Patch Extraction

June 22, 2025

Sat Jun 21st 2025, Jakob Balkovec

[ ]:
```
import cv2
import os
import xml.etree.ElementTree as ET
import numpy as np
import sys
import matplotlib.pyplot as plt
```

0.1 Parameters

[ ]:
```
# This is all provisional for now

image_dir = r"../data/raw/images"
xml_dir = r"../data/raw/groundtruth"
sample_image = r"diaretdb1_image001.png"
xml_file = r"diaretdb1_image001_01_plain.xml"

bigger_sample = {
  1: (r"diaretdb1_image001.png", r"diaretdb1_image001_01_plain.xml"),
  2: (r"diaretdb1_image002.png", r"diaretdb1_image002_01_plain.xml"),
  3: (r"diaretdb1_image003.png", r"diaretdb1_image003_01_plain.xml"),
  4: (r"diaretdb1_image004.png", r"diaretdb1_image004_01_plain.xml"),
  5: (r"diaretdb1_image005.png", r"diaretdb1_image005_01_plain.xml"),
  }
```

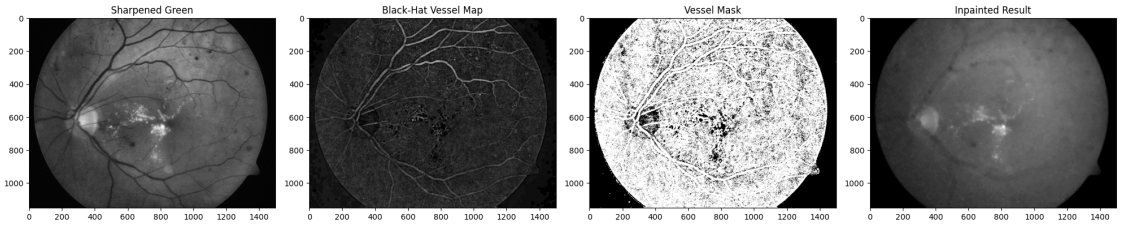
0.2 Denoise the Green Channel & Parse XML

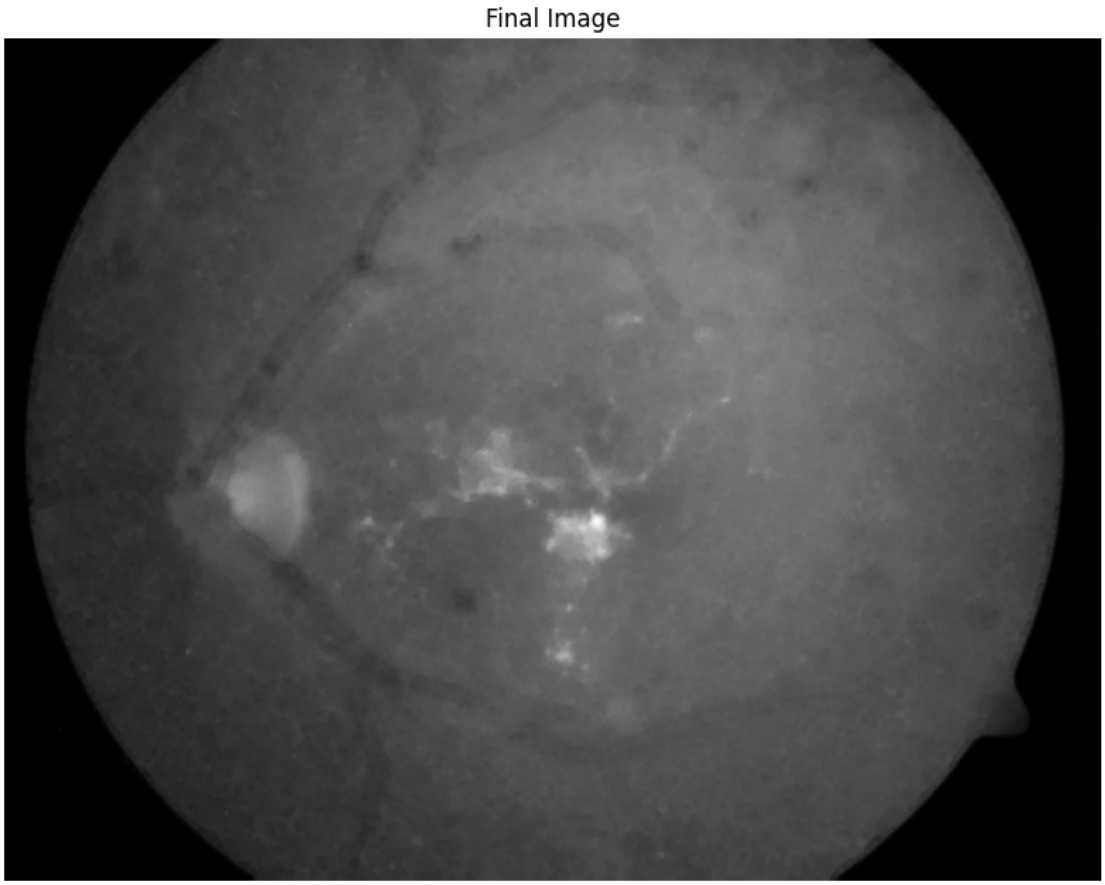
[ ]:
```
img_path = os.path.join(image_dir, sample_image)
img = cv2.imread(img_path)
img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
green_channel = img[:, :, 1]
green_blur = cv2.medianBlur(green_channel, 5)
```

0.3 Removing the Vessels (Option A)

[ ]:
```
def remove_vessels_inpaint_refined(green_img, kernel_size=15):

    clahe = cv2.createCLAHE(clipLimit=2.0, tileGridSize=(8, 8))
    enhanced = clahe.apply(green_img)

    blurred = cv2.GaussianBlur(enhanced, (0, 0), sigmaX=2)
    sharpened = cv2.addWeighted(enhanced, 1.5, blurred, -0.5, 0)

    kernel = cv2.getStructuringElement(cv2.MORPH_RECT, (kernel_size,␣
 ↪kernel_size))
    blackhat = cv2.morphologyEx(sharpened, cv2.MORPH_BLACKHAT, kernel)

    blackhat_norm = cv2.normalize(blackhat, None, 0, 255, cv2.NORM_MINMAX).
 ↪astype(np.uint8)
    _, mask_bin = cv2.threshold(blackhat_norm, 0, 255, cv2.THRESH_BINARY + cv2.
 ↪THRESH_OTSU)

    kernel_morph = cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (3, 3))
    cleaned = cv2.morphologyEx(mask_bin, cv2.MORPH_CLOSE, kernel_morph)
    dilated = cv2.dilate(cleaned, kernel_morph, iterations=1)

    inpainted = cv2.inpaint(green_img, dilated, 3, cv2.INPAINT_TELEA)

    return sharpened, blackhat_norm, dilated, inpainted


def plot_pipeline(sharp, mask, binary, final):
    plt.figure(figsize=(20, 5))
    plt.subplot(1, 4, 1); plt.title("Sharpened Green"); plt.imshow(sharp,␣
 ↪cmap='gray')
    plt.subplot(1, 4, 2); plt.title("Black-Hat Vessel Map"); plt.imshow(mask,␣
 ↪cmap='gray')
    plt.subplot(1, 4, 3); plt.title("Vessel Mask"); plt.imshow(binary,␣
 ↪cmap='gray')
    plt.subplot(1, 4, 4); plt.title("Inpainted Result"); plt.imshow(final,␣
 ↪cmap='gray')
    plt.tight_layout(); plt.show()

def plot_end_result(image):
    plt.figure(figsize=(10, 10))
    plt.imshow(image, cmap='gray')
    plt.title("Final Image")
    plt.axis('off')
    plt.show()
```

```
sharpened, vessel_map, vessel_mask, result =␣
  ↪remove_vessels_inpaint_refined(green_channel)
plot_pipeline(sharpened, vessel_map, vessel_mask, result)
plot_end_result(result)
```





0.4 Removing the Vessels (Option B)

[ ]:
```
# tensorflow compatibility issues...model written with tesorflow 2.x
#   tensorflow 2.x is not compatible with macOS M2 chip
#   cannot train the model on macOS M2 chip and current tensorflow setup
```

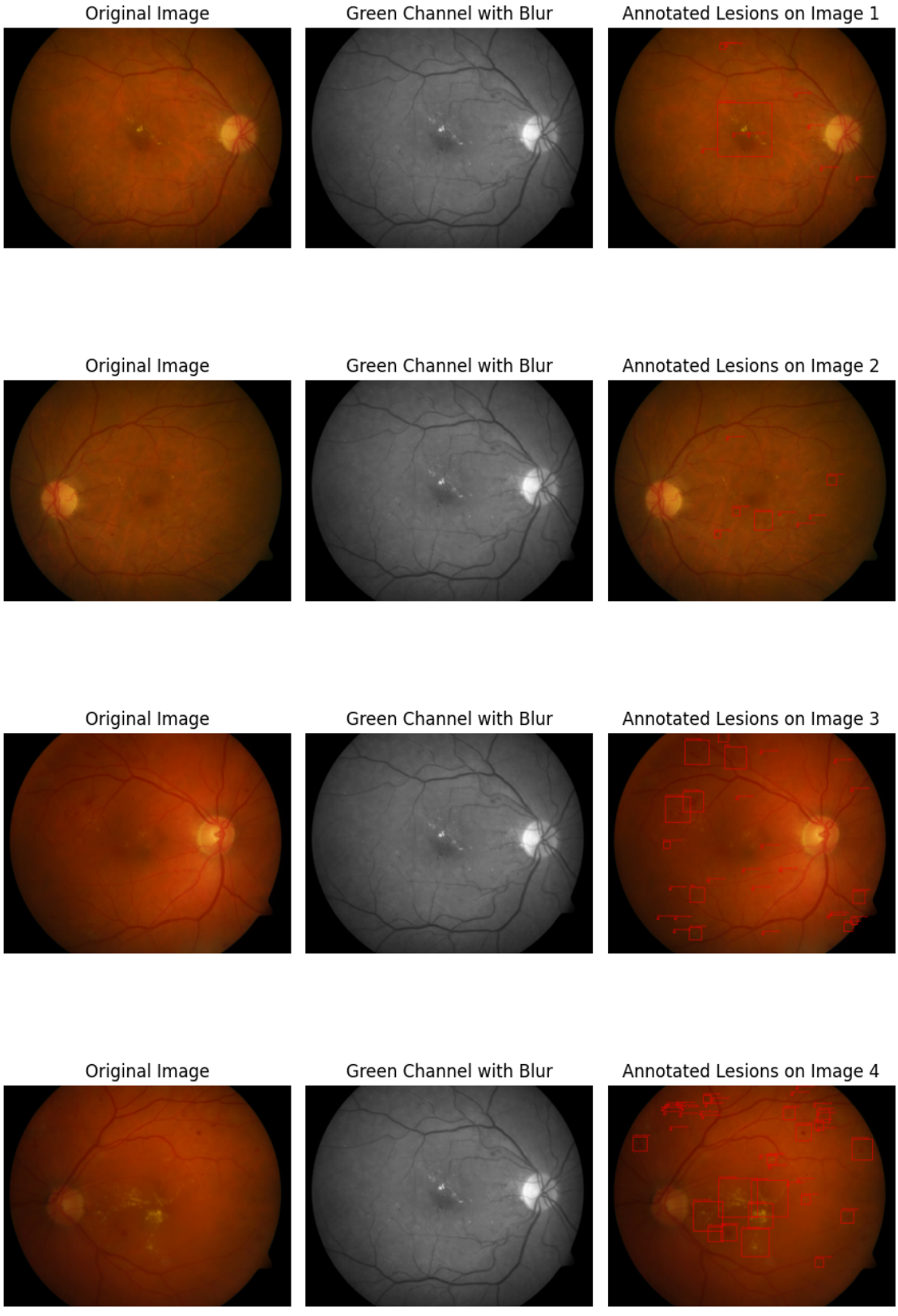
0.5 Original vs Annotated (Single Image)

[ ]:
```
def annotate_lesions_on_image(image_rgb, image_name, xml_file, box_color=(255,␣
 ↪0, 0), thickness=2):

    # Annotate lesions on the image based on the XML annotations

    image_base = os.path.splitext(image_name)[0]

    annotated = image_rgb.copy()

    tree = ET.parse(os.path.join(xml_dir, xml_file))
    root = tree.getroot()

    for mark in root.findall(".//marking"):
        coords = mark.find(".//centroid/coords2d").text
        radius_elem = mark.find(".//circleregion/radius")
        lesion_type = mark.find("markingtype").text

        x, y = map(float, coords.split(','))
        radius = float(radius_elem.text) if radius_elem is not None else -1.0

        x, y, radius = int(x), int(y), int(radius)

        cv2.rectangle(annotated, (x - radius, y - radius), (x + radius, y +␣
 ↪radius), box_color, thickness)
        cv2.putText(annotated, lesion_type, (x - radius, y - radius - 5), cv2.
 ↪FONT_HERSHEY_SIMPLEX,
                    0.4, box_color, 1, cv2.LINE_AA)

    return annotated

def plot_image_with_annotations(image_rgb, annotated_image, title):

    # Plot the original and annotated images side by side

    plt.figure(figsize=(12, 6))
    plt.subplot(1, 4, 1)
    plt.imshow(image_rgb)
    plt.title('Original Image')
    plt.axis('off')
```

```
    plt.subplot(1, 4, 2)
    plt.imshow(green_blur, cmap='gray')
    plt.title('Green Channel with Blur')
    plt.axis('off')

    plt.subplot(1, 4, 3)
    plt.imshow(annotated_image)
    plt.title(title)
    plt.axis('off')

    plt.tight_layout()
    plt.show()


annotated = annotate_lesions_on_image(img_rgb, sample_image, xml_file)
plot_image_with_annotations(img_rgb, annotated, "Annotated Lesions on Image")
```

0.6 Original vs Annotated (Sample of 5)

[ ]:
```
for i, (img_name, xml_name) in bigger_sample.items():
    img_path = os.path.join(image_dir, img_name)
    xml_path = os.path.join(xml_dir, xml_name)

    green_channel = img[:, :, 1]

    img = cv2.imread(img_path)
    img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

    annotated_image = annotate_lesions_on_image(img_rgb, img_name, xml_name)
    plot_image_with_annotations(img_rgb, annotated_image, f"Annotated Lesions␣
 ↪on Image {i}")
```


Original Image
Green Channel with Blur
Annotated Lesions on Image 1
Original Image
Green Channel with Blur
Annotated Lesions on Image 2
Original Image
Green Channel with Blur
Annotated Lesions on Image 3
Original Image
Green Channel with Blur
Annotated Lesions on Image 4


Original Image
Green Channel with Blur
Annotated Lesions on Image 5