REPORT

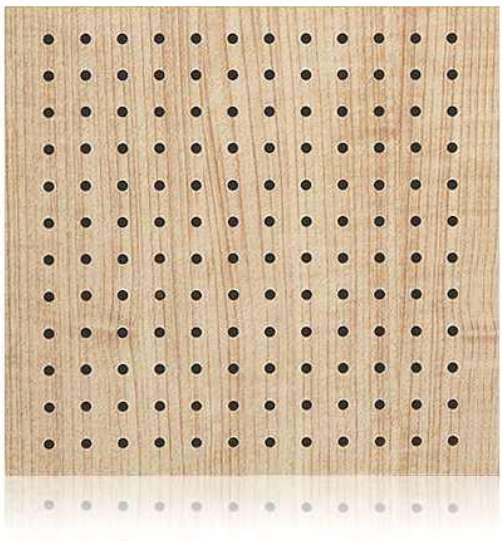
Lab03 : Pattern Recognition via Convolution

과목명	지능형영상처리
담당교수	유 훈 교수님
학과	융합전자공학과
학년	3학년
학번	201910906
이름	이학민
제출일	2023.05.22.

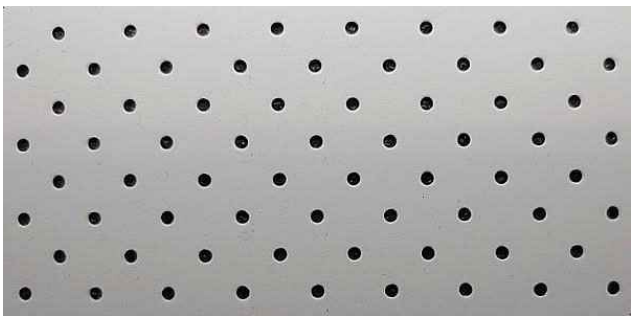
상명대학교
SANGMYUNG UNIVERSITY

I. 서론

지능형영상처리 11주차 수업에서 다룬 Lab03은 convolution을 통해 일정한 패턴을 인식하는 영상처리 실습이다. 테스트 이미지로 아래와 같은 2가지의 이미지를 사용하였다.



Test_cir02.png



Test_cir03.jpg

패턴인식은 이미지에서 패턴과 유사한 부분을 찾아내는 과정이다. 이때 패턴의 특징을 가진 영역은 일반적으로 패턴을 형성하는 픽셀값들이 상대적으로 높은 값을 가지게 된다. 패턴 이미지와 입력 이미지를 상호 비교하고 유사도를 계산하여 이러한 특징을 가진 부분을 찾는다. 특히 threshold 값에 따라 패턴의 인식 여부가 갈리는데, 다양한 threshold 값에 따라 인식되는 점의 개수가 어떻게 변화하는지 나열하였다. 특히 “Test_cir02.png”의 경우 이미지 하단에 비교적 흐릿한 부분이 존재하는데, 일반적인 방법으로 이곳에 위치한 점을 인식하기에는 다소 어려움이 발생하였다. 따라서 이를 해결하기 위한 간단한 절차도 소개할 것이다.

II. 본론

1) Ordinary Pattern Recognition

① Test_cir02.png

Matlab 코드 실행 시 출력되는 각 Figure이다. 특히 threshold 값에 따라 변하는 Figure5를 비교 및 분석하였다.

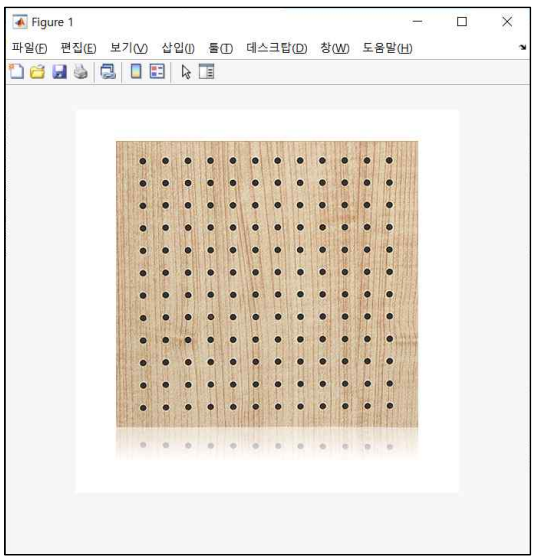


Figure 1 : 원본 이미지

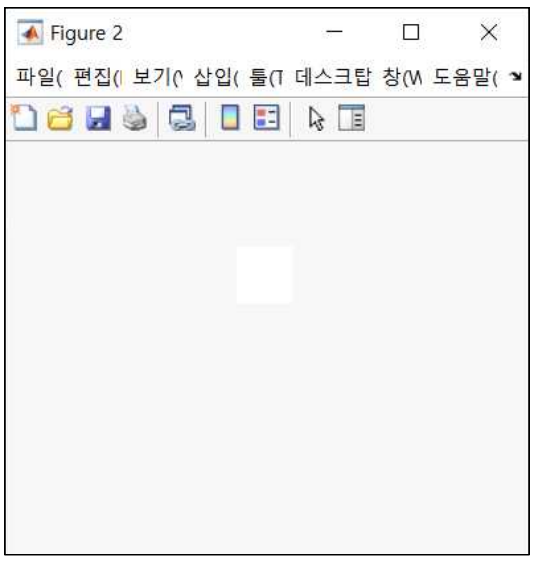
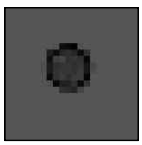


Figure 2 : 패턴으로 사용할 객체 이미지



아래의 Matlab 코드에서 imgY를 double형으로 형변환하지 않으면 Figure 2에서 다음과 같은 패턴이 얻어졌음을 볼 수 있다.

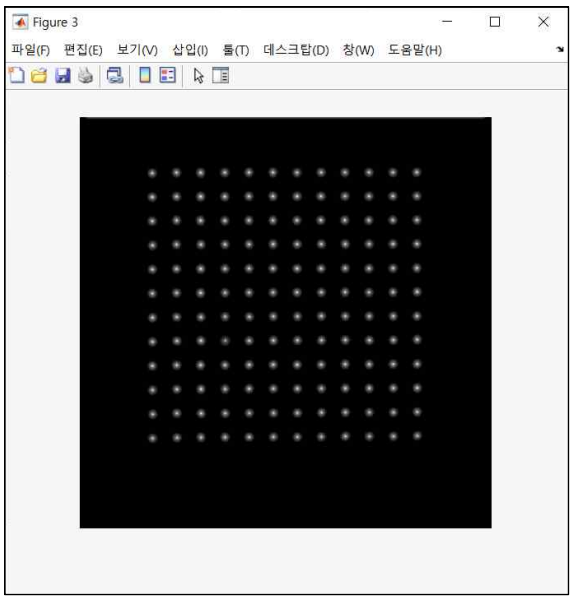


Figure 3 :
패턴인식 결과를 표시한 이미지로 입력 이미지와 패턴 이미지 간의 유사도를 계산하여 나온 결과를 보여준다.

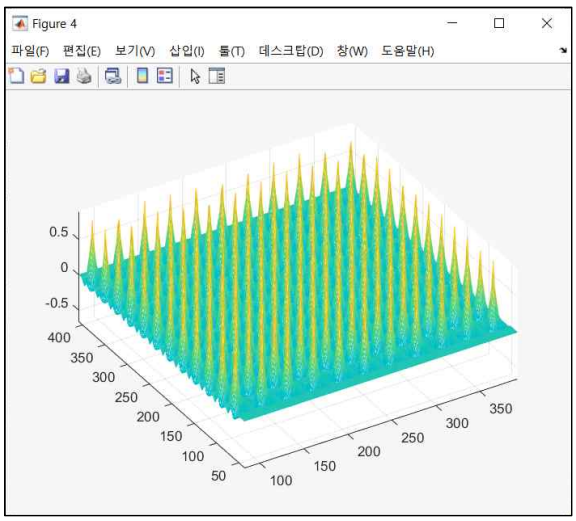
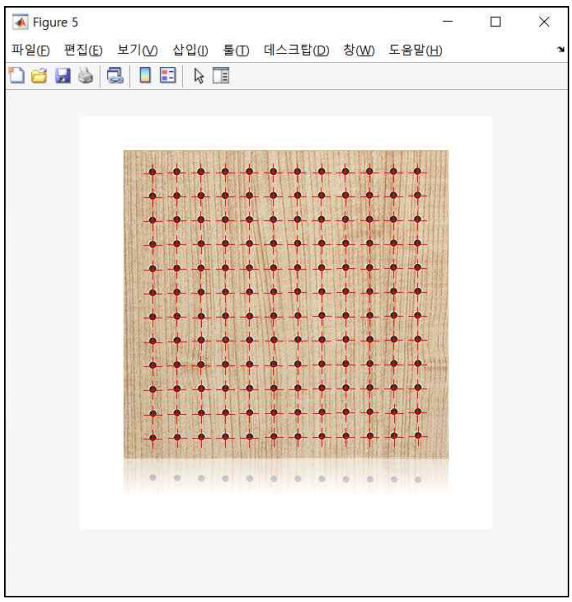
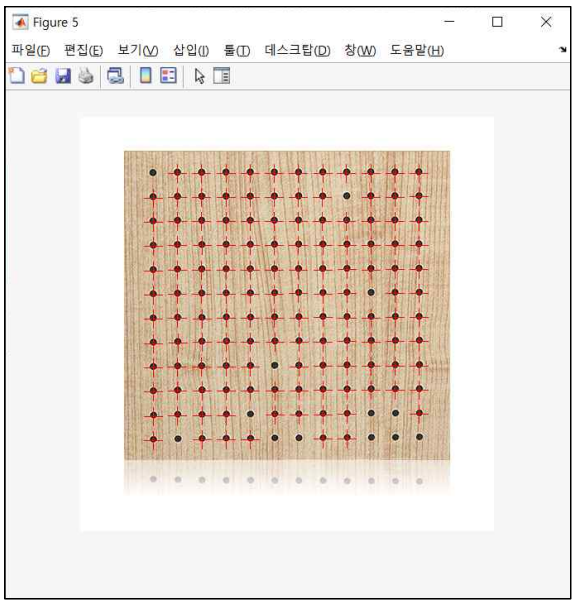


Figure 4 :
패턴인식 결과를 3D 형태로 표시한 메쉬 그래프로 입력 이미지에 대한 패턴 유사도의 분포를 시각화한 것이다.

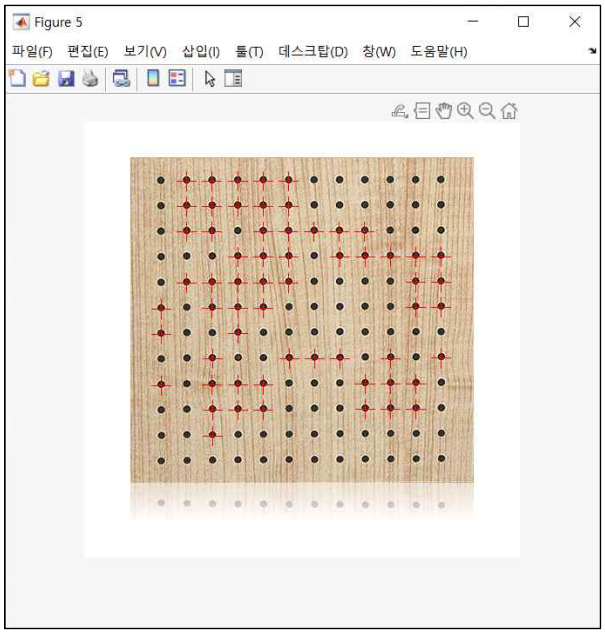
Figure 5 :
패턴인식 결과에 대해서 패턴이 감지된 위치에 빨간색 십자가 표시를 추가한 이미지이다.



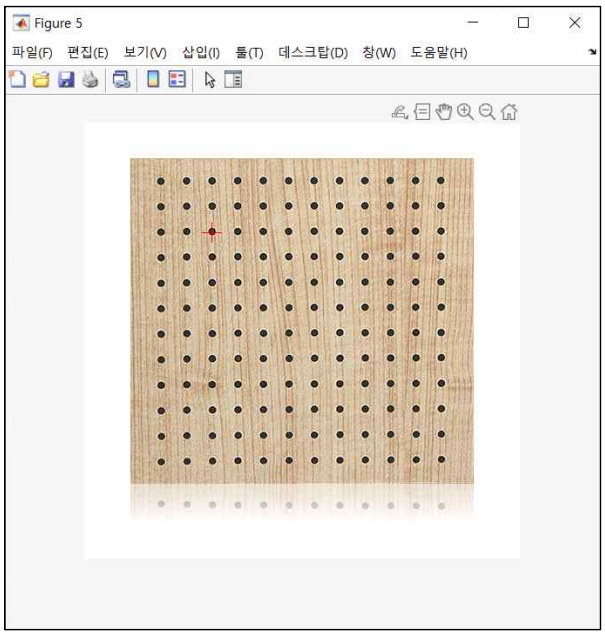
threshold = 0.6



threshold = 0.8



threshold = 0.9



threshold = 0.99

threshold는 패턴인식에서 사용되는 중요한 매개변수이다. 이 값은 패턴 이미지의 최댓값을 기준으로 설정되며, 패턴의 인식 여부를 결정하는 역할을 한다. 계산된 유사도 값이 threshold의 값보다 크면 해당 위치에서 패턴을 인식한 것으로 간주한다. threshold를 낮게 설정하면 낮은 유사도에서도 패턴을 인식할 수 있지만, 잘못된 인식이 발생할 가능성도 커진다. 반대로 threshold를 높게 설정하면 높은 유사도에서만 패턴을 인식하게 된다.
즉, threshold의 값은 패턴인식에서 얼마나 엄격하게 패턴을 인식할지를 결정하는 요소로 작용한다. 적절한 threshold의 설정은 패턴인식의 성능과 정확성에 큰 영향을 미치므로, 패턴의 특성과 입력 이미지의 특징을 고려하여 조정해야 한다.

Matlab code

```
% Pattern recognition via convolution
clear;
clc;

fname = "Test_cir02.png";
img = imread(fname);
figure(1); imshow(img);

imgY = (img(:,:,1) + img(:,:,2) + img(:,:,3))/3;
imgY = double(imgY);
obj = imgY(58:58+21,108:108+21);
figure(2); imshow(obj);

patt = flipud(fliplr(obj));
patt = patt/sum(patt(:));
patt = patt - mean(patt(:));

imgR = conv2(imgY, patt, 'same');
imgR = imgR/max(imgR(:));

figure(3); imshow(imgR);
figure(4); mesh(imgR);

num = 0;
rcpnt = zeros(num,2);
threshold = 0.8;
objsize = size(obj);
radr = ceil(objsize(1)/2);
radc = ceil(objsize(2)/2);

while(num<1000)
 [maxval, r, c] = max2d(imgR);
 if maxval < threshold
    break;
 end

num = num+1;
rcpnt(num,1) = r;
rcpnt(num,2) = c;
```

1. 이미지 로드 및 표시
imread 함수를 사용하여 이미지를 로드하고, imshow 함수를 사용하여 로드한 이미지를 표시한다.

2. 이미지 전처리
RGB 이미지를 그레이스케일로 변환하기 위해 R, G, B 채널의 평균값을 계산하고, 이미지를 double 자료형으로 변환한다.

3. 패턴 추출
이미지에서 패턴을 추출하기 위해 특정 영역을 선택한다. imgY에서 58:58+21 행과 108:108+21 열에 해당하는 영역을 선택하여 패턴으로 설정하고, 해당 패턴을 표시한다.

4. 패턴 컨볼루션
패턴을 컨볼루션하기 위해 flipud와 fliplr 함수를 사용하여 패턴을 상하좌우로 반전시킨다. 패턴을 정규화하기 위해 패턴의 합으로 나누고, 평균값을 빼준다. 그 후, conv2 함수를 사용하여 이미지 imgY와 패턴 patt를 컨볼루션하여 결과 이미지 imgR을 생성한다. 마지막으로, imgR을 최댓값으로 정규화한다.

5. 패턴인식 및 제거
imgR에서 패턴을 인식하고, 패턴의 최댓값과 임계값 threshold를 비교하여 패턴을 인식하는 반복문을 수행한다. 패턴이 인식되면 rcpnt에 해당 좌표를 저장하고, 패턴을 제거하기 위해 해당 영역을 0으로 설정한다. 이 과정은 최대 1000번 반복된다.

Matlab code

```
 % Erase
  rs = r-radr;
  re = r+radr;
  cs = c-radc;
  ce = c+radc;
  imgR(rs:re, cs:ce) = 0;
end

rad = radr;
color = [255, 0, 0]; % Red
imgC = DrawCross(img, rcpnt, rad, color);
figure(5); imshow(imgC);

function [maxval, r, c] = max2d(img)
%
 [row, col] = size(img);
 img = img';
 vec = img(:);

 [maxval, ind] = max(vec);
 r = floor((ind-1)/col);
 c = (ind-1) - r*col;
 r = r+1;
 c = c+1;
end

function [imgC] = DrawCross(img, rcpnt, rad,
color)
%
%

 imgC = img;
 [num, wid] = size(rcpnt);

 for n = 1:num
  r = rcpnt(n,1);
  c = rcpnt(n,2);

  imgC(r-rad:r+rad, c, 1) = color(1);
  imgC(r, c-rad:c+rad, 1) = color(1);

  imgC(r-rad:r+rad, c, 2) = color(2);
  imgC(r, c-rad:c+rad, 2) = color(2);

  imgC(r-rad:r+rad, c, 3) = color(3);
  imgC(r, c-rad:c+rad, 3) = color(3);
end

end
```

6. 결과 표시

인식된 패턴에 교차선을 그려서 시각적으로 표시한다. DrawCross 함수를 사용하여 이미지 img에 인식된 패턴 좌표 rcpnt, 반지름 rad, 색상 color를 기반으로 교차선을 그린다. 마지막으로, 결과 이미지 imgC를 표시한다.

7. 기타 함수

max2d 함수는 2차원 배열에서 최댓값의 좌표를 찾는 함수이다. 주어진 이미지 img를 전치시키고, 1차원 벡터로 변환한 후 최댓값과 해당 인덱스를 찾아 좌표 r과 c로 변환한다.

DrawCross 함수는 이미지 img에 교차선을 그리는 함수이다. 주어진 패턴 좌표 rcpnt, 반지름 rad, 색상 color를 기반으로 이미지에 교차선을 그린다.

② Test_cir03.jpg

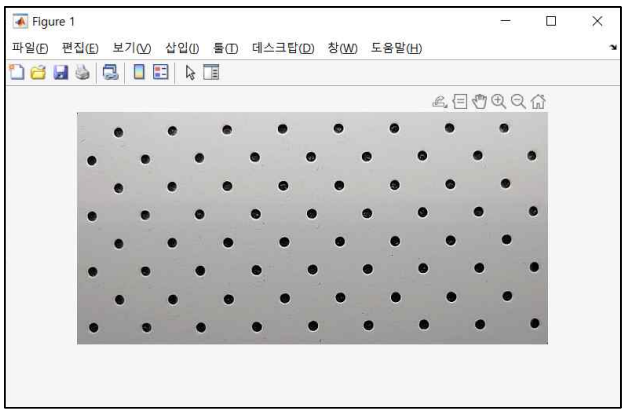


Figure 1 : 원본 이미지

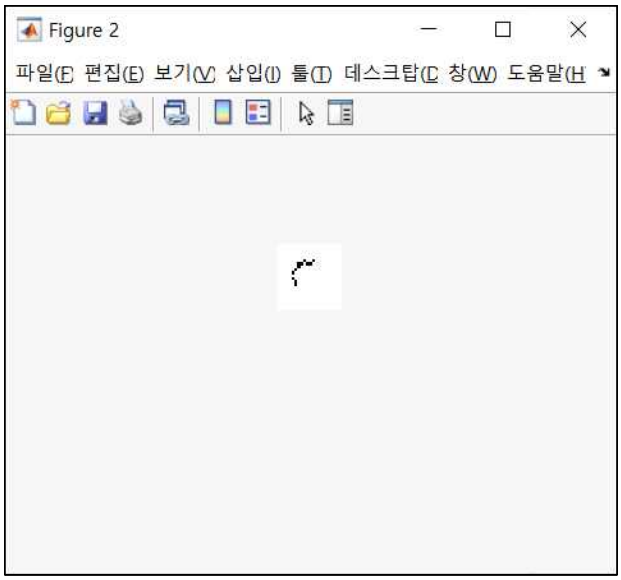
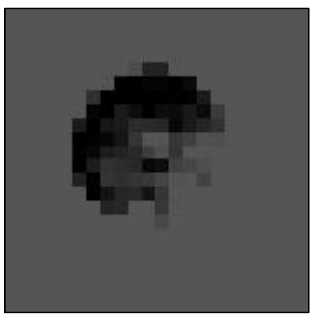


Figure 2 : 패턴으로 사용할 객체 이미지



아래의 Matlab 코드에서 imgY를 double형으로 형변환하지 않으면 Figure 2에서 다음과 같은 패턴이 얻어졌음을 볼 수 있다.

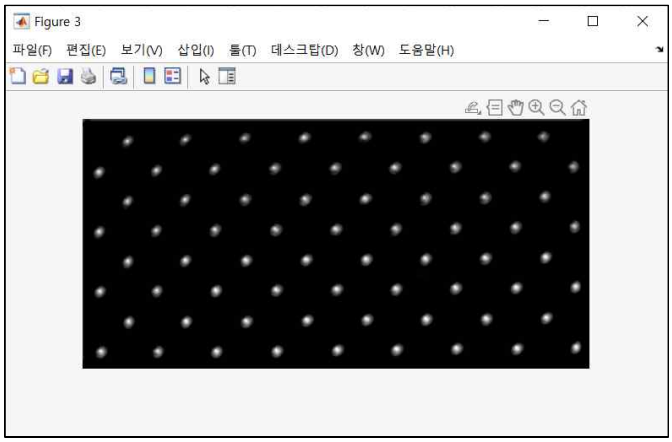


Figure 3 :
패턴인식 결과를 표시한 이미지로 입력 이미지와 패턴 이미지 간의 유사도를 계산하여 나온 결과를 보여준다.

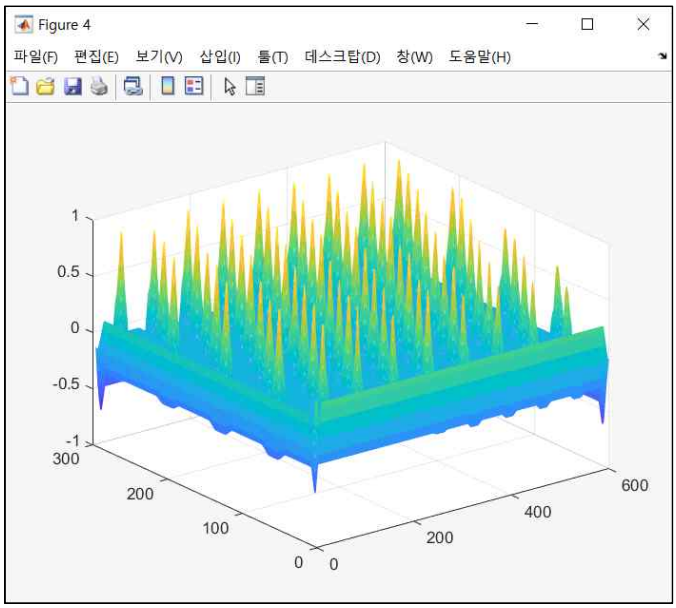
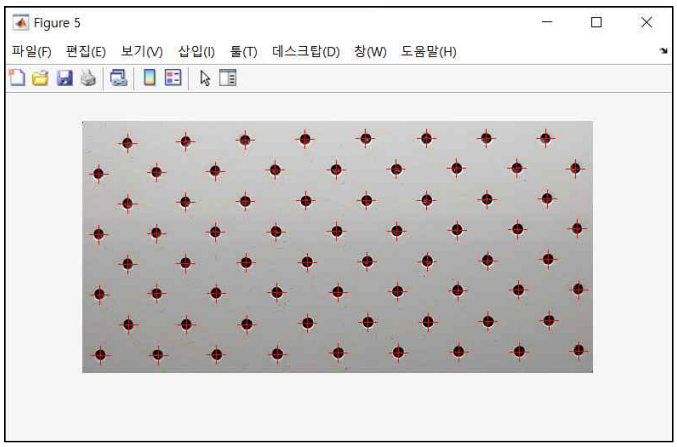
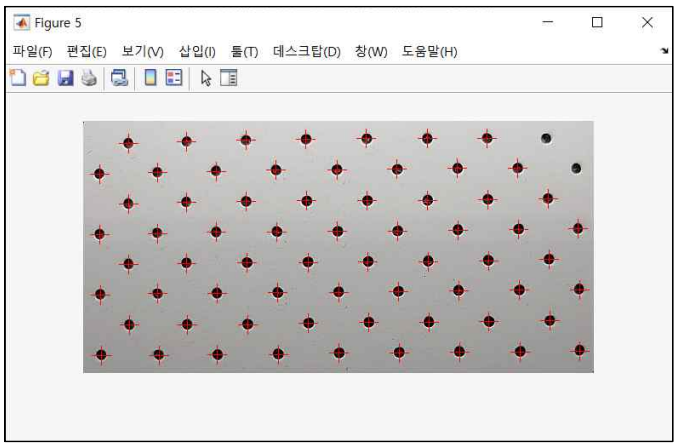


Figure 4 :
패턴인식 결과를 3D 형태로 표시한 메쉬 그래프로 입력 이미지에 대한 패턴 유사도의 분포를 시각화한 것이다.

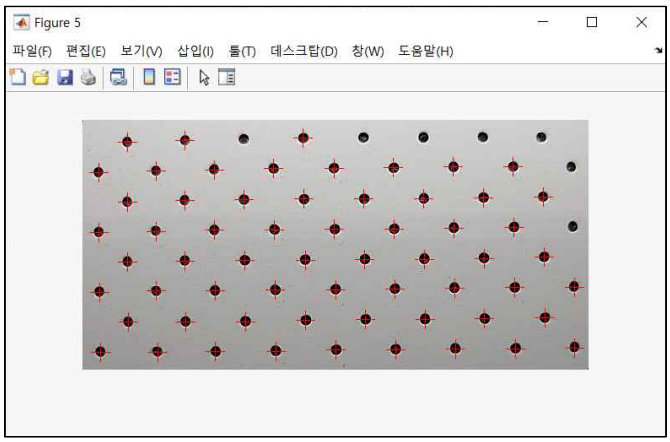
Figure 5 :
패턴인식 결과에 대해서 패턴이 감지된 위치에 빨간색 십자가 표시를 추가한 이미지이다.



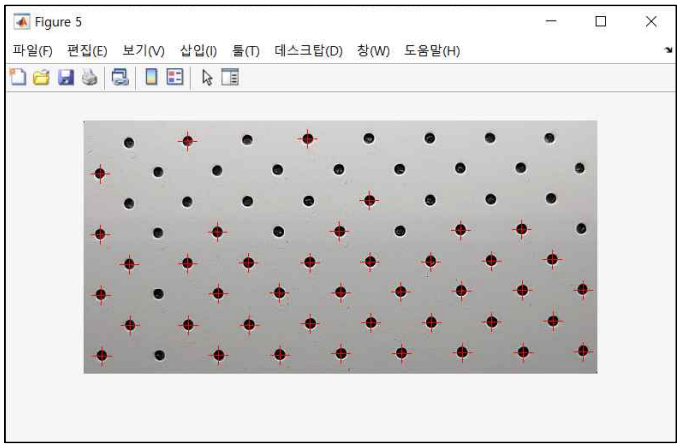
threshold = 0.6



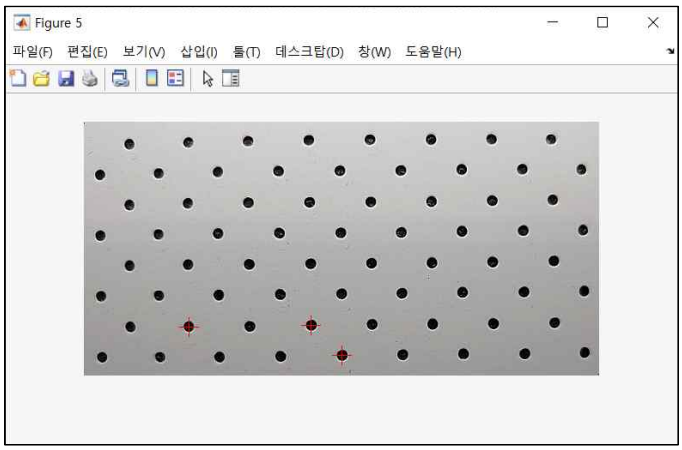
threshold = 0.7



threshold = 0.8



threshold = 0.9



threshold = 0.99

1)-①에서 설명한 것과 마찬가지로 threshold의 값에 따라 인식되는 점의 개수가 달라짐을 실험을 통해 알 수 있다. threshold의 값이 높다는 것은 패턴으로 인식하는 기준이 더 엄격한 것과 같다. 따라서 threshold의 값이 1에 가까워 질수록 인식되는 점의 개수가 줄어든다.

Matlab code

```
clear;
clc;

fname = "Test_cir03.jpg";
img = imread(fname);
figure(1); imshow(img);

imgY = (img(:,:,1) + img(:,:,2) + img(:,:,3))/3;
imgY = double(imgY);
obj = imgY(15:15+21,112:112+21);
figure(2); imshow(obj);

patt = flipud(fliplr(obj));
patt = patt/sum(patt(:));
patt = patt - mean(patt(:));

imgR = conv2(imgY, patt, 'same');
imgR = imgR/max(imgR(:));

figure(3); imshow(imgR);
figure(4); mesh(imgR);

num = 0;
rcpnt = zeros(num,2);
threshold = 0.6;
objsize = size(obj);
radr = ceil(objsize(1)/2);
radc = ceil(objsize(2)/2);

while(num<1000)
 [maxval, r, c] = max2d(imgR);
 if maxval < threshold
     break;
 end
 num = num+1;

 rcpnt(num,1) = r;
 rcpnt(num,2) = c;
```

1. 이미지 로드 및 표시
imread 함수를 사용하여 이미지를 로드하고, imshow 함수를 사용하여 로드한 이미지를 표시한다.

2. 이미지 전처리
RGB 이미지를 그레이스케일로 변환하기 위해 R, G, B 채널의 평균값을 계산하고, 이미지를 double 자료형으로 변환한다.

3. 패턴 추출
이미지에서 패턴을 추출하기 위해 특정 영역을 선택한다. imgY에서 15:15+21 행과 112:112+21 열에 해당하는 영역을 선택하여 패턴으로 설정하고, 해당 패턴을 표시한다.

4. 패턴 컨볼루션
패턴을 컨볼루션하기 위해 flipud와 fliplr 함수를 사용하여 패턴을 상하좌우로 반전시킨다. 패턴을 정규화하기 위해 패턴의 합으로 나누고, 평균값을 빼준다. 그 후, conv2 함수를 사용하여 이미지 imgY와 패턴 patt를 컨볼루션하여 결과 이미지 imgR을 생성한다. 마지막으로, imgR을 최댓값으로 정규화한다.

5. 패턴인식 및 제거
imgR에서 패턴을 인식하고, 인식된 패턴을 제거하는 반복문을 수행한다. 패턴의 최댓값이 threshold보다 작을 경우 반복문을 종료한다. 인식된 패턴의 좌표를 rcpnt에 저장하고, 패턴을 제거하기 위해 해당 영역을 0으로 설정한다. 이 과정은 최대 1000번 반복된다.

Matlab code

```
 % Erase
  rs = max(r-radr, 1);
  re = min(r+radr, size(imgR, 1));
  cs = max(c-radc, 1);
  ce = min(c+radc, size(imgR, 2));
  imgR(rs:re, cs:ce) = 0;
end

rad = radr;
color = [255, 0, 0]; % Red
imgC = DrawCross(img, rcpnt, rad, color);
figure(5); imshow(imgC);

function [maxval, r, c] = max2d(img)

 [row, col] = size(img);
 vec = img(:);

 [maxval, ind] = max(vec);

 r = mod(ind-1, row)+1;
 c = floor((ind-1)/row)+1;
end

function [imgC] = DrawCross(img, rcpnt, rad,
color)
 imgC = img;
 [num, ~] = size(rcpnt);
 for n = 1:num
  r = rcpnt(n,1);
  c = rcpnt(n,2);
  imgC(max(r-rad,1):min(r+rad,end),  c,  1)  =
color(1);
  imgC(r,  max(c-rad,1):min(c+rad,end),  1)  =
color(1);
  imgC(max(r-rad,1):min(r+rad,end),  c,  2)  =
color(2);
  imgC(r,  max(c-rad,1):min(c+rad,end),  2)  =
color(2);
  imgC(max(r-rad,1):min(r+rad,end),  c,  3)  =
color(3);
  imgC(r,  max(c-rad,1):min(c+rad,end),  3)  =
color(3);
end

end
```

6. 결과 표시

인식된 패턴에 교차선을 그려서 시각적으로 표시한다. DrawCross 함수를 사용하여 이미지 img에 인식된 패턴 좌표 rcpnt, 반지름 rad, 색상 color를 기반으로 교차선을 그린다. 마지막으로, 결과 이미지 imgC를 표시한다.

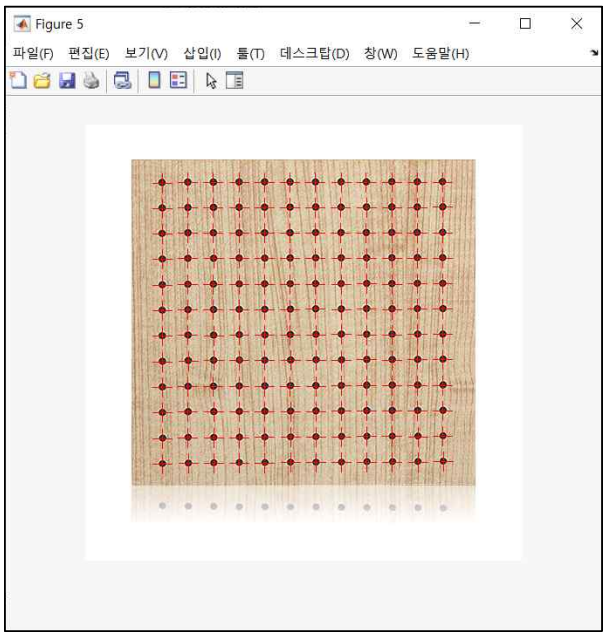
7. 기타 함수

max2d 함수는 2차원 배열에서 최댓값의 좌표를 찾는 함수이다. 주어진 이미지 img를 1차원 벡터로 변환한 후 최댓값과 해당 인덱스를 찾아 좌표 r과 c로 변환한다.

DrawCross 함수는 이미지 img에 교차선을 그리는 함수이다. 주어진 패턴 좌표 rcpnt, 반지름 rad, 색상 color를 기반으로 이미지에 교차선을 그린다.

2) Enhanced Pattern Recognition

앞서 “Test_cir02.png” 이미지 하단의 흐릿한 부분에 위치한 점은 threshold 값을 낮춰도 인식되지 않는다. 아래의 이미지는 threshold를 0.2로 설정하고 수행한 패턴인식의 결과이다.



threshold = 0.2

하단의 흐릿한 부분에 위치한 점은 threshold 값을 많이 낮춰도 여전히 인식되지 않는다. 이를 해결하기 위해서는 패턴인식의 성능을 개선해야 하고 패턴의 에지를 강화하는 방법을 택할 수 있다.

이미지의 고주파 영역을 증폭시키면 이미지의 에지가 강화되는 효과를 얻을 수 있다. 이미지의 고주파 영역을 강화하기 위해서 High Pass Filter를 적용하고 대표적으로 Sobel 필터 또는 Canny 필터를 사용할 수 있다. 일반적으로 Canny 필터가 Sobel 필터에 비해 좋은 성능을 보이지만, 계산량이 비교적 많고 값이 더 비싸다는 단점이 있다.

본 실습은 계산량이 많지 않기 때문에 Canny 필터를 사용해도 실험의 진행에 큰 차질이 없을 것으로 예상하였고, Matlab에서 Sobel 필터와 Canny 필터가 모두 기본적으로 제공되어 더 나은 결과를 얻고자 Canny 필터를 채택하였다.

다음은 Canny 필터를 적용하여 개선한 패턴인식의 결과와 Matlab 코드이다.

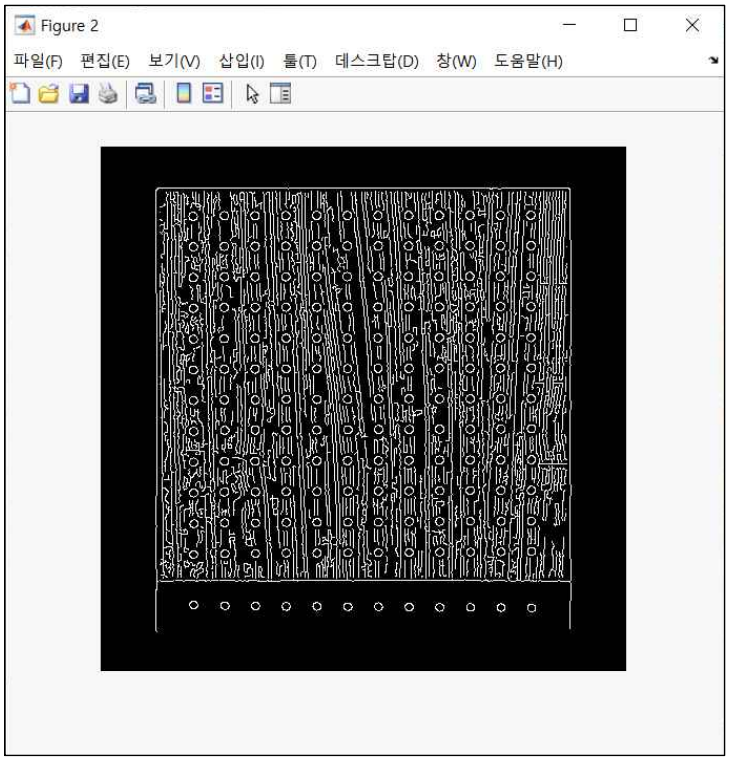


Figure 2 : Canny 필터를 사용하여 검출된 에지 이미지

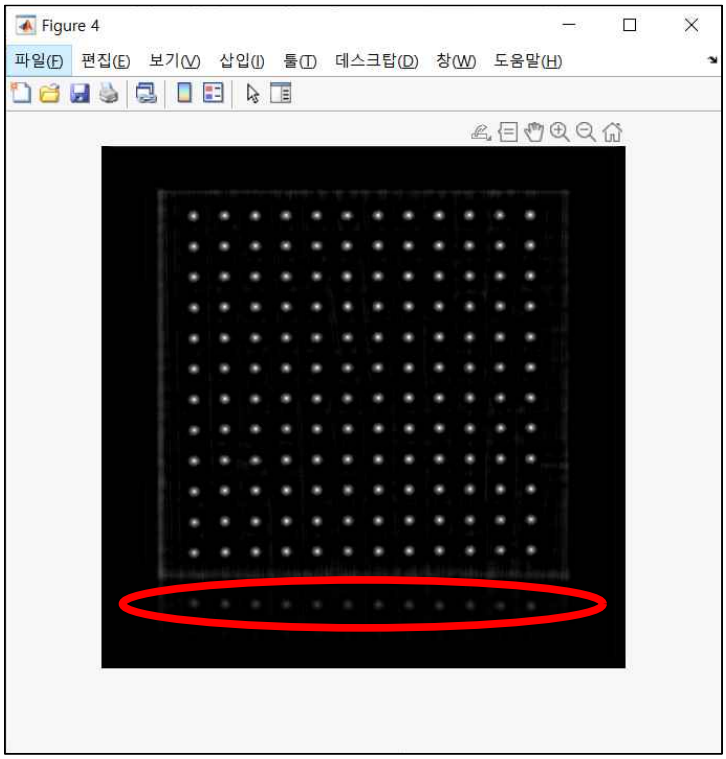


Figure 3 :

향상된 객체 특징을 갖는 이미지의 패턴인식 결과로 빨간색으로 표시한 곳을 보면 1)-①에서는 인식되지 않았던 흐릿한 영역의 점이 새롭게 나타남을 알 수 있다.

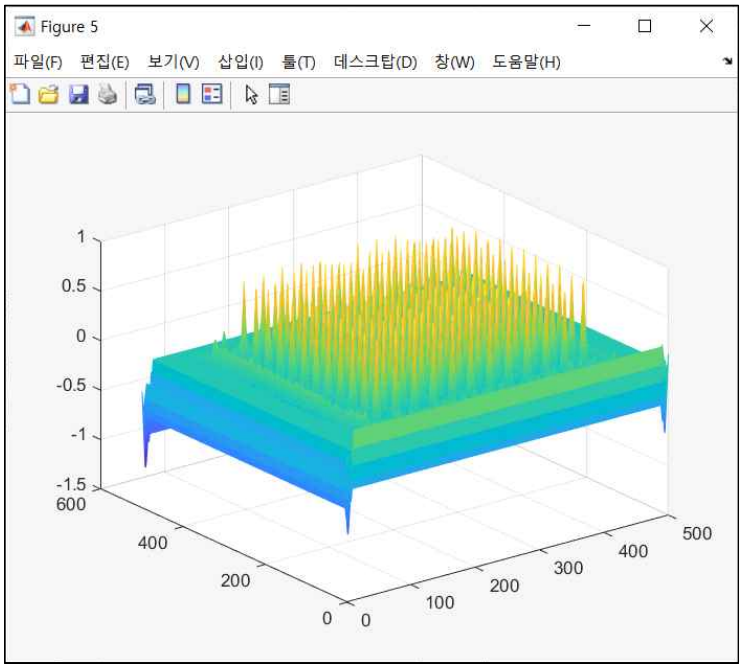


Figure 5 :
imgR의 3D 메시 플롯으로 이를 통해 시각화된 imgR의 픽셀 강도 값을 얻었다.

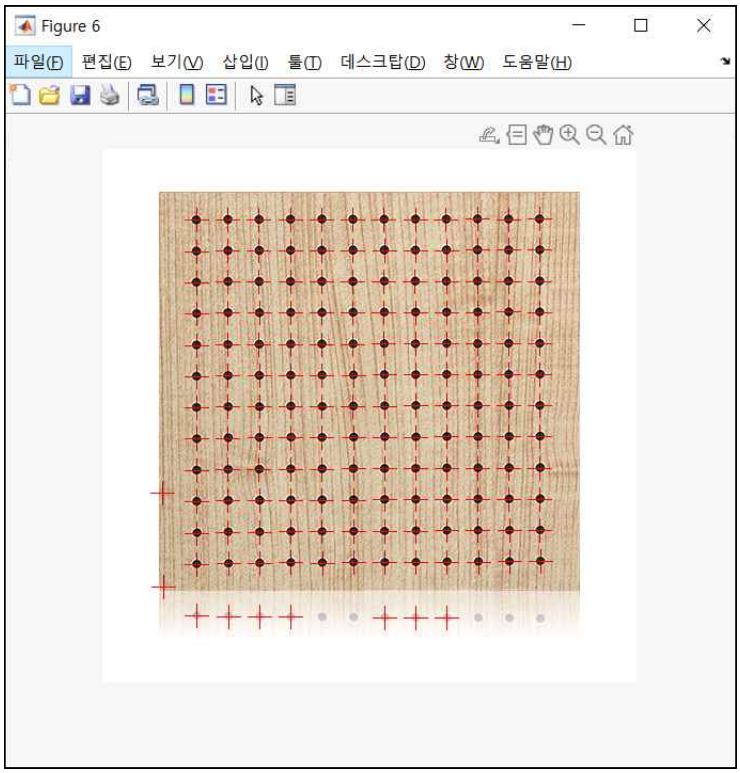


Figure 6 :
패턴인식 결과에 대해서 패턴이 감지된 위치에 빨간색 십자가 표시를 추가한 이미지이다. 하단의 흐릿한 영역에 있는 점도 일부 인식하지만, threshold 값이 0.245로 비교적 낮아 점이 아닌 일부 영역이 점으로 잘못 인식되는 오류도 생겼다.

Matlab code

```
clear;
clc;

fname = "Test_cir02.png";
img = imread(fname);
figure(1); imshow(img);

% Convert image to grayscale
img_gray = rgb2gray(img);

% Edge detection using Canny filter
edge_img = edge(img_gray, 'canny');

% Display the edge image
figure(2); imshow(edge_img);

imgY = double(img_gray);
obj = imgY(55:55+21, 107:107+21);
figure(3); imshow(obj);

patt = flipud(fliplr(obj));
patt = patt / sum(patt(:));
patt = patt - mean(patt(:));
```

1. 이미지 로드 및 표시
imread 함수를 사용하여 이미지를 로드하고, imshow 함수를 사용하여 로드한 이미지를 표시한다.

2. 이미지 전처리
rgb2gray 함수를 사용하여 컬러 이미지를 그레이스케일로 변환한다. 그리고 img_gray 변수에 그레이스케일 이미지를 저장한다.

3. 엣지 검출
edge 함수를 사용하여 그레이스케일 이미지에서 엣지를 검출한다. 이때 Canny 필터를 사용한다. edge_img에 검출된 엣지 이미지를 저장한다.

4. 패턴 추출
imgY에 그레이스케일 이미지를 double 자료형으로 변환하여 저장한다. 55:55+21 행과 107:107+21 열에 해당하는 영역을 obj로 선택하여 패턴으로 설정한다.

5. 패턴 컨볼루션 및 강조
obj를 상하좌우로 반전시켜 patt로 저장한다. patt를 전체 합으로 나누어 정규화하고 patt의 평균값을 빼주어 중심을 0으로 조정한다.

Matlab code

```
% Enhance edge component in the pattern image
obj_edges = edge(obj, 'canny');
obj_edges = double(obj_edges);
obj_edges(obj_edges > 0) = 1;
enhanced_obj = obj_edges + obj;
enhanced_obj = enhanced_obj / max(enhanced_obj(:));
enhanced_obj = enhanced_obj - mean(enhanced_obj(:));

imgR = conv2(imgY, enhanced_obj, 'same');
imgR = imgR / max(imgR(:));

figure(4); imshow(imgR);
figure(5); mesh(imgR);

num = 0;
rcpnt = zeros(num, 2);
threshold = 0.245;
objsize = size(obj);
radr = ceil(objsize(1) / 2);
radc = ceil(objsize(2) / 2);
```

6. 패턴인식 및 제거
edge 함수를 사용하여 obj에서 엣지를 검출한다. double 자료형으로 변환한 obj_edges를 사용하여 엣지를 강조한다. 엣지와 원본 이미지를 더하여 enhanced_obj를 생성하고. enhanced_obj를 정규화한다.
imgY와 enhanced_obj 사이의 2D 컨볼루션을 수행하여 imgR을 생성하고 imgR을 정규화합니다.

7. 피크 검출 및 제거
변수 초기화 및 설정 :
num : 검출된 피크의 수를 저장하는 변수
rcpnt : 검출된 피크의 위치를 저장하는 변수
threshold : 피크로 간주하기 위한 임계값
objsize : obj의 크기를 저장하는 변수
radr : obj의 행 크기의 절반을 저장하는 변수
radc : obj의 열 크기의 절반을 저장하는 변수

최댓값이 임계값보다 작다면 반복문을 종료
피크를 rcpnt에 저장
imgR에서 피크 영역을 제거

Matlab code

```
while (num < 1000)
 [maxval, r, c] = max2d(imgR);
 if maxval < threshold
      break;
 end

 num = num + 1;

 rcpnt(num, 1) = r;
 rcpnt(num, 2) = c;

 % Erase
 rs = max(1, r - radr);
 re = min(size(imgR, 1), r + radr);
 cs = max(1, c - radc);
 ce = min(size(imgR, 2), c + radc);
 imgR(rs:re, cs:ce) = 0;
end

rad = radr;
color = [255, 0, 0]; % Red
imgC = DrawCross(img, rcpnt, rad,
color);
figure(6); imshow(imgC);

function [maxval, r, c] = max2d(img)
 [row, col] = size(img);
 img = img';
 vec = img(:);

 [maxval, ind] = max(vec);
 r = floor((ind - 1) / col);
 c = (ind - 1) - r * col;
 r = r + 1;
 c = c + 1;
end
```

8. 결과 표시

인식된 패턴에 교차선을 그려서 시각적으로 표시한다. DrawCross 함수를 사용하여 이미지 img에 인식된 패턴 좌표 rcpnt, 반지름 rad, 색상 color를 기반으로 교차선을 그린다. 마지막으로, 결과 이미지 imgC를 표시한다.

9. 기타 함수

max2d 함수 :

2D 이미지에서 최댓값과 해당 위치를 반환하는 함수이다. 주어진 이미지 img를 1차원 벡터로 변환한 후 최댓값과 해당 인덱스를 찾아 좌표 r과 c로 변환한다.

Matlab code

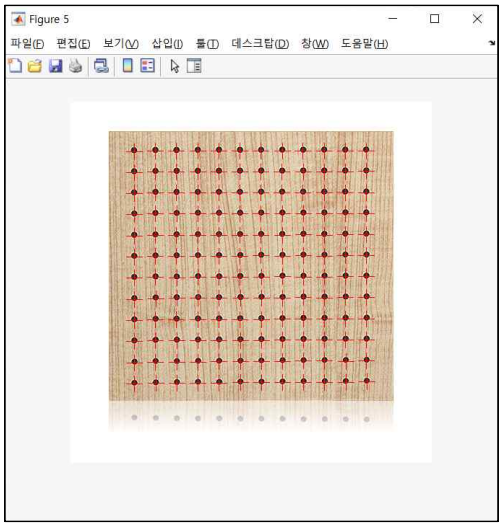
```
function [imgC] = DrawCross(img, rcpnt, rad, color)
 imgC = img;
 [num, ~] = size(rcpnt);

 for n = 1:num
  r = rcpnt(n, 1);
  c = rcpnt(n, 2);

  cs = max(1, c - rad);
  ce = min(size(imgC, 2), c + rad);
  imgC(r, cs:ce, 1) = color(1);

  rs = max(1, r - rad);
  re = min(size(imgC, 1), r + rad);
  imgC(rs:re, c, 2) = color(2);

  cs = max(1, c - rad);
  ce = min(size(imgC, 2), c + rad);
  imgC(r, cs:ce, 2) = color(2);

  rs = max(1, r – rad);
  re = min(size(imgC, 1), r + rad);
  imgC(rs:re, c, 3) = color(3);

  cs = max(1, c - rad);
  ce = min(size(imgC, 2), c + rad);
  imgC(r, cs:ce, 3) = color(3);
 end
end
```

DrawCross :
이미지에 피크 주위에 크로스를 그리는 함수이다. 주어진 패턴 좌표 rcpnt, 반지름 rad, 색상 color를 기반으로 이미지에 교차선을 그린다.

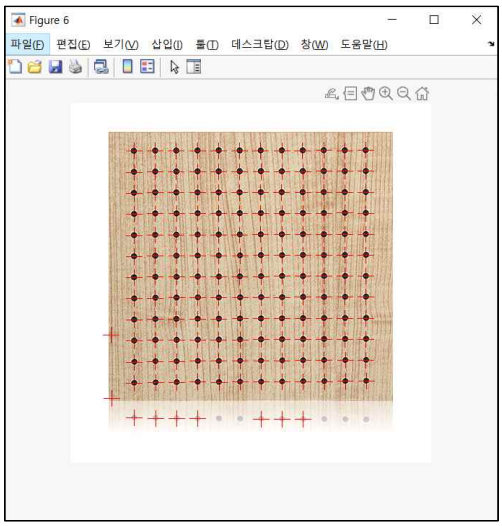
III. 결론

이번 실습을 통해 컨벌루션을 통한 패턴인식에 대해 학습하였다. 특히 흐릿한 영역이 일부 포함된 “Test_cir02.png”의 경우 흐릿한 영역에 속한 점들을 인식하는 것이 일반적인 패턴인식 방법으로는 불가능하였다. 따라서 Canny 필터를 거쳐 에지 영역을 강화하는 방법을 적용하였고 결과적으로 흐릿한 영역에 속한 점을 일부 인식할 수 있는 Matlab 코드를 얻었다.

하지만 이번에 설계한 Canny 필터를 적용한 패턴인식의 경우, 본론 2)-Figure6에서 볼 수 있듯이 실제로 점이 아닌 영역이 점으로 잘못 인식되는 오류가 있었다. 이를 해결하고자 갖은 노력을 다했으나 점이 아닌 영역의 값이 흐릿한 영역에 속한 점의 영역보다 민감하게 반응하여 threshold의 값을 조절하는 것으로는 해결이 불가능하였다. 또한 High Pass Filter를 통해 에지 영역을 강화하는 방안 이외의 새로운 수가 떠오르지 않았다. 이에 대한 해결책은 추후의 새로운 실습에서 학습하여 얻을 수 있을 것이라 기대한다.



Ordinary
Pattern Recognition
threshold = 0.245



Enhanced
Pattern Recognition
using Canny Filter
threshold = 0.245