


-요양원의 어르신들이 10년을 더 사실 수 있도록
잘 섬기는 기계만들기 프로젝트-



십년증수 팀 잘섬기계 프로젝트

팀 구성

team POSE <ul style="list-style-type: none">- openpose 코드 분석 및 수정- flutter를 사용한 front-end 담당	team YOLO <ul style="list-style-type: none">- yolact 코드 분석 및 수정- 잘섬기계 프로그램 back-end담당
김한울 (기획)	김택수(기술)
안은선 (디자인)	권재현(yolact)
김소연 (디자인)	

개발환경

SW

OS: ubuntu 18.04

GPU Driver: 460.86

CUDA: 11.1

cuDNN: 8.1.1

Pytorch: 1.8.1+cu111

torchvision: 0.9.1+cu111

torchaudio: 0.8.1

opencv: 4.5

HW

CPU: Intel - i9

GPU: RTX 3090 x2

RAM: 32G

Tool

visual studio

flutter

colab

Language

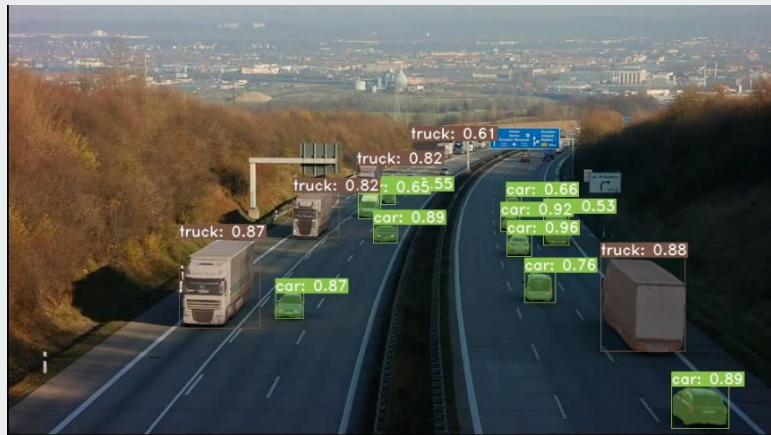
python: 3.8

dart

YOLACT: Real-time Instance Segmentation

yolo는 객체 탐지는 가능하지만 객체별 영역 탐지가 불가능
pixellib는 객체별 영역 탐지는 가능하지만 R-CNN기반 모델로 실시간으로 사용하기에는 한계가 존재

YOLACT는 2019년 국제 컴퓨터 비전 학회(ICCV)에서 발표된 모델
객체별 영역탐지를 실시간 수준에서 탐지할 수 있고, 모델 자체도 가벼운 편에 속한다는 것이 장점



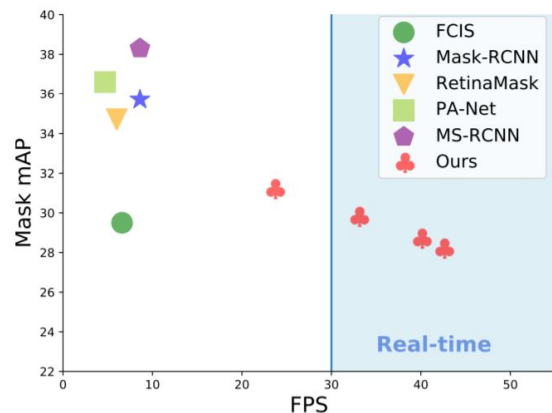


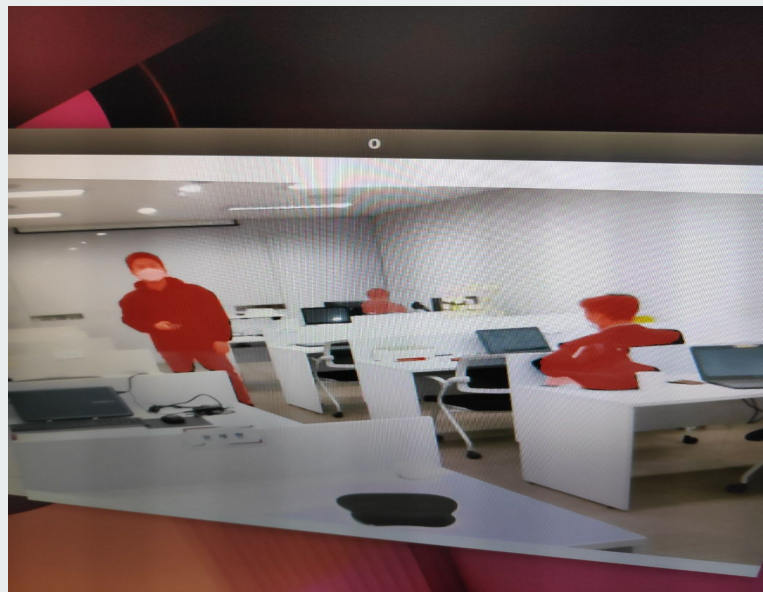
Figure 1: Speed-performance trade-off for various instance segmentation methods on COCO. To our knowledge, ours is the first *real-time* (above 30 FPS) approach with around 30 mask mAP on COCO *test-dev*.

Method	Backbone	FPS	Time	AP	AP ₅₀	AP ₇₅	AP _S	AP _M	AP _L
PA-Net [29]	R-50-FPN	4.7	212.8	36.6	58.0	39.3	16.3	38.1	53.1
RetinaMask [14]	R-101-FPN	6.0	166.7	34.7	55.4	36.9	14.3	36.7	50.5
FCIS [24]	R-101-C5	6.6	151.5	29.5	51.5	30.2	8.0	31.0	49.7
Mask R-CNN [18]	R-101-FPN	8.6	116.3	35.7	58.0	37.8	15.5	38.1	52.4
MS R-CNN [20]	R-101-FPN	8.6	116.3	38.3	58.8	41.5	17.8	40.4	54.4
YOLACT-550	R-101-FPN	33.5	29.8	29.8	48.5	31.2	9.9	31.3	47.7
YOLACT-400	R-101-FPN	45.3	22.1	24.9	42.0	25.4	5.0	25.3	45.0
YOLACT-550	R-50-FPN	45.0	22.2	28.2	46.6	29.2	9.2	29.3	44.8
YOLACT-550	D-53-FPN	40.7	24.6	28.7	46.8	30.0	9.5	29.6	45.5
YOLACT-700	R-101-FPN	23.4	42.7	31.2	50.6	32.8	12.1	33.3	47.1

Table 1: **MS COCO [28] Results** We compare to state-of-the-art methods for mask mAP and speed on COCO *test-dev* and include several ablations of our base model, varying backbone network and image size. We denote the backbone architecture with *network-depth-features*, where R and D refer to ResNet [19] and DarkNet [36], respectively. Our base model, YOLACT-550 with ResNet-101, is 3.9x faster than the previous fastest approach with competitive mask mAP.

모델 활용

사전학습된 모델이 인식하는 모든 객체가 아니라,
프로젝트에 필요한 특정 객체의 영역만 인식하도록 수정



특정영역만 검출

```
with timer.env('Copy'):  
    # 특정 객체만 검출  
    for i in range(len(t[0])):  
        if t[0][i] != 59:  
            t[1][i] = 0  
  
    args.top_k = 15  
    idx = t[1].argsort(0, descending=True)[:args.top_k]  
  
    if cfg.eval_mask_branch:  
        # Masks are drawn on the GPU, so don't copy  
        masks = t[3][idx]  
        classes, scores, boxes = [x[idx].cpu().numpy() for x in t[:3]]
```

```
COCO_CLASSES = ('person', 'bicycle', 'car', 'motorcycle', 'airplane', 'bus',  
                'train', 'truck', 'boat', 'traffic light', 'fire hydrant',  
                'stop sign', 'parking meter', 'bench', 'bird', 'cat', 'dog',  
                'horse', 'sheep', 'cow', 'elephant', 'bear', 'zebra', 'giraffe',  
                'backpack', 'umbrella', 'handbag', 'tie', 'suitcase', 'frisbee',  
                'skis', 'snowboard', 'sports ball', 'kite', 'baseball bat',  
                'baseball glove', 'skateboard', 'surfboard', 'tennis racket',  
                'bottle', 'wine glass', 'cup', 'fork', 'knife', 'spoon', 'bowl',  
                'banana', 'apple', 'sandwich', 'orange', 'broccoli', 'carrot',  
                'hot dog', 'pizza', 'donut', 'cake', 'chair', 'couch',  
                'potted plant', 'bed', 'dining table', 'toilet', 'tv', 'laptop',  
                'mouse', 'remote', 'keyboard', 'cell phone', 'microwave', 'oven',  
                'toaster', 'sink', 'refrigerator', 'book', 'clock', 'vase',  
                'scissors', 'teddy bear', 'hair drier', 'toothbrush')
```

t = [classes, scores, boxes, masks]

(t[0],t[1],t[2],t[3])의 길이는 검출한 객체의 수)



검출 객체 수 만큼 for문을 사용하여 객체의 class를 확인



지정한 class가 아닐 경우 score를 0으로 변환



threshold의 기준을 넘지 못하여 mask처리가 되지 않음



지정한 객체의 영역만 mask처리가 되어 색칠
(mask값이 1은 색칠, 0은 원본)

검출한 객체의 영역 좌표값 활용

```
159 with timer.env('Postprocess'):
160     save = cfg.rescore_bbox
161     cfg.rescore_bbox = True
162     t = postprocess(dets_out, w, h, visualize_lincomb = args.display_lincomb,
163                   crop_masks = args.crop,
164                   score_threshold = args.score_threshold)
165     cfg.rescore_bbox = save
166
167     ay, ax = t[3][0].shape
168
169     detected_area = np.zeros((ay,ax))
170     for i in range(len(t[0])):
171         if int(t[0][i]) == 67:
172             detected_area += t[3][i].cpu().detach().numpy()
173     if np.sum(detected_area) == 0:
174         print('카메라에 원하는 대상이 없습니다. 다시 시도해주세요.')
175     else:
176         np.savetxt('area.csv', detected_area, delimiter = ',')
177
178
1188
1189 def realtime_detecting():
1190     body_estimation = Body('./model/body_pose_model.pth')
1191     target_area = np.loadtxt('area.csv', delimiter = ',')
1192     print(f"Torch device: {torch.cuda.get_device_name()}")
1193
1194     cap = cv2.VideoCapture(0)
1195     cap.set(3, 640)
1196     cap.set(4, 480)
1197
1198     state_time = 0
1199     pose_score = 0
1200     post_pose = 0
1201
```

화면 캔버스 크기의 0으로 채워진 배열 `detected_area` 생성



검출 객체의 수 만큼 for문, 특정 class의 객체일 경우



mask 값을 pytorch tensor에서 numpy array로 변환,
`detected_area`에 행렬 연산



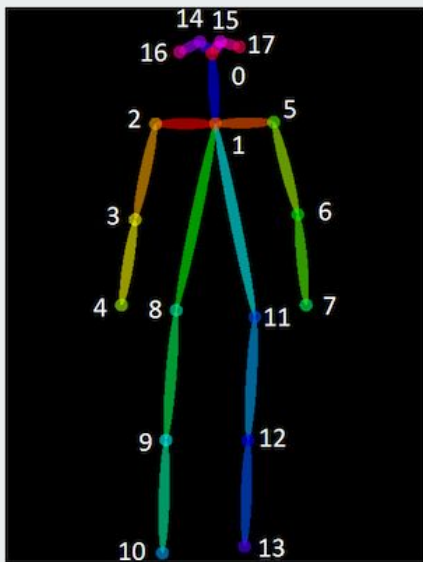
전체 객체의 mask값이 더해진 캔버스인 `detected_area`를
csv 파일로 저장



이후 `area.csv` 파일을 불러와 `target_area`로 설정,
mask영역 확인에 활용

OpenPose

신체의 키포인트와 그 키포인트들을 연결하는 선을 그어 스켈레톤을 보여주는 모델



출처: <https://github.com/Hzzzone/pytorch-openpose.git>

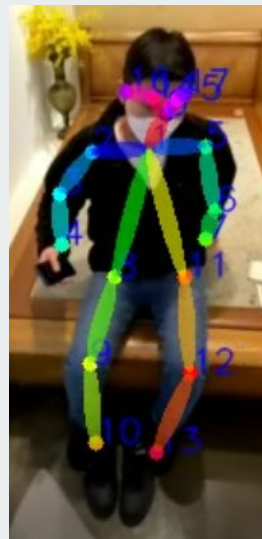
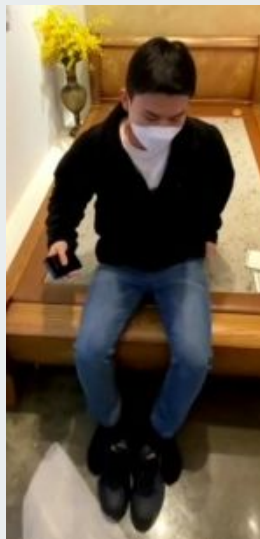
낙상방지 알고리즘

YOLACT를 통해 침대 영역 설정



양쪽 엉덩이 포인트가 침대 영역 안에 있는 상태
+ 한쪽 무릎이라도 침대 영역 이탈
+ 일정 시간 경과

OpenPose를 통해 신체 포인트 인식



-> 낙상 위험 경고

신체 포인트의 좌표값을 이용해 상황 판별

양쪽 엉덩이, 양쪽 무릎의 포인트 x,y값 정의



4개의 포인트가 전부 검출되지 않을 경우 **pass**,
전부 검출될 경우 위험,안전 판별



4개 포인트의 x,y로 target_area로 접근, mask 값이 전부
1일 경우 안전, 그 외의 경우에는 위험으로 판별



위험일 경우 카운팅, 기준 수치만큼 카운팅 될 경우 알람

안전일 경우 카운팅 수치 0으로 초기화

```
1215 print('RKnee point: ',RKnee_point)
1216 print('LKnee point: ',LKnee_point)
1217
1218 if len(RHip_point) == 0 or len(LHip_point) == 0 or len(RKnee_point) == 0 or len(LKnee_point) == 0:
1219     print('해당 포인트가 화면에 잡히지 않았습니다.')
1220
1221 else:
1222     state = []
1223
1224     RHip_x, RHip_y = int(RHip_point[0][0]),int(RHip_point[0][1])
1225     LHip_x, LHip_y = int(LHip_point[0][0]),int(LHip_point[0][1])
1226     RKnee_x, RKnee_y = int(RKnee_point[0][0]),int(RKnee_point[0][1])
1227     LKnee_x, LKnee_y = int(LKnee_point[0][0]),int(LKnee_point[0][1])
1228
1229     if target_area[RHip_y][RHip_x] and target_area[LHip_y][LHip_x] and target_area[RKnee_y][RKnee_x] and target_area[LKnee_y][LKnee_x]:
1230         object_state = 0
1231         state.append(object_state)
1232     else:
1233         object_state = 1
1234         state.append(object_state)
1235
1236 if sum(state) > 0:
1237     print('해당 포인트가 영역에서 벗어났습니다!')
1238     state_time += 1
1239     if state_time >= 3:
1240         playsound("./warning_bell.mp3")
1241
1242     # push_service = FCMNotification(api_key="") # 파이어베이스 서버 api값 입력
1243     # registration_id = "" # 기기 토큰값 입력
1244     # message_title = "낙상 위험 경고"
1245     # message_body = "낙상 방지를 위해 내용을 확인해주세요."
```

```
1242     # push_service = FCMNotification(api_key="") # 파이어베이스 서버 api값 입력
1243     # registration_id = "" # 기기 토큰값 입력
1244     # message_title = "낙상 위험 경고"
1245     # message_body = "낙상 방지를 위해 내용을 확인해주세요."
1246     # result = push_service.notify_single_device(registration_id=registration_id, message=message)
1247     # print(result)
1248
1249     state_time = 0
1250 else:
1251     print('안전한 상태입니다.')
1252     state_time = 0
1253
1254 RShoulder_point = util.get_RShoulder_point()
1255 LShoulder_point = util.get_LShoulder_point()
1256 Nose_point = util.get_Nose_point()
1257 Neck_point = util.get_Neck_point()
1258
```

욕창방지 알고리즘

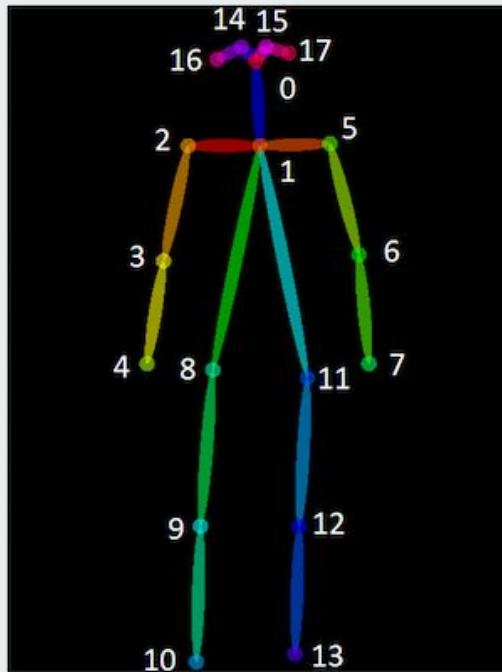
openpose로부터 받은 신체의 key points 좌표값을 인식

좌표값에 자세추정 알고리즘을 적용
-> 앞쪽, 뒤쪽, 오른쪽, 왼쪽 자세 분류

현재 자세와 과거 자세를 비교

자세가 같을 경우 스코어를 더해주고,
스코어가 일정이상일 경우 알람

현재 자세와 과거 자세가 다를 경우 스코어값
초기화



자세 추정 알고리즘

오른쪽 어깨, 왼쪽 어깨, 코, 목의 좌표 값과 유클리디안 거리를 응용
왼쪽, 오른쪽 어깨 간의 거리 값과 코와 목의 거리에 임의의 α 를 곱한 값을 비교
-> **Front or Back / Right or Left** 판별

Front or Back

왼쪽 어깨와 오른쪽 어깨의 x좌표값을 비교 -> **Front / Back** 판별

Right or Left

코와 목의 x 좌표값을 비교 -> **Right / Left** 판별

자세 추정 알고리즘

Front or Back

Front

```
if (LShoulder_x-RShoulder_x)**2 + (LShoulder_y-RShoulder_y)**2 >= ((Nose_x-Neck_x)**2 + (Nose_y-Neck_y)**2)*α:  
    if Nose_y <= Neck_y and LShoulder_x > RShoulder_x:  
        current_pose = 'F'  
    elif Nose_y > Neck_y and LShoulder_x <= RShoulder_x:  
        current_pose = 'F'  
    elif Nose_y > Neck_y and LShoulder_x > RShoulder_x:  
        current_pose = 'B'  
    elif Nose_y <= Neck_y and LShoulder_x <= RShoulder_x:  
        current_pose = 'B'
```

Back

Right or Left

Right

```
elif (LShoulder_x-RShoulder_x)**2 + (LShoulder_y-RShoulder_y)**2 < ((Nose_x-Neck_x)**2 + (Nose_y-Neck_y)**2)*α:  
    if Nose_x > Neck_x and Nose_y > Neck_y:  
        current_pose = 'R'  
    elif Nose_x <= Neck_x and Nose_y <= Neck_y:  
        current_pose = 'R'  
    elif Nose_x <= Neck_x and Nose_y > Neck_y:  
        current_pose = 'L'  
    elif Nose_x > Neck_x and Nose_y <= Neck_y:  
        current_pose = 'L'
```

Left



감사합니다!