import cv2

import numpy as np

from skimage.metrics import structural\_similarity as ssim

from skimage.metrics import mean\_squared\_error as mse

import matplotlib.pyplot as plt

# Load the original Nadia nadia

nadia = cv2.imread('/Users/kasi/Downloads/nadia\_orig.png', cv2.IMREAD\_GRAYSCALE)

# Create an nadia with impulse noise

noisy\_nadia = nadia.copy()

for \_ in range(200):

x, y = np.random.randint(0, nadia.shape[0]), np.random.randint(0, nadia.shape[1])

noisy\_nadia[x, y] = 255 if np.random.rand() > 0.5 else 0

height, width = nadia.shape

# Define the zoom factor and center point

zoom\_factor = 2.0 # Adjust this value to control the amount of zoom

center\_x, center\_y = width // 2, height // 2

# Calculate the dimensions of the zoomed region

zoom\_width = int(width / zoom\_factor)

zoom\_height = int(height / zoom\_factor)

# Calculate the region to be cropped

left = center\_x - zoom\_width // 2

top = center\_y - zoom\_height // 2

right = center\_x + zoom\_width // 2

bottom = center\_y + zoom\_height // 2

# Ensure that the crop dimensions are within the nadia boundaries

left = max(0, left)

top = max(0, top)

right = min(width, right)

bottom = min(height, bottom)

# Crop the region to be zoomed in

zoomed\_region = nadia[top:bottom, left:right]

# Resize the zoomed region to the original nadia size

zoomed\_region = cv2.resize(zoomed\_region, (right - left, bottom - top))

# Create a new nadia with the zoomed region pasted in the middle

distorted\_nadia = np.copy(nadia)

distorted\_nadia[top:bottom, left:right] = zoomed\_region

# Create an nadia with a tiny amount of noise

tiny\_noise = nadia.copy()

tiny\_noise = tiny\_noise.astype(np.float32)

tiny\_noise += np.random.normal(0, 5, tiny\_noise.shape)

tiny\_noise = np.clip(tiny\_noise, 0, 255).astype(np.uint8)

# Calculate SSIM and MSE for each nadia

ssim\_noisy = ssim(nadia, noisy\_nadia)

mse\_noisy = mse(nadia, noisy\_nadia)

ssim\_distorted = ssim(nadia, distorted\_nadia)

mse\_distorted = mse(nadia, distorted\_nadia)

ssim\_tiny\_noise = ssim(nadia, tiny\_noise)

mse\_tiny\_noise = mse(nadia, tiny\_noise)

# Display the nadias and SSIM/MSE values in the specified format

plt.figure(figsize=(15, 12))

plt.subplot(331)

plt.imshow(nadia, cmap='gray')

plt.title('Nadia Original')

plt.subplot(332)

plt.imshow(noisy\_nadia, cmap='gray')

plt.title(f'Noisy nadia\nSSIM: {ssim\_noisy:.4f}\nMSE: {mse\_noisy:.2f}')

plt.subplot(334)

plt.imshow(nadia, cmap='gray')

plt.title('Nadia Original')

plt.subplot(335)

plt.imshow(distorted\_nadia, cmap='gray')

plt.title(f'Distorted nadia\nSSIM: {ssim\_distorted:.4f}\nMSE: {mse\_distorted:.2f}')

plt.subplot(337)

plt.imshow(nadia, cmap='gray')

plt.title('Nadia Original')

plt.subplot(338)

plt.imshow(tiny\_noise, cmap='gray')

plt.title(f'Tiny Noise nadia\nSSIM: {ssim\_tiny\_noise:.4f}\nMSE: {mse\_tiny\_noise:.2f}')

plt.tight\_layout()

plt.show()

A close-up of a person's face

Description automatically generated

import cv2

import numpy as np

from skimage.metrics import structural\_similarity as compare\_ssim

cap = cv2.VideoCapture("/Users/kasi/Downloads/Foreman360p.mp4")

if (cap.isOpened()== False):

print("Error opening file")

video = []

# Read 10 frames

for x in range(10):

ret, frame = cap.read()

frame = cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY)

if ret == True:

video.append(frame)

print(frame.shape, ' ', frame[x][x].shape)

cap.release()

# adding gausian noise

noisy\_video = []

for frame in video:

standard\_dev = 25

mean = 0

gaussian\_noise = np.random.normal(mean, standard\_dev, frame.shape).astype(np.uint8)

noisy\_frame = cv2.add(frame, gaussian\_noise)

noisy\_video.append(noisy\_frame)

noisy\_video = np.array(noisy\_video)

print("Noisy Video:", noisy\_video.shape)

distorted\_video = []

for frame in video:

distorted\_frame = cv2.rotate(frame, cv2.ROTATE\_90\_CLOCKWISE)

distorted\_video.append(distorted\_frame)

distorted\_video = np.array(distorted\_video)

print("Distorted Video:", distorted\_video.shape)

original\_mssim\_distorted = compare\_ssim(video[0], cv2.resize(distorted\_video[0], (video[0].shape[1], video[0].shape[0]))) #compare ssim

original\_mssim = compare\_ssim(video[0], noisy\_video[0]) #comapre mssim

print("MSSIM for Original vs. Distorted Video:", original\_mssim\_distorted)

print("MSSIM for Original vs. Noisy Video:", original\_mssim)

A computer screen with white text

Description automatically generated