영상처리 2주차 과제

학번: 201404376

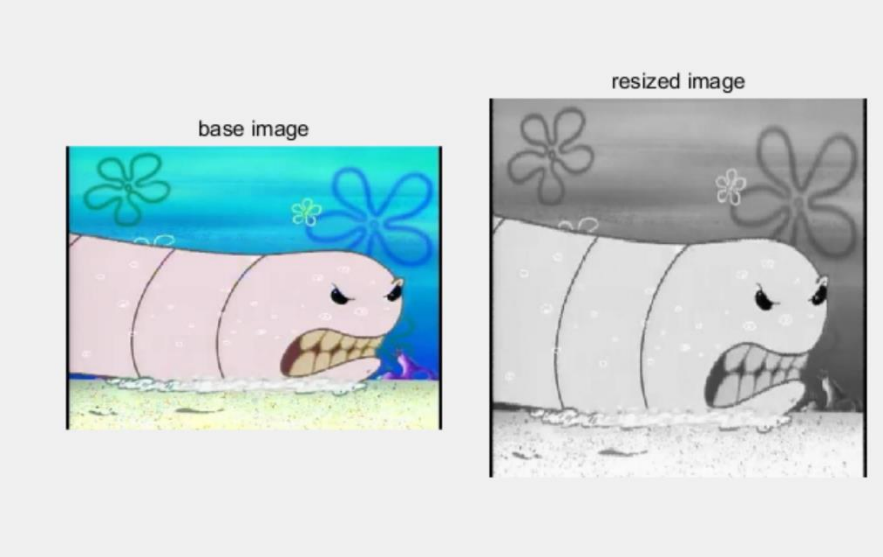
이름: 정성욱

1. 과제 내용

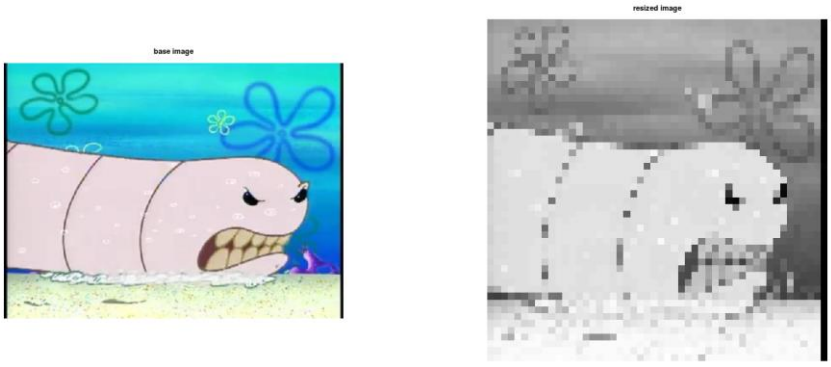
:: 구현한 과제에 대한 설명, 어떤 방식으로 접근해야 하는지

실행 화면:

1000,1000으로 늘렸을 때,

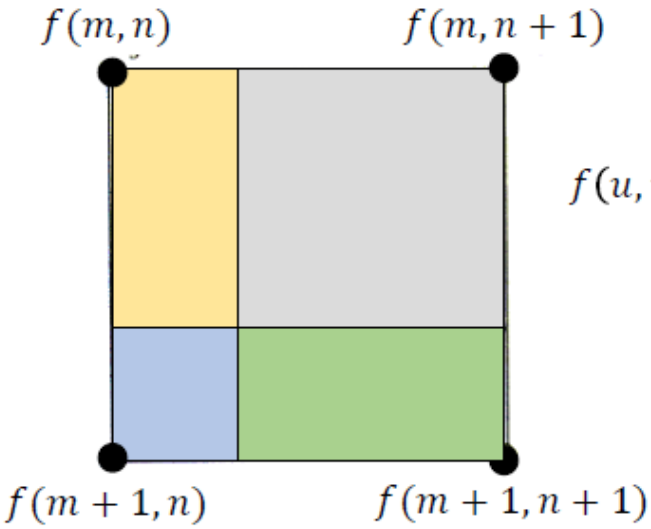


50,50으로 줄였을 때,



우선 과제의 목표는 blinear의 기법을 구현하여 이미지의 크기를 최대한 손실을 줄여 확대 축소 시키는 것입니다. 위의 사진에서 구현된 이미지의 원본의 크기는(640,480)의 크기를 가지고 있지만, 이 것을 (1000,1000)으로 늘린 것입니다.

300에서 1000으로 늘리면서 공백의 픽셀이 생깁니다. 즉, 표시가 불가능한 인덱스에 대한 표시를 해줘야 하는데 그것의 기준에 대한 설명입니다. 저는 bilinear의 가장 퓨어한 방법인



$$f(u,v) = (1-s)(1-t)\cdot f(m,n) + s(1-t)\cdot f(m,n+1) + (1-s)t\cdot f(m+1,n) + st\cdot f(m+1,n+1)$$

이것을 그대로 구현하였습니다. 즉 늘리고자 하는 이미지의 픽셀 값에 대응되는 원본 이미지의 값을 비율로 추론하여, 위의 bilinear의 공식을 대입하여, 인덱스의 값들을 채워 넣었는식으로 하였습니다. 줄일 때는 역으로 원본 이미지에서 줄여진 이미지에 비율로 대응하는 픽셀값을 bilinear 연산을 하여 채워 넣었습니다.

2. 구현한 방법에 대한 이유

:: 구현한 방법에 대한 설명 (왜 이렇게 구현했는지 자세히)

```
function re_img = my_bilinear(img,row,col)
   img = rgb2gray(img);
   re_img = zeros(row,col);
   [x,y] = size(img);
   pad_img = zeros(x+2,y+2);
   pad_img(2:x+1,2:y+1) = img;

   pad_img(1,1) = img(1,1);
   pad_img(1,y+2) = img(1,y);
   pad_img(x+2,1) = img(x,1);
   pad_img(x+2,y+2) =img(x,y);

   pad_img(2:x+1,1)=img(1:x,1);
   pad_img(2:x+1,y+2)=img(1:x,y);
   pad_img(1,2:y+1) = img(1,1:y);
   pad_img(x+2,2:y+1) = img(x,1:y);
   img = uint8(pad_img);
   r_p = x/row;
   c_p = y/col;

   for i = 1:row
      for j=1:col
         x_b = floor(i*r_p)+1;
         y_b = floor(j*c_p)+1;
         x_t = floor(i*r_p)+2;
         y_t = floor(j*c_p)+2;
         s = abs(i*r_p-x_b);
         t = abs(j*c_p-y_b);
         xbyb=(1-s)*(1-t)*img(x_b,y_b);
         xtyt=(s)*(t)*img(x_t,y_t);
         xtyb=((s)*(1-t)*img(x_b,y_t));
         xbyt=((1-s)*(t)*img(x_b,y_t));
         re_img(i,j) = xbyb+xtyt+xtyb+xbyt;
      end
   end
   re_img = uint8(re_img);
   end
```

이미지의 확대,축소를 할 때 이미지의 손실을 최대한 줄이기 위해, 원본이미지의 가장자리에 패딩을 하나씩 추가하였습니다.

어떤 이미지를 1000,1000으로 늘렸다고 칩시다. 그러면 제시한 그림 대로의 매커니즘으로 구현하자면, 980,980의 인덱스에 대한 색을 배정을 해야 합니다. 그것에 대한 과정을 설명하겠습니다. 우선 980에 대응되는 원본 이미지의 인덱스를 구해야 하는데, 원본이미지의 가로or세로 크기/늘리고자하는 이미지의 가로or 세로 크기로 배율을 잡아야합니다. 원본이미지의 크기를 360,460이라고 가정하겠습니다. 그러면 460/1000 이면 0.46이되고 여기에 980에 을 곱해주면 원본이미지에서 늘린이미지의 980점에 대응하는 위치가 나타납니다. 그러면 450.8이되는데 이 450.8은 좌표상에 없습니다. 마찬가지로 가로에 대해 같은 연산을 해주면 352.8이됩니다. 즉 980,980이 대응 하는 점은 360,460 에서 352.8,450.8이 되는 것입니다. 이게 위의 s,t가 되는 것입니다. 여기서 bilinear을 하기 위해 s,t좌표의 내림차순을 각각 x_b,y_b 라고 잡겠습니다. 그리고 이 x_b,y_b에 1을 더한 것이 x_t,y_t인데 이것은 위의 그림에서 m+1,n+1입니다.(코드에서는 x_b+1,y_b+1을하고 x_t는 x_b+2, y_t는 y_b+2로 기본 매커니즘과 다르게 1을 더했는데 그것은 매트랩이 0인덱스가 없기 때문에 0을 방지해주는 의미에서 1을 더한 것 입니다.) 여기에서 이제 비율을 구해야 하는데, f 내림을 한 352와 내림+1의 값인 353그리고 450과 451에서 0.8만큼이 비율이 되는 것입니다. 식으로 표현하면 (1-s)*(1-t)*f(m,n)+(s)*(t)*f(m+1,n+1) +(1-s)*(t)*f(m,n+1)+ (s)*(1-t)*f(m+1,n)이 됩니다 이공식을 대입하면 0.2*0.2*f(352,450)+0.8*0.8*f(353,451)+ 0.2*0.8*f(352,451) + 0.8*0.2*f(353,450)이 나오고 이것이 980,980에 들어갈 픽셀의 값이 되는 것입니다. 이 매커니즘을 전부 적용시켜서 구했습니다. 이미지의 축소에도 역시 적용시킬 수 있습니다.

느낀 점

:: 구현하면서 느낀 점, 어려웠던 점, 혹은 설명이 필요하다고 느낀 부분

인덱싱하는 것이 조금 어려웠는데 조교님의 기가 막힌 명강의 명ppt로 인해 잘 해결 할 수 있었습니다. 감사합니다!