Multi-Camera Object Detection 기반 실시간 확진자 실내 동선에 관한 연구

A Study on Multi-Camera Object Detection Based Real-time Interior Circulation of a confirmed case of Corona Virus

김용민, 김혜영, 서준호
(YongMin Kim, HyeYeong Kim, JunHo Seo)

한국항공대학교 소프트웨어공학 전공

초 록

코로나19 재확산 및 공공시설 방역이 대두되는 가운데 상기 팀원은 사태의 심각성을 인지하여 최소한의 비용으로 방역수칙과 안전관리에 효율성을 높이는 주제를 연구했다. 이에, Multi-Camera를 이용하여 실시간 Object Detection을 위해 YOLO, Darknet 모델을 사용하였고, 확진자 Tracking을 위해 Re-Identification 알고리즘을 사용했다. 사람 인식을 위한 데이터셋은 Market 1501을 사용하였으며, 이 연구를 통해 기존에 DeepSort알고리즘으로 사람을 인식하였을 때 장애물에 가려졌다가 나타났을 때 재인식을 못하는 상황 등을 해결하였으며, 결론적으로 소프트웨어를 활용하여 최소한의 비용과 인력으로 코로나19 방역과 예방에 도움을 줄 수 있음을 알게 되었다.

핵심용어: 객체인식, 인공지능, 멀티카메라, 코로나 바이러스
Key words: Object Detection, AI, Multi Camera, COVID-19

I. 서 론

2020년 초부터 확산되기 시작한 코로나 바이러스 감염증 - 19(이하 코로나 바이러스)는 전 세계적으로 전염병에 대한 공포심과 경제, 사회적인 손실이 발생하고 있다. 국내에서는 국가적 차원에서 지난 3월, 전염병 확산 방지를 위한 빅데이터 확진자 역학조사 시스템을 구축하였고, 이에 따라 평균 24시간 이상 소모되던 확진자 동선파악이 10분 내로 대폭 감소하는등, IT 강국으로서의 위상을 한층 더 높였다.

그러나 현재 실외 확진자의 동선파악은 가능하나 다중 이용시설 내부에서 확진자의 이동 동선파악에는 다소 어려움이 있다. 비말로 감염되는 바이러스 특성상 실외보다는 실내에서 감염될 확률이 크다. 또한 실내 공간에서는 거리두기가 쉽지않고, 환기가 잘 되지 않기 때문에 실외 공간보다 더 위험 할 수 있다. 실내에서는 바이러스가 오래 생존할 수 있는 금속, 유리, 플라스틱 등과 접촉할 가능성이 더 높기 때문에 전염병 확산 방지를 위해 실내 방역수칙을 강구해야 한다.

전염병 확산 방지의 핵심은 다중 이용시설의 통제에 있다. 사회적 거리두기 단계 격상으로 인해 인력은 이전보다 감소하였으나 관리의 난이도는 증가하였고, 기존에 열 감지 카메라 등 방역관리 시스템은 인원이 직접 통제하며 고가의 기기를 구매해야하는 부담이 있다.

본 논문에서는 적은 인력과 비용으로 관리가 수월할 수 있도록 Muliti-Camera를 통해 확진자를 Detection 하고, 동선을 Web Page에 그래프의 형태로 나타내었다. 이를통해 기존에 시설 관리자는 기존에 보유하고있는 CCTV장비를 통해 인력 소모를 줄이고 확진자의 동선 파악, 접촉자 위치파악 등 방역과 안전관리의 효율성을 높이는것을 목표로 한다.

이 논문의 구성은 2장에서 관련된 연구 조사, 3장에서 Multi-Camera 기반 DeepSort Alogorithm 모델 및 시스템 구현, 4장에서 실험 및 결론에 대하여 논한다.

II. 관련 연구

1. 딥러닝 기반 객체 검출 알고리즘

기존에 주로 딥러닝을 이용하여 영상 내 객체를 인식하는 알고리즘은 컨볼루젼 신경망 네트워크(Convolutional Neural Network)로 수많은 계층을 만들어 학

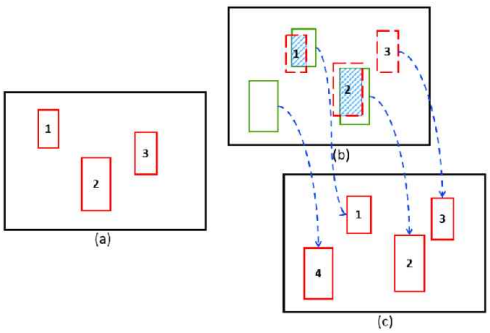
습하는 것을 기반으로 하고 있다. 이에 사용되는 R-CNN, Fast R-CNN등 모델이 있다. 우리는 실시간으로 다양한 카메라에서 다양한 인원을 검출하여야하기 때문에 더 빠른 예측 모델이 필요했다.

우리는 YOLOv3와 Darknet을 이용하여 실시간 Detection을 사용했다. 이미지 하나에 수천개의 계층이 필요한 R-CNN과 달리 하나의 신경망으로 평가하여 예측한다. 이는 R-CNN보다 약 1,000배 빠르다.[1]

2. 딥러닝 기반의 객체 추적 알고리즘

다양한 Object Tracking 알고리즘 중 대표적인 2가지는 DeepSort Algorithm과 Re-Identification Algorithm이 있다.

등속 운동과 선형 관측 모델을 사용한 Deepsort Algorithm은 이미지 공간에서 Kalman Filtering을 수행하고 Bounding box overlap을 측정하는 association metric과 함께 Hungarian method를 사용하여 객체를 추적한다.



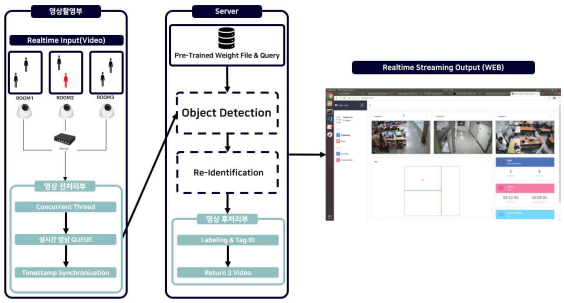
<프레임 별 Deep Sort Algorithm Tracking>

Re-Identification Algorithm[2]-[3]은 특정 객체를 이전의 다른 시간대나 공간, 상황에서 찾았던 목표객체와 새로운 객체간의 특징 비교를 통해 동일한지 검출한다. 또한 성능을 증가시키기 위한 방식으로 Style-Transfer를 적용하여 영상의 조명, 각도, 주변 환경 등이 변화하여도 검출 정확도를 유지 할 수 있도록 한다.

<Re-Identification Style-Transfer>

Vector Based Filtering을 수행하는 Deepsort Algorithm은 주로 도로에 설치된 CCTV등에 활용하는 것이 좋다. 이와 다르게 Re-Identification은 목표객체와 새로운 객체를 Dataset 기반의 이미지로 구분하기 때문에 이번 실험처럼 다양한 카메라 각도에서 실내 사람의 이동 동선을 파악하기에 적합하다.

III. 시스템 모델/구현



<전체 시스템 모델 흐름도>

모델을 설계함에 있어 각각의 연결된 3개의 방에 POE 카메라를 각각 설치하였고, 카메라 영상을 POE 허브를 통해 각각 5555, 5566, 5577 포트에서 받는다. 포트의 영상을 동기화 하여 Streaming 하기 위해 Timestamp를 지정하여 영상간의 Dealy를 최소화 하였다.

3-1)Video

```
def gen(camera_stream, feed_type, device):
    unique_name = (feed_type, device)
    num_frames = 0
    total_time = 0
    while True:
        time_start = time.time()
        now = time.localtime()
        cam_id, frame = camera_stream.get_frame(unique_name)
        if frame is None:
            break
        num_frames += 1
        time_now = time.time()
        total_time += time_now - time_start
        fps = num_frames / total_time
        cv2.putText(frame, cam_id, (int(20), int(20 * 5e-3 * frame.shape[0])), 0, 2e-3 * frame.shape[0], (255, 255, 255), 2)
        cv2.putText(frame, "Time : %04d/%02d/%02d %02d:%02d:%02d" %(now.tm_year,now.tm_mon,now.tm_mday,now.tm_hour,now.tm_min,now.tm_sec),
        (int(20), int(40 * 5e-3 * frame.shape[0])), 0, 2e-3 * frame.shape[0], (255, 255, 255), 2)
        if feed_type == 'yolo':
            cv2.putText(frame, "FPS: %.2f" % fps, (int(20), int(60 * 5e-3 * frame.shape[0])), 0, 2e-3 * frame.shape[0],(255, 255, 255), 2)
        frame = cv2.imencode('.jpg', frame)[1].tobytes()  # Remove this line for test camera
        yield (b'--frame¥r¥n'
               b'Content-Type: image/jpeg¥r¥n¥r¥n' + frame + b'¥r¥n')

@app.route('/video_feed/<feed_type>/<device>')
def video_feed(feed_type, device):
    """Video streaming route. Put this in the src attribute of an img tag."""
    port_list = (5555, 5566, 5577)
    if feed_type == 'camera':
        camera_stream = import_module('camera_server').Camera
        return Response(gen(camera_stream=camera_stream(feed_type, device, port_list), feed_type=feed_type, device=device),mimetype='multipart/x-mixed-replace; boundary=frame')
```

영상 전처리가 끝난 후 Pre-Train된 Weight파일로 각 영상의 사람 객체를 Detection 한다. 이후 추적하려는 Query Image와 함께 Detection된 사람 객체에 ID를 부여하여 추적한다. 이 객체가 다른 카메라로 이동

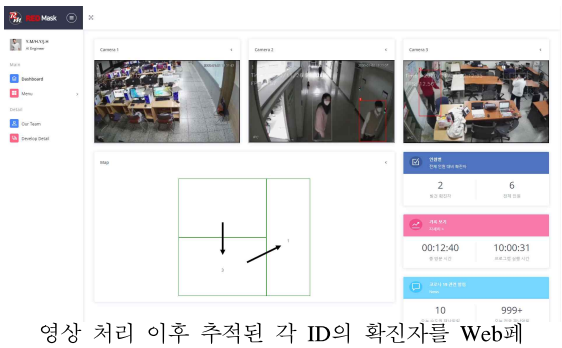
하였을 때, Re-Identification을 사용하여 동일인물인지 판단한다. 우리는 이 과정에서 Market 1501 Dataset을 이용하여 학습시켰다.

3-2) Detection

```
def detect(cfg, data,
           weights,
           images='data/samples',  # input folder
           output='output',  # output folder
           fourcc='mp4v',  # video codec
           img_size=416,
           conf_thres=0.5,
           nms_thres=0.5,
           dist_thres=1.0,
           save_txt=False,
           save_images=True):

    # Initialize
    device = torch_utils.select_device(force_cpu=False)
    torch.backends.cudnn.benchmark = False  # set False for reproducible results
    if os.path.exists(output):
        shutil.rmtree(output)  # delete output folder
    os.makedirs(output)  # make new output folder

    query_loader, num_query = make_data_loader(reidCfg)
    reidModel = build_model(reidCfg, num_classes=10126)
    reidModel.load_param(reidCfg.TEST.WEIGHT)
    reidModel.to(device).eval()

    query_feats = []
    query_pids  = []

    for i, batch in enumerate(query_loader):
        with torch.no_grad():
            img, pid, camid = batch
            img = img.to(device)
            feat = reidModel(img)
            query_feats.append(feat)
            query_pids.extend(np.asarray(pid))
    print(query_feats)
    query_feats = torch.cat(query_feats, dim=0)  # torch.Size([2, 2048])
    print("The query feature is normalized")
    query_feats = torch.nn.functional.normalize(query_feats, dim=1, p=2)
```

각각의 객체가 판단된 영상을 토대로 객체에 ID Tag와 Labeling을 부여하여 새로운 영상을 Web페이지에 Python Flask Framework로 구현한다.

3-3) UX/UI



영상 처리 이후 추적된 각 ID의 확진자를 Web페이지 아래 MAP에 현재 위치를 나타내고 Room 1 -> Room2로 넘어갈 때 JQuery를 이용해 화살표처리를 진행하였다.

우측 메뉴를 통해 현재 영상 내에 확진자의 수와 전체 검출된 사람의 수를 산정하였다. 또한 전체 영상 촬영 시간, 관련 코로나19 소식등을 나열하여 User Experience 향상을 추구했다.

IV. 실험 및 결과 고찰

본 논문에서는 딥러닝 Detection 모델을 학습하기 위해 YOLOv3와 Darknet 라이브러리를 사용하였다. 또한 Tracking을 위해서는 Re-Identification을 사용했다. 추가로 더 나은 학습 결과를 위해 CPU가 아닌 GPU 환경에서 학습을 진행하였고, NVIDA GTX 2070 SUPER CUDA를 통해 학습 장치에 I7-8700, DDR4 16GB RAM을 사용하였다.

본 연구를 통해 Object Detection과 Re-Identification을 활용하여 객체를 검출하였고, 환경이 변했을 때, 객체의 ID를 놓치지 않은 채로 연속해서 인식에 성공하였다. 본 연구 결과를 통해 추후에 코로나 바이러스 실내 확진자 추적에 도움이 되었으면 한다.

참 고 문 헌

1. Joseph Redmon, Ali Farhadi, (25 Dec 2016) "YOLO9000: Better, Faster, Stronger"
2. T. D'Orazio and G. Cicirelli, "People re-identification and tracking from multiple cameras: a review," 2012 19th IEEE International Conference on Image Processing.
3. Zhong, Zhun, et al. "Camera style adaptation for person re-identification." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018.
4. Murray, Samuel. "Real-time multiple object tracking-A study on the importance of speed." arXiv preprint arXiv:1709.03572 (2017).
5. Milan, Anton, et al. "MOT16: A benchmark for multi-object tracking." arXiv preprint arXiv:1603.00831 (2016)

이름 김 용 민 1995년 12월 31일생
2020년 12월 : 한국 항공대학교
소프트웨어 전공
(공학사)

이름 김 혜 영 1997 년 9월 15일생
2020년 12월 : 한국 항공대학교
소프트웨어 공학전공
(공학사)

이름 서 준 호 1995년 3월 4일생
2020년 12월 : 한국 항공대학교
소프트웨어 공학전공
(공학사)