📄 Script2Text – Lab Plan (4-Hour Session)

🔷 1. Research to Share with Team

- **Character Segmentation is Key**: Our biggest challenge is splitting handwritten text into lines, words, and characters.
- We've reviewed Casey & Lecolinet (1996):
  - Dissection (cutting based on whitespace/connected components).
  - Recognition-based (test multiple cuts).
  - Holistic (whole words, dictionary-based).
  - Hybrid (over-segment + check).
- **Next step for us**: Try *projection profiles* (pixel sums along rows/columns) to separate lines, then words.

👉 Blog update will cover: *"Research shows segmentation is the most error-prone step; we will begin with projection profiles for line segmentation in OpenCV."*

🔷 2. Project Tasks for Tomorrow

Divide work so everyone is productive:

Aaron (you): Line segmentation prototype

- Read handwritten sample (grayscale + binarised from Step 1).
- Use horizontal projection profile to detect line breaks.
- Draw bounding boxes around detected lines.
- Save outputs for GitHub/blog.

Teammate 2: Word segmentation

- From Aaron's line bounding boxes, apply vertical projection profiles to split into words.
- Save results as cropped word images.

Teammate 3: Character segmentation & template set

- Try `cv2.findContours` or connected components to split characters inside words.
- Start collecting **typed letters/digits** as templates (A–Z, 0–9). These will be used for template matching later.

🔷 3. Code Starters (to use in lab)

Line Segmentation with Projection Profiles

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

# load preprocessed image
I = cv2.imread("data/samples/handwritten_note.jpg", cv2.IMREAD_GRAYSCALE)

# threshold if not already binarised
_, binary = cv2.threshold(I, 0, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)

# invert if text is white on black
binary = 255 - binary

# horizontal projection profile
proj = np.sum(binary, axis=1)

# plot projection
plt.plot(proj)
plt.title("Horizontal Projection Profile")
plt.show()

# detect line regions
thresh = np.max(proj) * 0.1
lines = []
in_line = False
for i, val in enumerate(proj):
    if val > thresh and not in_line:
        start = i
        in_line = True
    elif val <= thresh and in_line:
        end = i
        lines.append((start, end))
        in_line = False

# draw bounding boxes on original
I_color = cv2.cvtColor(binary, cv2.COLOR_GRAY2BGR)
for (y1, y2) in lines:
    cv2.rectangle(I_color, (0, y1), (I_color.shape[1], y2), (0, 255, 0), 2)

cv2.imwrite("output/line_segmentation.jpg", I_color)
cv2.imshow("Line Segmentation", I_color)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Word Segmentation with Vertical Projections (inside one line)

```
line_img = binary[lines[0][0]:lines[0][1], :]  # crop first line
proj = np.sum(line_img, axis=0)

plt.plot(proj)
plt.title("Vertical Projection Profile (Line 1)")
plt.show()

# similar logic to split words here
```

Character Segmentation (Connected Components)

```
num_labels, labels, stats, centroids =
cv2.connectedComponentsWithStats(binary, connectivity=8)

output = cv2.cvtColor(binary, cv2.COLOR_GRAY2BGR)
for x, y, w, h, area in stats[1:]:  # skip background
    cv2.rectangle(output, (x, y), (x+w, y+h), (0,0,255), 1)

cv2.imwrite("output/char_segmentation.jpg", output)
```

🔷 4. Blog Updates to Draft

- **Research Post**: Add link to Casey & Lecolinet paper, summary of segmentation methods.
- **Progress Post (Lab 3)**:
    - Screenshots of projection profile graph.
    - Image with line bounding boxes drawn.
    - Short paragraph: *"This week we implemented line segmentation using horizontal projection profiles. This method counts the pixels row by row to detect empty gaps between lines. Next, we will extend this to words and characters."*
    - Link to GitHub notebook.

🔷 5. GitHub Plan

- Create new notebook: `notebooks/script2text_step2_segmentation.ipynb`
- Each person commits their part on their feature branch.
- Merge into `main` after testing.
- Force add segmentation output images to `output/` (like we did for preprocessing).

🔷 6. 4-Hour Lab Timeline

Hour 1: Setup → everyone pulls latest repo → create notebooks.
Hour 2: Aaron → line segmentation; others start word/char segmentation.
Hour 3: Debug + save output images.
Hour 4: Push commits → update blog with screenshots + research link.