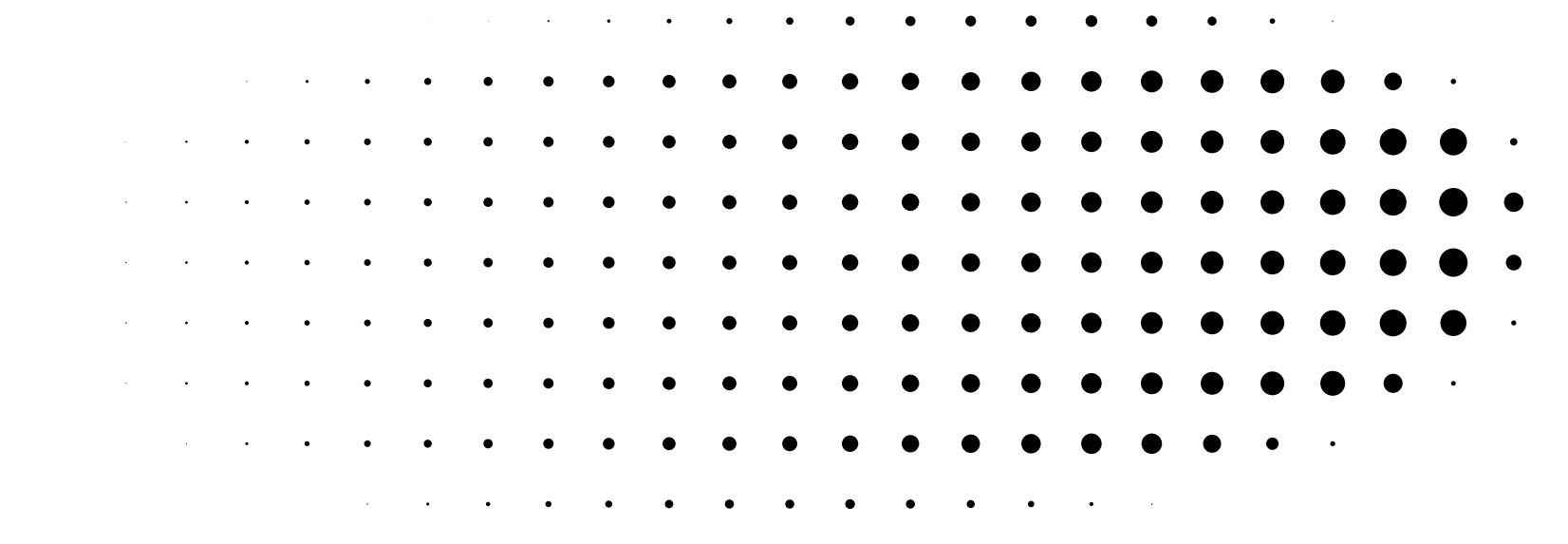


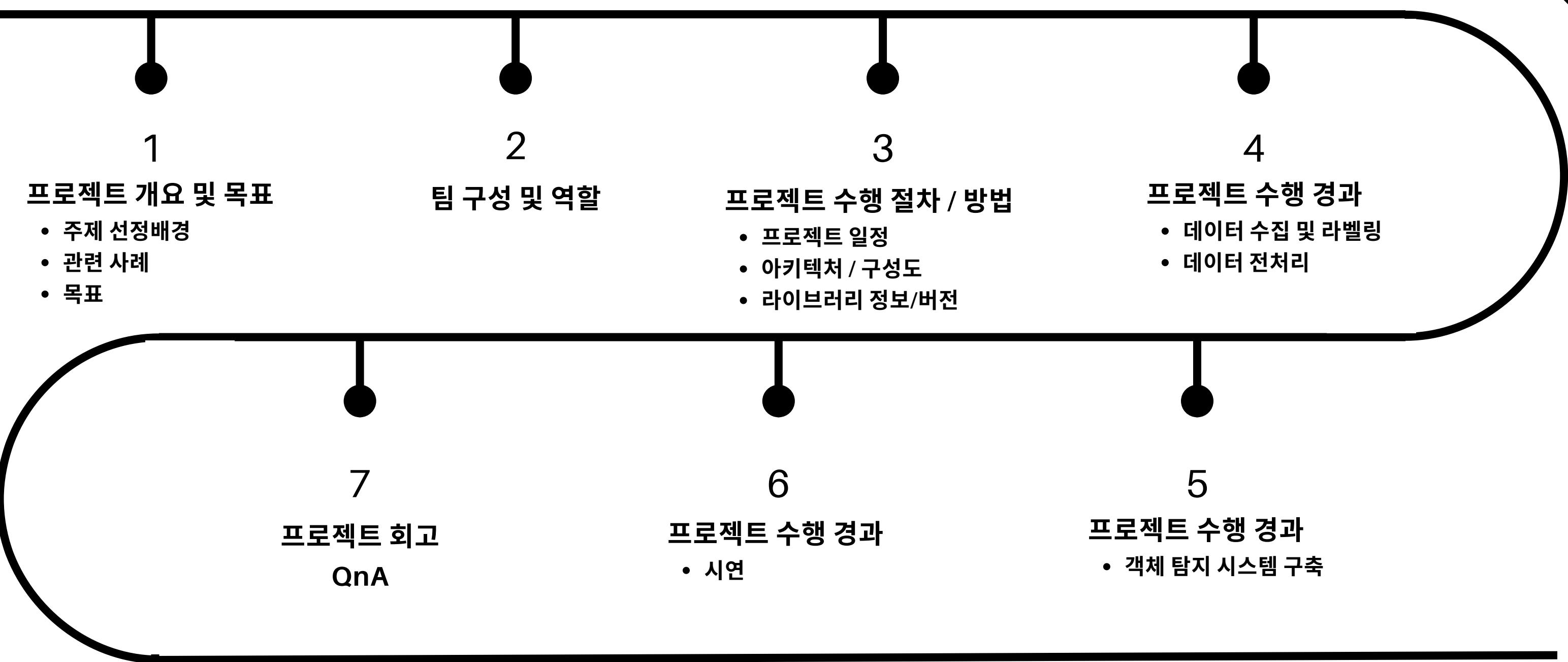
**TEAM. UNITY**

# **PROJECT II**

**Computer Vision**



# 프로젝트 목차



# 주제 선정 배경

## 패키징의 중요성

모든 제조 공정에서  
패키징 작업은 필수 요소  
**\*패키징 : 제품 포장**

## 브랜드 이미지 손상

패키징 손상으로 인한  
**소비자 불만족** 발생 시  
제조회사 **브랜드의 이미지가**  
**손상되는 문제** 발생  
(구매율 및 제품 신뢰도 하락)

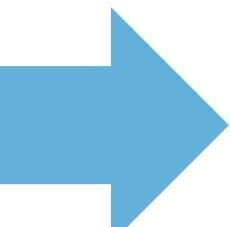
## 생산 효율성 저하

손상된 패키지로 인한  
재 생산 및 폐기 비용 발생  
**(비용적인 부담)**

## 객체 탐지 모델 활용

패키징 과정에서 불량을  
사전에 인식하고 자동으로  
분류하는 모델을 구축할  
필요성 대두

“BECAUSE”



# 관련 사례

## [세이브타임즈] 오뚜기 '포장 불량' 감작감작 납작면 전량 회수

해당 제품은 포장지 접착불량이 발견돼 전량 회수되고 있는 것으로 알려졌다.  
접착 문제로 상품에 문제가 생길 가능성이 있어서다.

<https://www.safetimes.co.kr/news/articleView.html?idxno=211945>

## [SBS Biz] 식약처, 종근당 감기약 '모드콜' 회수 명령.."포장 불량"

어린이 감기약에 하얀 이물질이 묻어 자발적 회수에 나섰던 종근당에 대해,  
식품의약품당국이 해당 제품에 대한 공식적인 회수 명령을 내렸습니다.

<https://biz.sbs.co.kr/article/20000140229>

## [데일리팜] 포장 불량 의약품 회수 빈번...내용고형제 자율점검

최근 의약품 포장·표시 불량으로 회수하는 사례가 늘어나면서  
식품의약품안전처가 제조업체들에게 자율점검을 예고했다.

<https://www.dailypharm.com/Users/News/NewsView.html?ID=315674>

**“다수의 관련 사례 존재”**

# 목 표

이미지의 다양한 라벨링  
방법을 통해  
이상치 검출

다양한 이미지 전처리  
기법 반영,  
모델별 성능 지표 비교 분석

패키징 결함, 손상  
문제 발생

YOLO 탐지 모델 사용

실시간 결함 감지 서비스 제공

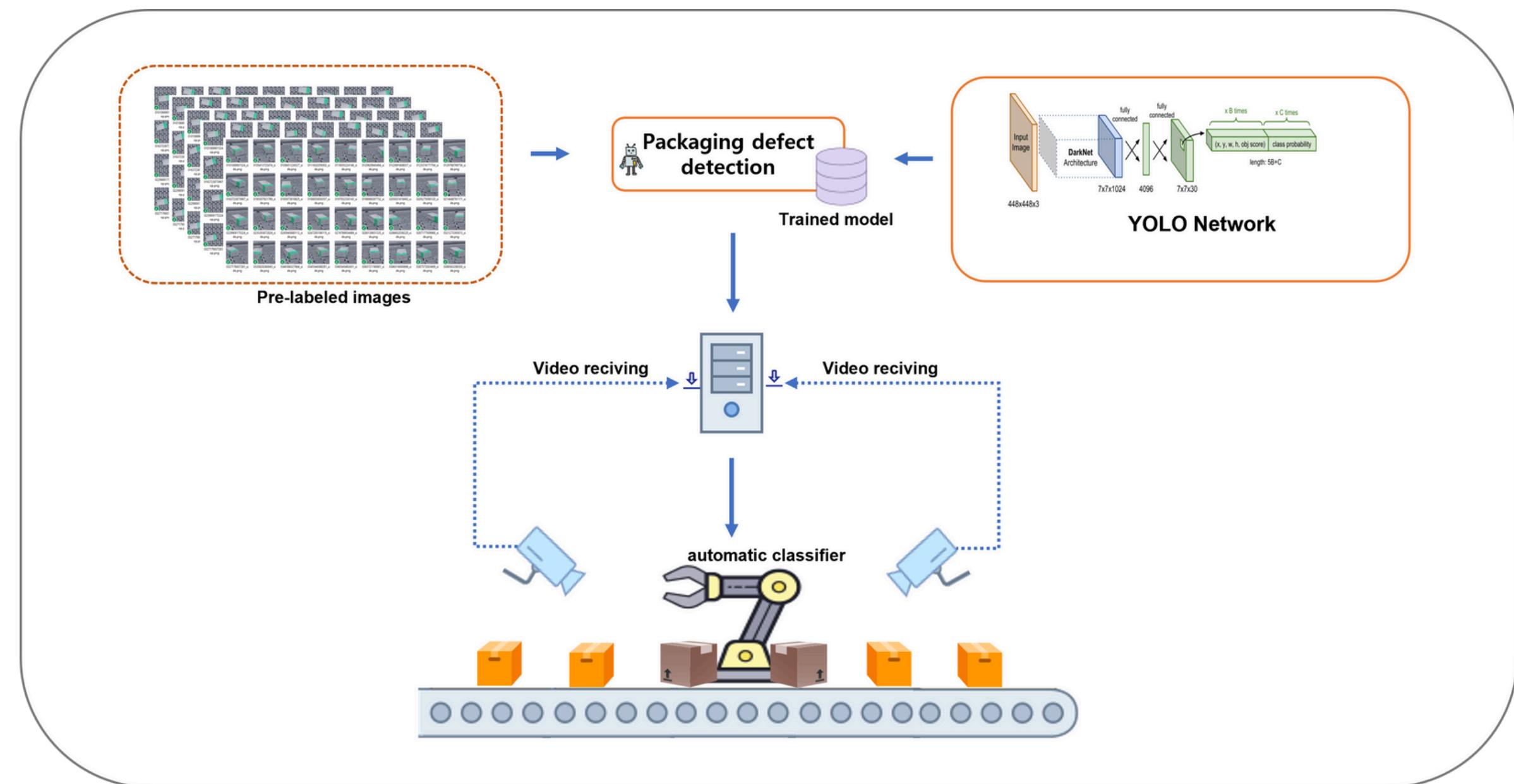
## 프로젝트 기대효과

- 객체 탐지 서비스 제공으로
- 제조공정의 품질 보증 강화
  - 생산 효율성 향상(불량율 감소)
  - 비용 절감 및 소비자 신뢰 유지

# 목표

## 기본 개념

실제 스마트팩토리에서 적용되는 실시간 이상 탐지 로직



# TEAM. UNITY



유현재

팀장



유현욱

팀원



이태훈

팀원

## <역 할>

## <업무>

### [총괄]

- 프로젝트 관리 및 조율(일정/목표)
- 전반적인 전략 수립(팀원 역할 분배)
- 평가 및 분석 지원(결과 방향 조정)
- 팀 커뮤니케이션(소통 및 피드백)
- 제공 서비스 ui 설계 및 구현

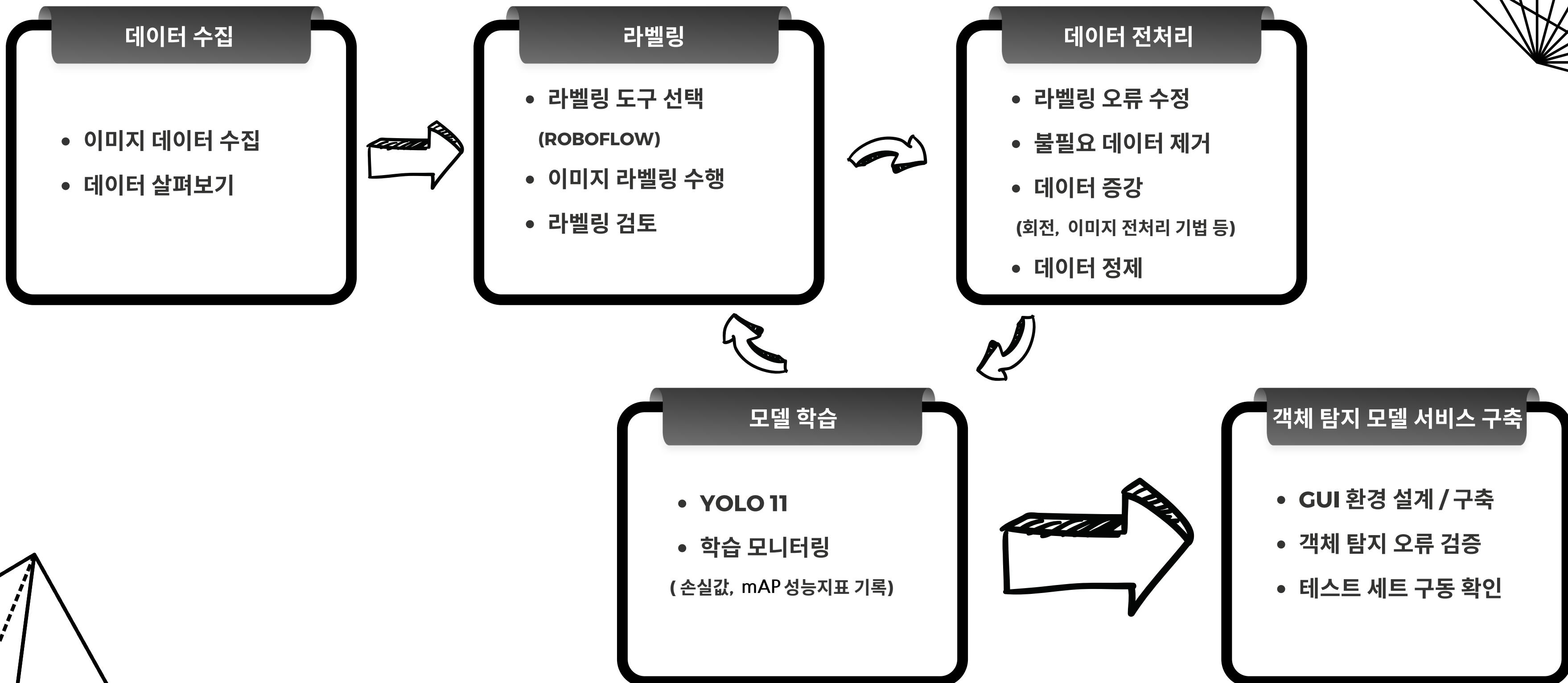
### [모델 개발자]

- 모델 학습 및 훈련
- 데이터 전처리 기법별 성능비교
- 모델 성능 모니터링
- 최신 연구 동향 반영(YOLO버전)

### [데이터 엔지니어]

- 데이터 수집 및 정제
- 데이터 증강 및 전처리
- 데이터셋 분할(학습, 검증, 테스트)
- YOLO형식에 맞는 데이터 셋 통합
- 모델 성능 지표 측정  
(precision, recall, mAP)

# 프로젝트 수행절차 / 방법



# 프로젝트 일정

구 분	기 간	활 동		비 고
사전 기획	2024. 09. 03(화) ~ 04(수)	프로젝트 기획 및 주제 선정	기획안 작성	아이디어 선정
데이터 선정 /수집	2024. 09. 04(수) ~ 11(수)	데이터 수집	데이터 라벨링	GITHUB 데이터 탐색/선정
데이터 전처리	2024. 09. 11(수) ~ 10.02(수)	데이터 정제(라벨링 오류)	데이터 증강 기법	
모델링	2024. 09. 25(수) ~ 10.09(수)	실시간 객체 탐지 시스템 구축	오류 검증	중간발표
총 개발기간	2024. 09. 03(화) ~ 10.09(수)			
최 종	2024.10.14(월)			최종발표

# 라이브러리 정보/버전

## 개발환경

### Laptop

프로세서: Intel(R) Core(TM) i7-8565U CPU @ 1.80GHz 1.99 GHz

시스템 종류: 64비트 운영 체제, x64 기반 프로세서

## Model

YOLO v11

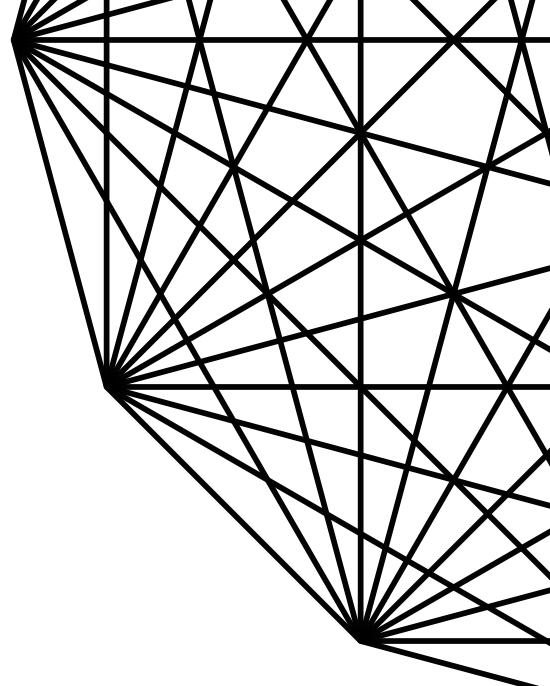
## 이미지 프로세싱

CLAHE  
Edge Detection  
Embossing

## Library

python	v.3.12.7
numPy	v.2.1.1
seaborn	v.0.13.1
pandas	v.2.2.3
env	v.0.1.0
ultralytics	v.8.3.4

torch	v.2.4.1
torchvision	v.0.19.1
opencv-python	v.4.10.0.84
matplotlib	v.3.9.2
scipy	v.1.13.1
PyYAML	v.6.0.2
tkinter	v.8.6



# YOLO - 주요 개념

- **You Only Look Once**

전체 이미지를 단 한 번만 본다.

YOLO 이전에 존재한 객체 탐지 모델들과 비교했을 때, 예를 들어 R-CNN은 한 장의 이미지를 여러 장으로 쪼개서 CNN 모델을 통과시키기 때문에 한 장의 이미지에서 객체 탐지 를 수행해도 실제로는 수 천장의 이미지를 모델에 통과시키지만 YOLO는 이미지 전체를 말 그대로 한 번만 보고 처리한다.

---

- **단순한 구조**

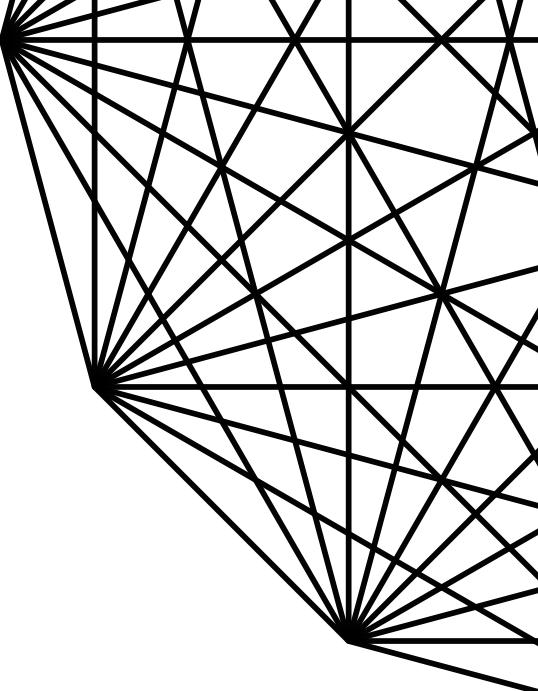
다른 객체 탐지 모델들은 다양한 전처리 모델과 인공 신경망 을 결합해 사용하지만 YOLO는 통합된 하나의 인공신경망에 서 전부 처리하기 때문에 비교적 간단한 구조를 갖는다.

- **Real-time Object Detection**

실시간 객체 탐지

"빠르다"라는 이름이 붙은 **Fast R-CNN**이 **0.5FPS**의 성능을 내는 반면에 **YOLO**는 기본적으로 **45FPS**의 성능을 보인다. 스트리밍을 하면서 동시에 화면 상의 객체를 부드럽게 탐지할 수 있는 특징이 YOLO를 유명하게 만든 계기가 됐다.

---



# YOLO - 네트워크 구조

- **backbone**

: 이미지의 중요한 특징을 추출하는 네트워크의 첫 번째 부분.

p1~p5까지 피라미드 구조로 설계.

각 계층은 다른 크기의 특징맵을 생성해서 다양한 크기의 객체를 포착.

C2f 모듈은 여러 레이어를 연결하고 특징 간의 관계를 고려한다.

Conv는 convolution 연산으로 특징을 추출.

## YOLOv8 모델 아키텍처

**ultralytics**

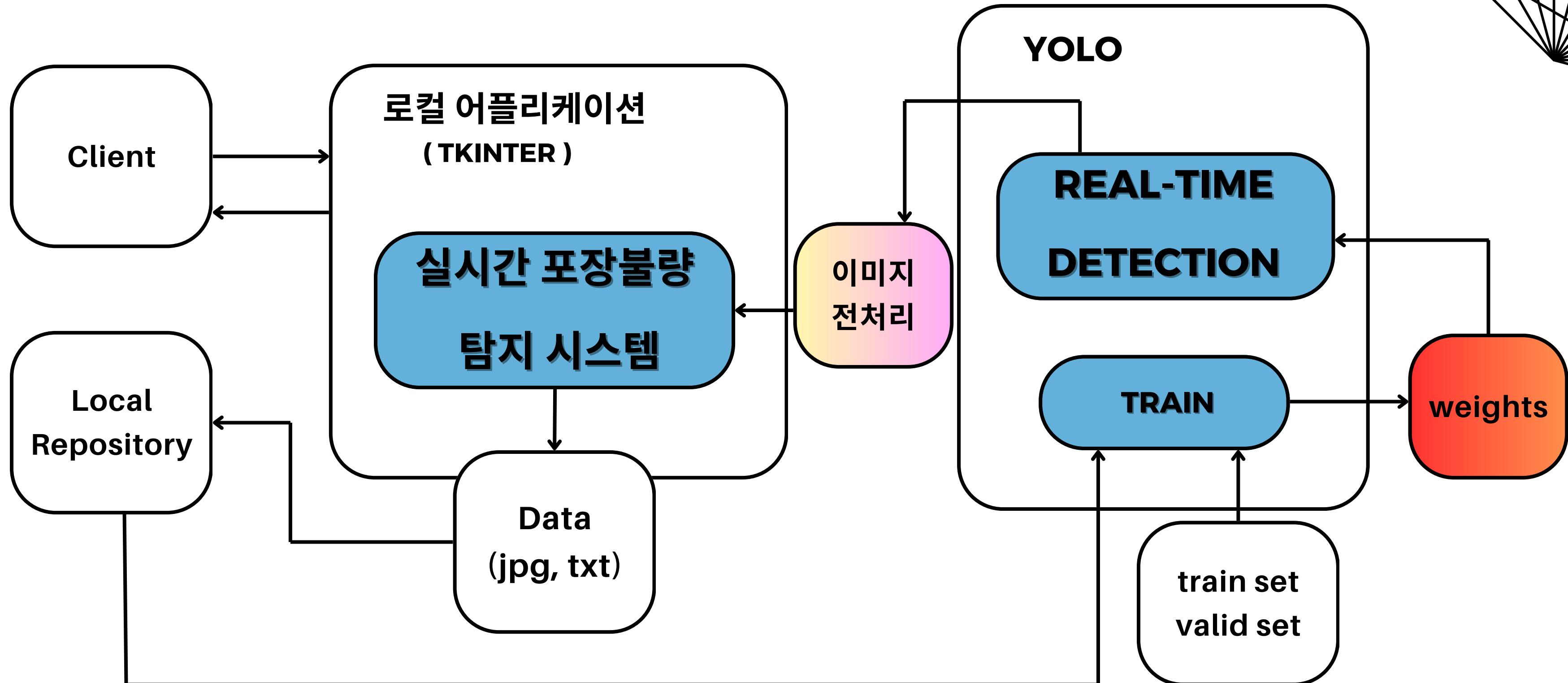
- **neck**

: p3~p가 상호 결합되어 다양한 스케일의 특징을 만들어 내며 이후 탐지에서 활용

- **head**

: Head는 최종적으로 객체를 탐지하는 역할.

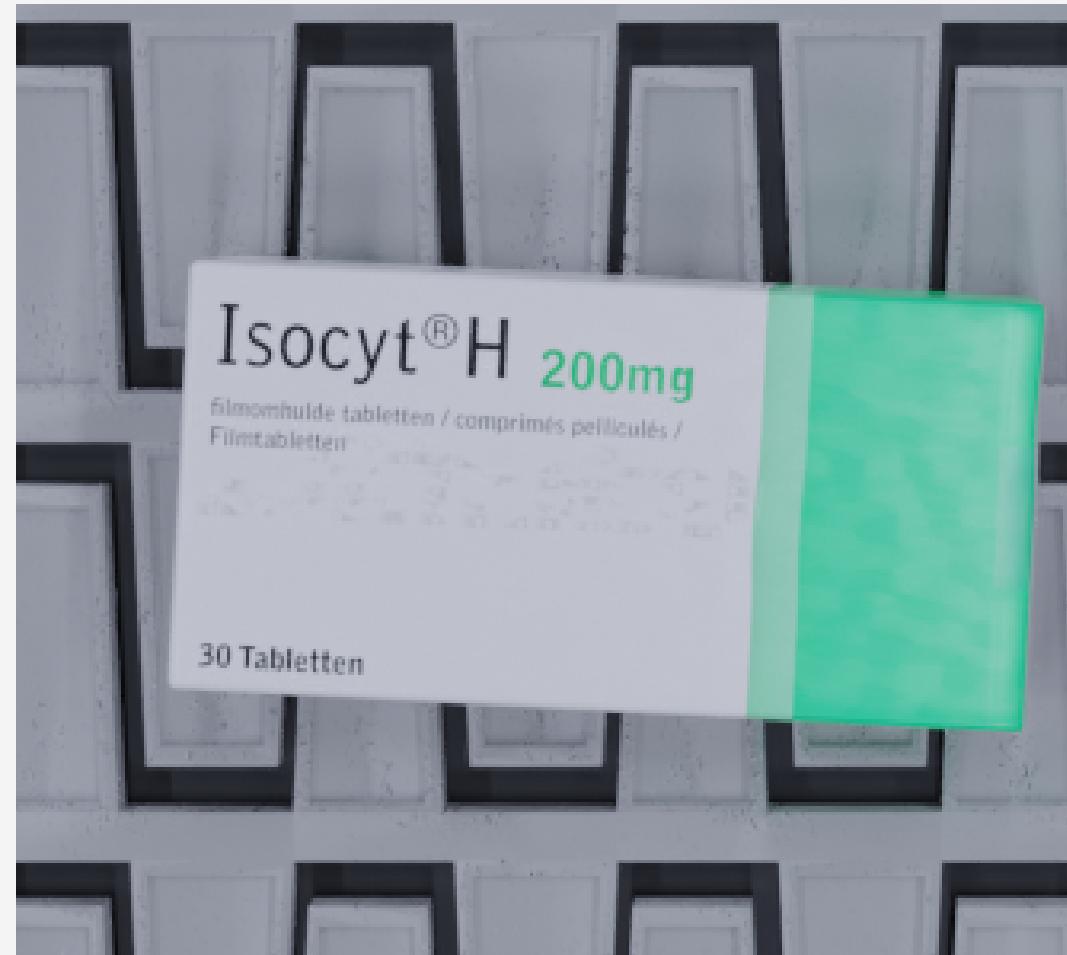
# 구성도 - 서비스 구성도



# 데이터 수집

## 데이터 셋 설명

패키징 과정 이상이 없는 데이터와 그 외 발생되는 손상들을 이미지로 저장한 데이터



<정상 데이터 이미지>



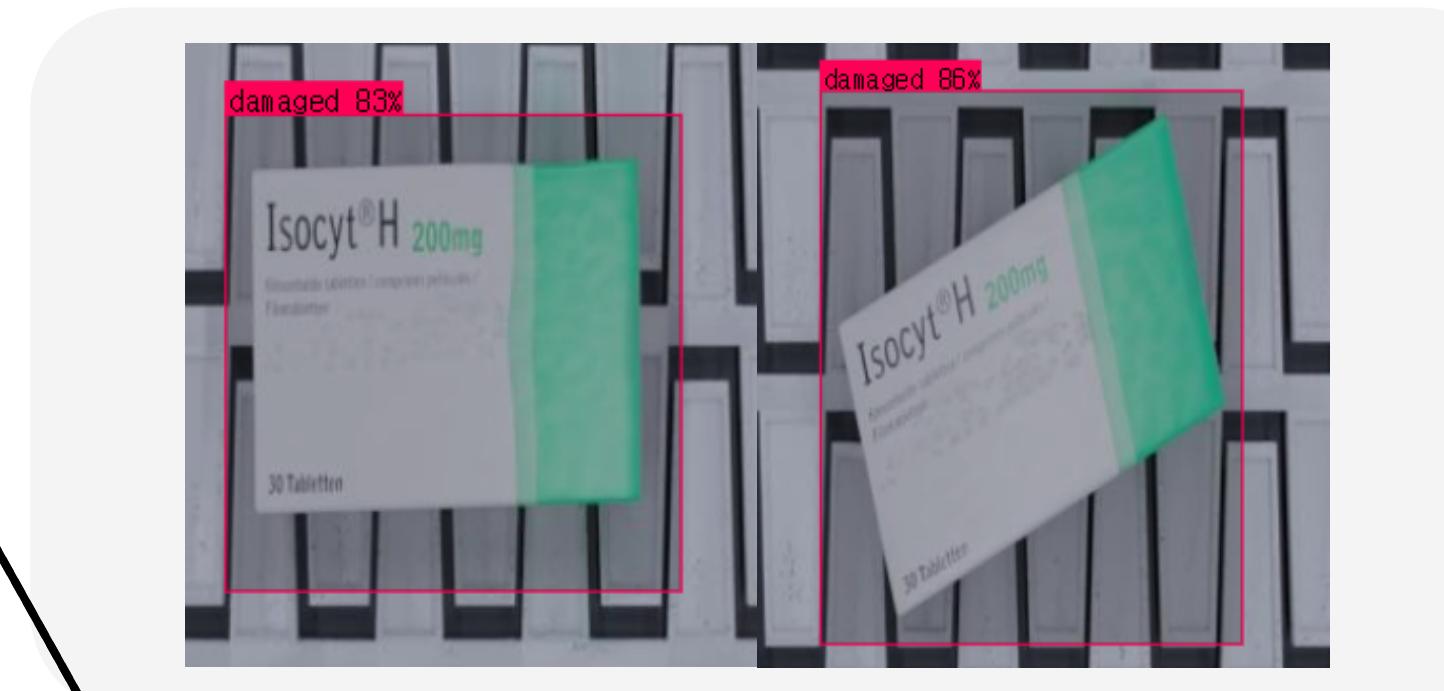
<불량 데이터 이미지 >

# 라벨링

## ✓ 1차 라벨링



## ✓ 수행 결과



Model summary (fused): 186 layers, 2,684,758 parameters, 0 gradients, 6.8 GFLOPs  

Class	Images	Instances	Box(P)	R	mAP50	mAP50-95
all	80	80	0.499	1	0.579	0.483
damaged	40	40	0.499	1	0.628	0.528
intact	40	40	0.499	1	0.53	0.437

  
Speed: 0.3ms preprocess, 4.0ms inference, 0.0ms loss, 3.1ms postprocess per image

### [ 1차 라벨링 ]

도구 : Roboflow 라벨링  
방식 :

- 패키징 박스 전체를 Class로 선정
- 클래스명 : intact(정상), damaged(손상)

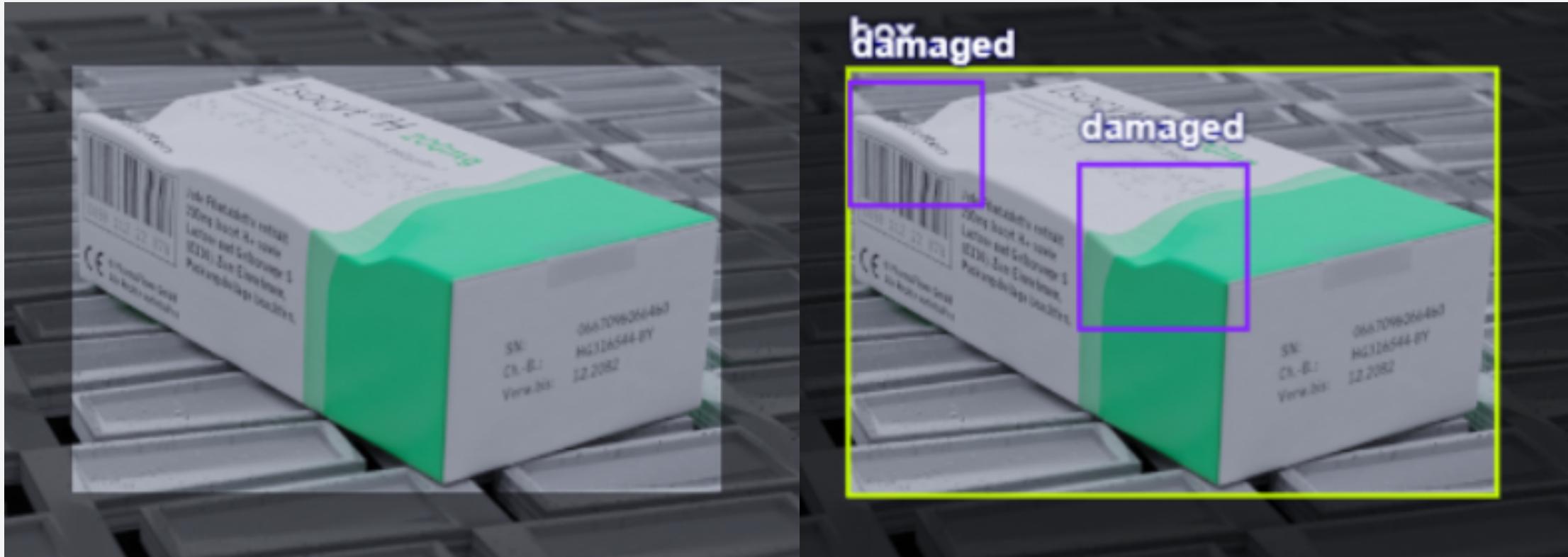
### [ 수행 결과 ]

- 정밀도(P) : 0.499
- 재현율(R) : 1

[ 문제점 ]  
intact(정상)인 이미지 데이터가  
damaged(손상) 이미지로 오인 판별 빈번함

# 라벨링

## ✓ 2차 라벨링



### [ 2차 라벨링 ]

도구 : Roboflow 라벨링

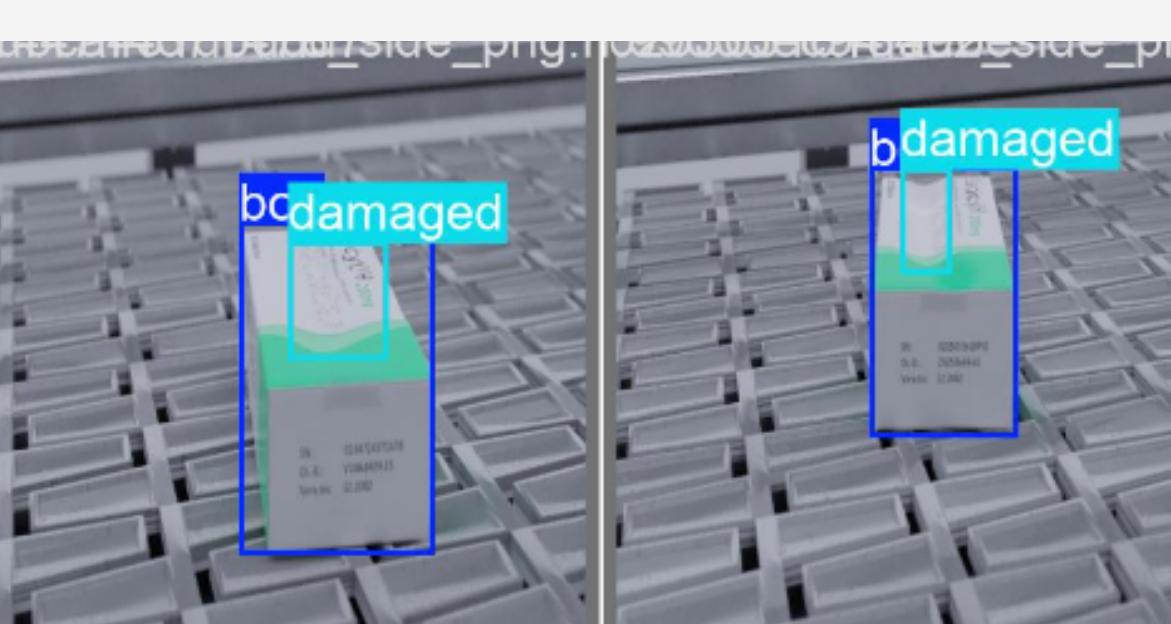
방식 :

- 박스전체 -> 전체 박스 Class, 손상부위 Class 별도 라벨링
- 클래스명 : box(전체 박스) , damaged(손상)

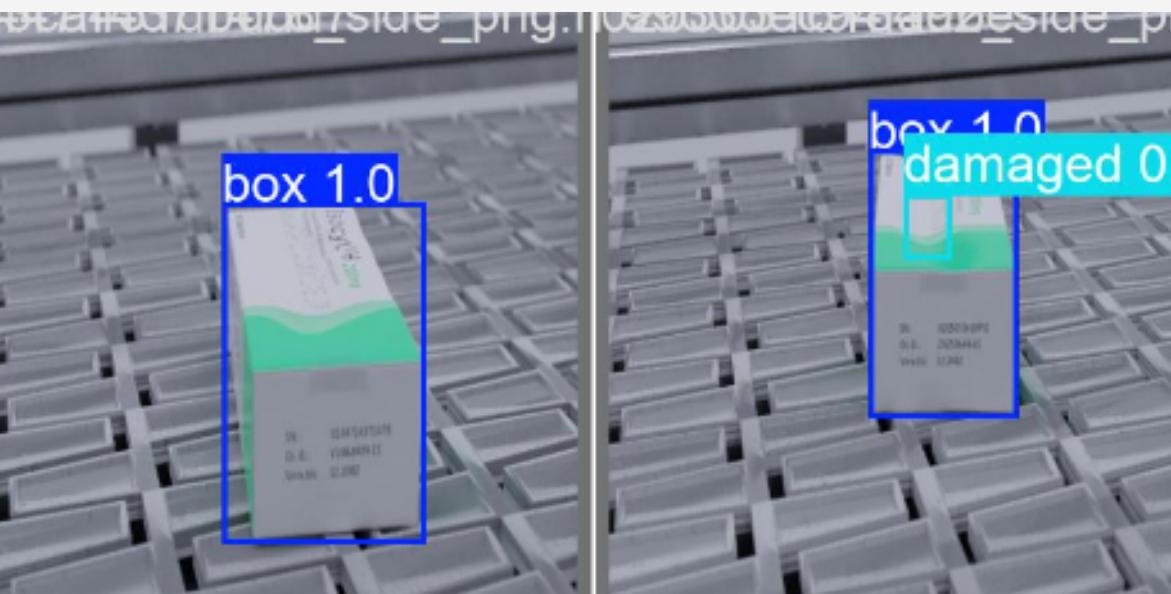
# 라벨링

## ✓ 수행 결과

TRUE



PRED



Class	Images	Instances	Box(P)	R	mAP50	mAP50-95):
all	80	108	0.689	0.696	0.701	0.557
box	80	80	0.99	1	0.995	0.982
damaged	25	28	0.389	0.393	0.408	0.132

### [ 수행 결과 ]

- 1차 라벨링 방식
  - 정밀도(P) : 0.499
  - 재현율(R) : 1

- 2차 라벨링 방식
  - 정밀도(P) : -> 0.689(상승)
  - 재현율(R) : -> 0.696(하락)

### [ 문제점 ]

- 예측 일부 이미지들이 **damaged**가 검출되지 않는 현상 발생

# 전처리

객체의 찌그러짐이나 손상정도를 감지함에 있어서 부각시키는 도구

- **CLAHE( 대비 제한 적응형 히스토그램 평활화)**  
: 이미지의 대비를 개선하는 기법
- **Edge Detection(윤곽선 검출)**  
: 이미지의 경계나 윤곽선을 찾아 강조하는 기법
- **Embossing**  
: 이미지의 경계와 깊이감을 강조하는 기법

# 전처리

## CLAHE(Contrast Limited Adaptive Histogram Equalization)

- 이미지를 작은 타일로 나누어 각 타일마다 개별적으로 히스토그램 평활화 수행
- 특정 대비 한계를 설정하여 과도한 대비 증폭과 노이즈 강조를 방지
- 조명이 나쁘거나 대조가 낮은 영역에서도 세부 정보를 유지하며 대조를 효과적으로 개선

### [ 적용 예시 ]

```
##### CLAHE #####
img_yuv = cv2.cvtColor(img, cv2.COLOR_BGR2YUV)
img_clahe = img_yuv.copy()
clahe = cv2.createCLAHE(clipLimit=3.0, tileGridSize=(8,8))
img_clahe[:, :, 0] = clahe.apply(img_clahe[:, :, 0])
img_clahe = cv2.cvtColor(img_clahe, cv2.COLOR_YUV2BGR)
```

# 전처리

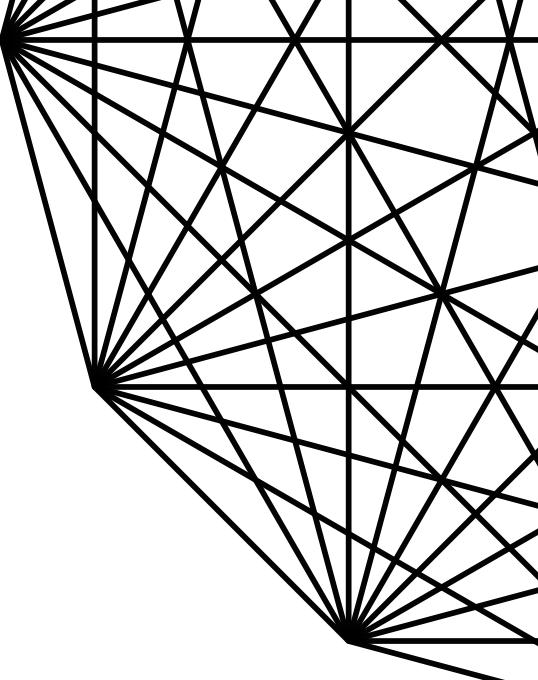
## CLAHE



[ Before ]



[ After ]



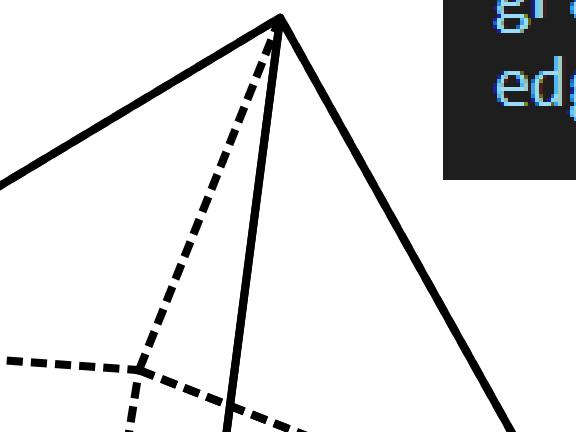
# 전처리

## Edge Detection

- 이미지 내 밝기 변화가 큰 부분인 엣지를 찾아 객체의 경계와 윤곽선을 강조.
- Sobel, Canny 등 다양한 알고리즘을 통해 엣지를 효과적으로 추출.
- 객체 인식, 이미지 분석 및 컴퓨터 비전 응용에 필수적인 기술.

### [ 적용 예시 ]

```
##### EDGES #####
gray_scale = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
edges = cv2.Canny(gray_scale, threshold1=100, threshold2=200, apertureSize=3)
```

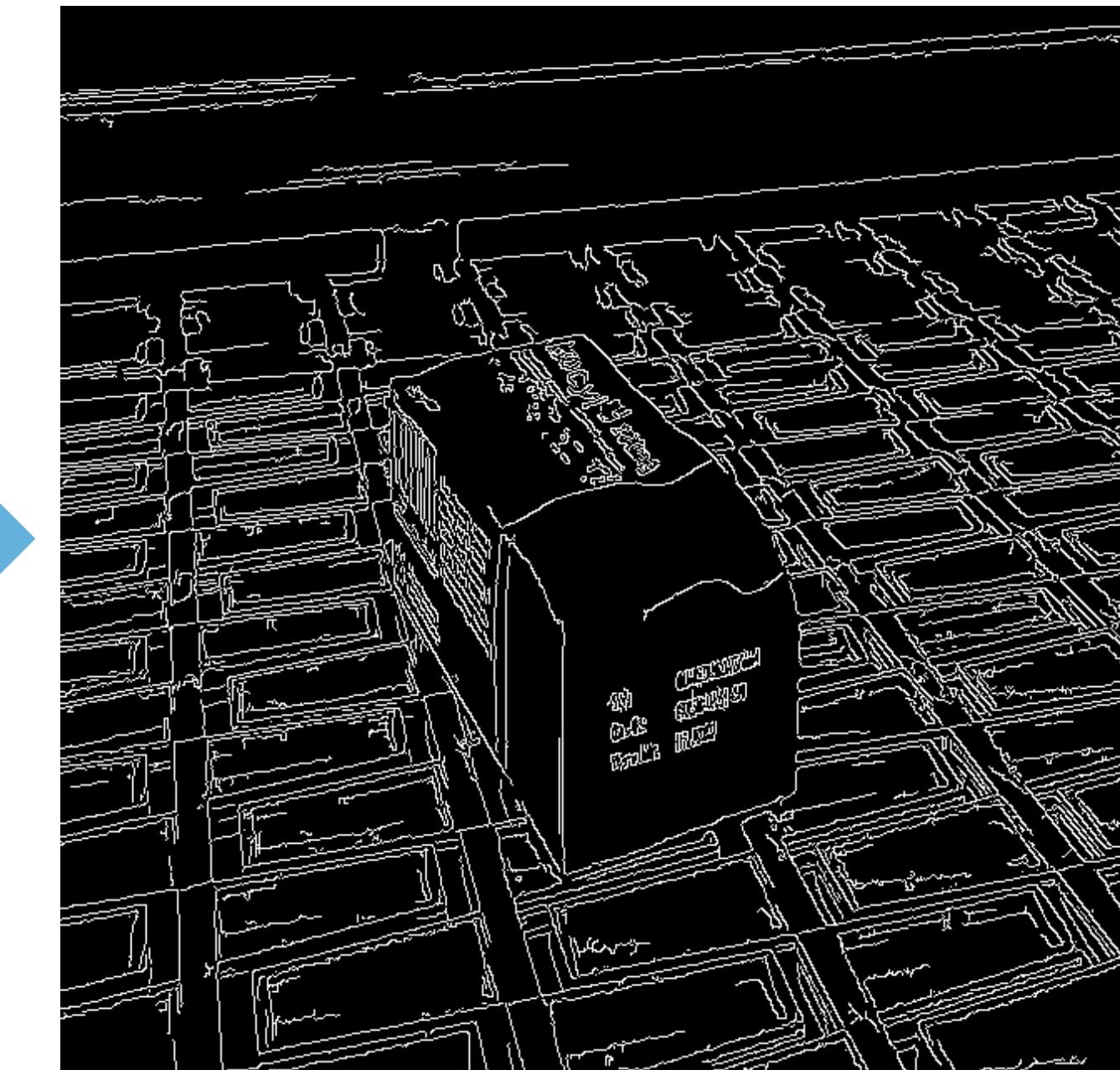
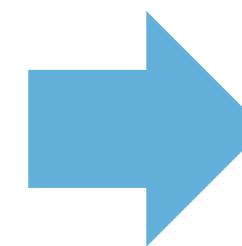


# 전처리

## Edge Detection



[ Before ]



[ After ]

# 전처리

## Embossing

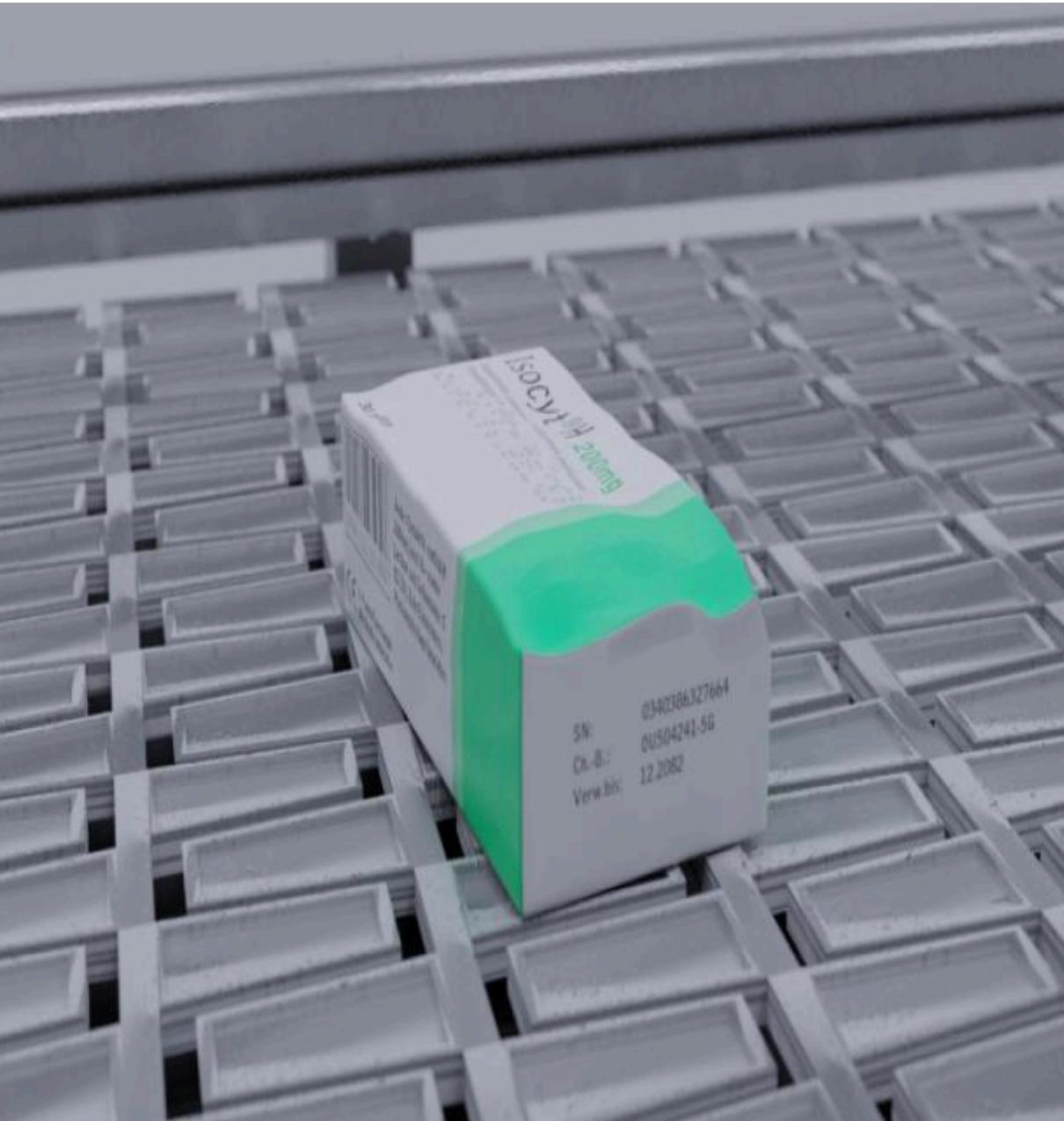
- 이미지에 입체감을 주기 위해 경계와 깊이감을 강조하는 필터링 기법.
- 일반적으로 그레이스케일 이미지에 특정 커널을 적용하여 수행.
- 이미지의 주요 형태를 강조하고 시각적 흥미를 증가시키는 데 사용.

### [ 적용 예시 ]

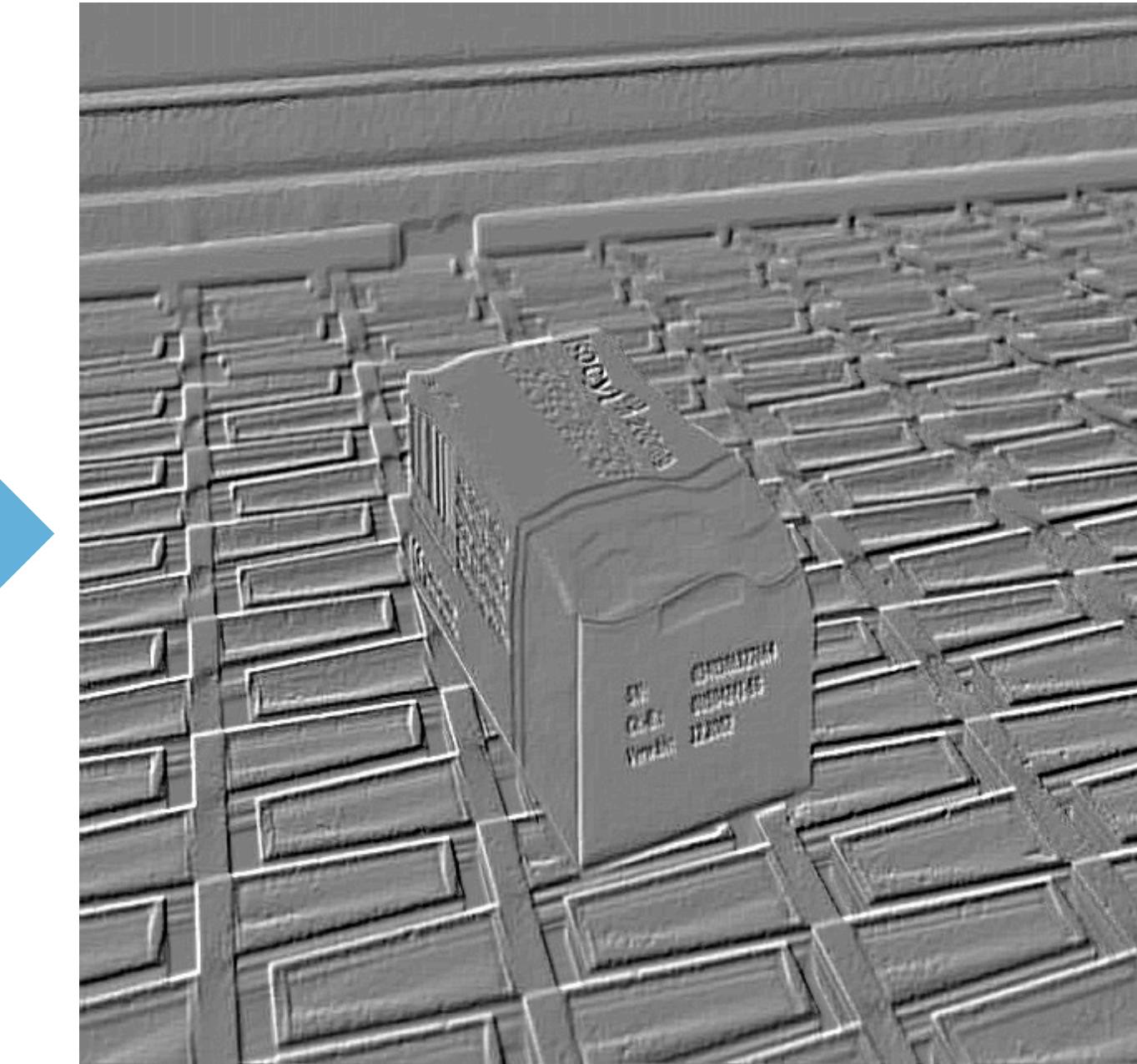
```
##### EMBoss #####
gray_scale = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
gray16 = np.int16(gray_scale)
emboss = np.uint8(np.clip(cv2.filter2D(gray16, -1, femboss) + 128, 0, 255))
```

# 전처리

## Embossing



[ Before ]



[ After ]

# 전처리 후 수행결과

**Origin**

Class	Images	Instances	Box(P)	R	mAP50	mAP50-95)
all	80	108	0.689	0.696	0.701	0.557
box	80	80	0.99	1	0.995	0.982
damaged	25	28	0.389	0.393	0.408	0.132

**CLAHE**

Class	Images	Instances	Box(P)	R	mAP50	mAP50-95)
all	80	138	0.767	0.763	0.758	0.565
box	79	79	0.978	1	0.988	0.962
damaged	32	59	0.557	0.525	0.528	0.167

**Edge Detection**

Class	Images	Instances	Box(P)	R	mAP50	mAP50-95)
all	80	80	0.499	1	0.623	0.502
box	40	40	0.499	1	0.671	0.553
damaged	40	40	0.499	1	0.576	0.451

**Embossing**

Class	Images	Instances	Box(P)	R	mAP50	mAP50-95)
all	80	80	0.704	0.918	0.888	0.745
box	40	40	0.703	0.886	0.884	0.758
damaged	40	40	0.705	0.95	0.891	0.731



## [ GUI 설명 ]

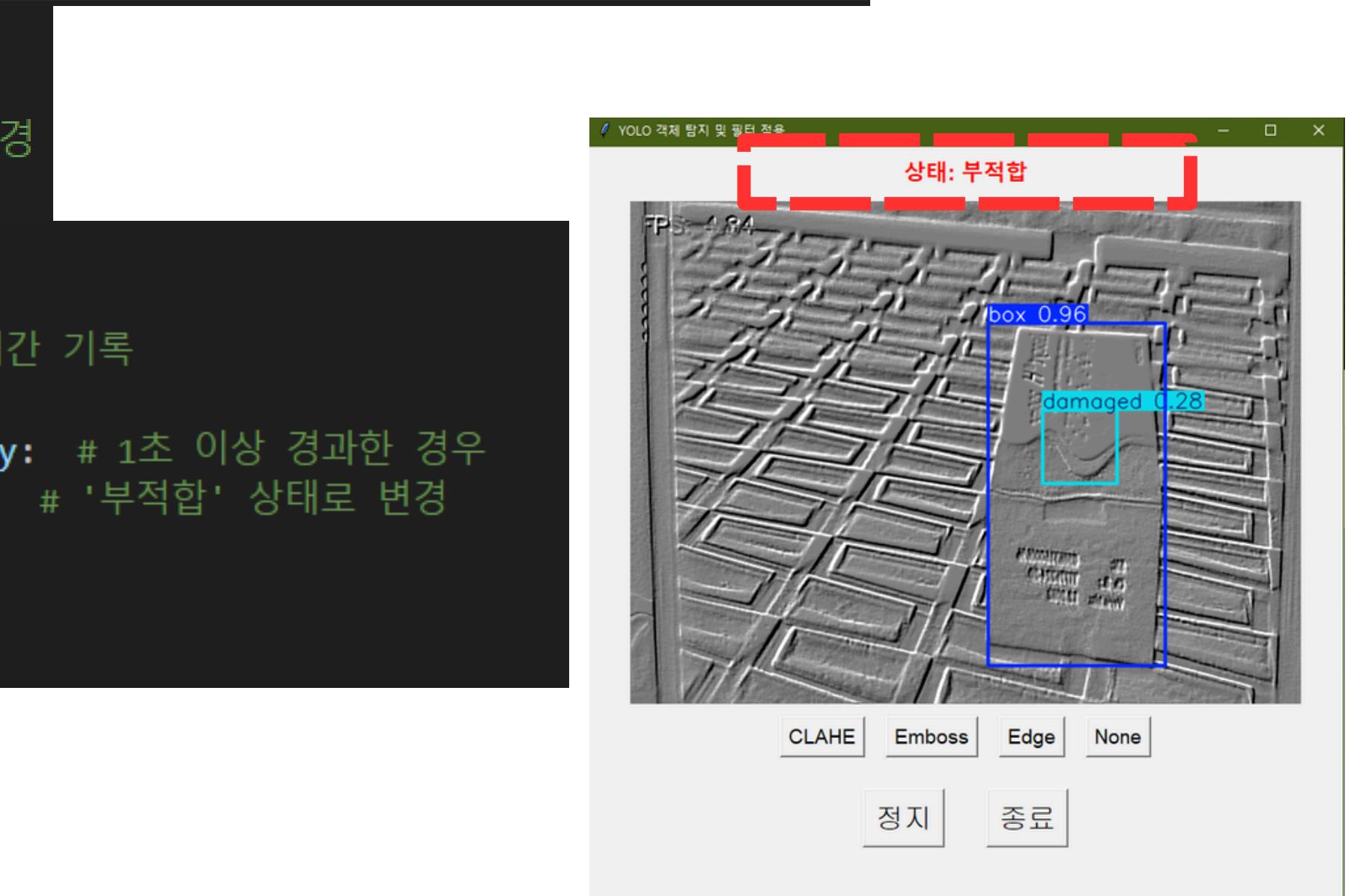
- ① 카메라에 통과되는 객체의 상태를 표기  
(정상 : 적합 / Damaged : 부적합)
- ② FPS : 초당 프레임
- ③ 객체 감지 카메라  
(레일에 지나가는 객체 탐지)
- ④ 전처리 필터 적용 버튼
- ⑤ 실시간 탐지 시스템 실행/정지 버튼
- ⑥ 프로그램 종료 버튼

## 박스의 상태에 따라 “적합” 또는 “부적합” 이라는 텍스트 출력

```
# 상태 텍스트
status_label = tk.Label(window, text="상태: 적합", font=("Helvetica", 16, "bold"), fg="green")
status_label.pack(pady=10)

# damaged_detected가 True로 유지된 시간을 추적할 변수
damaged_start_time = None
damaged_delay = 1.0 # 1초 이상 유지되어야 부적합으로 변경
last_saved_time = 0 # 마지막 저장 시간을 추적할 변수

# damaged_detected 상태 유지 시간 확인
if damaged_detected:
    if damaged_start_time is None: # 처음 감지되었을 때 시간 기록
        damaged_start_time = time.time()
    elif time.time() - damaged_start_time >= damaged_delay: # 1초 이상 경과한 경우
        status_label.config(text="상태: 부적합", fg="red") # '부적합' 상태로 변경
else:
    damaged_start_time = None
    status_label.config(text="상태: 적합", fg="green")
```



```
# 카메라 캡처
cap = cv2.VideoCapture(0)

# 프레임
video_label = tk.Label(window)
video_label.pack()

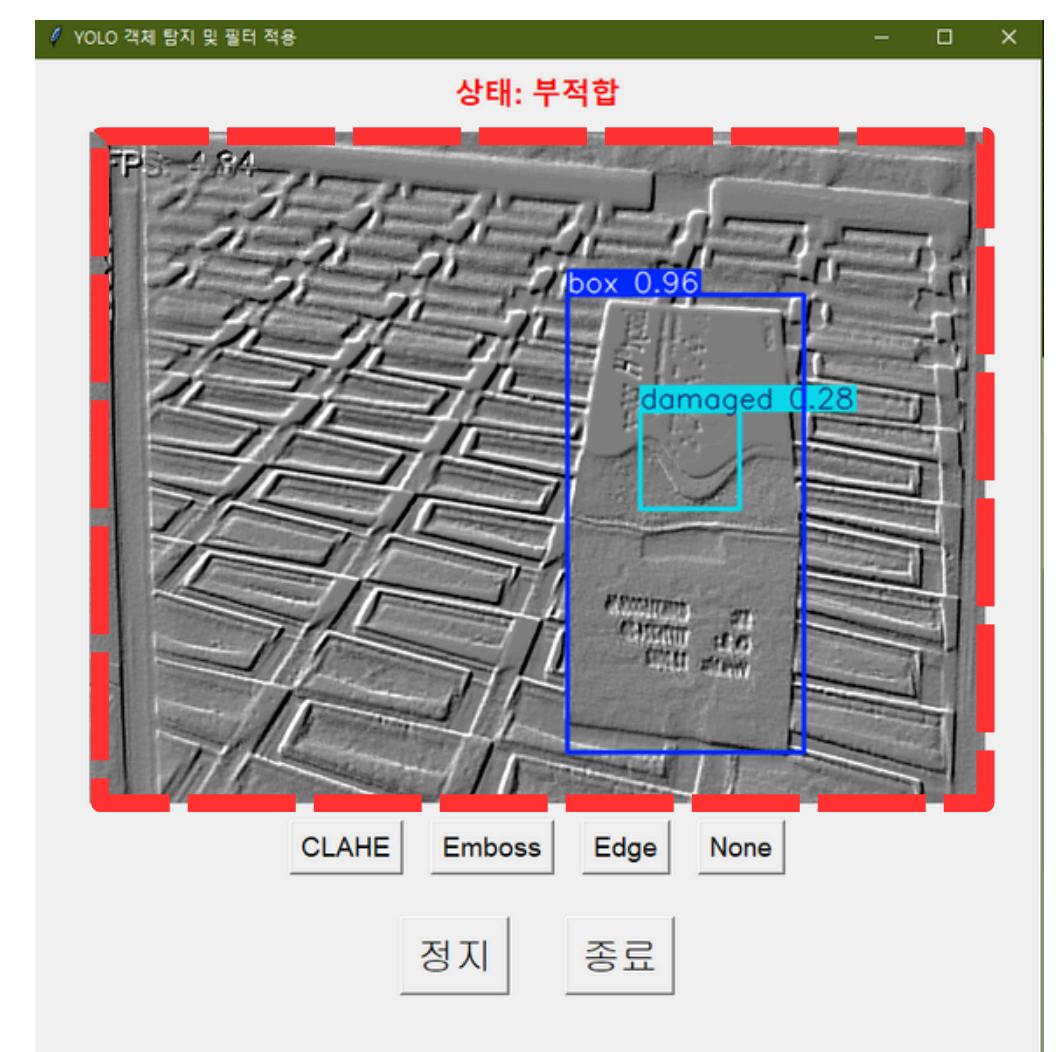
def update_frame():
    global prev_time, damaged_start_time, after_id, last_saved_time, filter_type
    ret, frame = cap.read()
    if not ret:
        print("웹캠에서 영상을 가져올 수 없습니다.")
        return

    # 좌우 반전
    frame = cv2.flip(frame, 1)

    # Tkinter 메인 루프 실행
    update_frame()
    window.mainloop()

# 웹캠 해제
cap.release()
```

카메라로 받아오는 영상의 매 프레임을 갱신

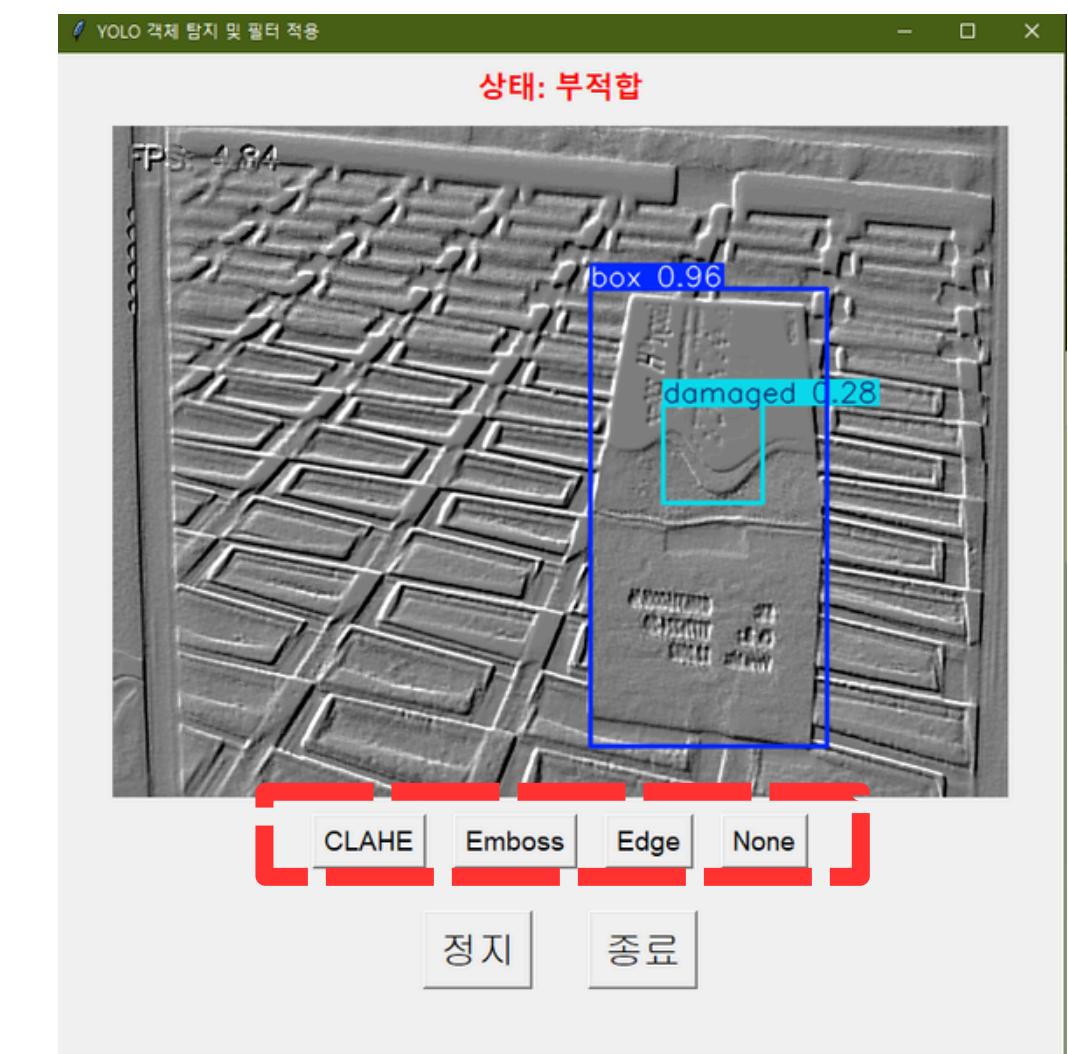


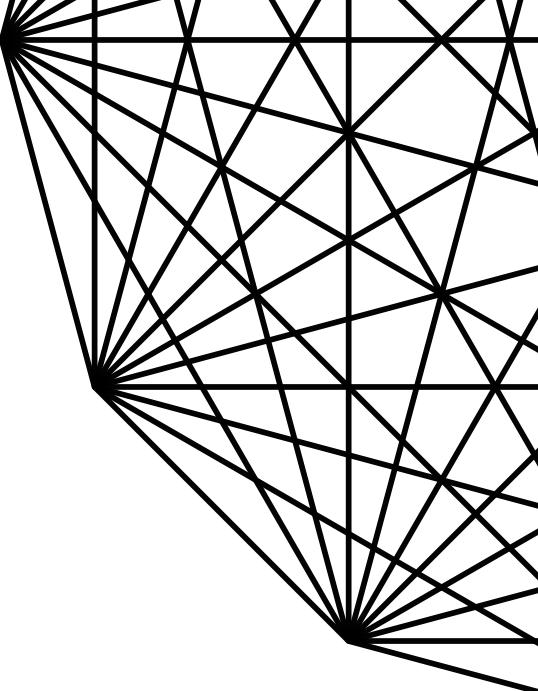
```
# CLAHE 적용 함수
def apply_clahe(frame):
    img_yuv = cv2.cvtColor(frame, cv2.COLOR_BGR2YUV)
    clahe = cv2.createCLAHE(clipLimit=3.0, tileGridSize=(8, 8))
    img_yuv[:, :, 0] = clahe.apply(img_yuv[:, :, 0])
    img_clahe = cv2.cvtColor(img_yuv, cv2.COLOR_YUV2BGR)
    return img_clahe

# Emboss 필터 적용 함수
def apply_emboss(frame):
    img_yuv = cv2.cvtColor(frame, cv2.COLOR_BGR2YUV)
    clahe = cv2.createCLAHE(clipLimit=3.0, tileGridSize=(8, 8))
    img_yuv[:, :, 0] = clahe.apply(img_yuv[:, :, 0])
    img_clahe = cv2.cvtColor(img_yuv, cv2.COLOR_YUV2BGR)
    gray = cv2.cvtColor(img_clahe, cv2.COLOR_BGR2GRAY)
    gray16 = np.int16(gray)
    embossed = np.uint8(np.clip(cv2.filter2D(gray16, -1, femboss) + 128, 0, 255))
    embossed = np.stack([embossed] * 3, axis=-1)
    return embossed

# 엣지 검출 필터 적용 함수
def apply_edge_detection(frame):
    img_yuv = cv2.cvtColor(frame, cv2.COLOR_BGR2YUV)
    clahe = cv2.createCLAHE(clipLimit=3.0, tileGridSize=(8, 8))
    img_yuv[:, :, 0] = clahe.apply(img_yuv[:, :, 0])
    img_clahe = cv2.cvtColor(img_yuv, cv2.COLOR_YUV2BGR)
    edges = cv2.Canny(img_clahe, 100, 200, 3)
    edges = np.stack([edges] * 3, axis=-1)
    return edges
```

## 프레임에 적용할 3가지 전처리 필터를 함수로 작성





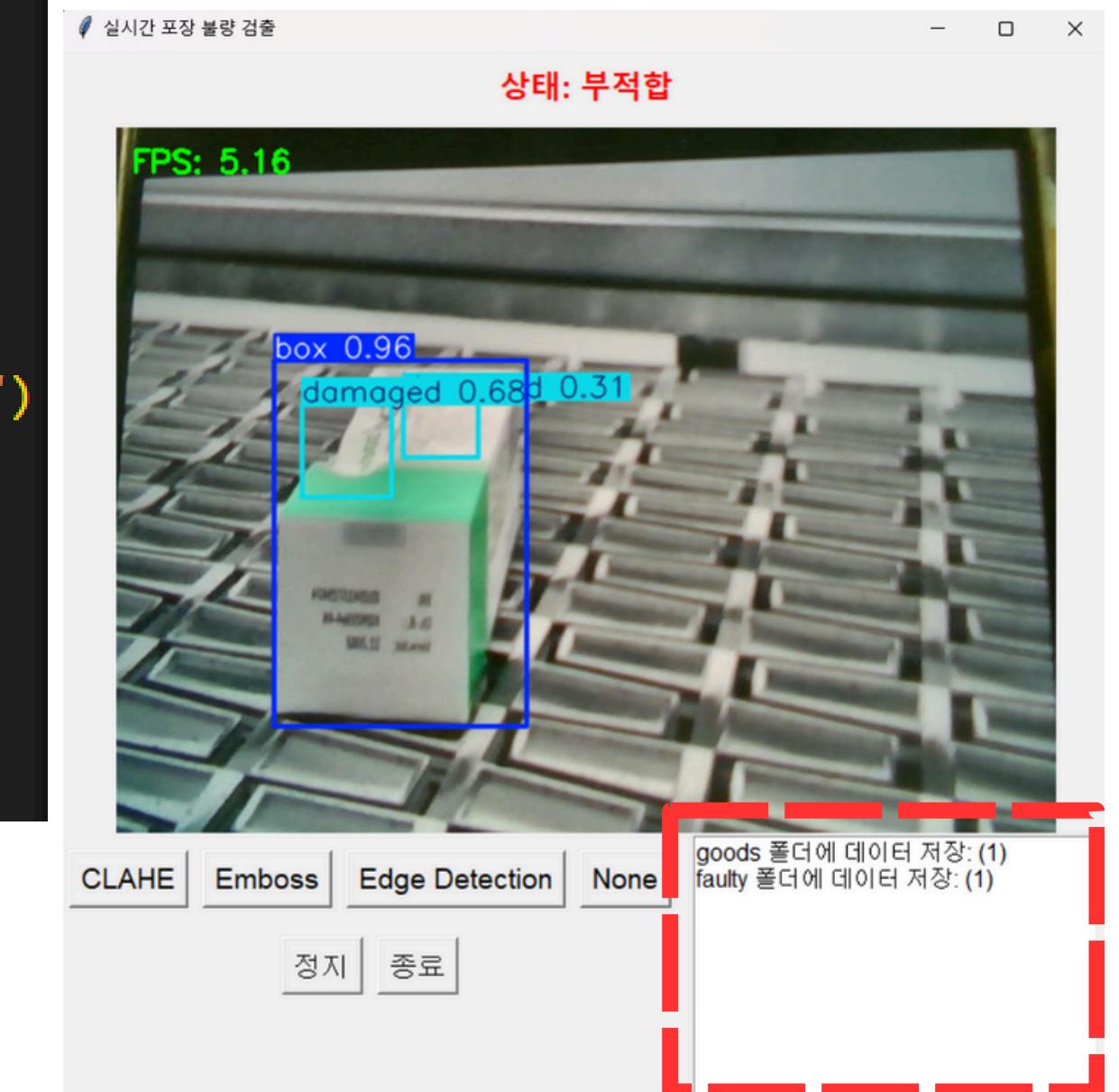
## 양품/불량품의 저장소를 나누고 저장할 때마다 개수 출력

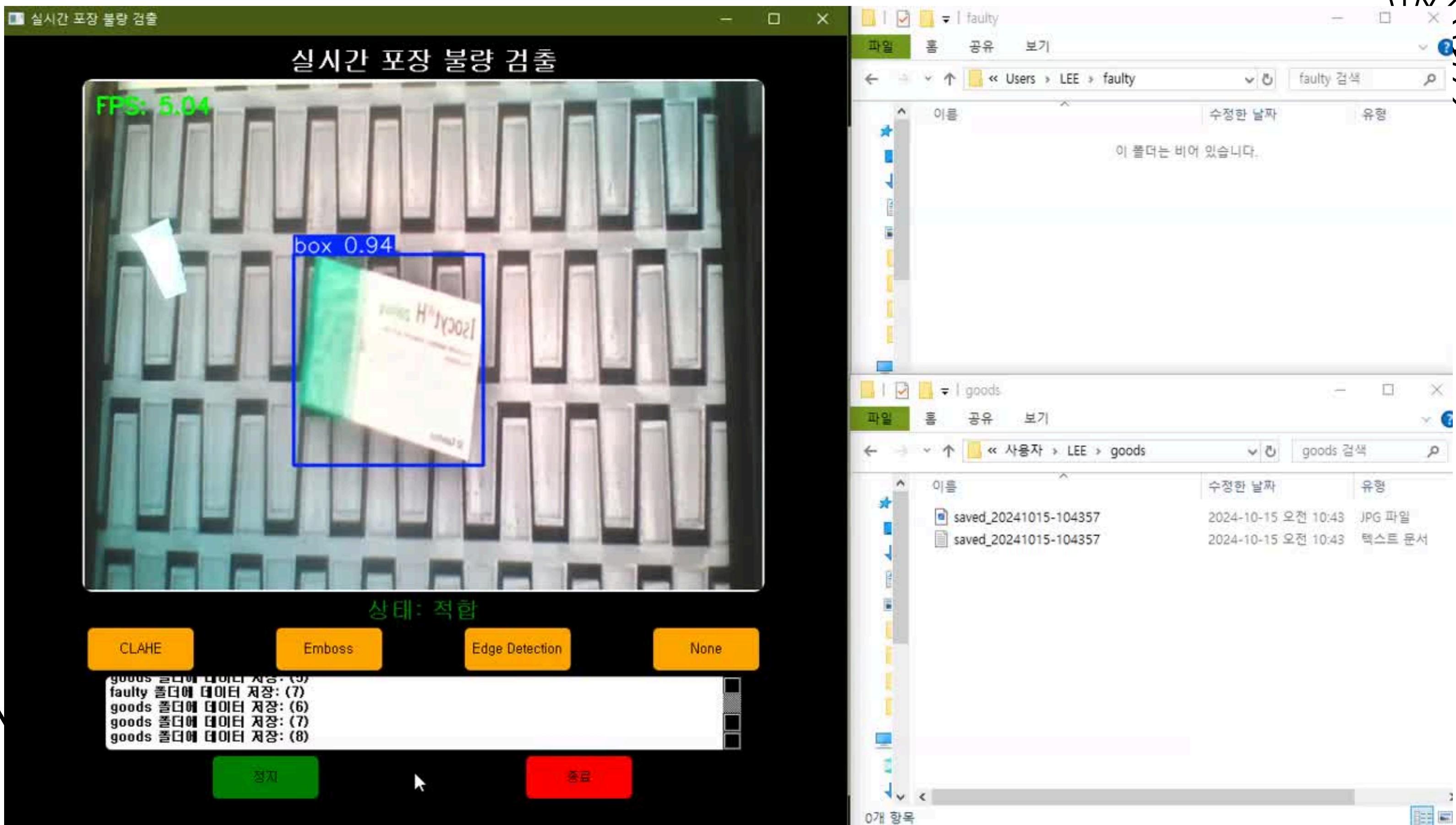
```

else:
    # damaged_detected가 False일 때, 초기화
    damaged_start_time = None
    damaged_duration = 0 # 부적합 상태 지속 시간 초기화
    status_label.config(text="상태: 적합", fg="green")

if current_time - no_damaged_start_time >= no_damaged_threshold:
    if damaged_duration >= status_maintenance_time:
        save_to_folder(faulty_folder, frame, labels_to_save)
        faulty_count += 1
        status_text.insert(tk.END, f"faulty 폴더에 데이터 저장: ({faulty_count})\n")
    else:
        save_to_folder(goods_folder, frame, labels_to_save)
        goods_count += 1
        status_text.insert(tk.END, f"goods 폴더에 데이터 저장: ({goods_count})\n")
damaged_duration = 0
no_damaged_start_time = current_time

```





# 프로젝트 회고 -유현재

## 완성도 평가

- 8/10
- 프로젝트를 진행하며 전반적인 흐름은 안정적으로 훌륭이나 데이터 훈련이나 이미지 전처리에서 최적의 하이퍼 파라미터를 찾는 등의 상세한 구현까지는 하지 못했음.

## 잘한부분 / 아쉬운점

- 하이퍼 파라미터 최적화, UI 디자인, DB와 서비스의 연동 후 나올 수 있는 부산물인 추가 수집 데이터를 처리하는 부분에서 아쉬움이 남음.

## 개선점 / 보완점(향후계획)

- 아쉬운 점에서 서술한 것들을 보완하여 조금 더 높은 정확도의 분류 성능을 내며 실제 서비스를 제공할 수 있을 정도의 완성도를 갖추는 것이 개인적인 욕심이다.

## 느낀점 / 성과

- 프로젝트를 진행하면서 사용한 대부분의 코드들은 이전에 이미 사용해봤던 코드이기 때문에 완전히 새로운 무언가를 시도한 것은 아니라 아쉬우면서도 다시 상기할 수 있는 기회이기도 했음.
- 프로젝트에서 사용한 코드에 대한 정확한 이해를 하며 넘어감.

# 프로젝트 회고 -유현욱

## 완성도 평가

- 완성도 평가 점수(10점) : 9점
- 스스로 9점을 준 이유는 목표로 했던 웹 구현을 아직 하지 못했고, 웹 구현을 하고 실제 사용자들이 사용할 수 있는 프로그램까지 만들었다면 조금 더 만족 할 만한 프로젝트였지 않나 싶습니다.

## 잘한부분 / 아쉬운점

- 프로젝트의 방향이 바뀌고 실제 주어진 1주일이라는 기간 내에 목표로 했던 프로그램을 완성하여 시연까지 했다는 점에서는 잘했다고 느껴집니다.
- 하지만 완성도 평가에서도 언급했던 Flask를 이용한 웹 서버 및 프로그램 구현단계까지는 가지 못했기 때문에 그 점이 조금 아쉽습니다.

## 개선점 / 보완점(향후계획)

- 향후에 Flask를 공부해서 웹 서버 구현을 목표로 하고 있고,
- GUI 환경 개선 및 실제 제조 현장에서 바로 사용 가능하도록 만들 계획입니다.

## 느낀점 / 성과

- 여러가지 분석 알고리즘을 사용한 점, 이러한 분석 로직을 기반으로 프로젝트를 수행한 점은 앞으로 회사에서 프로젝트를 수행할 때 많은 도움으로 와닿을 듯합니다.

# 프로젝트 회고 -이태훈

## 완성도 평가

- 완성도 평가 점수(10점) : 10점
- 프로젝트 방향성이 달라짐에 따라(기존 분석만 -> 실시간 감지 서비스 프로그램) 프로젝트의 목적과 기업에서 원하는 명확해졌기에 스스로 높은 점수를 매겨보았습니다.

## 잘한부분 / 아쉬운점

- 이미지에 대한 전처리를 어떻게 해야 할지 막막했지만 3가지 전처리를 수행하면서 평가지표를 높일 수 있었습니다.
- 3가지 이외에도 다양한 기법들이 있지만 다 사용해보지 못한점, 프로그램을 DB와 웹서비스로 만들지 못한점이 아쉽습니다.

## 개선점 / 보완점(향후계획)

- 데이터셋별로 전처리 기법에 따른 성능지표가 달라질 수 있어, 프로젝트에서 수행하지 못한 전처리 과정을 모두 수행하여 데이터 셋별로 전처리 필터를 적용할수 있는 프로그램으로 업그레이드를 하고 싶습니다.
- 또한 이미지 전처리 말고도 아키텍처 수정 및 하이퍼파라미터 튜닝등을 보완하여 웹 서비스로 구축하는 것까지 진행해볼 계획입니다.

## 느낀점 / 성과

- 짧은 기간에 목표했던 프로그램 구현까지 완료했다는 점에서 프로젝트의 일부 성과가 있다고 생각됩니다. 그러나 아직 바로 사용하지 못하는 점 또한 단점으로 존재하여 프로젝트가 끝나는 날까지, 끝이 나도 개선을 하여 실제 스마트 팩토리에 바로 적용 가능하게끔 만들어놓고 싶습니다.

# Q&A