

MATLAB을 이용한

디지털 영상처리의 기초

- 점 처리: 인접한 픽셀의 데이터를 사용하지 않고 자신 픽셀의 정보만으로 새로운 데이터를 생성, input 영상 크기 = output 영상 크기
- 영역 처리: 한개의 픽셀 데이터를 바꾸기 위해 인접한 데이터로부터 정보를 가져와 새로운 데이터 생성, input 영상 크기 = output 영상 크기
- 기하학적 처리: 영상의 크기가 변한다. 100×100 자리를 200×200 으로 키우면, 30000개의 픽셀은 어디서 오는지? → 데이터의 보간

6.1 데이터의 보간(interpolation)

4개의 점 x_1, x_2, x_3 및 x_4 가 등간격으로 분포하고 이들의 값이 각각 $f(x_1), f(x_2), f(x_3)$ 및 $f(x_4)$ 라 한다. 라인 $x_1 \sim x_4$ 를 따라 8개의 점 x'_1, x'_2, \dots, x'_8 을 구한다. 그림 6.1에 이를 나타내었다.

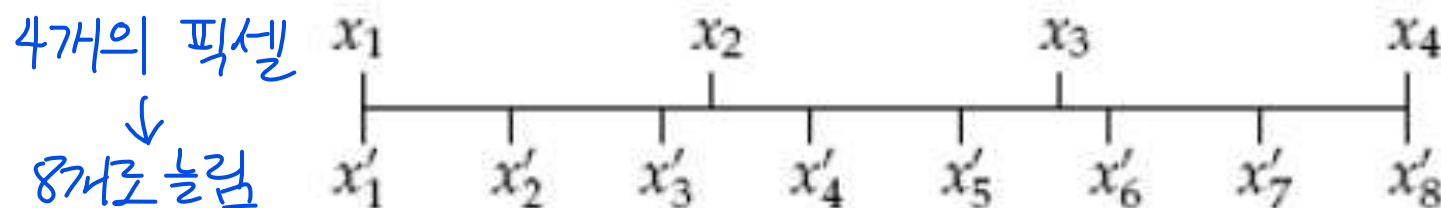


그림 6.1 4개의 점을 8개의 점으로 변환

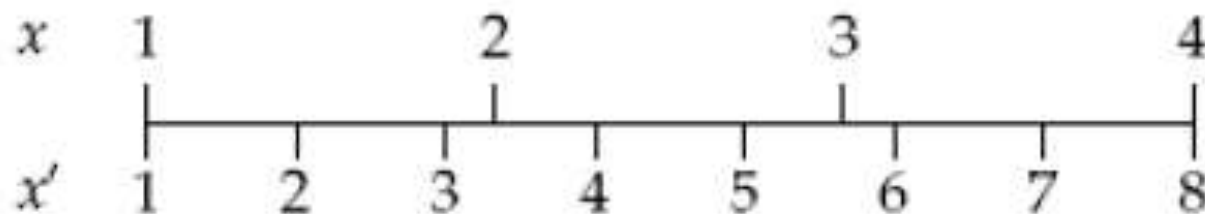
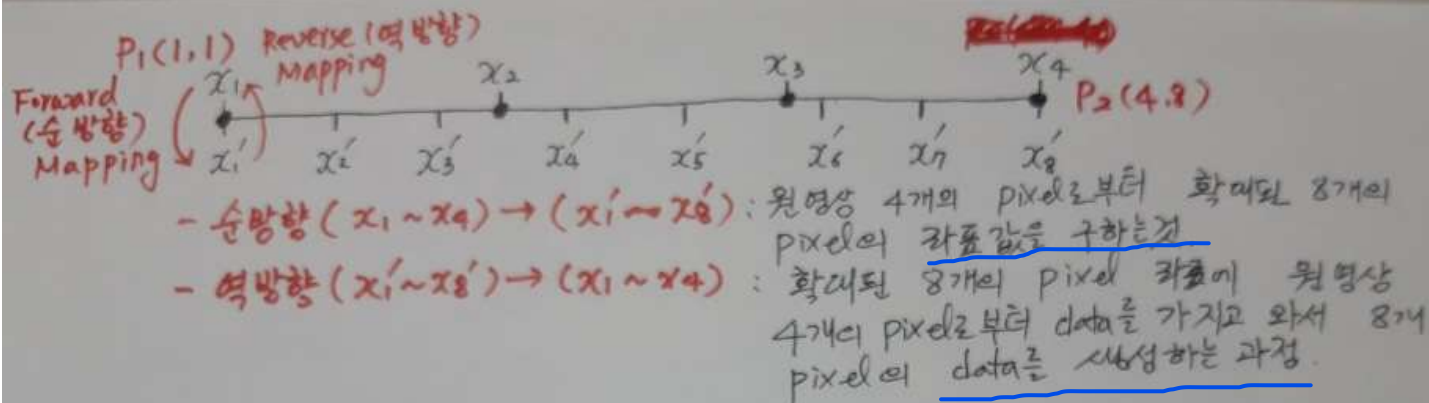


그림 6.2 그림 6.1의 변형



- 순방향: x 로부터 x' 생성

두 점점 $P_1(x, x') = (1, 1)$, $P_2(x, x') = (4, 8)$

일차 방정식 $x' = ax + b$ 라 하면

$$P_1 \text{ 으로부터 } 1 = a + b \rightarrow a = 1 - b$$

$$P_2 \text{ 으로부터 } 8 = 4a + b \rightarrow 8 = 4(1 - b) + b$$

$$\text{정리하면 } a = \frac{1}{3}, b = \frac{-4}{3}$$

$$\text{따라서 순방향 식 } x' = \frac{1}{3}x - \frac{4}{3} = \frac{1}{3}(x - 4)$$

$$\text{역방향 식 } x = \frac{3}{1}x' + \frac{4}{1} = \frac{1}{1}(3x' + 4)$$

$$x' = \frac{1}{3}(7x - 4),$$

$$x = \frac{1}{7}(3x' + 4).$$

데이터가 안섞인

최근접보간법

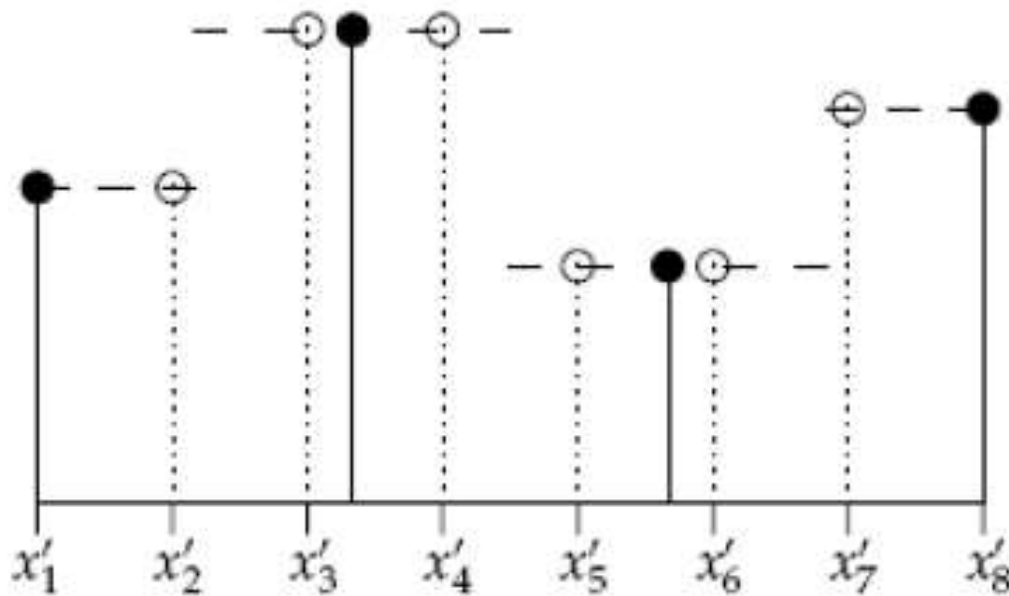


그림 6.3 최근접보간법

선형보간법

데이터가 섞임

$$\frac{F - f(x_1)}{\lambda} = \frac{f(x_2) - f(x_1)}{1}$$

$$F = \lambda f(x_2) + (1 - \lambda)f(x_1).$$

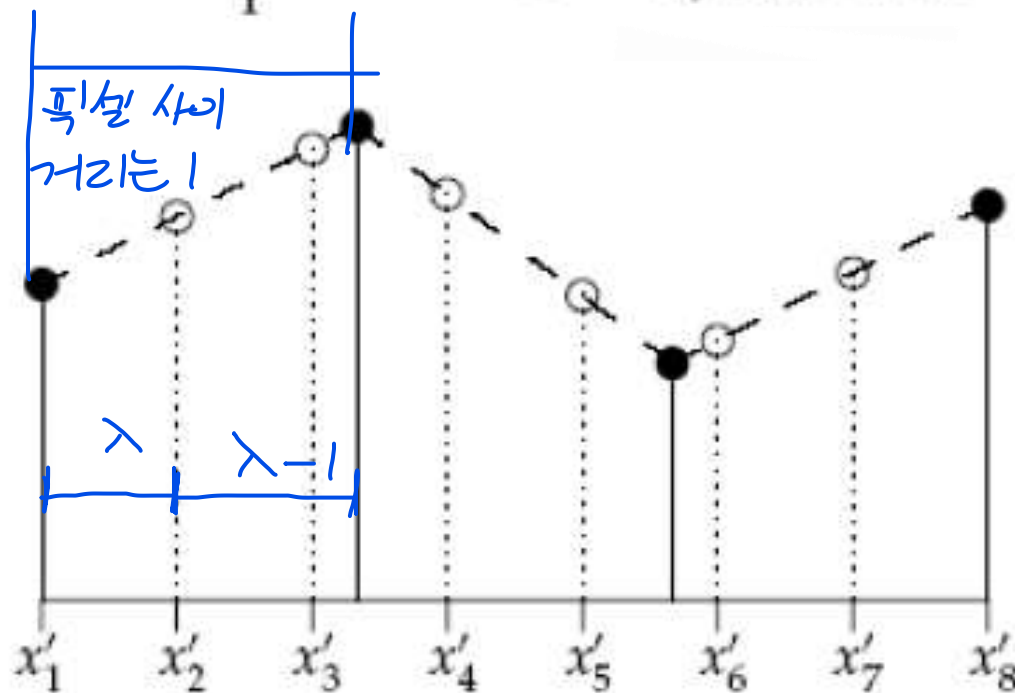
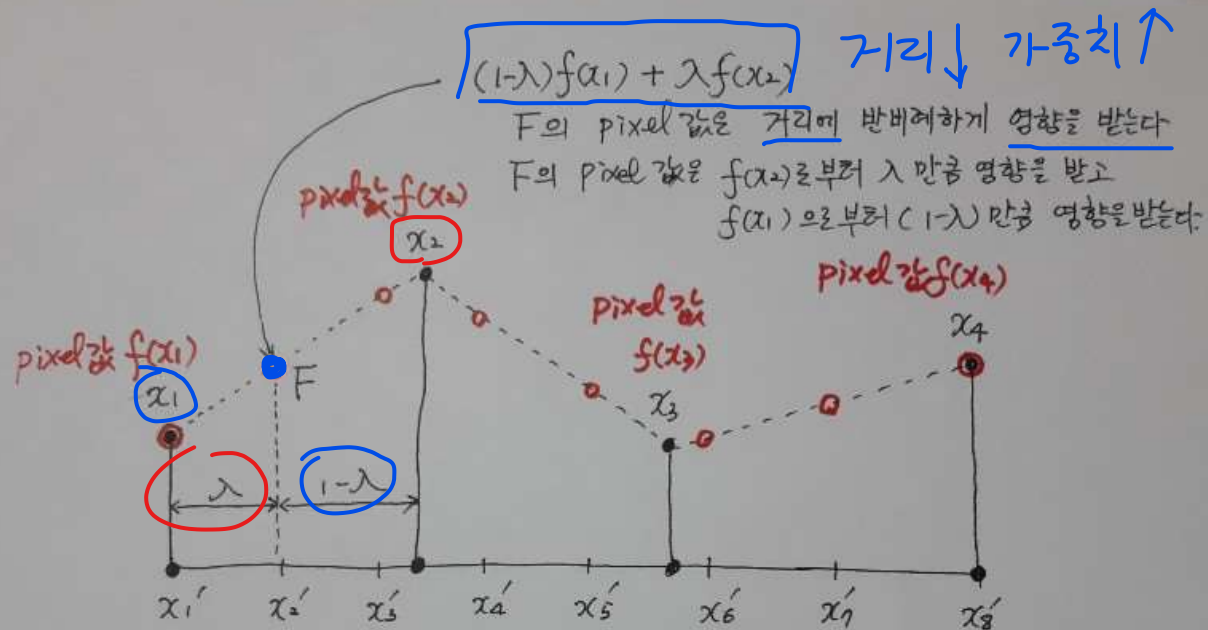


그림 6.4 선형 보간법



예) $f(x_1) = 3$, $f(x_2) = 6$ 이라면 ($\lambda = 0.3$ 이라 가정)

$$F = (1-\lambda)f(x_1) + \lambda f(x_2) = 0.7 \times 3 + 0.3 \times 6$$

픽셀은

$$= 2.1 + 1.8 = 3.9 \approx 4$$

정수

따라서 순방향 mapping으로 형성된 라플 x_2 의
pixel 값은 역방향 mapping으로 원래 영상 라플
 x_1 과 x_2 의 사이에 위치하고 그때의 pixel 값은
 $f(x_2) = 4$ 를 결정된다.

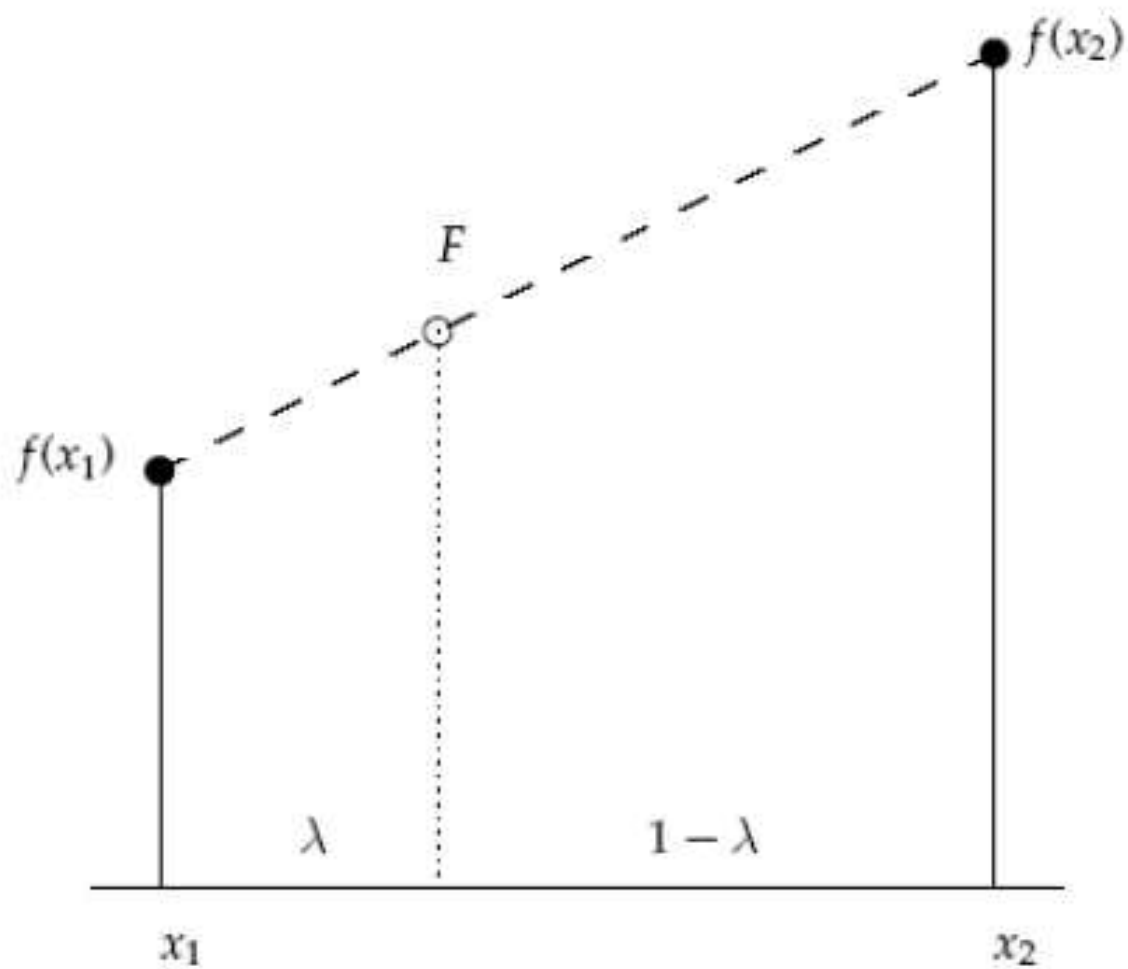


그림 6.5 선형적 보간 값 계산

6.2 영상 보간법 이번엔 1차원이 아닌 2차원 보간법

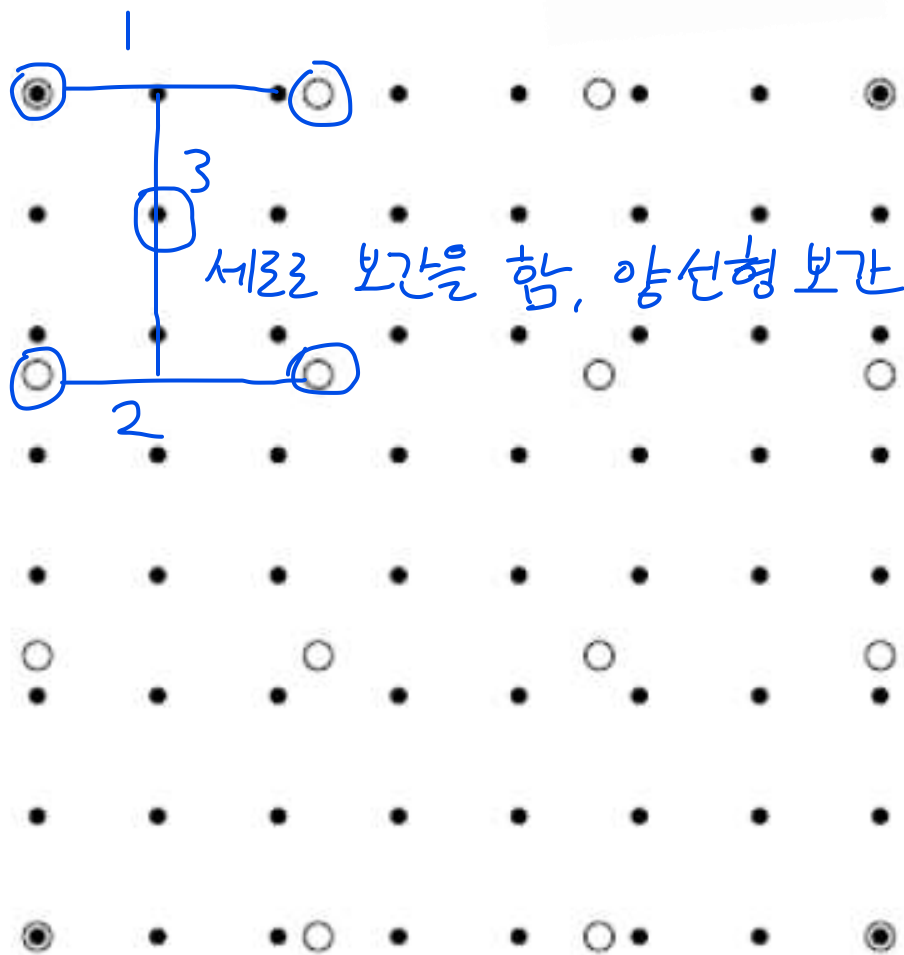


그림 6.6 영상 보간법

양선형보간법(bilinear interpolation)

$$f(x, y') = \mu f(x, y + 1) + (1 - \mu)f(x, y)$$

이 되고, 또

$$f(x + 1, y') = \mu f(x + 1, y + 1) + (1 - \mu)f(x + 1, y).$$

y' 의 열을 따라 계산하면,

$$f(x', y') = \lambda f(x + 1, y') + (1 - \lambda)f(x, y'),$$

이 되고 이들을 종합하면,

$$\begin{aligned} f(x', y') &= \lambda(\mu f(x + 1, y + 1) + (1 - \mu)f(x + 1, y)) \\ &\quad + (1 - \lambda)(\mu f(x, y + 1) + (1 - \mu)f(x, y)) \\ &= \lambda\mu f(x + 1, y + 1) + \lambda(1 - \mu)f(x + 1, y) + (1 - \lambda)\mu f(x, y + 1) \\ &\quad + (1 - \lambda)(1 - \mu)f(x, y). \end{aligned}$$

보간을 총 3번

$(x', y') = \left(\frac{1}{3}(7x - 4), \frac{1}{3}(7y - 4) \right)$
 순방향

$(x, y) = \left(\frac{1}{7}(3x' + 4), \frac{1}{7}(3y' + 4) \right)$
 역방향

3p A를 이용함

$p_1(x, x') = p_1(1, 1)$
 $p_2(x, x') = p_1(4, 8)$

4개의 영상 점 사이
 8개의 영상 점 사이

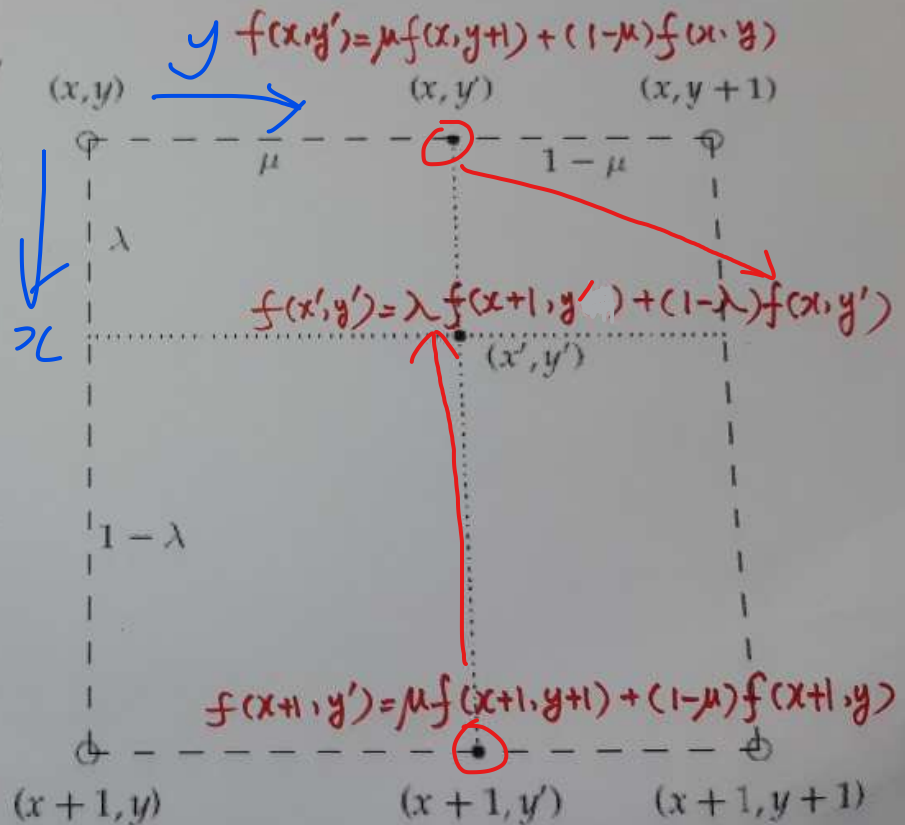
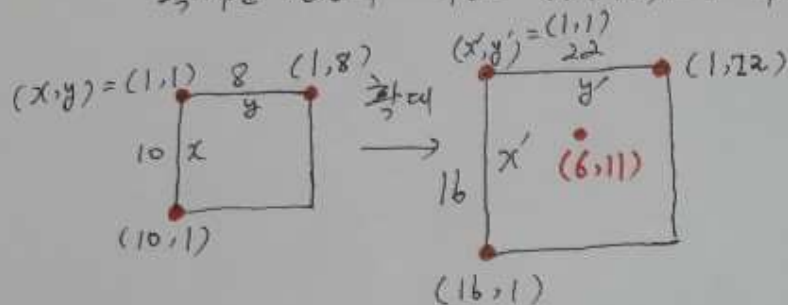


그림 6.7 4개의 영상 점 사이의 보간

문제) 10x8 영상(80 pixels)을 16x22 영상(352 pixels)으로 확대 했을 때
확대된 영상의 좌표 (6, 11)의 pixel 값은?



x에 대한 순방향 mapping

$$P_1(1,1) \rightarrow P_2(10,16)$$

$$x' = ax + b$$

$$1 = a + b \rightarrow a = 1 - b$$

$$16 = 10a + b \rightarrow 16 = 10(1 - b) + b$$

$$\text{정리하면 } a = \frac{5}{3}, b = -\frac{2}{3}$$

순방향식

$$x' = \frac{5}{3}x - \frac{2}{3}$$

역방향식

$$x = \frac{1}{5}(3x' + 2)$$

y에 대한 순방향 mapping

$$P_1(1,1) \rightarrow P_2(8,22)$$

$$y' = ay + b$$

$$1 = a + b \rightarrow a = 1 - b$$

$$22 = 8a + b$$

정리하면

$$a = 3, b = -2$$

순방향식

$$y' = 3y - 2$$

$$\text{역방향식 } y = \frac{1}{3}(y' + 2)$$

역방향식

$$x = \frac{1}{5}(3x' + 2)$$

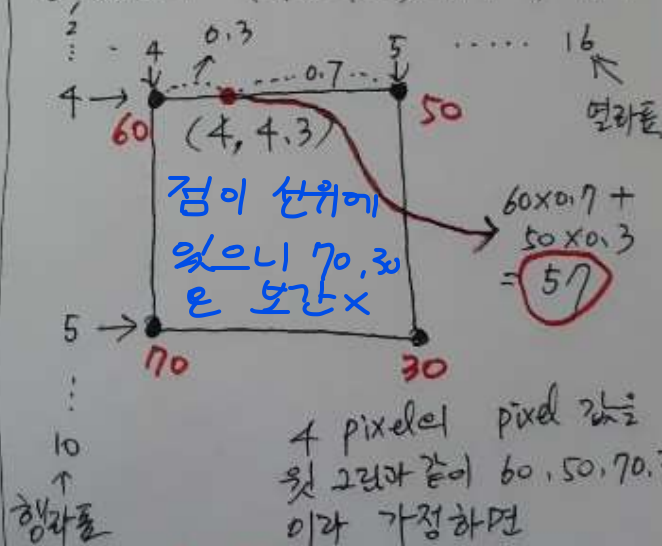
$$y = \frac{1}{3}(y' + 2)$$

예 좌표 (6, 11)을 대입하면

$$x = \frac{1}{5}(3 \times 6 + 2) = 4$$

$$y = \frac{1}{3}(11 + 2) = \frac{13}{3} = 4.3$$

따라서 (6, 11)의 pixel 값은
원 영상의 (4, 4.3)에서 가져온다



1은 1로 10은 16으로
자극의 양극점 변화

(6, 11)의 pixel 값은 57이다.

MATLAB은 `imresize` 함수를 가지고 있는데, 아래와 같이 처리할 수 있다.

`imresize(A,k,'method')`

k는 4로 늘려라

여기서 A는 영상의 형식이고, k는 척도계수이며, 'method'는 최근접 또는 양선형 중 하나이다. 또 다른 한 가지 방법은 `imresize`를 이용하는 것이다.

`imresize(A,[m,n],'method')`

여기서 `[m,n]`은 출력의 척도 사이즈를 나타낸다. 또 하나의 선택적 파라미터가 있는데, 그것은 사이즈를 축소하기 전에 영상에 적용할 저역통과필터의 형태와 사이즈를 선택할 수 있다. 상세한 것은 도움말을 보기 바란다.

```
>> c=imread('cameraman.tif');  
>> head=c(33:96,90:153); 세로축 33~96, 가로축 90~153 을 써어낸다  
>> imshow(head)  
>> head4n=imresize(head,4,'nearest'); imshow(head4n)  
>> head4b=imresize(head,4,'bilinear'); imshow(head4b)
```



그림 6.8 카메라맨의 머리 영상



(a)



(b)

그림 6.9 보간에 의한 척도변환 결과 (a)최근접법 (b)양선형법

6.3 일반적 보간법

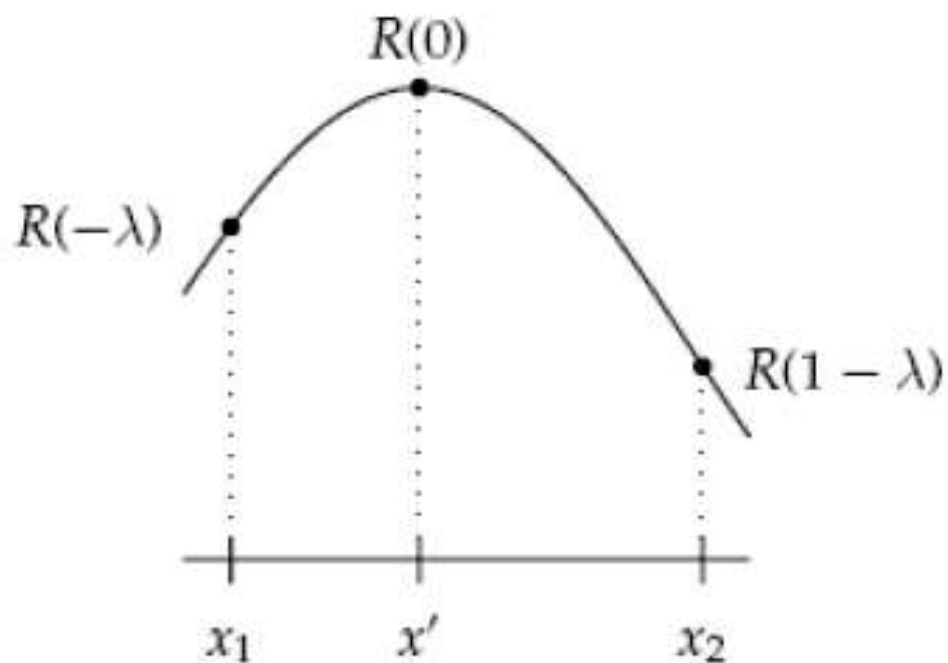


그림 6. 10 일반적 보간법의 원리

$$R_1(u) = \begin{cases} 1 + u & \text{if } u \leq 0 \\ 1 - u & \text{if } u \geq 0 \end{cases}$$

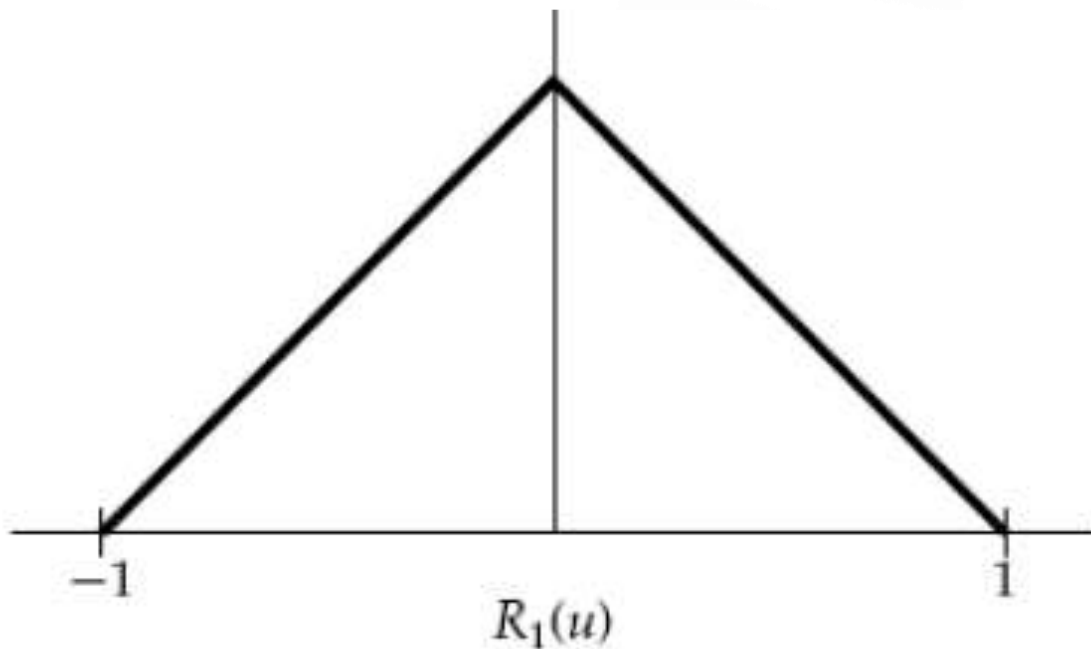
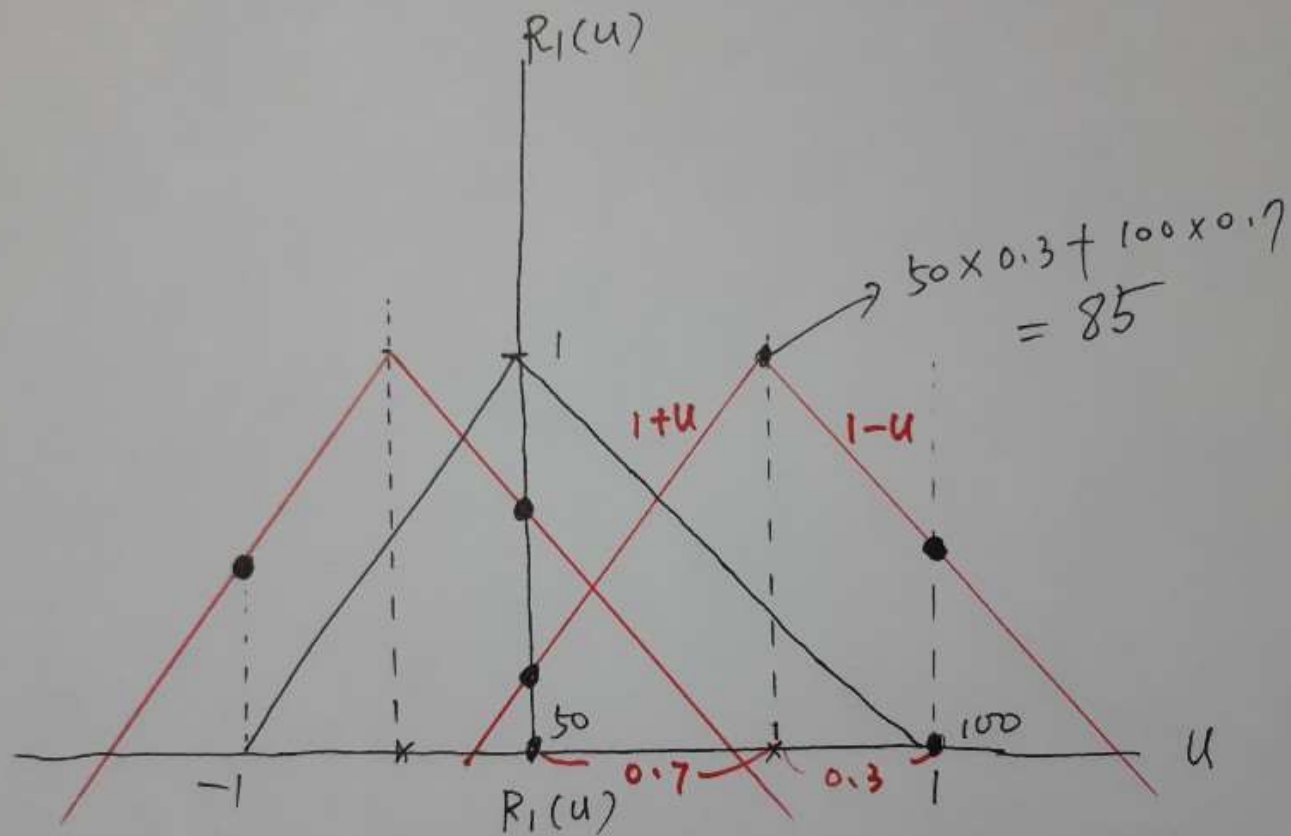


그림 6.11 2개의 보간함수의 예



3차곡선보간법(cubic interpolation)

가중치

$$R_3(u) = \begin{cases} 1.5|u|^3 - 2.5|u|^2 + 1 & \text{if } |u| \leq 1, \\ -0.5|u|^3 + 2.5|u|^2 - 4|u| + 2 & \text{if } 1 < |u| \leq 2. \end{cases}$$

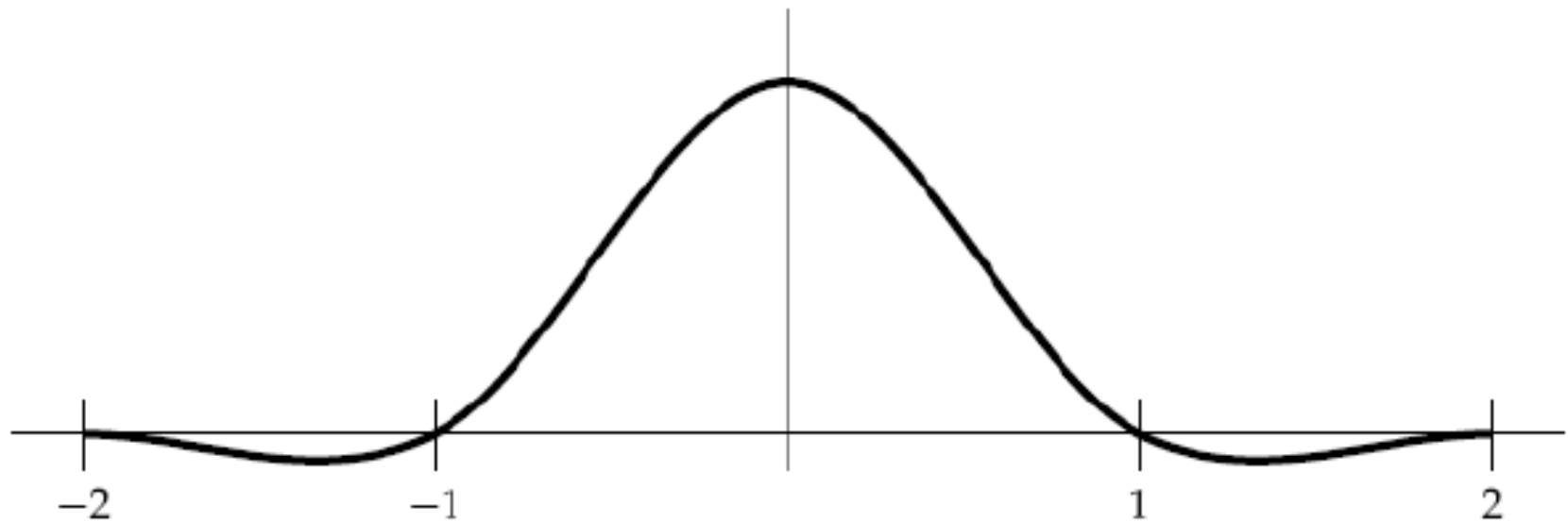


그림 6.12 3차 곡선함수 $R_3(u)$

$$\underline{f(x')} = R_3(-1-\lambda)f(x_1) + R_3(-\lambda)f(x_2) + R_3(1-\lambda)f(x_3) + \underline{R_4(2-\lambda)f(x_4)},$$

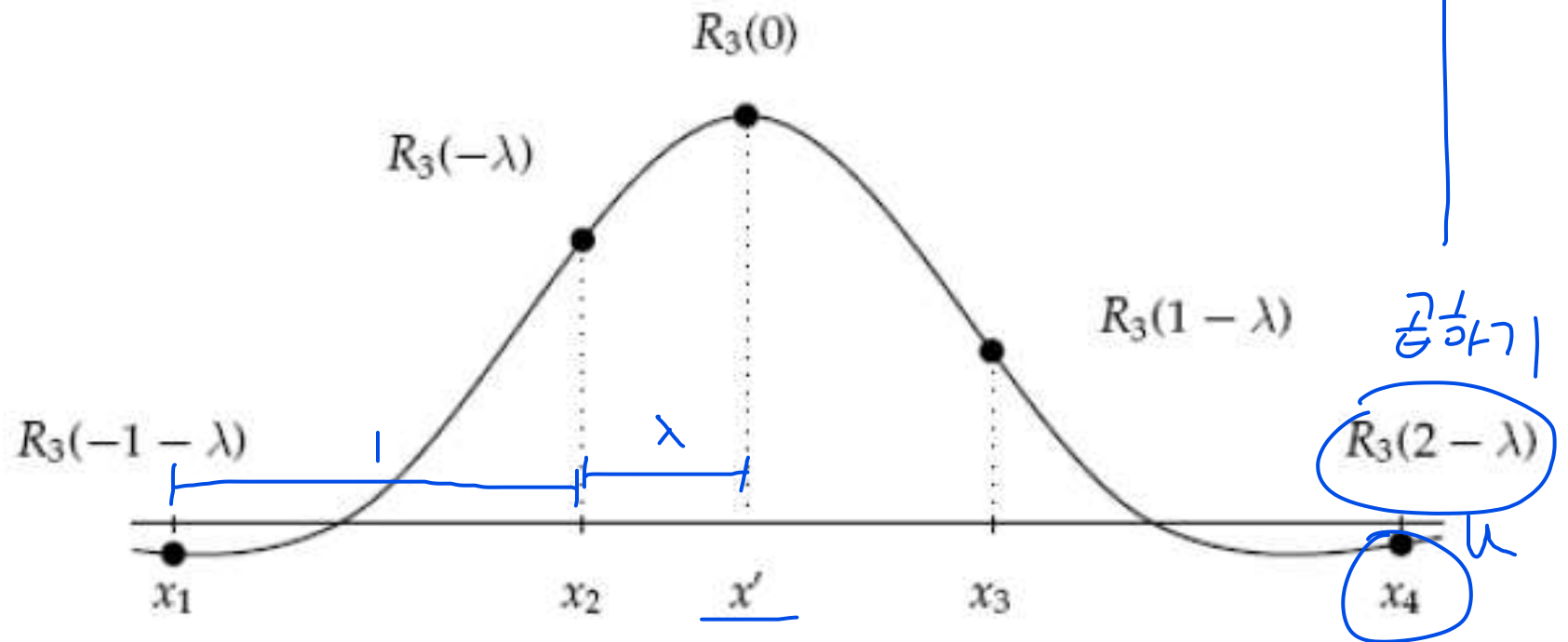
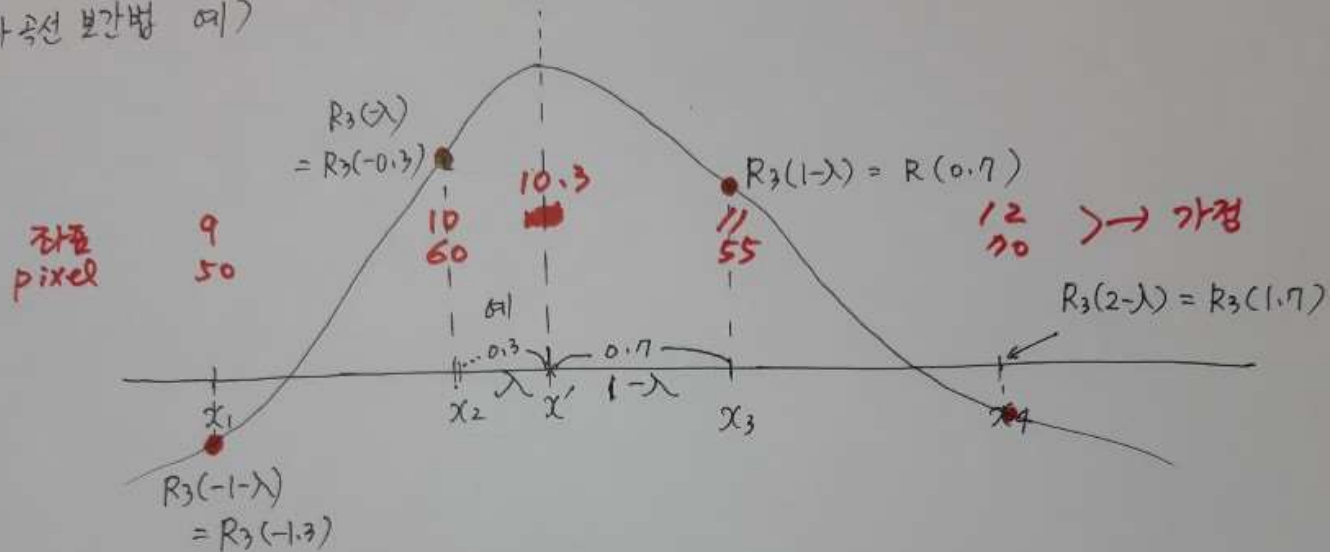


그림 6.13 $R_3(u)$ 에 의한 3차곡선보간법

3차 곡선 보간법 예)



$$R_3(-1.3) = -0.5 \times |-1.3|^3 + 2.5 \times |-1.3|^2 - 4 \times |-1.3| + 2 = -0.0735 \quad 1 < |u| \leq 2$$

$$R_3(-0.3) = 1.5 \times |-0.3|^3 - 2.5 \times |-0.3|^2 + 1 = 0.8155 \quad |u| \leq 1$$

$$R_3(0.7) = 1.5 \times |-0.7|^3 - 2.5 \times |-0.7|^2 + 1 = 0.2895 \quad |u| \leq 1$$

$$R_3(1.7) = -0.5 \times |-1.7|^3 + 2.5 \times |-1.7|^2 - 4 \times |-1.7| + 2 = -0.0315 \quad 1 < |u| \leq 2$$

만일 x_1, x_2, x_3, x_4 의 라플 값이 9, 50, 60, 55 이고

2개의 pixel 값이 각각 50, 60, 55, 70 이라 한다면

x' (역방향 매핑에 의해 정해진 라플) 은 10.3 이고 ($\lambda = 0.3$)

$$x' \text{에 대한 pixel 값은 } 50 \times (-0.0735) + 60 \times 0.8155 + 55 \times (0.2895) + 70 \times (-0.0315) = 58.97 \approx 59$$

x' 의 pixel 값

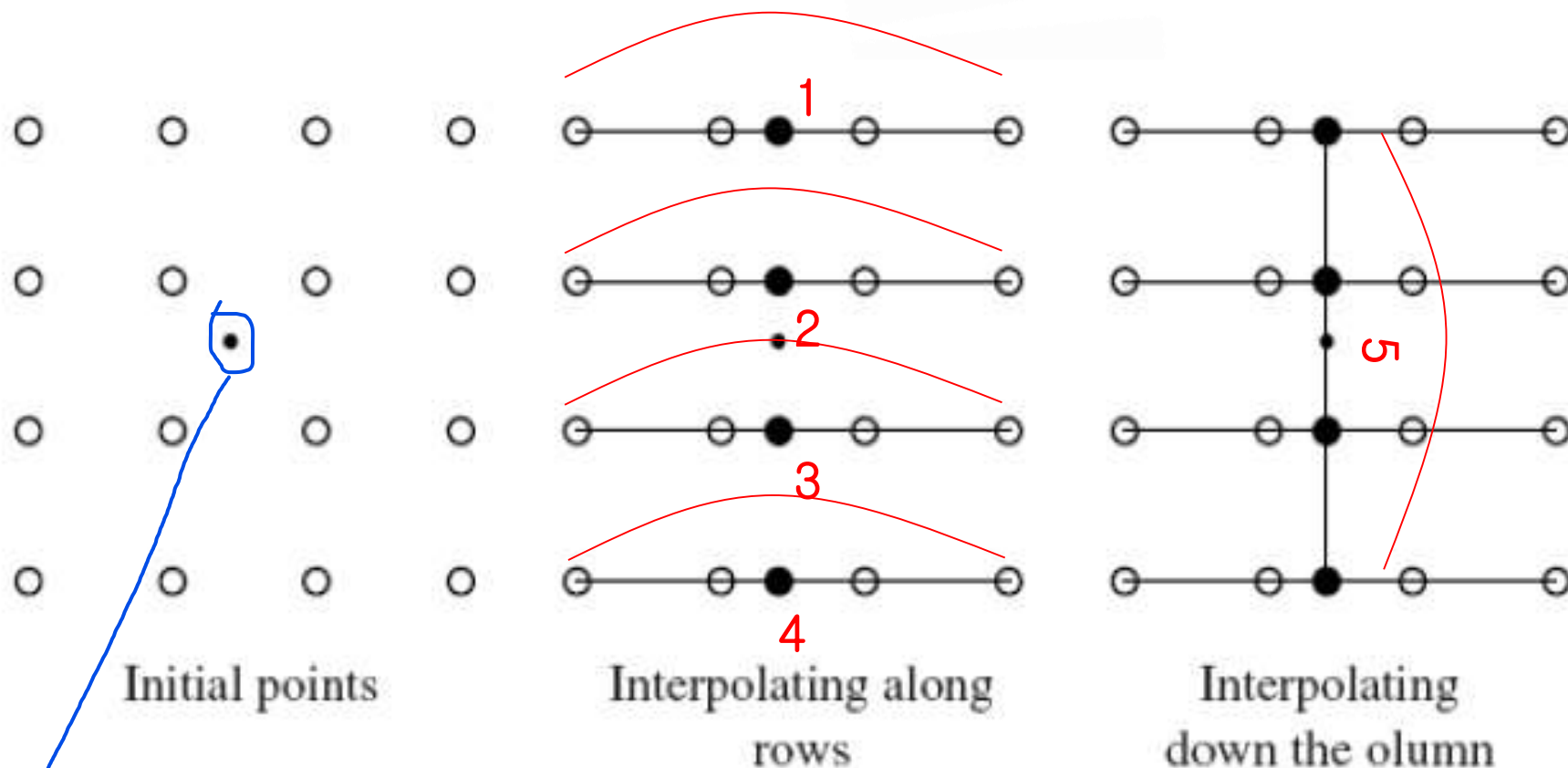


그림 6.14 2차원 영상의 3차곡선보간법
보간 5번

16개의 점에게
영상함수

MATLAB으로 이 보간을 처리 하기 위해 imresize 함수의 'bicubic'방법을 이용한다. 카메라맨 영상의 머리 부분을 확장하기 위하여 아래의 명령을 사용한다.

```
>> head4c=imresize(head,4,'bicubic');imshow(head4c)
```

이 결과는 그림 6.15와 같다.



그림 6.15 3차곡선보간에 의한 결과 영상

6.4 공간필터링에 의한 확대

2의 배수로 영상을 확대할 경우 빠르고 대략적인 보간은 선형필터링을 이용한다. 그 첫 단계로 이 매트릭스의 행과 열의 데이터 사이에 0을 채워 넣는다. 그러면 매트릭스의 사이즈는 가로 및 세로방향으로 2배가 되어 전체는 원래의 매트릭의 4배가 된다. m_2 가 m 에 0을 채운 매트릭스라고 하면 아래 식과 같다.

$$m_2(i, j) = \begin{cases} m((i+1)/2, (j+1)/2) & \text{if } i \text{ and } j \text{ are both odd,} \\ 0 & \text{otherwise.} \end{cases}$$

인덱스가 홀수면 (들 다)
 else 0 interleaving

$$m_2(i, j) =$$

최근접보간법

$$\begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

$m(1,1)$	0	$m(1,2)$	0	$m(1,3)$	0	$m(1,4)$	0	$m(1,5)$
0	0	0	0	0	0	0	0	0
$m(2,1)$	0	$m(2,2)$	0	$m(2,3)$	0	$m(2,4)$	0	$m(2,5)$
0	0	0	0	0	0	0	0	0
$m(3,1)$	0	$m(3,2)$	0	$m(3,3)$	0	$m(3,4)$	0	$m(3,5)$
0	0	0	0	0	0	0	0	0
$m(4,1)$	0	$m(4,2)$	0	$m(4,3)$	0	$m(4,4)$	0	$m(4,5)$
0	0	0	0	0	0	0	0	0

- $m = \text{magic}(4)$ 가로 세로 대각선 합이 같은 4×4


- $m =$

- | | | | |
|----|---|---|----|
| 16 | 2 | 3 | 13 |
|----|---|---|----|

- | | | | |
|---|----|----|---|
| 5 | 11 | 10 | 8 |
|---|----|----|---|

- | | | | |
|---|---|---|----|
| 9 | 7 | 6 | 12 |
|---|---|---|----|

- | | | | |
|---|----|----|---|
| 4 | 14 | 15 | 1 |
|---|----|----|---|



- >> k=[m;zeros(4,4)]

- k =

- 16 2 3 13

- 5 11 10 8

- 9 7 6 12

- 4 14 15 1

- | | | | |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |

- a2=reshape(k,4,8)

섞임

- a2 =

- 16 0 2 0 3 0 13 0
- 5 0 11 0 10 0 8 0
- 9 0 7 0 6 0 12 0
- 4 0 14 0 15 0 1 0

- `>> kk=[a2';zeros(8,4)]`

a2^① 행과 열 바꾸기

- `kk =`

- 16 5 9 4
- 0 0 0 0
- 2 11 7 14
- 0 0 0 0
- 3 10 6 15
- 0 0 0 0
- 13 8 12 1
- 0 0 0 0
- 0 0 0 0
- 0 0 0 0
- 0 0 0 0
- 0 0 0 0
- 0 0 0 0
- 0 0 0 0
- 0 0 0 0
- 0 0 0 0



- `reshape(kk,8,8)`

- `ans =`

- | | | | | | | | |
|----|---|----|---|----|---|----|---|
| 16 | 0 | 5 | 0 | 9 | 0 | 4 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | 0 | 11 | 0 | 7 | 0 | 14 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 3 | 0 | 10 | 0 | 6 | 0 | 15 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 13 | 0 | 8 | 0 | 12 | 0 | 1 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |



```
function out=zeroint(a)
%
% ZEROINT(A) produces a zero-interleaved version of the matrix A.
% For example:
%
%   a=[1 2 3;4 5 6];
%   zeroint(a)
%
%       1       0       2       0       3
%       0       0       0       0       0
%       4       0       5       0       6
%
% [m,n]=size(a); a2=reshape([a;zeros(m,n)],m,2*n);
% out=reshape([a2';zeros(2*n,m)],2*n,2*m)';
```

example)
 $a = [1 \ 2 \ 3; 4 \ 5 \ 6]; \rightarrow m \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$

$[m, n] = \text{size}(a);$ o/o $m=2, n=3$

$a2 = \text{reshape}([a; \text{zeros}(m, n)], m, 2*n);$

$\begin{array}{ccc} 1 & 2 & 3 \\ 4 & 5 & 6 \\ \hline 0 & 0 & 0 \\ 0 & 0 & 0 \end{array} \rightarrow \begin{array}{cccccc} 1 & 0 & 2 & 0 & 3 & 0 \\ 4 & 0 & 5 & 0 & 6 & 0 \end{array}$

$\text{out} = \text{reshape}([a2'; \text{zeros}(2*n, m)], 2*n, 2*m);$

$\begin{array}{cc} 1 & 4 \\ 0 & 0 \\ 2 & 5 \\ 0 & 0 \\ 3 & 6 \\ 0 & 0 \end{array}$

$\begin{array}{cc} 1 & 4 \\ 0 & 0 \\ 2 & 5 \\ 0 & 0 \\ 3 & 6 \\ \hline 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{array}$

$\begin{array}{cccc} 1 & 0 & 4 & 0 \\ 0 & 0 & 0 & 0 \\ 2 & 0 & 5 & 0 \\ 0 & 0 & 0 & 0 \\ 3 & 0 & 6 & 0 \\ 0 & 0 & 0 & 0 \end{array}$

$\begin{array}{cccccc} 1 & 0 & 2 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 4 & 0 & 5 & 0 & 6 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{array}$

```
>> m=magic(4)
```

```
m =
```

16	2	3	13
5	11	10	8
9	7	6	12
4	14	15	1

```
>> m2=zerosint(m)
```

```
m2 =
```

16	0	2	0	3	0	13	0
0	0	0	0	0	0	0	0
5	0	11	0	10	0	8	0
0	0	0	0	0	0	0	0
9	0	7	0	6	0	12	0
0	0	0	0	0	0	0	0
4	0	14	0	15	0	1	0
0	0	0	0	0	0	0	0

1	1	0
1	1	0
0	0	0

‘1’의 위치에 동일한 data가 반복된다.

```
>> filter2([1 1 0;1 1 0;0 0 0],m2)
```

```
ans =
```

16	16	2	2	3	3	13	13
16	16	2	2	3	3	13	13
5	5	11	11	10	10	8	8
5	5	11	11	10	10	8	8
9	9	7	7	6	6	12	12
9	9	7	7	6	6	12	12
4	4	14	14	15	15	1	1
4	4	14	14	15	15	1	1



```
K=imread('cameraman.tif');  
K1=zeros(K);  
imshow(K1);  
K2=filter2([1 1 0; 1 1 0; 0 0 0], K1);
```

Mathworks



Zero interleaving



Nearest neighbor



Bilinear



Bicubic

그림 6.16 공간필터링에 의한 확대 결과

6.5 스케일링에 의한 축소

영상을 축소한다는 것은 영상의 최소화라고 한다. 영상을 축소하는 한 가지 방법은 짝수 번째 화소 혹은 홀수 번째 화소를 지워서 없애는 것이다. 원 영상을 1/16로 만들려면 각각 i 와 j 방향으로 4번째 화소 값만 뽑아낸다. 이 방법을 서브샘플링(subsampling)이라 한다 이는 `imresize`의 최근접에 대응하고 구현하기 쉽다. 그러나 이 방법은 영상의 고주파성분의 특성이 좋지 않다. 예를 들어 흰 사각형에 하나의 원으로 구성되는 큰 영상을 처리하면 아래와 같다.

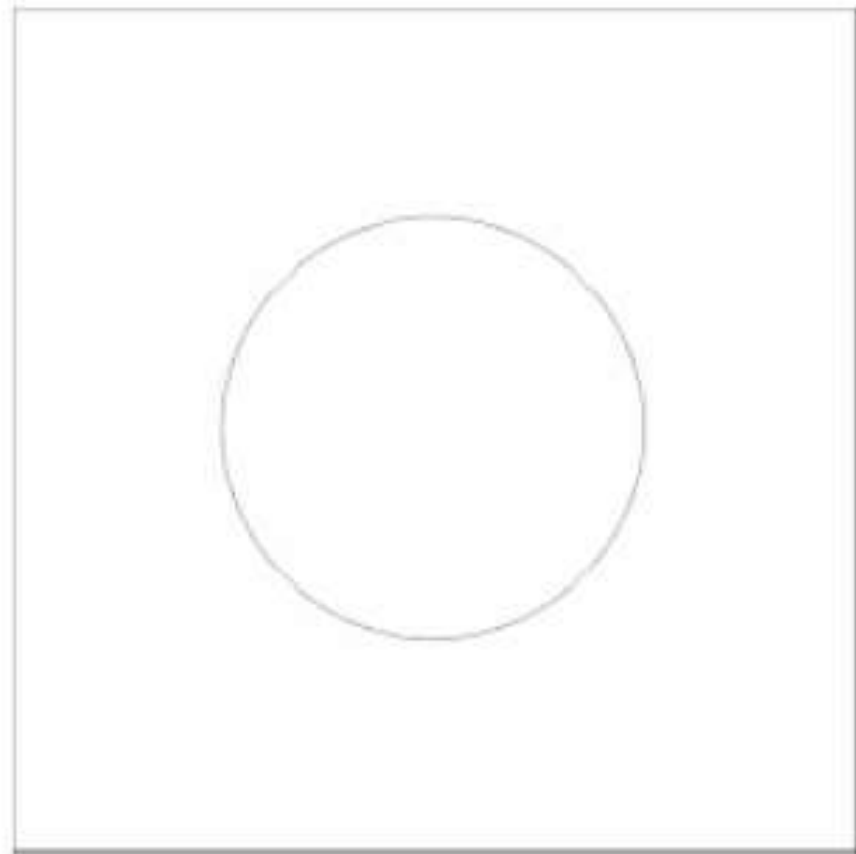
```
>> tr=imresize(t,0.25);
```

 default : 최근접 보간법

```
>> trc=imresize(t,0.25,'bicubic');
```



(a)



(b)

그림 6.18 영상의 최소화 (a) 최근접 방법 (b) 3차곡선 방법

안색음

색음

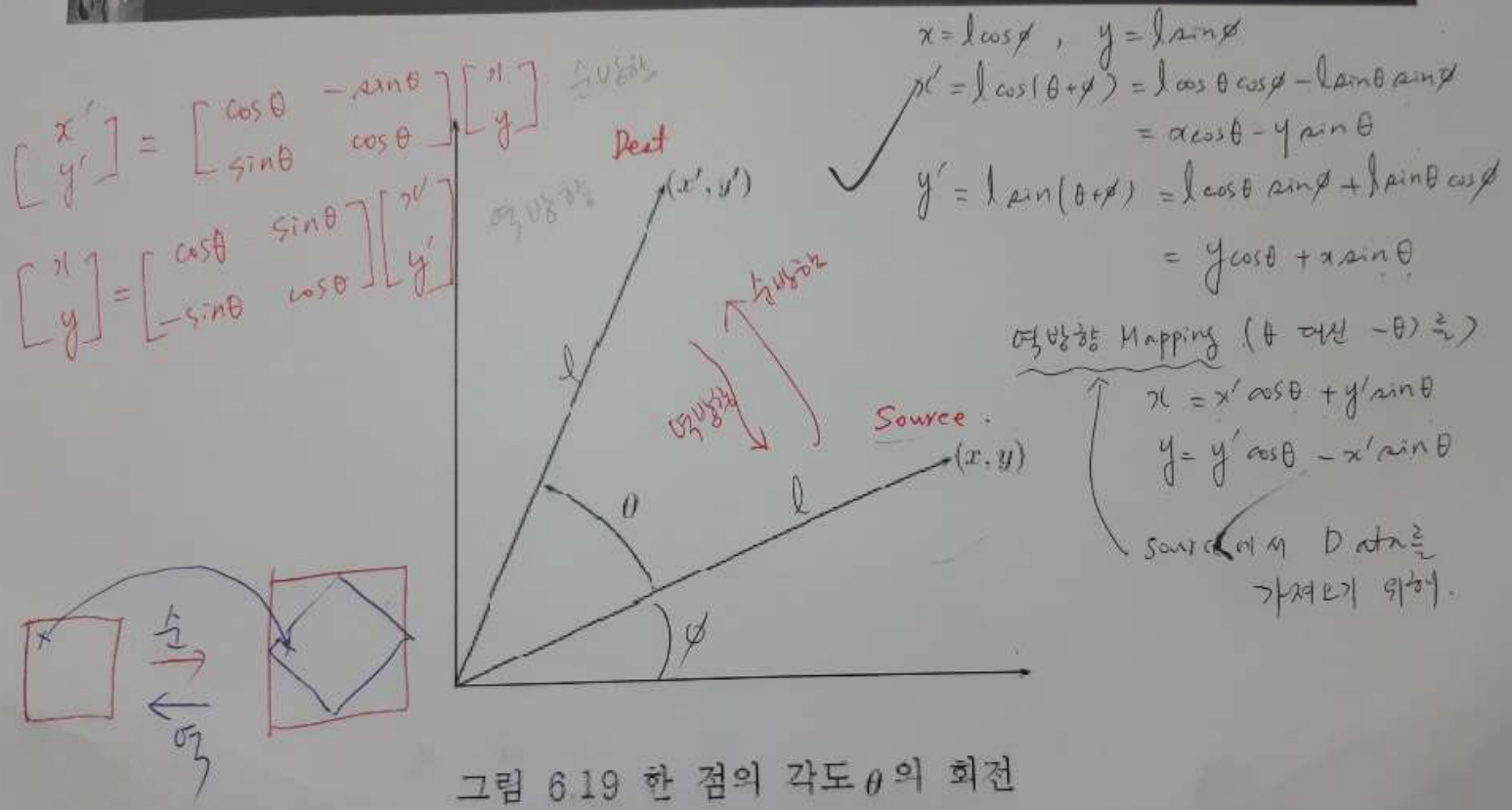
6.6 회전 처리

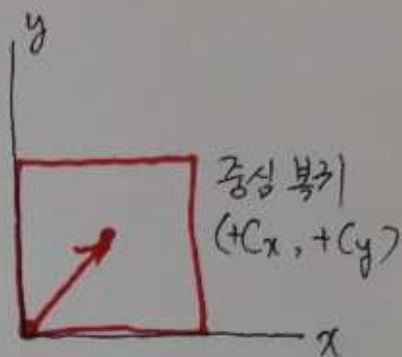
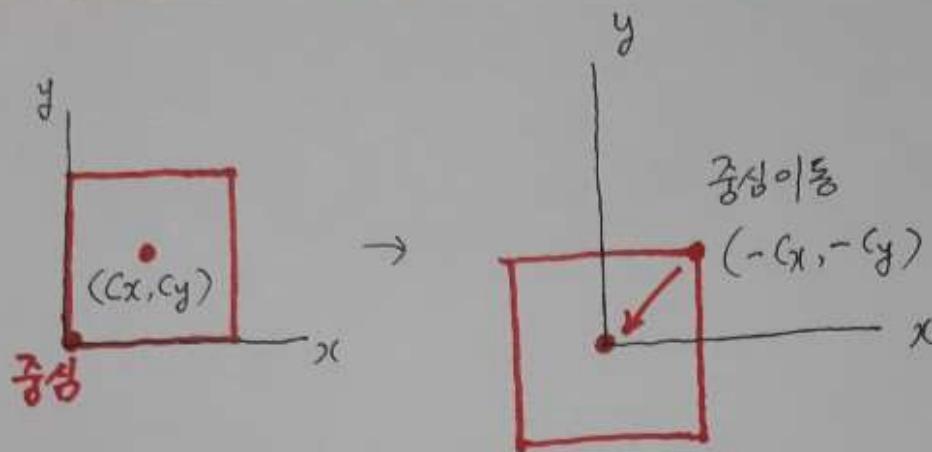
스케일링(척도변환)에 대한 보간처리에서 영상의 회전에 대하여 동일한 이론을 적용할 수 있다. 먼저, 그림 6.19와 같이 점(x,y)를 다른 점(x',y')까지 θ 만큼 반시계방향으로 회전되어 사상되는 것이 아래 식과 같이 매트릭스의 곱으로 얻어진다는 것을 기억하라.

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \cdot \text{순방향}$$

이와 같이 매트릭스가 직교하므로(그 역은 전치와 같다) 다음과 같이 표현할 수 있다.

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} \cos \theta & \sin \theta \\ -\sin \theta & \cos \theta \end{bmatrix} \begin{bmatrix} x' \\ y' \end{bmatrix} \cdot \text{역방향}$$





- 원점 중심 때
 - 순방향: $x' = x \cos \theta - y \sin \theta$
 $y' = y \cos \theta + x \sin \theta$
 - 역방향: $x = x' \cos \theta + y' \sin \theta$
 $y = y' \cos \theta - x' \sin \theta$
- 영상의 중심 기준 회전
 - 순방향: $x' = (x - C_x) \cos \theta - (y - C_y) \sin \theta$
 $y' = (y - C_y) \cos \theta + (x - C_x) \sin \theta$
 - 역방향: $x = (x' + C_x) \cos \theta + (y' + C_y) \sin \theta$
 $y = (y' + C_y) \cos \theta - (x' + C_x) \sin \theta$

MATLAB에서의 영상회전은 `imrotate`; 라는 명령을 이용하여 얻을 수 있는데, 그 구문은 아래와 같다.

`imrotate(image,angle,'method')`

여기서, `imresize`에서와 같이 최근접법, 양선형법 혹은 3차원곡선 보간법을 사용할 수 있다. `imresize`와 같이 파라미터가 생략되면 최근접 보간법을 적용하게 된다.

예를 들어 카메라맨 영상을 60° 회전하자. 이때 한번은 최근접법을 또 한번은 3차원 곡선보간법을 아래와 같이 처리한다.

```
>> cr=imrotate(c,60);  
>> imshow(cr)  
>> crc=imrotate(c,60,'bicubic');  
>> imshow(crc)
```

이 결과는 그림 6.24와 같다. 2개의 영상 사이에서 눈으로 확실히 구별은 할 수 없지만, 최근접 이웃화소 보간법은 약간 거친 영상이다.



(a)



(b)

그림 6.24 보간처리한 회전 (a)최근접보간법 (b)3차곡선보간법