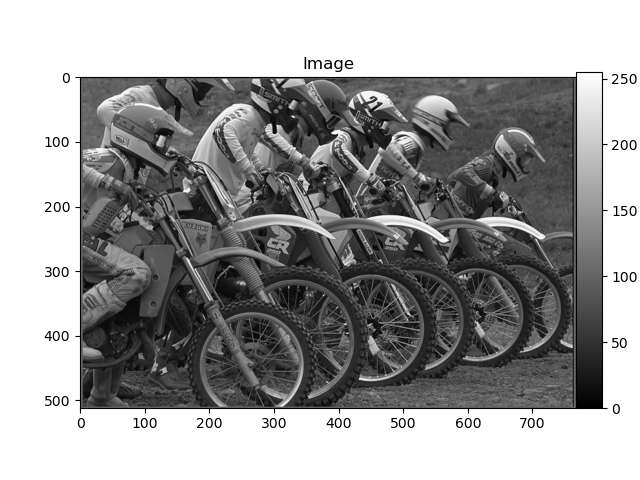
Report Lab 2 – Misha Tsysin, 0033922418

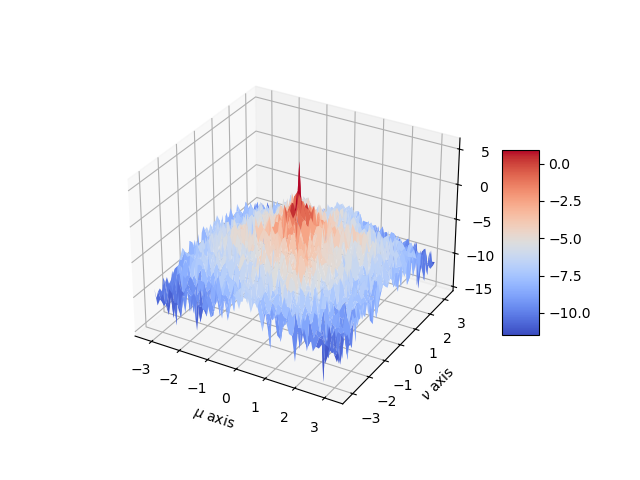
# Power Spectral Density of an Image

The grayscale image img04g.tif looks like:

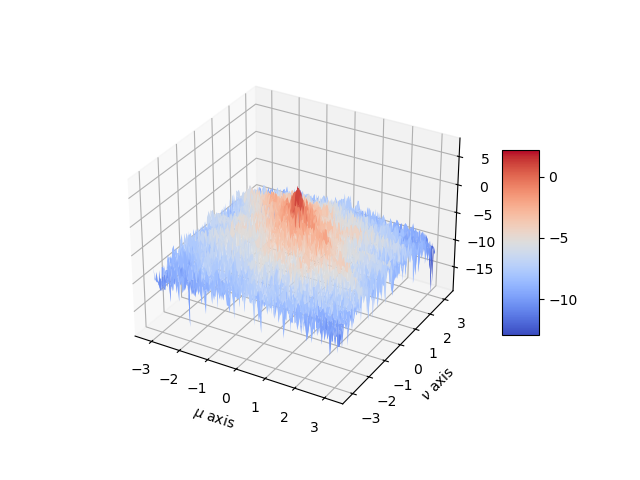


The PSD plots are:

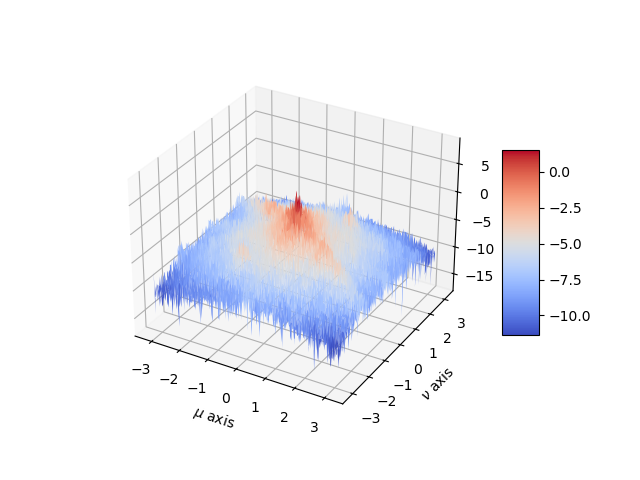
For block size 64:



128:



256:



We see that the PSD is still noisy no matter the size of the block.

The improved PSD estimate is a lot less noisy:

A graph of a function

Description automatically generated

# Power Spectral Density of a 2-D AR Process

The 2d Gaussian random noise will look like:

A graph with numbers and dots

Description automatically generated with medium confidence

Now we can derive the difference equation for the given filter:

By multiplying out and taking the inverse Z transform we can get:

Applying this to the input noise, we can get:

A close-up of a graph

Description automatically generated

The theoretical PSD for this transfer function is equal to:

We know the transfer function. Thus, we just need to find an expression for the original PSD .

We know that PSD of a white noise is just a constant. Thus we can simply look at the variance of each of the gaussians we use for each pixel. For a uniform random variable distributed in [-0.5 , 0.5] the variance is:

Thus the final answer is:

The theoretical PSD will look like:

A graph of a function

Description automatically generated with medium confidence

While the one obtained by BetterSpecAnal is:  
A graph of a graph with a red and blue gradient

Description automatically generated with medium confidence

# Code:

BetterSpecAnal.py:

import numpy as np

import matplotlib.pyplot as plt

from PIL import Image

def BetterSpecAnal(x: np.ndarray, N: int, n: int):

W = np.outer(np.hamming(N), np.hamming(N))

h, w = x.shape

h\_begin = int((h - n \* N) / 2)

w\_begin = int((w - n \* N) / 2)

res = np.zeros((N, N))

for i in range(n):

for j in range(n):

t = W \* x[h\_begin + i \* N: h\_begin + (i+1) \* N,

w\_begin + j \* N: w\_begin + (j+1) \* N]

res += (1 / N\*\*2) \* np.abs(np.fft.fftshift(np.fft.fft2(t) \*\* 2)) \* (1/n\*\*2)

return np.log(res)

if \_\_name\_\_ == "\_\_main\_\_":

im = Image.open('img04g.tif')

print('Read img04.tif.')

print('Image size: ', im.size)

# Import Image Data into Numpy array.

# The matrix x contains a 2-D array of 8-bit gray scale values.

x = np.double(np.array(im))/255.0

N, n = 64, 5

# Plot the result using a 3-D mesh plot and label the x and y axises properly.

fig = plt.figure()

ax = fig.add\_subplot(111, projection='3d')

a = b = np.linspace(-np.pi, np.pi, num = N)

X, Y = np.meshgrid(a, b)

PSD = BetterSpecAnal(x, N, n)

surf = ax.plot\_surface(X, Y, PSD, cmap=plt.cm.coolwarm)

ax.set\_xlabel('$\mu$ axis')

ax.set\_ylabel('$\\nu$ axis')

ax.set\_zlabel('Z Label')

fig.colorbar(surf, shrink=0.5, aspect=5)

plt.show()

Part2.py

import numpy as np

import matplotlib.pyplot as plt

from BetterSpecAnal import BetterSpecAnal

N = 512

x = np.random.uniform(-0.5, 0.5, (N, N))

x\_scaled = 255 \* (x + 0.5)

plt.imshow(np.uint8(x\_scaled), cmap=plt.cm.gray)

plt.show()

y = np.zeros((N, N))

for i in range(N):

for j in range(N):

y[i, j] += 3 \* x[i, j]

if i > 0:

y[i, j] += 0.99 \* y[i-1, j]

if j > 0:

y[i, j] += 0.99 \* y[i, j-1]

if i > 0 and j > 0:

y[i, j] -= 0.9801 \* y[i-1, j-1]

print(np.max(y))

plt.imshow(np.uint8(np.clip(y + 127, 0, 255)), cmap=plt.cm.gray)

plt.show()

s = BetterSpecAnal(y, 64, 5)

U, V = np.meshgrid(np.linspace(-np.pi, np.pi, 64), np.linspace(-np.pi, np.pi, 64))

print (U, V)

theoretical\_s = (1/12) \* np.abs(3 / ((1 - 0.99 \* np.exp(-1j \* U)) \* (1 - 0.99 \* np.exp(-1j \* V)))) \*\* 2

# Plot the result using a 3-D mesh plot and label the x and y axises properly.

fig = plt.figure()

ax = fig.add\_subplot(111, projection='3d')

a = b = np.linspace(-np.pi, np.pi, num = 64)

X, Y = np.meshgrid(a, b)

surf = ax.plot\_surface(X, Y, s, cmap=plt.cm.coolwarm)

ax.set\_xlabel('$\mu$ axis')

ax.set\_ylabel('$\\nu$ axis')

ax.set\_zlabel('Z Label')

fig.colorbar(surf, shrink=0.5, aspect=5)

plt.show()

# Plot the result using a 3-D mesh plot and label the x and y axises properly.

fig = plt.figure()

ax = fig.add\_subplot(111, projection='3d')

a = b = np.linspace(-np.pi, np.pi, num = 64)

X, Y = np.meshgrid(a, b)

surf = ax.plot\_surface(X, Y, np.log(theoretical\_s), cmap=plt.cm.coolwarm)

ax.set\_xlabel('$\mu$ axis')

ax.set\_ylabel('$\\nu$ axis')

ax.set\_zlabel('Z Label')

fig.colorbar(surf, shrink=0.5, aspect=5)

plt.show()