

Homework #1

2023 EE402
Digital Image Processing

1. (20 pts) You are preparing a report and have to insert in it an image of size 2048×2048 pixels.
 - a) Assuming no limitations on the printer, what would the resolution in line pairs per mm have to be for the image to fit in a space of size 5×5 cm?
 - b) What would the resolution have to be in dpi (dots per inch) for the image to fit in 2×2 inches?

a) 2048 pixel을 표현하기 위해서는 2048개의 line이, 1024개의 line pair가 필요하다.
1024개의 line pair가 5cm (= 50mm) 안에 들어가려면,

$$1024 \text{ (line pairs)} / 50 \text{ (mm)} = 20.48 \text{ (line pairs/mm)}$$

를 만족해야 한다.

답: 20.48 (line pairs per mm)

b) 2048개의 pixel이 2inch 안에 들어가려면,

$$2048 \text{ (dots)} / 2 \text{ (inch)} = 1024 \text{ (dpi)}$$

를 만족해야 한다.

답: 1024 (dpi)

2. (30 pts)

- a) Suppose that a flat area with center at (x_0, y_0) is illustrated by a light source with intensity distribution

$$i(x, y) = K \exp(-[(x - x_0)^2 + (y - y_0)^2])$$

Assume for simplicity that the reflectance of the area is constant and equal to 1.0, and let $K=255$. If the intensity of the resulting image is quantized using k bits, and the eye can detect an abrupt change of eight intensity levels between adjacent pixels, what is the highest value of k that will cause visible false contouring? (15 pts)

- b) Sketch the image in (a) for $k=2$. (15 pts)

a) Intensity of the point (x, y) 를 $f(x, y)$ 라 하자. 이는 아래와 같이 계산 된다.

$$f(x, y) = i(x, y) \cdot r(x, y)$$

이때, $i(x,y)$ 는 illumination at the point (x,y) , $r(x,y)$ 는 reflectance at the point (x,y) 이다. 문제에서 주어진 정보를 이용하면, $f(x,y)$ 는 아래와 같이 표현된다.

$$\begin{aligned} f(x,y) &= i(x,y) \cdot r(x,y) \\ &= K \exp(-[(x-x_0)^2 + (y-y_0)^2]) \cdot 1 \\ &= 255 \cdot \exp(-[(x-x_0)^2 + (y-y_0)^2]) \end{aligned}$$

즉, $f(x,y)$ 는 $\{(x-x_0)^2 + (y-y_0)^2\} = 0$ 때 최댓값 255를 갖고, $\{(x-x_0)^2 + (y-y_0)^2\}$ 값이 무한히 커질 때 최솟값 0을 갖는다.

$$0 \leq f(x,y) \leq 255$$

이 image는 k bits를 이용하여 quantize되었으므로, 2^k 개의 gray level을 표현할 수 있다. 이때, eye는 pixel 사이에 intensity level이 8 차이날 때 그 변화를 감지할 수 있다고 하였으므로, visible false contouring을 느끼게 하기 위해서는 아래 조건을 만족해야 한다.

$$(255+1)/2^k \geq 8$$

$$2^{8-k} \geq 2^3$$

$$8-k \geq 3$$

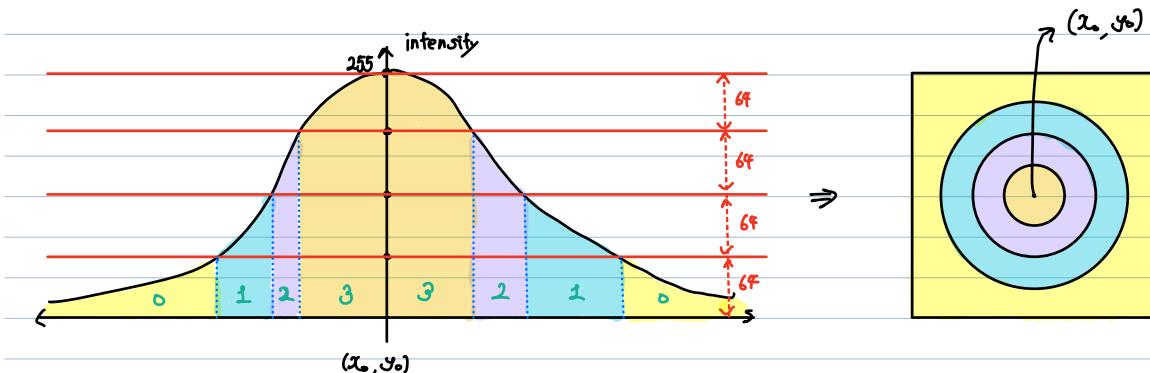
$$\therefore k \leq 5$$

따라서, 주어진 조건을 만족하는 highest value of k 는 5이다.

$$\therefore k = 5$$

b) $k=2$ 인 경우, $2^2 = 4$ 개의 gray level로 이미지를 표현하여, 각 gray level은 $64(256/4)$ 의 intensity interval을 갖는다. 즉, 4개와 같은 형태로 표현될 수 있다.

| intensity interval | gray level | 2-bit |
|--------------------|------------|-------|
| [0, 64] | 0 | 00 |
| [64, 128] | 1 | 01 |
| [128, 192] | 2 | 10 |
| [192, 256] | 3 | 11 |



$f(x,y) = 255 \cdot \exp(-[(x-x_0)^2 + (y-y_0)^2])$ 에서, $(x-x_0)^2 + (y-y_0)^2$ 이 원의 넓임인 이므로, image sketch는 위와 같이 (x_0, y_0) 를 중심으로 축심원을 그리며 총 4개의 gray level을 갖는 형태일 것이라 예상할 수 있다.

3. (30 pts) Consider the image segment shown in the following figure.

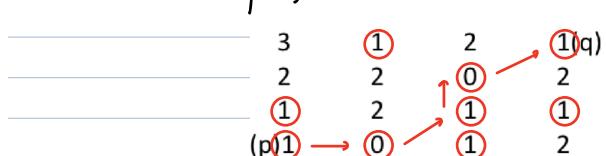
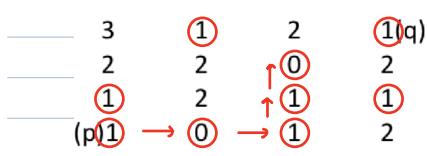
| | | | |
|------|---|---|------|
| 3 | 1 | 2 | 1(q) |
| 2 | 2 | 0 | 2 |
| 1 | 2 | 1 | 1 |
| (p)1 | 0 | 1 | 2 |

- a) Let $V=\{0, 1\}$ be the set of intensity values used to define adjacency. Compute the lengths of the shortest 4-, 8-, and m-path between p and q. If a particular path does not exist between two points, explain why. (15 pts)
- b) Repeat (a) but using $V=\{1, 2\}$. (15 pts)

이미지의 좌측 상단 pixel을 $(0, 0)$, (p)를 $(4, 0)$, (q)를 $(0, 4)$, 우측 하단을 $(4, 4)$ 라 하자.

a) (4-path)

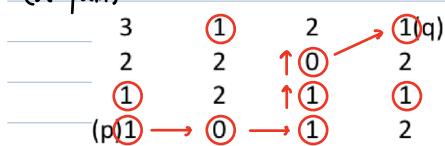
(8-path)



4-path은 존재하지 않는다.

(q)의 4-adjacent pixel이 존재하지 않기 때문이다.

(m-path)



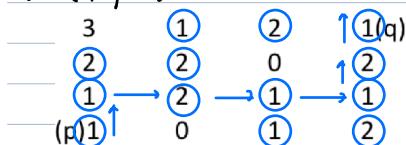
path: $(p) \rightarrow (3,1) \rightarrow (2,2) \rightarrow (1,2) \rightarrow (0,1) \rightarrow (1,0) \rightarrow (2,1) \rightarrow (3,2) \rightarrow (q)$

length : 5

path : $(p) \rightarrow (3,1) \rightarrow (2,2) \rightarrow (1,2) \rightarrow (q)$

length : 4

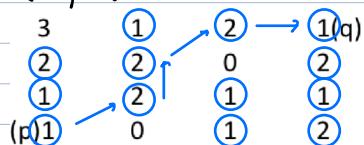
b) (4-path)



path : $(p) \rightarrow (2,0) \rightarrow (2,1) \rightarrow (2,2) \rightarrow (2,3) \rightarrow (1,3) \rightarrow (q)$

length : 6

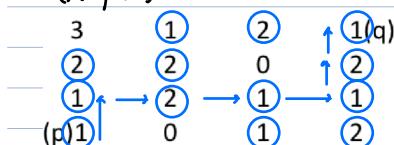
(8-path)



path : $(p) \rightarrow (2,1) \rightarrow (1,2) \rightarrow (0,1) \rightarrow (1,0) \rightarrow (2,1) \rightarrow (3,2) \rightarrow (q)$

length : 4

(m-path)



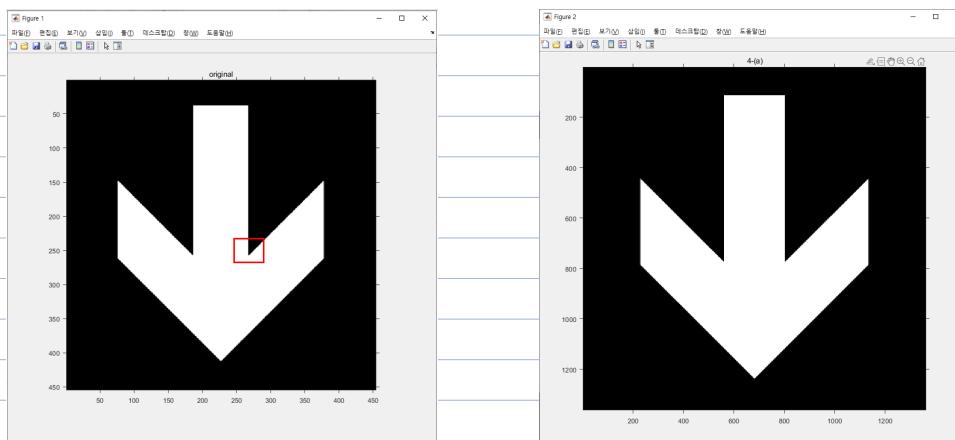
path : $(p) \rightarrow (2,0) \rightarrow (2,1) \rightarrow (2,2) \rightarrow (2,3) \rightarrow (1,3) \rightarrow (q)$

length : 6

4. (80 pts) Write a MATLAB program for the affine transformation for scaling.
- The program should consist of two files named main.m and Scaling.m provided with this homework question. The main.m should call the function in Scaling.m that is responsible for increasing the image and spatial transformation of coordinates. **Show the result of your program.** (20 pts)
 - Discuss the difference between the results obtained by your program and MATLAB functions** (i.e., `affine2d(...)` function). (10 pts)
 - Explain the bilinear interpolation and write a program for the bilinear interpolation** which should be in the `Bilinear_Interp.m` file. Your main function should use this function to improve the quality of the scaled image obtained in (a). **Discuss the effect of the bilinear interpolation on the image quality** (50 pts).

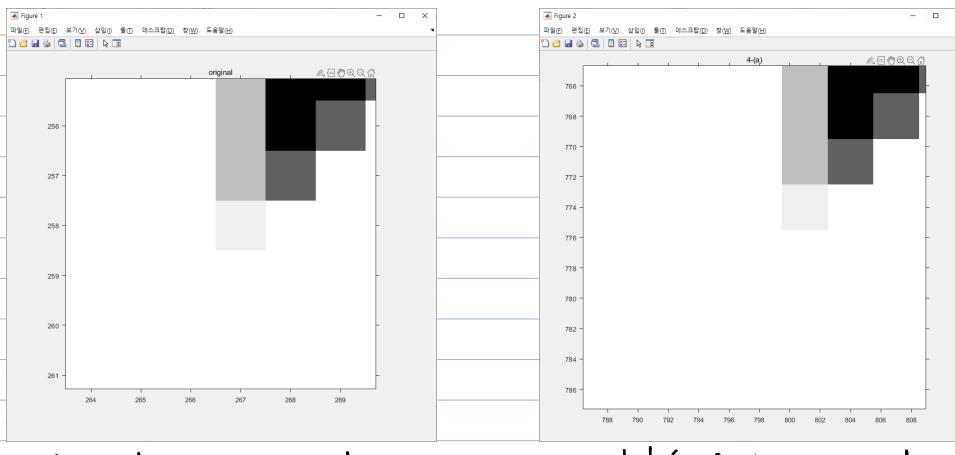
* 모든 코드와 결과 이미지는 대입으로 제출하세요.

(a)



< original image >

< scaled (x3) image >

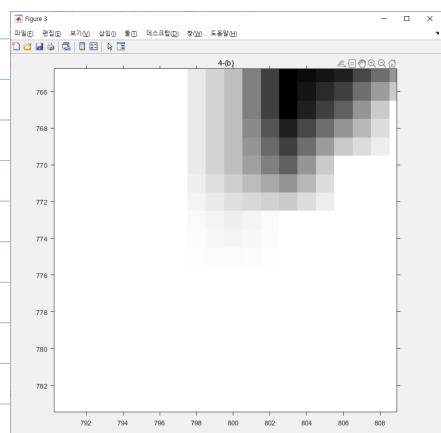
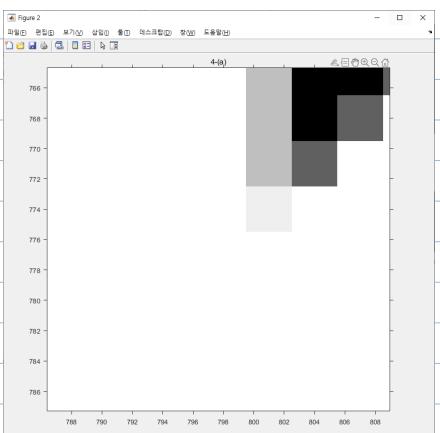


< original image (zoomed) >

< scaled (x3) image (zoomed) >

상단 2개의 image는 각각 원본 image와 확장 이미지를 가로, 세로 각각 3배로 (`4(x)`) 한 결과이다. 아래 2개의 image는 원본 이미지와 scaling 된 image의 red box에 해당하는 영역을 확대한 결과이다. 원본 이미지와 scaled image가 정확히 같은 pixel 값을 갖는 것을 확인할 수 있다.

b)



$$A = \begin{bmatrix} 3 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

<result of 4-(a)>

<result of MATLAB function>

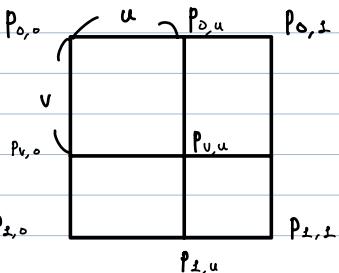
<Affine Matrix>

4-(a)의 결과와 MATLAB의 `affineform2d` function을 이용하여 같은 결과의 동일한 영역을 확대한 결과이다.

`affineform2d` 함수에서 사용한 affine matrix는 위와 같다. 두 이미지는 모두 가로, 세로 각 3배 커진 상태이다. 특히 그림에서는 nearest neighbor interpolation을 사용하면서 같은 값을 갖는 pixel 영역이 9배 (가로 3배 × 세로 3배) 씩 증가한 것을 알 수 있다.

오른쪽 그림에서 MATLAB의 function (`affineform2d`)를 이용했을 때, nearest neighbor interpolation을 적용했을 때보다 원본 다양한 intensity level을 갖는 것을 확인할 수 있고, 이를 통해 결과 image의 edge가 더욱 smooth하게 느껴질 수 있다는 것을 알 수 있다.

c) Bilinear interpolation 이란, 1D에서의 linear interpolation을 2D로 확장시킨 개념이다. 이미 값을 알고 있는 4개의 pixel 사이에 위치한 영역의 pixel 값을, 4개의 pixel 각 영역의 정 사이의 거리를 기반으로 추정한다.



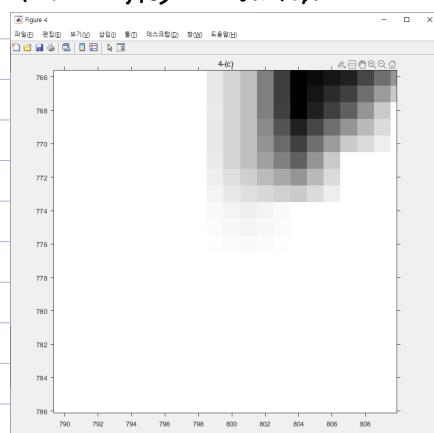
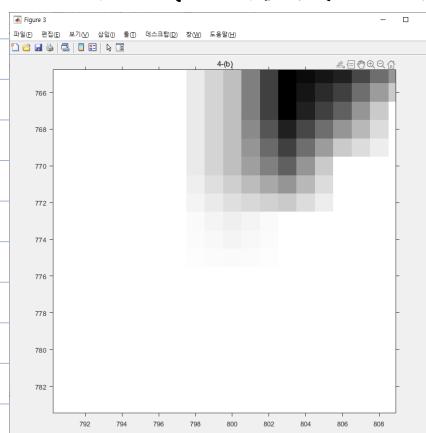
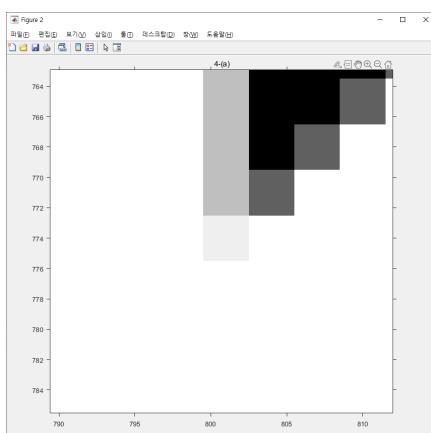
$P_{0,0}, P_{0,1}, P_{1,0}, P_{1,1}$ 의 값은 이미 알고 있는 상황에서, 입력의 점 (u, v) 의 값은 아래와 같이 계산할 수 있다.

$$P_{0,u} = (1-u) P_{0,0} + u P_{0,1}$$

$$P_{1,u} = (1-u) P_{1,0} + u P_{1,1}$$

$$P_{v,u} = (1-v) P_{0,u} + v P_{1,u}$$

$$= (1-v)(1-u) P_{0,0} + (1-v)u P_{0,1} + v(1-u) P_{1,0} + vu P_{1,1}$$



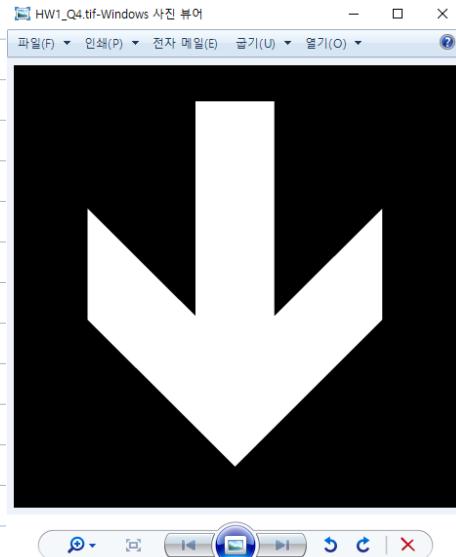
<result of 4-(a)>

<result of MATLAB function>

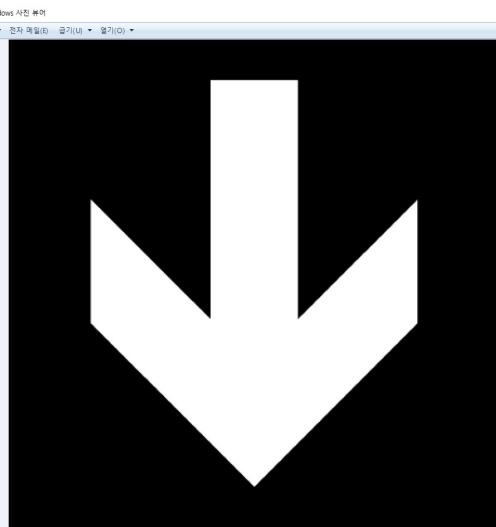
<result of bilinear interpolation>

위의 3개의 image를 비교하면 nearest neighbor interpolation을 사용한 (a)에 비해 (b), (c)가 훨씬 더 매끄러운 결과를 갖는다는 것을 알 수 있다. 또, (b)와 (c)가 갖는 intensity level이 동일한 것을 알 수 있다. 수업시간에 interpolation에 대해 배울 때, bicubic interpolation이 fine detail을 가장 잘 보존하지만 computational cost가 높아 주로 bilinear interpolation을 사용한다고 배웠듯이, MATLAB 내장함수 역시 같은 이유로 bilinear interpolation을 사용하여 구현되어 있는 것이라 추측할 수 있다.

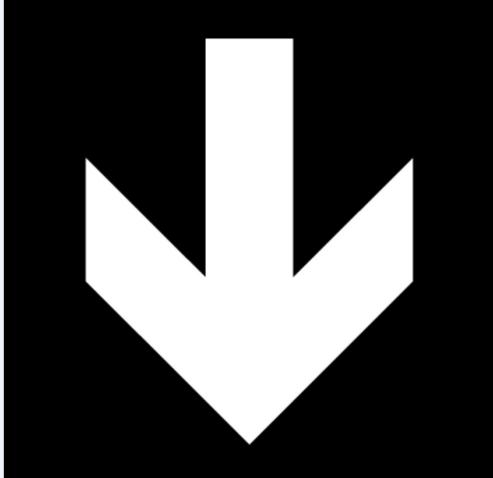
전체 결과



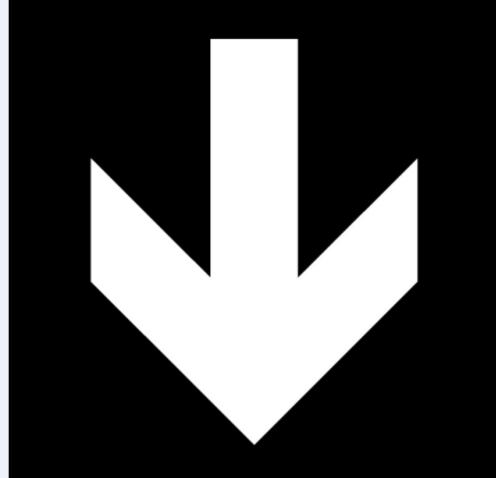
<input>



<4-a>



<4-b>



<4-c>