



# Healthy Fruit Selector

:DSP Project

---

1685029 구지희 1772022 신주연 1772049 최혜원

The slide is decorated with various hand-drawn fruit illustrations in a whimsical style. These include a blueberry cluster at the top left, a lemon slice at the top center, a green kiwi slice at the top right, a watermelon slice at the top right, a lime at the middle right, a lemon slice at the middle right, a strawberry at the bottom left, a banana at the bottom left, a green kiwi slice at the bottom center, a cherry at the bottom right, and an orange at the bottom right. The word 'CONTENTS' is centered on the left side of the slide.

# CONTENTS

- 
1. MOTIVATION
  2. FLOW CHART
  3. KEY CONCEPT & PROCESS
    - IMAGE BINARIZATION
    - MEASURE DIAMETER
    - DETECT DAMAGE PART
  4. DEMO
  5. REFERENCE

# 01. Motivation

## 비파괴 감귤선별기 현장컨설팅 만족도 높다

ⓒ 위계욱 기자 | ⓒ 승인 2018.11.30 10:01 | ⓒ 댓글 0

| 고품질 감귤 유통으로 생산자소비자 동시만족

문제는 비파괴 감귤선별기가 워낙 덩치가 크고 다양한 시스템이 접목된 장치라 일반인들은 선별기 시스템을 이해하고 운영하는 것은 물론 설치가 잘 됐는지조차 판가름할 수 없다는 점이다.



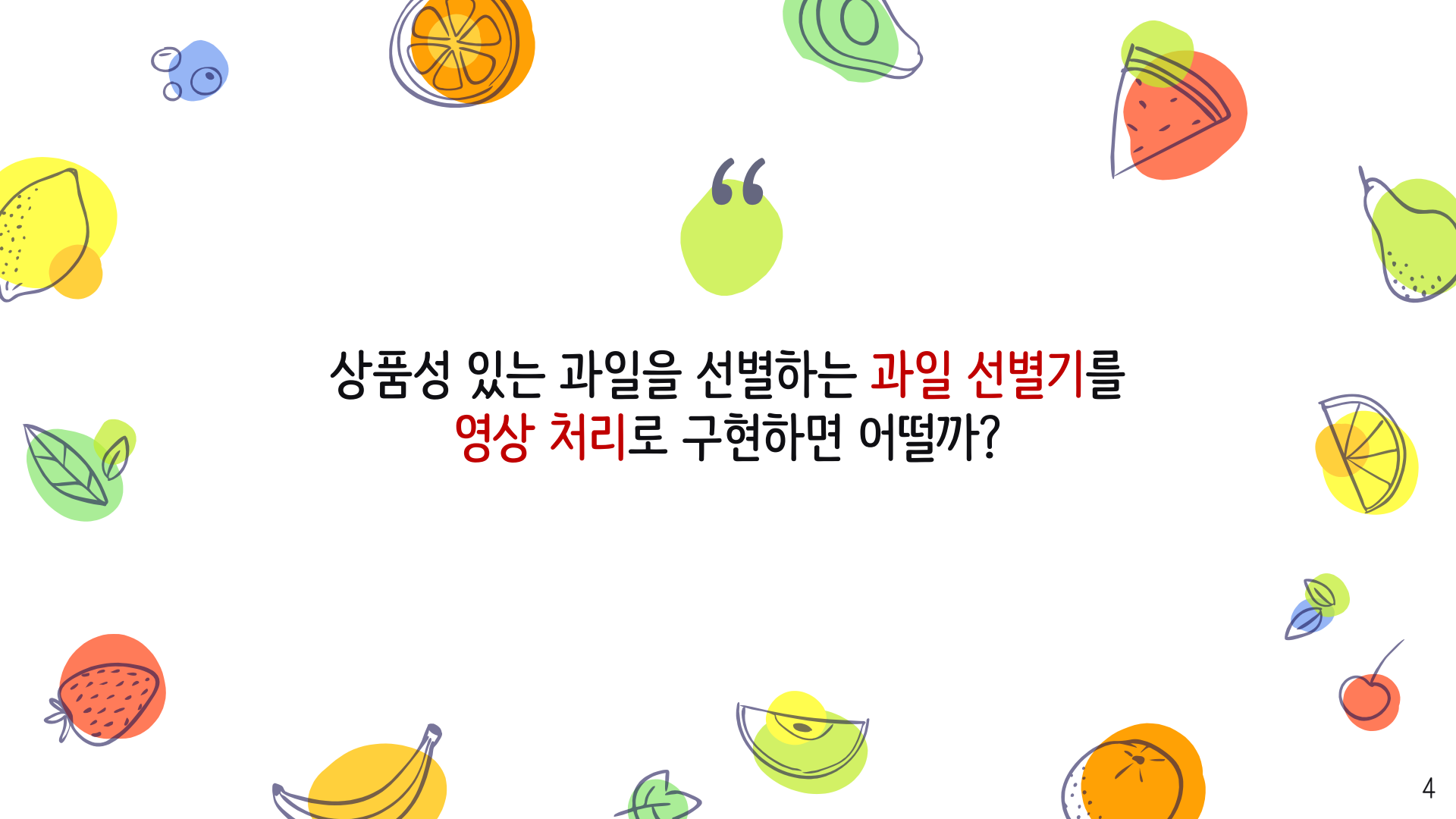
딥러닝으로 비파괴 감귤 선별기를 구현한 사례  
빛을 이용하여 귤의 당도와 산도를 측정하는 비파괴 선별기



다양한 시스템이 접목된 큰 선별기라  
일반인들이 이해하기 어렵다.

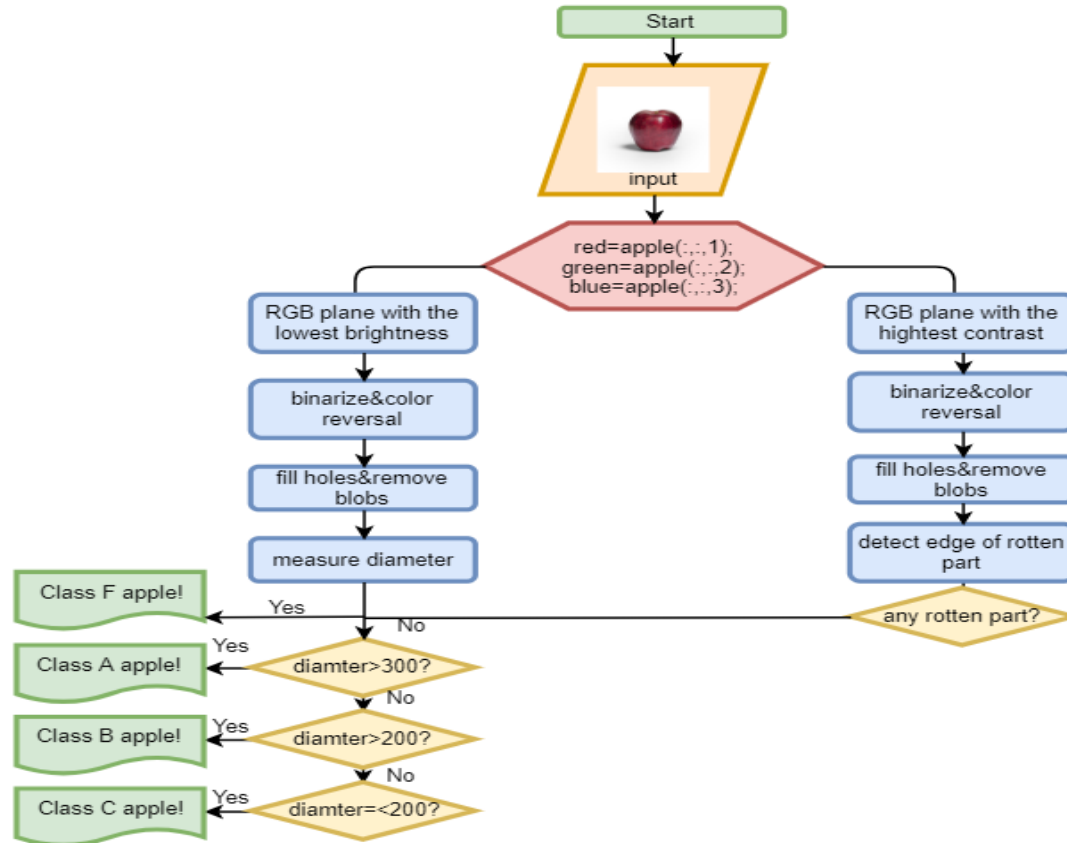


과일의 선별과정을 눈으로 직접 볼 수 있는  
작은 비파괴 과일 선별기를 만들면 어떨까?



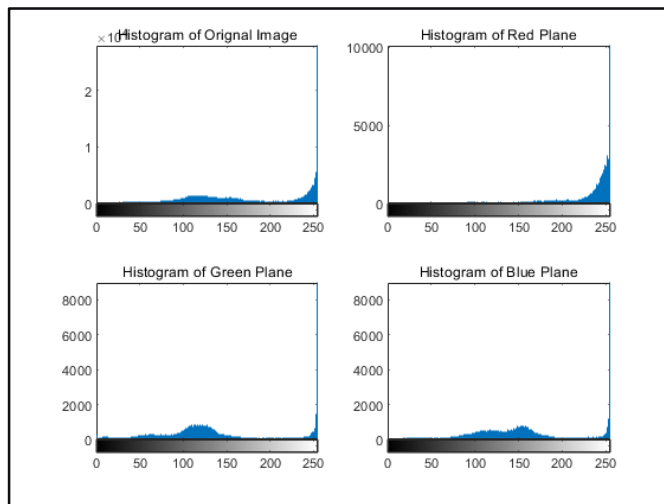
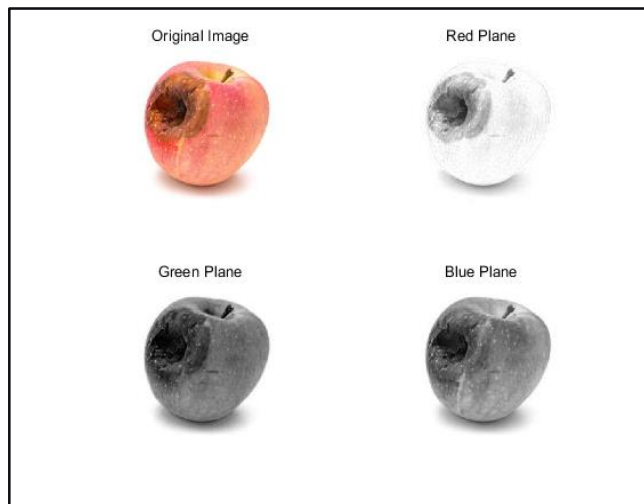
상품성 있는 과일을 선별하는 과일 선별기를  
영상 처리로 구현하면 어떨까?

# 02. Flow Chart



## 03. Key Concept & Process

### 1) Image Binarization : RGB plane, imhist()



```
%% 1)segment image
%divide image into
%its respective RGB intensities
red=apple(:,:,1);
green=apple(:,:,2);
blue=apple(:,:,3);

%histogram 보여주기
figure;
subplot(2,2,1); imhist(apple)
subplot(2,2,2); imhist(red);
subplot(2,2,3); imhist(blue);
subplot(2,2,4); imhist(green)
```

이미지를 각각 RGB 요소로 나누고, 히스토그램을 이용해 밝기 값을 수치적으로 확인

- 히스토그램(histogram): 영상의 밝기 값에 대한 분포를 보여주는 그래프
- imhist(): 회색조 영상의 히스토그램을 계산. 가로축은 0에 가까울수록 어두운 색, 256에 가까울수록 밝은 색을 의미

# 03. Key Concept & Process

## 1) Image Binarization : select RGB plane

```

[cntR,binR]= imhist(red);
[cntB,binB]= imhist(blue);
[cntG,binG]= imhist(green);

%diameter는 어두운 색상 많이 갖고있는 plane 선택
%damage는 밝은 색상 ""
[M,index1] = max([sum(cntR(1:100,:)), sum(cntB(1:100,:)), sum(cntG(1:100,:))]);
[M,index2] = min([sum(cntR(1:100,:)), sum(cntB(1:100,:)), sum(cntG(1:100,:))]);

if index1 == 1
    thPlane1 = red;
elseif index1 == 2
    thPlane1 = blue;
else
    thPlane1 = green;
end

if index2 == 1
    thPlane2 = red;
elseif index2 == 2
    thPlane2 = blue;
else
    thPlane2 = green;
end

```

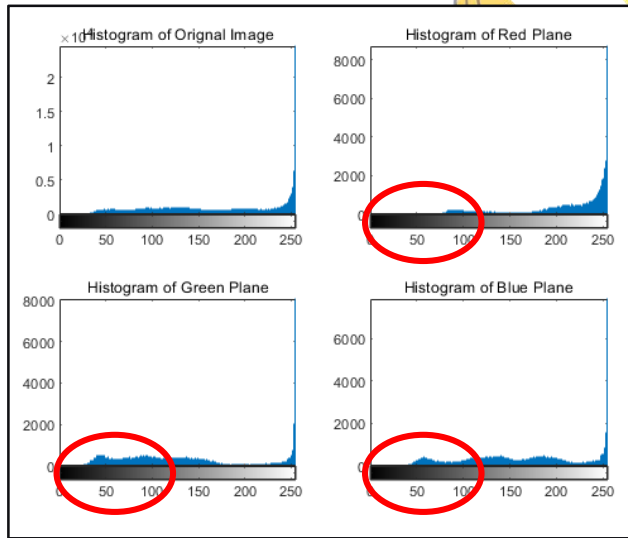
RGB 각각의 밝기 정도를 분석하여 목적에 맞는 plane 적용

- Diameter 측정 : 1~100사이의 값 분포가 가장 큰 것
- Damage 감지 : 1~100 사이의 값 분포가 가장 작은 것  
(\* 0에 가까울수록 어두운 색, 256에 가까울수록 밝은 색)

diameter plane



damage plane



# 03. Key Concept & Process

## 1) Image Binarization : imbinarize()

```
% 3) 이미지 바이너리화
-%threshold the chosen plane
figure;
%binary할 plane의 임계값을 이미지 전체에서 자동으로 설정해주는 내장함수
bw1=imbinarize(thPlane,0.65);
```

```
% 3) 이미지 바이너리화
figure;
bw1=imbinarize(thPlane,0.95);
subplot(2,2,1); imshow(bw1); title('chosen plane for damaged apple detection');
bw2 = imcomplement(bw1);
```

```
subplot(2,2,1); imshow(bw1); title('chosen plane for damaged apple detection');
bw2 = imcomplement(bw1);

%remove noise
fill=imfill(bw2, 'holes'); %fill any holes
subplot(2,2,2); imshow(fill); title('Holes filled');

%remove any alobs on the border of the image
clear1 =imclearborder(fill);
subplot(2,2,3); imshow(clear1); title('Remove blocs on border');

%Remove blobs that are smaller than 7 pixels across
se=strel('disk',7);
open=imopen(fill,se);
subplot(2,2,4); imshow(open); title('Remove small blobs');
```

**imbinarize( image , level )**: 임계값을 기준으로 이미지를 이진화

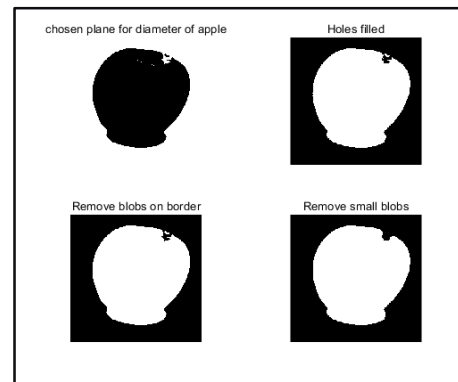
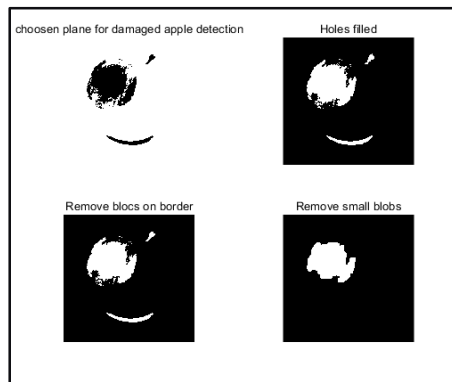
\* level : 0~1 사이 값으로, test을 통해서 얻어낸 임계값

이미지 영역 구분을 명확하게 하기 위해 hole fill, 작은 BLOB 제거

\* imfill() 함수는 hole을 흰색으로 채우기 때문에 색 반전 필요

배경과 물체의 색 대비가 커야 정확한 결과 도출

→ input이 흰색 배경에서 찍은 사진이어야 한다는 한계점





# 03. Key Concept & Process

## 2) Measure Diameter(size)

```
%diameter classification
if(diameter>300)
    class='A';
    result = 'A';
elseif(diameter>200)
    class='B';
    result = 'B';
else
    class='C';
    result = 'C';
end
```

### %% 4) 사과 직경 재기 Measure apple diameter

```
stats=regionprops(open, 'Centroid', 'MajorAxisLength', 'MinorAxisLength');
center=stats.Centroid;
diameter = mean([stats.MajorAxisLength stats.MinorAxisLength],2);
radii = diameter/2;
```

### imshow()

: 같은 거리에서 촬영했다는 가정을 위해 모든 input 사진을 400x400으로 조절하고 크기를 비교

### regionprops()

: 객체의 질량 중심과 주요 축 길이, 보조 축 길이를 반환.  
여기서 얻은 두 축 길이의 평균이 직경으로 도출  
\*사과는 대체로 구 형을 띄기 때문에 직경으로 사이즈를 판단  
\*길게 늘어진 그림자가 있을 경우 직경 측정에 오차가 발생할 수 있음

평균 직경 길이를 임의의 기준으로 나누어 크기 등급 분류



diameter 303.7600



diameter 201.2879

# 03. Key Concept & Process

## 3) Detect Rotten part

%% 3) 이미지 바이너리화

```
figure;
bw1=imbinarize(thPlane,0.4);
subplot(2,2,1); imshow(bw1);
bw2 = imcomplement(bw1);
```

%% 4) 손상부분 발견

```
[B]=bwboundaries(open);
damage_count = length(B);
figure;
imshow(apple);
hold on;
for k=1:length(B), boundary = B{k};
    plot(boundary(:,2), boundary(:,1), 'r','LineWidth',2);
end
```

%damage 여부

```
if damage_count > 0
    status = '썩은 사과이다'
    result = 'F'
else
    status = '썩지 않은 사과이다'
end
```

썩은 부분 검출은 이진화의 임계값 0.35로 지정  
(sample 사진들로 test한 결과 임계값이 0.3~0.4일 때 구분 가능)

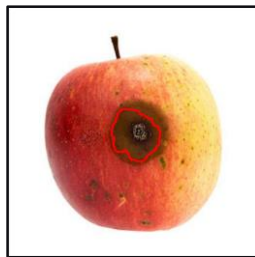
\*한계점: test로 구한 임계 값이기 때문에 사진 밝기와 그림자에 의한 오차 가능성

**bwboundaries()**

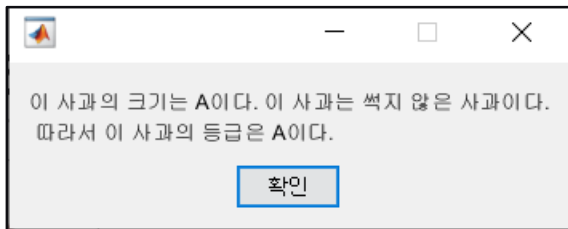
: 이진 영상에 있는 객체의 외부 경계선을 추적

썩은 부분의 경계선 픽셀 위치로 구성된 셀형 배열 B를 반환

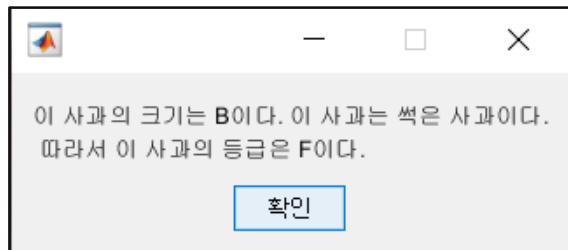
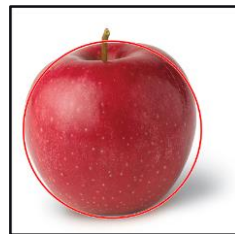
Loop를 통해 경계선을 boundary 변수에 저장하고 썩은 부분을 표시.



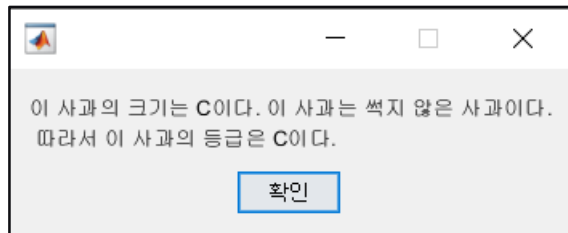
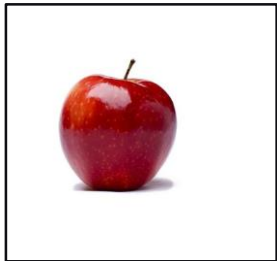
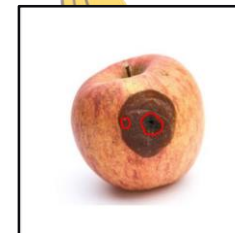
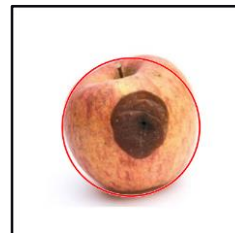
## 04. Demo



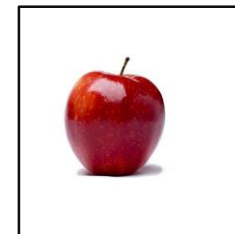
damage\_count 0  
diameter 312.7352



damage\_count 1  
diameter 263.0602



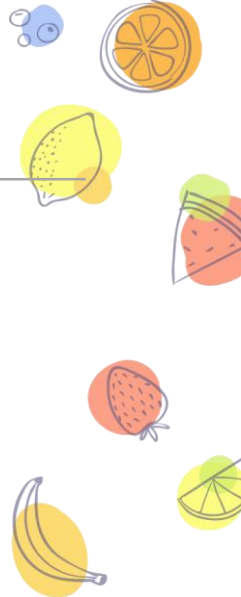
damage\_count 0  
diameter 183.1602



## 05. REFERENCE

### 참고문헌

- [1] 위계욱 기자, “비파괴 감귤선별기 현장 컨설팅 만족도 높다.”, 농업인 신문, 2018.11.30,  
<http://www.nongupin.co.kr/news/articleView.html?idxno=53422>
- [2] 신규식 기자, “햇사레 복숭아 맛 보세요”, CJB 뉴스, 2019-08-02 ,  
[https://www.cjb.co.kr/home/sub.php?menukey=61&mod=view&P\\_NO=190731010&PRO\\_CODE=4](https://www.cjb.co.kr/home/sub.php?menukey=61&mod=view&P_NO=190731010&PRO_CODE=4)
- [3] 영상 데이터의 히스토그램, “imhist”, <https://kr.mathworks.com/help/images/ref/imhist.html> (2019.11.30)
- [4] Marhworks Image Processing toolbox 문서, “RGB Plane”, <https://kr.mathworks.com/help/images/Display-Separated-Color-Channels-of-an-RGB-Image.html>(2019.11.30)
- [5] 임계값을 지정하여 2차원 회색조 영상 이진화 , “imbinarize”,  
<https://kr.mathworks.com/help/images/ref/imbinarize.html> (2019.11.30)
- [6] 영상 영역의 속성 측정, “regionprops”, <https://kr.mathworks.com/help/images/ref/regionprops.html> (2019.11.30)
- [7] Marhworks Image Processing toolbox 문서, “bwboundaries”,  
<https://kr.mathworks.com/help/images/ref/bwboundaries.html> (2019.11.30)
- [8] 원 안의 가장 큰 사각형 자르기, “가장 큰 원” <https://coday.me/ko/qa/20190824/1340905.html>(2019.11.20)
- [9] 경계 검출과 모폴로지를 사용하여 세포 검출하기, “edge detection”,  
<https://kr.mathworks.com/help/images/detecting-a-cell-using-image-segmentation.html> (2019.12.1)
- [10] Digital image processing, “Image Processing Color Representation ”,  
<http://ceng503.cankaya.edu.tr/uploads/files/Digital%20Image%20Processing-6.pdf> (2019.11.20)
- [11] 오토(Otsu) 방법을 사용한 전역 영상 이진화, “graythresh”,  
<https://kr.mathworks.com/help/images/ref/graythresh.html> (2019.12.3)



The background is white and decorated with various colorful, stylized illustrations of fruits and leaves. In the top left, there are small blue bubbles. Below them is a yellow lemon. To the right of the lemon is a green leaf. In the top center, there is an orange slice. To its right is a green kiwi. In the top right, there is a slice of watermelon. Below the watermelon is a green pear. In the middle right, there is a yellow lemon slice. Below that is a small green leaf. In the bottom right, there is a red cherry. In the bottom center, there is a green apple. To its left is a yellow banana. In the bottom left, there is a red strawberry. The central text "감사합니다!" is written in a bold, orange, sans-serif font.

감사합니다!