REPORT

Lab02 : Image Composition using Alpha Blending

과목명	지능형영상처리
담당교수	유 훈 교수님
학과	융합전자공학과
학년	3학년
학번	201910906
이름	이학민
제출일	2023.05.11.

상명대학교
SANGMYUNG UNIVERSITY

I. 서론

지능형영상처리 10주차 수업에서 다룬 Lab02은 두 영상 A, B에 대해서 이미지A의 특정 블록을 복사해서 이미지B의 특정 위치에 붙여 넣어 영상을 합성하는 실습이다. 실제에서 아래의 'baboon.png'와 'lena.png'를 테스트 이미지로 사용하여 Image Composition을 수행하였고, 'baboon.png'의 왼쪽 눈을 'lena.png'의 왼쪽 눈 위치에 합성하는 과정을 보일 것이다.
Image Composition은 alpha blending의 개념이 확장된 것으로, 사용하는 Mask에 따라 결과의 성능이 달라지기 때문에 Mask에 따른 성능을 비교 및 분석할 수 있다. Mask는 사용자의 설계에 따라 다양한 형태를 가질 수 있고 본 실습에서 기본이 되는 Mask는 다음과 같다.

① Binary Circle Mask

우선 alpha blending의 개념과 실습 결과에 대해 설명하고 Mask의 기본적인 형태와 이미지 합성 결과를 알아본다. 또한 Binary Circle Mask에서 발생하는 문제점을 개선하기 위한 방법으로 다음과 같은 방법이 있다.

② Filtering using Low Pass Filter
③ Triangular Mask
④ Gaussian Mask

Matlab에서 각 경우에 대하여 Image Composition을 수행하였을 때 얻어진 결과 이미지와 작성한 코드 및 설명을 첨부하였다.

Test image_A - 'baboon.png'

Test image_B - 'lena.png'

II. 본론

1) alpha blending

Image Composition은 alpha blending과 같은 개념을 응용한 기술이기 때문에 서로 밀접한 관련이 있다. 따라서 alpha blending 실습을 먼저 진행하였다. alpha blending은 Image Averaging의 일반화된 형태로 각 이미지에 가중치를 부여한 Image Overlapping 기법이다. TV나 영화 등에서 alpha 값을 천천히 바꿔감에 따라 이미지가 서서히 변하는 효과를 구현할 수 있다.

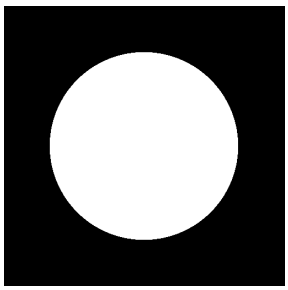
alpha값의 변화에 따른 합성 결과

Matlab code	

% Test_ alpha blending
clear;
clc;
imgA = imread("baboon.png");
imgB = imread("lena.png");
alp = 0:0.1:1;
for i=1:length(alp)
ap = alp(i);
img = (1-ap)*imgA + ap*imgB;
figure(i);
imshow(img);
pause(1);
end
	"baboon.png"과 "lena.png" 이미지를 읽어와서 alpha 값에 따라 두 이미지를 합성하는 코드이다.
	0부터 1까지 0.1 간격으로 alpha 값을 생성하고 생성한 alpha 값들을 하나씩 가져와서 반복적으로 이미지 합성을 수행한다.
	위 이미지는 alpha 값에 따라 합성된 이미지를 차례로 출력한 결과이다.

2) Fundamentals

① Binary Circle Mask



Binary Circle Mask

Result of Image Composition using Binary Circle Mask

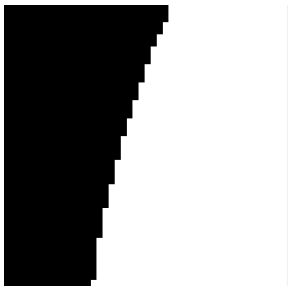


Image Scaling of Mask

Image Scaling of Result

Binary Circle Mask의 형태와 이를 이용한 이미지 합성의 결과이다. boundary를 기준으로 mask의 값이 0에서 1로 급격하게 변화하면서 이미지 합성이 부자연스럽게 이루어졌음을 확인하였다. 실제 이미지 합성이 부드럽게 이루어지기 위해서는 이러한 문제점이 개선된 mask를 설계하여야 하는데 mask의 값이 완만하게 변화하도록 수정하면 된다.
또한 mask와 이미지 합성 결과를 확대해보면 두 이미지의 경계가 매끄럽지 못한 모습을 볼 수 있는데, Low Pass Filter를 거쳐 날카로운 경계를 완화할 수 있다.
이미지 합성 결과의 개선에 대한 각 방법의 세부적인 실습 내용은 추후 다룬다. Binary Circle Mask에 대한 Matlab 코드와 설명은 다음과 같다.

Matlab code

```
imgA = imread("baboon.png");
imgB = imread("lena.png");

[row,col,dep] = size(imgA);
msksize = [row,col];
rad = min(msksize)/3;
imMsk = CircleMask(msksize, rad);
figure;
imshow(imMsk);

% image composition
imgA = double(imgA);
imgB = double(imgB);
imgMsk = double(repmat(imMsk, [1,1,3]));
imgR = imgA.*imgMsk + imgB.*(1-imgMsk);
figure;
imshow(imgR/255);

function imMsk = CircleMask(msksize,
rad)
% imMsk = CircleMask(msksize, rad)
% msksize = [row, col] of size of mask
% rad : radius for circle

rows = msksize(1);
cols = msksize(2);
center = msksize/2;

% Meshgrid
[x,y] = meshgrid(1:rows, 1:cols);

% Distance
dist = sqrt( (x-center(2)).^2 +
(y-center(1)).^2 );

% Binary mask
imMsk = dist <= rad;
end
```

두 이미지와 mask를 이용하여 이미지를 합성하는 코드이다. 먼저 두 이미지를 double 형식으로 변환한다. 그리고 mask를 3개의 채널로 확장하여 imgMsk 변수에 저장한다. 이렇게 만들어진 imgMsk 변수는 각 채널이 모두 같은 값을 가지는 3차원 배열이다. 합성된 이미지를 구하기 위해서 첫 번째 이미지와 mask, 그리고 두 번째 이미지와 (1-mask)를 곱하고 각각을 더한다. 이 결과를 imgR 변수에 저장하고, 이를 imshow 함수를 사용하여 화면에 표시한다. imshow 함수를 사용하기 전에는 imgR 배열을 0~1 사이의 값으로 정규화하여야 하므로, imgR 배열의 모든 값을 255로 나눠주었다.

이미지에 대한 Circle mask를 만드는 함수이다. 여기에서는 첫 번째 이미지의 크기 정보를 바탕으로 mask 크기를 정하고, mask 반지름을 mask 크기의 1/3로 지정하였다.

② Filtering using Low Pass Filter on Binary Circle Mask

Binary Circle Mask
by Low Pass Filter

Image Scaling
of Mask

Result of Image Composition using Binary Circle Mask by Low Pass Filter

Image Scaling of Result

필터 계수 값을 설정하여 Low Pass Filtering을 수행한 모습이다. mask에서 급격히 값이 변하는 것은 고주파 영역에 해당된다. 따라서 LPF 수행 시 mask의 경계가 매끄러워지는 결과를 얻을 수 있다. Filtering을 수행한 이미지를 보면 기존의 Binary Circle Mask의 거친 경계가 눈에 띄게 줄어들었음을 확인할 수 있다. 하지만 Triangular Mask나 Gaussian Mask처럼 mask의 값이 mask의 중심부로 갈수록 점진적으로 증가하는 형태가 아니기 때문에 이미지 합성 결과에서 다소 이질감이 느껴진다.

Matlab code

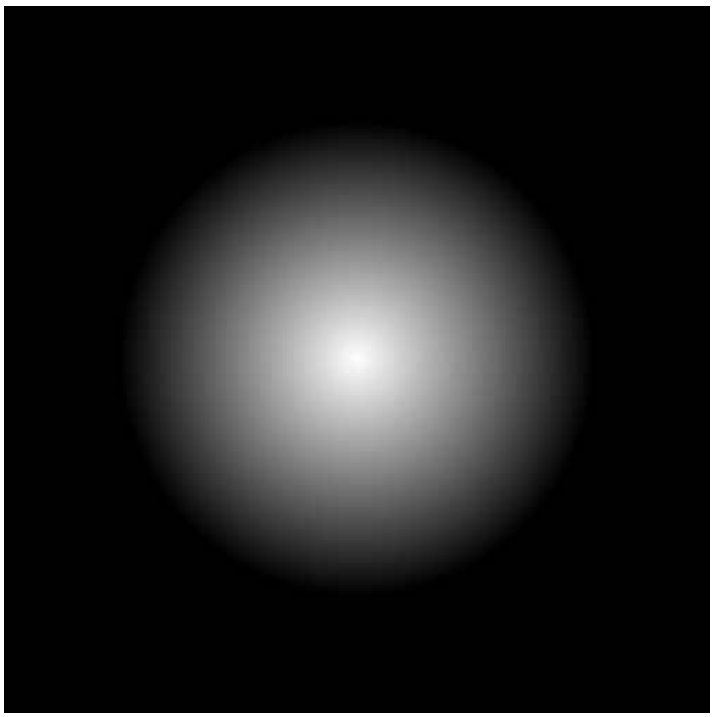
```
imgA = imread("baboon.png");
imgB = imread("lena.png");

[row,col,dep] = size(imgA);
msksize = [row,col];
rad = min(msksize)/3;
imMsk = CircleMask(msksize, rad);

h = [1 2 3 4 5 4 3 2 1];
h = conv(h,h);
hh = h'*h;
hh = hh/sum(hh(:));
imMsk = double(imMsk);
imMsk = imfilter(imMsk, hh);
figure;
imshow(imMsk);

% image composition
imgA = double(imgA);
imgB = double(imgB);
imgMsk = double(repmat(imMsk, [1,1,3]));
imgR = imgA.*imgMsk + imgB.*(1-imgMsk);
figure;
imshow(imgR/255);

function imMsk = CircleMask(msksize, rad)
% imMsk = CircleMask(msksize, rad)
% msksize = [row, col] of size of mask
% rad : radius for circle

rows = msksize(1);
cols = msksize(2);
center = msksize/2;

% Meshgrid
[x,y] = meshgrid(1:rows, 1:cols);

% Distance
dist = sqrt( (x-center(2)).^2 + (y-center(1)).^2 );

% Binary mask
imMsk = dist <= rad;
end
```

우선 h = [1 2 3 4 5 4 3 2 1]이라는 1차원 배열을 만들고, 이를 conv 함수를 사용하여 자기 자신과 convolution 연산을 하여 hh라는 2차원 배열을 만든다. 이렇게 만들어진 hh배열은 Gaussian 필터로 사용된다.

그 후, 이미지를 실수형으로 변환한 뒤 imfilter 함수를 사용하여 Gaussian 필터와 이미지의 convolution 연산을 수행한다. 이렇게 얻어진 이미지는 블러 처리된 이미지가 되며, imshow 함수를 사용하여 출력한다.

나머지는 앞서 설명한 2)-①의 Binary Circle Mask의 코드와 동일하다.

③ Triangular Mask



Triangular Mask

Result of Image Composition using Triangular Mask

mask의 경계면에서 급격한 값의 변화를 없애기 위해 Triangular Mask를 설계하였다. mask의 중심부로 갈수록 삼각파 함수를 따라 값이 1에 가까워지고 중심부에서 멀어질수록 0에 수렴한다. 이미지 합성 결과를 보았을 때 Binary Circle Mask를 이용한 결과보다 더 부드럽지만, 중심부 주위만 뚜렷하고 중심부에서 멀어질수록 점점 흐릿해져 퍼져 보이는 단점을 보완할 필요가 있다.

Matlab code

```
imgA = imread("baboon.png");
imgB = imread("lena.png");

[row, col, dep] = size(imgA);
msksize = [row, col];
rad = min(msksize) / 3;
imMsk = TriangleMask(msksize, rad);

% 주기와 진폭 설정
period = 2 * rad;
amplitude = 1;

% 삼각파 함수 구현
center = msksize / 2;
[x, y] = meshgrid(1:col, 1:row);
dist = sqrt((x - center(2)).^2 + (y - center(1)).^2);
imMsk = (dist <= rad) .* (1 - dist / rad);
figure;
imshow(imMsk);

% image composition
imgA = double(imgA);
imgB = double(imgB);
imgMsk = double(repmat(imMsk, [1, 1, 3]));
imgR = imgA .* imgMsk + imgB .* (1 - imgMsk);
figure; imshow(imgR/255);

function imMsk = TriangleMask(msksize, rad)
% imMsk = TriangleMask(msksize, rad)
% msksize = [row, col] of size of mask
% rad : radius for triangle mask
rows = msksize(1);
cols = msksize(2);
center = msksize / 2;

% Meshgrid
[x, y] = meshgrid(1:cols, 1:rows);

% Distance
dist = sqrt((x - center(2)).^2 + (y - center(1)).^2);

% Triangle mask
imMsk = (dist >= rad) .* 0 + (dist < rad) .* (1 - dist / rad);
end
```

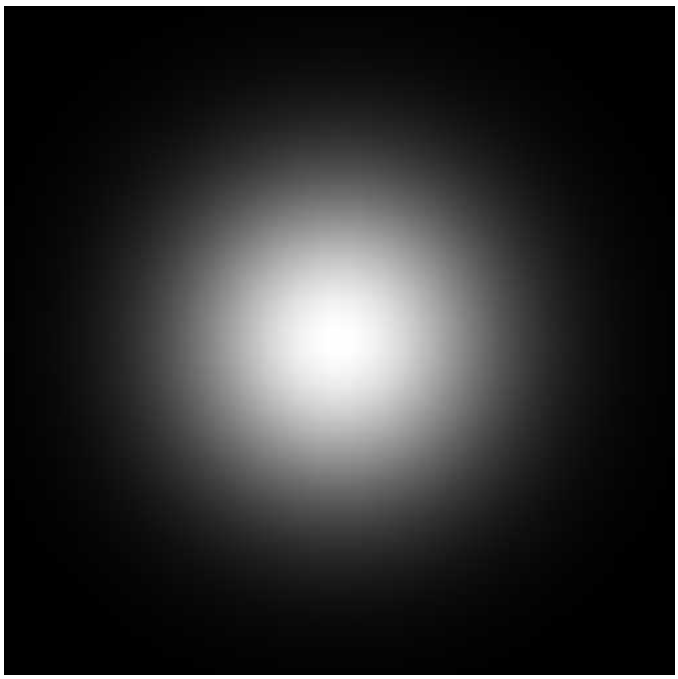
"baboon.png"와 "lena.png" 이미지를 읽어와서 중심을 기준으로 한 삼각형 모양의 mask를 생성하고, 이를 사용하여 이미지를 합성하는 코드이다.

TriangleMask 함수를 호출하여 주어진 크기 msksize와 반지름 rad를 기반으로 삼각형 모양의 mask imMsk를 생성한다.

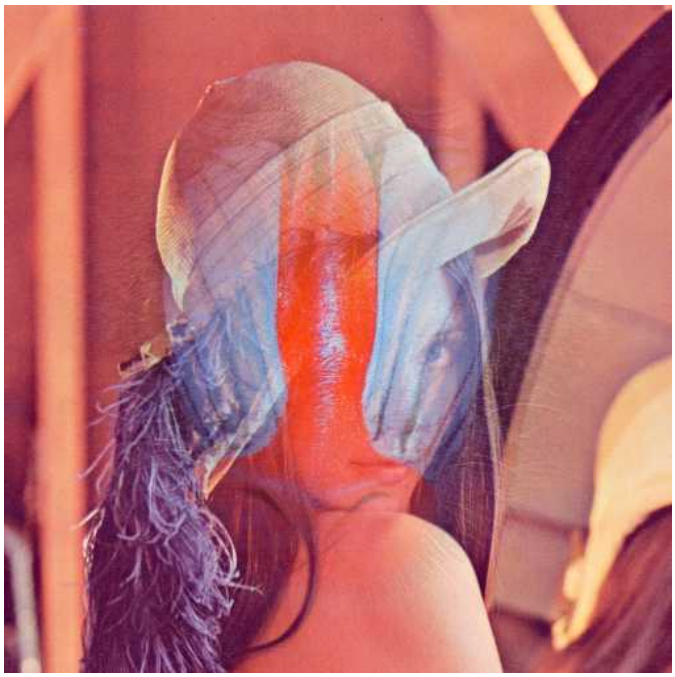
주기와 진폭은 mask에서 삼각파 함수를 사용할 때의 파라미터이다. 주기는 삼각파의 한 주기의 길이를 나타내며, 진폭은 삼각파의 최대 높이를 나타낸다. 주기를 반지름의 2배로 설정한 이유는 mask의 중심에서 원의 가장자리까지의 거리를 한 주기로 설정하기 위함이다. 중심에서 원의 가장자리까지 거리가 주기의 반절인 경우, 삼각파 함수의 값은 0이 되며, 이를 기준으로 원의 내부와 외부를 나누어 표현할 수 있다.
진폭을 1로 설정한 이유는 mask의 내부와 외부를 명확하게 구분하기 위해서이다. 진폭이 1인 경우, mask의 내부는 1의 값을 가지며, 외부는 0의 값을 가지게 된다.

마지막으로 거리가 반지름 이내인 영역은 1에서 거리를 나눈 값으로 계산하고, 그 외의 영역은 0으로 설정하여 mask의 값을 얻는다.

④ Gaussian Mask



Gaussian Mask



Result of Image Composition using Gaussian Mask

mask의 값이 Gaussian 함수를 따르도록 mask를 설계하였다. 앞서 보았던 Triangular와 비슷한 양상을 보이지만 Gaussian 함수는 1차 함수 꼴의 삼각파 함수에 비해서 값의 변화가 부드럽다. 결과 이미지를 보았을 때 Binary Circle Mask보다 훨씬 부드럽게 합성된 것을 알 수 있다. 하지만 Gaussian 함수의 특성 때문에 mask의 중심부만 강하게 강조되고 중심부에서 멀어질수록 많이 흐릿해진다. 따라서 넓게 퍼진 듯한 이미지 합성 결과를 얻게 되는 단점이 있다. 이는 Gaussian 함수의 sigma 값을 조절하여 제어 가능하다.

Matlab code

```
imgA = imread("baboon.png");
imgB = imread("lena.png");

[row,col,dep] = size(imgA);
msksize = [row,col];
sig = min(msksize)/6;
imMsk = GaussMask(msksize, sig);
figure;
imshow(imMsk);

% image composition
imgA = double(imgA);
imgB = double(imgB);
imgMsk = double(repmat(imMsk, [1,1,3]));
imgR = imgA.*imgMsk + imgB.*(1-imgMsk);
figure;
imshow(imgR/255);

function imMsk = GaussMask(msksize, sig)
% imMsk = CircleMask(msksize, rad)
% msksize = [row, col] of size of mask
% sig : sigma for gaussian function

rows = msksize(1);
cols = msksize(2);
center = msksize/2;

% Meshgrid
[x,y] = meshgrid(1:rows, 1:cols);

% Distance
% exp(- (x^2 + y^2)/2*sigma^2 )
dist   =   exp(   -((x-center(2)).^2   +
(y-center(1)).^2) / (2*sig^2) );

% Gaussian mask
imMsk = dist/max(dist(:));
end
```

imgA의 크기를 구하고, mask의 크기를 이미지의 크기와 같도록 설정한다. 그리고 mask에 사용할 Gaussian 필터의 sigma 값을 설정한다.

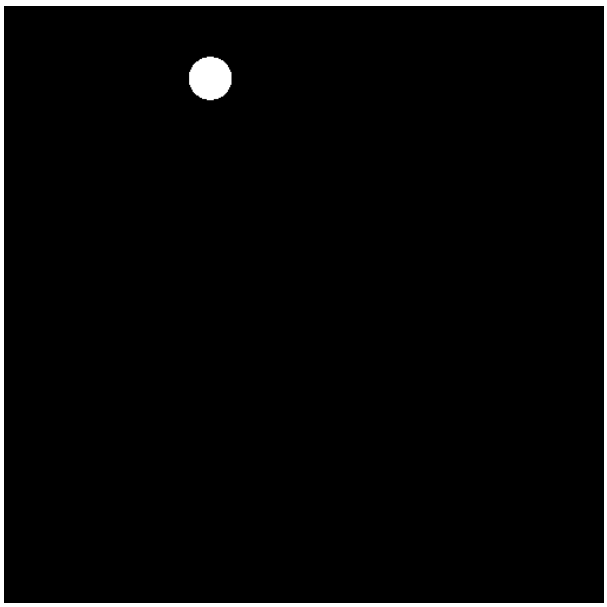
이미지 합성을 위해서 먼저 이미지를 double 형식으로 변환한다. 그리고 mask를 만들기 위해 Gaussian 필터 함수를 호출하여 mask 이미지를 생성한다. 마지막으로, 생성한 mask 이미지와 imgA, 그리고 imgB를 사용하여 합성 이미지를 만든다.

Gaussian 필터 함수는 함수 입력으로 mask 크기와 sigma 값을 받는다. 함수 내부에서는 mask 크기를 이용하여 Gaussian 분포를 따르는 필터 mask를 생성합니다. 이때, 필터 mask를 만들기 위해서는 먼저 중심 좌표를 계산하고, meshgrid 함수를 사용하여 격자점 좌표를 생성한다. 그리고 각 좌표와 중심 좌표 사이의 거리를 구한 후, 이를 Gaussian 함수에 대입하여 필터 mask를 생성한다. 마지막으로, 필터 mask를 최댓값으로 나누어 정규화한다.

3) Image Composition

앞서 알아본 mask 설계와 Filtering을 실제 Image Composition 실습에 적용하는 과정이다. 이번에는 imgA의 특정 부분을 imgB의 임의의 위치에 합성하는 실습이다. 사용하는 mask에 따라 이미지 합성 결과에 차이가 존재한다. imgA의 특정 부분을 mask를 통해 탐색하는 절차와 더불어 imgB의 임의의 위치를 설정하는 법을 Matlab 코드 설명란에 담았고 각 mask에 대하여 그 크기를 제어하는 방법과 크기에 따른 이미지 합성 결과도 포함하였다.
수행 결과 성능이 가장 좋지 않았던 Binary Circle Mask를 기준으로 각 개선 방안을 적용했을 때 이미지 합성 결과가 얼마나 향상되는지 비교·분석하였다.

① Binary Circle Mask



Binary Circle Mask

Result of Image Composition using Binary Circle Mask

Matlab code

```
imgA = imread("baboon.png");
imgB = imread("lena.png");

[rowA, colA, ~] = size(imgA);
[rowB, colB, ~] = size(imgB);

% 합성할 위치 설정
xOffset = 95; % x 방향으로의 이동 거리
yOffset = 205; % y 방향으로의 이동 거리

% 합성할 위치에서 imgA의 크기만큼의 영역 설정
startX = xOffset + 1;
startY = yOffset + 1;
endX = startX + colA - 1;
endY = startY + rowA - 1;

% 합성할 영역이 이미지 영역을 벗어나는 경우 클리핑
startX = max(startX, 1);
startY = max(startY, 1);
endX = min(endX, colB);
endY = min(endY, rowB);

msksize = [rowA, colA];
rad = min(msksize) / 28;
imMsk = CircleMask(msksize, rad);
figure;
imshow(imMsk);

% imgA를 imgB의 위치에 합성
imgA_double = im2double(imgA);
imgB_double = im2double(imgB);

% 합성할 영역에서만 연산 수행하도록 클리핑
imgB_clip = imgB_double(startY:endY, startX:endX, :);
imgA_clip = imgA_double(1:min(rowA, endY-startY+1), 1:min(colA, endX-startX+1), :);
imMsk_clip = imMsk(1:min(rowA, endY-startY+1), 1:min(colA, endX-startX+1));
imgB_clip = imgA_clip .* imMsk_clip + imgB_clip .* (1 - imMsk_clip);

% 합성된 영역을 원본 이미지에 삽입
imgB_double(startY:endY, startX:endX, :) = imgB_clip;
imgB = im2uint8(imgB_double);
figure;
imshow(imgB);
```

imgA의 특정 영역을 imgB의 지정된 위치에 합성하는 코드이다.

먼저, xOffset과 yOffset을 사용하여 합성할 위치를 설정한다. 합성할 위치를 지정하기 위해 값을 바꾸어가며 반복 수행한 결과, xOffset은 95, yOffset=205의 값을 찾았다.

startX, startY, endX, endY를 계산하여 합성할 영역을 지정하고, 이미지 영역을 벗어나는 경우에는 clipping 작업을 수행하였다.

또한 im2double 함수를 사용하여 imgA와 imgB를 double 형식으로 변환한 후, 합성할 영역에서만 연산을 수행하기 위해 이미지와 mask를 clipping 하였다. imgA_clip와 imMsk_clip를 사용하여 합성을 수행하고, imgB_clip에 합성된 결과를 저장하였다.

마지막으로, 합성된 영역을 원본 이미지 imgB_double에 삽입하고, im2uint8 함수를 사용하여 이미지를 uint8 형식으로 변환한 후, 결과를 시각화하였다.

이를 통해 imgA의 특정 영역이 imgB의 지정된 위치에 합성된 이미지를 얻을 수 있다.

mask의 원 크기를 조절하기 위해서는 rad 값을 바꾸어주면 되는데, ‘baboon’의 왼쪽 눈 이외의 정보가 최대한 담기지 않는 rad의 값을 찾기 위해 나누는 수를 바꾸어가며 반복 수행하였다. 결과적으로 rad는 min(msksize)를 28로 나누었을 때 가장 이상적인 결과를 보였다.

Matlab code	

function imMsk = CircleMask(msksize, rad)
% imMsk = CircleMask(msksize, rad)
% msksize = [row, col] of size of mask
% rad : radius for circle
 	
rows = msksize(1);
cols = msksize(2);
centerX = msksize(2) / 2.9;
centerY = msksize(1) / 8.0;
 	
% Meshgrid
[x, y] = meshgrid(1:rows, 1:cols);
 	
% Distance
dist = sqrt((x - centerX).^2 + (y - centerY).^2);
 	
% Binary mask
imMsk = dist <= rad;
 	
end
	mask의 원을 'baboon.png'의 왼쪽 눈에 맞추기 위해 기존의 center = msksize/2;을 centerX와 centerY로 구분하였다. centerX는 x축(cols)의 값과 관련, centerY는 y축(rows)의 값과 관련이 있다.
	원하는 위치에 mask의 원을 맞추기 위해 나누는 값을 바꾸어가며 반복 수행하였고 적절한 상수를 구하였다.
	centerX는 2.9, centerY는 8.0으로 각 값을 나누었을 때 mask의 원이 'baboon.png'의 왼쪽 눈에 위치하였고 mask를 통해 왼쪽 눈을 추출할 수 있었다.

Binary Circle Mask는 가장 단순한 구조의 Mask로 이미지 합성 성능이 매우 떨어진다. mask의 값이 0과 1로 극단적인 경우만 존재하기 때문에 2개의 이미지를 합성하는 경우 자연스러운 변화를 얻을 수 없다. 2) Fundamentals에서 알 수 있듯이 개선된 성능을 보이는 여러 mask를 사용해 근본적인 문제를 해결할 수 있고, mask를 확대했을 때 보이는 거친 경계는 LPF를 통해 다소 줄일 수 있다. 아래 그림은 rad 값에 따른 이미지 합성 결과이다.

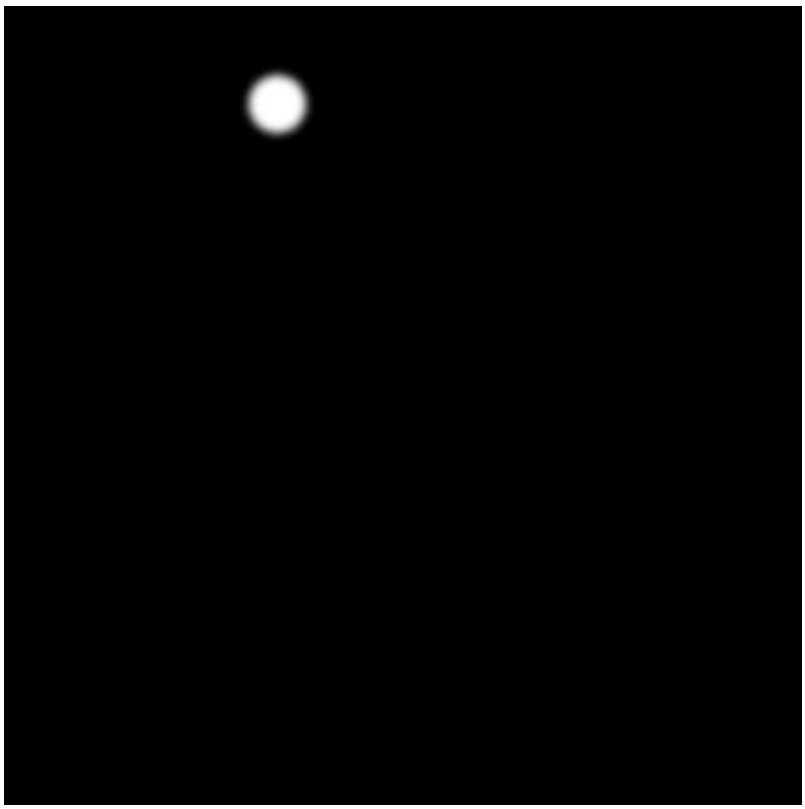
rad=min(msksize)/10

rad=min(msksize)/20

rad=min(msksize)/50

rad=min(msksize)/28

② Filtering using Low Pass Filter on Binary Circle Mask



Binary Circle Mask by Low Pass Filter

Result of Image Composition using Binary Circle Mask by Low Pass Filter

Matlab code

```
imgA = imread("baboon.png");
imgB = imread("lena.png");

[rowA, colA, ~] = size(imgA);
[rowB, colB, ~] = size(imgB);

% 합성할 위치 설정
xOffset = 95; % x 방향으로의 이동 거리
yOffset = 205; % y 방향으로의 이동 거리

% 합성할 위치에서 imgA의 크기만큼의 영역 설정
startX = xOffset + 1;
startY = yOffset + 1;
endX = startX + colA - 1;
endY = startY + rowA - 1;

% 합성할 영역이 이미지 영역을 벗어나는 경우 클리핑
startX = max(startX, 1);
startY = max(startY, 1);
endX = min(endX, colB);
endY = min(endY, rowB);

msksize = [rowA, colA];
rad = min(msksize) / 28;
imMsk = CircleMask(msksize, rad);

h = [1 2 3 4 5 4 3 2 1];
h = conv(h,h);
hh = h'*h;
hh = hh/sum(hh(:));
imMsk = double(imMsk);
imMsk = imfilter(imMsk, hh);
figure;
imshow(imMsk);

% imgA를 imgB의 위치에 합성
imgA_double = im2double(imgA);
imgB_double = im2double(imgB);

% 합성할 영역에서만 연산 수행하도록 클리핑
imgB_clip = imgB_double(startY:endY, startX:endX, :);
imgA_clip = imgA_double(1:min(rowA, endY-startY+1), 1:min(colA, endX-startX+1), :);
imMsk_clip = imMsk(1:min(rowA, endY-startY+1), 1:min(colA, endX-startX+1));
imgB_clip = imgA_clip .* imMsk_clip + imgB_clip .* (1 - imMsk_clip);
```

빨간색으로 표시한 코드에 대한 설명은 2)-② Filtering using Low Pass Filter on Binary Circle Mask의 Matlab 코드 설명란에 있다. convolution 연산을 반복적으로 사용할수록 더 매끄러운 결과를 도출할 수 있다.

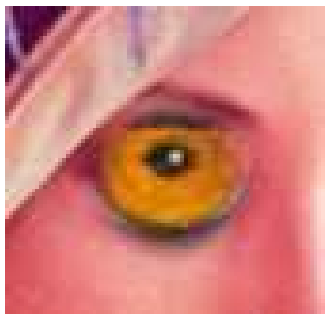
나머지 코드는 3)-① Matlab 코드 설명란의 내용과 같다.

Matlab code	

% 합성된 영역을 원본 이미지에 삽입
imgB_double(startY:endY, startX:endX, :) = imgB_clip;
imgB = im2uint8(imgB_double);
figure;
imshow(imgB);
 	
function imMsk = CircleMask(msksize, rad)
% imMsk = CircleMask(msksize, rad)
% msksize = [row, col] of size of mask
% rad : radius for circle
 	
rows = msksize(1);
cols = msksize(2);
centerX = msksize(2) / 2.9;
centerY = msksize(1) / 8.0;
 	
% Meshgrid
[x, y] = meshgrid(1:rows, 1:cols);
 	
% Distance
dist = sqrt((x - centerX).^2 + (y - centerY).^2);
 	
% Binary mask
imMsk = dist <= rad;
end
	3)-① Matlab 코드 설명란의 내용과 같다.

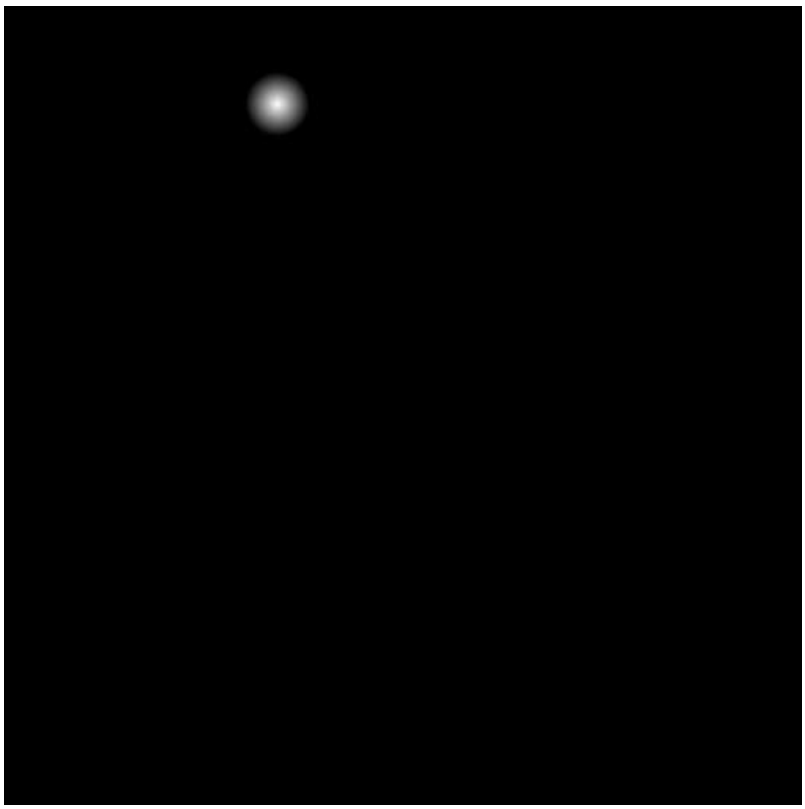
Low Pass Filter를 거친 Binary Circle Mask는 boundary의 고주파 영역이 다소 제거되어 매끄러운 모습을 보인다. Binary Circle Mask만 사용했을 때 'baboon.png'의 왼쪽 눈 주위의 초록색 피부가 제대로 제거되지 않아 합성 시 이질감을 발생하였다. Filtering 후의 결과를 보면 초록색 피부에 blur가 생겨 흐려지는 효과가 일어났다. 따라서 이미지 합성이 다소 자연스러워졌음을 확인하였다. 비교를 위해 확대한 사진을 첨부하였다.

Not Filtered (Only Binary Circle Mask)



Filtered (Low Pass Filtering)

③ Triangular Mask



Triangular Mask

Result of Image Composition using Triangular Mask

Matlab code

```
imgA = imread("baboon.png");
imgB = imread("lena.png");

[rowA, colA, ~] = size(imgA);
[rowB, colB, ~] = size(imgB);

% 합성할 위치 설정
xOffset = 95; % x 방향으로의 이동 거리
yOffset = 205; % y 방향으로의 이동 거리

% 합성할 위치에서 imgA의 크기만큼의 영역 설정
startX = xOffset + 1;
startY = yOffset + 1;
endX = startX + colA - 1;
endY = startY + rowA - 1;

% 합성할 영역이 이미지 영역을 벗어나는 경우 클리핑
startX = max(startX, 1);
startY = max(startY, 1);
endX = min(endX, colB);
endY = min(endY, rowB);

msksize = [rowA, colA];
rad = min(msksize) / 25;
imMsk = TriangleMask(msksize, rad);
figure;
imshow(imMsk);

% imgA의 특정 부분 추출
imgA_clip = imgA(1:min(rowA, endY-startY+1),
1:min(colA, endX-startX+1), :);

% imgA를 imgB의 위치에 합성
imgA_double = double(imgA_clip);
imgB_double = double(imgB);

% 합성할 영역에서만 연산 수행하도록 클리핑
imgB_clip = imgB_double(startY:endY, startX:endX,
:);
imMsk_clip = imMsk(1:min(rowA, endY-startY+1),
1:min(colA, endX-startX+1));
imgB_clip = imgA_double .* imMsk_clip + imgB_clip .*
(1 - imMsk_clip);

% 합성된 영역을 원본 이미지에 삽입
imgB_double(startY:endY, startX:endX, :) =
imgB_clip;
imgB = uint8(imgB_double);
figure;
imshow(imgB);
```

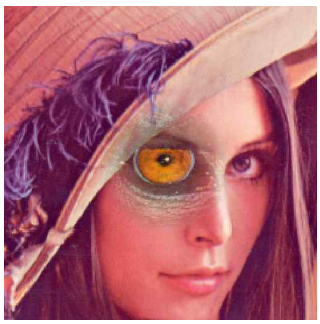
3)-① Matlab 코드 설명란의 내용과 같다.

다만 TriangleMask에서 가장 적합한 rad 값은 반복 수행 결과 min(msksize)를 25로 나눈 값임을 얻었다.

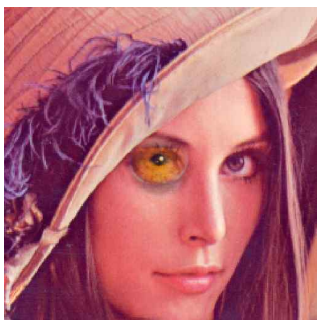
Matlab code	

function imMsk = TriangleMask(msksize, rad)
% imMsk = TriangleMask(msksize, rad)
% msksize = [row, col] of size of mask
% rad : radius for triangle mask
 	
rows = msksize(1);
cols = msksize(2);
centerX = msksize(2) / 2.9;
centerY = msksize(1) / 8.0;
 	
% Meshgrid
[x, y] = meshgrid(1:cols, 1:rows);
 	
% Distance
dist = sqrt((x - centerX).^2 + (y - centerY).^2);
 	
% Triangle mask
imMsk = (dist >= rad) .* 0 + (dist < rad) .*
(1 - dist / rad);
 	
end
	mask 위치 설정에 대한 설명은
	3)-① Matlab 코드 설명란의 내용과 같다.
	
	나머지 TriangleMask에 대한 내용은
	2)-③ Matlab 코드와 같다.

Triangular Mask 사용 시 mask의 원 크기를 적절하게 설정해주어야 한다. 원 크기가 너무 크면 'baboon'의 왼쪽 눈 이외의 필요 없는 정보가 과도하게 합성되어 흐릿한 잔상처럼 보이게 된다. 원 안에서 mask의 값은 삼각파 함수의 값을 따른다. 아래 그림은 rad 값에 따른 이미지 합성 결과이다.

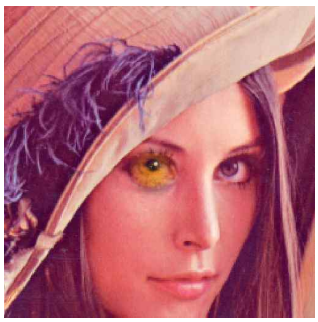


rad=min(msksize)/10



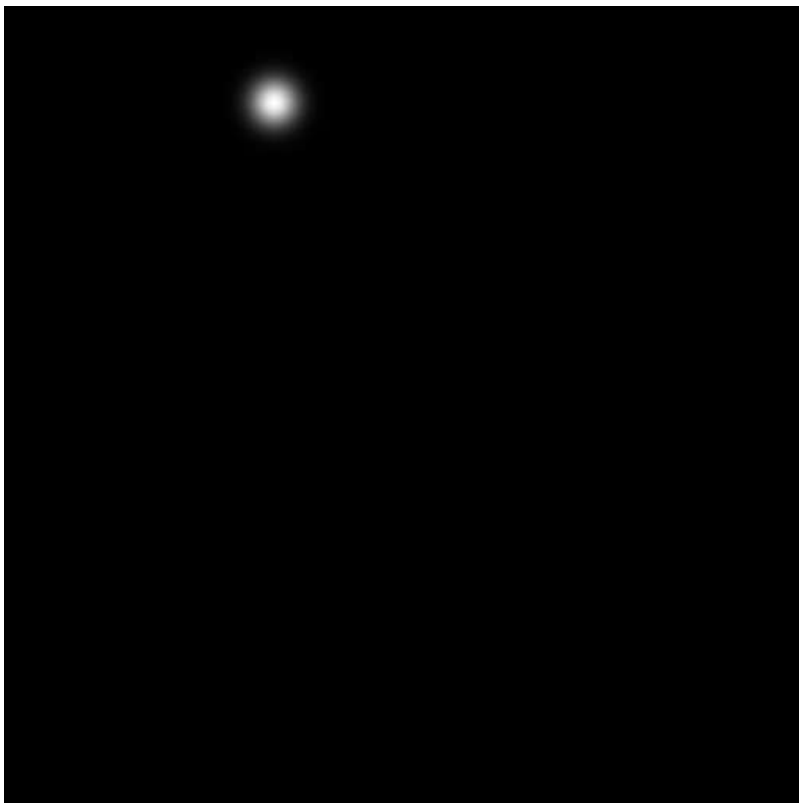
rad=min(msksize)/20

rad=min(msksize)/50



rad=min(msksize)/25

④ Gaussian Mask



Gaussian Mask

Result of Image Composition using Gaussian Mask

Matlab code

```
imgA = imread("baboon.png");
imgB = imread("lena.png");

[rowA, colA, ~] = size(imgA);
[rowB, colB, ~] = size(imgB);

% 합성할 위치 설정
xOffset = 95; % x 방향으로의 이동 거리
yOffset = 205; % y 방향으로의 이동 거리

% 합성할 위치에서 imgA의 크기만큼의 영역 설정
startX = xOffset + 1;
startY = yOffset + 1;
endX = startX + colA - 1;
endY = startY + rowA - 1;

% 합성할 영역이 이미지 영역을 벗어나는 경우 클리핑
startX = max(startX, 1);
startY = max(startY, 1);
endX = min(endX, colB);
endY = min(endY, rowB);
msksize = [rowA, colA];

sig = min(msksize)/50;
imMsk = GaussMask(msksize, sig);
figure;
imshow(imMsk);

% imgA를 imgB의 위치에 합성
imgA_double = double(imgA);
imgB_double = double(imgB);

% 합성할 영역에서만 연산 수행하도록 클리핑
imgB_clip = imgB_double(startY:endY, startX:endX,
:);
imgA_clip = imgA_double(1:min(rowA, endY-startY+1),
1:min(colA, endX-startX+1), :);
imMsk_clip = imMsk(1:min(rowA, endY-startY+1),
1:min(colA, endX-startX+1));
imgB_clip = imgA_clip .* imMsk_clip + imgB_clip .* (1
- imMsk_clip);

% 합성된 영역을 원본 이미지에 삽입
imgB_double(startY:endY, startX:endX, :) =
imgB_clip;
imgB = uint8(imgB_double);
figure;
imshow(imgB);
```

Gaussian 함수를 따르는 mask를 만들기 위해 사용자 함수 GaussMask를 생성하였다. 매개변수로 msksize와 sig를 사용하는데, sig는 Gaussian 함수의 sigma 값으로 Gaussian 함수의 폭을 결정한다. 따라서 sigma의 값을 조절함에 따라 mask의 크기가 바뀜을 알 수 있다. sig의 값이 커지면 Gaussian 함수의 폭이 좁아져 mask의 크기가 작아지고, sig의 값이 작아지면 Gaussian 함수의 폭이 넓어져 mask의 크기가 커진다. 이 실습에서 적절한 sig로 min(msksize)를 50으로 나눈 값을 사용하였다.

나머지 코드는 3)-① Matlab 코드 설명란의 내용과 같다.

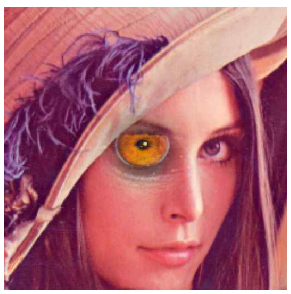
Matlab code

```
function imMsk = GaussMask(msksize, sig)
% imMsk = GaussMask(msksize, sig)
% msksize = [row, col] of size of mask
% sig : sigma for gaussian function

rows = msksize(1);
cols = msksize(2);
centerX = msksize(2) / 2.9;
centerY = msksize(1) / 8.0;

% Meshgrid
[x, y] = meshgrid(1:cols, 1:rows);

% Distance
% exp(- (x^2 + y^2) / (2 * sigma^2) )
dist = exp(-((x - centerX).^2 + (y - centerY).^2) / (2 * sig^2));

% Gaussian mask
imMsk = dist / max(dist(:));

end
```

mask 위치 설정에 대한 설명은
3)-① Matlab 코드 설명란의 내용과 같다.

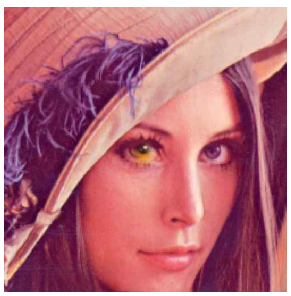
나머지 GaussMask에 대한 내용은
2)-④ Matlab 코드와 같다.

마찬가지로 Gaussian Mask도 mask의 크기를 적절하게 설정해야 한다. mask의 크기가 너무 크면 'baboon'의 왼쪽 눈 이외의 필요 없는 정보가 과도하게 합성되어 흐릿한 잔상이 넓게 퍼져 보이는 단점이 있다. mask의 값은 Gaussian 함수의 값을 따르며 Gaussian 함수는 sigma 값에 의해 폭이 결정되므로 sig 값을 조절하여 mask의 크기를 제어한다. 아래 그림은 sig 값에 따른 이미지 합성 결과이다.



sig=min(msksize)/30

sig=min(msksize)/70



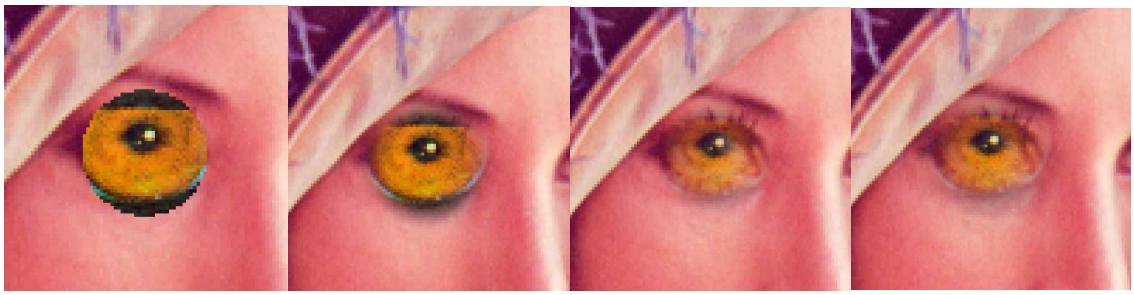
sig=min(msksize)/100

sig=min(msksize)/50

III. 결론

이번 실습에서는 alpha blending을 이용한 Image Composition을 수행하였다. 다양한 mask를 사용하여 이미지를 합성한 결과를 비교하면 다음과 같다.

(Binary Circle Mask -> LPF on Binary Circle Mask -> Triangular Mask -> Gaussian Mask 순)



합성된 이미지 결과를 확대하여 나열한 모습에서 알 수 있듯이 오른쪽으로 갈수록 보다 높은 성능의 이미지 합성이 이루어진다.

Binary Circle Mask는 가장 기본적인 구조의 Mask인데, 간단한 구조인 만큼 개선 사항이 많다. 단순히 0과 1로만 이루어진 Mask의 한계로 이미지 합성 진행 시 경계가 뚜렷하게 드러나며 갑작스러운 이미지 변화로 인해 강한 이질감이 느껴진다. 경계에서 값이 급격하게 변하는 것을 줄이기 위해 Low Pass Filtering을 수행하면 고주파 영역이 줄어들면서 Mask의 경계가 부드러워진다. 따라서 Filtering을 수행하기 전보다 매끄러운 합성 결과를 얻을 수 있다.

이미지 합성 성능을 개선하기 위해서 Filtering을 거치는 방법 외에 새로운 Mask를 설계하는 방안도 있다. 수업 시간에 Gaussian 함수를 이용하는 것이 가장 이상적인 값을 갖는 Mask를 설계하는 방법이라고 배웠다. 이를 검증해보기 위해 실습에서 Triangular Mask를 만들어 비교해 보았는데 눈에 뚜렷하게 보이는 차이를 얻지 못했다. 나름의 이유를 생각해보았는데, 현재 실습의 경우 동물과 사람의 '눈'이라는 비교적 작고 좁은 이미지를 합성하는 과정이다. 따라서 Gaussian Mask의 장점이 비교적 덜 드러났다고 할 수 있다. 수학적으로 Gaussian 함수의 부드러운 곡선이 삼각파 함수의 직선보다 Mask 설계할 때 유리함이 틀림없다. 따라서 합성하는 이미지의 크기가 커지고 넓어질수록 Gaussian Mask가 Triangular Mask에 비해 효과적인 성능을 보일 것이라고 예상한다.

Image Composition에서 Mask의 종류가 미치는 영향이 가장 크지만, 같은 Mask도 크기를 어떻게 지정하냐에 따라서 그 결과는 천차만별임을 실습을 통해 알 수 있었다. Image Composition을 수행하는 사용자가 최적의 Mask 크기를 결정하기 위해 radius 또는 sigma 값을 적절하게 설정하는 것이 매우 중요함을 알았다.