과제#2 Term Project

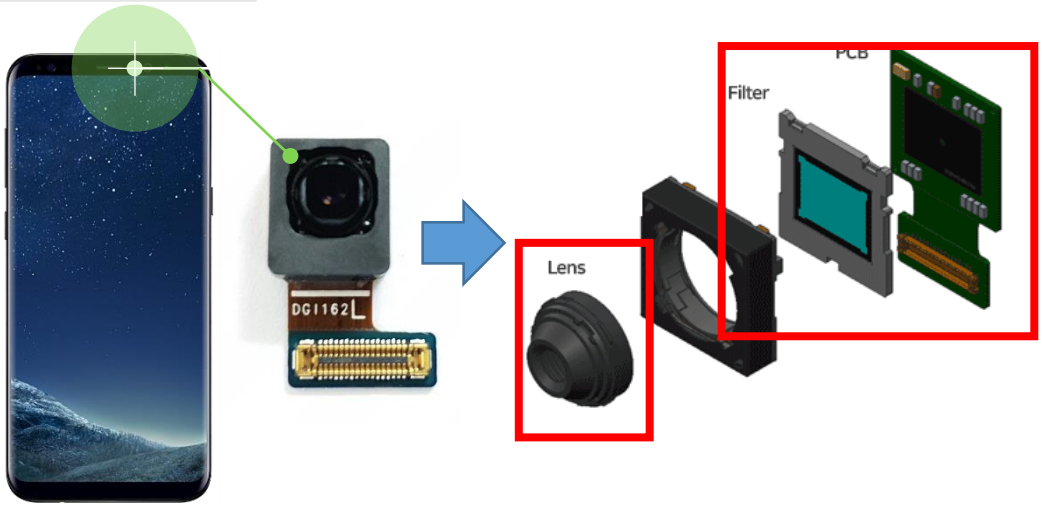
카메라 모듈 – 이물질 검사를 위한 영상처리

2020254013
김병근

2021.10.20

1. Open CV처리용 영상 수집

자사 생산 제품

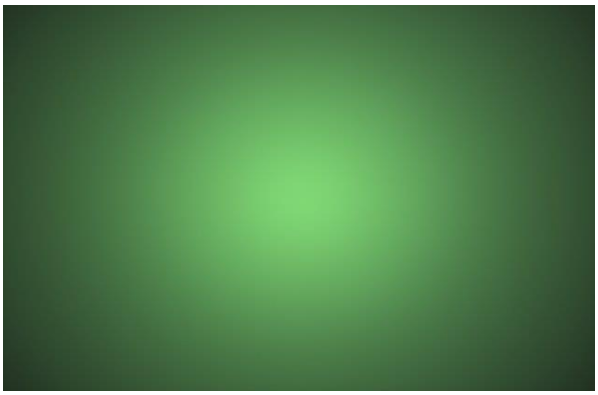


이물질 검증용 영상

영상을 실제 폰에 삽입 후, 육안으로는 확인 中

[전수 검사] Lens, Filter, Sensor(PCB)의 이물질 확인

OK



NG(Stain,멍) - 확대

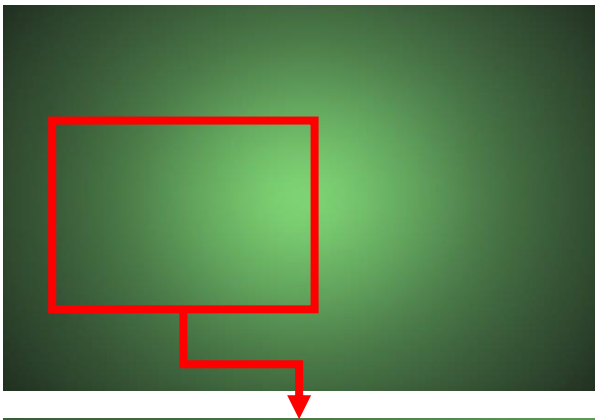


IMAGE 1장
원본(183mb)
→ 전처리 (100kb이하)

2. Term project 진행 개요

진행 내역

1.촬영영상

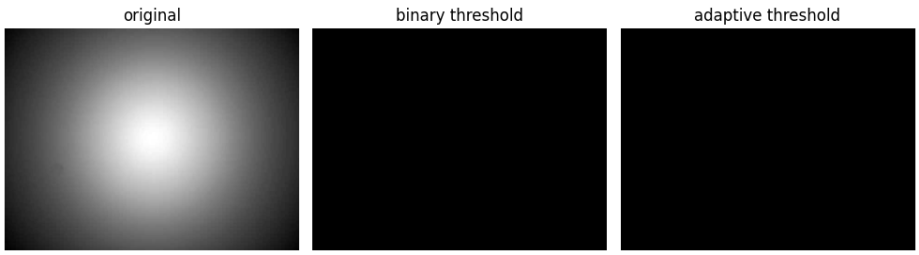
✓ Galaxy 시리즈 카메라 모듈
- 폰검사용 촬영영상 샘플

2.전처리

✓ 183mb/1개(화소마다 다름) → 10 kb로 전처리

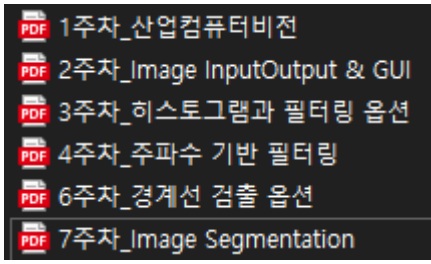
3.분석

✓ 수업시간에 실습한 내용으로
영상 → 이물질 검출 테스트

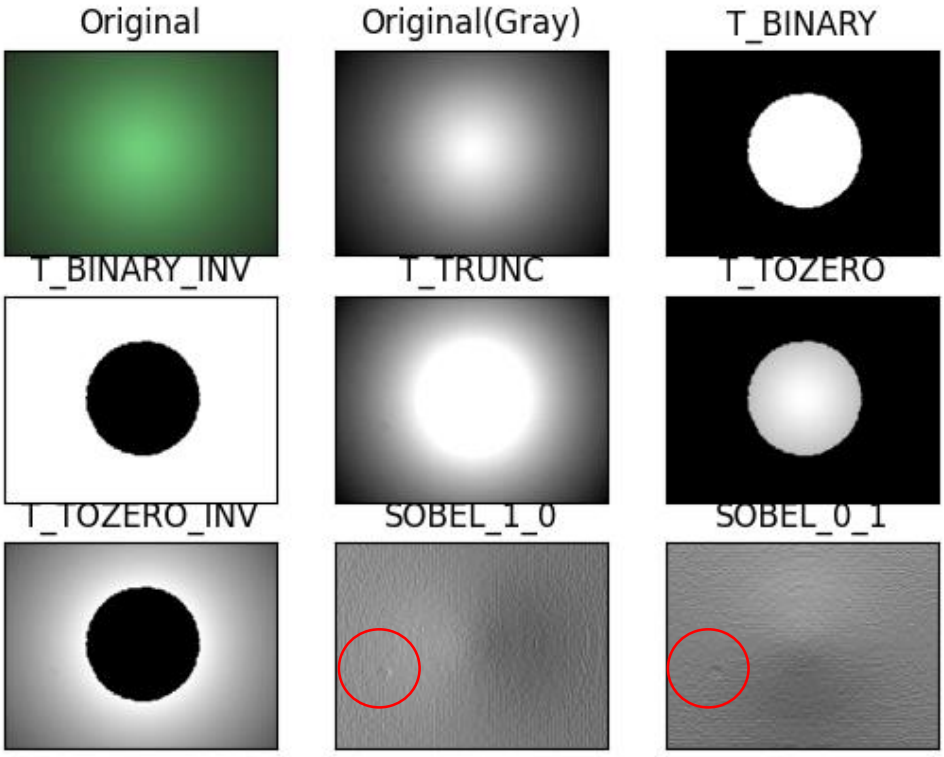


4.방법 선정

✓ 6주차 경계선 검출로 선정
- threshold: 다양한 스타일옵션
- sobel filter: 미분차수 변경



5.영상처리



3. Term project 개발

Python source

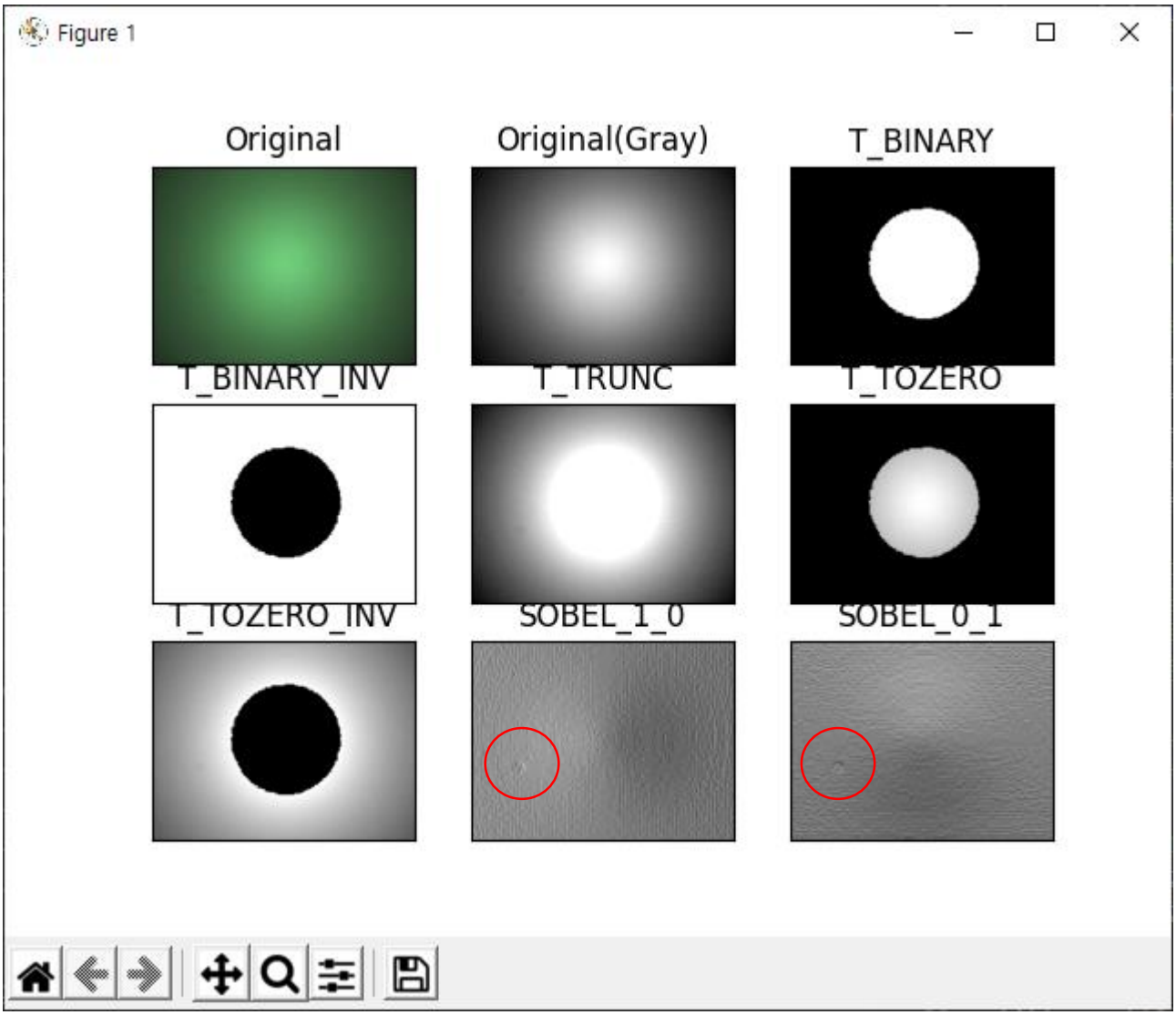
```python
import cv2
import numpy as np
from matplotlib import pyplot as plt

imgOrg = cv2.imread('stain2.jpg')
img = cv2.imread('stain2.jpg',0)

#----- cv2.threshold(src, threshold_value, value, flag)
# src: 입력 영상(grayscale)
# threshold_value: 픽셀 문턱값
# value: 픽셀 threshold값보다 커질때 적용되는 최대값
# flag: threshold 적용 스타일
ret,thresh1 = cv2.threshold(img,127,255,cv2.THRESH_BINARY)
ret,thresh2 = cv2.threshold(img,127,255,cv2.THRESH_BINARY_INV)
ret,thresh3 = cv2.threshold(img,127,255,cv2.THRESH_TRUNC)
ret,thresh4 = cv2.threshold(img,127,255,cv2.THRESH_TOZERO)
ret,thresh5 = cv2.threshold(img,127,255,cv2.THRESH_TOZERO_INV)

# ----- cv2.Sobel(src, ddepth, dx, dy, dst, ksize, scale, delta, borderType) -----
# src: 입력 영상
# ddepth: 출력 영상의 dtype (-1: 입력 영상과 동일)
# dx, dy: 미분 차수 (0, 1, 2 중 선택, 둘 다 0일 수는 없음)
# ksize: 커널의 크기 (1, 3, 5, 7 중 선택)
# scale: 미분에 사용할 계수
# delta: 연산 결과에 가산할 값

# dx, dy: 0/1/2 중 실습 시 배운 미분차수가 가장 좋음
thresh6 = cv2.Sobel(img, cv2.CV_32F, 1, 0)  #dx/dy(미분차수) = 1 / 0
thresh7 = cv2.Sobel(img, cv2.CV_32F, 0, 1)  #dx/dy(미분차수) = 0 / 1


titles = ['Original', 'Original(Gray)','T_BINARY','T_BINARY_INV','T_TRUNC',
          'T_TOZERO','T_TOZERO_INV', 'SOBEL_1_0', 'SOBEL_0_1']
images = [imgOrg, img, thresh1, thresh2, thresh3, thresh4, thresh5, thresh6, thresh7]
for i in range(9):
    plt.subplot(3,3,i+1)
    plt.imshow(images[i],'gray')
    plt.title(titles[i])
    plt.xticks([])
    plt.yticks([])
plt.show()
```

3. Term project 개발

결과 영상

✓ **실습으로 배운 다양한 Filter와 경계선검출의 영상처리를 해본 결과**
자사 CM(Camera module)영상은 적합하지 않았다.
좀 더 배우고, OpenCV lib를 통해 다양한 처리기법을 사용하면
될거 같습니다

✓ **Threshold는 이물질의 경계선이 아닌 밝기에 대한 경계선을 검출 했음**
✓ **Sobel filter는 Original 영상으로 보는 거 보다. 쉽게 식별 가능 함**



4. Term project 개발(10/20 검수 후)

가우시안 4회 및 모폴로지 필터 적용

Morphological_filter.py U × 김병근_산업 컴퓨터비전 실제_2차 Term project(모폴

.vscode > Morphological_filter.py

```python
import cv2
import numpy as np

#Dilation(팽창)
def imageDilation(image, kernel, iterations = 1):
    return cv2.dilate(image, kernel=kernel, iterations=iterations)

#Erosion(침식)
def imageErosion(image, kernel, iterations = 1):
    return cv2.erode(image, kernel=kernel, iterations=iterations)

#Opening
def imageOpening(image, iterations = 1):
    kernel = np.ones((3, 3), np.uint8)
    erosion = imageErosion(image, kernel, iterations)
    return imageDilation(erosion, kernel, iterations)

#Closing
def imageClosing(image, iterations = 1):
    kernel = np.ones((3, 3), np.uint8)
    dilation = imageDilation(image, kernel, iterations)
    return imageErosion(dilation, kernel, iterations)
```

```python
import cv2
import numpy as np
from matplotlib import pyplot as plt
import Morphological_filter as mo

imgOrg = cv2.imread('stain2.jpg')
img = cv2.imread('stain2.jpg',0)

for sigma in range(1, 4):
    img = cv2.GaussianBlur(img, (0, 0), sigma)

choiseNo = np.int(input("횟수입력: "))
kernel = cv2.getStructuringElement(cv2.MORPH_RECT, (3,3))

imgCopyD = img.copy()
imgCopyE = img.copy()
imgCopyO = img.copy()
imgCopyC = img.copy()

#D(dilation), E(erosion), O(open), C(close)
for i in range(choiseNo):
    imgCopyD = mo.imageDilation(imgCopyD, kernel)
    imgCopyE = mo.imageErosion(imgCopyE, kernel)
    imgCopyO = mo.imageOpening(imgCopyO)
    imgCopyC = mo.imageClosing(imgCopyC)

# ---- cv2.Sobel(src, ddepth, dx, dy, dst, ksize, scale, delta, borderType) -----
# src: 입력 영상,                # ddepth: 출력 영상의 dtype (-1: 입력 영상과 동일)
# dx, dy: 미분 차수 (0, 1, 2 중 선택, 둘 다 0일 수는 없음)
# ksize: 커널의 크기 (1, 3, 5, 7 중 선택)   # scale: 미분에 사용할 계수
# delta: 연산 결과에 가산할 값
# dx, dy: 0/1/2 중 실습 시 배운 미분차수가 가장 좋음
sobelD1 = cv2.Sobel(imgCopyD, cv2.CV_32F, 1, 0)  #dx/dy(미분차수) = 1 / 0
sobelD2 = cv2.Sobel(imgCopyD, cv2.CV_32F, 0, 1)  #dx/dy(미분차수) = 0 / 1
sobelE1 = cv2.Sobel(imgCopyE, cv2.CV_32F, 1, 0)
sobelE2 = cv2.Sobel(imgCopyE, cv2.CV_32F, 0, 1)
sobelO1 = cv2.Sobel(imgCopyO, cv2.CV_32F, 1, 0)
sobelO2 = cv2.Sobel(imgCopyO, cv2.CV_32F, 0, 1)
sobelC1 = cv2.Sobel(imgCopyC, cv2.CV_32F, 1, 0)
sobelC2 = cv2.Sobel(imgCopyC, cv2.CV_32F, 0, 1)
sobelT1 = cv2.Sobel(imgCopyC, -1, 0, 1, kernel)
sobelT2 = cv2.Sobel(imgCopyC, -1, 1, 0, kernel)
```

5 출력 터미널 디버그 콘솔

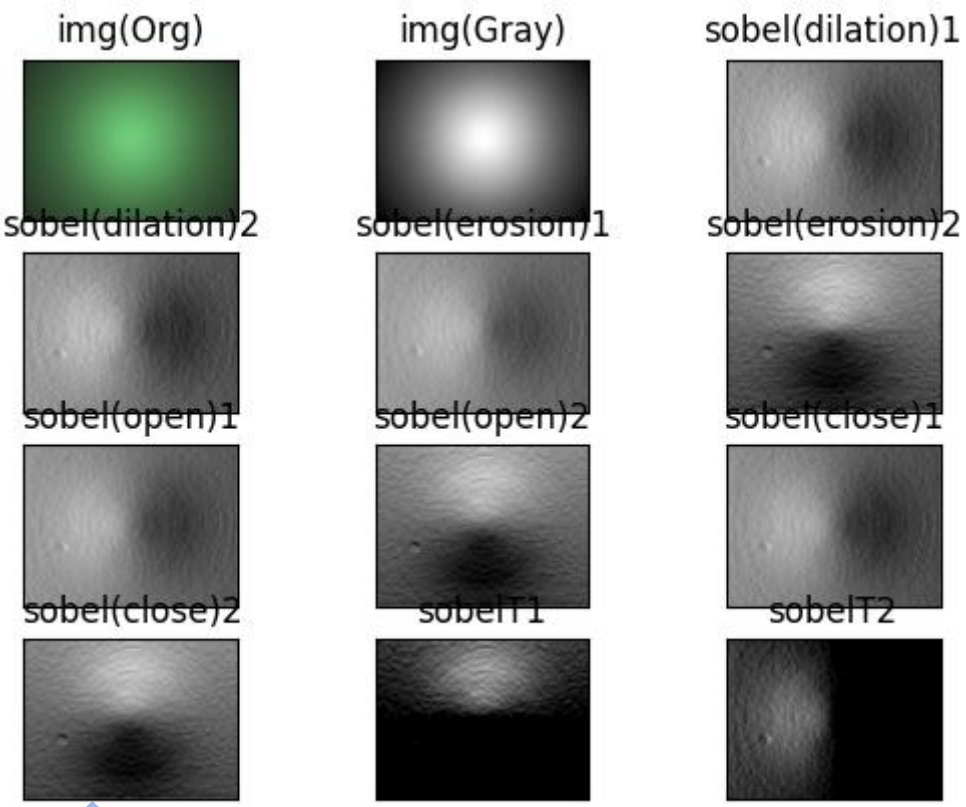
4. Term project 개발(10/20 검수 후)

결과 영상

1. 가우시안 필터 4회
2. 모폴로지 필터 5회를 모든 case에 적용한 후
   D(dilation), E(erosion), O(open), C(close)
3. Sobel filter로 추가 적용 함

※ 결론: 바로 Sobel로 처리하는 것 보다,
1번~2번을 거친 후, Sobel 처리 시 …. 확연히 “얼룩(Stain)”이 보임

추가 작업



기존