PSEUDOCODE

```
# Import necessary libraries
import cv2
import numpy as np
import ezdxf
from shapely.geometry import Polygon
import tkinter as tk
from tkinter import filedialog, messagebox, simpledialog
from PIL import Image, ImageTk

# Function to smooth and round contours
def offset_and_smooth_contour(contour, offset_distance=5.0,
smoothing_factor=0.02, rounding=True):
    Convert contour to polygon
    Set join_style to rounded if rounding is True, otherwise mitered
    Offset the polygon with the specified distance
    if offset polygon is valid and has an exterior:
        Convert offset polygon back to contour format
        Calculate contour length and set epsilon for smoothing
        Smooth the contour using approxPolyDP
        return smoothed_contour
    else:
        return original contour

# Function to detect contours and handle inner/outer ring shapes
def detect_and_smooth_contours(image, min_contour_area=100):
    Convert image to grayscale
    Apply Gaussian blur to reduce noise
    Apply binary thresholding for segmentation
    Detect edges using Canny edge detection
    Find contours with hierarchy

    Initialize list for filtered contours
    for each contour and its hierarchy:
        if contour area > min_contour_area and contour is outer contour:
            Add contour to filtered_contours

    Apply offset_and_smooth_contour to each filtered contour
    return list of smoothed contours

# Function to compactly nest contours on a canvas
def compact_nest_contours(contours, canvas_size=(1000, 1000)):
    Initialize blank image (nested_image) for contour nesting
```

```
    Initialize list for nested contours
    Set starting x, y positions and row height

    for each contour:
        Calculate bounding rectangle of contour
        if contour width exceeds canvas width:
            Move to the next row on the canvas
        Shift contour to fit within current x, y position on canvas
        Draw shifted contour on nested_image
        Update x position and row height

    return nested_image with compactly nested contours

# Function to export contours to a DXF file with 1:1 scaling in meters
def export_to_dxf(filename, scale_factor=1.0):
    if nested contours exist:
        Create a new DXF document
        for each contour in nested contours:
            Scale contour by scale_factor
            Add contour as polyline to DXF modelspace
        Save DXF document to specified filename
    else:
        Show error message for no nested contours

# Function to save nested image as JPEG
def save_jpeg():
    if nested_image exists:
        Open file dialog to select save location
        if a filename is chosen:
            Save nested_image as JPEG to chosen location
            Show success message
    else:
        Show error message for no nested image

# Function to capture image from camera
def capture_image():
    Capture a frame from camera
    if frame is captured successfully:
        Detect and smooth contours in frame
        Add detected contours to all_contours list
        Update preview with captured frame
        Show success message with number of contours
    else:
        Show error message for capture failure
```

```
# Function to apply offset and round contours
def apply_offset_and_round():
    Prompt user for offset distance and smoothing factor
    if both values are provided:
        Apply offset_and_smooth_contour with provided parameters to all_contours
        Nest processed contours in compact layout on canvas
        Update preview with nested image
        Show success message

# Function to save nested contours as DXF
def save_dxf():
    if processed_contours exist:
        Open file dialog to select save location for DXF
        if a filename is chosen:
            Set pixel_to_meter_ratio for scaling
            Export nested contours to DXF with scaling
            Show success message
    else:
        Show error message for no processed contours

# Function to update the image preview in GUI
def update_preview(image):
    Convert image to RGB
    Convert image to a format compatible with Tkinter
    Update preview label in GUI with processed image

# Function to continuously capture frames for preview
def show_preview():
    Capture frame from camera
    if frame is captured successfully:
        Update preview with captured frame
    Schedule next preview frame update

# Main Application Window Setup
Open camera feed
Initialize global lists for contours
Create main Tkinter window

Add buttons for capture, offset, save DXF, and save JPEG functionalities
Add label for image preview

Start showing preview frames
Start Tkinter main loop
```

Release camera and destroy OpenCV windows on exit

GITHUB: https://github.com/DivyaRanjith06/DivyaRanjith06/blob/main/LAM