2022 KBS 미디어창의기술전 수상작 소개

AI 기술을 적용한 사물 인식 자동추적 카메라시스템 제작

박승화 KBS 중계기술국 사원

들어가며

KBS 기술본부 내 기술운영부 주관으로 매년 개최하는 'KBS 미디어창의기술전' 본심사가 지난 2022년 12월 2일(금) 본관 6층 대회의실에서 진행되었다. 29번째를 맞이한 미디어창의기술전은 연구형식(제작/송출/송신 운용방식개선을 위한 시스템/장비 개발, 멀티 플랫폼 서비스 개발), 그리고 자유형식(미디어 환경변화에 능동적으로 대처할 수 있는 창의적인 고안, 논문, 기획서 등)의 2가지 형식으로 응모 영역을 확대하였다.

미디어창의기술전은 KBS 전 직원을 대상으로 기획안 접수를 받은 후 예비 심사와 본심사를 통한 공정한 심사를 통해 수상작이 결정된다. 또한 본심사는 사내 공청과 기술인협회 유튜브 라이브를 통한 실시간 중계를 하였다. 이번 미디어창의기술전은 대상(연구개발형식) 1팀, 우수상(연구개발형식) 1팀, 노력상(연구개발형식) 2팀, 노력상(자유형식) 1팀이 수상의 영광을 안게 되었다.

수상내역	출품명	출품자
노력상(자유형식)	AI 기술을 활용한 흑백 아카이브 영상 컬러 복원 방안	미디어기술연구부 이용건 외 1人
노력상 (연구개발형식)	AI OCR을 활용한 '재난정보' 자막 체크 프로그램	재난미디어센터 김성민 외 3人
	QR 코드를 활용한 기자재 관리 애플리케이션 큐위드(QWITH)	진주방송국 김재원 외 2人
우수상(연구개발형식)	DTV 중계기 PA Power Supply 조립 키트 개발	목포방송국 박관수 외 2人
대상(연구개발형식)	AI 기술을 적용한 사물 인식 자동추적 카메라시스템	중계기술국 박승화 외 3人

중계방송 현장은 다른 제작 환경과 달리 예상치 못한 환경적 제약과 변수가 다수 존재합니다. 현재 많은 방송 제작 현장에서 사용 중인 PTZ Remote 시스템의 경우 필연적으로 조작을 위한 인력이 필요한 반면, 이번에 자체 개발한 'AI 기술을 적용한 사물 인식 자동추적 카메라 시스템'의 경우에는 딥러닝을 통하여 원하는 객체를 따라 자동으로 동작할 수 있도록 하였습니다. 2022년 6월에 있었던 누리호 2차 발사 POOL 제작 당시 빠른 누리호 발사 장면에 적용하였으며, 향후에는 개선 과정을 거쳐 다양한 스포츠 중계방송에도 적용해 보고자 합니다.

시스템 개요

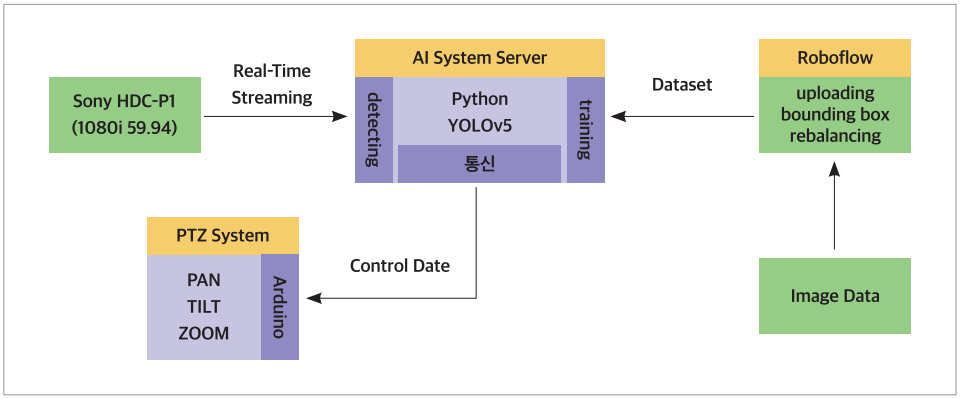

그림 1. AI 기술을 적용한 사물 인식 자동추적 카메라 시스템 개요

컴퓨터 Vision의 하위 분야 중 하나로 디지털 이미지 또는 디지털 영상에서 유의미한 특정 객체를 감지하는 것을 Object Detection(객체 인식)이라고 합니다. 현재 Object Detection은 Image Retrieval(이미지 검색), Image Annotation(이미지 주석), Face Detection(얼굴 인식) 등 다양한 분야에 사용되고 있습니다. 이번에 개발한 시스템의 경우 Object Detection 구현을 위해 2020년 출시된 Yolo Model(You Only Look Once Version 5)을 적용하였습니다. 카메라로 촬영되는 영상의 다양한 이미지 중에서 유사한 이미지 또는 크기가 다른 이미지도 제거하는 과정을 거쳐 원하는 객체만을 따라 자동으로 PTZ(Pan/Tilt/Zoom) 시스템을 구동시키기 위해서는 AI 시스템이 객체 인식을 보다 정확하게 할 수 있도록 해주는 Deep Learning(딥러닝) 과정을 통해 사물의 인식률을 높이고자 하였습니다.

Dataset - 객체 인식을 위한 준비 과정

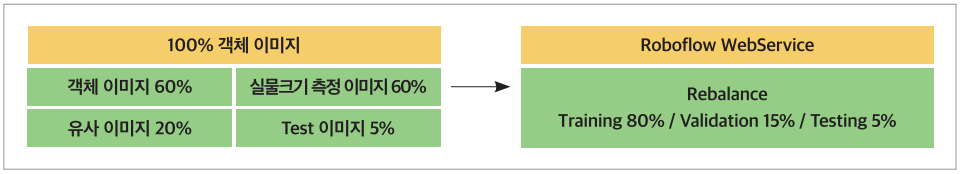

그림 2. Roboflow Webservice Dataset 생성 과정

Object Detection을 위해서는 원하는 Image Object를 AI가 인식할 수 있도록 훈련하는 과정이 필요합니다. 이 훈련 과정을 위해서는 Dataset이 필요하며, 이러한 Dataset 생성을 위해 무

료로 제공되는 Roboflow Web-Service를 활용하였습니다. Roboflow는 Computer Vision 기술을 이용해 다양한 Application에 활용할 수 있도록 Dataset을 생성해주는 무료 WebService입니다. 예를 들면, Roboflow에서는 객체 인식을 원하는 파란색 배구공 Image를 다양한 방식으로 수집하여 Roboflow Web-Service에 업로드 후, 각각의 Image에서 원하는 부분에 대한 Bounding Box를 설정하고, 사용자가 업로드한 이미지를 사용하여 Training, Validation, Testing이라는 Balancing 과정을 거쳐 YOLOv5에 적용 가능한 형태의 Dataset을 생성하게 됩니다. 아래의 그림은 Roboflow WebService에 Upload 한 이미지에 대한 Rebalancing 과정과 YOLOv5에 적용 가능한 형태의 Dataset을 Export 하는 과정을 보여줍니다.

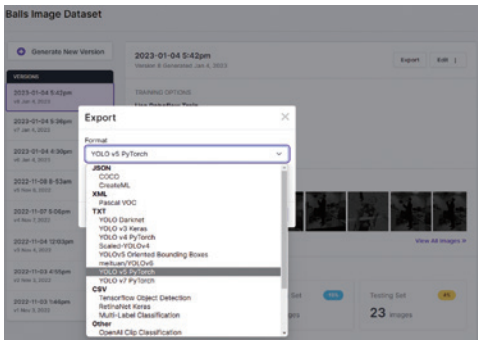

그림 3. Roboflow WebService Rebalancing

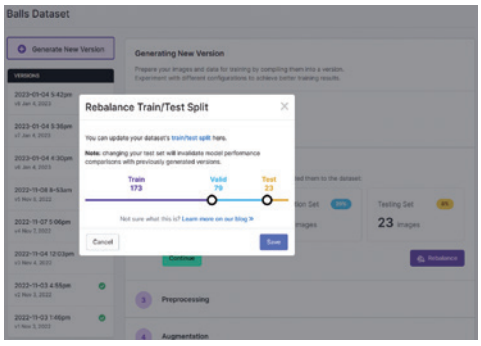

그림 4. Roboflow WebService Dataset Export

YOLOv5 Training (Deep Learning)

AI 시스템에서 원하는 객체의 인식률을 높이기 위해서는 지속적인 Image Training 과정이 필요합니다. 이를 위해 Roboflow에서 생성된 Dataset을 YOLOv5 Small Model에 적용 후 Epoch(Training 횟수) 값 조정을 통해서 AI 시스템의 객체 인식률을 높였습니다. YOLOv5에는 4가지 모델(Small, Medium, Large, XLarge(Extra-Large))이 존재하며 각각의 모델은 서로 다른 Configuration과 Parameter Size를 갖습니다. 각 모델의 Parameter 값은 AI 전체 시스템의 성능에 영향을 미치며 결국에는 전체 객체 인식 시스템의 정확도에 영향을 주게 됩니다. 이번에 개발한 시스템에서는 빠른 처리 속도에 중점을 두어 Small Model을 적용하였습니다.

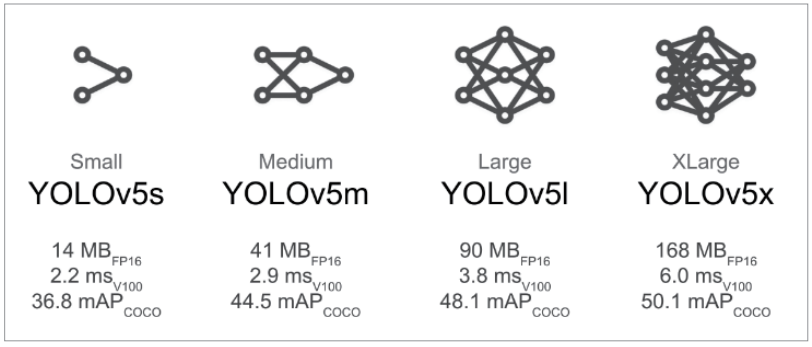

그림 5. YOLOv5의 4가지 모델

파라미터	내용
FP16	half floating-point precision
V100	inference time in milliseconds
mAP	mean average precision

다음의 그림은 YOLOv5 Small Model에 Epoch 값(training 횟수)을 50으로 설정하였을 경우 각각의 결과값을 보여줍니다.

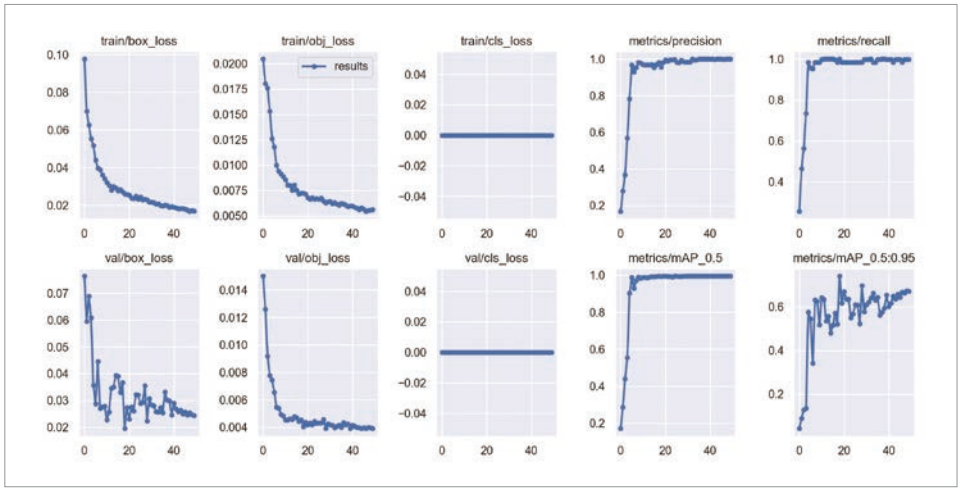

그림 6. Epoch가 50일 때 YOLOv5 Small Model 결과값

Python을 이용한 AI Object Detection

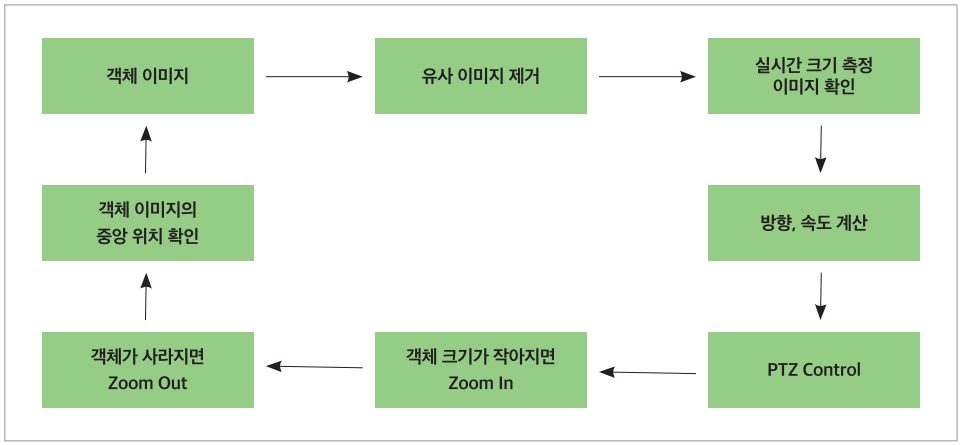

그림 7. AI 딥러닝 과정

YOLOv5를 통해 도출된 Image Training 결과값을 활용하여 Python으로 프로그래밍 된 AI System은 Real-Time Streaming(실시간 카메라 영상신호)에서 원하는 Object를 검출하게 됩니다. Python으로 구현한 AI 시스템은 다양한 Object 중 인식률이 높은 객체 이미지만을 추출하고 인식률이 낮은 이미지는 제외하게 됩니다. 예를 들어, 이번에 구현한 파란색 배구공을 원하는 객체로 인식시키기 위해서는 사람 손보다 작거나 사람 얼굴보다 큰 공도 유사 이미지로 인식하여 제거하도록 프로그래밍하였습니다. 이 외에도 이미지의 가로와 세로 비율, 이미지 색상(파란색), 객체 수 제한 등의 추가적인 필터링을 프로그래밍하여 원하는 객체를 카메라 영상 내에서 정확히 찾을 수 있도록 하였습니다. AI 시스템의 Training과 필터링 과정을 거쳐 보다 정확히 원하는 객체 이미지를 찾게 되면 Python으로 구현된 AI 시스템은 움직이는 벡터와 속도값을 산출해 PTZ 카메라 구동 시스템에 전달하게 됩니다.

다음은 AI System Server의 객체 인식과 PTZ 시스템 구동을 위한 Arduino와의 통신 Program과 실제 Image Detection이 이루어지는 화면을 보여줍니다.

```
        if save_img or save_crop or view_img:  # Add bbox to image
            c = int(cls)  # integer class
            # label = None if hide_labels else (names[c] if hide_conf else f'{names[c]} {conf:.2f}')
            # annotator.box_label(xyxy, label, color=colors(c, True))
            x2 = int((xyxy[0]+xyxy[2]) / 2)
            y2 = int((xyxy[1]+xyxy[3]) / 2)
            s = (xyxy[0]-xyxy[2])*(xyxy[1]-xyxy[3]) ------------------------------------------------------ 이미지 크기구하기
            r = (xyxy[0]-xyxy[2])/(xyxy[1]-xyxy[3]) ------------------------------------------------------ 이미지 가로 세로 비구하기
            if s > 200 and s < 14000 :
                if conf > 0.01 : ------------------------------------------------------------------------- 인식률 로 구분하기
                    if r > 0.85 and r < 1.15 : ----------------------------------------------------------- 가로세로 비로 물체 구분
                        #label = None if hide_labels else (names[c] if hide_conf else f'{names[c]} {conf:.2f}')
                        label = None if hide_labels else (names[c] if hide_conf else f'{"Ball"} {conf:.2f}')
                        #label= "Ball"+str(conf)
                        annotator.box_label(xyxy, label, color=colors(c, True))
                        cv2.circle(im0, (x2, y2), 4, (0, 255, 0), -1)
                        #text = "x: " + str(x2) + ", y: " + str(y2)+"r:"+ str(r)
                        #text = "x: " + str(x2) + ", y: "+ str(y2)
                        pasttime = round(time.process_time() )
                        zoomduration = pasttime - pasttimezoom
                        text = str(zoomduration)
                        cv2.putText(im0, text, (x2 - 10, y2 - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 255, 0), 2)
                        send_screen_pos(x2*2,int(y2*1.5))


                        if s< 2000 and zoomduration > 50 :
                            zoomin ------------------------------------------ 물체가 작거나 정지 해있으면 ZOOM IN
                            zoom0 = zoomin.encode('utf-8')
                            arduino.write(zoom0)

                        if s> 4000 and s<7000:
                           zoomstop = '0'  ------------------------------------ ZOOM STOP
                           zoom1 = zoomstop.encode('utf-8')
                           arduino.write(zoom1)

                        if s> 9000 and zoomduration > 30:
                           zoomout = 'M'   ------------------------------------ 크끼까 크거나 물체가 사라지면 ZOOM OUT
                           zoom2 = zoomout.encode('utf-8')
                           arduino.write(zoom2)


            # cv2.namedWindow('image')
          #  img = np.zeros((256, 256, 3), np.uint8)  # 행렬 생성, (가로, 세로, 채널(rgb)),bit)

        if save_crop:
            save_one_box(xyxy, imc, file=save_dir / 'crops' / names[c] / f'{p.stem}.jpg', BGR=True)

else :
    send_screen_pos(640, 360)
    pasttimezoom = round(time.process_time() )
    timeduration = pasttimezoom - pasttime
    if timeduration > 10 : ---------------------------------------- 객체 인식하지 안으면 ZOOM OUT
       zoomout = 'M'
       zoom2 = zoomout.encode('utf-8')
       arduino.write(zoom2)
       #print("time = '" + timeduration + "' ")


# Stream results
im0 = annotator.result()
if view_img:
    im1 = cv2.resize(im0, (1280, 720))# 화면 크기 조정
    cv2.rectangle(im1, (500, 260), (780,460), (0, 255, 255), 2)  ---------------------  객체 안정구역 표시
    cv2.rectangle(im1, (0, 0), (1280, 720), (0, 255, 255), 2)
    cv2.imshow("image", im1)
    cv2.waitKey(1)  # 1 millisecond
```

그림 8. Object Detection Program

```
def send_screen_pos(x, y):
        if x <= 500:
            data0='Q' ------------------------------------- PAN LEFT
        if x >= 780 :
            data0='W' ------------------------------------- PAN RIGHT
        if y <= 260 :
            data1='Y' ------------------------------------- TILT DOWN
        if y >= 460 :
            data1='U' ------------------------------------- TILT UP

        if x > 500 and x < 780: ---------------------------- PAN STOP
            data0 = 'E'
        if y > 260 and y < 460: ---------------------------- TILT STOP
            data1 = 'I'


      # ARDUINO에 COMMAND 송출
        dataindex = "X{0:d}Y{1:d}Z".format(x, y)
        print("output = '" + dataindex+ data0 + "' ") -------- PAN COMMAND PC모니터에 출력
        print("output = '" + dataindex+ data1 + "' ") -------- TILT COMMAND PC모니터에 출력
        result0 = data0.encode('utf-8')
        result1 = data1.encode('utf-8')
        arduino.write(result0) ------------------------------- PAN COMMAND 송출
        arduino.write(result1) -------------------------------  TILT COMMAND 송출
```

그림 9. Arduino와의 통신 Program

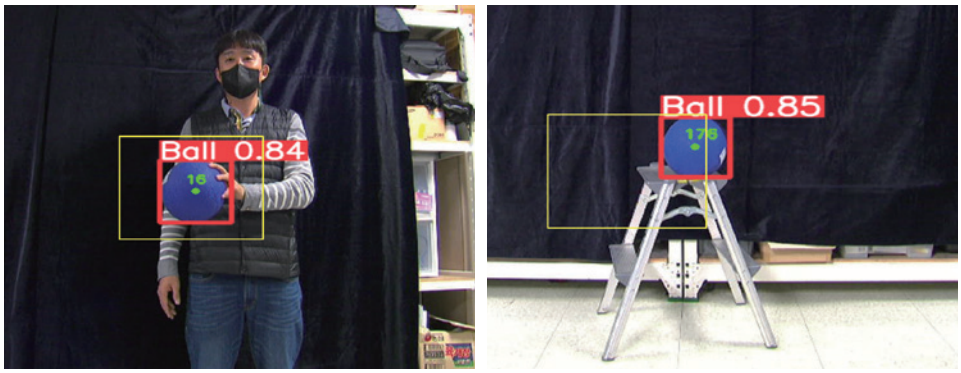

그림 10. 이미지 인식 결과

PTZ(Pan/Tilt/Zoom) 구동시스템

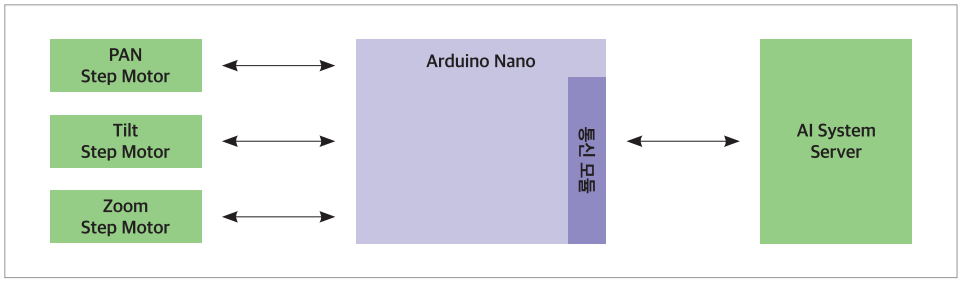

그림 11. PTZ(Pan/Tilt/Zoom) 시스템

실시간 카메라 영상신호에서 원하는 객체 이미지를 추출한 AI 시스템으로부터 산출된 객체의 벡터와 속도값은 PTZ 시스템에 전달되어 움직이는 물체를 자동으로 추적할 수 있도록 구현하였습니다. PTZ 구동시스템은 속도제어를 위해 Stepping Motor를 기반으로 제작하였으며, 감속기는 Worm Gear 50:1을 사용하였습니다. AI 시스템과의 통신은 아두이노를 기반으로 제작하였습니다.

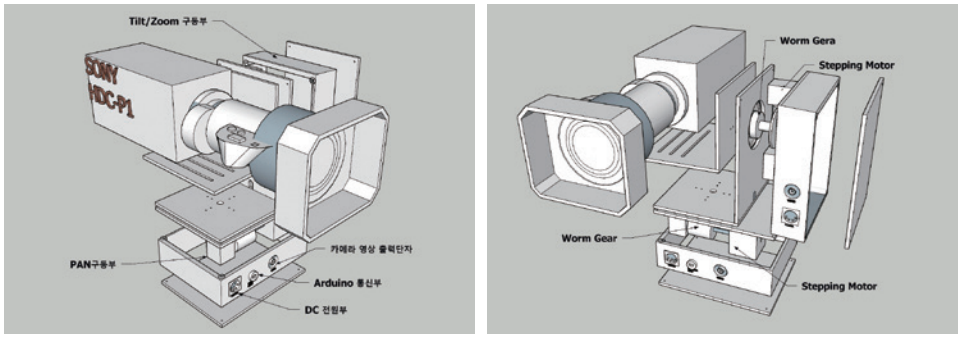

그림 12. PTZ 시스템 주요 부분 명칭

기본적인 동작 방식은 카메라 영상에서 원하는 객체가 없을 경우 추적을 위한 객체를 찾기 위해 Zoom Out 동작을 하며, Pan/Tilt 동작은 정지하게 됩니다. 만약 Zoom Out 된 영상에서 원하는 객체를 Detecting 하게 되면, Zoom/Pan/Tilt 동작을 통해 Dataset에 맞는 객체 사이즈를 카메라 영상의 중앙에 유지하기 위해 Zoom In 동작을 통해 최적화된 객체 크기를 유지합니다. 또한, 객체가 영상 내에서 2개가 감지되면 PTZ 동작을 멈추는 방식입니다. 이와 같은 방식으로 카메라 영상이 원하는 객체를 지속해서 추적하는 과정을 반복하게 됩니다.

그림 13. 구현된 사물 인식 자동추적 카메라시스템

마치며

AI 기술은 현재 초기 단계로 누구나 쉽게 접근하여 개발할 수 있는 상황이며, 객체 인식 부분 만큼이나 객체 인식 활용방안도 중요한 부분입니다. 예를 들면, 스포츠 분야에서 원하는 선수를 인식하고 추적하는 과정만큼이나 이 영상을 방송 제작에 어떻게 활용할지 역시 중요하게 생각해야 할 부분인 것입니다.

이번에 개발한 시스템을 지속해서 발전시킨다면 스포츠 제작 시 원하는 스포츠 선수의 등번호를 인식한 추적, 음악 프로그램에서 얼굴 및 의상의 색상을 통한 원하는 가수의 트래킹 등 다양한 분야에 적용이 가능할 것으로 생각합니다.