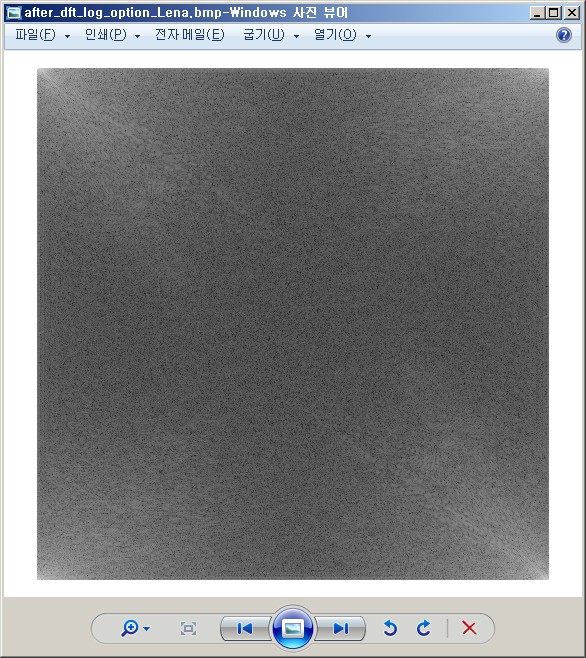
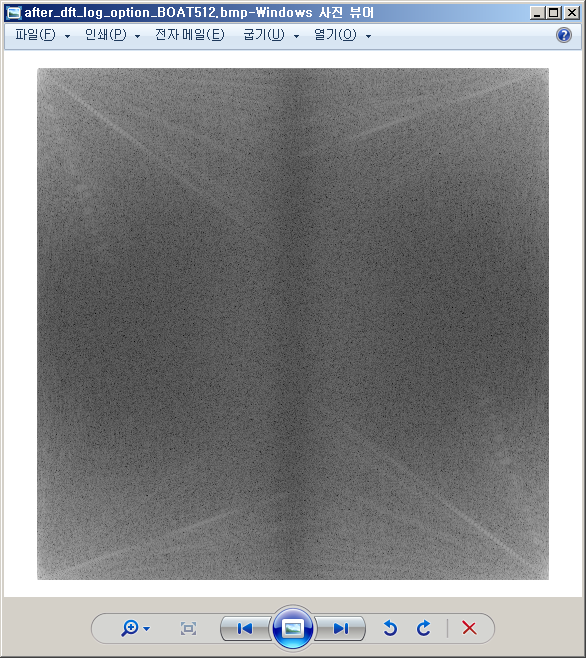
**영상처리 HW5**

**2015253039 권진우**

**1. boat 영상에 대해서 수행 (**BOAT512와 Lena, 총 2개의 이미지에 대해 수행하였습니다.**)**

**(a) 512x512 Forward DFT를 수행하고 모니터에 주파수 스펙트럼을 적절한 스케일로 표시하여 쉽게 관찰할 수 있도록 한다.**

**- BOAT512 DFT(FFT) 수행 후 Frequency - Lena DFT(FFT) 수행 후 Frequency**



- > Log Scaling (dst = |1 + |src||)을 사용하여서 주파수 스펙트럼을 표시하여 관찰 할 수 있도록 하고 파일로 저장하였습니다.

**(b) 512x512 역DFT를 수행하여 재구성된 영상을 얻는다. MSE를 계산하라.**

**- BOAT512 Inverse DFT 수행 후 Time Domain - Lena Inverse DFT 수행 후 Time Domain**



**DFT 영상 MSE**

 <- **BOAT512** 영상에 대한 Mean Square Error

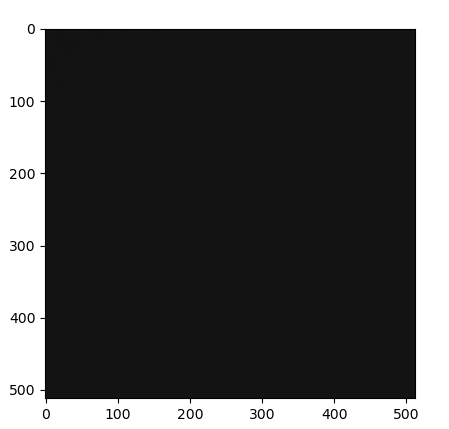
 <- **Lena** 영상에 대한 Mean Square Error

- 위와 같이 10^-28 의 아주 작은 MSE 값이 나왔으며 실제로 원 영상과 Inverse-DFT 후의 영상을 비교해보면 차이가 없었으며 단순히 Frequency Domain 으로 변환 후 재구성한 것이기에 당연한 결과입니다.

다만, Python 언어로 작성하여서 double형을 사용하는 C언어보다는 float형을 사용하기에 약간 MSE가 높을 것으로 예상 됩니다. 하지만 아주 작은 MSE 값이기 때문에 거의 차이가 없다고 봐도 무방합니다.

**2. boat 영상에 대해서 수행**

**(a) 512x512 포워드 DCT를 수행하고 모니터에 주파수 스펙트럼을 적절한 스케일로 표시하여 쉽게 관찰할 수 있도록 한다.**

**<- Energy Compaction이 좌상단에 몰렸다.**

**(b) 512x512 역DCT를 수행하여 재구성된 영상을 얻는다. MSE를 계산하라.**

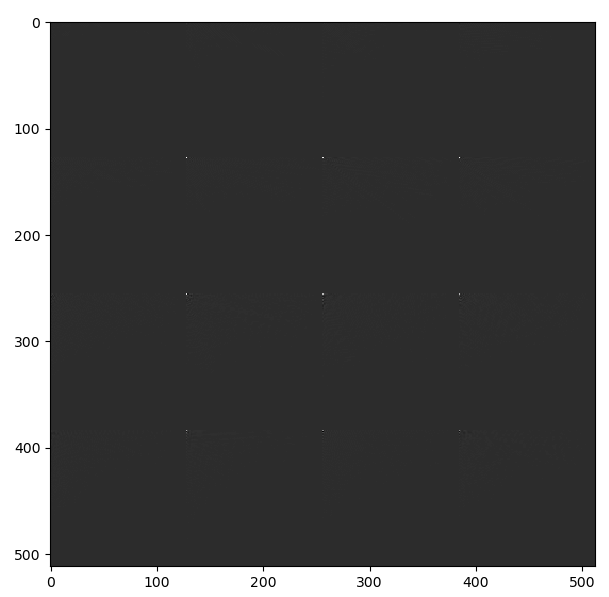
** <- MSE**

<- Inverse-DCT가 수행되어 재구성 된 BOAT512

2-D DCT는 너무 시간이 오래 걸리기 때문에 1-D DCT를 구현하여 사용

**(c) (a)와 (b)를 128x128 DCT로 수행해라**

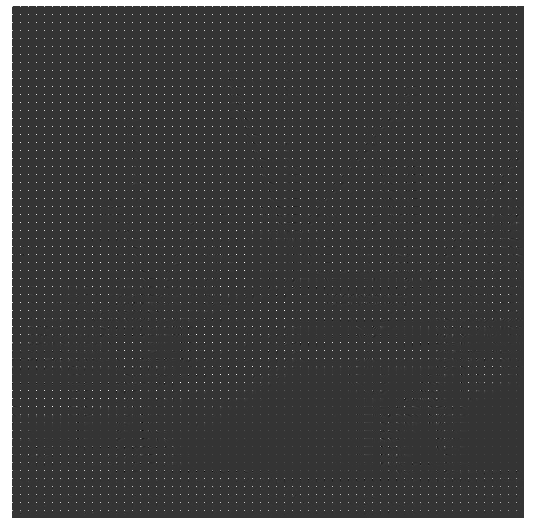
 **<- MSE**



**-** 128x128 Block Size를 사용하여서 DCT를 수행하고 1-D DCT를 사용하였으며 Inverse-DCT의 결과 원 영상과 차이를 느낄 수 없게 위와 같은 결과가 나왔습니다.

**(d) (a)와 (b)를 8x8 DCT로 수행해라**

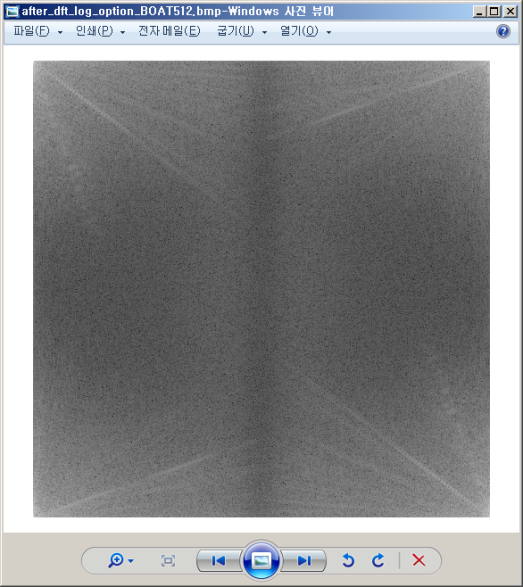
** <- MSE**

****

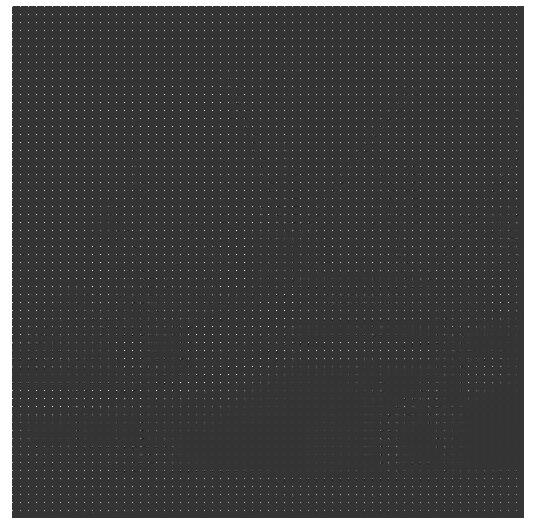
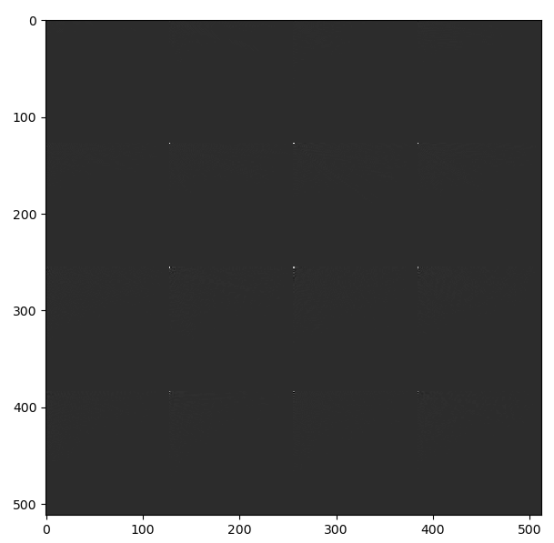
- 8x8 Block Size를 사용하여서DCT를 수행하였고 1-D DCT를 사용하여 기존 2-D DCT보다 빠르게 연산을 수행할 수 있도록 구현하였고 Block Size가 작을수록 DCT의 속도가 빨랐습니다. 또한 Inverse-DCT또한 수행 속도가 빨랐습니다.

**3. 위의 과제에 따라 DFT와 DCT의 성능을 비교한다. 또한 DCT 블록 크기가 변환 성능에 미치는 영향을 계산 부하 및 재구성된 영상 품질 측면에서 분석하십시오.**

DFT와 DCT에 있어서 DFT는 복소수(허수)를 사용함으로 인하여서 계산 복잡도가 높아지고 그에 따라 구현하기가 더욱 까다로우며 다루기가 힘들어집니다. FFT 기법을 사용하여 빠르게 DFT를 수행을 할 수 있었지만 DCT(8x8 DCT기준)보다 느리며 또한 결과 영상에서 Energy Compaction이 되지 못하고 4방향의 꼭지점으로 Energy가 모이게 되어 FFT 이후에도 영상을 다룰 때 번거롭게 됩니다. 복구 후에는 원영상과 비교하여 차이가 거의 없었습니다.

<- DFT 결과 예시 : Energy Compaction이 좋지 못하다.

반면 DCT는 Cosine 함수를 사용한 기저함수를 사용하여 계산복잡도 측면에 있어서 유리하며 2-D DCT기법이 아닌 계산복잡도가 더 낮은 1-D DCT를 사용하면 더욱 빠르게 DCT를 수행 할 수 있었습니다. 또한 DCT Block의 Size를 필요에 맞게 조절하여 DCT를 수행할 수 있었는데 DCT Block size가 작을수록 수행 시간이 빨랐습니다. 또한 수행 후 영상을 보면 아래와 같이 Energy Compaction이 매우 좋기 때문에 Frequency Domain에 있어서 고주파 부분을 제거하여 압축효과를 내기가 쉽고 복구 후에도 원영상과 비교 했을 때 차이점을 느낄 수 없었습니다.



<- 128x128, 8x8 DCT 결과 예시

-> Energy Compaction이 좋다.

DCT의 블록 Size에 따라서도 변환 시간에 대한 성능 차와 재구성 영상 품질에 있어서 차이가 발생하게 됩니다.

예를 들면, 8x8 block을 사용한다면 512x512영상에 대한 DCT에 있어서 계산 복잡도는 다음과 같습니다.

=> 8x8x8 x 64x64 x 2 = 2,097,152

- block size에 따른 계산 복잡도 (**512x512** 영상, **1-D DCT** compute 기준)

|  |  |  |  |
| --- | --- | --- | --- |
| **Block Size** | **계산 복잡도를 구하는 식** | **계산 복잡도(계산 수)** | **1-D 8x8기준 배율** |
| **4x4** | **4x4x4x128x128x2** | **2,097,152** | **0.5 배** |
| **8x8** | **8x8x8x64x64x2** | **4,194,304** | ***1.0* 배** |
| **32x32** | **32x32x32x16x16x2** | **16,777,216** | **4.0 배** |
| **64x64** | **64x64x64x8x8x2** | **33,554,432** | **8.0 배** |
| **128x128** | **128x128x128x4x4x2** | **67,108,864** | **16.0 배** |
| **512x512** | **512x512x512x1x1x2** | **268,435,456** | **64.0 배** |

- 8x8 block size에 대한 식의 의미는 다음과 같습니다.

(1칸 계산 : 8, 1줄 계산 : 8x8, 1Block 계산 : 8x8x8, 512x512영상에 대한 block 수 : 64x64, X/Y : x2)

2-D dimension의 경우는 1-D에 비하여 복잡도가 높으며 계산복잡도는 아래와 같습니다.

- block size에 따른 계산 복잡도 (**512x512** 영상**, 2-D DCT** 기준)

|  |  |  |  |
| --- | --- | --- | --- |
| **Block Size** | **계산 복잡도를 구하는 식** | **계산 복잡도(계산 수)** | **1-D 8x8기준 배율** |
| **8x8** | **8x8x8x8x64x64** | **16,777,216** | **4.0 배** |
| **32x32** | **32x32x32x32x16x16** | **268,435,456** | **64.0 배** |
| **512x512** | **512x512x512x512x1x1** | **68,719,476,736** | **16,384.0 배** |

2-D DCT의 계산 복잡도의 경우 하나의 점에 대해서 전체의 Block 픽셀이 계산복잡도에 영향을 미치기 때문에 1-D DCT보다 8x8 기준 4배, 32x32 기준 16배의 계산 복잡도가 차이가 나게 되고 Block Size가 커질수록 그 차이는 기하급수적으로 벌어지게 된다. 따라서, 1-D DCT를 주로 사용하며 변환시간 속도적인 측면에 있어서 Block의 Size가 작은 것이 절대적으로 유리하기 때문에 적은 Block 크기를 사용하여 DCT를 수행합니다.

**\* 1-D DCT와 2-D DCT에 있어서 재구성 품질 면에서는 차이가 없이 같습니다. 따라서, 1-D DCT가 좋습니다.**

재구성 품질에 있어서 Block Size가 큰 것이 작은 것 보다 유리할 줄 알았지만 아래 결과와 같이 큰 Block Size일수록 Frequency Domain으로 변환 후 재구성 계산에 있어서 손실되는 값이 상대적으로 컸습니다. Block Size가 작을수록 상대적으로 손실이 작은 것을 알 수 있습니다. 따라서 재구성 품질 면에서도 작은 Block Size 의 DCT 기법이 조금 더 우세합니다. 하지만 Block Size가 너무 작게 되면 단순 Frequency Domain으로 변환 후 재구성하는 것이 아닌 Frequency domain 고주파 영역에 대해서 삭제를 함으로써 압축을 하고 재구성을 하는 경우에서는 Block Size가 너무 작게 되면 손실이 오히려 더 커질 것으로 예상됩니다. 따라서 Block Size는 적절한 Size가 좋으며 Block이 너무 크면 변환 시간에 있어서 성능이 나빠지게 되고, Block이 너무 작으면 Frequency Domain에서 영상을 압축시키거나 다룰 때 손실되는 값이 커지고 다루기 번거로울 수 있습니다.

**DCT에서 block size에 따른 MSE는 다음과 같습니다.**











\* e-23 = 10^-23

사람의 눈으로는 원 영상과 재구성된 영상의 차이를 느낄 수 없었습니다. 따라서 품질에서는 큰 차이를 느낄 수 없으며 변환 속도 측면에서 훨씬 우월한 작은 size의 Block을 사용합니다.

DFT와 DCT의 기본 원리는 영상(Time Domain)을 기저함수를 사용하여 Frequency(주파수) Domain으로 변환 후 고주파 영역에 해당하는 부분을 삭제함으로써 영상의 압축을 꽤 할 수 있고 또한 네트워크를 통한 전송에 있어서도 중요한 Frequency 부분을 안정적으로 보내도록 함으로써 손실을 줄이며 Energy가 적은 Frequency 부분이 손실 되더라도 영상을 재구성 하였을 때, 사람의 눈에는 거의 차이가 없을 정도로 복구가 가능하도록 합니다. 실시간 영상 전송에서도 사용이 되므로 빠르고 쉽게 다룰 수 있는 Block 단위로 나눈 DCT를 사용하게 되는 것 같습니다.

이상으로 보고서를 마치며 **Source Code는 상당히 긴 관계로 보고서에 싣지는 못하였으며 대신 Source Code 주석을 상세하게 달아 놓았습니다.**

**Source Code 실행 방법 및 개발 환경 & 언어, 사용된 패키지**

- 파일에는 agent.py, perform.py, .raw 영상 이미지, DCT,DFT 수행된 이미지 등이 들어있습니다.

- 프로그램 실행은 **agent.py에서 컴파일을 수행하여 실행**시키면 됩니다.

- agent.py에서 실행, perform.py에서 구현

- Console 화면을 통해서 1~7번의 메뉴 선택을 통해서 프로그램을 수행하면 됩니다.

- BOAT512.raw이미지와 Lena.raw이미지 2개 중에 선택하여 실행하도록 만들었습니다.

- DFT, DCT 수행은 이미지 선택 후 실행하도록 하였습니다.

- DFT는 FFT기법을 사용하였습니다.

**- *\*\*\* 1-D DCT와 2-D DCT를 모두 구현*하여 선택하여 사용할 수 있습니다.**

**- 개발환경**

- OS : Windows7

- IDE : Pycharm 2019, Anaconda3, 그림판, Microsoft office Word

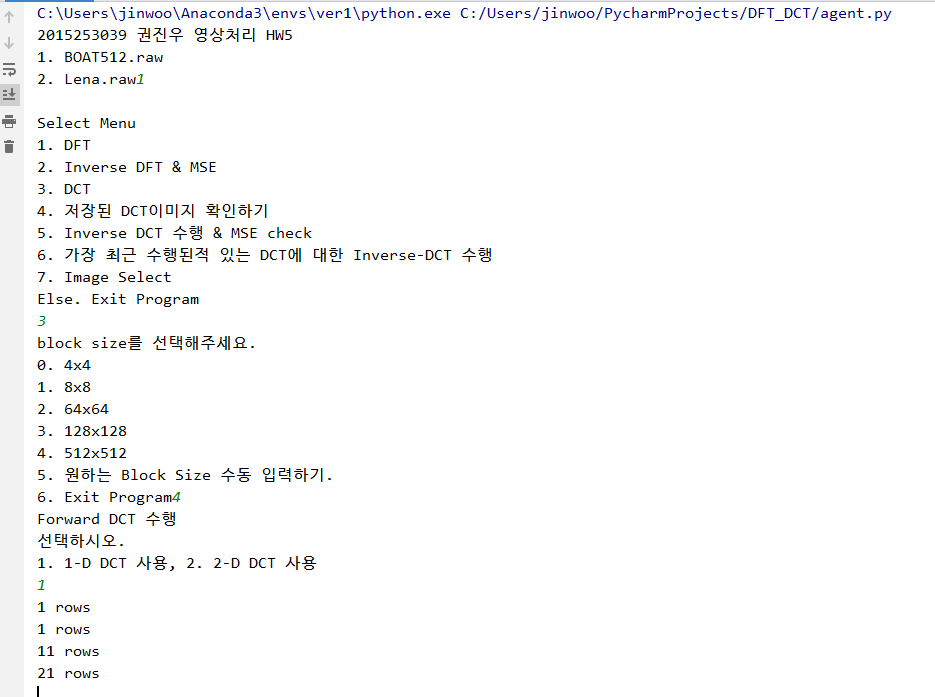
**- 개발 언어**

- Python 3.6 version

**- 사용된 package**

- Numpy, PIL, math, os, matplotlib

수행 예시



**읽어주셔서 감사합니다.**