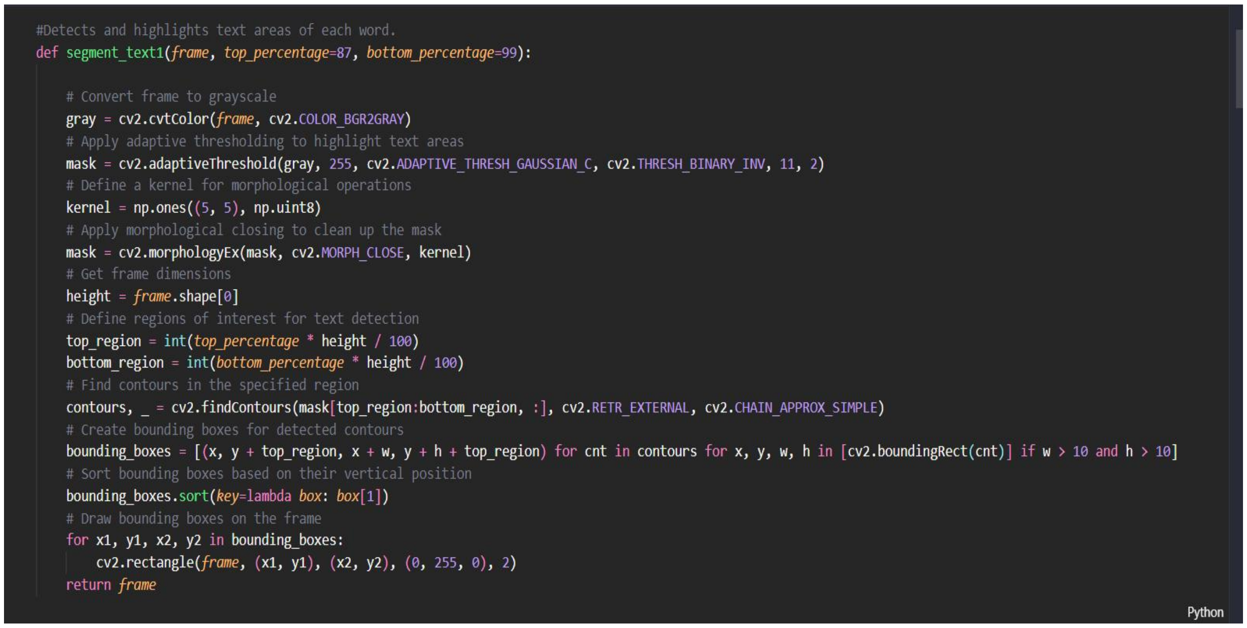
Image processing presentation

Team Members	
Name	**ID**
Alaa Amer Mohammed	20200090
Sama Hussien	20200232
Marwa shaaban	20200516
Zahraa Ahmed Abdelkafy	20201082

1. Reading and Processing , Preprocessing & Noise Removal of the Video

Methodology:

The segment_text1 function detects and highlights text areas in an image frame by converting it to grayscale and applying adaptive threshold. It uses morphological operations to clean up the mask, identifies contours in a specified region, and creates bounding boxes around detected text areas. Finally, it draws these bounding boxes on the frame and returns the annotated frame.



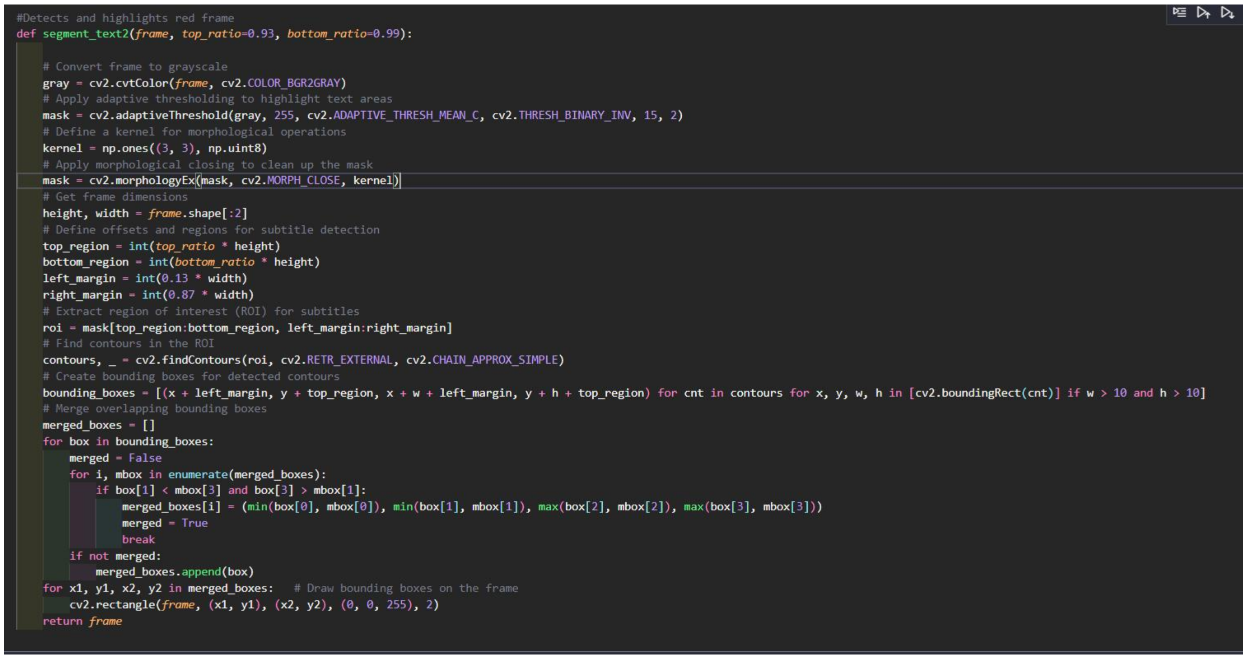
```python
#Detects and highlights text areas of each word.
def segment_text1(frame, top_percentage=87, bottom_percentage=99):

    # Convert frame to grayscale
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    # Apply adaptive thresholding to highlight text areas
    mask = cv2.adaptiveThreshold(gray, 255, cv2.ADAPTIVE_THRESH_GAUSSIAN_C, cv2.THRESH_BINARY_INV, 11, 2)
    # Define a kernel for morphological operations
    kernel = np.ones((5, 5), np.uint8)
    # Apply morphological closing to clean up the mask
    mask = cv2.morphologyEx(mask, cv2.MORPH_CLOSE, kernel)
    # Get frame dimensions
    height = frame.shape[0]
    # Define regions of interest for text detection
    top_region = int(top_percentage * height / 100)
    bottom_region = int(bottom_percentage * height / 100)
    # Find contours in the specified region
    contours, _ = cv2.findContours(mask[top_region:bottom_region, :], cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
    # Create bounding boxes for detected contours
    bounding_boxes = [(x, y + top_region, x + w, y + h + top_region) for cnt in contours for x, y, w, h in [cv2.boundingRect(cnt)] if w > 10 and h > 10]
    # Sort bounding boxes based on their vertical position
    bounding_boxes.sort(key=lambda box: box[1])
    # Draw bounding boxes on the frame
    for x1, y1, x2, y2 in bounding_boxes:
        cv2.rectangle(frame, (x1, y1), (x2, y2), (0, 255, 0), 2)
    return frame
```

Python

2.Segmentation

Methodology:

The segment_text2 function detects and highlights text areas in an image frame by converting it to grayscale and applying adaptive threshold. It extracts a region of interest for subtitle detection and finds contours within this region. Bounding boxes are created for the detected contours, with overlapping boxes merged. Finally, it draws red bounding boxes on the frame and returns the annotated frame.

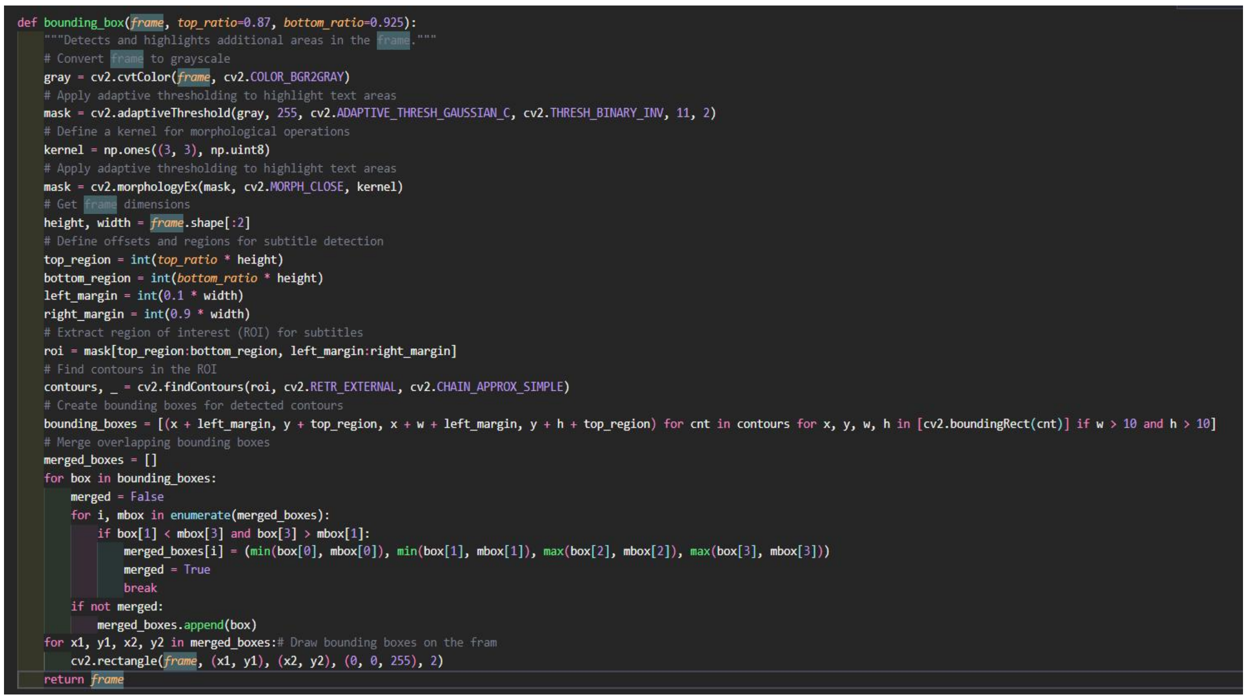


```python
#Detects and highlights red frame
def segment_text2(frame, top_ratio=0.93, bottom_ratio=0.99):

    # Convert frame to grayscale
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    # Apply adaptive thresholding to highlight text areas
    mask = cv2.adaptiveThreshold(gray, 255, cv2.ADAPTIVE_THRESH_MEAN_C, cv2.THRESH_BINARY_INV, 15, 2)
    # Define a kernel for morphological operations
    kernel = np.ones((3, 3), np.uint8)
    # Apply morphological closing to clean up the mask
    mask = cv2.morphologyEx(mask, cv2.MORPH_CLOSE, kernel)
    # Get frame dimensions
    height, width = frame.shape[:2]
    # Define offsets and regions for subtitle detection
    top_region = int(top_ratio * height)
    bottom_region = int(bottom_ratio * height)
    left_margin = int(0.13 * width)
    right_margin = int(0.87 * width)
    # Extract region of interest (ROI) for subtitles
    roi = mask[top_region:bottom_region, left_margin:right_margin]
    # Find contours in the ROI
    contours, _ = cv2.findContours(roi, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
    # Create bounding boxes for detected contours
    bounding_boxes = [(x + left_margin, y + top_region, x + w + left_margin, y + h + top_region) for cnt in contours for x, y, w, h in [cv2.boundingRect(cnt)] if w > 10 and h > 10]
    # Merge overlapping bounding boxes
    merged_boxes = []
    for box in bounding_boxes:
        merged = False
        for i, mbox in enumerate(merged_boxes):
            if box[1] < mbox[3] and box[3] > mbox[1]:
                merged_boxes[i] = (min(box[0], mbox[0]), min(box[1], mbox[1]), max(box[2], mbox[2]), max(box[3], mbox[3]))
                merged = True
                break
        if not merged:
            merged_boxes.append(box)
    for x1, y1, x2, y2 in merged_boxes:   # Draw bounding boxes on the frame
        cv2.rectangle(frame, (x1, y1), (x2, y2), (0, 0, 255), 2)
    return frame
```

3.Calculating Bounding Box Dimensions

Methodology:

The segment_text3 function detects and highlights text areas in an image frame by converting it to grayscale and applying adaptive threshold. It extracts a region of interest for subtitle detection and finds contours within this region. Bounding boxes are created for the detected contours, with overlapping boxes merged. Finally, it draws red bounding boxes on the frame and returns the annotated frame.



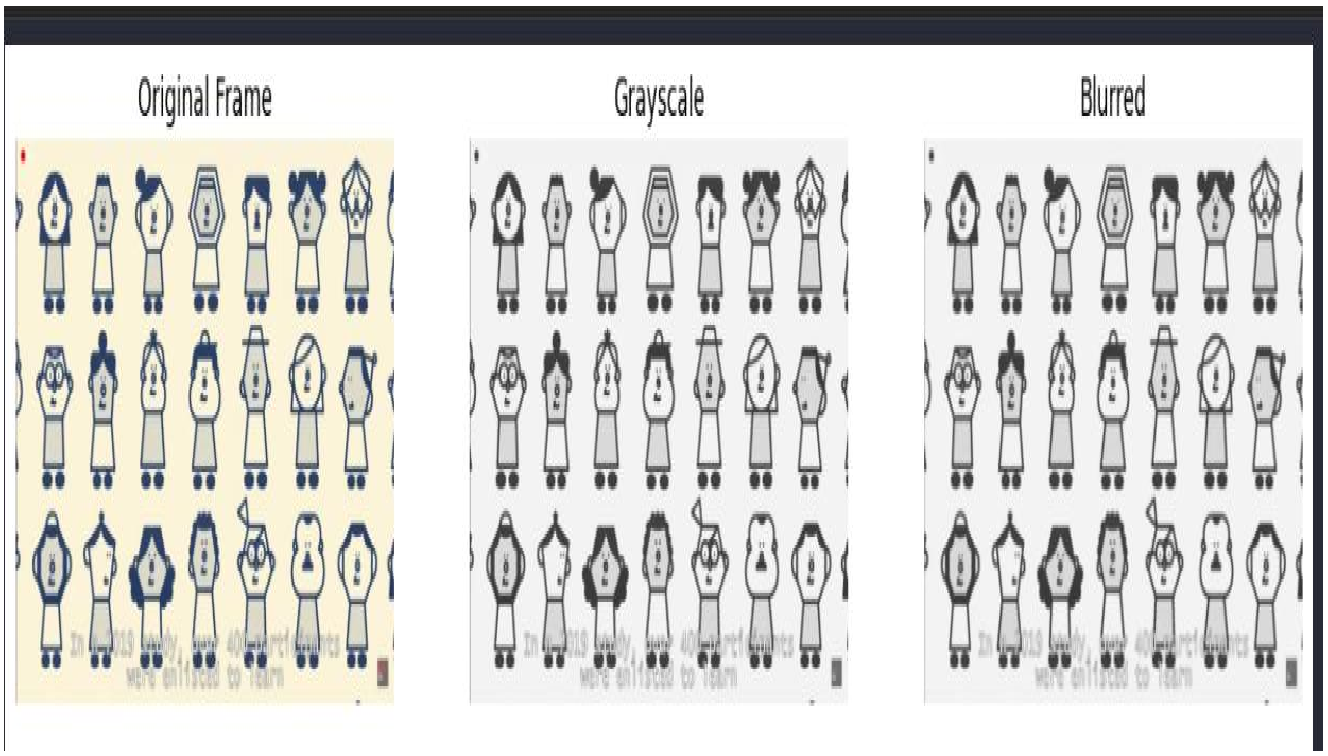
```python
def bounding_box(frame, top_ratio=0.87, bottom_ratio=0.925):
    """Detects and highlights additional areas in the frame."""
    # Convert frame to grayscale
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    # Apply adaptive thresholding to highlight text areas
    mask = cv2.adaptiveThreshold(gray, 255, cv2.ADAPTIVE_THRESH_GAUSSIAN_C, cv2.THRESH_BINARY_INV, 11, 2)
    # Define a kernel for morphological operations
    kernel = np.ones((3, 3), np.uint8)
    # Apply adaptive thresholding to highlight text areas
    mask = cv2.morphologyEx(mask, cv2.MORPH_CLOSE, kernel)
    # Get frame dimensions
    height, width = frame.shape[:2]
    # Define offsets and regions for subtitle detection
    top_region = int(top_ratio * height)
    bottom_region = int(bottom_ratio * height)
    left_margin = int(0.1 * width)
    right_margin = int(0.9 * width)
    # Extract region of interest (ROI) for subtitles
    roi = mask[top_region:bottom_region, left_margin:right_margin]
    # Find contours in the ROI
    contours, _ = cv2.findContours(roi, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
    # Create bounding boxes for detected contours
    bounding_boxes = [(x + left_margin, y + top_region, x + w + left_margin, y + h + top_region) for cnt in contours for x, y, w, h in [cv2.boundingRect(cnt)] if w > 10 and h > 10]
    # Merge overlapping bounding boxes
    merged_boxes = []
    for box in bounding_boxes:
        merged = False
        for i, mbox in enumerate(merged_boxes):
            if box[1] < mbox[3] and box[3] > mbox[1]:
                merged_boxes[i] = (min(box[0], mbox[0]), min(box[1], mbox[1]), max(box[2], mbox[2]), max(box[3], mbox[3]))
                merged = True
                break
        if not merged:
            merged_boxes.append(box)
    for x1, y1, x2, y2 in merged_boxes:# Draw bounding boxes on the fram
        cv2.rectangle(frame, (x1, y1), (x2, y2), (0, 0, 255), 2)
    return frame
```

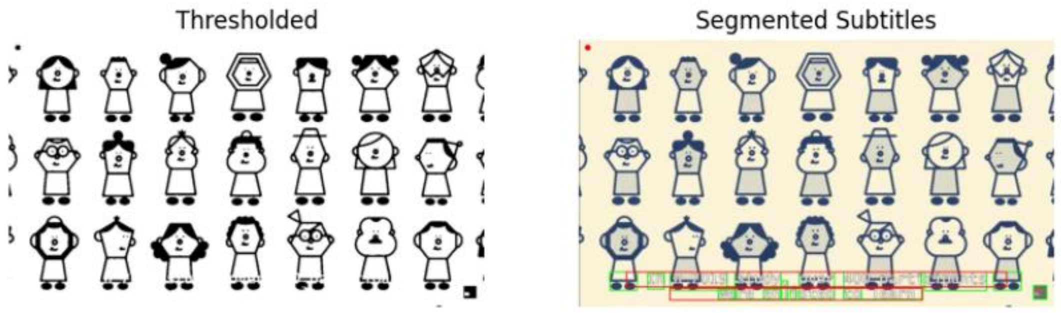
4.Screenshots of Results

Methodology:

Displayed processed frames using ‘cv2_imshow’.

Highlighted bounding boxes in green for individual objects and in red for grouped objects.


Original Frame
Grayscale
Blurred
In a 2019 study, over 400 participants
were enlisted to learn


Thresholded
Segmented Subtitles

Thank you