Numerical Methods

HW02(interpolation)

담당교수 : 심동규 교수님

학 번 : 2019202050

성 명 : 이강현

Introduction

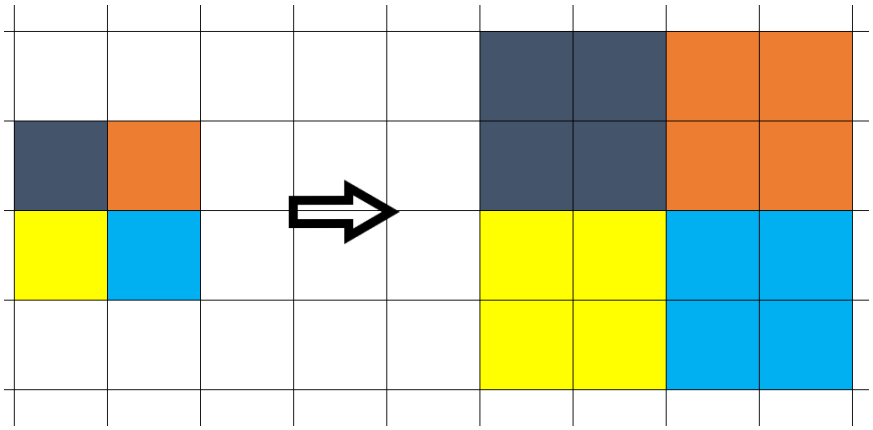
이번 과제에서는 Nearest Neighbor, Bilinear, Bicubic, Six-tab 4가지의 interpolation 방법을 구현해 보고 이미지를 upsampling하는데 구현한 interpolation을 적용하여 본다. 또 적용한 이미지의 PSNR을 측정하여 이미지의 품질을 어느정도 올렸는지를 확인해본다.

Interpolation Method

-Nearest Neighbor Interpolation-

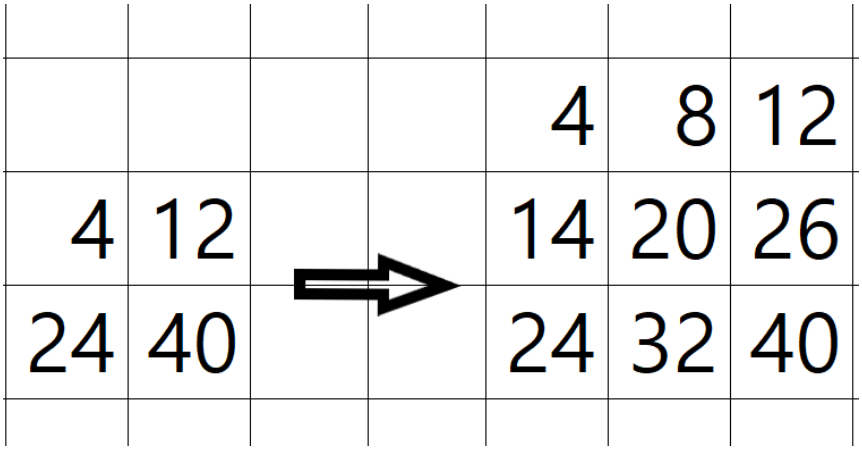
보간할 픽셀을 가장 가까운 픽셀을 이용하여 보간하는 방법

간단하여 계산량이 적고 빠른 알고리즘이지만 blockness가 발생하거나 이미지 품질이 낮아질 수 있다.



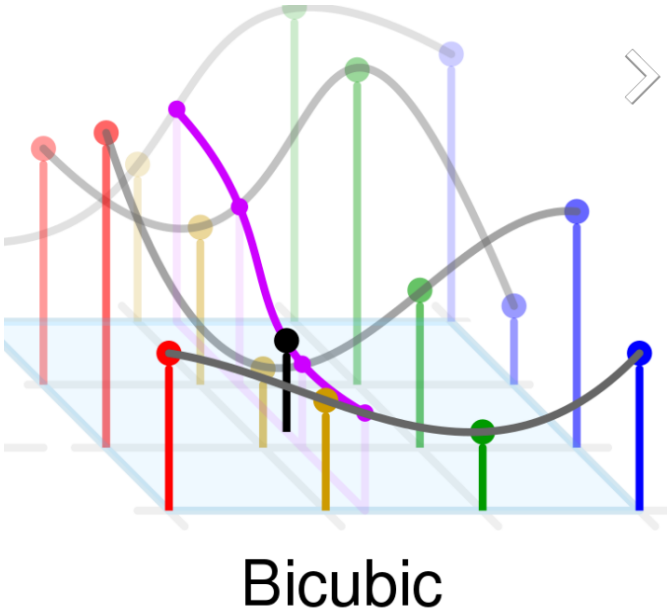
-Bilinear Interpolation-

인접한 4개의 픽셀들의 값들을 이용하여 거리비를 이용하여 선형보간하는 방법



-Bicubic Interpolation-

인접한 4개의 픽셀을 이용하여 비선형으로 보간하는 방법인 cubic interpolation을 x축 y축으로 확장하여 2차원으로 보간하는 방법. 점을 보간하기 위해 16개의 픽셀값을 이용한다.



-Six-tab Interpolation-

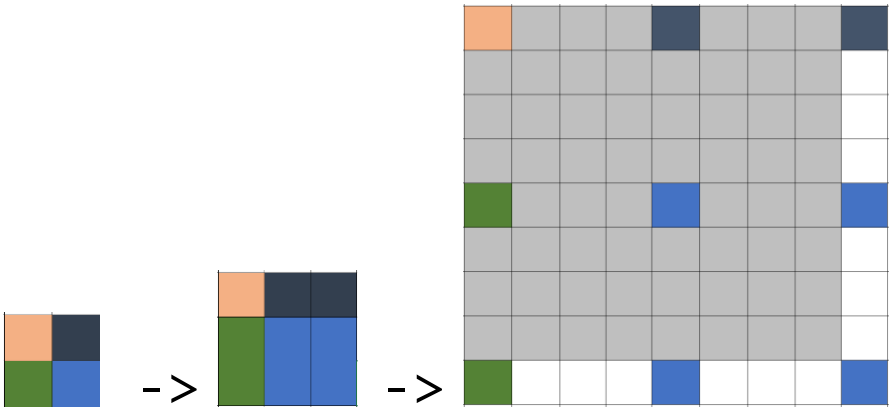
가로 6개의 픽셀과 필터를 곱하여 나온 값들을 연산하고 이를 통해 생성된 새로 6개의 값들과 필터를 곱하여 중앙값을 구하고 중앙값과 주어진 픽셀 사이의 값들은 평균을 내어 구하는 방법

Experiments

먼저 각 알고리즘을 구현한 방법에 대한 설명을 진행하고 그 이후 Barbara,Couple,Lena 세개의 사진의 PSNR과 PYUV를 이용한 사진 열람을 통해 성능을 비교해본다.

Nearest Neighbor Interpolation은 가장 간단한 알고리즘이었다. 이중 반복문을 이용해 행과 열을 순차적으로 접근할 수 있게 하고 512x512만큼 순회하면서 4로 나눈 몫이 같은 인덱스에는 같은 값들을 넣어주는 방식으로 구현하였다.

나머지 세 개의 interpolation방식을 설명하기 전 실시한 padding에 대해 설명한다.



위와 같이 원본사진을 가장 가까운 픽셀로 복사하고 확장시킨 후 알고리즘에 맞게 보간한 후 회색 음영된 부분만 추출하였다.

Bilinear Interpolation은 먼저 4개의 점으로 둘러쌓여야 그 내부의 픽셀들을 보간할 수 있기에 사진의 오른쪽과 아래부분은 추가적으로 하나의 픽셀이 더 있어야 보간이 가능했다. 따라서 가장 가까운 픽셀을 복사하는 방식으로 padding하여 129x129를 이용하여 513x513 이미지를 만들어내고 512x512만큼 뽑아내는 방법으로 구현하였다. 2개의 주어진 픽셀 사이의 3개의 픽셀중 가운데 값을 평균으로 채우고 그 양옆 값도 평균으로 채우는 것을 반복하였다. 가로 두 픽셀을 이용하여 사이에 있는 3개의 픽셀을 채우고 세로 두 픽셀을 이용하여 사이에 있는 3개의 픽셀을 채우고 나면 4개의 총 16개의 점으로 둘러 쌓인 내부 9개의 픽셀들은 주어진 픽셀 4개로 만들어낸 픽셀들을 이용하여 같은 방법으로 채웠다.

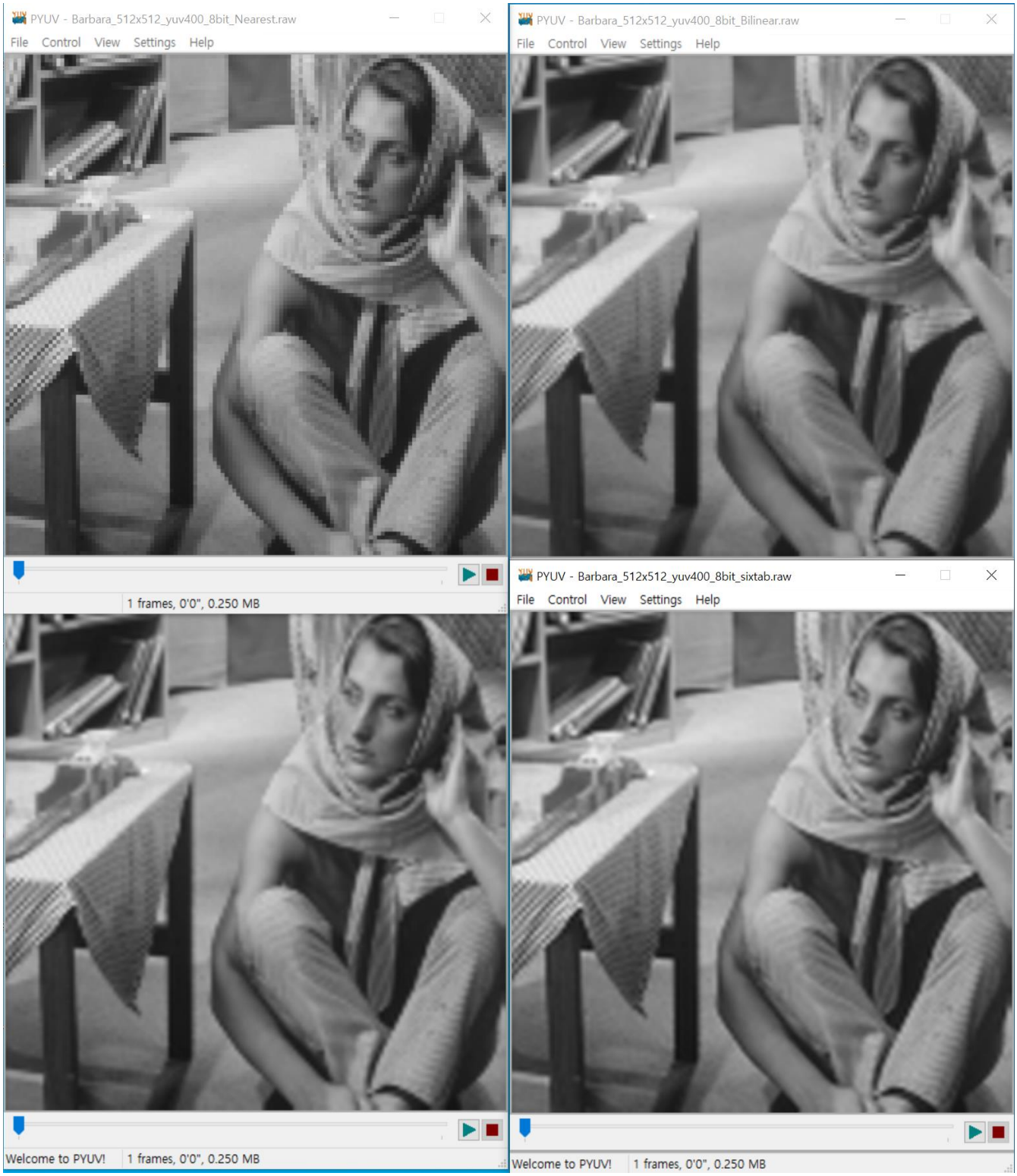
Bicubic Interpolation은 가로 4개의 픽셀들을 이용하여 연립방정식을 세우고 가우소 소거법을 이용하여 4개의 픽셀값들을 만족하는 삼차방정식의 계수를 구한 후 보간할 값들의 인덱스를 대입하여 값을 얻어내는 방식으로 구현했다. 이때 방정식의 x값은 -6,-2,2,6으로 픽셀이 떨어져 있는 크기만을 고려하여 고정하였고 이때 y값으로 해당 픽셀값을 넣어주어 방정식을 구했다. 이를 새로 4픽셀에도 적용한 후 만들어낸 값들을 이용하여 또 삼차방정식을 구하고 방정식에 대입하여 내부 픽셀 또한 채웠다. Bicubic은 4개의 점을 이용하여 보간하기에 좌,상에 패딩 1, 우,하에 패딩 2를 주어 원본 이미지를 131x131로 만들어 524x524사진으로 만들고 512x512만을 추출하였다.

Six-tab interpolation은 6개의 원소를 가진 필터를 가로,세로 6개의 픽셀에 적용하여 픽셀들의 가운데 픽셀을 채우고 가운데 픽셀들에 필터를 적용하여 중앙의 값을 얻어낸다 그 이후 중앙의 값과 채워진 픽셀들의 값들을 이용하여 평균을 내어 빈 부분을 채우는 방식으로 구현하였다. six-tab 방식을 총 6개의 픽셀들을 이용하므로 padding은 좌,상 2, 우,하에 3만큼 적용하여 133x133을 532x532로 만든 후 512x512를 추출하였다.

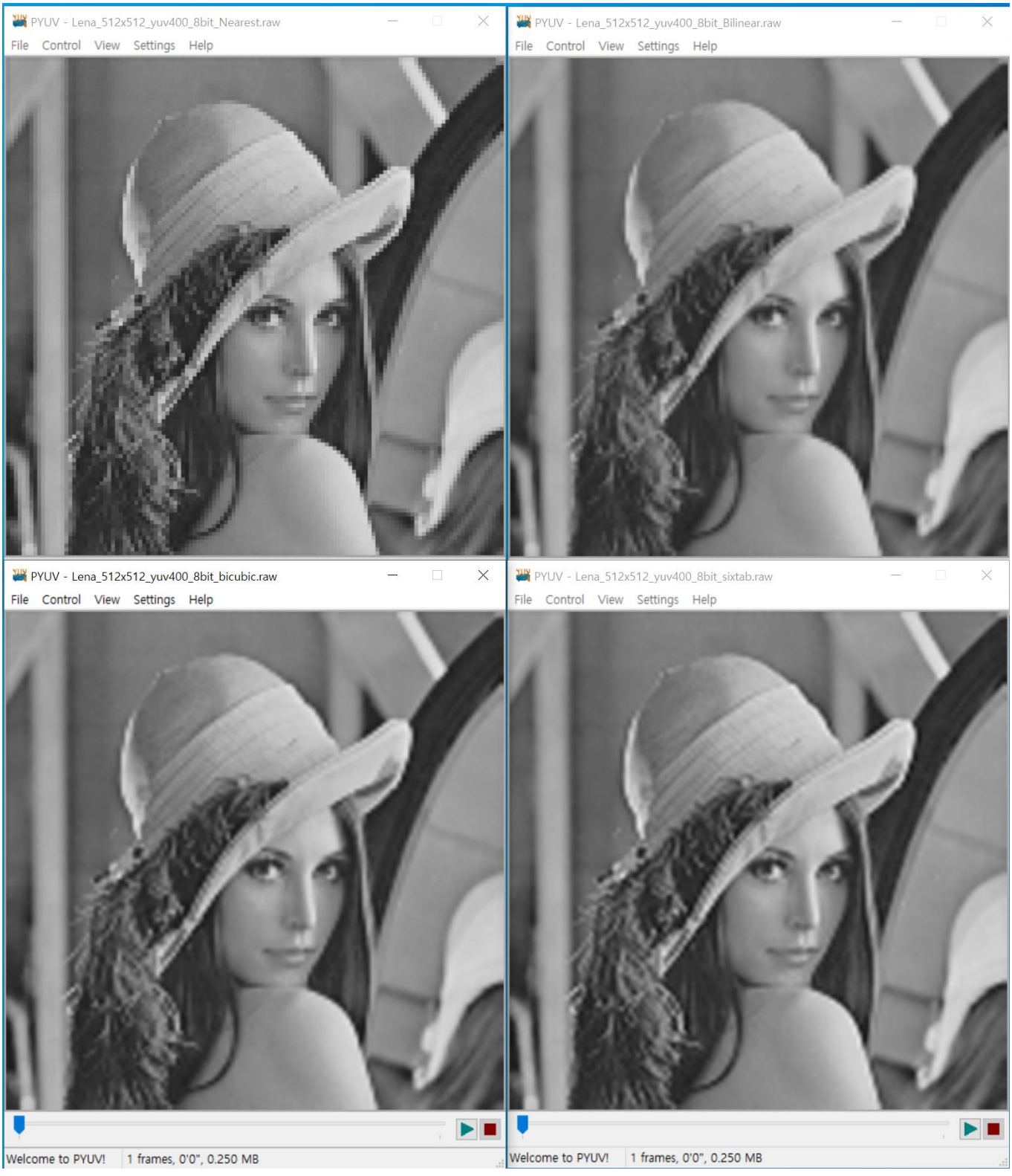
아래는 생성된 사진의 결과이다.

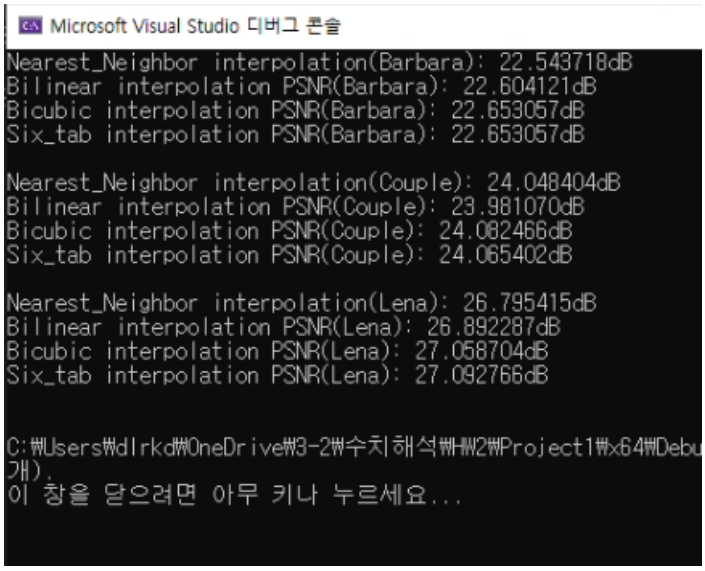
1	2
3	4

1.Nearest 2.Bilinear 3.Bicubic 4.Six-tab




PYUV - Couple_512x512_yuv400_8bit_Nearest.raw
File Control View Settings Help
PYUV - Couple_512x512_yuv400_8bit_Bilinear.raw
File Control View Settings Help
PYUV - Couple_512x512_yuv400_8bit_bicubic.raw
File Control View Settings Help
PYUV - Couple_512x512_yuv400_8bit_sixtab.raw
File Control View Settings Help
Welcome to PYUV!
1 frames, 0'0", 0.250 MB
Welcome to PYUV!
1 frames, 0'0", 0.250 MB


PYUV - Lena_512x512_yuv400_8bit_Nearest.raw
File Control View Settings Help
PYUV - Lena_512x512_yuv400_8bit_Bilinear.raw
File Control View Settings Help
PYUV - Lena_512x512_yuv400_8bit_bicubic.raw
File Control View Settings Help
Welcome to PYUV! 1 frames, 0'0", 0.250 MB
PYUV - Lena_512x512_yuv400_8bit_sixtab.raw
File Control View Settings Help
Welcome to PYUV! 1 frames, 0'0", 0.250 MB



눈으로 확인하고 판단한 것과 PSNR은 다르게 나왔다. 눈으로 보기에는 Nearest_Neighbor 방식이 많이 안 좋아 보여서 훨씬 낮게 나올 것 같았으나 PSNR은 각각의 알고리즘마다 큰 차이가 나지 않는 것 같았다. Bicubic과 Sixtab은 우열을 가리기 힘들었고 성능은 Nearest<Bilinear<Cubic=Six-tab으로 나왔다.

Conclusion

여러가지 interpolation 방식을 구현해보면서 사진을 upsampling 해보았는데 결과적으로는 확실히 하나의 가로 세로 4배를 interpolation 한다는게 극적인 효과를 보여주진 못하는 것 같다. 구현의 차이가 존재하겠지만 정보량이 많이 손실되어 그런 것 같다. 또 사진마다 똑같은 기법을 적용했으나 다른 PSNR을 가지는 것으로 보아 사진의 특성을 고려하여 interpolation의 구현을 진행하면 더 좋은 결과를 얻어낼 수도 있을 것 같았다. 또한 Bicubic을 구현함에 있어서 가우스 소거법을 이용하여 삼차식을 매번 구하는 방법으로 구현했는데 이 방법이 가장 좋게 나올 것 같다는 예상을 했고 부드러워지는 효과는 얻었으나 뭔가 엄청 좋은 효과를 보여주지는 못했다. Couple 사진은 오히려 Nearest_Neighbor 방식이 Bilinear보다 좋게 나왔는데 사진에서 특정 경계를 기준으로 픽셀값이 매우 크게 변화하는 경우가 많다면 오히려 선형성을 가지는 것보다 더 좋게 나올 수도 있다고 예상하였다. 사진의 특성을 너무 많이 고려하게 되면 다양한 사진에 적용하기 어려워질테고 범용적으로 알고리즘을 구현하면 전체적으로 품질이 낮아질텐데 이와 같은 상황에선 어떻게 알고리즘을 선택하고 사용할지 고민해보는 것이 좋을 것 같다.