```
from ultralytics import YOLO
import cv2
import gradio as gr
import numpy as np


model = YOLO('chair_table_trashcan.pt')


target_classes = ['chair', 'table', 'trash can']

# Define distinct colors for each target class (BGR format for OpenCV)
class_colors = {
    'chair': (255, 0, 0),      # Blue for chair
    'table': (0, 255, 0),         # Green for table
    'trash can': (0, 0, 255)      # Red for trash can
}

# Map model class names to desired labels
label_map = {
    'chair': 'chair',
    'table': 'table',
    'trash can': 'trash can'
}

def detect_objects(image):
    """
    Detects specified objects in the uploaded image and draws bounding boxes with
distinct colors.

    Parameters:
    - image (numpy.ndarray): The input image in RGB format.

    Returns:
    - output_image (numpy.ndarray): The output image with bounding boxes and
labels.
    """
    # Convert the image from RGB (Gradio format) to BGR (OpenCV format)
    image_bgr = cv2.cvtColor(image, cv2.COLOR_RGB2BGR)

    # Run YOLOv8 inference
    results = model(image_bgr)

    # Iterate through the results
    for result in results:
        for box in result.boxes.data:
            x1, y1, x2, y2, confidence, class_id = box.cpu().numpy()
            class_id = int(class_id)

            # Get the class name from the model
            class_name = model.names[class_id]

            # Check if the detected class is in the target list
            if class_name in target_classes:
                # Select color based on class
                color = class_colors.get(class_name, (0, 255, 0))  # Default to
green if class not in the dictionary

                # Draw the bounding box
                cv2.rectangle(
                    image_bgr,
                    (int(x1), int(y1)),
                    (int(x2), int(y2)),
                    color,
                    2
                )

                # Prepare the label with mapped class name and confidence
                display_name = label_map.get(class_name, class_name)
                label = f"{display_name} {confidence:.2f}"

                # Calculate text size for background
                (text_width, text_height), _ = cv2.getTextSize(
                    label,
                    cv2.FONT_HERSHEY_SIMPLEX,
                    0.5,
                    1
                )

                # Draw a filled rectangle behind the text for better visibility
                cv2.rectangle(
                    image_bgr,
                    (int(x1), int(y1) - text_height - 10),
                    (int(x1) + text_width, int(y1)),
                    color,
                    cv2.FILLED
                )

                # Put the class name and confidence score above the bounding box
                cv2.putText(
                    image_bgr,
                    label,
                    (int(x1), int(y1) - 5),
                    cv2.FONT_HERSHEY_SIMPLEX,
                    0.5,
                    (255, 255, 255),  # White color for text
                    1
                )

    # Convert the image back from BGR to RGB for display in Gradio
    output_image = cv2.cvtColor(image_bgr, cv2.COLOR_BGR2RGB)

    return output_image
```