

Image analysis through the MATLAB

PRESENTATION CONTENTS

01 INTRODUCTION

02 ALGORITHM

03 MATLAB PROGRAMM

04 CONCLUSION

05 FUTURE WORK

01 INTRODUCTION

문제점

- HOW TO ANALYSIS LOTS OF PICTURES

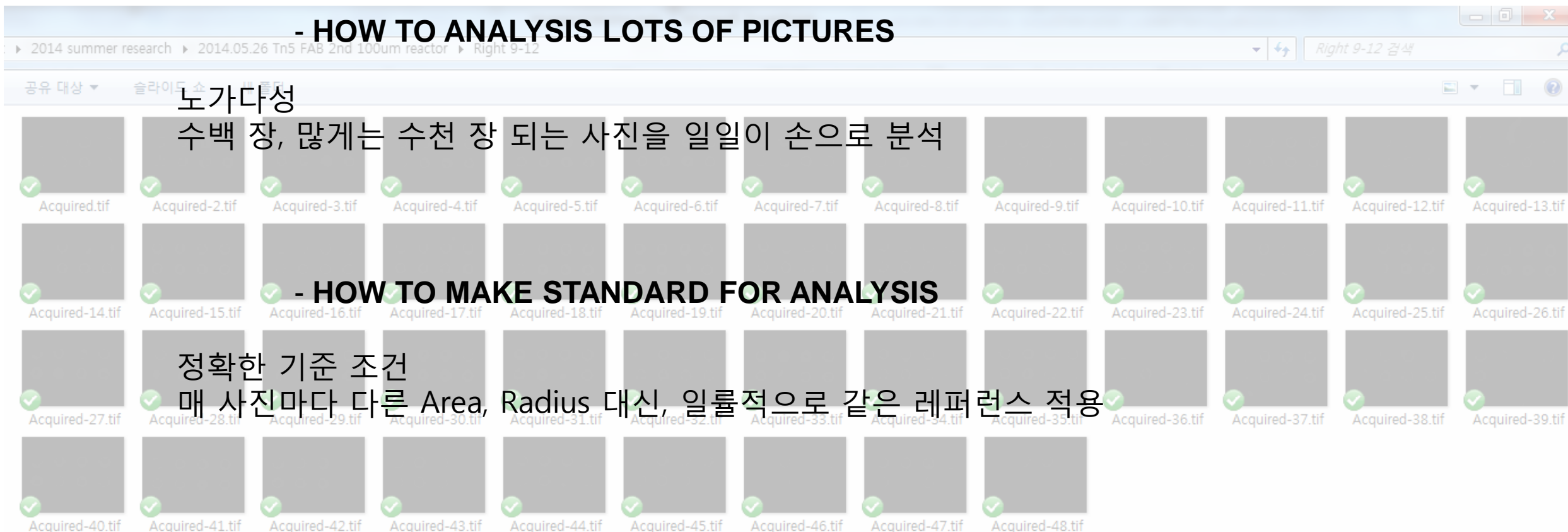
노가다성

수백 장, 많게는 수천 장 되는 사진을 일일이 손으로 분석

- HOW TO MAKE STANDARD FOR ANALYSIS

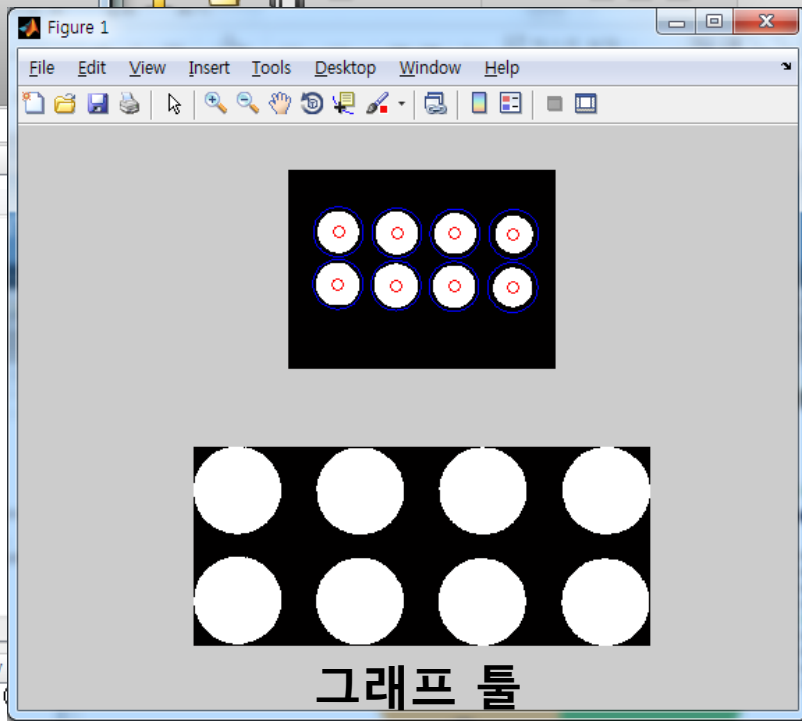
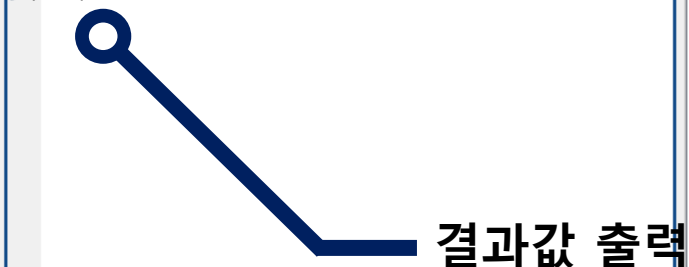
정확한 기준 조건

매 사진마다 다른 Area, Radius 대신, 일률적으로 같은 레퍼런스 적용



01 INTRODUCTION

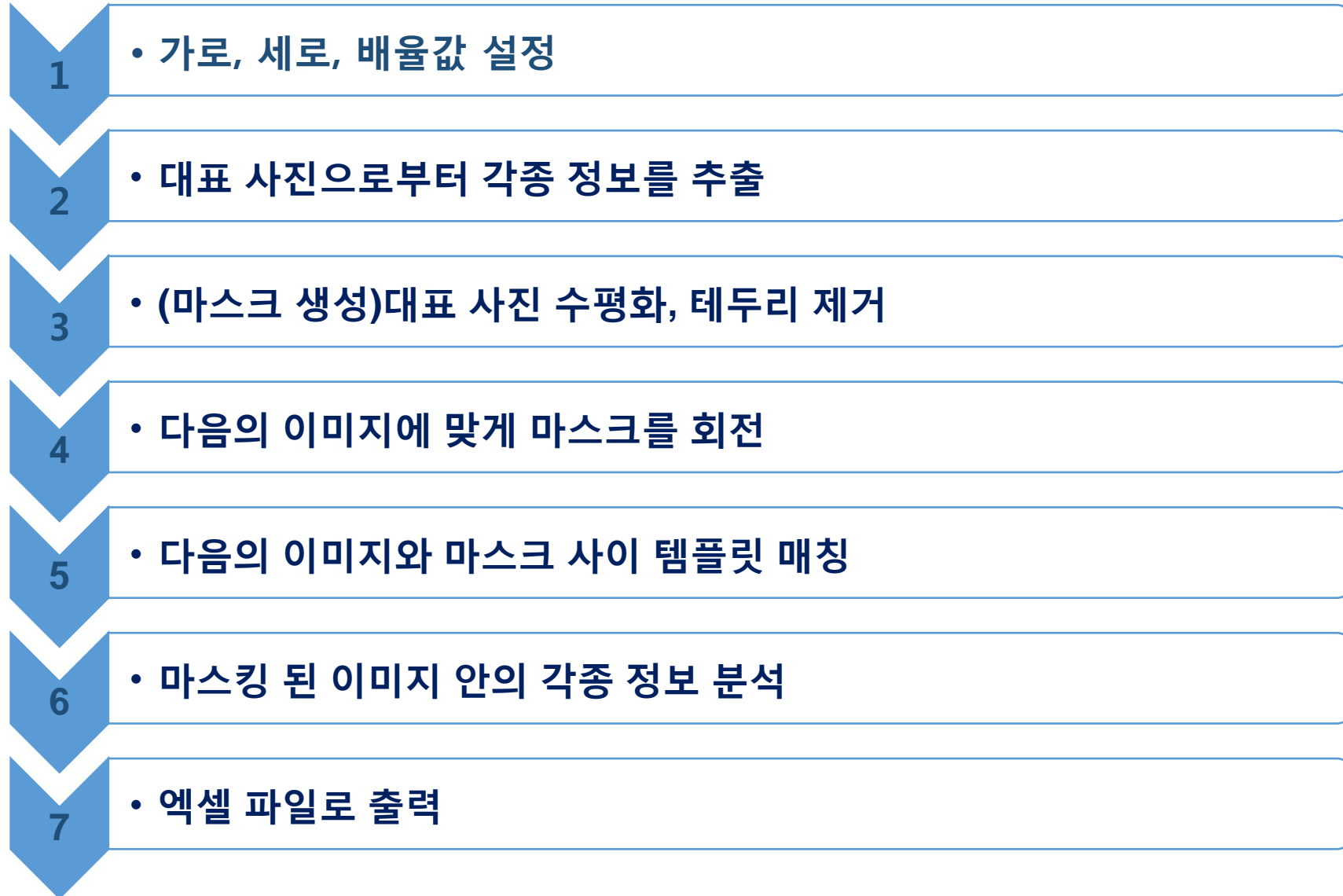
매트랩 사용 환경



스크립스의 간편성
(매트릭스 기반)

MATLAB

02 ALGORITHM

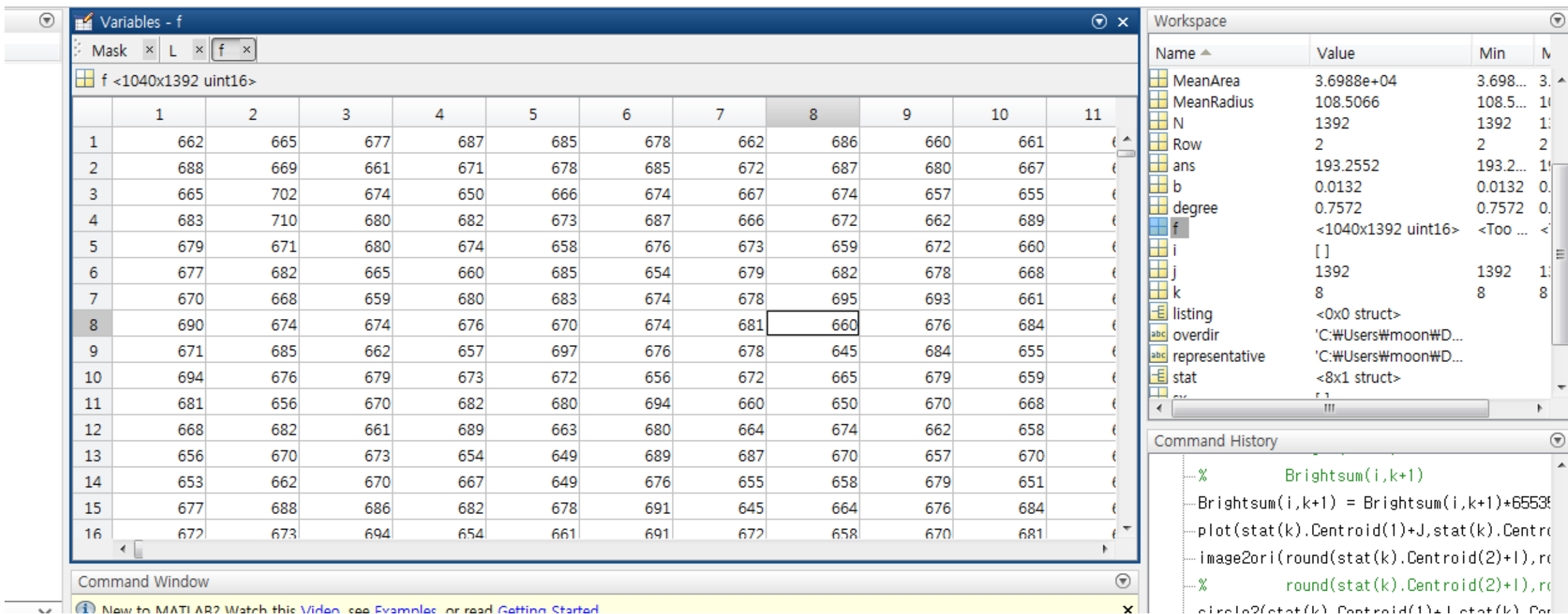


03 MATLAB PROGRAMM

이미지 읽기

1. `f=imread(filename);`

해당 경로의 파일을 매트릭스 형태로 입력
(Types : unit8, unit16, double, logical...)



The image displays the MATLAB interface with two windows open: 'Variables - f' and 'Workspace'.

Variables - f window: Shows a matrix `f` of size `<1040x1392 uint16>`. The matrix is displayed as a grid of values. The first 16 rows and 11 columns are visible. The value at row 8, column 8 is highlighted.

	1	2	3	4	5	6	7	8	9	10	11
1	662	665	677	687	685	678	662	686	660	661	
2	688	669	661	671	678	685	672	687	680	667	
3	665	702	674	650	666	674	667	674	657	655	
4	683	710	680	682	673	687	666	672	662	689	
5	679	671	680	674	658	676	673	659	672	660	
6	677	682	665	660	685	654	679	682	678	668	
7	670	668	659	680	683	674	678	695	693	661	
8	690	674	674	676	670	674	681	660	676	684	
9	671	685	662	657	697	676	678	645	684	655	
10	694	676	679	673	672	656	672	665	679	659	
11	681	656	670	682	680	694	660	650	670	668	
12	668	682	661	689	663	680	664	674	662	658	
13	656	670	673	654	649	689	687	670	657	670	
14	653	662	670	667	649	676	655	658	679	651	
15	677	688	686	682	678	691	645	664	676	684	
16	672	673	694	654	661	691	672	658	670	681	

Workspace window: Shows a list of variables in the workspace. The variable `f` is highlighted, showing its size as `<1040x1392 uint16>`.

Name	Value	Min	Max
MeanArea	3.6988e+04	3.698...	3.698...
MeanRadius	108.5066	108.5...	108.5...
N	1392	1392	1392
Row	2	2	2
ans	193.2552	193.2...	193.2...
b	0.0132	0.0132	0.0132
degree	0.7572	0.7572	0.7572
f	<1040x1392 uint16>	<Too ...	<Too ...
i	[]		
j	1392	1392	1392
k	8	8	8
listing	<0x0 struct>		
overdir	'C:\Users\moon\W...		
representative	'C:\Users\moon\W...		
stat	<8x1 struct>		

Command History window: Shows the command `Brightsum(i,k+1)` and the assignment `Brightsum(i,k+1) = Brightsum(i,k+1)+65535`.

03 MATLAB PROGRAMM

히스토그램 평활화

2. `histed=intrans(image,'stretch',mean2(im2double(image)),0.9);`

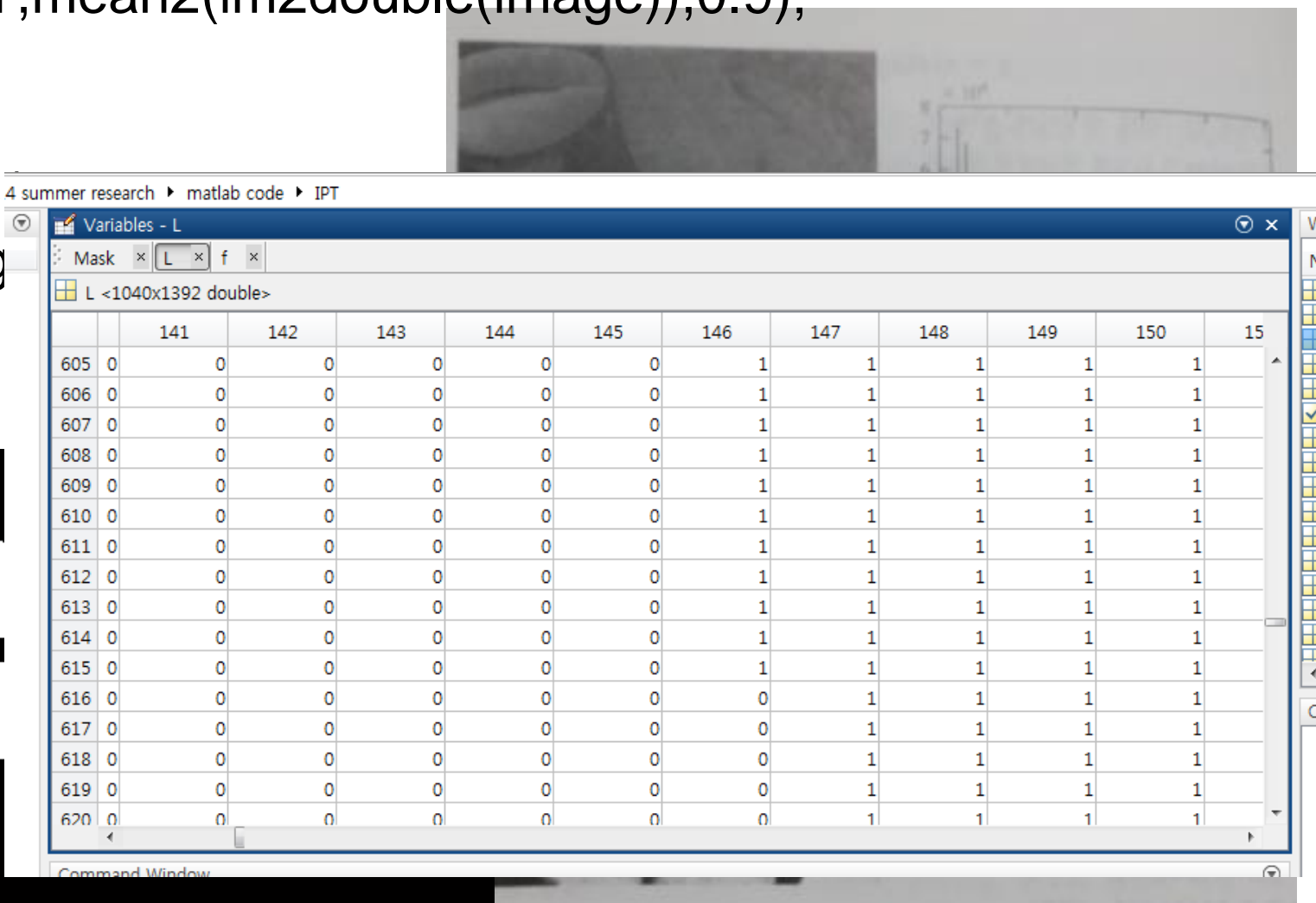
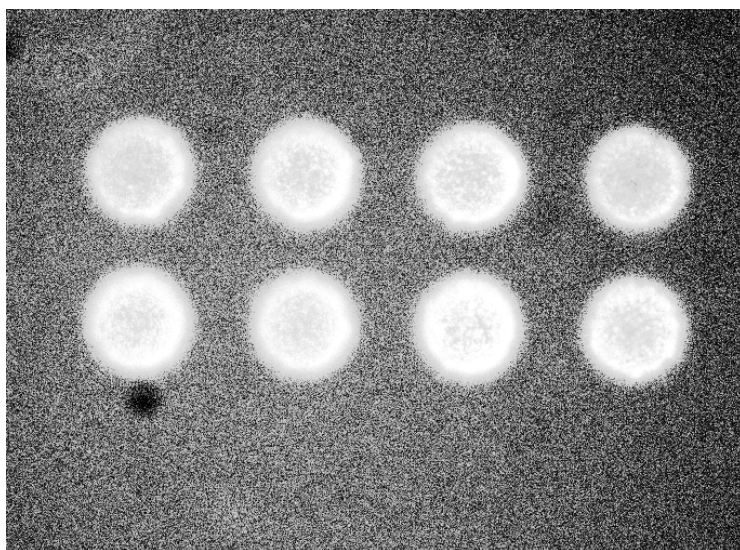
`histed=histeq(histed);`

`h=fspecial('gaussian',50,15);`

`gh=imfilter(histed,h,'replicate');`

`gh=im2bw(gh,1*graythresh(gh));`

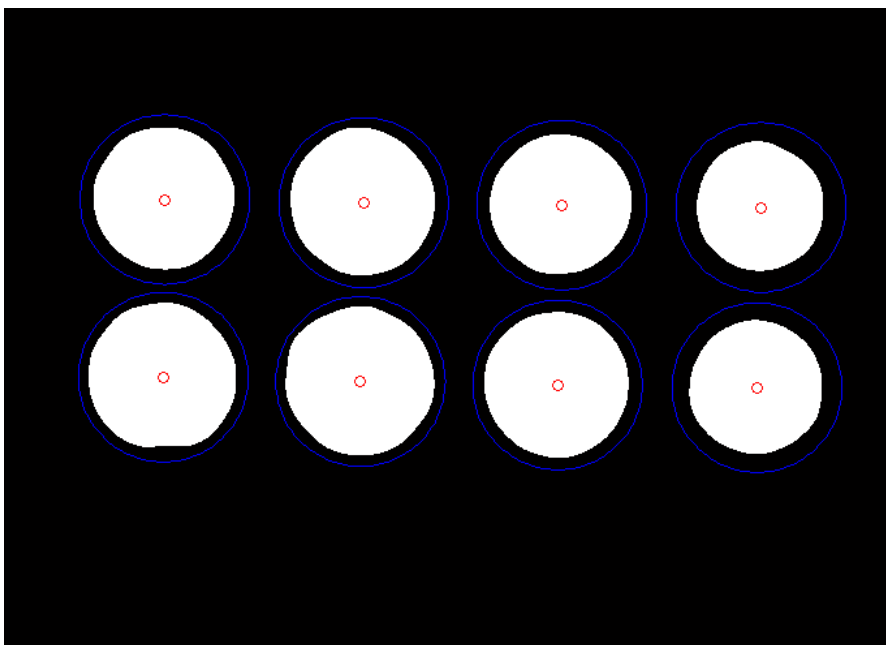
해당 이미지를 히스토그램 분석
문턱치 처리로 이진화



3. MeanArea=sum(noesort)/n;
 MeanRadius=sqrt(MeanArea/pi);
 stat = regionprops(L,'centroid');

대표 이미지의 평균 면적, 평균 반지름을 구하고,
 각 영역 별 면적 중심(센트로이드)를 추출.

-> 대표 사진을 기준으로 중심간 거리, 샘플의 면적을 정함



대표 사진 정보

n개보다 많으면 n개가 될때까지 가장 작은 면적을 가진 라벨링 없앴

```
for i=1:num
    noe(i)= sum(sum(L==i));    %Number of dots(area)
end
noesort=sort(noe);

NumberOfRemoval=0;
if n < num
    for k=1:num-n    %가장 작은 면적을 가진 라벨부터
        label=find(noe==noesort(k));    %index of min

        for i=1:size(L,1)
            for j=1:size(L,2)
                if L(i,j) == label
                    L(i,j)=0;
                end
            end
        end
        NumberOfRemoval=NumberOfRemoval+1;
    end
    MeanArea=sum(noesort(num-n+1:end))/n;
    MeanRadius=sqrt(MeanArea/pi);
else
    MeanArea=sum(noesort)/num;
    MeanRadius=sqrt(MeanArea/pi);
end
```


4. `Maskrotated = imrotate(Mask,degree,'nearest','crop');`

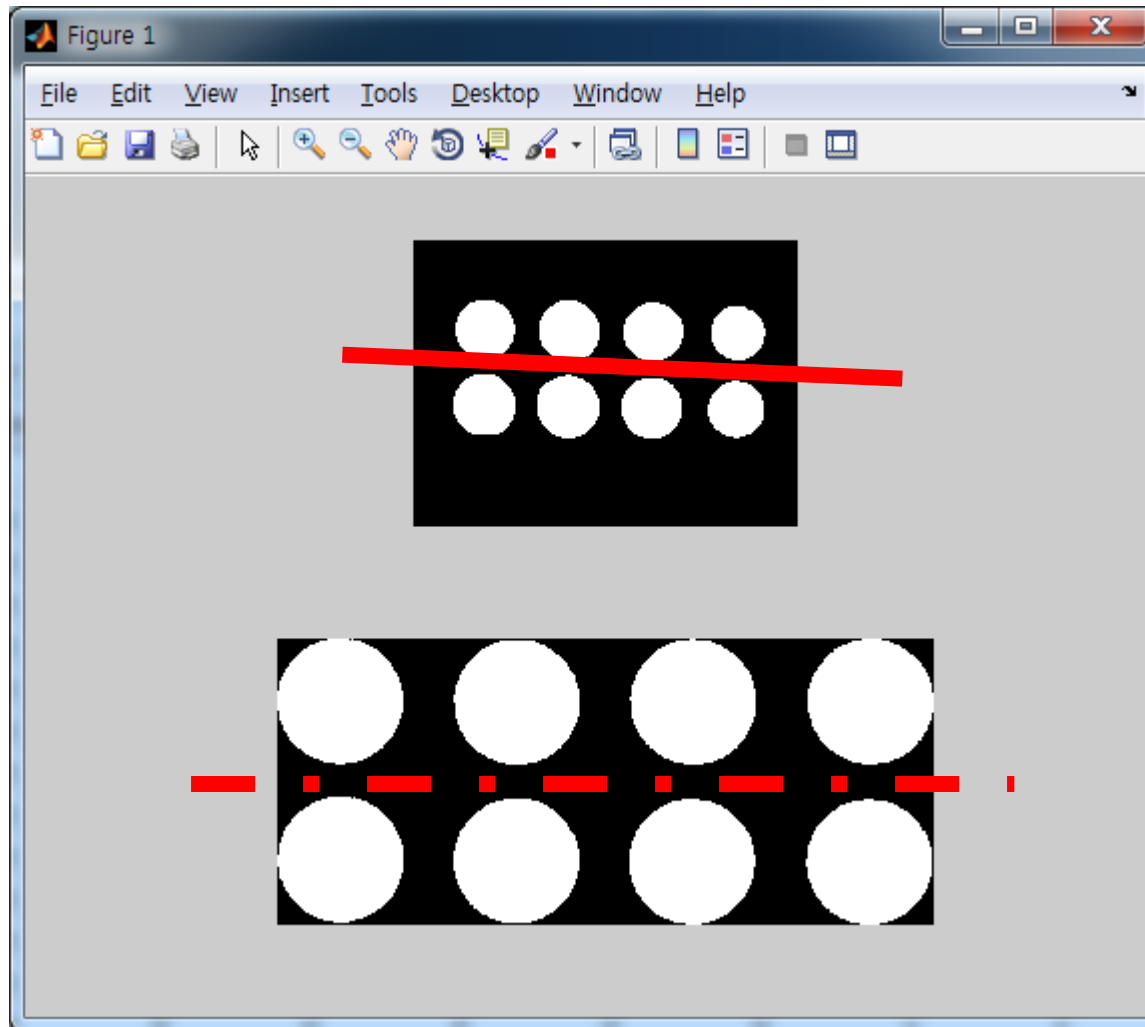
해당 마스크를 알맞은 기울기에 맞게 회전

마스크 Mask

반시계 방향으로 회전 degree

회전 후 테두리를 사각형에 맞게 자름 'crop'

-> 일정한 면적, 중심 간의 거리 및
수평화된 마스크 생성(아래 그림)

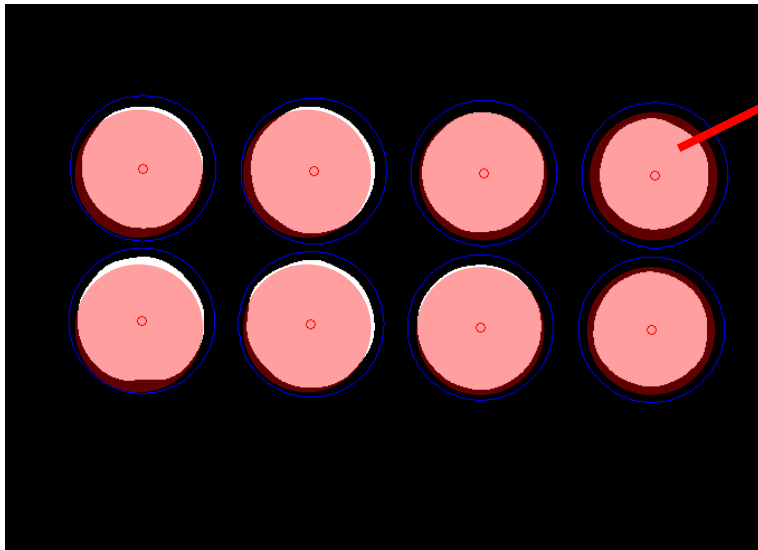


5. `g = dftcorr(image2,Maskrotated);`
 다음의 이미지에서 마스크가 가장 잘 매칭되는 지점을 찾음

새로운 이미지 image2

마스크 Maskrotated

라 할때, Image2 – Maskrotated 의 값(차이)이
 가장 작게 되는 지점을 image2 에서 찾음



Mask

12.3.3 코릴레이션에 의한 매칭

코릴레이션은 원리적으로는 상당히 단순하다. 영상 $f(x, y)$ 가 주어지면, 코릴레이션 문제는 영상에서 주어진 부영상(마스크 또는 템플릿이라고도 부름) $w(x, y)$ 와 매칭되는 위치들을 찾는 것이다. 대체로, $w(x, y)$ 는 $f(x, y)$ 보다 훨씬 작다. 매칭을 찾는 한 가지 방법은 $w(x, y)$ 를 공간 필터로 보고 3.4.1절에서 설명한 것과 정확히 같은 방법으로 f 에서 w 의 각 위치에 대해서 곱들의 합(또는 정규화된 버전)으로 계산한다. 이때 $f(x, y)$ 에서 $w(x, y)$ 의 최적 매칭은 결과인 코릴레이션 영상에서 최대값을 갖는 위치이다. $w(x, y)$ 가 작지 않다면, 방금 설명한 이 방법은 일반적으로 계산상 비용이 크므로, 공간 코릴레이션의 실제적인 구현은 보통 하드웨어 지향 솔루션들에 의존한다.

프로토타입화를 위해서, 대안은 코릴레이션 정리를 사용해서 주파수 영역에서 코릴레이션을 구현하는 것이다. 이 방법은 제 4장에서 논의한 컨볼루션 정리와 같이 공간 코릴레이션을 영상 변환들의 곱으로 나타내는 것이다. “ \circ ”는 코릴레이션을 나타내고, “ $*$ ”는 복소 공액을 나타낸다고 하면, 코릴레이션 정리에 의해 다음 식이 성립한다.

$$f(x, y) \circ w(x, y) \Leftrightarrow F(u, v) H^*(u, v)$$

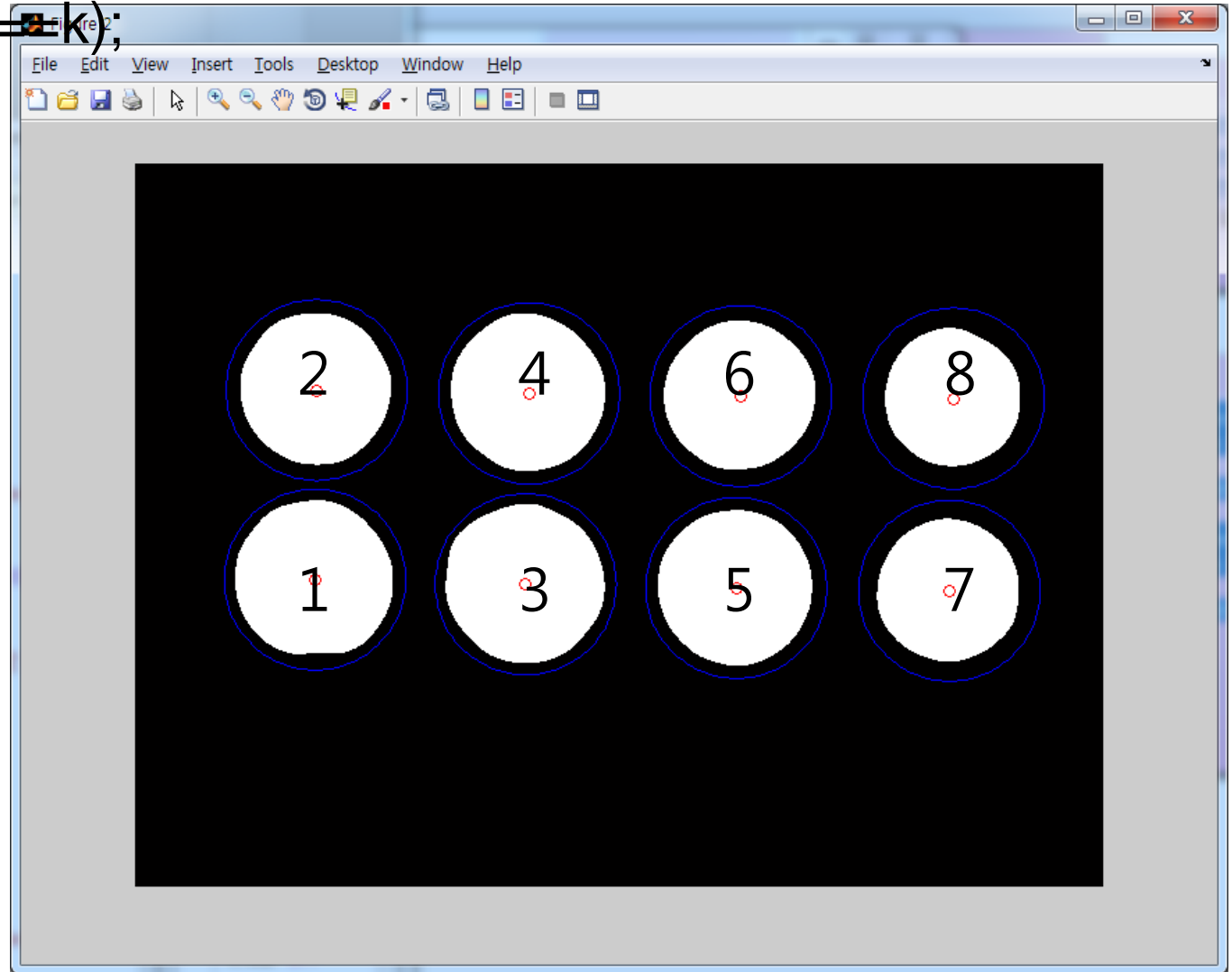
즉, 공간 코릴레이션은 한 함수의 변환에 다른 함수의 변환의 공액을 곱한 후 역 푸리에 변환을 함으로써 구할 수 있다. 역으로, 다음 식도 성립한다.

$$f(x, y) w^*(x, y) \Leftrightarrow F(u, v) \circ H(u, v)$$

6. `[xMask, yMask] = find(Maskrotated==k);`

`image2ori(xMask(i),yMask(i));`

마스크와 다음의 이미지와 겹치는 지점 중,
각 라벨의 순서로 다음의 이미지의 원본으로부터
값을 추출



03 MATLAB PROGRAMM

엑셀 파일로 출력

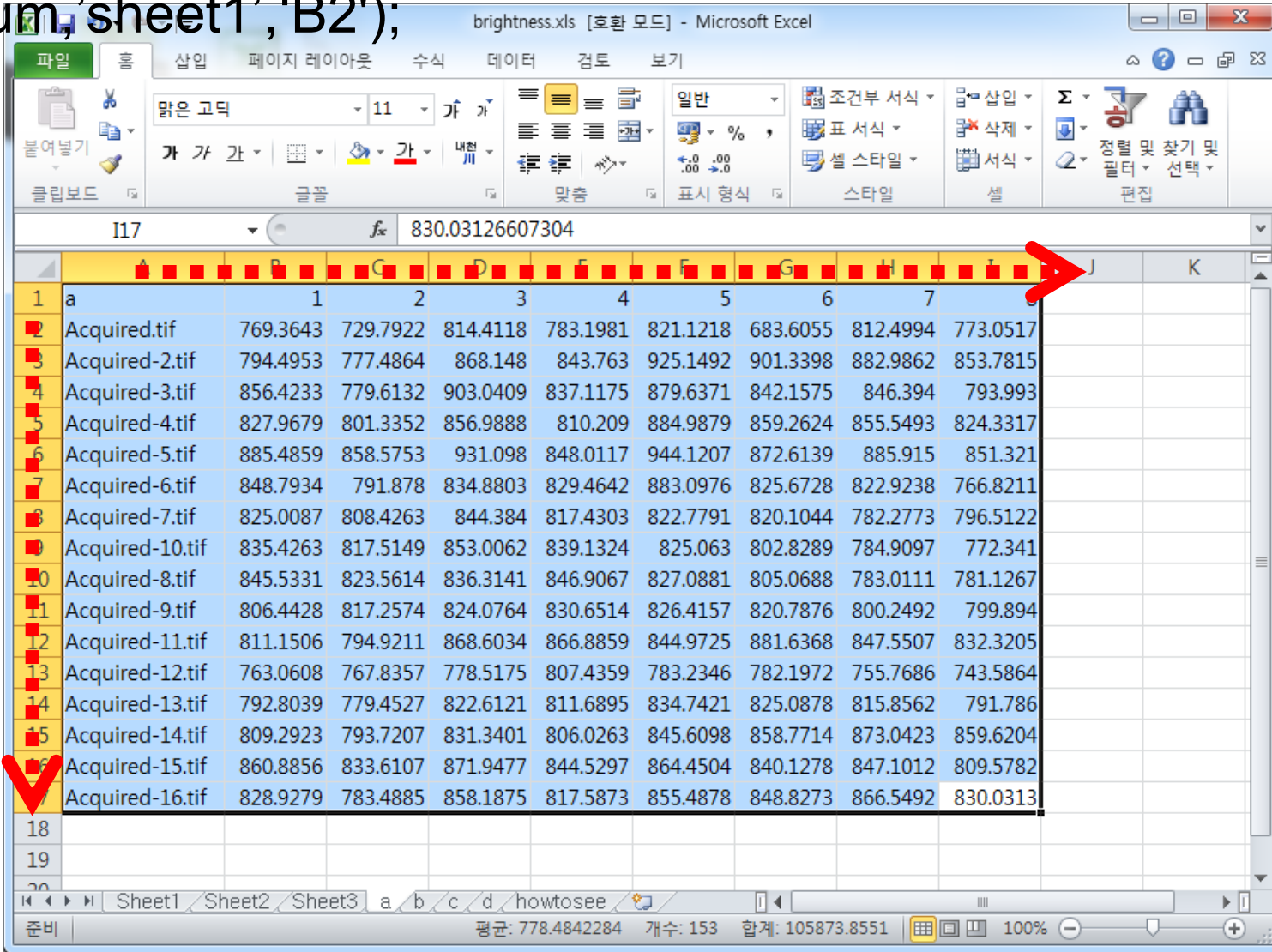
7. `xlswrite('경로\파일명.xls', Brightsum, 'sheet1', 'B2');`

해당 경로에 해당 파일명으로 엑셀파일을 저장.

저장할 데이터 매트릭스 셋 Brightsum

저장할 시트 'sheet1'

데이터 배치를 시작할 셀 'B2'

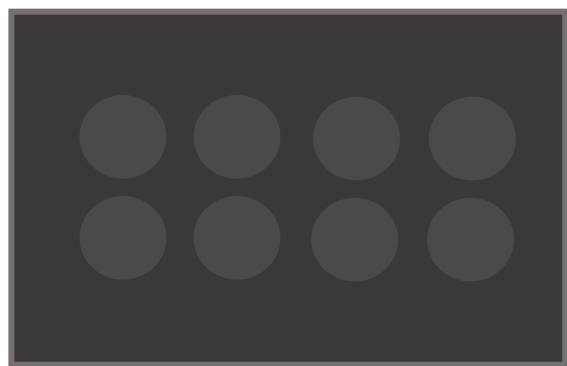


brightness.xls [호환 모드] - Microsoft Excel

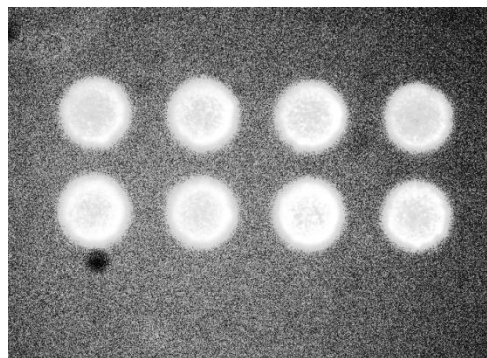
	a	b	c	d	e	f	g	h	i	j	k
1		1	2	3	4	5	6	7			
2	Acquired.tif	769.3643	729.7922	814.4118	783.1981	821.1218	683.6055	812.4994	773.0517		
3	Acquired-2.tif	794.4953	777.4864	868.148	843.763	925.1492	901.3398	882.9862	853.7815		
4	Acquired-3.tif	856.4233	779.6132	903.0409	837.1175	879.6371	842.1575	846.394	793.993		
5	Acquired-4.tif	827.9679	801.3352	856.9888	810.209	884.9879	859.2624	855.5493	824.3317		
6	Acquired-5.tif	885.4859	858.5753	931.098	848.0117	944.1207	872.6139	885.915	851.321		
7	Acquired-6.tif	848.7934	791.878	834.8803	829.4642	883.0976	825.6728	822.9238	766.8211		
8	Acquired-7.tif	825.0087	808.4263	844.384	817.4303	822.7791	820.1044	782.2773	796.5122		
9	Acquired-10.tif	835.4263	817.5149	853.0062	839.1324	825.063	802.8289	784.9097	772.341		
10	Acquired-8.tif	845.5331	823.5614	836.3141	846.9067	827.0881	805.0688	783.0111	781.1267		
11	Acquired-9.tif	806.4428	817.2574	824.0764	830.6514	826.4157	820.7876	800.2492	799.894		
12	Acquired-11.tif	811.1506	794.9211	868.6034	866.8859	844.9725	881.6368	847.5507	832.3205		
13	Acquired-12.tif	763.0608	767.8357	778.5175	807.4359	783.2346	782.1972	755.7686	743.5864		
14	Acquired-13.tif	792.8039	779.4527	822.6121	811.6895	834.7421	825.0878	815.8562	791.786		
15	Acquired-14.tif	809.2923	793.7207	831.3401	806.0263	845.6098	858.7714	873.0423	859.6204		
16	Acquired-15.tif	860.8856	833.6107	871.9477	844.5297	864.4504	840.1278	847.1012	809.5782		
17	Acquired-16.tif	828.9279	783.4885	858.1875	817.5873	855.4878	848.8273	866.5492	830.0313		
18											
19											
20											

준비 | 평균: 778.4842284 | 개수: 153 | 합계: 105873.8551 | 100%

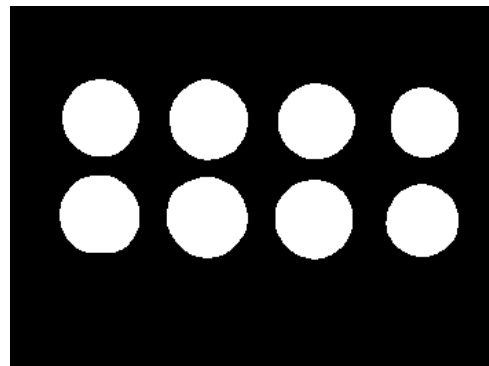
04 CONCLUSION



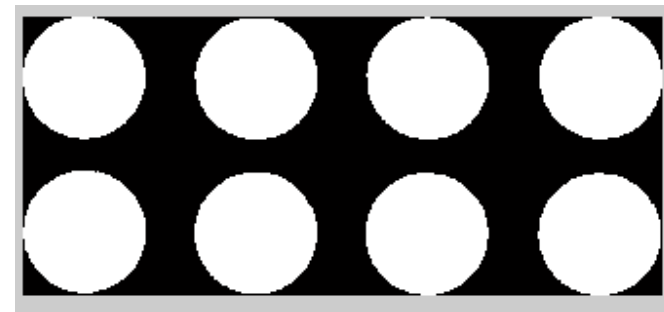
대표 사진 원본



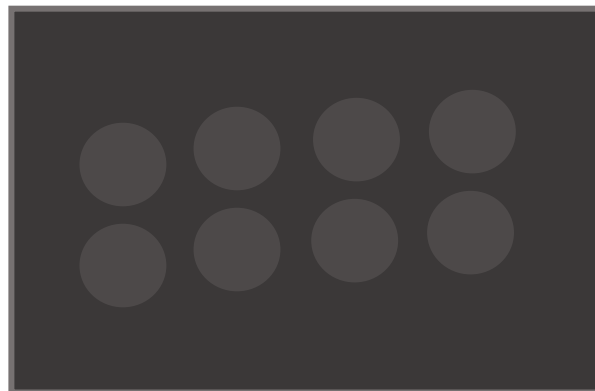
히스토그램



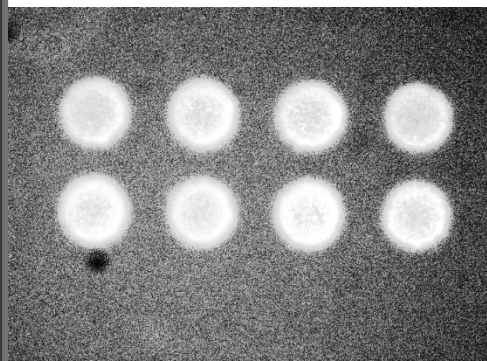
이진화 및 회전



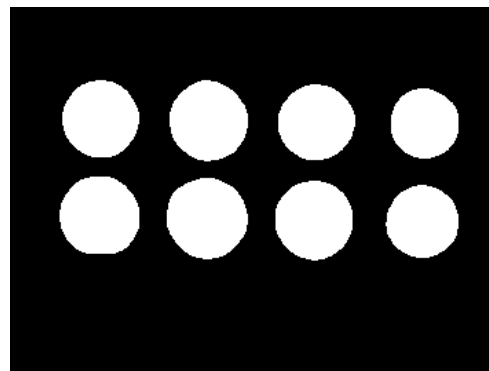
대표 마스크



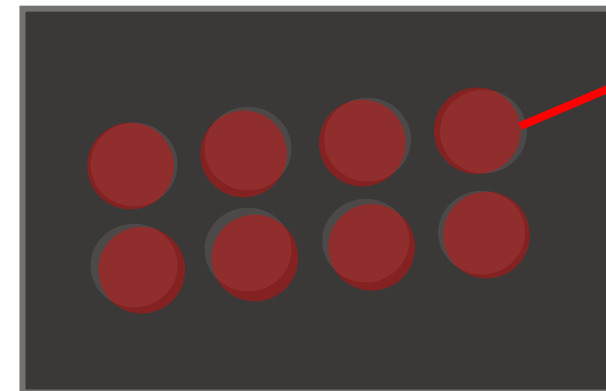
분석 사진 원본



히스토그램



이진화 및 회전



마스킹 및 자료 추출

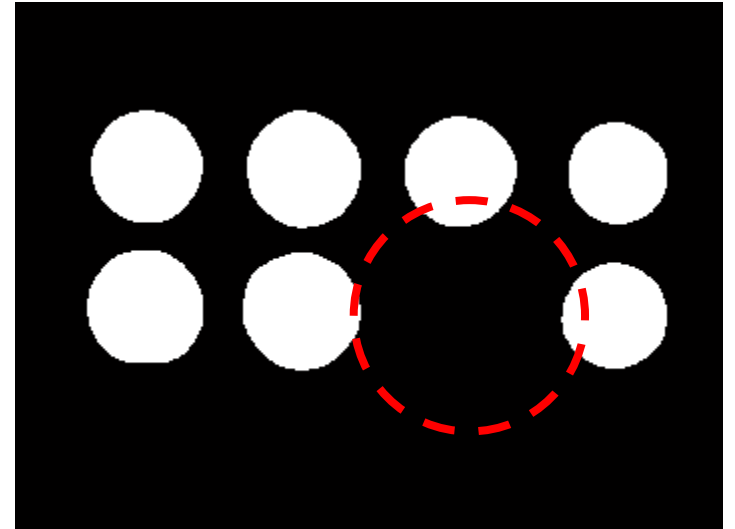
Mask

04 CONCLUSION

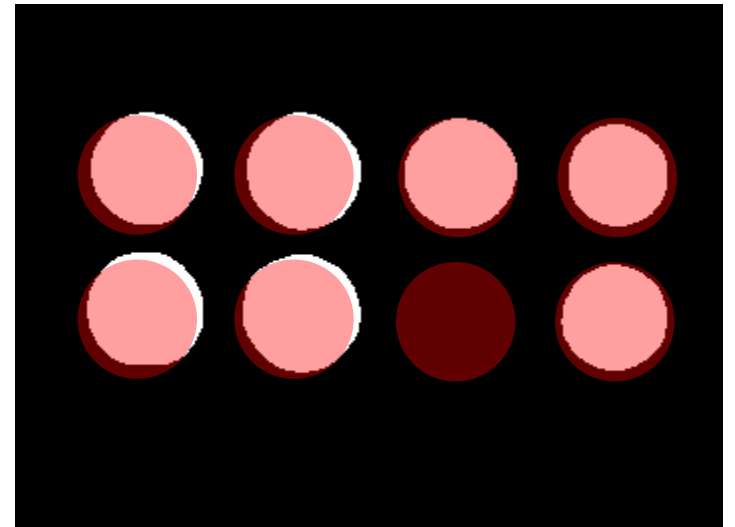
한계점

객체 인식을 통한 도형 검출X

새롭게 분석하는 이미지의 퀄리티에 따라 원으로 인식하지 못하는 문제
(실험 샘플의 반응이 전혀 일어나지 않았을 경우)



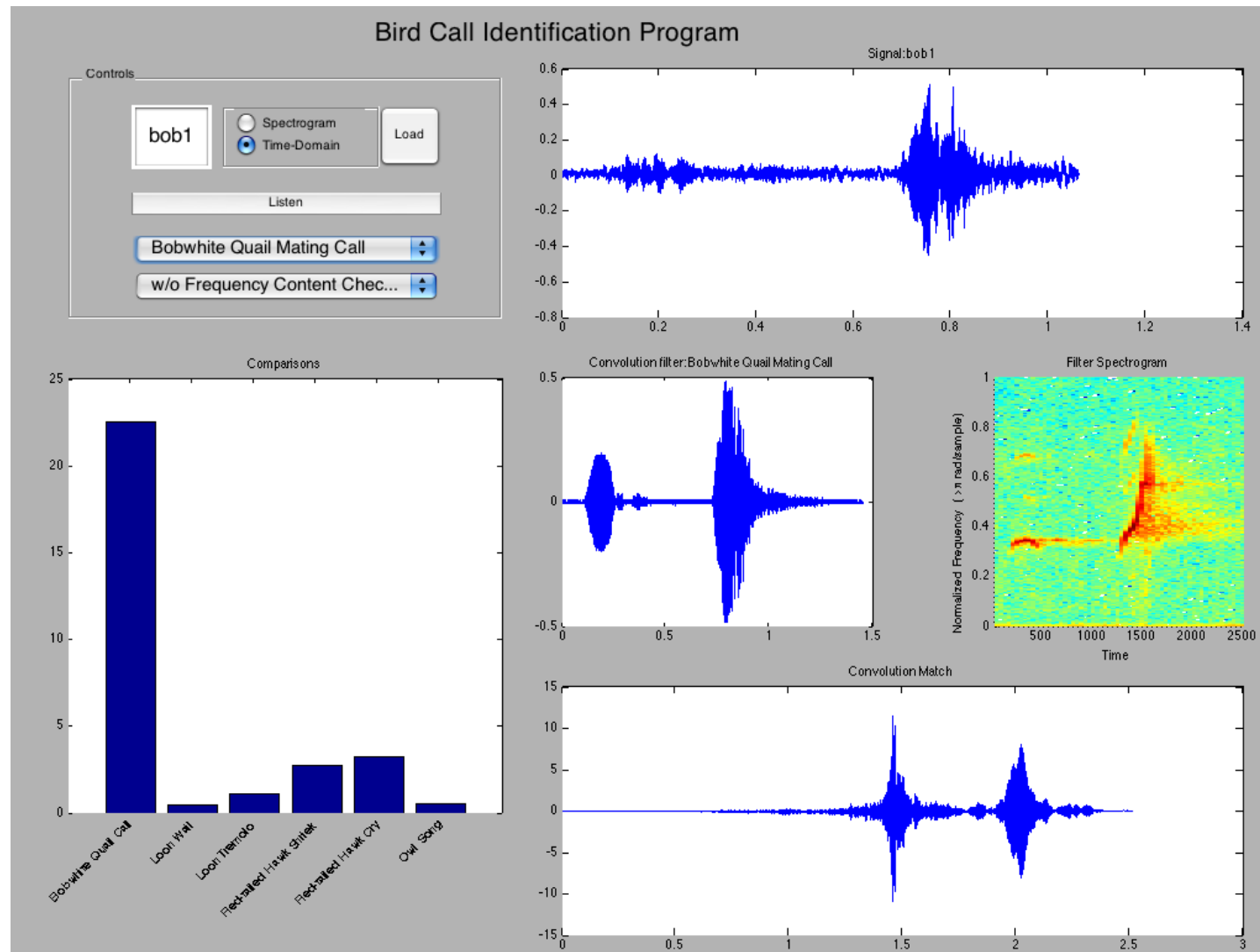
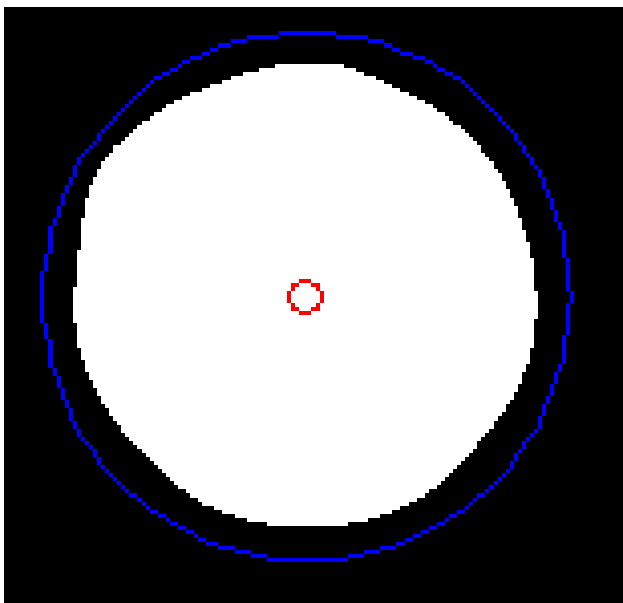
대신, 대표 이미지에서 정확한 정보를 입력 받으면,
새롭게 분석하는 이미지의 퀄리티와 상관없이, 이미지 분석



05 FUTURE WORK

1. GUI 그래픽 기반 인터페이스

2. 확장성 (원 이외의 도형으로 인식)



Thank you