

REPORT

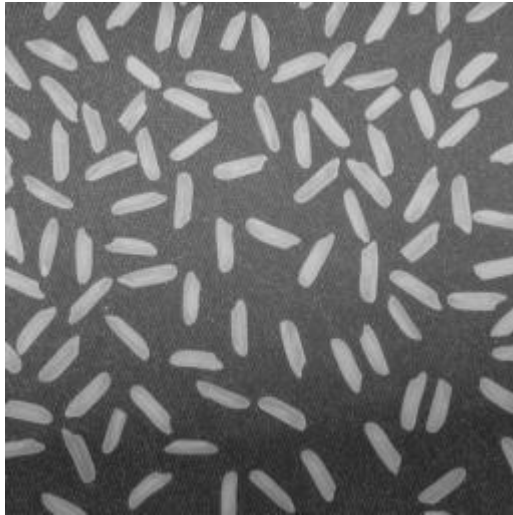
Lab04 : Methods of Image Segmentation



| | |
|------|-------------|
| 과목명 | 지능형영상처리 |
| 담당교수 | 유 훈 교수님 |
| 학과 | 융합전자공학과 |
| 학년 | 3학년 |
| 학번 | 201910906 |
| 이름 | 이학민 |
| 제출일 | 2023.05.26. |

I. 서론

지능형영상처리 12주차 수업에서 다룬 Lab04은 배경과 쌀알을 분리하여 쌀알의 패턴을 인식하는 Image Segmentation 실습이다. 테스트 이미지로 아래와 같은 2가지 이미지를 사용하였다.



rice1.png



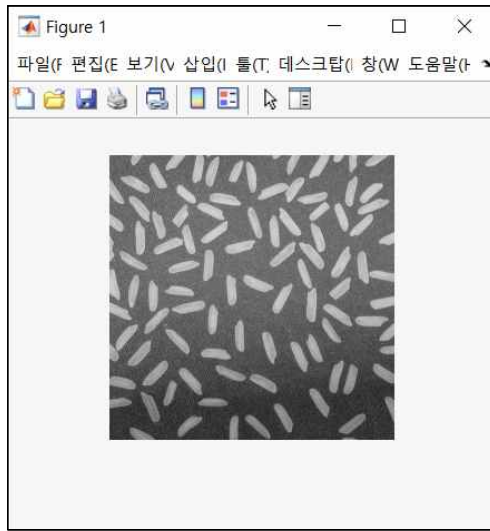
rice2.png

배경과 쌀알을 구분할 수 있도록 다양한 절차의 Image Segmentation 기법을 소개한다. 촬영된 이미지는 조명의 영향의 영향을 받아 실제로는 같은 색임에도 불구하고 이미지 히스토그램 상에서 다양한 밝기를 보인다. 따라서 조명으로 인한 영향을 최소화하기 위하여 이미지 히스토그램을 조작하는 과정을 거친다. 특히 'rice2.png'의 경우 쌀알과 배경의 색이 비슷하여 히스토그램 상에서 각 대상이 가지는 밝기의 값을 명확히 구분해내기 어렵다. 따라서 High Pass Filter를 이용한 이미지의 에지 성분 강화를 통해 쌀알을 인식하는 방법을 채택하였다.

II. 본론

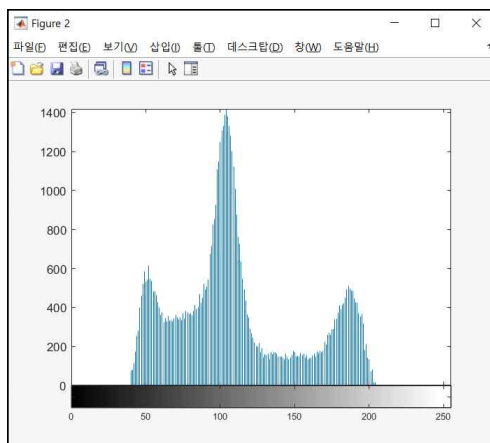
1) Test of detecting rice grains using Threshold Value

수업 시간에 진행한 쌀알 검출 기초 학습에 대한 내용이다. 결과 이미지를 보면 다수의 문제가 발생하는 것을 알 수 있는데, 본론에서는 이를 어떻게 해결하는지에 대한 내용을 다룬다.



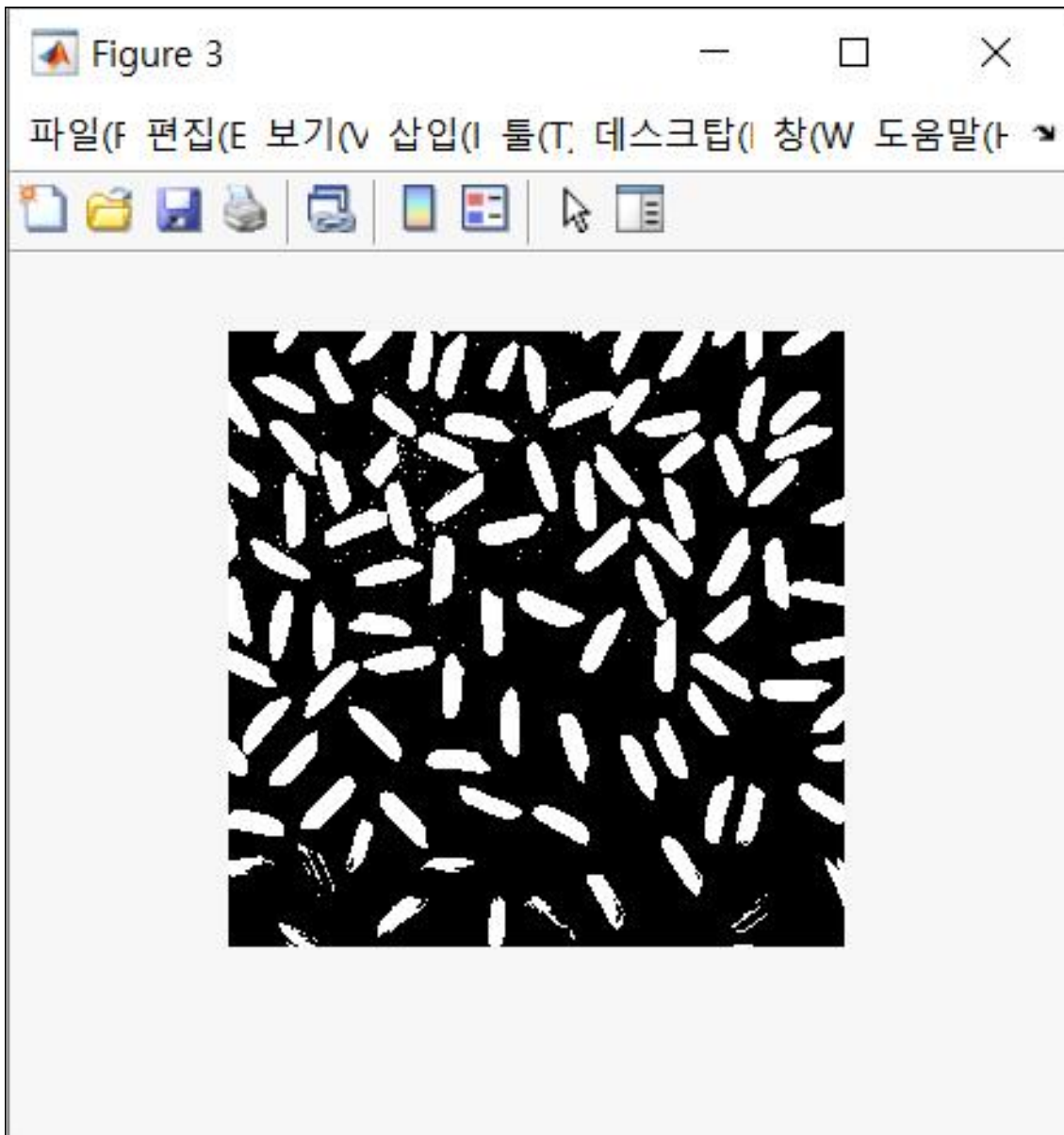
Original Image

figure(1)은 'rice1.png'의 원본을 나타낸다.



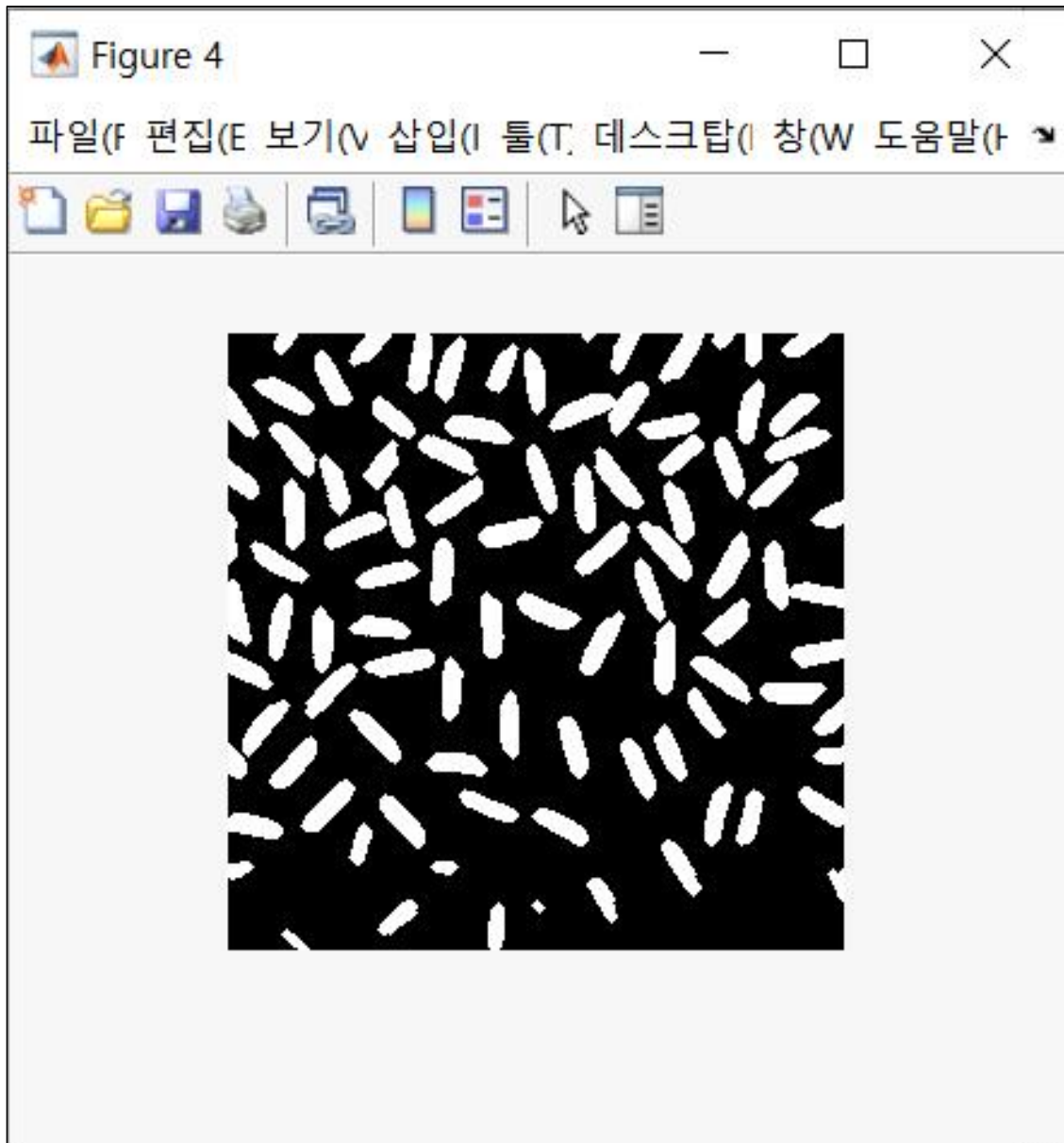
PDF of
Original Image

figure(2)는 'rice1.png'의 픽셀값을 나타내는 히스토그램이다. 히스토그램에서 볼 수 있는 3개의 봉우리 중 맨 왼쪽 봉우리는 다른 부분에 비해 어둡게 찍힌 배경으로 인해 나타난다. 가운데 봉우리는 회색 계열의 배경색으로 인한 것이고 맨 오른쪽 봉우리는 쌀알에 해당하는 픽셀값의 분포를 보여준다. 이미지의 히스토그램에서 배경에 해당하는 봉우리가 2개가 보이는 이유는 이미지를 획득할 때 조명이 개입되었기 때문이다. 본론-2)에서 이로 인해 발생하는 패턴 인식의 오류를 개선하는 방법에 대해 알아본다.



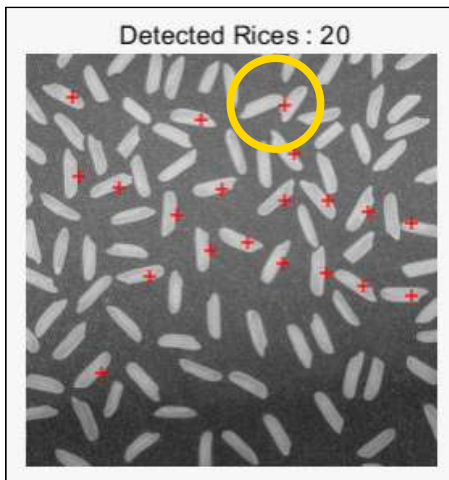
Binary Image of threshold = 128

figure(3)은 threshold = 128일 때 원본 이미지를 이진화한 모습이다. threshold는 이진화할 때 사용되는 기준값으로 픽셀의 값이 threshold보다 크면 1(흰색), threshold보다 작으면 0(검정색)으로 변환한다. 조명으로 인해 어둡게 찍힌 하단에 위치한 일부 쌀알의 경우 데이터가 많이 소실되었음을 볼 수 있다. 또한 배경의 이진화 과정에서 salt 잡음이 다소 섞여있음을 확인하였다.



Erosion & Dilation to remove noise

배경을 이진화하면서 생긴 노이즈를 제거하기 위해 erosion(침식)을 수행하였다. 침식을 수행하면 salt 잡음과 같은 작은 노이즈를 쉽게 없앨 수 있고, 서로 붙어있는 쌀알을 떨어뜨려 놓는 효과도 얻을 수 있다. 다만 쌀알의 정보도 함께 소실될 우려가 있어 사용자가 적당한 침식 크기의 값을 지정해야 한다. 여기에서는 erosion의 역과정인 dilation(팽창)도 수행하였다. figure(4)에 따르면 이진화 과정에서 정보가 많이 소실되었던 하단의 일부 쌀알은 erosion이 수행되며 아예 사라졌음을 알 수 있다. 노이즈는 사라졌지만, 거리가 가까운 쌀알이 서로 연결되어있는 문제점은 여전히 해결되지 않아 패턴인식의 수행에서 차질이 생길 것으로 예상하였다.



Detected Rice
in order of size

아래에 기술한 Matlab 코드에 따라 이진화 후 후처리 된 쌀알의 area 중 그 값이 큰 상위 20개의 쌀알을 검출하였다. 노란색으로 표시한 부분은 2개의 쌀알이 하나로 인식된 것으로 후처리 과정에서 erosion을 통해 각 쌀알의 영역을 확실하게 분리해 줄 필요성이 있음을 보여 준다.

또한 쌀알의 개수를 직접 세본 결과 총 101개인데, Matlab 코드에서 Num = 101로 바꾸어 코드를 수행하면 다음과 같은 오류가 발생한다.

```

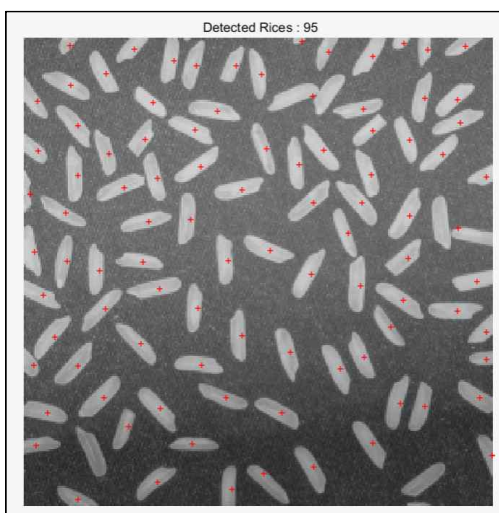
명령 창
위치 1의 인덱스가 배열 경계를 초과합니다. 인덱스는 95을(를) 초과해서는 안 됩니다.

오류 발생: tabular/dotParenReference (111번 라인)
    b = b(rowIndices,colIndices);

오류 발생: Test_DetectingRiceGrains (51번 라인)
    r = ordered.Centroid(n,1);

fx >>
  
```

쌀알의 후보로 택할 수 있는 이진화 이미지의 흰색 영역의 개수가 최대 95개라는 뜻으로 101개의 모든 쌀알을 인식할 순 없음을 나타낸다. Num = 95로 낮추어 코드를 수행한 결과이다.



- ① 2개의 쌀알을 1개로 인식
- ② 일부 쌀알 인식 불가

결과를 분석해보면 다음과 같은 2가지의 문제점이 발생하였음을 알 수 있다.
문제점 해결 방안은 본론-2)에서 다루도록 한다.

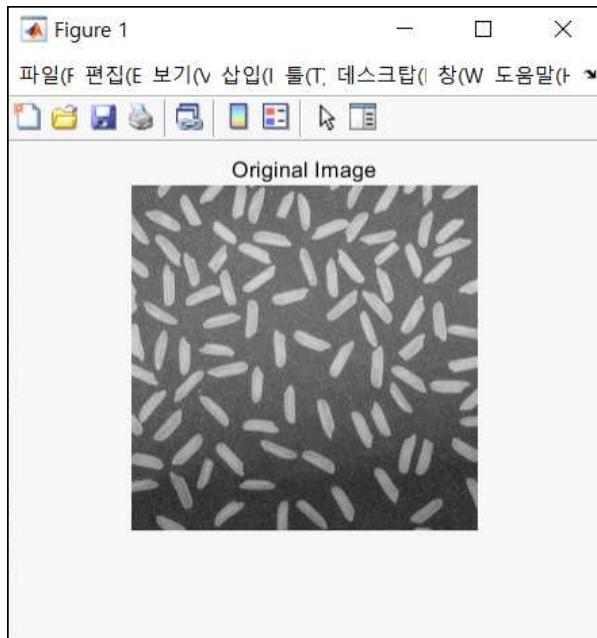
| Matlab code | |
|---|--|
| <pre> % Detecting rice grains clear; clc; % Read an image img = imread("rice1.png"); if size(img, 3)==1 gray = img; else gray = rgb2gray(img); end figure(1); imshow(gray); % Plot the PDF figure(2); imhist(gray); title('PDF of Image'); % Thresholding th = 128; imgB = gray>th; figure(3); imshow(imgB); % Binary filtering se = strel('diamond', 2); imgB = imerode(imgB, se); % 깎으면 붙어있는 쌀알이 없어지는 효과를 얻음. imgB = imdilate(imgB, se); % 깎고 팽창시키면 noise가 제거됨. figure(4); imshow(imgB); </pre> | <ol style="list-style-type: none"> 1. 이미지 읽기: "rice1.png" 파일을 읽고, 컬러 이미지인 경우에는 그레이스케일로 변환합니다. 2. 그레이스케일 이미지 출력: 그레이스케일 이미지를 출력하여 원본 이미지를 확인할 수 있습니다. 3. 화소 분포 함수(PDF) 플롯: 그레이스케일 이미지의 화소 분포 함수를 계산하여 히스토그램을 플롯합니다. 이를 통해 이미지의 밝기 분포를 시각화할 수 있습니다. 4. 이진화: 임계값(threshold)을 기준으로 그레이스케일 이미지를 이진화합니다. 임계값 이상인 픽셀은 흰색으로, 임계값 이하인 픽셀은 검은색으로 설정됩니다. 5. 바이너리 필터링: 다이아몬드 형태의 구조 요소를 사용하여 이진화된 이미지에 이진 필터링을 적용합니다. 먼저, 침식(erosion)을 통해 이미지의 주변 잡음을 제거하고, 이어서 팽창(dilation)을 통해 객체들을 보다 완전하게 만듭니다. 이러한 과정을 통해 쌀알들을 분리하고 잡음을 제거합니다. |

| Matlab code | |
|---|---|
| <pre> stats = regionprops(imgB, {'Area', 'Centroid'}); tab = struct2table(stats); % Sorting ordered = sortrows(tab, 1, "descend"); % Show results figure(5); imshow(img); hold on; Num = 20; title([' Detected Rices : ', num2str(Num)]); for n=1:Num r = ordered.Centroid(n,1); c = ordered.Centroid(n,2); % text(r,c,num2str(n)); text(r,c, '+', 'Color', 'red'); end hold off; </pre> | <p>6. 객체 특징 추출: 이진화 및 바이너리 필터링된 이미지에서 객체의 특징을 추출합니다. 'Area'와 'Centroid'를 추출합니다. 'Area'는 객체의 면적을, 'Centroid'는 객체의 중심 좌표를 나타냅니다.</p> <p>7. 정렬: 추출된 객체 특징을 'Area'를 기준으로 크기순으로 정렬합니다.</p> <p>8. 결과 표시: 원본 이미지 위에 감지된 쌀알을 텍스트 "+"로 표시합니다. 'Num' 변수에 지정된 개수만큼 상위 쌀알을 표시합니다.</p> |

2) Improved Image Segmentation by Thresholding

앞서 알아본 쌀알의 인식 불가 문제점을 개선하기 위해 다음과 같은 방안을 제시할 수 있다.

- ❑ 이미지에 나타나는 조명의 영향을 최소화 (이미지의 히스토그램 조작)
(대역폭이 좁은 Low Pass Filter를 사용, 즉 filter의 크기를 매우 큰 값으로 지정
e.g. Gaussian Filter, Average Filter, etc.)
- ❑ Optimized Binary Image threshold 값 자동제어



Original Image

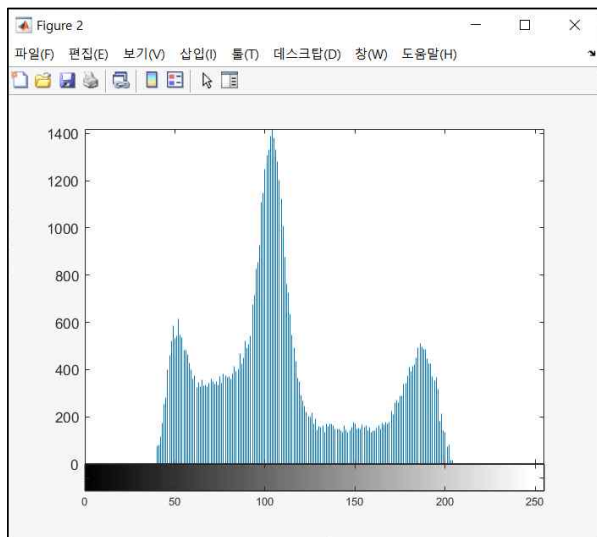
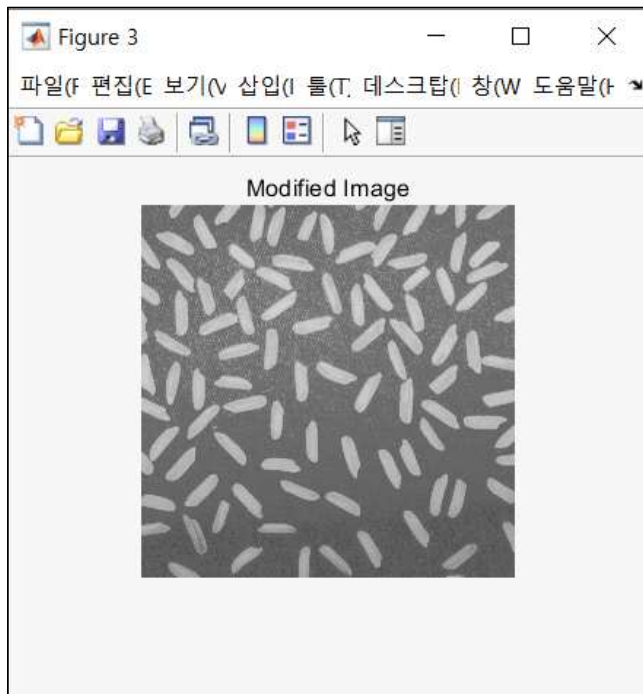


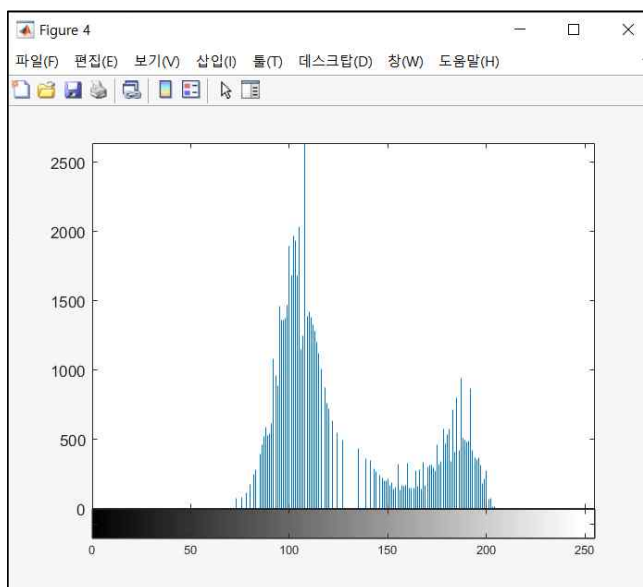
Image Histogram
of Original Image

다음의 figure는 본론-1)에서 설명한 내용과 일치한다.



Background
Equalized Image

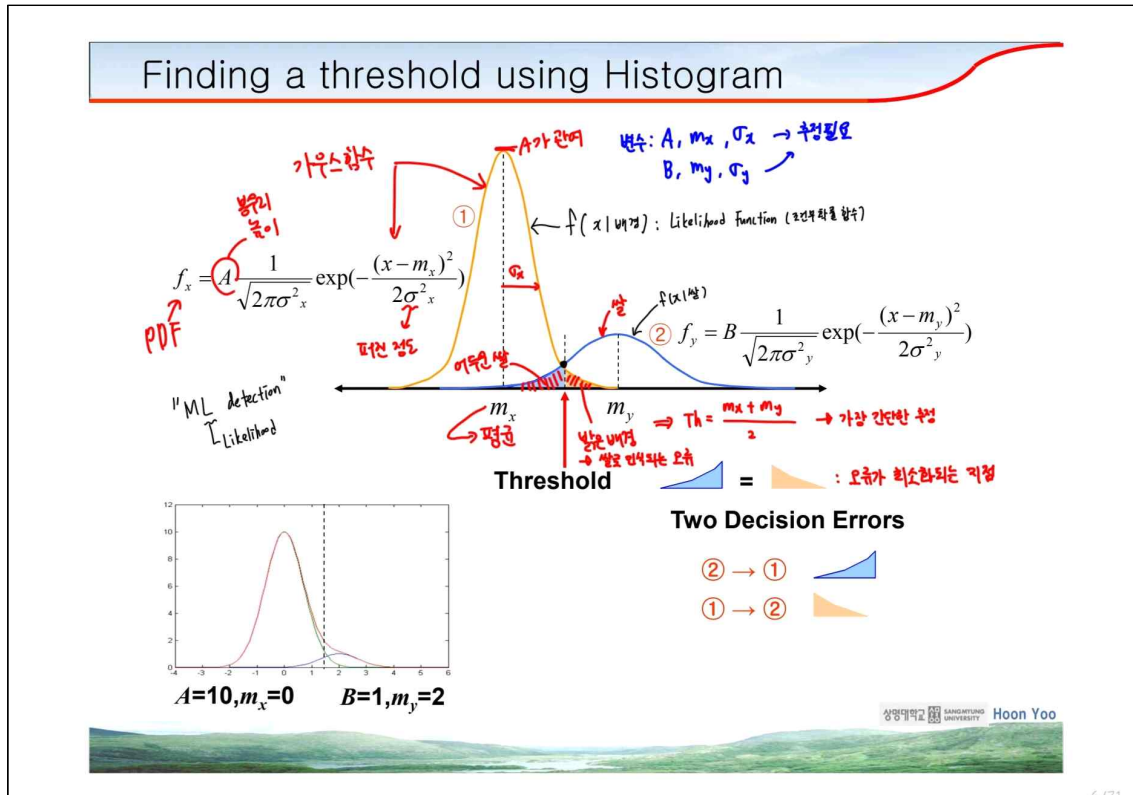
figure(3)에 수정된 히스토그램이 적용된 이미지를 출력하였다. 아래의 figure(4)에 해당하는 이미지 히스토그램에서 배경의 봉우리가 하나로 통일되어 있기 때문에 배경의 밝기가 원본 이미지보다 일관되게 나타남을 알 수 있다.



Modified
Image Histogram

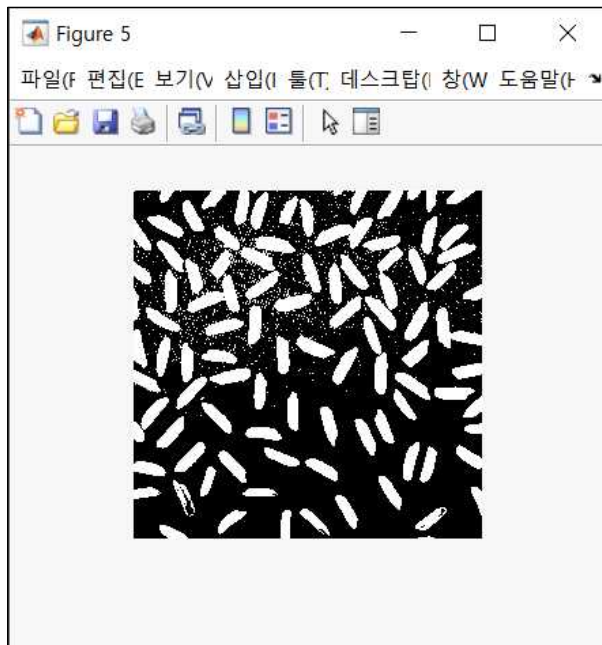
figure(4)는 원본 이미지의 히스토그램에서 맨 왼쪽에 해당하는 봉우리를 제거하는 과정을 거쳐 수정된 이미지의 히스토그램이다. 조작된 히스토그램의 두 봉우리는 가우스 분포와 비슷한 양상을 보이도록 하였으며, 두 개의 봉우리 중 왼쪽이 배경에 의해 나타나는 픽셀값의 분포도, 오른쪽이 쌀알에 의해 나타나는 픽셀값의 분포도이다.

만약 배경과 쌀알의 분포도가 모두 가우스 함수를 따른다고 가정하였을 때, 최적화된 threshold값은 다음의 과정을 통해 자동적으로 제어할 수 있다.



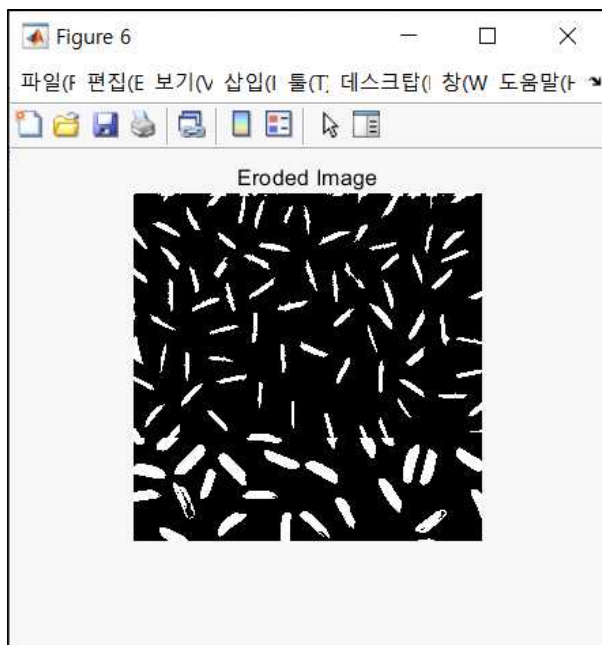
지능형영상처리 수업 中 threshold 제어에 대한 내용

하지만 figure(4)의 히스토그램은 완벽한 가우스 분포를 따르지 않기 때문에 식으로 직접 계산하는 것에 한계가 있었다. 따라서 여러 번의 반복 실험 결과 최적의 결과가 나타났던 140을 optimized threshold 값으로 결정하였다.



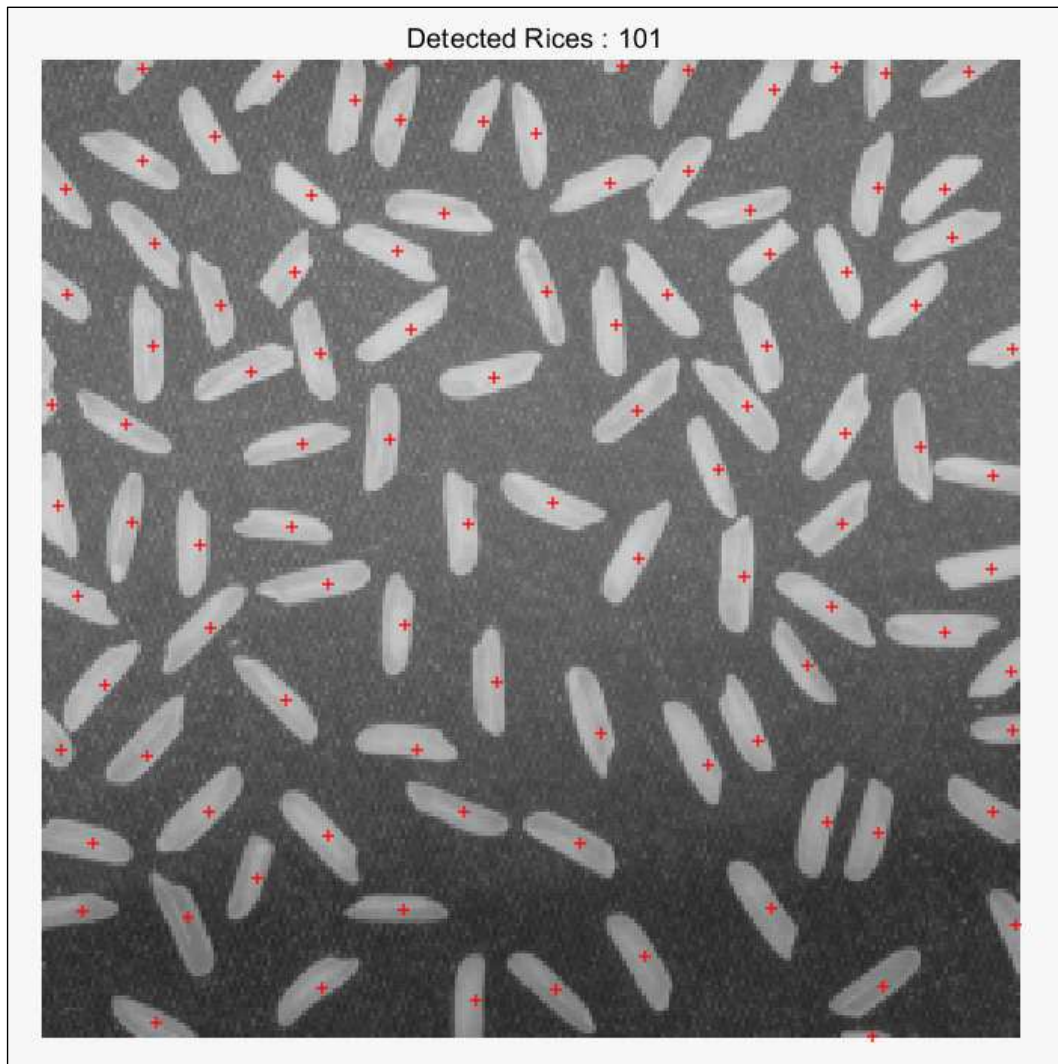
Binary Image of
threshold = 140

figure(5)는 threshold = 140으로 이진화한 결과 이미지이다. 여전히 salt 잡음이 다소 포함되어 있음을 알 수 있고 배경의 균일화로 인해 원본 이미지의 이진화에서 소실되었던 일부 쌀 알도 현재는 잘 보존됨을 확인하였다.



Eroded Image

노이즈를 제거하고 서로 영역을 공유하는 쌀알을 분리하기 위해 Erosion(침식) 과정을 수행하였다. figure(4)는 Matlab 코드에서 알 수 있듯이 데이터 소실을 방지하기 위해 필요한 범위에만 부분적으로 Erosion을 수행한 모습이다. 이미지 팽창은 필요하지 않아 Dilation(팽창) 과정은 생략하였다.



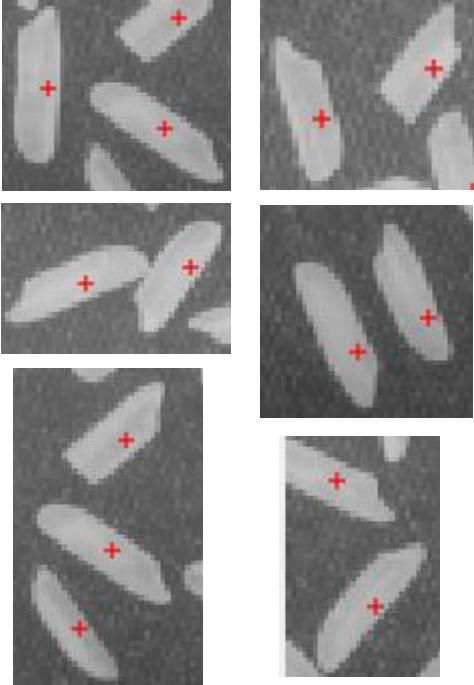

Result of Rice Grains Recognition

Matlab 코드는 이미지에서 쌀알을 검출하는 과정을 담고 있다. 히스토그램 조작과 이진화, 이진 필터링을 통해 노이즈를 제거하고, 영역 분석을 수행하여 쌀알의 중심점을 찾고 이미지에 표시하였다.

쌀알이 모두 탐지되는지 알아보기 위해 Num = 101로 설정 후 코드를 수행하였다. 본론-1)에서 발생하였던 문제가 모두 해결되어 101개의 쌀알이 전부 인식되었다.

다음의 표에서 검출된 쌀알의 크기를 2가지로 분류하고 특징을 서술하였다.

쌀알의 크기에 따른 분류와 특징

| | 크기가 큰 쌀알 | 크기가 작은 쌀알 |
|----|--|--|
| 사진 |  |  |
| 특징 | <p>크기가 큰 쌀알의 경우 이미지의 이진화와 침식(Erosion) 과정에서 비롯되는 데이터 소실이 해당 쌀알에 큰 영향을 미치지 않는다. 따라서 크기가 큰 쌀알은 정보의 유실에 상대적으로 둔감하다.</p> <p>따라서 사용자가 의도한 대로 쌀알 한가운데 비교적 가깝도록 인식 마크 '+'가 표시된다.</p> | <p>크기가 작은 쌀알의 경우 이미지의 이진화와 침식(Erosion)에서 조금만 데이터가 소실되어도 큰 쌀알에 비해 큰 비율의 정보를 잃는 것이다. 따라서 크기가 작은 쌀알은 정보의 유실에 민감하다.</p> <p>쌀알로 인식이 되었다고 해도 인식 과정에서 해당 쌀알 크기에 비해 많은 비율의 데이터가 유실되기 때문에 정확한 쌀알의 위치를 지정하기 어려워져 인식 마크 '+'가 쌀알의 중심점에 위치하기 힘들다.</p> |

| Matlab code | |
|---|--|
| <pre> clear; clc; % Read an image img = imread("rice1.png"); if size(img, 3) == 1 gray = img; else gray = rgb2gray(img); end figure(1); imshow(gray); title('Original Image'); figure(2); hist = imhist(gray); % Compute histogram figure(2); imhist(gray); % Set histogram values below x-axis 85 to 0 hist(1:85) = 0; % Interpolate values for x = 1 to 138 based on Gaussian curve x = 1:numel(hist); y = hist; % Find the peak position and values [peakValue, peakIndex] = max(hist); peakPosition = x(peakIndex); % Define Gaussian curve parameters based on the peak position and values sigma = (peakPosition - 85) / 2; % Standard deviation mu = peakPosition; % Mean % Generate Gaussian curve gaussianCurve = peakValue * exp(-(x - mu).^2 / (2*sigma^2)); </pre> | <ol style="list-style-type: none"> 1. 이미지를 읽어와 이미지가 흑백 이미지인지 컬러 이미지인지 확인합니다. 흑백 이미지인 경우 gray 변수에 이미지를 저장하고, 컬러 이미지인 경우 rgb2gray 함수를 사용하여 흑백 이미지로 변환합니다. 2. 원본 이미지와 해당 이미지의 히스토그램을 출력합니다. 3. 히스토그램에서 x축 값이 85보다 작은 부분을 0으로 설정합니다. 4. 히스토그램에서 x축 값이 85보다 작은 부분을 0으로 설정합니다. 5. 1에서 138 사이의 값에 대해 가우시안 곡선을 사용하여 보간(interpolation)합니다. 이를 위해 해당 범위의 x값과 히스토그램 값을 가지고 가우시안 곡선을 생성합니다. |

| Matlab code | |
|---|---|
| <pre> % Interpolate the values for x = 1 to 138 using the Gaussian curve interpolatedValues = interp1(x, gaussianCurve, 1:138, 'linear'); % Update the histogram values with the interpolated values hist(1:138) = interpolatedValues; % Apply modified histogram to the image modified_gray = histeq(gray, hist); % Display modified image figure(3); imshow(modified_gray); title('Modified Image'); % Display modified histogram figure(4); imhist(modified_gray); % Thresholding th = 140; imgB = modified_gray > th; figure(5); imshow(imgB); </pre> | <p>6. 생성된 가우시안 곡선을 사용하여 1에서 138 사이의 값을 보간합니다. interp1 함수를 사용하여 x값을 1부터 138까지 선형적으로 보간하고, 해당 보간값으로 히스토그램을 업데이트합니다.</p> <p>7. 수정된 히스토그램을 적용하여 이미지를 수정합니다. histeq 함수를 사용하여 히스토그램을 평활화하고, 수정된 이미지를 modified_gray 변수에 저장합니다. 또한 수정된 이미지와 수정된 히스토그램을 출력합니다.</p> <p>8. 임계값(threshold)을 설정하여 이진화(binanzation)를 수행합니다. 임계값을 기준으로 픽셀 값을 비교하여 이진 이미지를 생성합니다.</p> |

| Matlab code | |
|---|--|
| <pre> % Binary filtering se1 = strel('diamond', 3); % Apply erosion to the specified horizontal region horizontalRegion = imgB(3:180, :); horizontalRegion = imerode(horizontalRegion, se1); imgB(3:180, :) = horizontalRegion; se2 = strel('diamond', 2); % Apply erosion to the specified vertical region vertitalRegion = imgB(:, 160:165); vertitalRegion = imerode(vertitalRegion, se2); imgB(:, 160:165) = vertitalRegion; figure(6); imshow(imgB); title('Eroded Image'); stats = regionprops(imgB, {'Area', 'Centroid'}); tab = struct2table(stats); % Sorting ordered = sortrows(tab, 1, "descend"); % Show results figure(8); imshow(img); hold on; Num = 101; title([' Detected Rices : ', num2str(Num)]); for n = 1:Num r = ordered.Centroid(n,1); c = ordered.Centroid(n,2); text(r, c, '+', 'Color', 'red'); end hold off; </pre> | <p>9. 이진 필터링을 사용하여 노이즈를 제거합니다. strel 함수를 사용하여 다이아몬드 형태의 구조 요소(strel)를 생성하고, imerode 함수와 imdilate 함수를 사용하여 이진 이미지를 깎고 팽창시킴으로써 노이즈를 제거합니다.</p> <p>10. regionprops 함수를 사용하여 이진 이미지에서 각 영역의 속성과 중심점을 추출합니다. 추출된 속성은 stats 변수에 저장되고, struct2table 함수를 사용하여 테이블 형태로 변환하여 tab 변수에 저장합니다.</p> <p>11. 영역의 크기를 기준으로 정렬합니다.</p> <p>12. 결과 이미지에 각 쌀알을 표시합니다. Num 변수에 저장된 값만큼 상위 쌀알 영역을 선택하고, 해당 영역의 중심점을 가져와서 이미지에 텍스트 또는 마커를 표시합니다.</p> |

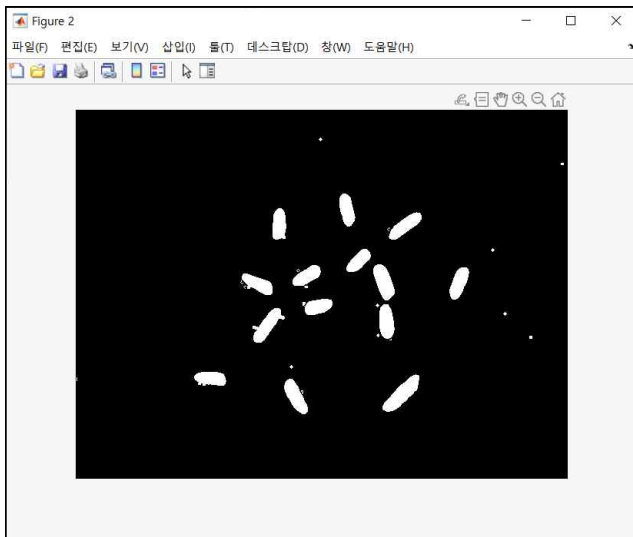
3) High Pass Filter for Detecting Edges

한편, 배경과 쌀알이 명확하게 색으로 구분되는 'rice1.png'와 다르게 'rice2.png'의 경우 쌀알과 배경의 색이 비슷해 히스토그램을 이용하여 thresholding하는 것에 어려움이 발생한다. 따라서 고주파 필터를 이용해 원본 이미지의 에지를 강화하고 에지 분석을 통해 쌀알을 인식하는 방법을 통해 실습을 접근하였다. 자세한 설명은 다음과 같다.



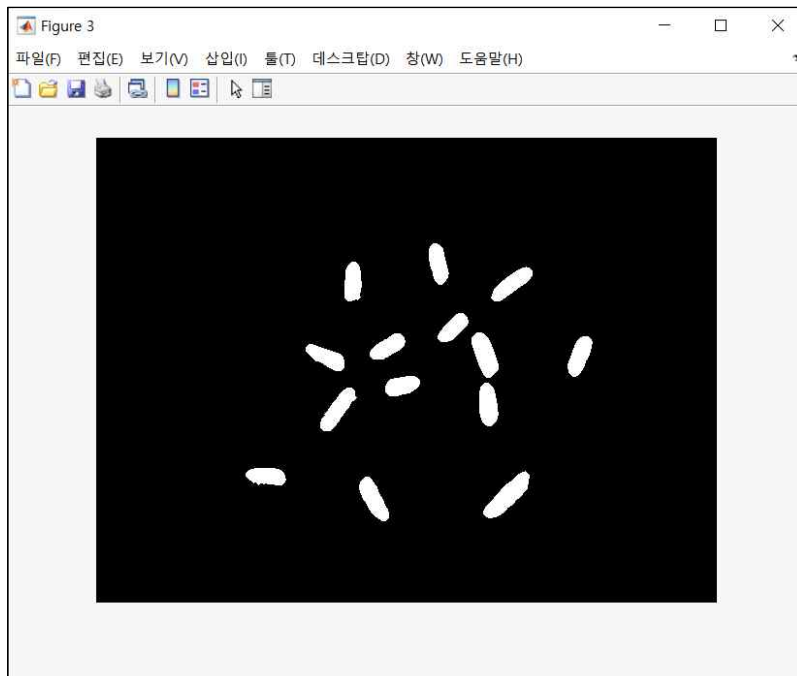
Original Image

figure(1)은 'rice1.png'의 원본을 나타낸다.



Modified
Binary Image

원본 이미지에 Sobel 필터를 적용하여 고주파 성분을 감지하고 Canny 필터를 다시 적용해 에지 성분을 더 정확하게 찾아내었다. 이후 에지 영역으로 인식된 곳을 흰색(1), 그렇지 않은 곳을 0으로 이진화하여 만든 Binary Image가 figure(2)이다. 쌀알 이외의 작은 잡음이 포함되어 있고 앞서 활용한 Erosion(침식)을 통해 쉽게 제거할 수 있다.



Noise removed
Binary Image

figure(3)은 Erosion(침식) 과정을 통해 노이즈를 제거하고 쌀알의 정보만 남긴 Binary Image이다.



Result of
Rice Grains
Recognition

주어진 이미지에서 쌀알을 검출하기 위해 에지 검출과 이진화를 사용하였다. 이진 필터링을 통해 노이즈를 제거하고, 영역 분석을 수행하여 쌀알의 중심점을 찾고 이미지에 표시하였다. 총 14개의 쌀알이 모두 인식되는지 확인하기 위해 Num = 14로 설정 후 코드를 수행한 모습이다.

| Matlab code | |
|---|---|
| <pre> clear; clc; fname = "rice2.png"; img = imread(fname); figure(1); imshow(img); % Convert image to grayscale img_gray = rgb2gray(img); % Edge detection using Sobel & Canny filter edge_img = edge(img_gray, 'sobel'); edge_img = edge(edge_img, 'canny'); % Fill the edge area with white filled_img = imfill(edge_img, 'holes'); % Display the filled image figure(2); imshow(filled_img); % Binary filtering imgB = filled_img; se = strel('diamond', 3); imgB = imerode(imgB, se); imgB = imdilate(imgB, se); figure(3); imshow(imgB); </pre> | <ol style="list-style-type: none"> 1. 읽어 들인 이미지를 흑백 이미지로 변환합니다. 2. Canny 알고리즘을 사용하여 에지(Edge)를 검출합니다. 먼저 Sobel 필터를 사용하여 이미지에 에지를 감지한 후, Canny 필터를 적용하여 더 정확한 에지를 추출합니다. 3. 에지 영역을 흰색으로 채웁니다. imfill 함수를 사용하여 에지 영역을 채워진 이미지를 생성합니다. 4. 채워진 이미지를 이진 필터링하여 노이즈를 제거합니다. imerode 함수와 imdilate 함수를 사용하여 이미지를 깎고 팽창시킴으로써 노이즈를 제거합니다. |

| Matlab code | |
|---|---|
| <pre> stats = regionprops(imgB, {'Area', 'Centroid'}); tab = struct2table(stats); % Sorting ordered = sortrows(tab, 1, "descend"); % Show results figure(4); imshow(img); hold on; Num = 14; title([' Detected Rices : ', num2str(Num)]); for n=1:Num r = ordered.Centroid(n,1); c = ordered.Centroid(n,2); % text(r,c,num2str(n)); text(r,c,'+', 'Color','red'); end hold off; </pre> | <p>5. 이진 이미지에서 각 영역의 속성과 중심점을 추출합니다. regionprops 함수를 사용하여 영역의 면적과 중심점을 추출하고, 추출된 속성을 테이블 형태로 변환하여 tab 변수에 저장합니다.</p> <p>6. 영역의 크기를 기준으로 정렬합니다.</p> <p>7. 결과 이미지에 각 쌀알을 표시합니다. Num 변수에 저장된 값만큼 상위 쌀알 영역을 선택하고, 해당 영역의 중심점을 가져와서 이미지에 텍스트 또는 마커를 표시합니다.</p> |

III. 결론

이번 실습에서는 영상 분할에 대해 학습하였다. 쌀알 패턴을 인식하는 과정에서 다음과 같은 난관이 있었다.

- ① 히스토그램 조작을 통한 배경 균일화
- ② 2개로 인식되는 쌀알 분리하기
- ③ 쌀알과 배경이 비슷한 색일 경우 thresholding의 사용 불가

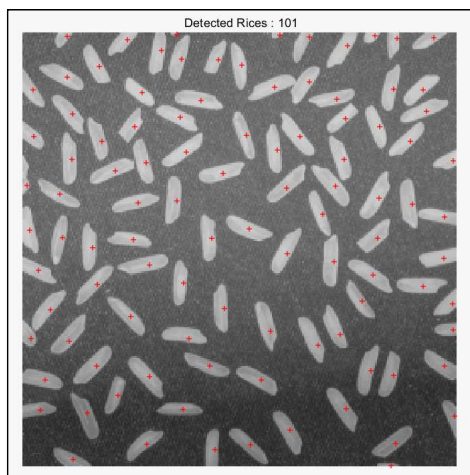
히스토그램을 조작하여 배경을 균일화하는 작업에서 가장 큰 시간을 쏟았다. 하지만 완벽한 가우스 분포와는 다소 거리감이 있는 히스토그램을 얻은 것이 최선이었다. 추후 더 연구해보며 가우스 분포에 가까운 히스토그램으로 개선할 것이다.

쌀알 2개가 1개로 인식되도록 Erosion하는 과정에서, 수업 시간에 배운대로 이미지 전체를 Erosion하니 영역을 공유하는 쌀알을 떨어뜨려 놓으면 크기가 작은 쌀알의 정보가 유실되는 딜레마가 있었다. 이미지의 일부만 Erosion하는 코드를 적용하여 문제를 해결하였다.

마지막으로 thresholding이 사용 불가능했던 'rice2.png'의 경우 High Pass Filter를 이용하여 에지를 감지해 비교적 쉽게 쌀알을 감지해냈다.

이 과정을 수행하며 저번 주의 Lab03_Pattern Recognition에서 점을 인식하는 실습 중 미처 해결하지 못했던 부분에 대한 솔루션을 찾게 되었다. 특히 그동안 학습했던 코드를 토대로 ChatGPT의 큰 도움 없이 직접 코드를 작성했다는 점에서 큰 의미가 있었다.

앞으로 남은 Term Project에서는 그동안 배운 Point Operations, Image Composition, Pattern Recognition, Image Segmentation을 아우르는 주제를 선정하여 수행할 것이다.



'rice1.png'



'rice2.png'