PROGRAM:

```
from flask import Flask, render_template, request, jsonify, redirect, url_for
import pytesseract
import cv2
import os
import time
import threading

app = Flask(_name_)

# Set up Tesseract executable path
pytesseract.pytesseract.tesseract_cmd = r'C:\Program Files\Tesseract-OCR\tesseract.exe'

# Ensure upload and output directories exist
os.makedirs('static/uploads', exist_ok=True)
os.makedirs('static/outputs', exist_ok=True)

# Global variable to store detected text from webcam
detected_text = ""
detected_text_lock = threading.Lock()
camera_running = False

@app.route('/')
def index():
    return render_template('index.html')

@app.route('/extract', methods=['POST'])
def extract_text():
    if 'file' not in request.files:
        return "No file uploaded", 400

    file = request.files['file']
```

```
    if file.filename == '':
        return "No file selected", 400

    # Save the uploaded image temporarily
    image_path = os.path.join('static', 'uploads', file.filename)
    file.save(image_path)

    # Read the image
    img = cv2.imread(image_path)

    # Perform text extraction from the image
    imgchar = pytesseract.image_to_string(img)
    print("Extracted text from image:", imgchar)  # Debugging line

    # Draw boxes around detected text
    imgboxes = pytesseract.image_to_boxes(img)
    h, w, _ = img.shape
    for box in imgboxes.splitlines():
        box = box.split(' ')
        x, y, w_box, h_box = int(box[1]), int(box[2]), int(box[3]), int(box[4])
        cv2.rectangle(img, (x, h - y), (w_box, h - h_box), (0, 255, 0), 2)

    # Save the image with bounding boxes
    output_image_path = os.path.join('static', 'outputs', 'output_' + file.filename)
    cv2.imwrite(output_image_path, img)

    return render_template('result.html', text=imgchar, output_image=output_image_path)

@app.route('/extract_video', methods=['POST'])
def extract_text_from_video():
    if 'file' not in request.files:
        return "No file uploaded", 400
```

```
file = request.files['file']
if file.filename == '':
    return "No file selected", 400

# Save the uploaded video temporarily
video_path = os.path.join('static', 'uploads', file.filename)
file.save(video_path)

cap = cv2.VideoCapture(video_path)

if not cap.isOpened():
    return "Cannot open video", 400

frames_text = []
cntr = 0
while cap.isOpened():
    ret, frame = cap.read()
    if not ret:
        break

    cntr += 1
    if cntr % 8 == 0:  # Process every 8th frame for text extraction
        imgchar = pytesseract.image_to_string(frame)
        print(f"Extracted text from frame {cntr}: {imgchar}")  # Debugging line
        frames_text.append(imgchar)

        imgboxes = pytesseract.image_to_boxes(frame)
        h, w, _ = frame.shape
        for box in imgboxes.splitlines():
            box = box.split(' ')
            x, y, w_box, h_box = int(box[1]), int(box[2]), int(box[3]), int(box[4])
```

```
            cv2.rectangle(frame, (x, h - y), (w_box, h - h_box), (0, 0, 255), 2)

    cap.release()

    if not frames_text:
        return render_template('result_video.html', text="No text detected in the video.")

    return render_template('result_video.html', text="\n".join(frames_text))

def capture_text_from_camera():
    global detected_text, camera_running
    camera_running = True
    cap = cv2.VideoCapture(0)

    if not cap.isOpened():
        camera_running = False
        return "Cannot open camera", 400

    try:
        time.sleep(1)  # Allow the camera to warm up

        while camera_running:
            ret, frame = cap.read()
            if not ret:
                break

            imgchar = pytesseract.image_to_string(frame)
            with detected_text_lock:
                detected_text += imgchar.strip() + "\n"

            imgboxes = pytesseract.image_to_boxes(frame)
            h, w, _ = frame.shape
```

```
            for box in imgboxes.splitlines():
                box = box.split(' ')
                x, y, w_box, h_box = int(box[1]), int(box[2]), int(box[3]), int(box[4])
                cv2.rectangle(frame, (x, h - y), (w_box, h - h_box), (0, 0, 255), 1)

            # Display the processed frame
            cv2.imshow('Camera', frame)

            if cv2.waitKey(1) & 0xFF == ord('q'):
                break
    finally:
        cap.release()
        cv2.destroyAllWindows()
        camera_running = False

@app.route('/start_camera', methods=['POST'])
def start_camera():
    if not camera_running:
        threading.Thread(target=capture_text_from_camera).start()
    return redirect(url_for('camera_page'))

@app.route('/camera', methods=['GET'])
def camera_page():
    return render_template('camera.html')

@app.route('/stop_camera', methods=['POST'])
def stop_camera():
    global detected_text
    with detected_text_lock:
        text_to_return = detected_text.strip()  # Strip to remove any trailing newlines
        detected_text = ""  # Reset detected text
    print("Detected text from camera:", text_to_return)  # Debugging line
```

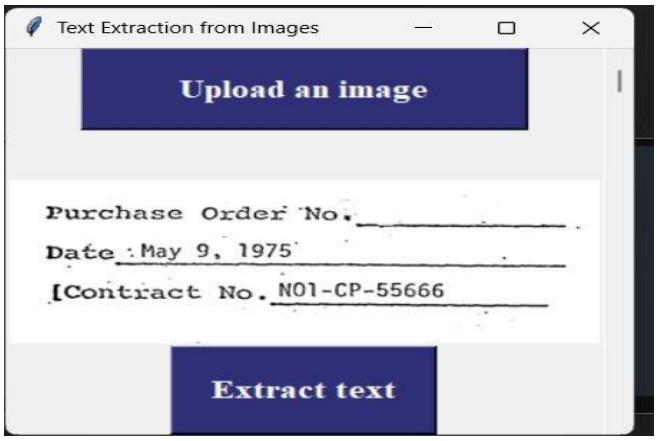
```
    return render_template('result_camera.html', text=text_to_return)


@app.route('/get_detected_text', methods=['GET'])
def get_detected_text():
    with detected_text_lock:
        text_to_return = detected_text
    return jsonify(text=text_to_return)


if _name_ == '_main_':


    app.run(debug=True)
```

OUTPUT:




```
RESULT

Purchase Order No.
Date May 9,1975
[contract No.NO1-CP-55666
```

THIS IS BATCH 11 PRESENTING THE PROJECT?"AUTOMATED TEXT RECOGNITION FROM VISUAL?CONTENT USING OPENCV" IN K.RAMAKRISHNA

THIS IS BATCH 11 PRESENTING THE PROJ
"AUTOMATED TEXT RECOGNITION FROM V
CONTENT USING OPENCV" IN K.RAMAKRIS
COLLEGE OF TECHNOLOGY (JUNE-2024)