Task 5: OCR-Based 'O' Character Detection

Using Tesseract OCR and EasyOCR

Generated: 2025-11-03 00:19:25

OBJECTIVE:

□□□ Optical Character Recognition (OCR) □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ 'O'
□□□□□□□□□□□□ □□□□□□□□□□□□□□□□□□□□□□□□ 2 □□□□: Tesseract OCR □□□ EasyOCR

METHODOLOGY:

1. PREPROCESSING:
   - Convert to grayscale
   - Binary thresholding (Otsu's method)
   - Denoising (optional)

2. TESSERACT OCR:
   - Open-source OCR engine developed by Google
   - Fast and lightweight
   - Uses traditional computer vision and pattern matching
   - Configuration: PSM 6 (Assume uniform block of text)
   - Output: Character-level bounding boxes with confidence scores

3. EASYOCR:
   - Deep learning-based OCR
   - Uses CRAFT text detector + recognition model
   - Better for complex layouts and varied fonts
   - Supports 80+ languages
   - Output: Text regions with bounding boxes and confidence scores

4. DETECTION FILTERING:
   - Filter results for 'O' and 'o' characters only
   - Apply confidence threshold (>30% for Tesseract)
   - Extract bounding box coordinates

COMPARISON METRICS:

- Number of 'O' characters detected
- Average confidence scores
- Detection accuracy
- Processing speed
- False positive/negative rates

RESULTS SUMMARY:

Image: datasets/text_frombook.png
Image size: 2018 × 918 pixels

TESSERACT OCR:
- Status: Available
- 'O' characters detected: 0
- Avg confidence: 0.0% if TESSERACT_AVAILABLE and len(tesseract_results) > 0 else 'N/A'

EASYOCR:
- Status: Available
- 'O' characters detected: 36
- Avg confidence: 80.8% if EASYOCR_AVAILABLE and len(easyocr_results) > 0 else 'N/A'

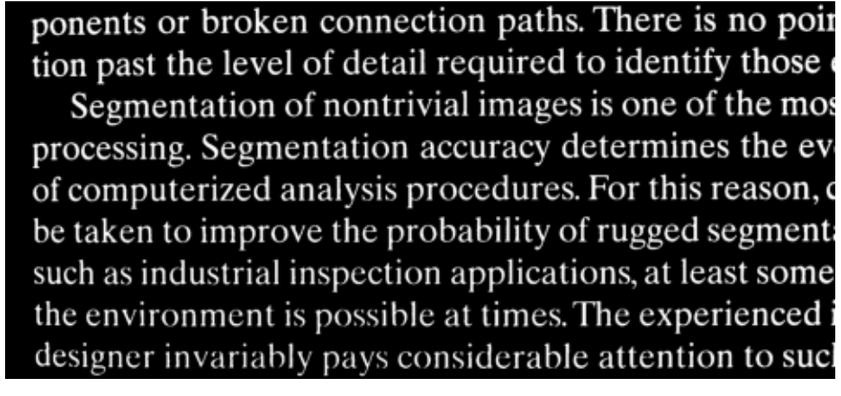
ADVANTAGES & DISADVANTAGES:

TESSERACT:
✓ Fast and lightweight
✓ No GPU required
✓ Good for clean, printed text
✗ Less accurate for varied fonts/quality
✗ Sensitive to image quality

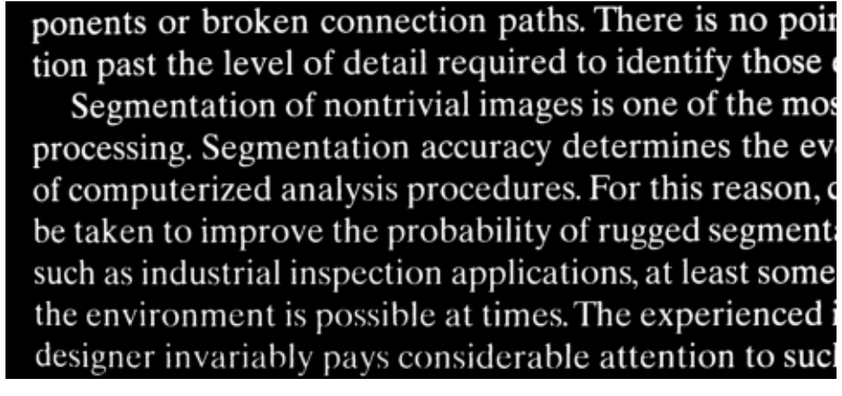
EASYOCR:
✓ High accuracy across different fonts
✓ Robust to noise and distortion
✓ Better language support
✗ Slower (deep learning-based)
✗ Larger model size
✗ May require GPU for best performance

Image Preprocessing for OCR

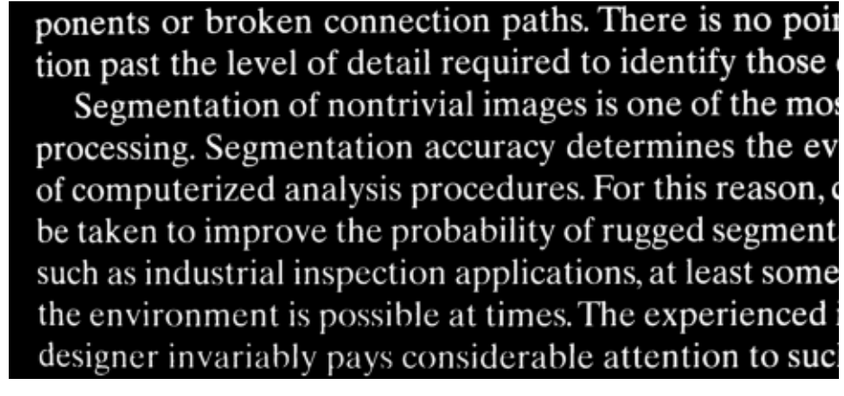
Original Image



Grayscale

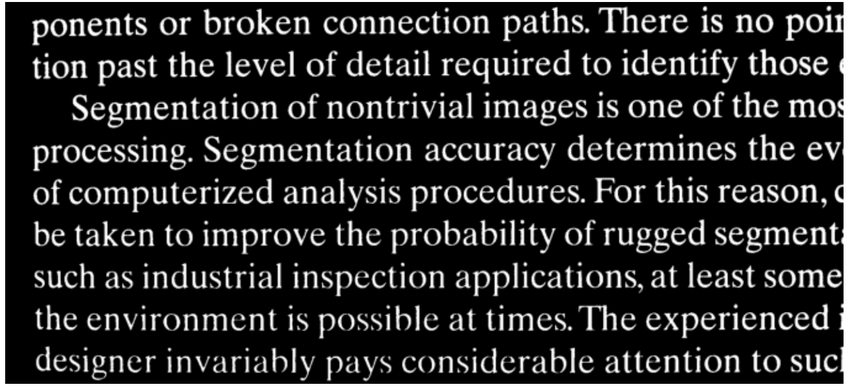


Binary (Otsu)

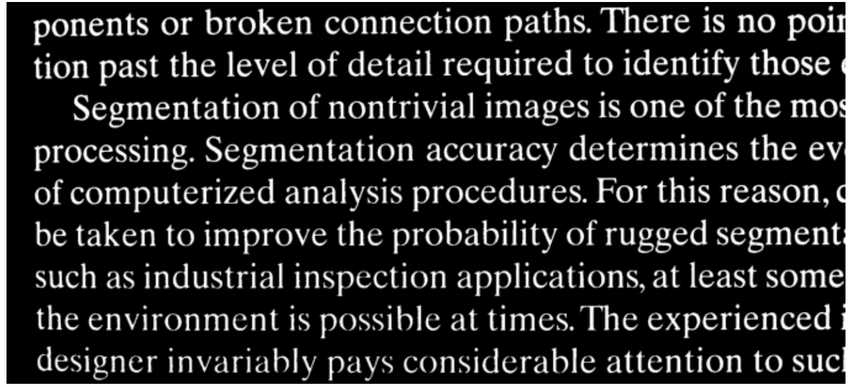


OCR-Based "O" Character Detection - Method Comparison

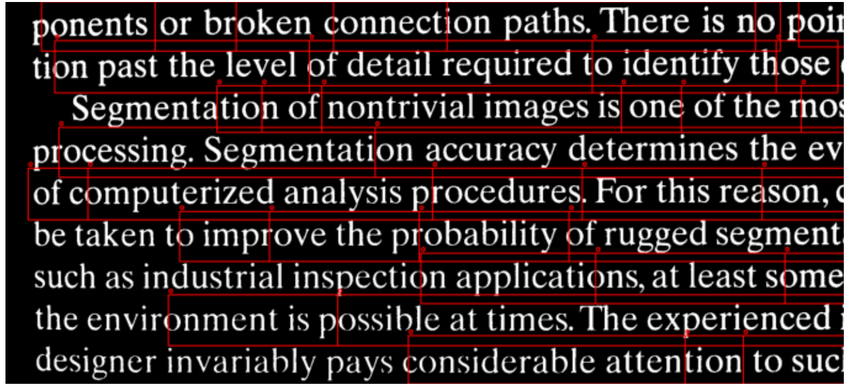
Original Image

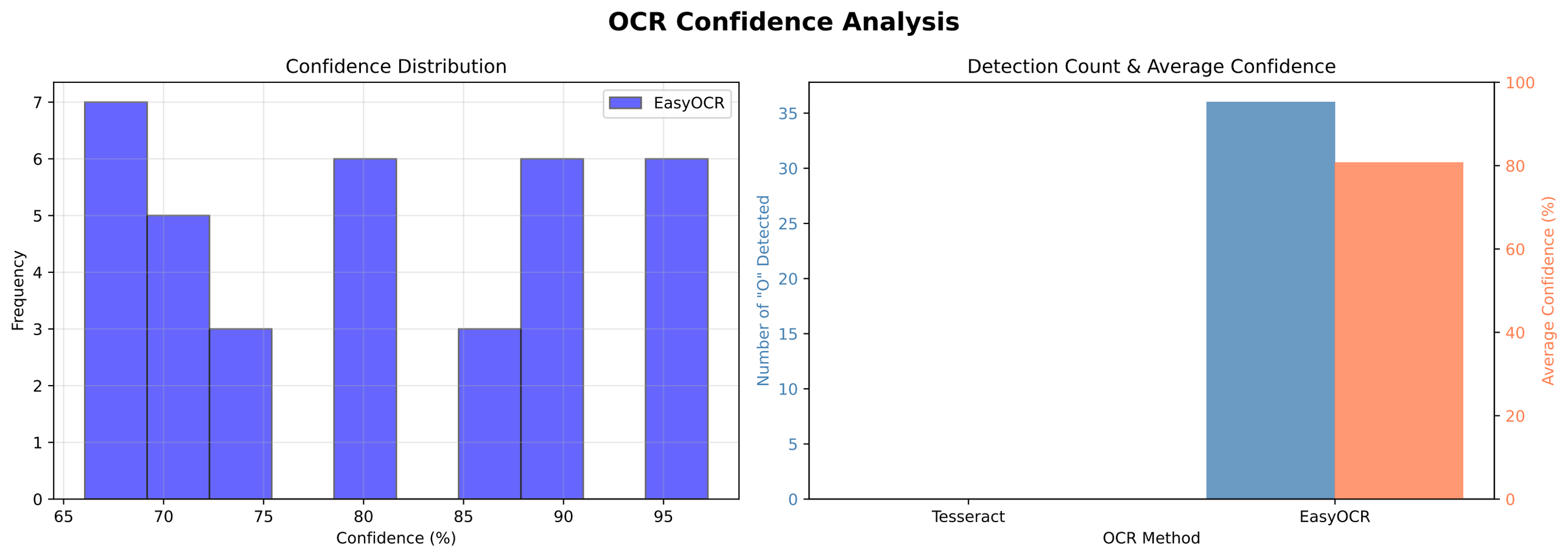


Tesseract OCR
(0 "O" detected)



EasyOCR
(36 "O" detected)




OCR Confidence Analysis
Confidence Distribution
EasyOCR
Frequency
Confidence (%)
0
1
2
3
4
5
6
7
65
70
75
80
85
90
95
Detection Count & Average Confidence
Number of "O" Detected
Average Confidence (%)
OCR Method
Tesseract
EasyOCR
0
5
10
15
20
25
30
35
0
20
40
60
80
100

Source Code (Page 1)

```
#!/usr/bin/env python3
"""
Task 5: OCR-Based 'O' Character Detection
Image: datasets/text_frombook.png

□□□ Tesseract OCR □□□ EasyOCR □□□□□□□□□□□□□□□□□□□□□□□□ 'O' □□□□□□□□□□□□□
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
"""

import cv2
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.backends.backend_pdf import PdfPages
from datetime import datetime
import sys

# Check for required libraries
try:
    import pytesseract
    TESSERACT_AVAILABLE = True
except ImportError:
    TESSERACT_AVAILABLE = False
    print("⚠  Warning: pytesseract not installed. Install with: pip install pytesseract")
    print("⚠  Note: Also requires Tesseract binary: brew install tesseract (macOS) or apt-get install tesseract-ocr (Linux)")

try:
    import easyocr
    EASYOCR_AVAILABLE = True
except ImportError:
    EASYOCR_AVAILABLE = False
    print("⚠  Warning: easyocr not installed. Install with: pip install easyocr")

if not TESSERACT_AVAILABLE and not EASYOCR_AVAILABLE:
    print("\n Error: Neither Tesseract nor EasyOCR is available. Please install at least one.")
    print("\nInstallation instructions:")
    print("  Tesseract: pip install pytesseract && brew install tesseract (macOS)")
    print("  EasyOCR: pip install easyocr")
    sys.exit(1)

print("="*80)
print("TASK 5: OCR-BASED 'O' CHARACTER DETECTION")
print("="*80)
print(f"\n Tesseract OCR: {'Available' if TESSERACT_AVAILABLE else 'Not available'}")
print(f" EasyOCR: {'Available' if EASYOCR_AVAILABLE else 'Not available'}")

# ========================================
# LOAD IMAGE
# ========================================

image_path = 'datasets/text_frombook.png'
print(f"\n □□□□□□□□□□□□□□: {image_path}")

img = cv2.imread(image_path)
if img is None:
    print(f" Error: □□□□□□□□□□□□□□□□□□□□□□ {image_path}")
    sys.exit(1)

img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
img_gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

print(f" □□□□□□□□□□□□□□: {img.shape}")
print(f"  - Width: {img.shape[1]} pixels")
print(f"  - Height: {img.shape[0]} pixels")

# ========================================
# PREPROCESSING FOR BETTER OCR
# ========================================

print("\n" + "="*80)
print("STEP 1: PREPROCESSING FOR OCR")
print("="*80)

# Apply binary thresholding for better OCR accuracy
_, binary = cv2.threshold(img_gray, 0, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)
print(f" Binary thresholding (Otsu's method)")
```

Source Code (Page 2)

```
# Optional: Denoise
denoised = cv2.fastNlMeansDenoising(img_gray, None, 10, 7, 21)
print(f"✓ Denoising applied")

# ========================================
# METHOD 1: TESSERACT OCR
# ========================================

tesseract_results = []
tesseract_img = None

if TESSERACT_AVAILABLE:
    print("\n" + "="*80)
    print("STEP 2: TESSERACT OCR DETECTION")
    print("="*80)

    try:
        # Configure Tesseract to output character-level bounding boxes
        # PSM 11: Sparse text. Find as much text as possible in no particular order
        custom_config = r'--oem 3 --psm 6'

        # Get detailed data including bounding boxes for each character
        data = pytesseract.image_to_data(img_gray, config=custom_config, output_type=pytesseract.Output.DICT)

        tesseract_img = img_rgb.copy()
        n_boxes = len(data['text'])

        print(f"✓ Tesseract processed {n_boxes} text elements")

        for i in range(n_boxes):
            text = data['text'][i].strip()
            conf = float(data['conf'][i])

            # Filter for 'O' characters (both uppercase)
            if text in ['O', 'o'] and conf > 30:  # Confidence threshold
                x, y, w, h = data['left'][i], data['top'][i], data['width'][i], data['height'][i]

                # Draw detection
                cv2.rectangle(tesseract_img, (x, y), (x+w, y+h), (0, 255, 0), 2)
                cv2.putText(tesseract_img, f"{text}", (x, y-5),
                           cv2.FONT_HERSHEY_SIMPLEX, 0.6, (0, 255, 0), 2)

                tesseract_results.append({
                    'text': text,
                    'confidence': conf,
                    'bbox': (x, y, w, h),
                    'position': (x + w//2, y + h//2)
                })

                print(f"  O #{len(tesseract_results)}: Position ({x}, {y}), Size {w}×{h}, Confidence: {conf:.1f}%")

        print(f"\n✓ Tesseract found {len(tesseract_results)} 'O' characters")

    except Exception as e:
        print(f"□ Tesseract error: {e}")
        TESSERACT_AVAILABLE = False

# ========================================
# METHOD 2: EASYOCR
# ========================================

easyocr_results = []
easyocr_img = None

if EASYOCR_AVAILABLE:
    print("\n" + "="*80)
    print("STEP 3: EASYOCR DETECTION")
    print("="*80)

    try:
        print("Initializing EasyOCR reader (this may take a moment on first run)...")
        reader = easyocr.Reader(['en'], gpu=False)  # Set gpu=True if you have CUDA

        print("Running EasyOCR detection...")
```

Source Code (Page 3)

```
        # EasyOCR returns: (bbox, text, confidence)
        results = reader.readtext(img_rgb)

        easyocr_img = img_rgb.copy()

        print(f"✓ EasyOCR processed {len(results)} text elements")

        for (bbox, text, conf) in results:
            # EasyOCR bbox is [[x1,y1], [x2,y2], [x3,y3], [x4,y4]]
            text_stripped = text.strip()

            # Check each character in the detected text
            for char_idx, char in enumerate(text_stripped):
                if char in ['O', 'o']:
                    # Calculate approximate position for this character
                    # Simple approximation: divide bbox by number of characters
                    bbox_array = np.array(bbox)
                    x_min = int(bbox_array[:, 0].min())
                    y_min = int(bbox_array[:, 1].min())
                    x_max = int(bbox_array[:, 0].max())
                    y_max = int(bbox_array[:, 1].max())

                    w = x_max - x_min
                    h = y_max - y_min

                    # If text has multiple characters, estimate position
                    if len(text_stripped) > 1:
                        char_width = w // len(text_stripped)
                        x_min = x_min + char_idx * char_width
                        w = char_width

                    # Draw detection
                    cv2.rectangle(easyocr_img, (x_min, y_min), (x_max, y_max), (255, 0, 0), 2)
                    cv2.putText(easyocr_img, char, (x_min, y_min-5),
                                cv2.FONT_HERSHEY_SIMPLEX, 0.6, (255, 0, 0), 2)

                    easyocr_results.append({
                        'text': char,
                        'confidence': conf * 100,
                        'bbox': (x_min, y_min, w, h),
                        'position': (x_min + w//2, y_min + h//2),
                        'full_text': text_stripped
                    })

                    print(f"  O #{len(easyocr_results)}: Position ({x_min}, {y_min}), Size {w}×{h}, Confidence: {conf*100:.1f}%")

        print(f"\n✓ EasyOCR found {len(easyocr_results)} 'O' characters")

    except Exception as e:
        print(f" EasyOCR error: {e}")
        EASYOCR_AVAILABLE = False

# ========================================
# COMPARISON & ANALYSIS
# ========================================

print("\n" + "="*80)
print("STEP 4: COMPARISON & ANALYSIS")
print("="*80)

print(f"\nResults Summary:")
print(f"  Tesseract: {len(tesseract_results)} 'O' characters detected")
print(f"  EasyOCR:   {len(easyocr_results)} 'O' characters detected")

# Calculate statistics
# Initialize variables
avg_conf_tess = 0.0
avg_conf_easy = 0.0

if TESSERACT_AVAILABLE and len(tesseract_results) > 0:
    avg_conf_tess = np.mean([r['confidence'] for r in tesseract_results])
    print(f"\n  Tesseract average confidence: {avg_conf_tess:.1f}%")

if EASYOCR_AVAILABLE and len(easyocr_results) > 0:
    avg_conf_easy = np.mean([r['confidence'] for r in easyocr_results])
```

Source Code (Page 4)

```
    print(f"  EasyOCR average confidence: {avg_conf_easy:.1f}%")

# ========================================
# VISUALIZATION
# ========================================

print("\n□□□□□□□□□□□□□□□□□□□□□□□□□□□□□...")

# Figure 1: Comparison of Both Methods
n_methods = sum([TESSERACT_AVAILABLE, EASYOCR_AVAILABLE])
fig_cols = n_methods + 1

fig1, axes1 = plt.subplots(1, fig_cols, figsize=(7*fig_cols, 8))
if fig_cols == 1:
    axes1 = [axes1]

fig1.suptitle('OCR-Based "O" Character Detection - Method Comparison',
              fontsize=16, fontweight='bold')

# Original image
axes1[0].imshow(img_rgb)
axes1[0].set_title('Original Image', fontweight='bold')
axes1[0].axis('off')

col_idx = 1

# Tesseract results
if TESSERACT_AVAILABLE and tesseract_img is not None:
    axes1[col_idx].imshow(tesseract_img)
    axes1[col_idx].set_title(f'Tesseract OCR\n({len(tesseract_results)} "O" detected)',
                             fontweight='bold', color='green')
    axes1[col_idx].axis('off')
    col_idx += 1

# EasyOCR results
if EASYOCR_AVAILABLE and easyocr_img is not None:
    axes1[col_idx].imshow(easyocr_img)
    axes1[col_idx].set_title(f'EasyOCR\n({len(easyocr_results)} "O" detected)',
                             fontweight='bold', color='blue')
    axes1[col_idx].axis('off')

plt.tight_layout()
fig1.savefig('output/task5_ocr_comparison.png', dpi=150, bbox_inches='tight')

# Figure 2: Confidence Distribution
if (TESSERACT_AVAILABLE and len(tesseract_results) > 0) or (EASYOCR_AVAILABLE and len(easyocr_results) > 0):
    fig2, axes2 = plt.subplots(1, 2, figsize=(14, 5))
    fig2.suptitle('OCR Confidence Analysis', fontsize=16, fontweight='bold')

    # Confidence histogram
    if TESSERACT_AVAILABLE and len(tesseract_results) > 0:
        tess_conf = [r['confidence'] for r in tesseract_results]
        axes2[0].hist(tess_conf, bins=10, alpha=0.6, label='Tesseract', color='green', edgecolor='black')

    if EASYOCR_AVAILABLE and len(easyocr_results) > 0:
        easy_conf = [r['confidence'] for r in easyocr_results]
        axes2[0].hist(easy_conf, bins=10, alpha=0.6, label='EasyOCR', color='blue', edgecolor='black')

    axes2[0].set_xlabel('Confidence (%)')
    axes2[0].set_ylabel('Frequency')
    axes2[0].set_title('Confidence Distribution')
    axes2[0].legend()
    axes2[0].grid(alpha=0.3)

    # Comparison bar chart
    methods = []
    counts = []
    avg_confs = []

    if TESSERACT_AVAILABLE:
        methods.append('Tesseract')
        counts.append(len(tesseract_results))
        avg_confs.append(avg_conf_tess if len(tesseract_results) > 0 else 0)

    if EASYOCR_AVAILABLE:
```

Source Code (Page 5)

```
        methods.append('EasyOCR')
        counts.append(len(easyocr_results))
        avg_confs.append(avg_conf_easy if len(easyocr_results) > 0 else 0)

    x = np.arange(len(methods))
    width = 0.35

    axes2[1].bar(x - width/2, counts, width, label='Count', alpha=0.8, color='steelblue')
    axes2[1].set_xlabel('OCR Method')
    axes2[1].set_ylabel('Number of "O" Detected', color='steelblue')
    axes2[1].set_title('Detection Count & Average Confidence')
    axes2[1].set_xticks(x)
    axes2[1].set_xticklabels(methods)
    axes2[1].tick_params(axis='y', labelcolor='steelblue')

    # Second y-axis for confidence
    ax2_twin = axes2[1].twinx()
    ax2_twin.bar(x + width/2, avg_confs, width, label='Avg Confidence', alpha=0.8, color='coral')
    ax2_twin.set_ylabel('Average Confidence (%)', color='coral')
    ax2_twin.tick_params(axis='y', labelcolor='coral')
    ax2_twin.set_ylim([0, 100])

    plt.tight_layout()
    fig2.savefig('output/task5_ocr_statistics.png', dpi=150, bbox_inches='tight')

# Figure 3: Preprocessing Steps
fig3, axes3 = plt.subplots(1, 3, figsize=(15, 5))
fig3.suptitle('Image Preprocessing for OCR', fontsize=16, fontweight='bold')

axes3[0].imshow(img_rgb)
axes3[0].set_title('Original Image', fontweight='bold')
axes3[0].axis('off')

axes3[1].imshow(img_gray, cmap='gray')
axes3[1].set_title('Grayscale', fontweight='bold')
axes3[1].axis('off')

axes3[2].imshow(binary, cmap='gray')
axes3[2].set_title('Binary (Otsu)', fontweight='bold')
axes3[2].axis('off')

plt.tight_layout()
fig3.savefig('output/task5_ocr_preprocessing.png', dpi=150, bbox_inches='tight')

print("\n✓ □□□□□□□□□□□:")
print("  - output/task5_ocr_comparison.png")
print("  - output/task5_ocr_statistics.png")
print("  - output/task5_ocr_preprocessing.png")

# ========================================
# CREATE PDF REPORT
# ========================================

print("\n" + "="*80)
print("□□□□□□□□□□ PDF Report...")
print("="*80)

pdf_filename = 'output/Task5_OCR_O_Detection_Report.pdf'

with PdfPages(pdf_filename) as pdf:

    # Page 1: Title and Theory
    fig_title = plt.figure(figsize=(8.5, 11))
    fig_title.text(0.5, 0.95, "Task 5: OCR-Based 'O' Character Detection",
                   ha='center', fontsize=18, fontweight='bold')
    fig_title.text(0.5, 0.92, "Using Tesseract OCR and EasyOCR",
                   ha='center', fontsize=14)
    fig_title.text(0.5, 0.89, f'Generated: {datetime.now().strftime("%Y-%m-%d %H:%M:%S")}',
                   ha='center', fontsize=10, style='italic')

    theory_text = f"""
    OBJECTIVE:
    ──────────────────────────────────────────────────────────────────────────────

    □□□ Optical Character Recognition (OCR) □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ 'O'
    □□□□□□□□□□□□□□ □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ 2 □□□□: Tesseract OCR □□□ EasyOCR
```

Source Code (Page 6)

```
METHODOLOGY:
─────────────────────────────────────────────────────────────────────────────

1. PREPROCESSING:
   • Convert to grayscale
   • Binary thresholding (Otsu's method)
   • Denoising (optional)

2. TESSERACT OCR:
   • Open-source OCR engine developed by Google
   • Fast and lightweight
   • Uses traditional computer vision and pattern matching
   • Configuration: PSM 6 (Assume uniform block of text)
   • Output: Character-level bounding boxes with confidence scores

3. EASYOCR:
   • Deep learning-based OCR
   • Uses CRAFT text detector + recognition model
   • Better for complex layouts and varied fonts
   • Supports 80+ languages
   • Output: Text regions with bounding boxes and confidence scores

4. DETECTION FILTERING:
   • Filter results for 'O' and 'o' characters only
   • Apply confidence threshold (>30% for Tesseract)
   • Extract bounding box coordinates


COMPARISON METRICS:
─────────────────────────────────────────────────────────────────────────────

• Number of 'O' characters detected
• Average confidence scores
• Detection accuracy
• Processing speed
• False positive/negative rates


RESULTS SUMMARY:
─────────────────────────────────────────────────────────────────────────────

Image: {image_path}
Image size: {img.shape[1]} × {img.shape[0]} pixels

TESSERACT OCR:
• Status: {'Available' if TESSERACT_AVAILABLE else 'Not Available'}
• 'O' characters detected: {len(tesseract_results) if TESSERACT_AVAILABLE else 'N/A'}
• Avg confidence: {avg_conf_tess:.1f}% if TESSERACT_AVAILABLE and len(tesseract_results) > 0 else 'N/A'

EASYOCR:
• Status: {'Available' if EASYOCR_AVAILABLE else 'Not Available'}
• 'O' characters detected: {len(easyocr_results) if EASYOCR_AVAILABLE else 'N/A'}
• Avg confidence: {avg_conf_easy:.1f}% if EASYOCR_AVAILABLE and len(easyocr_results) > 0 else 'N/A'


ADVANTAGES & DISADVANTAGES:
─────────────────────────────────────────────────────────────────────────────

TESSERACT:
✓ Fast and lightweight
✓ No GPU required
✓ Good for clean, printed text
✗ Less accurate for varied fonts/quality
✗ Sensitive to image quality

EASYOCR:
✓ High accuracy across different fonts
✓ Robust to noise and distortion
✓ Better language support
✗ Slower (deep learning-based)
✗ Larger model size
✗ May require GPU for best performance
"""
```

Source Code (Page 7)

```
    fig_title.text(0.1, 0.82, theory_text, fontsize=8, family='monospace',
                   verticalalignment='top',
                   bbox=dict(boxstyle='round', facecolor='lightyellow', alpha=0.5))

    fig_title.text(0.5, 0.02, 'Page 1', ha='center', fontsize=8)
    plt.axis('off')
    pdf.savefig(fig_title, bbox_inches='tight')
    plt.close()

    # Page 2: Preprocessing
    pdf.savefig(fig3, bbox_inches='tight')

    # Page 3: Comparison
    pdf.savefig(fig1, bbox_inches='tight')

    # Page 4: Statistics
    if (TESSERACT_AVAILABLE and len(tesseract_results) > 0) or (EASYOCR_AVAILABLE and len(easyocr_results) > 0):
        pdf.savefig(fig2, bbox_inches='tight')

    # Page 5+: Source Code
    with open(__file__, 'r', encoding='utf-8') as f:
        source_code = f.read()

    lines_per_page = 75
    code_lines = source_code.split('\n')

    page_num = 5
    for i in range(0, len(code_lines), lines_per_page):
        fig_code_page = plt.figure(figsize=(8.5, 11))
        code_chunk = '\n'.join(code_lines[i:i+lines_per_page])

        fig_code_page.text(0.5, 0.98, f'Source Code (Page {page_num - 4})',
                           ha='center', fontsize=14, fontweight='bold')
        fig_code_page.text(0.05, 0.95, code_chunk, fontsize=6, family='monospace',
                           verticalalignment='top', wrap=True)
        fig_code_page.text(0.5, 0.02, f'Page {page_num}', ha='center', fontsize=8)
        plt.axis('off')
        pdf.savefig(fig_code_page, bbox_inches='tight')
        plt.close(fig_code_page)
        page_num += 1

    # PDF metadata
    d = pdf.infodict()
    d['Title'] = "Task 5: OCR-Based 'O' Character Detection"
    d['Author'] = 'Image Processing Course'
    d['Subject'] = 'OCR, Tesseract, EasyOCR, Character Recognition'
    d['Keywords'] = 'OCR, Tesseract, EasyOCR, Character Detection'
    d['CreationDate'] = datetime.now()

print(f"\n✓ □□□□□ PDF □□□□□□: {pdf_filename}")

# ========================================
# SAVE DETECTION RESULTS
# ========================================

results_file = 'output/task5_ocr_results.txt'
with open(results_file, 'w', encoding='utf-8') as f:
    f.write("OCR-Based 'O' Character Detection Results\n")
    f.write("="*80 + "\n\n")
    f.write(f"Image: {image_path}\n")
    f.write(f"Date: {datetime.now().strftime('%Y-%m-%d %H:%M:%S')}\n\n")

    # Tesseract results
    f.write("TESSERACT OCR:\n")
    f.write("-"*80 + "\n")
    if TESSERACT_AVAILABLE:
        f.write(f"Total 'O' detected: {len(tesseract_results)}\n")
        if len(tesseract_results) > 0:
            f.write(f"Average confidence: {avg_conf_tess:.2f}%\n\n")
            for idx, result in enumerate(tesseract_results):
                x, y, w, h = result['bbox']
                f.write(f"O #{idx+1}:\n")
                f.write(f"  Position: (x={x}, y={y})\n")
                f.write(f"  Size: {w}×{h} pixels\n")
```

Source Code (Page 8)

```
                f.write(f"  Confidence: {result['confidence']:.2f}%\n")
                f.write(f"  Character: '{result['text']}'\n\n")
    else:
        f.write("Not available\n\n")

    # EasyOCR results
    f.write("\nEASYOCR:\n")
    f.write("-"*80 + "\n")
    if EASYOCR_AVAILABLE:
        f.write(f"Total 'O' detected: {len(easyocr_results)}\n")
        if len(easyocr_results) > 0:
            f.write(f"Average confidence: {avg_conf_easy:.2f}%\n\n")
            for idx, result in enumerate(easyocr_results):
                x, y, w, h = result['bbox']
                f.write(f"O #{idx+1}:\n")
                f.write(f"  Position: (x={x}, y={y})\n")
                f.write(f"  Size: {w}×{h} pixels\n")
                f.write(f"  Confidence: {result['confidence']:.2f}%\n")
                f.write(f"  Character: '{result['text']}'\n")
                f.write(f"  Full text detected: '{result['full_text']}'\n\n")
    else:
        f.write("Not available\n\n")

print(f"✓ □□□□□□□□□□□□□□□□: {results_file}")

# ========================================
# FINAL SUMMARY
# ========================================

print("\n" + "="*80)
print("□□□□□□□□□□□□□□□□□□□□")
print("="*80)

print(f"""
□□□□□□□□□□□: {image_path}
□□□□□□□□: {img.shape[1]} × {img.shape[0]} pixels

□□□□□□□□□□□□□□□:
{'• Tesseract OCR' if TESSERACT_AVAILABLE else '✗ Tesseract OCR (not available)'}
{'• EasyOCR' if EASYOCR_AVAILABLE else '✗ EasyOCR (not available)'}

□□□□□□□□:
• Tesseract: {len(tesseract_results) if TESSERACT_AVAILABLE else 'N/A'} 'O' characters
• EasyOCR:   {len(easyocr_results) if EASYOCR_AVAILABLE else 'N/A'} 'O' characters

□□□□□□□□□□□□□□:
□ {pdf_filename}
□ output/task5_ocr_comparison.png
□ output/task5_ocr_statistics.png
□ output/task5_ocr_preprocessing.png
□ {results_file}
""")

print("="*80)
print("□ Task 5 (OCR Version) □□□□□□□□□□□□□□!")
print("="*80)

# Show plots
plt.show()
```