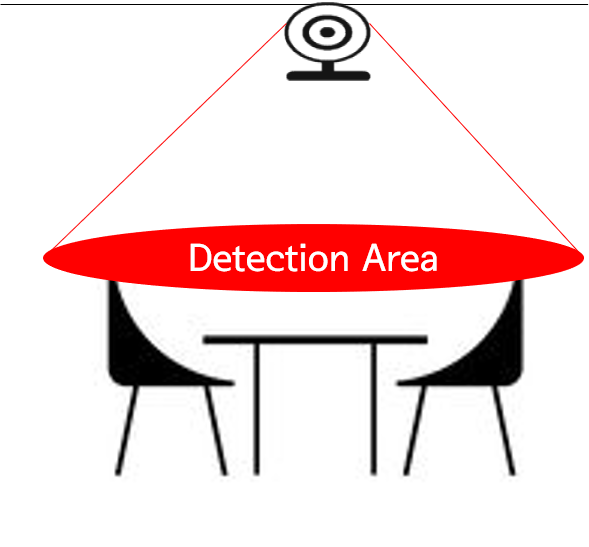
1. 시스템 구조

전체 탐지 구조

◀ 설치 장면 예정

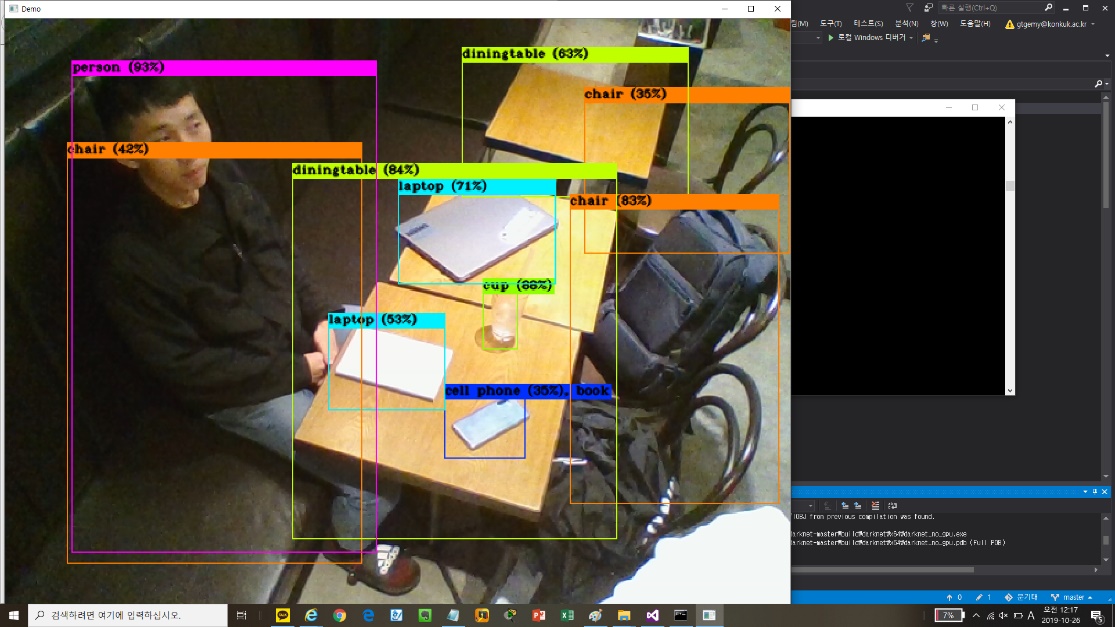
1. INPUT: Webcam or 라즈베리파이

설치 위치: 각 테이블 천장에 설치 예정

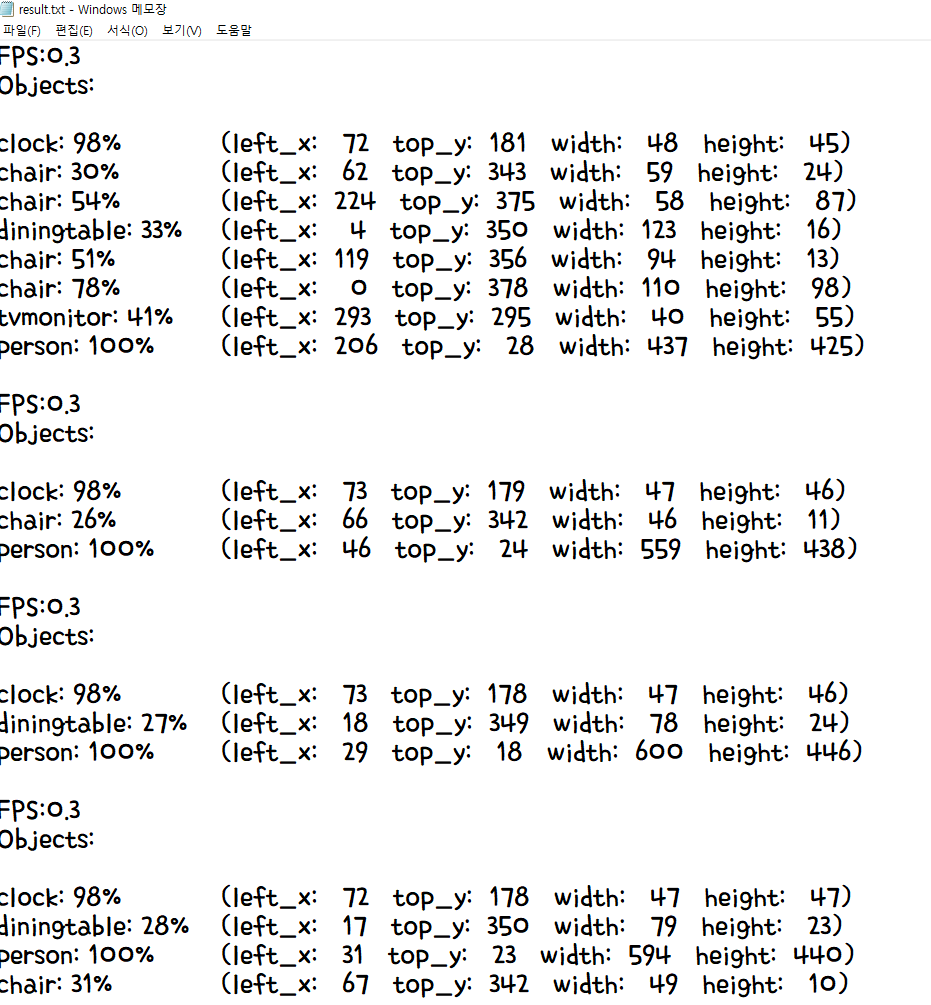
설치 범위: 테이블 및 의자가 한 프레임 안에 나오도록

 ◀ Webcam 시점

1. Process: YOLOv3 (.cfg, .weights)
2. Input data: real-time video → Object detection

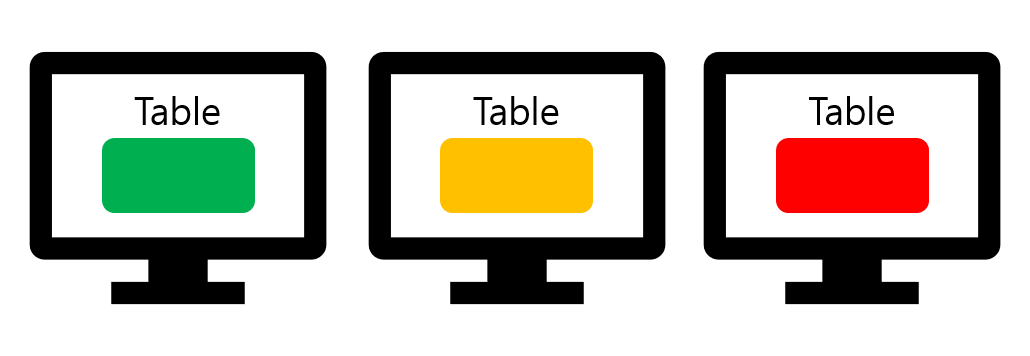
 ◀Object detection 예시

1. Object Detection → File.txt에 저장 (해당 인지된 객체에 대한 결과값 저장)

◀ 예시

예상방향: 현재는 데이터가 누적되어 저장, 그러나 객체탐지결과 또한 실시간으로   
인지된 해당 객체의 정보만 필요하기에 파일을 덮어 갱신하는 방향으로 개발 예정

1. Output: 라즈베리파이 or 모니터



1. 기능
2. 탐지 기준

* 미 주문 자리 (주문을 하지 않고 자리에 있는 경우)
* 유령 자리 (자리에 물건만 있고 사람이 장기적인 시간동안 없는 경우)
* 빈 자리 (정상적으로 이용한 후 자리를 비운 경우)

1. 탐지 객체 (9개)

* 주문유무 기준: 컵
* 테이블 위 객체 기준: 핸드폰, 가방, 책, 노트북, 지갑
* 탐지 제외기준: 테이블, 의자
* 사람
* 객체 수: 9개, 처리방식: Class=9 처리

1. 프로세스 개요
   1. 인지 데이터 구분

- 사람 인지 유무 (0,1로 판단) //0-사람 미인지, 1-사람 인지

- 책상 위 물건 인지 유무 (0,1로 판단) //0-테이블만 인지, 1-물건 인지

- 컵 인지 유무 (0,1로 판단) //0-미 주문, 1-주문

- 공석시간 인지 (Timer=20min 기준) ///20min↓-Checking, 20min↑-공석 처리

B. 데이터 평가

Evaluation Struct = [man, stuff, cup, timer]

(1). [0, don’t care, don’t care, don’t care, 20] = 유령자리

- 빈 자리와의 구분 기준: 유령자리의 경우 stuff || cup 둘 중 하나는 탐지가 되어야 한다.

- Timer 측정 기준: 사람이 탐지되지 않는 시점부터 다시 탐지될 때까지 시간 측정

**- 20분 기준**이며, 해당 물건은 종업원 카운터에 정리될 예정

(2). [1, don’t care, 0, don’t care]= 미 주문 자리

- 사람과 컵에 대한 인지만 하면 됨- 사람은 인지되고 컵은 인지되지 않는 경우

(3). [0, 0, 0, don’t care]= 빈 자리

- 사람-물건-컵 모두 인지되지 않는 시점이어야 한다.

- Chair/ Table 이외의 모든 객체가 인식되지 않으면 빈 자리로 평가한다.

※ 추가사항 발생 시 평가 테이블 갱신 예정

※don’t care: 값에 관계없이

1. 결과 값에 따른 처리방법
   1. 각 탐지 기준 결과에 따라 좌석 표시등 색깔을 다르게 표기할 예정

* 정상 사용 중인 자리: 빨간색
* 미 주문 자리: 보라색
* 유령 자리: 주황색
* 빈 자리: 초록색
  1. 결과에 따른 Action
* Red Light: None Action (정상 사용 중이기 때문에 다른 액션을 취하지 않는다.)
* 미 주문 자리: 해당 사용자에게 알리고 주문하거나 나가도록 조치한다.
* 유령 자리: 종업원이 가서 자리를 치운다. (치운 후엔 detection을 통해 빈 자리 처리)
* 빈 자리: None Action (빈 자리이게 다른 액션을 취하지 않는다.)
* 예외사항 시 추가 및 업데이트 예정

1. 동작과정

Final Output  
Input: Object detection  
Output: Print flow Action

Processor: YOLOv3 Model  
Input: Real-Time video data  
output: Object detection

Camera [WebCam]  
Input: real-time situation

* 과정1. 데이터 수집 [Real-Time video]

(1). YOLOv3를 이용해 Webcam 실행 (라즈베리파이 or 스마트폰 캠 사용 예정)

(2). Webcam을 통해 카페 테이블 상황을 수집

* 과정2. 데이터 처리 [YOLOv3 Model]

(1). .weight 파일 실행 [훈련된 데이터 로딩]

darknet에서 제공된 yolov3.weights파일 사용하며 인지되지 않는 객체에   
대해서는 추가로 training시킬 예정

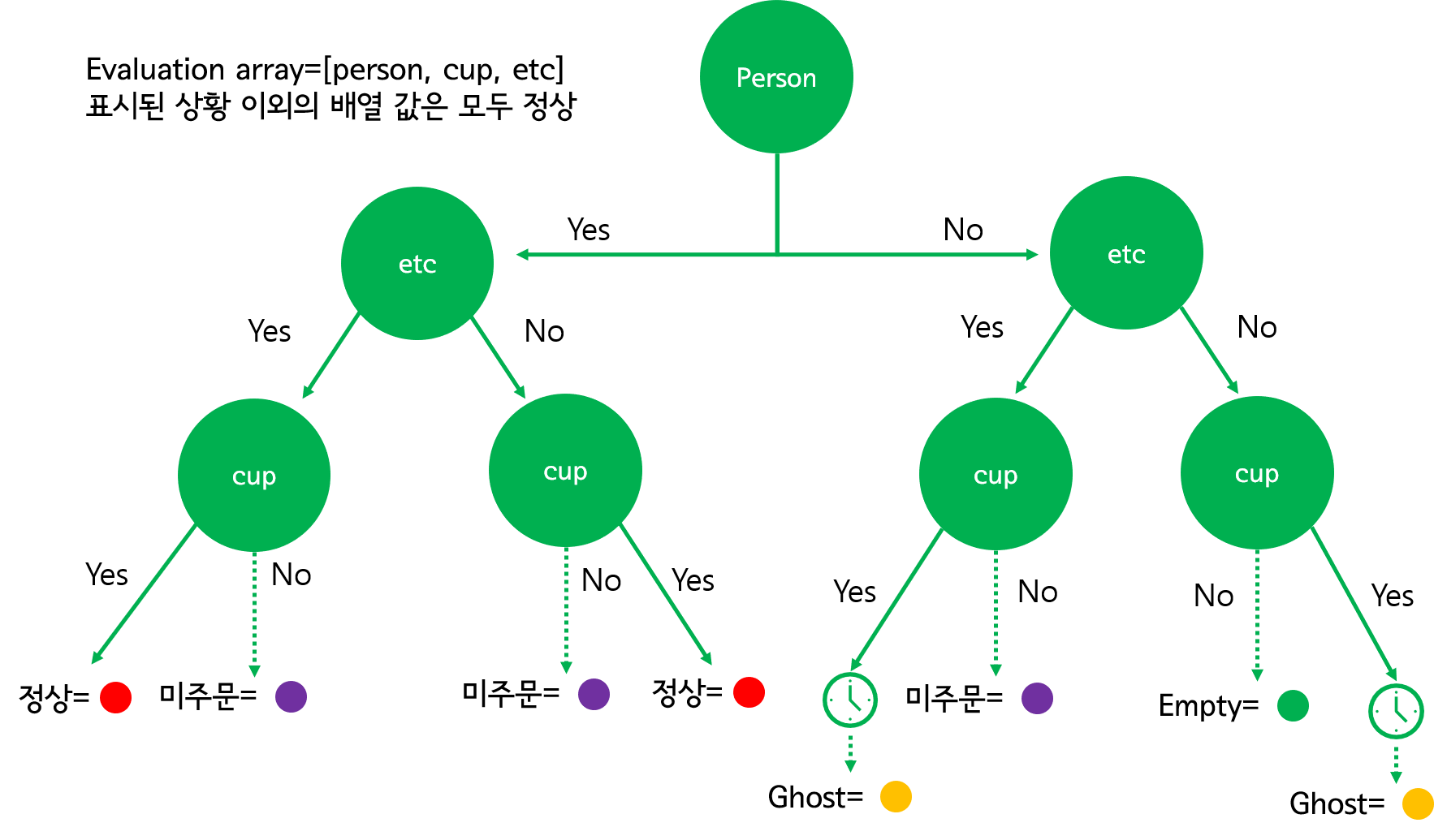
(2). cfg 파일 실행 [실제 평가]

.weights파일로 입력된 데이터에 대해 평가 (실제 인지 실행)

(3). 입력된 데이터 (Webcam data)를 YOLOv3 Model을 통해 평가

인지된 객체에 따른 평가 [유령/미 주문/빈자리/정상]

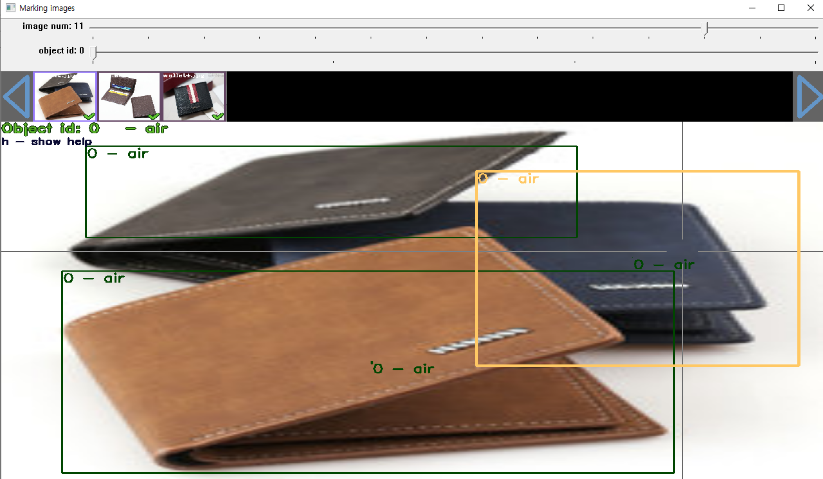
Array형으로 처리, 각 기준객체에 따른 배열형에 따라 평가 결과값



* 과정3. 출력 [라즈베리파이 or 모니터]

(1). 라즈베리파이 or 노트북 모니터를 통해 현 테이블 상황에 따른 표시등을 표시

1. 트레이닝
2. 개요
   1. Bounding box prediction (set object)
      1. 실제 트레이닝은 yolov3와 연동하여 트레이닝 할 예정



* yolo\_mark를 이용한 bounding box prediction

여러 샘플 사진을 통해 트레이닝 할 객체에 대해 box로 객체 트레이닝

* 1. Class Prediction
     1. 등록할 객체 수를 클래스 개수로 설정

Ex) object.data

classes= 3

train = data/train.txt

valid = data/train.txt

names = data/obj.names

backup = backup/

* + 1. Data 파일 내용: 클래스 수, 트레이닝 할 파일 설정
  1. Register Class name
     1. 등록할 객체에 대한 이름 설정

Ex) object.names: 객체 이름 파일

air

bird

wallet

* + 1. 실제 각 객체에 대한 좌표 값

Ex) train.txt: 트레이닝 된 객체 파일에 대한 정보

data/img/air1.jpg

data/img/air2.jpg

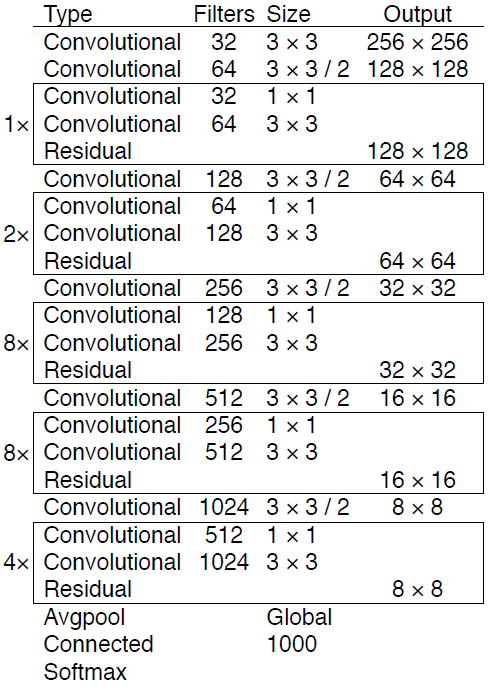
Ex) about air1.jpg: 트레이닝 된 객체의 좌표 값

0 0.400781 0.195139 0.596875 0.254167

0 0.445313 0.694444 0.743750 0.561111

* + 1. 트레이닝

Train.txt objects.names, objects.data파일에 대해 convolutional file로 트레이닝

▶ Convolution 파일 내용

* + 1. 트레이닝 결과

이후 트레이닝 된 weights파일을 통해 detection 실행

