```
import tkinter as tk
from tkinter import filedialog, messagebox
from PIL import Image, ImageTk
import requests
import cv2
import numpy as np

def ocr_space_file(filename, overlay=False, api_key='K84765758388957'):

    payload = {
        'isOverlayRequired': overlay,
        'apikey': api_key,
        'language': 'eng',
    }

    with open(filename, 'rb') as f:
        r = requests.post('https://api.ocr.space/parse/image',
                          files={filename: f},
                          data=payload)

    return r.json()

def preprocess_image(image_path):
    """ Preprocess the image to enhance OCR results. """
    img = cv2.imread(image_path)
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    gray = cv2.GaussianBlur(gray, (5, 5), 0)
    _, thresh = cv2.threshold(gray, 150, 255, cv2.THRESH_BINARY_INV)

    preprocessed_path = 'preprocessed_image.png'
    cv2.imwrite(preprocessed_path, thresh)

    return preprocessed_path

def load_image(image_path):
    """ Load and display the image in the Tkinter window. """
    img = Image.open(image_path)
    img = img.resize((250, 250), Image.LANCZOS)  # Resize image
    img = ImageTk.PhotoImage(img)

    panel = tk.Label(window, image=img)
    panel.image = img  # Keep a reference
    panel.grid(column=1, row=1)

def display_result(result):
    """ Display the OCR result in the Tkinter window. """
    if result['OCRExitCode'] == 1:
        parsed_text = result['ParsedResults'][0]['ParsedText']
        result_label.config(text=f"Detected Text: {parsed_text}")
    else:
        result_label.config(text="Error in processing the image.")

def upload_image():
    """ Function to upload and process the image. """
    uploaded_image_path = filedialog.askopenfilename(filetypes=[("Image Files",
"*.png;*.jpg;*.jpeg")])
    if uploaded_image_path:
        preprocessed_image_path = preprocess_image(uploaded_image_path)  #
Preprocess the image
```

```
        load_image(uploaded_image_path)  # Load and display the original image
        result = ocr_space_file(preprocessed_image_path)  # Call the OCR function
        display_result(result)  # Display the result

def capture_image():
    """ Function to capture image from the webcam. """
    cap = cv2.VideoCapture(0)
    if not cap.isOpened():
        messagebox.showerror("Error", "Could not open webcam.")
        return

    ret, frame = cap.read()
    if ret:
        # Save the captured frame
        cv2.imwrite('captured_image.png', frame)
        cap.release()  # Release the webcam
        load_image('captured_image.png')  # Load and display the captured image
        preprocessed_image_path = preprocess_image('captured_image.png')  #
Preprocess the captured image
        result = ocr_space_file(preprocessed_image_path)  # Call the OCR function
        display_result(result)  # Display the result
    else:
        messagebox.showerror("Error", "Failed to capture image.")
        cap.release()

# Create main window
window = tk.Tk()
window.title("OCR Application")

# Upload button
upload_btn = tk.Button(window, text="Upload Image", command=upload_image)
upload_btn.grid(column=0, row=0)

# Capture button
capture_btn = tk.Button(window, text="Capture Image", command=capture_image)
capture_btn.grid(column=0, row=1)

# Label to display result
result_label = tk.Label(window, text="")
result_label.grid(column=0, row=2)

# Start the application
window.mainloop()
```