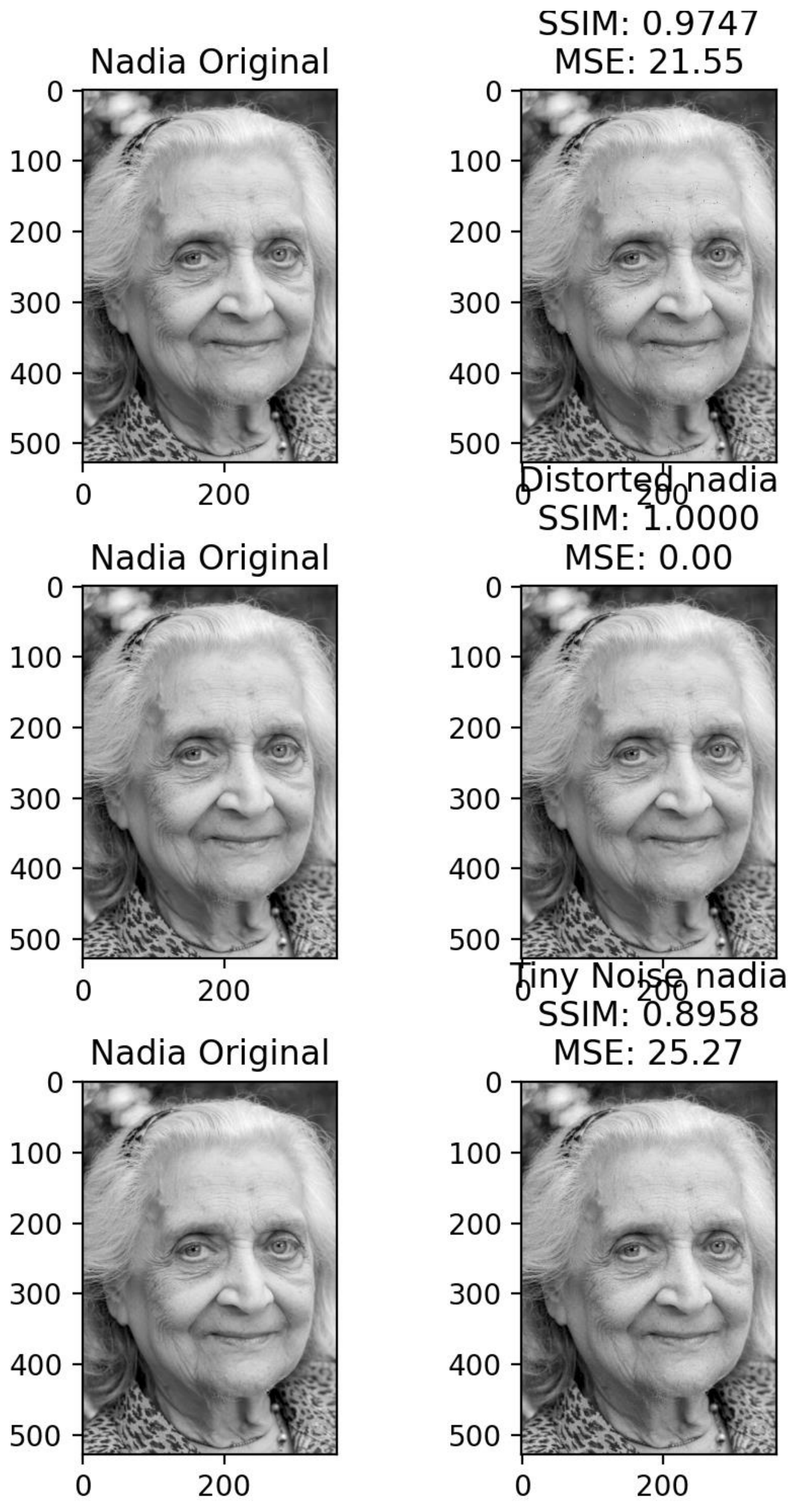
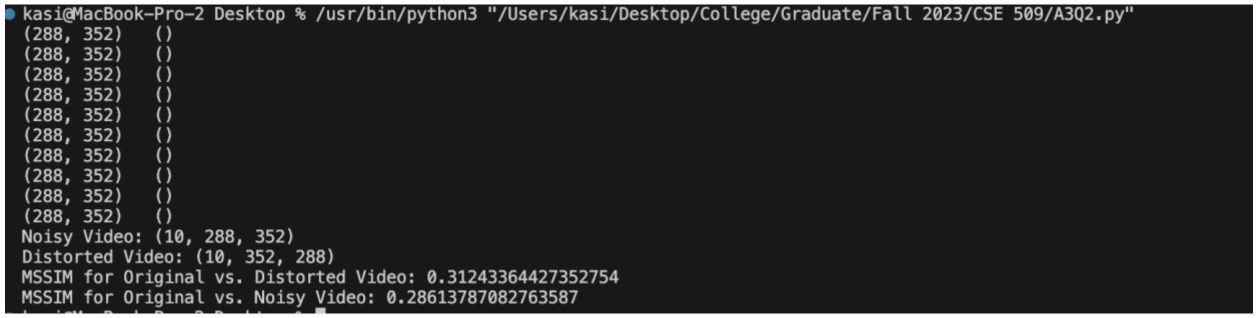
```python
import cv2
import numpy as np
from skimage.metrics import structural_similarity as ssim
from skimage.metrics import mean_squared_error as mse
import matplotlib.pyplot as plt


# 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)
```

```python
# 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()
```


Nadia Original
SSIM: 0.9747
MSE: 21.55
Distorted nadia
Nadia Original
SSIM: 1.0000
MSE: 0.00
Tiny Noise nadia
Nadia Original
SSIM: 0.8958
MSE: 25.27
0
100
200
300
400
500
0
200

```python
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)
```

```python
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)
```



```
kasi@MacBook-Pro-2 Desktop % /usr/bin/python3 "/Users/kasi/Desktop/College/Graduate/Fall 2023/CSE 509/A3Q2.py"
 (288, 352)   ()
 (288, 352)   ()
 (288, 352)   ()
 (288, 352)   ()
 (288, 352)   ()
 (288, 352)   ()
 (288, 352)   ()
 (288, 352)   ()
 (288, 352)   ()
 (288, 352)   ()
 Noisy Video: (10, 288, 352)
 Distorted Video: (10, 352, 288)
 MSSIM for Original vs. Distorted Video: 0.31243364427352754
 MSSIM for Original vs. Noisy Video: 0.28613787082763587
```