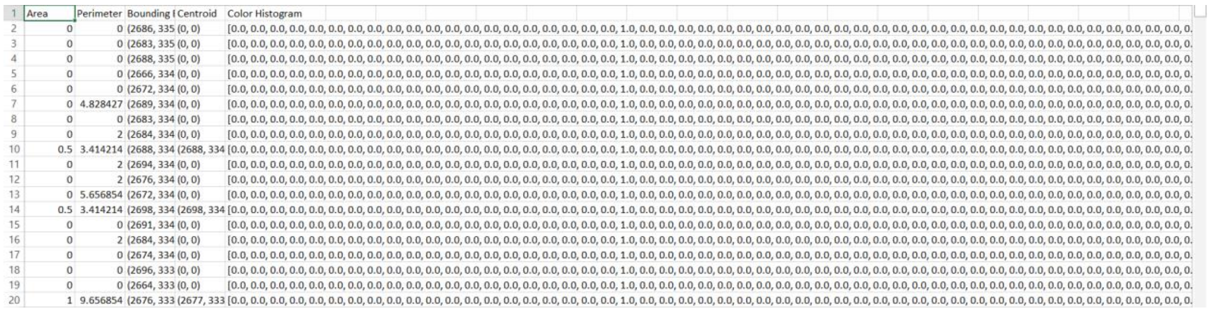
2. Feature Extraction (extract_features function)

Function:

```python
def extract_features(image, thresholded):
    contours, _ = cv2.findContours(thresholded, cv2.RETR_EXTERNAL, 
cv2.CHAIN_APPROX_SIMPLE)

    features = {
        "Area": [],
        "Perimeter": [],
        "Bounding Box": [],
        "Centroid": [],
        "Color Histogram": []
    }

    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

    for contour in contours:
        # Shape Features
        area = cv2.contourArea(contour)
        perimeter = cv2.arcLength(contour, True)
        x, y, w, h = cv2.boundingRect(contour)
        M = cv2.moments(contour)
        if M["m00"] != 0:
            cX = int(M["m10"] / M["m00"])
            cY = int(M["m01"] / M["m00"])
        else:
            cX, cY = 0, 0

        # Color Features
        mask = np.zeros(gray.shape, np.uint8)
        cv2.drawContours(mask, [contour], -1, 255, -1)
        masked_img = cv2.bitwise_and(image, image, mask=mask)
        hist = cv2.calcHist([masked_img], [0, 1, 2], mask, [8, 8, 8], 
[0, 256, 0, 256, 0, 256])
        hist = cv2.normalize(hist, hist).flatten()

        # Store features
        features["Area"].append(area)
        features["Perimeter"].append(perimeter)
        features["Bounding Box"].append((x, y, w, h))
        features["Centroid"].append((cX, cY))
        features["Color Histogram"].append(hist.tolist())  # Save as a 
list to be CSV-compatible

    return pd.DataFrame(features)
```

Description:

- **Input:** Original image and a thresholded image.
- **Process:** Detects contours in the thresholded image and extracts various features for each detected object:
    - **Shape Features:** Area, perimeter, bounding box, and centroid.
    - **Color Features:** A 3D color histogram in the RGB color space.
- **Output:** A pandas DataFrame containing all the extracted features.



1	Area	Perimeter	Bounding I	Centroid	Color Histogram
2	0	0	(2686, 335	(0, 0)	[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.
3	0	0	(2683, 335	(0, 0)	[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.
4	0	0	(2688, 335	(0, 0)	[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.
5	0	0	(2666, 334	(0, 0)	[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.
6	0	0	(2672, 334	(0, 0)	[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.
7	0	4.828427	(2689, 334	(0, 0)	[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.
8	0	0	(2683, 334	(0, 0)	[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.
9	0	2	(2684, 334	(0, 0)	[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.
10	0.5	3.414214	(2688, 334	(2688, 334	[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.
11	0	2	(2694, 334	(0, 0)	[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.
12	0	2	(2676, 334	(0, 0)	[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.
13	0	5.656854	(2672, 334	(0, 0)	[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.
14	0.5	3.414214	(2698, 334	(2698, 334	[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.
15	0	0	(2691, 334	(0, 0)	[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.
16	0	2	(2684, 334	(0, 0)	[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.
17	0	0	(2674, 334	(0, 0)	[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.
18	0	0	(2696, 333	(0, 0)	[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.
19	0	0	(2664, 333	(0, 0)	[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.
20	1	9.656854	(2676, 333	(2677, 333	[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.

3. Creating Reference Features (create_reference_features function)

Function:

```
def create_reference_features():
    reference_images = {
        "orange": "orange.jpg",
        "green_apple": "green.jpg"
    }
    reference_features = {}

    for label, img_path in reference_images.items():
        features_csv = f"{label}_features.csv"
        if os.path.exists(features_csv):
            # Load features if already saved
            print(f"Loading saved features for {label} from
{features_csv}")
            reference_features[label] = pd.read_csv(features_csv,
converters={"Color Histogram": eval})
        else:
            # Extract and save features
            print(f"Extracting and saving features for {label}")
            image = cv2.imread(img_path)
            if image is None:
                print(f"Error: Image {img_path} not found.")
                continue
            thresholded = segment_image(image)
            features_df = extract_features(image, thresholded)
            features_df.to_csv(features_csv, index=False)
            reference_features[label] = features_df

    return reference_features
```

Description:

- **Purpose:** Extracts features from reference images (e.g., oranges and green apples) and saves them for future use.
- **Input:** A dictionary mapping labels (e.g., "orange", "green_apple") to image file paths.
- **Process:**
    - For each reference image, check if the features have already been extracted and saved to a CSV file.
    - If not, segment the image, extract the features, and save them to a CSV file.
- **Output:** A dictionary where each key is a label, and the corresponding value is a DataFrame of features for that label.

Reference Images

Orange

Green apple

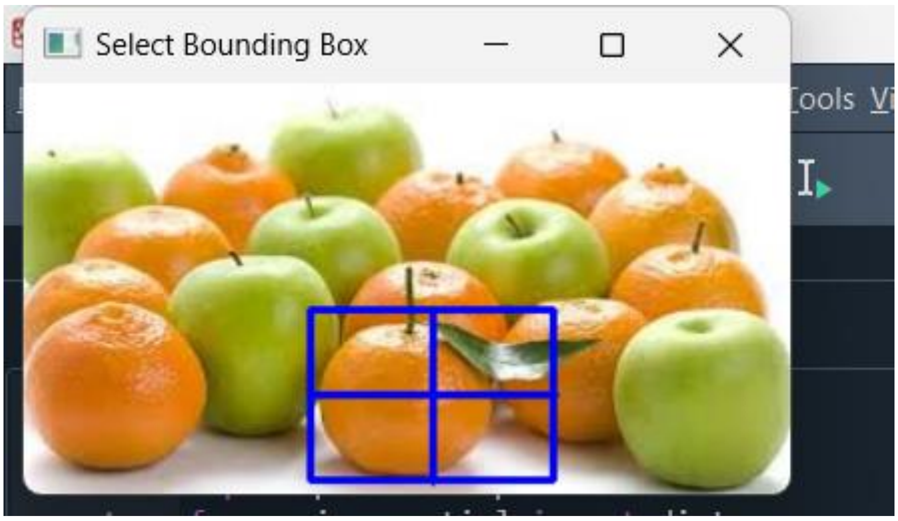
4. Extracting Features from a Bounding Box (extract_features_from_bbox function)

Function:

```
def extract_features_from_bbox(image, bbox):
    x, y, w, h = bbox
    cropped_image = image[y:y+h, x:x+w]
    mask = np.ones(cropped_image.shape[:2], dtype="uint8") * 255
    hist = cv2.calcHist([cropped_image], [0, 1, 2], mask, [8, 8, 8],
[0, 256, 0, 256, 0, 256])
    hist = cv2.normalize(hist, hist).flatten()
    return hist
```

Description:

- **Input:** Original image and a bounding box (x, y, w, h).
- **Process:** Crops the image according to the bounding box and extracts a normalized color histogram from this region.
- **Output:** The extracted color histogram as a flattened array.



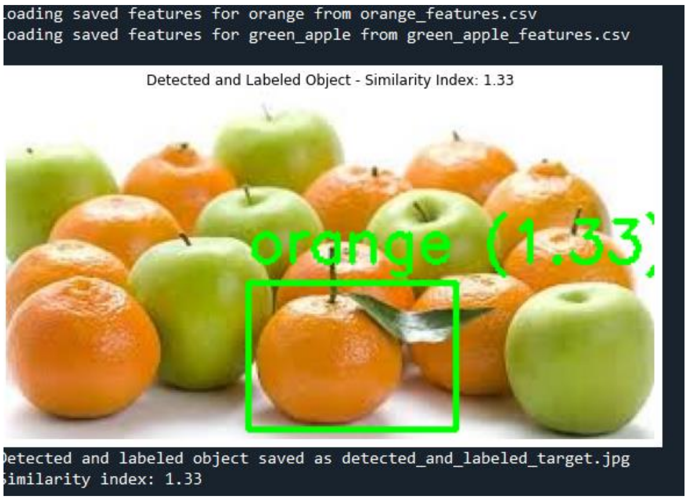
5. Comparing Features for Object Detection (compare_features function)

Function:

```
def compare_features(reference_features, target_histogram):
    best_match = None
    best_distance = float('inf')

    for label, ref_features_df in reference_features.items():
        for ref_index, ref_row in ref_features_df.iterrows():
            ref_histogram = np.array(ref_row["Color Histogram"])
            dist = distance.euclidean(ref_histogram, target_histogram)

            if dist < best_distance:
                best_distance = dist
                best_match = label

    return best_match, best_distance
```

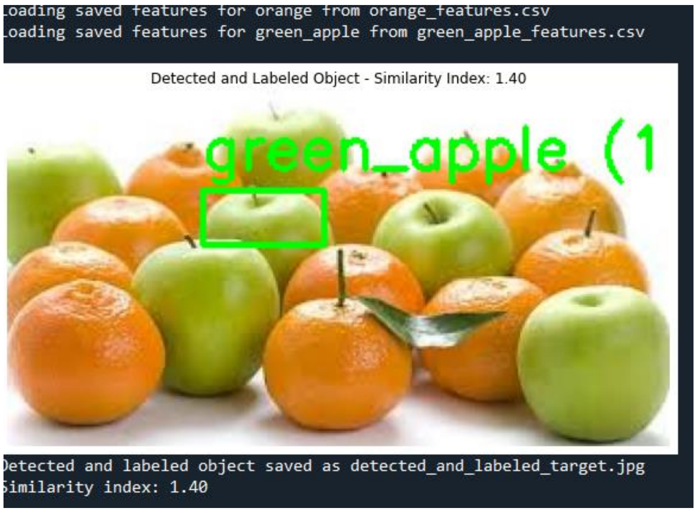
Description:

- **Input:** A dictionary of reference features and the histogram of the target object.
- **Process:** Compares the target histogram against each reference histogram using Euclidean distance.
- **Output:** The label of the best matching reference object and the corresponding similarity index (Euclidean distance).



```
.oading saved features for orange from orange_features.csv
.oading saved features for green_apple from green_apple_features.csv
```

```
)etected and labeled object saved as detected_and_labeled_target.jpg
imilarity index: 1.33
```



```
oading saved features for orange from orange_features.csv
oading saved features for green_apple from green_apple_features.csv
```

```
etected and labeled object saved as detected_and_labeled_target.jpg
imilarity index: 1.40
```

6. Detecting and Labeling Objects (detect_and_label_bbox function)

Function:

```
def detect_and_label_bbox(image, bbox, reference_features):
    target_histogram = extract_features_from_bbox(image, bbox)
    label, similarity_index = compare_features(reference_features,
target_histogram)

    x, y, w, h = bbox
    cv2.rectangle(image, (x, y), (x + w, y + h), (0, 255, 0), 2)
    label_text = f"{label} ({similarity_index:.2f})"
    cv2.putText(image, label_text, (x, y - 10),
cv2.FONT_HERSHEY_SIMPLEX, 0.9, (0, 255, 0), 2)

    return image, similarity_index
```

Description:

- **Input:** Original image, bounding box, and reference features.
- **Process:** Extracts the histogram from the bounding box, compares it with reference features, and labels the object with the best match and similarity index.
- **Output:** The labeled image and the similarity index.