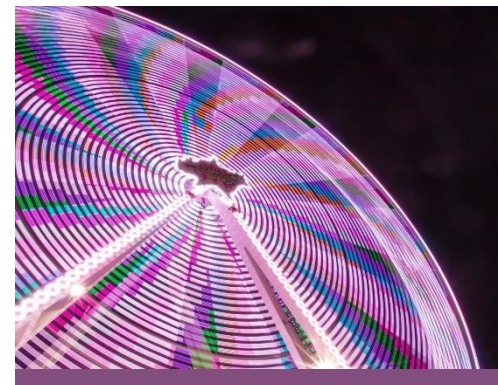
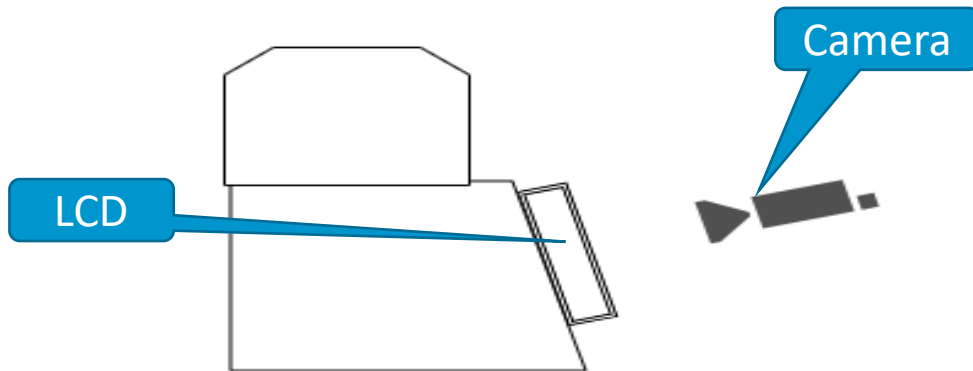
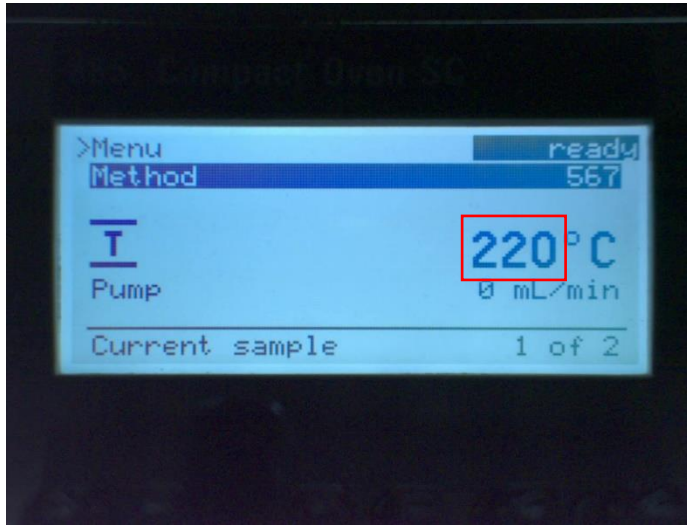


# 산업 컴퓨터 비전 실제

기말 텀프로젝트 - LCD 숫자 검출 및 인식



# 개요



## 기말 텀프로젝트

- 중간 프로젝트 때 정한 문제 혹은 새롭게 정한 현업에서의 컴퓨터 비전 문제에 대해서 최종 프로젝트를 진행함
- 14주차 강의까지 배운 내용들을 활용해서 특징 추출, 대응점 탐색, 옵티컬 플로우, 파노라마 스티칭, 카메라 캘리브레이션, 스테레오 매칭 등을 적용
- 12월 15일 강의시간에 각자 5page 내외로 발표
- 발표자료와 소스코드를 e-campus를 통해 제출 (제출 기한 12/15 11:59PM)

## 주제

- LCD에 표시된 온도 인식
  - ✓ LCD 영역 추출
  - ✓ 개별의 숫자 분리
  - ✓ 숫자 인식(knn 이용)

# 영상 처리 순서

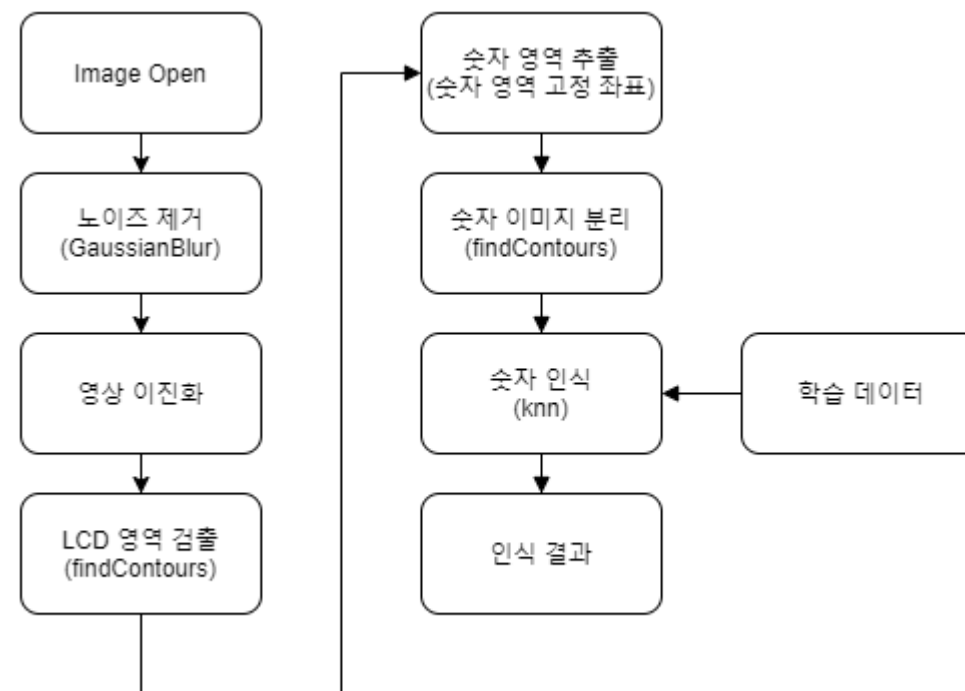
원본 영상



## 학습 영상

[illegible]

## 영상 처리 순서

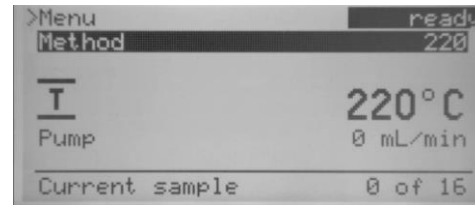


# 영상 처리 순서에 따른 이미지

원본 영상



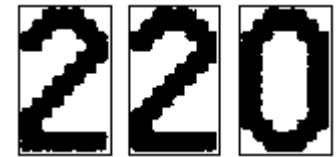
LCD 영역 추출



숫자 영역 추출



숫자 분리



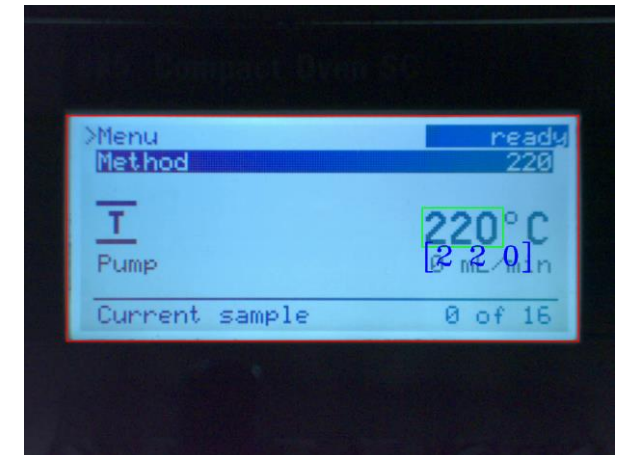
LCD 영역 추출 code

```
# LCD 영역 검출
return_value, threshold_image = cv2.threshold(img_gray, 0, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)
im2, contours, hierarchy = cv2.findContours(threshold_image, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

cnt = contours[0]

contours_min = np.argmin(contours[0], axis=-1)
contours_max = np.argmax(contours[0], axis=-1)
x_min = contours[0][contours_min[0]][0][0]
y_min = contours[0][contours_min[1]][0][1]
x_max = contours[0][contours_max[0]][0][0]
y_max = contours[0][contours_max[1]][0][1]
```

숫자 인식 및 결과




# knn train

Train 영상  
(train\_numbers.png)

0  
1  
2  
3  
4  
5  
6  
7  
8  
9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9

개별 영상 사이즈

40  40

code

```
def kNN_train(train_fname, nclass, nsample):  
    size = (40, 40) # 숫자 영상 크기  
    train_img = cv2.imread(train_fname, cv2.IMREAD_GRAYSCALE) # 학습 영상 적재  
    h, w = train_img.shape[:2]  
    dy = h % size[1] // 2  
    dx = w % size[0] // 2  
    train_img = train_img[dy:h-dy-1, dx:w-dx-1] # 학습 영상 여백 제거  
    cv2.threshold(train_img, 32, 255, cv2.THRESH_BINARY, train_img)  
  
    cells = [np.hsplit(row, nsample) for row in np.vsplit(train_img, nclass)]  
    nums = [find_number(c) for c in np.reshape(cells, (-1, 40, 40))]  
    trainData = np.array([place_middle(n, size) for n in nums])  
    labels = np.array([i for i in range(nclass) for j in range(nsample)], np.float32)  
  
    knn = cv2.ml.KNearest_create()  
    knn.train(trainData, cv2.ml.ROW_SAMPLE, labels) # k-NN 학습 수행  
    return knn
```

# 숫자 영역 추출 및 인식

## 숫자 영역 추출

```
# 숫자 영역 추출
digit_img = th_lcd_image[y_digit_min:y_digit_max, x_digit_min: x_digit_max]

#숫자 분리
fcimg, contours, hierarchy = cv2.findContours(digit_img.copy(), cv2.RETR_LIST, cv2.CHAIN_APPROX_SIMPLE)

digitCnts = []
roi_imgs = []
contours.reverse()
i = 0
for cnt in contours:
    area = cv2.contourArea(cnt)
    [x, y, w, h] = cv2.boundingRect(cnt)

    print(x, y, w, h)
    # if the contour is sufficiently large, it must be a digit
    if w >= 15 and (h >= 30 and h <= 80) and x > 0 and y > 0 and i < 4:
        digitCnts.append(cnt)
        each_digit_img = digit_img[y:y+h, x: x+w]
        roi_imgs.append(each_digit_img)
        cv2.rectangle(digit_img, (x, y), (x + w, y + h), (0, 0, 255))
    i = i+1
```

## 숫자 인식

```
#숫자 인식
numbers = [find_number(cell) for cell in roi_imgs]
datas = [place_middle(num, (40, 40)) for num in numbers]
datas = np.reshape(datas, (len(datas), -1))

_, resp1, _, _ = nknn.findNearest(datas, K1)
resp1 = resp1.flatten().astype('int')

print(resp1)
```



# 문제점 및 추가 과제

## 숫자 영역 추출

- 현재 - 절대 좌표 입력
- 향후 - 자동 추출 할 수 있는 방법 고려

## 인식 알고리즘 개선

- 현재 - knn 적용
- 많은 데이터 추출 필요
- 데이터 추출 후 다른 알고리즘 적용 및 인식률 비교/개선 필요

## 물리적 환경 변화에 대응 가능한 전처리 방법 개선

- 다양한 영상 Skew에 대한 테스트 필요
- Skew 보정 알고리즘 적용 필요