Aim:

To develop social distancing application

Algorithm:

Load the Image: Load the image where you want to detect social distancing.

Define Bounding Boxes: Define bounding boxes around the objects (such as people) whose social distancing you want to monitor.

Calculate Distance: Calculate the distance between the centers of each pair of bounding boxes using the Euclidean distance formula.

Draw Bounding Boxes: Draw bounding boxes around each detected object and optionally draw a combined bounding box around pairs that are too close.

Display the Results: Display the image with the bounding boxes and a label indicating whether social distancing is being maintained or not.

Program code:

```
import cv2

import numpy as np

from google.colab.patches import cv2_imshow


# Function to calculate Euclidean distance between centers of two bounding boxes

def calculate_distance(bbox1, bbox2):

    # Calculate centers of the bounding boxes

    center1 = (bbox1[0] + bbox1[2] // 2, bbox1[1] + bbox1[3] // 2)

    center2 = (bbox2[0] + bbox2[2] // 2, bbox2[1] + bbox2[3] // 2)


    # Calculate Euclidean distance

    distance = np.sqrt((center1[0] - center2[0])**2 + (center1[1] - center2[1])**2)


    return distance


# Function to draw bounding box with color based on distance

def draw_bounding_box(image, bbox, color):

    cv2.rectangle(image, (bbox[0], bbox[1]), (bbox[0] + bbox[2], bbox[1] + bbox[3]), color, 2)
```

```
# Check if image file exists
image_path = '/content/ii.jpg'

# Check if the image file exists
if not os.path.exists(image_path):
    print(f"Error: Image file '{image_path}' not found.")
else:
    # Load the image
    image = cv2.imread(image_path)

    # Check if image was successfully loaded
    if image is None:
        print(f"Error: Unable to load image '{image_path}'")
    else:
        # Define bounding boxes (x, y, width, height)
        bbox1 = (100, 50, 200, 150)
        bbox2 = (300, 200, 180, 120)

        # Draw individual bounding boxes
        draw_bounding_box(image, bbox1, (0, 255, 0))  # Green for bbox1
        draw_bounding_box(image, bbox2, (0, 255, 0))  # Green for bbox2

        # Calculate distance between bounding boxes
        distance = calculate_distance(bbox1, bbox2)

        # Determine color based on distance
        if distance < 200:  # Adjust threshold as needed
            color = (0, 255, 0)  # Green if distance is less
            print("Social Distancing")
        else:
```

```
        color = (0, 0, 255)  # Red if distance is more
        print("Not maintaining Social Distancing")

    # Draw bounding box around both objects based on distance color
    bbox_combined = (min(bbox1[0], bbox2[0]), min(bbox1[1], bbox2[1]),
                     max(bbox1[0] + bbox1[2], bbox2[0] + bbox2[2]) - min(bbox1[0], bbox2[0]),
                     max(bbox1[1] + bbox1[3], bbox2[1] + bbox2[3]) - min(bbox1[1], bbox2[1]))

    draw_bounding_box(image, bbox_combined, color)

    # Display the image with bounding boxes and distance
    cv2.putText(image, f'Distance: {distance:.2f} pixels', (50, 30), cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 255, 255), 2)

    # Show the image in the notebook
    cv2_imshow(image)
```

Output: