Guillaume BELDILMI & Guillaume GAUGUET

Outils mathématiques pour l'informatique : Sources

`ft.py`

```python
import numpy as np

def ft(X):
    """Fourier Transform (FT)"""
    N = len(X)
    if N == 1:
        return X
    else:
        x = np.zeros(N, dtype=complex)
        for k in range(N):
            x[k] = np.sum(X*np.exp(-2j*np.pi*k*np.arange(N)/N))
        return x

def ift(X):
    """Inverse Fourier Transform (IFT)"""
    N = len(X)
    if N == 1:
        return X
    else:
        x = np.zeros(N, dtype=complex)
        for k in range(N):
            x[k] = np.sum(X*np.exp(2j*np.pi*k*np.arange(N)/N))
        return x/N

def fft(X):
    """Fast Fourier Transform (FFT)"""
    N = len(X)
    if N == 1:
        return X
    else:
        x = np.zeros(N, dtype=complex)
        xe = fft(X[::2])
        xo = fft(X[1::2])
        for k in range(N//2):
            p = xe[k]
            q = np.exp(-2j*np.pi*k/N) * xo[k]
```

```
            x[k] = p+q
            x[k+(N//2)] = p-q
        return x

def ifft(X):
    """Inverse Fast Fourier Transform (IFFT)"""
    N = len(X)
    if N == 1:
        return X
    else:
        x = np.zeros(N, dtype=complex)
        xe = ifft(X[::2])
        xo = ifft(X[1::2])
        for k in range(N//2):
            p = xe[k]
            q = np.exp(2j*np.pi*k/N) * xo[k]
            x[k] = p+q
            x[k+(N//2)] = p-q
        return x/N

def fft2d(X):
    """2D Fast Fourier Transform (FFT)"""
    N, M = X.shape
    x = np.array([fft(X[i]) for i in range(N)])
    x = np.array([fft(x[:,i]) for i in range(M)]).T
    return x

def ifft2d(X):
    """2D Inverse Fast Fourier Transform (FFT)"""
    N, M = X.shape
    x = np.array([ifft(X[i]) for i in range(N)])
    x = np.array([ifft(x[:,i]) for i in range(M)]).T
    return x
```

cmp.py

```
import numpy as np
import time as bb
import matplotlib.pyplot as plt
from ft import ft, ift, fft, ifft

def t(S, D = -1):
    """Test FT on 1D signal and return the time it took"""
    N = len(S)
    r = [-1, -1]
    td = bb.time_ns()
```

```
    ss = ft(S)
    tf = bb.time_ns()
    r[0] = (tf - td) / 1000000000.
    if D != -1:
        for i in range(N):
            if i > D:
                ss[i] = 0
    print("FT: ")
    for i in range(N):
        print(i, "...", ss[i])
    td = bb.time_ns()
    sss = ift(ss)
    tf = bb.time_ns()
    r[1] = (tf - td) / 1000000000.
    print("IFT: ")
    for i in range(N):
        print(i, "...", sss[i])
    plt.subplot(2, 3, 2)
    plt.stem(np.arange(N), np.abs(ss))
    plt.title("FT")
    plt.subplot(2, 3, 3)
    plt.stem(np.arange(N), np.abs(sss))
    plt.title("IFT")
    return r

def tf(S, D = -1):
    """Test FFT on 1D signal and return the time it took"""
    N = len(S)
    r = [-1, -1]
    td = bb.time_ns()
    ss = fft(S)
    tf = bb.time_ns()
    r[0] = (tf - td) / 1000000000.
    if D != -1:
        for i in range(N):
            if i > D:
                ss[i] = 0
    print("FFT: ")
    for i in range(N):
        print(i, "...", ss[i])
    td = bb.time_ns()
    sss = ifft(ss)
    tf = bb.time_ns()
    r[1] = (tf - td) / 1000000000.
    print("IFFT: ")
    for i in range(N):
        print(i, "...", sss[i])
    plt.subplot(2, 3, 5)
```

```
    plt.stem(np.arange(N), np.abs(ss))
    plt.title("FFT")
    plt.subplot(2, 3, 6)
    plt.stem(np.arange(N), np.abs(sss))
    plt.title("IFFT")
    return r

def cmp(N, D = -1):
    """Compare the time it takes to compute the FT and the FFT"""
    s = np.random.rand(N)
    print("Original signal: ", s)
    plt.figure("Test 1D deformed" if D != -1 else "Test 1D")
    plt.subplot(2, 3, 1)
    plt.stem(np.arange(N), np.abs(s))
    plt.title("Original signal")
    ttf = t(s, D)
    ttff = tf(s, D)
    print("FT: ", ttf[0], "s\t", "IFT: ", ttf[1], "s")
    print("FFT: ", ttff[0], "s\t", "IFFT: ", ttff[1], "s")
    plt.show()

if __name__ == "__main__":
    n = 24
    d = n // 2
    cmp(n)
    cmp(n, d)
```

dd.py

```
import numpy as np
import matplotlib.pyplot as plt
from ft import fft2d, ifft2d

def dd(N, D = -1):
    """Test fft on a random 2D signal"""
    s = np.random.rand(N, N)
    ss = fft2d(s)
    if D != -1:
        for i in range(N):
            for j in range(N):
                if i+j > D:
                    ss[i, j] = 0
    sss = ifft2d(ss)
    ss[0, 0] = 0
    plt.figure("Test 2D deformed" if D != -1 else "Test 2D")
    plt.subplot(1, 3, 1)
```

```
    plt.imshow(s)
    plt.title("Original signal")
    plt.subplot(1, 3, 2)
    plt.imshow(np.abs(ss))
    plt.title("FFT (without the first pixel)")
    plt.subplot(1, 3, 3)
    plt.imshow(np.abs(sss))
    plt.title("IFFT")
    plt.show()

if __name__ == "__main__":
    n = 8
    d = 2
    dd(n)
    dd(n, d)
```

gr.py

```
import numpy as np
import matplotlib.pyplot as plt
from ft import fft2d, ifft2d

def gr(P, N = -1):
    """Test fft on custom image converted to grayscale"""
    img = plt.imread(P)
    i = np.dot(img[...,:3], [0.299, 0.587, 0.114])
    ii = fft2d(i)
    if N != -1:
        for j in range(i.shape[0]):
            for k in range(i.shape[1]):
                if j+k > N:
                    ii[j, k] = 0
    iii = ifft2d(ii)
    ii[0, 0] = 0
    plt.figure("Test gray deformed" if N != -1 else "Test gray")
    plt.subplot(1, 3, 1)
    plt.imshow(i)
    plt.title("Original image")
    plt.subplot(1, 3, 2)
    plt.imshow(np.abs(ii))
    plt.title("FFT (without the first pixel)")
    plt.subplot(1, 3, 3)
    plt.imshow(np.abs(iii))
    plt.title("IFFT")
    plt.show()
```

```
if __name__ == "__main__":
    gr("img/lenna.png")
    gr("img/lenna.png", 100)
```

rgb.py

```
import numpy as np
import matplotlib.pyplot as plt
from ft import fft2d, ifft2d

def rgb(P, N = -1):
    """Test fft on each channel of a custom image"""
    img = plt.imread(P)
    r = img[:, :, 0]
    g = img[:, :, 1]
    b = img[:, :, 2]
    rr = np.fft.fft2(r)
    gg = np.fft.fft2(g)
    bb = np.fft.fft2(b)
    if N != -1:
        for i in range(img.shape[0]):
            for j in range(img.shape[1]):
                if i+j > N:
                    rr[i, j] = 0
                    gg[i, j] = 0
                    bb[i, j] = 0
    rrr = np.fft.ifft2(rr)
    ggg = np.fft.ifft2(gg)
    bbb = np.fft.ifft2(bb)
    nimg = np.zeros((img.shape[0], img.shape[1], 3))
    nimg[:, :, 0] = abs(rrr)
    nimg[:, :, 1] = abs(ggg)
    nimg[:, :, 2] = abs(bbb)
    m = np.max(nimg)
    nimg /= m
    rr[0, 0] = 0
    gg[0, 0] = 0
    bb[0, 0] = 0
    plt.figure("Test rgb deformed" if N != -1 else "Test rgb")
    plt.subplot(3, 3, 1)
    plt.imshow(img)
    plt.title("Original image")
    plt.subplot(3, 3, 4)
    plt.imshow(np.abs(rr))
    plt.title("FFT (R) (without the first pixel)")
    plt.subplot(3, 3, 5)
```

```
    plt.imshow(np.abs(gg))
    plt.title("FFT (G) (without the first pixel)")
    plt.subplot(3, 3, 6)
    plt.imshow(np.abs(bb))
    plt.title("FFT (B) (without the first pixel)")
    plt.subplot(3, 3, 7)
    plt.imshow(np.abs(rrr))
    plt.title("IFFT (R)")
    plt.subplot(3, 3, 8)
    plt.imshow(np.abs(ggg))
    plt.title("IFFT (G)")
    plt.subplot(3, 3, 9)
    plt.imshow(np.abs(bbb))
    plt.title("IFFT (B)")
    plt.subplot(3, 3, 3)
    plt.imshow(nimg)
    plt.title("Reconstructed image")
    plt.show()

if __name__ == "__main__":
    rgb("img/jl.png")
    rgb("img/jl.png", 100)
```