Supplementary 1.

Program code for quantification of atherosclerotic vascular lesion (ApoE−/−model)

Below is the complete program code used to segment the vascular wall,
calculate the area of atherosclerotic lesion, and visualize the results.
The code is written in Python (version 3.9) using the NumPy, OpenCV, and Pillow libraries.

```
----------------------------------------------------------------

import cv2
import numpy as np
from PIL import Image

# Importing libraries:
#cv2 (OpenCV) is used to work with HSV color space and masks,
# numpy — for operations with arrays of pixels,
# Pillow — for uploading images.

# ----------------------------------------------------------------
# Image loading and preprocessing
# ----------------------------------------------------------------

img = Image.open("vessel.png").convert("RGB")
img = np.array(img)

# The image of the vessel is loaded in RGB format and converted to a NumPy array,
# which allows for subsequent pixel processing.

# ----------------------------------------------------------------
# Color space conversion
# ----------------------------------------------------------------
```

```
hsv = cv2.cvtColor(img, cv2.COLOR_RGB2HSV)

# RGB → HSV translation is performed to increase the stability of color segmentation,
# because the Hue channel directly reflects the hue and is less sensitive to light.

# ----------------------------------------------------------------
# Color segmentation of affected areas
# ----------------------------------------------------------------

# Red is represented by two ranges of shades due to the cyclical nature of Hue
red1 = cv2.inRange(hsv, (0, 70, 50), (10, 255, 255))
red2 = cv2.inRange(hsv, (170, 70, 50), (180, 255, 255))
red = (red1 | red2) > 0

# Orange color
orange = cv2.inRange(hsv, (10, 70, 50), (25, 255, 255)) > 0

# Yellow color
yellow = cv2.inRange(hsv, (25, 70, 50), (35, 255, 255)) > 0

# All the listed colors refer to atherosclerotic lesions of the vascular wall.

# ----------------------------------------------------------------
# Segmentation of the intact vascular wall
# ----------------------------------------------------------------

green = cv2.inRange(hsv, (35, 70, 50), (85, 255, 255)) > 0

# The green color corresponds to an unaffected (intact) vascular wall.

# ----------------------------------------------------------------
# Formation of masks without overlap
# -----------------------------------------------------------------

lesion = red | orange | yellow

# The lesion mask includes red, orange and yellow colors.

healthy = green & (~lesion)

# An intact wall is defined as green pixels,
# from which the areas classified as affected are excluded.
# This step prevents double counting of pixels at the boundaries of the color classes.

vessel = lesion | healthy

 The overall vessel mask is formed as a union of affected and intact areas.

# -----------------------------------------------------------------
# Quantification of areas
# -----------------------------------------------------------------

total_px = int(vessel.sum())
lesion_px = int(lesion.sum())
healthy_px = int(healthy.sum())

# The area is determined by counting the number of pixels in each mask.

lesion_pct = lesion_px / total_px * 100
healthy_pct = healthy_px / total_px * 100

# Percentage values are calculated relative to the total area of the vessel.
# The sum of the percentages of the affected and intact area is 100%.
# ----------------------------------------------------------------
# Creating a clean red-green overlay
# ----------------------------------------------------------------

overlay = np.zeros_like(img)

overlay[lesion] = [255, 0, 0] # red is an atherosclerotic lesion.
overlay[healthy] = [0, 255, 0] # green color — intact wall

# The overlay does not contain orange or yellow colors and is used
# for binary visualization of lesion and norm.

# ----------------------------------------------------------------
# Overlay overlay on the original image
# ----------------------------------------------------------------

overlay_on_image = cv2.addWeighted(
    img, 0.6,
    overlay, 0.4,
    0
)

# ----------------------------------------------------------------
# Forming a three-panel image
# ----------------------------------------------------------------

h, w, _ = img.shape
panel = np.zeros((h, w * 3, 3), dtype=np.uint8)

panel[:, 0:w] = img
panel[:, w:2*w] = overlay
panel[:, 2*w:3*w] = overlay_on_image
```

The three-panel image includes:

(1) the original image of the vessel,

(2) a segmented red-green mask,

and (3) an overlay of the mask on the original.

```
# ------------------------------------------------------------------
# Saving the results
# ------------------------------------------------------------------

Image.fromarray(overlay).save("overlay_clean.png")
Image.fromarray(panel).save("three_panel.png")

# The resulting images are used to visualize the results
# and are included in the report or application.
```

End of the program code