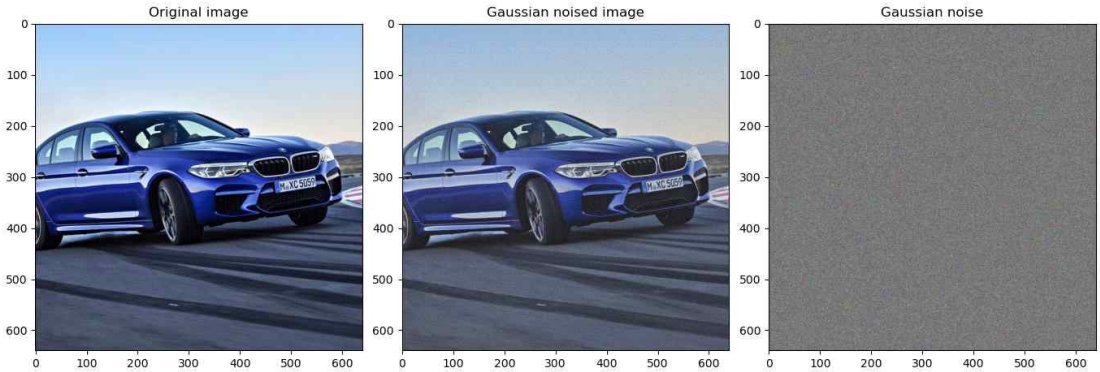
Computer Vision Programming Assignment 2

2019038359 Jiyong Boo

1. Mean Filtering



(1) Gaussian Noise

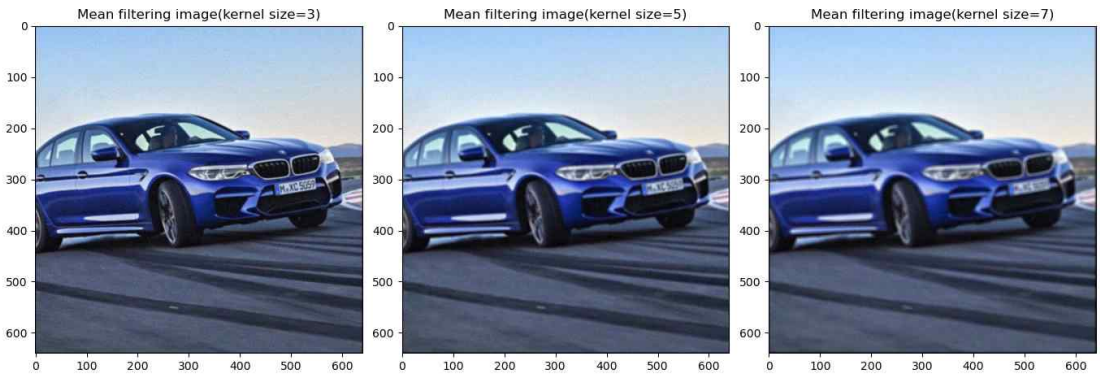
Gaussian Noise는 가우시안 분포(정규 분포)를 따르는 랜덤한 노이즈이다. 따라서 정규 분포를 따르는 랜덤한 수를 생성해주는 Numpy.random.normal() 함수를 이용해, 원본 이미지와 같은 크기의 노이즈를 생성하였다. 그리고 원본 이미지에 생성한 노이즈를 더하여 noised image를 얻을 수 있었다. 위 결과는 N(0,10)을 따르는 가우시안 노이즈를 적용한 것이다.

+ 가우시안 노이즈를 포함해 이후 진행하는 이미지 processing 과정에서 픽셀 값의 type이 uint8(0~255)가 아닌, 소수점을 갖는 float type 값들을 갖게 된다. 따라서 처리된 이미지들을 시각화 하기 위해서 float type의 이미지를 uint8 type으로 변환하는 float2uint8() 함수를 만들었다.

(2) Mean Filtering

Mean Filtering은 kernel size 안의 픽셀들의 평균으로 픽셀값을 반환하여 노이즈를 필터링 하는 방법이다. Mean Filtering을 적용하면 이미지의 크기가 (원본 이미지 크기 - kernel size + 1)로 감소한다. 따라서 필터링 적용 후 이미지의 크기가 원본 이미지와 같게 하기 위해 Padding을 적용하는 함수를 구현하였다.

Padding 함수는 원본 이미지 좌우상하 맨 끝에 (kenel size//2)만큼 0 값을 갖는 픽셀을 추가해주는 것으로 구현하였다.



Kernel size를 3,5,7로 하여 Mean filtering을 적용한 결과가 위와 같이 나오게 되었다. Kernel size가 클수록 noise가 줄어들지만, 이미지가 더 blur해지는 것을 확인할 수 있다.

(3) PSNR

$$MSE = \frac{\sum_{M,N}[I_1(m,n) - I_2(m,n)]^2}{M * N} \qquad PSNR = 10\log_{10}\left(\frac{R^2}{MSE}\right)$$

위 식을 토대로 PSNR 함수를 구현하였다. PSNR은 픽셀 level에서 계산되는 값으로, 클수록 두 이미지가 비슷하다는 것을 의미한다.

```
PSNR(filtering kernel size = 3) :  28.81812059286544
MSE(filtering kernel size = 3) :  85.36273111979166
PSNR(filtering kernel size = 5) :  28.555095522518833
MSE(filtering kernel size = 5) :  90.69238199869791
PSNR(filtering kernel size = 7) :  28.470392316196072
MSE(filtering kernel size = 7) :  92.47857503255209
```

Kernel size가 3,5,7인 mean filter를 적용한 이미지들과 gaussian noised 이미지 사이의 PSNR과 MSE를 계산한 결과 위와 같이 나왔다. Kernel size가 커질수록 PSNR 값이 작아지는 것을 확인할 수 있었다. 이는 kernel size가 커짐에 따라 blur가 심해지면서, 필터 적용 전 이미지와 픽셀 값 차이가 커졌기 때문이라 생각된다.

2. Unsharp Masking

(1) Gaussian Filtering

$$\underset{\text{image}}{F} + \alpha\,(F - \underbrace{F * H}_{\text{blurred image}}) = (1+\alpha)\,F - \alpha\,(F * H) = F * (\underbrace{[1+\alpha]\,e}_{\text{(Identity kernel)}} - \alpha H)$$

위 식에 따라 Sharpen Filter를 만들기 위해서, 우선 Gaussian Filter를 구현하였다.

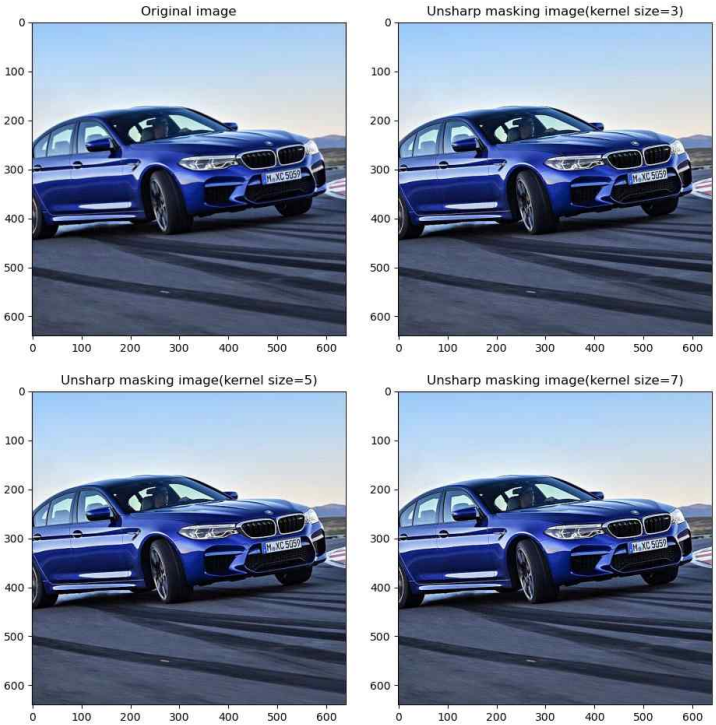
$$G_\sigma = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

Gaussian Filter를 구현하기 위해, Kernel size에 맞춰 x,y값을 설정하고 위 식을 이용해 필터를 구현하였다.

(2) Unsharp Masking

Gaussian filter를 이용하여 blurred image를 생성하고, 원본 이미지에 blurred image를 뺌으로써 unsharp masking을 구현하였다. unsharp masking은 원본 이미지에 high pass filter를 적용한 이미지를 더하여, 특징이 더 두드러지게 한다.

Kernel size를 3,5,7, alpha는 1로 하여 unsharp masking을 적용한 결과 다음과 같이 나왔다. 원본 이미지 자체가 외곽선 등의 특징이 뚜렷하여 시각적으로 큰 차이는 없으나, 자세히 보면 특징이 조금 더 뚜렷해진 것을 확인할 수 있었다. 그리고 Kernel size에 따른 결과의 큰 차이는 없는 것으로 보인다.

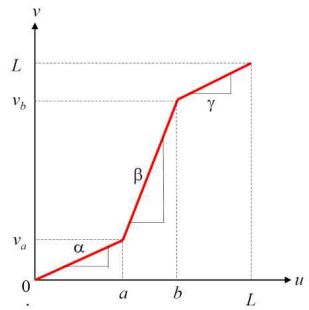


3. Contrast stretching

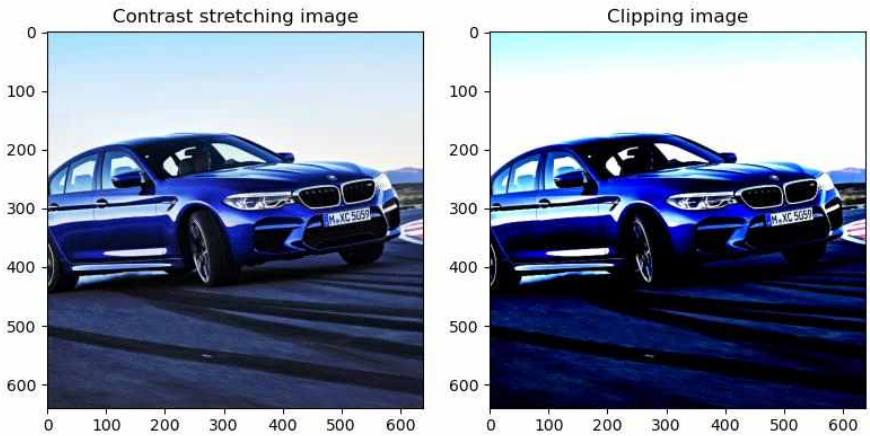
(1) Contrast stretching

$$v = f(u), u \in [0, L], v \in [0, L]$$

$$v = \begin{cases} \alpha u, & 0 \le u < a \\ \beta(u-a) + v_a & a \le u < b \\ \gamma(u-b) + v_b & b \le u < L \end{cases}$$



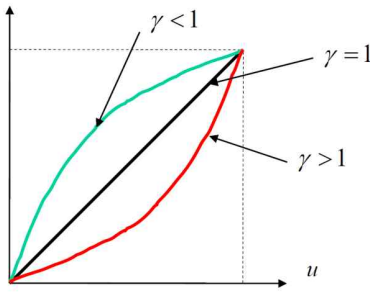
위 식에 따라 Contrast stretching을 적용하는 함수를 구현하였다. a,b,va,vb 값을 매개변수로 설정하고, alpha, beta, gamma 값은 a,b,va,vb를 이용하여 계산하였다. a=80,va=50,b=180,vb=210/ a=80,va=0,b=180,vb=255 (clipping) 으로 contrast stretching을 실행한 결과 다음과 같이 나왔다.



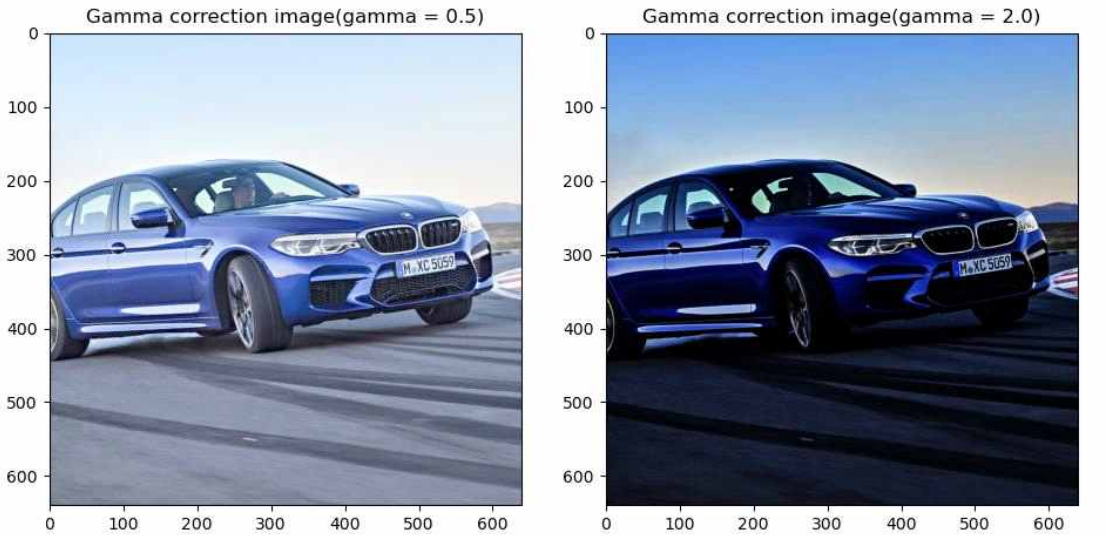
Contrast stretching 결과 어두운 픽셀(값이 낮은 픽셀)은 더 어두워지고, 밝은 픽셀(값이 높은 픽셀)은 더 밝아진 것을 확인할 수 있었다. Clipping에서는 그 정도가 더 큰 것도 확인할 수 있었다

(2) Gamma correction

$$v = cu^{\gamma}$$



위 식에 따라 gamma correction은 적용하는 함수를 구현하였다고, gamma = 0.5, 2.0으로 적용한 결과 다음과 같이 나왔다.



gamma가 1보다 작을 땐 이미지가 전체적으로 밝아지고, gamma가 1보다 클 땐 이미지가 전체적으로 어두워지는 것을 확인할 수 있었다.

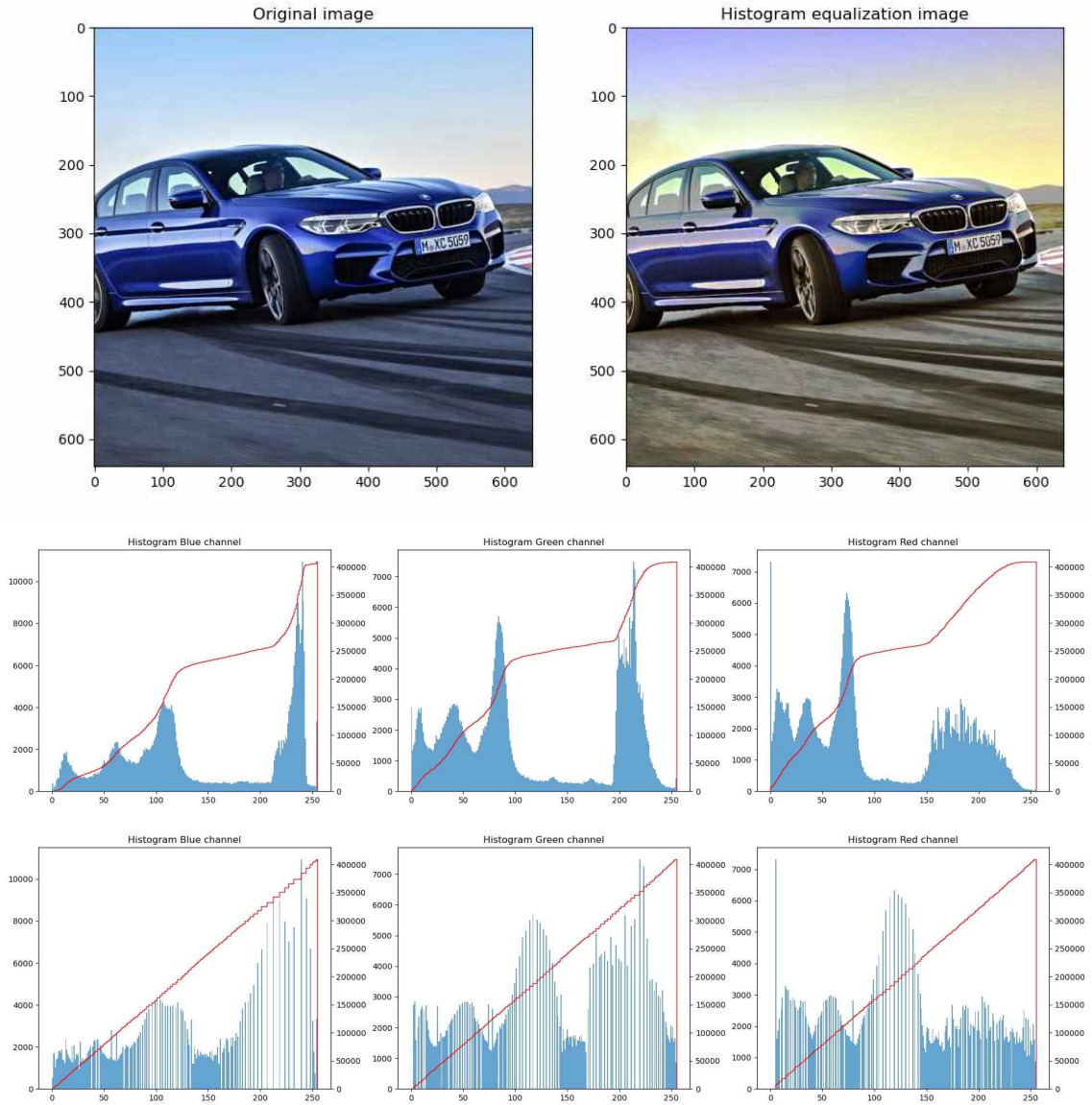
4. Histogram Equalization

(1) Histogram equalization

$$v \equiv F_u(u) \equiv \int_0^u p_u(u)du = \Pr[\mathbf{u} \le u]$$

$$F_u(u) : CDF(\text{Cumulative Distributi on Function})$$

위 식에 따라 Histogram equalization을 적용하는 함수를 구현하였다. RGB 각 채널 별로 histogram equalization을 진행하였다. 이미지의 픽셀값 히스토그램을 만들고, 그것을 토대로 0~255인 픽셀값의 CDF를 계산하였다. 그리고 round(CDF * 255) 한 값으로 픽셀 값을 변환함으로써 Histogram equalization을 적용하였다. 결과는 다음과 같다.



Histogram Equalization을 적용한 이미지와 원본 이미지를 비교해보면, 원본 이미지는 전체적으로 파란색이지만 Histogram Equalization을 적용한 이미지는 붉은색도 포함된 것을 확인할 수 있다.

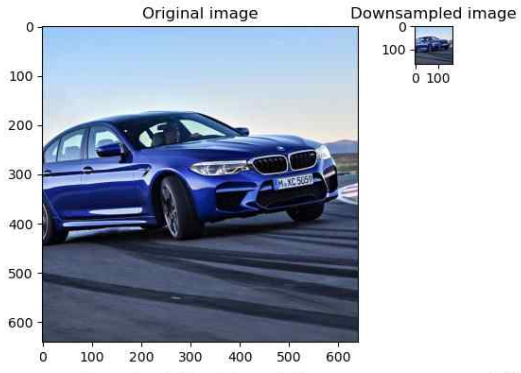
또한 각 이미지의 채널별 히스토그램은 시각화한 결과, 이미지의 픽셀 값별 히스토그램을 나타내는 파란색 막대가 전 픽셀값에 걸쳐 고르게 분포하도록 변한 것을 확인할 수 있었다. 그리고 누적 히스토그램은 나타내는 빨간색 선 그래프도 histogram equalization을 적용한 이후, 1차원 선형으로 바뀐 것을 확인할 수 있었다.

5. Image Upsampling

(1) Down sample

Mean pooling 방식을 이용하여 이미지의 높이, 너비를 각각 1/4배로 downsample 하는 함수를 구현하였다. 4*4 픽셀의 값들을 평균내어 하나의 픽셀로 만듦으로써 구현하였다. 이미지

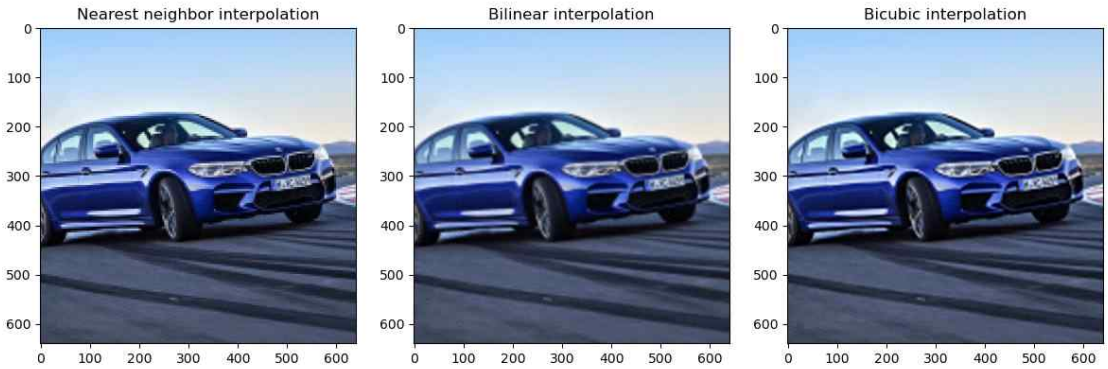
의 높이와 너비가 4의 배수가 아닌 경우, 나머지는 버리도록 구현하였다. 결과는 다음과 같다.



원본 이미지의 사이즈가 640*640이었는데, down sample한 결과 160*160이 된 것을 확인할 수 있었다.

(2) Interporation by cv2.resize() function

opencv의 cv2.resize() 함수를 이용하여 interporation을 진행하였다. cv2.resize() 함수의 interporation변수를 각각 cv2.INTER_NEAREST, cv2.INTER_LINREAR, cv2.INTER_CUBIC으로 하여 nearest neighbor, bilinear, bicubic interporation을 적용하였고 결과는 다음과 같다.



Nearest neighbor interporation 결과 이미지가 부드럽지 않게 upsample 되었고, bilinear와 bicubic interporation 결과 이미지가 부드럽게 upsample 되었다. 그러나 원본 이미지와 비교했을 땐, 다소 blur된 상태로 upsample 된 것을 확인할 수 있었다.

각각의 Interporation 결과와 원본 이미지 사이의 PSNR을 계산한 결과 다음과 같다.

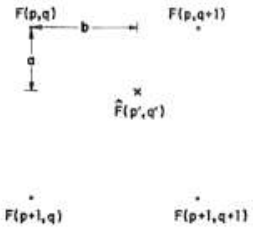
```
PSNR,MSE(nearest neighbor) :  (33.83207497915184, 26.90746988932292)
PSNR,MSE(bilinear) :  (33.72961002702496, 27.549857584635415)
PSNR,MSE(bicubic) :  (33.899522817029926, 26.492813313802085)
```

눈으로 보았을 땐, nearest neighbor < bilinear < bicubic 순으로 부드럽게 복원된 것으로 보였으나, PSNR은 bilinear < nearest neighbor < bicubic 순으로 나왔다. 그 이유를 분석해보았을 때, PSNR은 픽셀 level에서 일치도를 계산한 것이므로, 이미지에 따라 다르게 계산될 것이라 생각된다.

(+) Interporation by 직접 구현

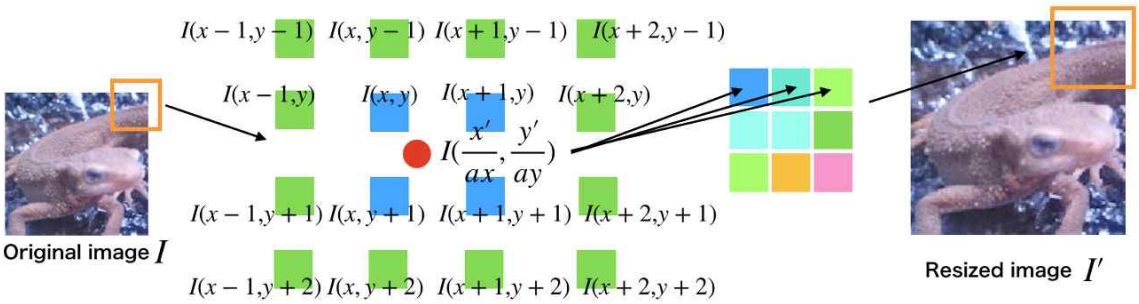
다음 식들을 활용해 직접 구현해 보았다.

[bilinear]



$$F(p',q') = (1-a)[(1-b)F(p,q) + bF(p,q+1)] + a[(1-b)F(p+1,q) + bF(p+1,q+1)]$$

[bicubic]



$$d_{x1} = \left|\frac{x'}{ax} - (x-1)\right| \quad d_{x2} = \left|\frac{x'}{ax} - x\right| \quad d_{x3} = \left|\frac{x'}{ax} - (x+1)\right| \quad d_{x4} = \left|\frac{x'}{ax} - (x+2)\right|$$

$$d_{y1} = \left|\frac{y'}{ay} - (y-1)\right| \quad d_{y2} = \left|\frac{y'}{ay} - y\right| \quad d_{y3} = \left|\frac{y'}{ay} - (y+1)\right| \quad d_{y4} = \left|\frac{y'}{ay} - (y+2)\right|$$

$$h(t) = \begin{cases} (a+2)|t|^3 - (a+3)|t|^2 + 1 & when\ |t| \le 1 \\ a|t|^3 - 5a|t|^2 + 8a|t| - 4a & when\ 1 < |t| \le 2 \\ 0 & else \end{cases}$$

$$I'(x',y') = \frac{1}{\sum_{j=1}^{4}\sum_{i=1}^{4} h(d_{xi})h(d_{yj})} \sum_{j=1}^{4}\sum_{i=1}^{4} I(x+i-2, y+j-2)h(d_{xi})h(d_{yi})$$