Digital signal Processing Report

BEng in Software and Electronic Engineering

Student Name: Paulina Osikoya

Student Number: G00348898

Lecturer: Michelle Lynch

Module: Digital Signal Processing

Year: 4

Preface

This report is presented in partial fulfilment of the requirements for the degree of Bachelor of Engineering in Software & Electronic Engineering at Galway-Mayo Institute of Technology. This report will be presented in week nine as part of the continuious assesment of the five credit module Digital Signal Processing(DSP). The construction and implementation of this project is based off my own ideologys and opinions around the topic and thus is my own work and , except where otherwise accredited. Where the work of others has been used or incorporated during this project have been acknowledged and apporiately referenced.

Technologies used:

- Python3
- OpenCV
- numpy
- matplotlib
- nft and fft and libraries associated with fourier transform

Topics that will be disccused in this report

For my DSP Report, the topic that I decided to choose of the three was Fourier and the following topics that I will be commentating on and analyising in depth in my report are the following:

- Why I choose my topic?
- Introduction to Fourier
- Who was Fourier and how his series has revolutionized the world we live in today?
- What is the Fourier transform?
- Signals and their importance
- Humans viewing images vs comptuers
- The Fourier tranform and analysis:
  - DFT
  - FFT
  - DCT
  - IRFT
- Applications of Fourier
  - Brief explaination of sound analysis in Fourier
- What i have deduced from the Fourier transform?
- Conclusion

Why I choose my topic?

I choose the Fourier transform as my report topic for the module DSP because, while studying this topic, I became very intrigue by the operations and applications of the transforms and how they can be apply in the world that we live in today. A few of my interests where closely tied to how the Fourier transform can be ultilized in image processing and audio of which i will be covering later in this report

In addition, I choose this topic to deepen my understanding in this particular topic and also I wanted to see what I could create with Fourier and also what new findings I might aquire while researching and analysizing this topic in great depth

Who was Fourier and how his series has revolutionized the world we live in today?

While reading mutiple sources, I was able to gain an understanding about the French Mathematician Joesph Fourier and his contribution to the world of science, engineering and physics which is the fourier series.The Fourier series still plays a prevalant role in modern mathematics.

Fourier's contriubution revolunized the world we live in today premised on the fact that his equation makes it easier for us to compute and visualize things in their sine and cosine components and without his exploration into this topic this wouldn't have be possible.

What is the Fourier transform?

The Fourier Transform is a linear function that is used in many of applications, such as image analysis, image filtering, image reconstruction and image compression[1]. contain

Before we use the fourier transform on an image, it is important to note that every image is in the spatial domain. The spatial domain deals with time and the frequency domain deals with frequencies and it is the Fourier transform that converts an image from the spatila domain to the frequency domain. while researching, i found a interesting thread on the Fourier transform on the question forum website Quor and a mathmetcian, by occupation, describd the Fourier transform, i think in a clear informative way which was:

"The Fourier Transform is a very important image processing tool which is used to decompose an image into its sine and cosine components. The output of the transformation represents the image in the Fourier or frequency domain, while the input image is the spatial domain equivalent".[1]

The fourier transform can be broken into several different aspects which are the following:

- FFT Fast Fourier Trasform
- DFT Discrete Fourier Transform
- DCT Discrete Cosine Transform
- IRFT Infared Fourier Transform

in regard to IRFT I have added a creative aspect where i have used deductive resoning and analysis of the various different ouptut from the frequenct spectrum to be able to digtusih betweeen different matasisi in thedifferent types og cancrrsa

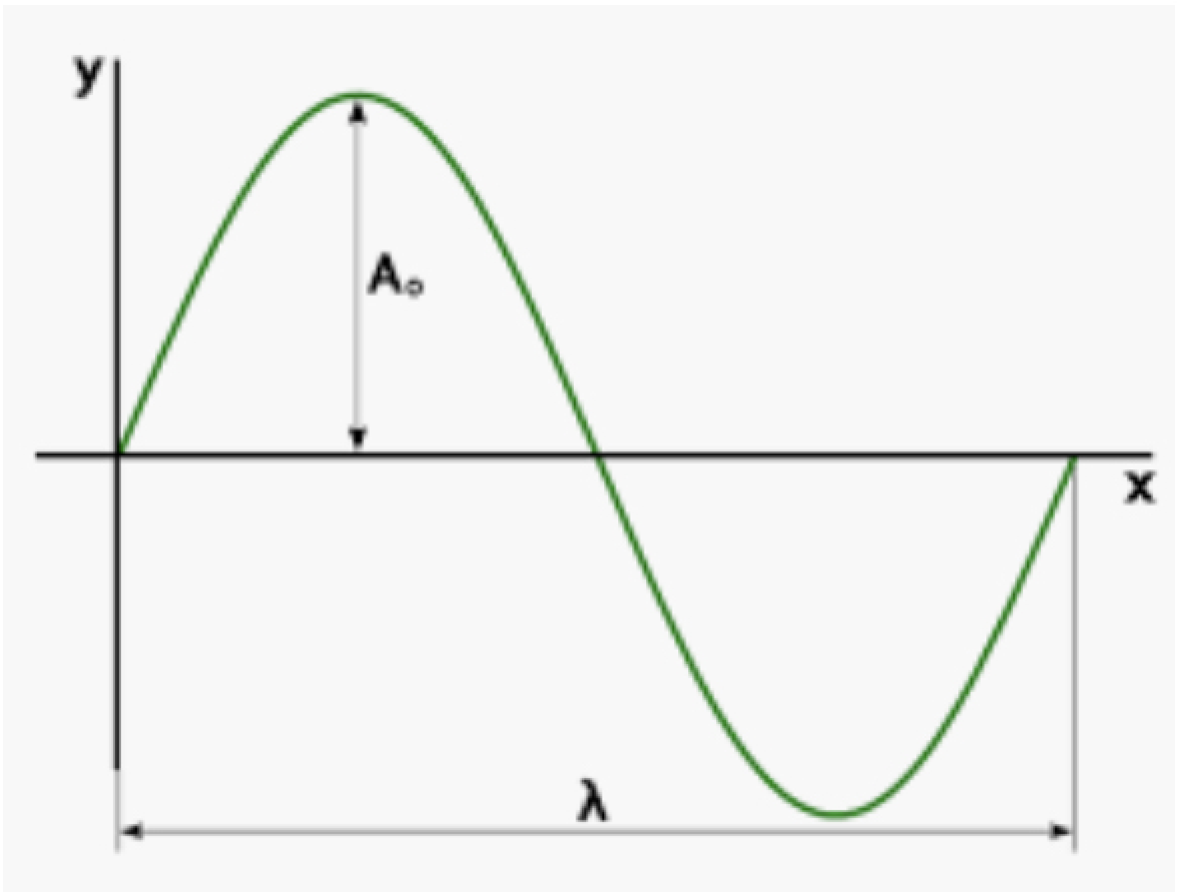
I will delve into these areas later in my report

Signals and their importance

Signals are very important as they allow us gain greater insight when we are analysizing images that have been converted into the frequency domain by the Fourier transform. Signals enable us to recognize patterns in frequency domain and these patternscan been used for detecting objects and finding similarity between objects aswell.

In Fourier, we mainly deal with Sinuoids signals which look like this:

Signal picture



Humans viewing images vs comptuers

Before we even begin to step into the analysis of the fourier series, it is very important for us to understand two vital things; how we, humans see images and how computers view images and how they process them.

Take the two images that i took in Amsterdam below for instance, to the human eye they look and may seem very similar. but the image on the left hand side has been slightly altered by me using my iphone's photo edititng software that come with the phone.In relation to the photo on the left, I altered various aspects of the photo like: brightness, staturation, focus and sharpness. The modifications that I made to the photo were so slight that I wanted to make sure the photo still retained a major resemblance to the orginal.

Two images side by side

However, while the photo may have looked identical to the us, The computer would be able to pick up on these differences and would be able to identify the differences between the two photos.

And you may ask why that maybe?

That is because unlike humans, computers are machines and they are designed to interpret images as a matrix.In the matrix, the colour in each matrix element is coresponds a number.Depending on the number that is assocaited with elment gives it its colour. Larger numbers have darker colours on the colour spectrum and opposite for smaller numbers.

In regard to the Fourier transform on images, the computer would read in the images as a matrix and take the fourier transform along the row and columns to compute the the images in the frequency domain

The Fourier tranform and analysis

DFT

When looking at the Discrete Fourier Transform, we will take for example white noise. in the time domain, each amplitude of white noise is seen as function of time.When you translate in to the freqency domain you would get out a straight line and each freqency would be equally represented because it is composed of a sum of cosines and sines of different frequencies. but in practice the output would not be so perfect to due to fact that the sensor that you record on would not be equally senstive to all frequencies andthe white noise wouldn't be perfect white noise either. perdoic signal that you can decomposed the opersation of converting the white noise to a straight line in the frequency domain is done by the DFT white noise

fourier transform is expressed as this and is the summation of all samples mutliplied by the complex co-effcients.

The Discrete Fourier Transform can be expressed as

$$\mathbf{F}(n) = \sum_{k=0}^{N-1} f(k)\, e^{-j2\pi nk/N} \qquad (n = 0..1\ :N-1)$$

The relevant inverse Fourier Transform can be expressed as

$$f(k) = \frac{1}{N} \sum_{n=0}^{N-1} \mathbf{F}(n)\, e^{j2\pi nk/N} \qquad (\ k = 0,\ 1,\ 2...,N-1\ )$$

FFT

The FFT which stands for the Fast Fourier Transform is a quicker way to compute the DFT. How the FFT is achieved is by taking advantage of the periodic and predictable nature of a sine wave. like the DFT the FFT can also be used for applications listed above

DCT

The DCT whch stands for Discrete Cosine Transform is very similar to DFT. This transform also takes images from the spatial domain to the frequency domain but the key difference between the two is the DCT only uses real numbers for its transform and the DFT uses complex numbers which comprises of real and imaginary parts

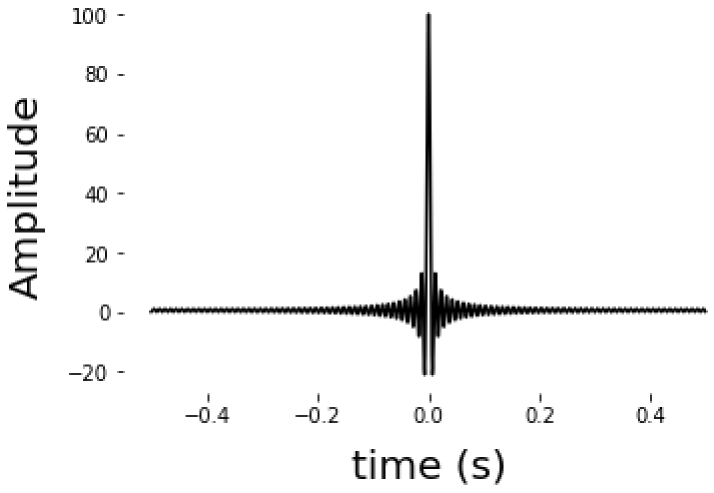
IRFT

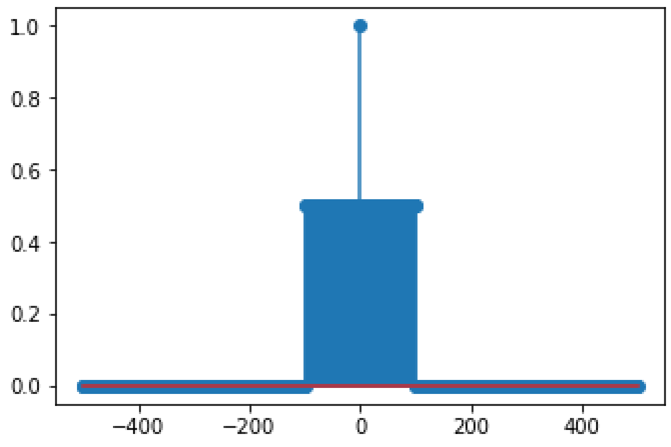
Fourier-transform infrared is used to take infared images to translate them into the frequenchy domain so analysis can be performed and also pattern similarity can be observed also.

Here in this examples we see an impulse signal and we can see at certain parts of the signal the values are near zero and there is a spike and afterwards the signals where the values are low repeat

In [16]:

```
# -*- coding: utf-8 -*-
"""
Created on Thu Jan 30 11:06:59 2020

@author: G00348898
"""

#importing the needed libraries
import numpy as np
import matplotlib as mpl
import matplotlib.pyplot as plt

#setting the different vars
fs = 1000
duration = 1
N = fs * duration # total number of samples
t = np.linspace(-duration/2, duration/2, num=N, endpoint=False)

#calibrating the graph
impulse_t = np.zeros(t.size)
for f in range(0,100):
    s = np.cos(2*np.pi*f*t)
    impulse_t = impulse_t + s

#getting the DFT
impulse_f = np.fft.fft(impulse_t)
impulse_f_abs= (1/N)*np.abs(impulse_f)
freqs =np.fft.fftfreq(N, 1/fs)

#plotting the graph
fig = plt.figure(figsize = (12,8))
ax = fig.add_subplot(2,2,1)
ax.plot(t, impulse_t, color='k')

#formatting the graph
ax.spines['bottom'].set_visible(False)
ax.spines['top'].set_visible(False)
ax.spines['left'].set_visible(False)
ax.spines['right'].set_visible(False)

#setting the names on either axis
ax.set_xlabel('time (s)', fontsize=20, labelpad=10)
ax.set_ylabel('Amplitude', fontsize=20, labelpad=10)

fig2 = plt.figure(figsize = (12,8))
ax = fig2.add_subplot(2,2,2)
markerlines, stemlines, baseline = ax.stem(freqs, impulse_f_abs)
```

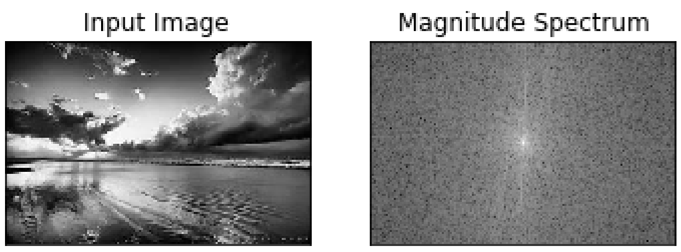




Here we see an image that has been translated to the frequency domain and is plotted on a magnitude spectrum. the magnitude spectrum can help us identifty what the image in the spatial domain total energy from the signal. Also the second picture below captures how the image that is now in the frequency domain can go back to the spatial domain by using the inverse fourier transform [2]

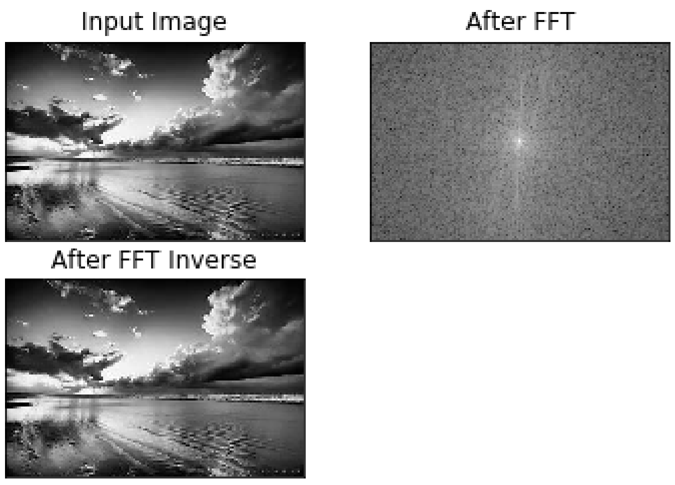
In [39]:

```
import numpy as np
import cv2
from matplotlib import pyplot as plt

img = cv2.imread('sky.jpg',0)

dft = cv2.dft(np.float32(img),flags = cv2.DFT_COMPLEX_OUTPUT)
dft_shift = np.fft.fftshift(dft)

magnitude_spectrum = 20*np.log(cv2.magnitude(dft_shift[:,:,0],dft_shift[:,:,1]))

plt.subplot(121),plt.imshow(img, cmap = 'gray')
plt.title('Input Image'), plt.xticks([]), plt.yticks([])
plt.subplot(122),plt.imshow(magnitude_spectrum, cmap = 'gray')
plt.title('Magnitude Spectrum'), plt.xticks([]), plt.yticks([])
plt.show()
```



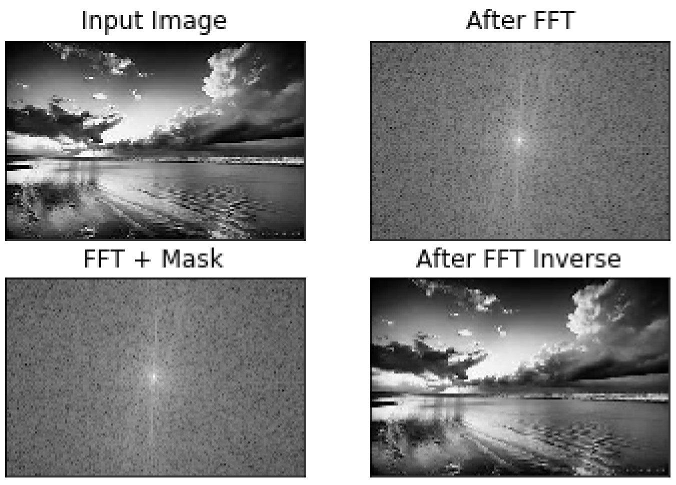
In [11]:

```
img = cv2.imread("sky.jpg", 0)
rows, cols = img.shape
crow, ccol = int(rows / 2), int(cols / 2)  # center

# Circular HPF mask, center circle is 0, remaining all ones

mask = np.ones((rows, cols, 2), np.uint8)
r = 80
center = [crow, ccol]
x, y = np.ogrid[:rows, :cols]
mask_area = (x - center[0]) ** 2 + (y - center[1]) ** 2 <= r*r
mask[mask_area] = 1

# apply mask and inverse DFT
fshift = dft_shift * mask

fshift_mask_mag = 2000 * np.log(cv2.magnitude(fshift[:, :, 0], fshift[:, :, 1]))

f_ishift = np.fft.ifftshift(fshift)
img_back = cv2.idft(f_ishift)
img_back = cv2.magnitude(img_back[:, :, 0], img_back[:, :, 1])

plt.subplot(2, 2, 1), plt.imshow(img, cmap='gray')
plt.title('Input Image'), plt.xticks([]), plt.yticks([])
plt.subplot(2, 2, 2), plt.imshow(magnitude_spectrum, cmap='gray')
plt.title('After FFT'), plt.xticks([]), plt.yticks([])
plt.subplot(2, 2, 3), plt.imshow(img_back, cmap='gray')
plt.title('After FFT Inverse'), plt.xticks([]), plt.yticks([])
plt.show()
```



In [32]:

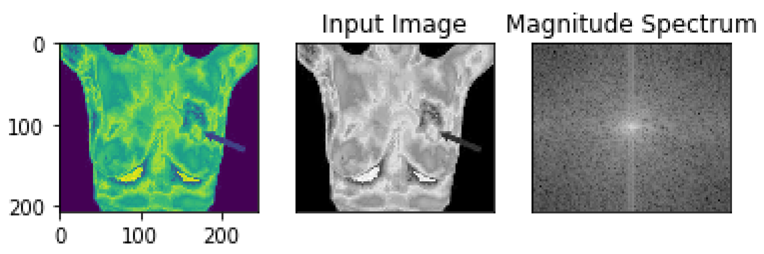
```
rows, cols = img.shape
crow, ccol = int(rows / 2), int(cols / 2)  # center

# Circular HPF mask, center circle is 0, remaining all ones

mask = np.ones((rows, cols, 2), np.uint8)
r = 80
center = [crow, ccol]
x, y = np.ogrid[:rows, :cols]
mask_area = (x - center[0]) ** 2 + (y - center[1]) ** 2 <= r*r
mask[mask_area] = 1

# apply mask and inverse DFT
fshift = dft_shift * mask

fshift_mask_mag = 2000 * np.log(cv2.magnitude(fshift[:, :, 0], fshift[:, :, 1]))

f_ishift = np.fft.ifftshift(fshift)
img_back = cv2.idft(f_ishift)
img_back = cv2.magnitude(img_back[:, :, 0], img_back[:, :, 1])

plt.subplot(2, 2, 1), plt.imshow(img, cmap='gray')
plt.title('Input Image'), plt.xticks([]), plt.yticks([])
plt.subplot(2, 2, 2), plt.imshow(magnitude_spectrum, cmap='gray')
plt.title('After FFT'), plt.xticks([]), plt.yticks([])
plt.subplot(2, 2, 3), plt.imshow(fshift_mask_mag, cmap='gray')
plt.title('FFT + Mask'), plt.xticks([]), plt.yticks([])
plt.subplot(2, 2, 4), plt.imshow(img_back, cmap='gray')
plt.title('After FFT Inverse'), plt.xticks([]), plt.yticks([])
plt.show()
```



Here i took time to research the topic Infared Fourier transform and this area is well documented in the world of science and medicine. Here the fourier transform is used in infared images in the spatial domain to detect pattern similarities in the freqency domain and from this they can discover things like cancerous cells forming in the body and the type of cancer it is based on the pattern.This has been done for Breast Cancer and Prostate Cancer. As part of my research, although i didn't have equipment capable of taking infared pictures, i took pictures off the web of infared images of breast cancer and also of prostate cancer and the results were quite interesting.the two images of the breast cancer show quite smiliar results on the magnitude spectrum and the prostate cancer image show a different result and also on the zoomed up images of the types of cancer cells the results were very similar. If i had the proper equipment the result would be more accurate but i found it very interesting with what i could achive.[3][4]

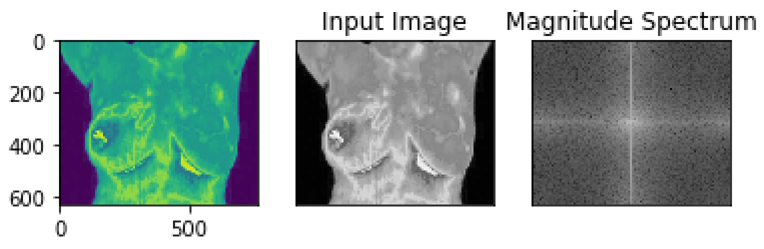
In [24]:

```
import cv2
import numpy as np
from matplotlib import pyplot as plt

img = cv2.imread('img2.jpg',0)
f = np.fft.fft2(img)
fshift = np.fft.fftshift(f)
magnitude_spectrum = 20*np.log(np.abs(fshift))

plt.subplot(131),plt.imshow(img)
plt.subplot(132),plt.imshow(img, cmap = 'gray')
plt.title('Input Image'), plt.xticks([]), plt.yticks([])
plt.subplot(133),plt.imshow(magnitude_spectrum, cmap = 'gray')
plt.title('Magnitude Spectrum'), plt.xticks([]), plt.yticks([])
plt.show()
```



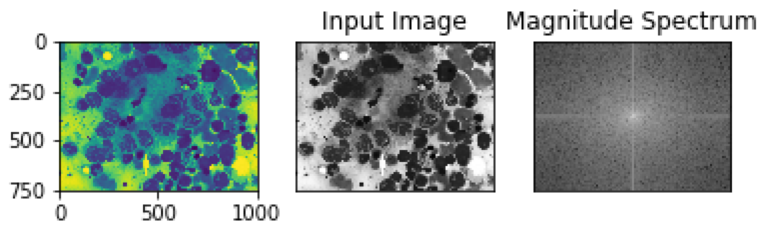
In [17]:

```
import cv2
import numpy as np
from matplotlib import pyplot as plt

img = cv2.imread('img1.png',0)
f = np.fft.fft2(img)
fshift = np.fft.fftshift(f)
magnitude_spectrum = 20*np.log(np.abs(fshift))

plt.subplot(131),plt.imshow(img)
plt.subplot(132),plt.imshow(img, cmap = 'gray')
plt.title('Input Image'), plt.xticks([]), plt.yticks([])
plt.subplot(133),plt.imshow(magnitude_spectrum, cmap = 'gray')
plt.title('Magnitude Spectrum'), plt.xticks([]), plt.yticks([])
plt.show()
```



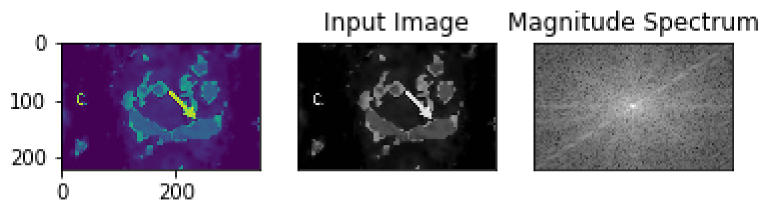
In [4]:

```
import cv2
import numpy as np
from matplotlib import pyplot as plt

img = cv2.imread('breast cancer cells.jpg',0)
f = np.fft.fft2(img)
fshift = np.fft.fftshift(f)
magnitude_spectrum = 20*np.log(np.abs(fshift))

plt.subplot(131),plt.imshow(img)
plt.subplot(132),plt.imshow(img, cmap = 'gray')
plt.title('Input Image'), plt.xticks([]), plt.yticks([])
plt.subplot(133),plt.imshow(magnitude_spectrum, cmap = 'gray')
plt.title('Magnitude Spectrum'), plt.xticks([]), plt.yticks([])
plt.show()
```



In [6]:

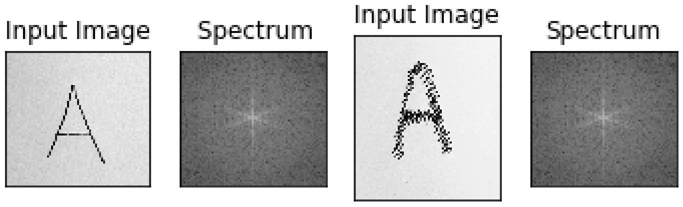
```
import cv2
import numpy as np
from matplotlib import pyplot as plt

img = cv2.imread('prostate cancee.jpg',0)
f = np.fft.fft2(img)
fshift = np.fft.fftshift(f)
magnitude_spectrum = 20*np.log(np.abs(fshift))

plt.subplot(131),plt.imshow(img)
plt.subplot(132),plt.imshow(img, cmap = 'gray')
plt.title('Input Image'), plt.xticks([]), plt.yticks([])
plt.subplot(133),plt.imshow(magnitude_spectrum, cmap = 'gray')
plt.title('Magnitude Spectrum'), plt.xticks([]), plt.yticks([])
plt.show()
```



Here i have drawn out various letters from the alphabet and have plotted them with their corresponding magnitude spectrum and from this i could see that no matter how many times i drew the letter "A" different it still showed on the magnitude spectrum as the same for any given drawing of A
however when i drew letters "B" , "C", "D", they showed up different on the magnitude spectrum which affirmes the fact that the fourier transform can be used for text detection. This is often why websites often given their users blurred out images of letters to confirm that their aren't robots/computers as computer have difficulty reading text.

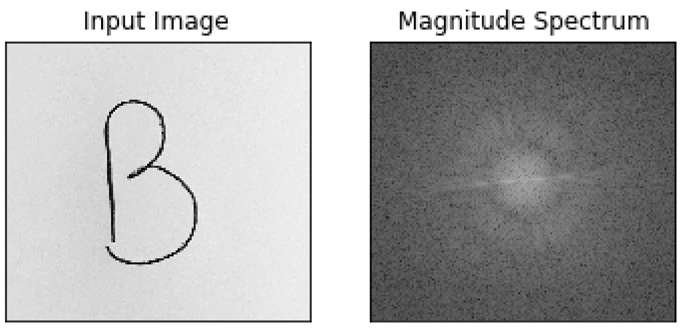
In [51]:

```
import numpy as np
import cv2
from matplotlib import pyplot as plt

img = cv2.imread('a1.jpg',0)
img1 = cv2.imread('a2.jpg',0)
img2 = cv2.imread('a3.jpg',0)

dft = cv2.dft(np.float32(img),flags = cv2.DFT_COMPLEX_OUTPUT)
dft_shift = np.fft.fftshift(dft)

dft1 = cv2.dft(np.float32(img),flags = cv2.DFT_COMPLEX_OUTPUT)
dft_shift1 = np.fft.fftshift(dft1)

magnitude_spectrum = 20*np.log(cv2.magnitude(dft_shift[:,:,0],dft_shift[:,:,1]))
magnitude_spectrum1 = 20*np.log(cv2.magnitude(dft_shift1[:,:,0],dft_shift1[:,:,1]))

plt.subplot(141),plt.imshow(img, cmap = 'gray')
plt.title('Input Image'), plt.xticks([]), plt.yticks([])
plt.subplot(142),plt.imshow(magnitude_spectrum, cmap = 'gray')
plt.title(' Spectrum'), plt.xticks([]), plt.yticks([])
plt.subplot(143),plt.imshow(img1, cmap = 'gray')
plt.title('Input Image'), plt.xticks([]), plt.yticks([])
plt.subplot(144),plt.imshow(magnitude_spectrum1, cmap = 'gray')
plt.title('Spectrum'), plt.xticks([]), plt.yticks([])

plt.show()
```



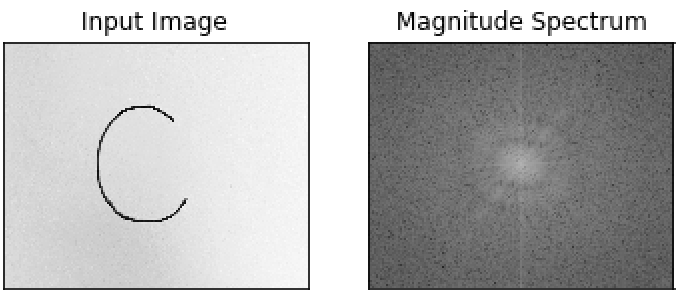
In [52]:

```
import numpy as np
import cv2
from matplotlib import pyplot as plt

img = cv2.imread('b.jpg',0)

dft = cv2.dft(np.float32(img),flags = cv2.DFT_COMPLEX_OUTPUT)
dft_shift = np.fft.fftshift(dft)

magnitude_spectrum = 20*np.log(cv2.magnitude(dft_shift[:,:,0],dft_shift[:,:,1]))

plt.subplot(121),plt.imshow(img, cmap = 'gray')
plt.title('Input Image'), plt.xticks([]), plt.yticks([])
plt.subplot(122),plt.imshow(magnitude_spectrum, cmap = 'gray')
plt.title('Magnitude Spectrum'), plt.xticks([]), plt.yticks([])
plt.show()
```



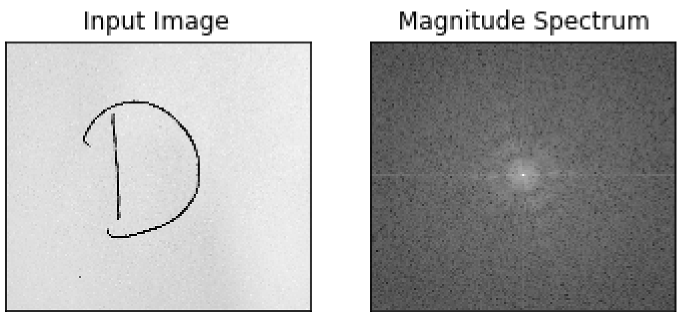
In [53]:

```
import numpy as np
import cv2
from matplotlib import pyplot as plt

img = cv2.imread('c.jpg',0)

dft = cv2.dft(np.float32(img),flags = cv2.DFT_COMPLEX_OUTPUT)
dft_shift = np.fft.fftshift(dft)

magnitude_spectrum = 20*np.log(cv2.magnitude(dft_shift[:,:,0],dft_shift[:,:,1]))

plt.subplot(121),plt.imshow(img, cmap = 'gray')
plt.title('Input Image'), plt.xticks([]), plt.yticks([])
plt.subplot(122),plt.imshow(magnitude_spectrum, cmap = 'gray')
plt.title('Magnitude Spectrum'), plt.xticks([]), plt.yticks([])
plt.show()
```



In [54]:

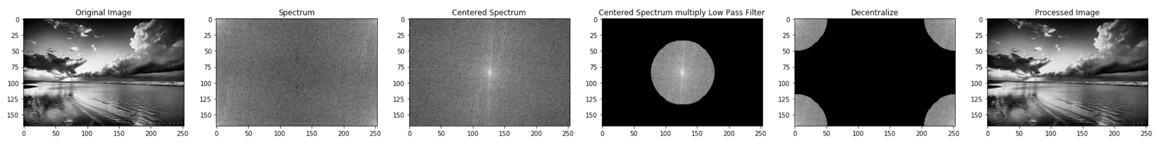
```
import numpy as np
import cv2
from matplotlib import pyplot as plt

img = cv2.imread('d.jpg',0)

dft = cv2.dft(np.float32(img),flags = cv2.DFT_COMPLEX_OUTPUT)
dft_shift = np.fft.fftshift(dft)

magnitude_spectrum = 20*np.log(cv2.magnitude(dft_shift[:,:,0],dft_shift[:,:,1]))

plt.subplot(121),plt.imshow(img, cmap = 'gray')
plt.title('Input Image'), plt.xticks([]), plt.yticks([])
plt.subplot(122),plt.imshow(magnitude_spectrum, cmap = 'gray')
plt.title('Magnitude Spectrum'), plt.xticks([]), plt.yticks([])
plt.show()
```



Here we have taken our sky image again and we have applied a low pass filter to the image. low pass filters are very good for cutting out high frequencies and it is commonly used for things like blurring an image [2]

In [27]:

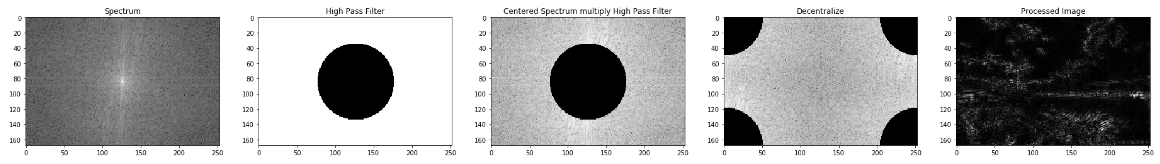
```
import cv2
import numpy as np
import matplotlib.pyplot as plt
from math import sqrt,exp
def distance(point1,point2):

    return sqrt((point1[0]-point2[0])**2 + (point1[1]-point2[1])**2)

def idealFilterLP(D0,imgShape):
    base = np.zeros(imgShape[:2])
    rows, cols = imgShape[:2]
    center = (rows/2,cols/2)
    for x in range(cols):
        for y in range(rows):
            if distance((y,x),center) < D0:
                base[y,x] = 1
    return base

def idealFilterHP(D0,imgShape):
    base = np.ones(imgShape[:2])
    rows, cols = imgShape[:2]
    center = (rows/2,cols/2)
    for x in range(cols):
        for y in range(rows):
            if distance((y,x),center) < D0:
                base[y,x] = 0
    return base

def butterworthLP(D0,imgShape,n):
    base = np.zeros(imgShape[:2])
    rows, cols = imgShape[:2]
    center = (rows/2,cols/2)
    for x in range(cols):
        for y in range(rows):
            base[y,x] = 1/(1+(distance((y,x),center)/D0)**(2*n))
    return base

def butterworthHP(D0,imgShape,n):
    base = np.zeros(imgShape[:2])
    rows, cols = imgShape[:2]
    center = (rows/2,cols/2)
    for x in range(cols):
        for y in range(rows):
            base[y,x] = 1-1/(1+(distance((y,x),center)/D0)**(2*n))
    return base

def gaussianLP(D0,imgShape):
    base = np.zeros(imgShape[:2])
    rows, cols = imgShape[:2]
    center = (rows/2,cols/2)
    for x in range(cols):
        for y in range(rows):
            base[y,x] = exp(((-distance((y,x),center)**2)/(2*(D0**2))))
    return base

def gaussianHP(D0,imgShape):
    base = np.zeros(imgShape[:2])
    rows, cols = imgShape[:2]
    center = (rows/2,cols/2)
    for x in range(cols):
        for y in range(rows):
            base[y,x] = 1 - exp(((-distance((y,x),center)**2)/(2*(D0**2))))
    return base

plt.figure(figsize=(6.4*5, 4.8*5), constrained_layout=False)

img = cv2.imread("sky.jpg", 0)
plt.subplot(161), plt.imshow(img, "gray"), plt.title("Original Image")

original = np.fft.fft2(img)
plt.subplot(162), plt.imshow(np.log(1+np.abs(original)), "gray"), plt.title("Spectrum")

center = np.fft.fftshift(original)
plt.subplot(163), plt.imshow(np.log(1+np.abs(center)), "gray"), plt.title("Centered Spe
ctrum")

LowPassCenter = center * idealFilterLP(50,img.shape)
plt.subplot(164), plt.imshow(np.log(1+np.abs(LowPassCenter)), "gray"), plt.title("Cente
red Spectrum multiply Low Pass Filter")

LowPass = np.fft.ifftshift(LowPassCenter)
plt.subplot(165), plt.imshow(np.log(1+np.abs(LowPass)), "gray"), plt.title("Decentraliz
e")

inverse_LowPass = np.fft.ifft2(LowPass)
plt.subplot(166), plt.imshow(np.abs(inverse_LowPass), "gray"), plt.title("Processed Ima
ge")

plt.show()
```



Here we have taken our sky image again and we have applied various high pass filters to them. High pass filters are very good for cutting out low frequencies and it is commonly used for things like edge detection[2]

In [30]:

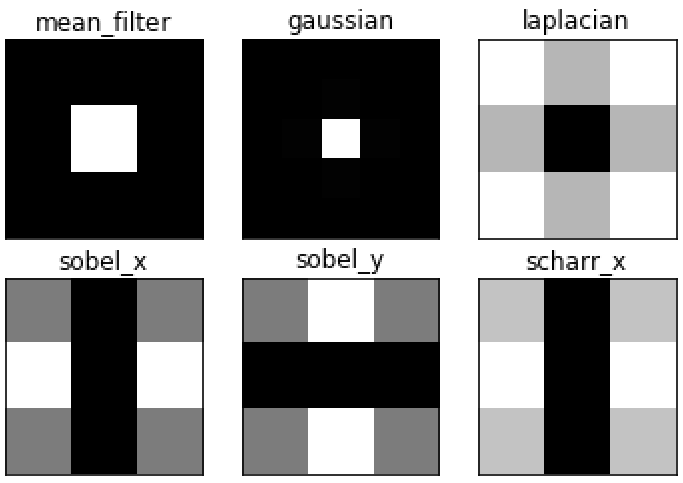
```
img = cv2.imread("sky.jpg", 0)
original = np.fft.fft2(img)
center = np.fft.fftshift(original)

plt.figure(figsize=(6.4*5, 4.8*5), constrained_layout=False)

plt.subplot(151), plt.imshow(np.log(1+np.abs(center)), "gray"), plt.title("Spectrum")

HighPass = idealFilterHP(50,img.shape)
plt.subplot(152), plt.imshow(np.abs(HighPass), "gray"), plt.title("High Pass Filter")

HighPassCenter = center * idealFilterHP(50,img.shape)
plt.subplot(153), plt.imshow(np.log(1+np.abs(HighPassCenter)), "gray"), plt.title("Cent
ered Spectrum multiply High Pass Filter")

HighPass = np.fft.ifftshift(HighPassCenter)
plt.subplot(154), plt.imshow(np.log(1+np.abs(HighPass)), "gray"), plt.title("Decentrali
ze")

inverse_HighPass = np.fft.ifft2(HighPass)
plt.subplot(155), plt.imshow(np.abs(inverse_HighPass), "gray"), plt.title("Processed Im
age")

plt.show()
```



Here the Fourier transform has been applied to the various filters and from the images below we can see which filters are high pass filter and which filters and low pass

In [22]:

```
import cv2
import numpy as np
from matplotlib import pyplot as plt

# simple averaging filter without scaling parameter
mean_filter = np.ones((3,3))

# creating a guassian filter
x = cv2.getGaussianKernel(5,10)
gaussian = x*x.T

# different edge detecting filters
# scharr in x-direction
scharr = np.array([[-3, 0, 3],
                   [-10,0,10],
                   [-3, 0, 3]])
# sobel in x direction
sobel_x= np.array([[-1, 0, 1],
                   [-2, 0, 2],
                   [-1, 0, 1]])
# sobel in y direction
sobel_y= np.array([[-1,-2,-1],
                   [0, 0, 0],
                   [1, 2, 1]])
# laplacian
laplacian=np.array([[0, 1, 0],
                    [1,-4, 1],
                    [0, 1, 0]])

filters = [mean_filter, gaussian, laplacian, sobel_x, sobel_y, scharr]
filter_name = ['mean_filter', 'gaussian','laplacian', 'sobel_x', \
                'sobel_y', 'scharr_x']
fft_filters = [np.fft.fft2(x) for x in filters]
fft_shift = [np.fft.fftshift(y) for y in fft_filters]
mag_spectrum = [np.log(np.abs(z)+1) for z in fft_shift]

for i in range(6):
    plt.subplot(2,3,i+1),plt.imshow(mag_spectrum[i],cmap = 'gray')
    plt.title(filter_name[i]), plt.xticks([]), plt.yticks([])

plt.show()
```



Here we can see that the Mean filter and the Guassian filter are low pass filters and the rest listed are high pass filters[4]

In [ ]:

```
- to do
do the rec of the etters
do the formting of the shape
tidy up the general shape
mention at the end music and sounds and how they are made
ref
medium source
https://medium.com/@hicraigchen/digital-image-processing-using-fourier-transform-in-pyt
hon-bcb49424fd82
orielly music
https://www.oreilly.com/library/view/elegant-scipy/9781491922927/ch04.html
    edge detection github
    https://akshaysin.github.io/fourier_transform.html#.XmaLYqj7Tcs
        letter rec
        https://github.com/crowoy/Letter-Recognition/blob/master/src/FourierAnalysis.ip
ynb
            oscon talk
            https://github.com/calebmadrigal/FourierTalkOSCON
```

Applications of Fourier

Brief explaination of sound analysis in Fourier

Through out my project I have gone into great depth of explaining and analysising the fourier series in relation to image processing and how important it is in the world that we live in today.

However, during my research, i foud that while the fourier transform is used for image processing it is also used for sound analysis. Based on this information I wanted to see if I could do something creative and then provide my analysis on it.

Being a musican, I wanted to see what woud happen when one applies fourier to piano pieces and what the output would be. So remised on this fact, I decided to record myself playing two piano pieces. The first one being song without word by felix Mendelssohn and Porco ross by Joe Hisaishi. The first piece is rather sombre slow piece and the second is a more upbeat cheerful jazz piece.

My results were rather interesting...
When i viewed the somber piece in the frequency domain amongst other songs with a similar theme, i noticied that song that are precieved as sad or sombre to the huan ear have lower harmonics in the freqency domain thus giving it its sad attribute

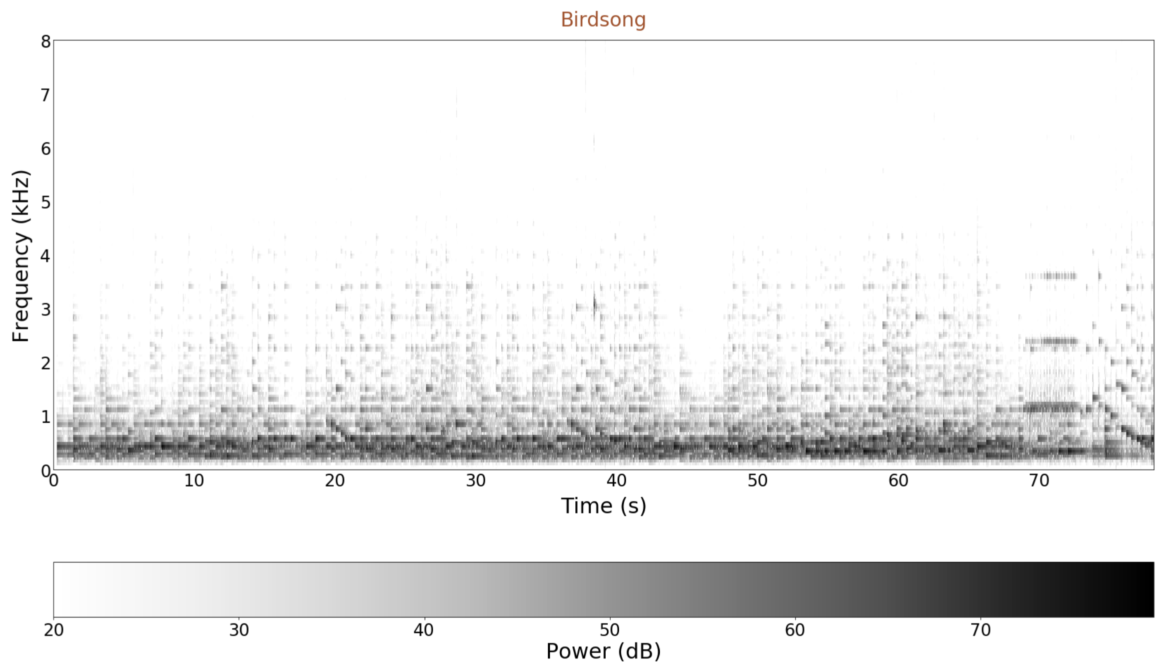
However songs with a more up-beat tone had higher harmonics and this translates as to why the sound is happy and pleasing to the human ear

Piano Song number 1: Song without Words by felix Mendelssohn

In [5]:

```
# -*- coding: utf-8 -*-
"""
Created on Thu Feb  6 13:37:19 2020

@author: G00348898
"""

# -*- coding: utf-8 -*-
"""
Freesounds:
Nightingale near Cologne https://freesound.org/people/reinsamba/sounds/14854/
Blackbird in the morning: https://freesound.org/people/Sesom42/sounds/431911/
Skylark over UK cornfield https://freesound.org/people/squashy555/sounds/244357/
Xeno-Canto:
https://www.xeno-canto.org/species/Hirundo-rustica
https://www.xeno-canto.org/species/Cuculus-canorus

@author: Michelle Lynch
"""
import numpy as np
import matplotlib.pyplot as plt
from scipy import signal
from scipy.io import wavfile

# Read in original audio & average channels, scipy works well with 16-bit depth wav fil
es
#rate, audio = wavfile.read('14854__reinsamba__nightingale-song.wav')
#rate, audio = wavfile.read('../Audio/Birdsong/431911__sesom42__blackbird-in-the-mornin
g.wav') # Mono
#rate, audio = wavfile.read('../Audio/Birdsong/244357__squashy555__skylark.wav')
rate, audio = wavfile.read('sadSong.wav')

audio = np.mean(audio, axis=1)
N = audio.shape[0]
#N = audio.size                # comment in for mono & comment out previous 2 lines
L = N / rate
print(f'Audio length: {L:.2f} seconds')

# Spectrogram calculation using SciPy
freqs_sp, times_sp, spec_sp = signal.spectrogram(audio, fs=rate, window='hanning',
                                                  nperseg=1024, noverlap=1024-100,
                                                  detrend=False, scaling='spectrum')
spec_db_sp = 10*np.log10(spec_sp)

# Plot original audio in time-domain
def plot_time_domain():
    fig = plt.figure(figsize=(14,12))
    ax = fig.add_subplot(111)
    ax.plot(np.arange(N) / rate, audio)
    ax.set_xlabel('Time (s)', fontsize=20, labelpad=10)
    ax.set_ylabel('Amplitude', fontsize=20, labelpad=10);

# Plot spectrogram, time & frequency-domain
def plot_spec_scipy():
    fig, ax = plt.subplots(figsize=(28,16))
    im = ax.pcolormesh(times_sp, freqs_sp/1000, spec_db_sp, vmax=spec_db_sp.max(), vmin
=20, cmap=plt.cm.Greys)
    cb = fig.colorbar(im, ax=ax, orientation="horizontal")
    ax.set_ylabel('Frequency (kHz)', fontsize=30, labelpad=10)
    ax.set_xlabel('Time (s)', fontsize=30, labelpad=10);
    ax.set_title('Birdsong', fontsize=28, pad=20, color='sienna');
    ax.set_ylim(0, 8) # Change maximum frequency range to suit audio
    ax.tick_params(axis='both', which='both', labelsize=24, length=0)
    cb.set_label('Power (dB)', fontsize=30)
    cb.ax.tick_params(labelsize=24)

plot_spec_scipy();
```

```
Audio length: 78.16 seconds
```

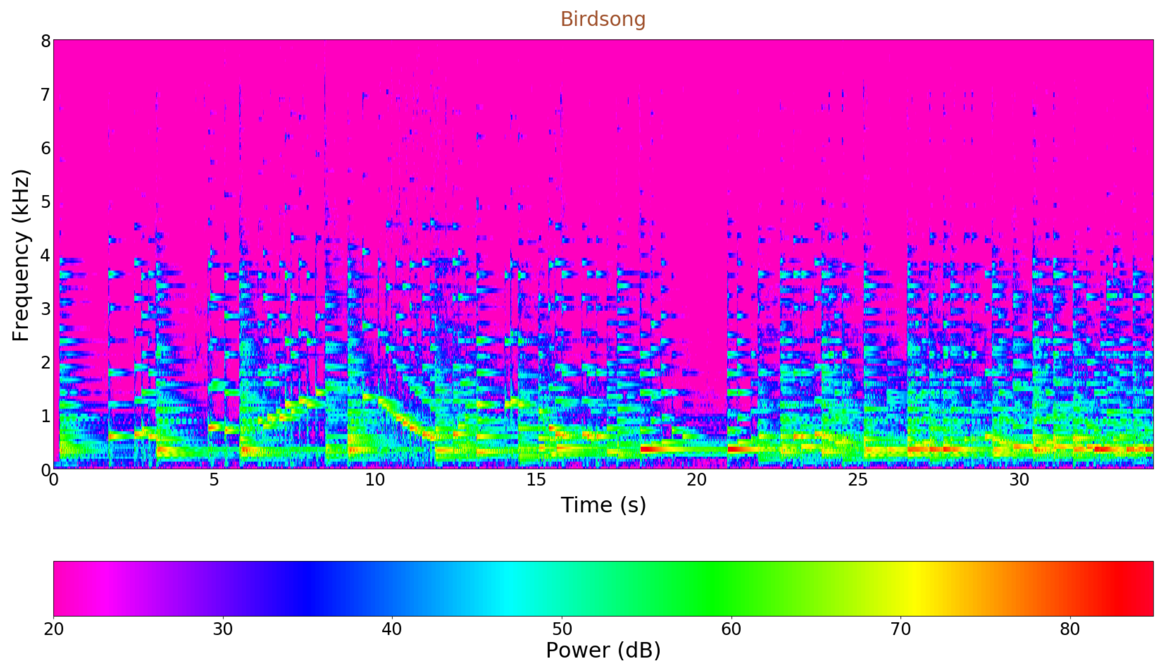


Piano Song number 2: Porco Rosso by Joe Hiashi

In [26]:

```
# -*- coding: utf-8 -*-
"""
Created on Thu Feb  6 13:37:19 2020

@author: G00348898
"""

# -*- coding: utf-8 -*-
"""
Freesounds:
Nightingale near Cologne https://freesound.org/people/reinsamba/sounds/14854/
Blackbird in the morning: https://freesound.org/people/Sesom42/sounds/431911/
Skylark over UK cornfield https://freesound.org/people/squashy555/sounds/244357/
Xeno-Canto:
https://www.xeno-canto.org/species/Hirundo-rustica
https://www.xeno-canto.org/species/Cuculus-canorus

@author: Michelle Lynch
"""
import numpy as np
import matplotlib.pyplot as plt
from scipy import signal
from scipy.io import wavfile

# Read in original audio & average channels, scipy works well with 16-bit depth wav fil
es
#rate, audio = wavfile.read('14854__reinsamba__nightingale-song.wav')
#rate, audio = wavfile.read('../Audio/Birdsong/431911__sesom42__blackbird-in-the-mornin
g.wav') # Mono
#rate, audio = wavfile.read('../Audio/Birdsong/244357__squashy555__skylark.wav')
rate, audio = wavfile.read('happySong.wav')

audio = np.mean(audio, axis=1)
N = audio.shape[0]
#N = audio.size                # comment in for mono & comment out previous 2 lines
L = N / rate
print(f'Audio length: {L:.2f} seconds')

# Spectrogram calculation using SciPy
freqs_sp, times_sp, spec_sp = signal.spectrogram(audio, fs=rate, window='hanning',
                                                 nperseg=1024, noverlap=1024-100,
                                                 detrend=False, scaling='spectrum')
spec_db_sp = 10*np.log10(spec_sp)

# Plot original audio in time-domain
def plot_time_domain():
    fig = plt.figure(figsize=(14,12))
    ax = fig.add_subplot(111)
    ax.plot(np.arange(N) / rate, audio)
    ax.set_xlabel('Time (s)', fontsize=20, labelpad=10)
    ax.set_ylabel('Amplitude', fontsize=20, labelpad=10);

# Plot spectrogram, time & frequency-domain
def plot_spec_scipy():
    fig, ax = plt.subplots(figsize=(28,16))
    im = ax.pcolormesh(times_sp, freqs_sp/1000, spec_db_sp, vmax=spec_db_sp.max(), vmin
=20, cmap="gist_rainbow_r")
    cb = fig.colorbar(im, ax=ax, orientation="horizontal")
    ax.set_ylabel('Frequency (kHz)', fontsize=30, labelpad=10)
    ax.set_xlabel('Time (s)', fontsize=30, labelpad=10);
    ax.set_title('Birdsong', fontsize=28, pad=20, color='sienna');
    ax.set_ylim(0, 8) # Change maximum frequency range to suit audio
    ax.tick_params(axis='both', which='both', labelsize=24, length=0)
    cb.set_label('Power (dB)', fontsize=30)
    cb.ax.tick_params(labelsize=24)

plot_spec_scipy();
```

```
Audio length: 34.20 seconds
```



Conclusion and what I have learnt

Before Starting this report, i would say that I started off with very basic knowledge about Fourier and it's applications and i am happy about what i could achieve with this research and of analysis that i made learning about fourier and how it is applied to image processing and sound engineering and know i can say that i have developed a better understanding of the fourier tranform and why it is used and the benefits that if offers to different apllication of the world eg cancer development ,license plate recognititon, images image compression etc.

References

- 1
  ref: [1]"Image Transforms - Fourier Transform", Homepages.inf.ed.ac.uk, 2003. [Online]. Available: https://homepages.inf.ed.ac.uk/rbf/HIPR2/fourier.htm. [Accessed: 13- Mar- 2020].

- 2

  ref: [2]"Fourier Transform — OpenCV-Python Tutorials 1 documentation", Opencv-python-tutroals.readthedocs.io, 2013. [Online]. Available: https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_transforms/py_fourier_transform/py_fourier_tı transform. [Accessed: 13- Mar- 2020].
- 3

  ref:[3]S. S, "Fourier transform infrared spectroscopy in cancer detection. - PubMed - NCBI", Ncbi.nlm.nih.gov, 2005. [Online]. Available: https://www.ncbi.nlm.nih.gov/pubmed/16556041. [Accessed: 13- Mar- 2020].
- 4

  ref: [4]"Characterization of ovarian cancer cells and tissues", biomedcentral.com, 2018. [Online]. Available: https://ovarianresearch.biomedcentral.com/articles/10.1186/s13048-018-0434-8. [Accessed: 13- Mar- 2020]..

```
In [ ]:
```