컴퓨터 비전 및
음성 인식을 통한
아동 유괴 예방

POLEAS (POSCO LEAD SAFE)

10기 A반 4조

김재원 | 김한결 | 성수호 | 이예랑 | 인지윤 | 진성희

목차

2.3 활용 기술

2.3.3 컴퓨터 비전(Computer Vision) - 모션 인식

1) 모션 인식 기술 소개

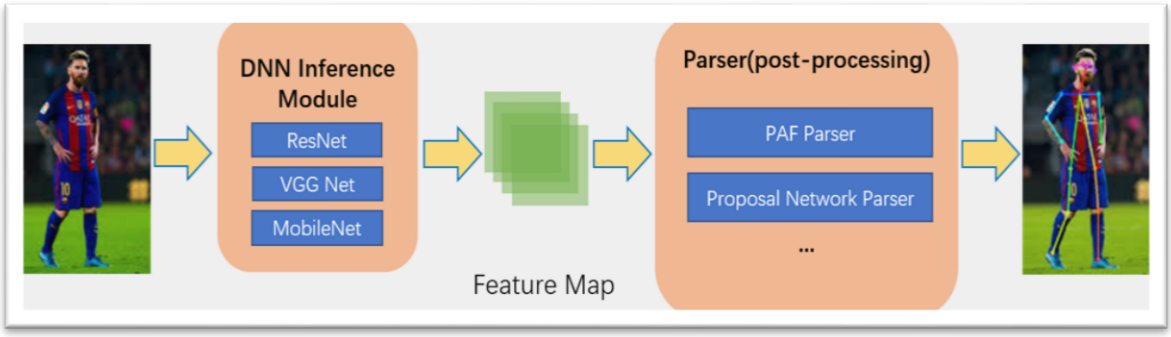
- 하이퍼포즈(HyperPose)란, 실상에서 사람의 모션을 인식하기 위한 c++언어 기반의 라이브러리 이다.

<그림 1 하이퍼포즈 features inference>

1.1) 하이퍼 포즈 특징

① Flexible training platform (유연한 학습 플랫폼)

하이퍼포즈는 모션인식 모델을 구축하기 위한 python API를 제공한다. 사용자는 직접 데이터를 증가시킬 수 있고, 학습을 위해 병렬 GPU 사용, DNN의 변경(예를 들어 ResNet에서 MobileNet으로)이 가능하다. 그래서 상황에 맞는 모델 구축이 가능하다.



<그림 2 데이터를 병렬 처리하여 여러 모듈과 파서를 동시에 처리>

② High-performance pose estimation (고성능 모션인식)

하이퍼포즈는 파이프라인 병렬처리, TensorRT[1]로 모델 추론, CPU/GPU의 혼합사용을 통해서 실시간 모션인식이 가능하게 한다. 모션인식으로 활용했던 가장 대중적인 텐서플로우의 tf-pose 혹은 openpose 보다 최소 4~10개 빠른 inference 속도를 보여준다.

Pose estimation 라이브러리간 성능비교

Back-Bone	TF-Pose	OpenPose	HyperPose
VGG	-	8 FPS	27.3 FPS
MobileNet	8.5 FPS	-	84.3 FPS
TinyVGG	-	-	124.9 FPS

<그림 3 하이퍼포즈와 기존 모션인식 라이브러리 성능비교>

1.2) 성능향상

① 배치 사이즈 조절

PAF[2](Part Association Field) 사후처리는 느리다. Batch 사이즈 조절은 PAF를 가속화시키지 않으며 속도 향상이 거의 없을 것이다. 하지만 Pose Proposal network[3]를 사용한다면 파이프라인의 처리속도에서 많은 향상을 보여줄 것이다. 배치사이즈는 8 ~ 128정도 조절하여 FPS를 늘릴 수 있다.

---

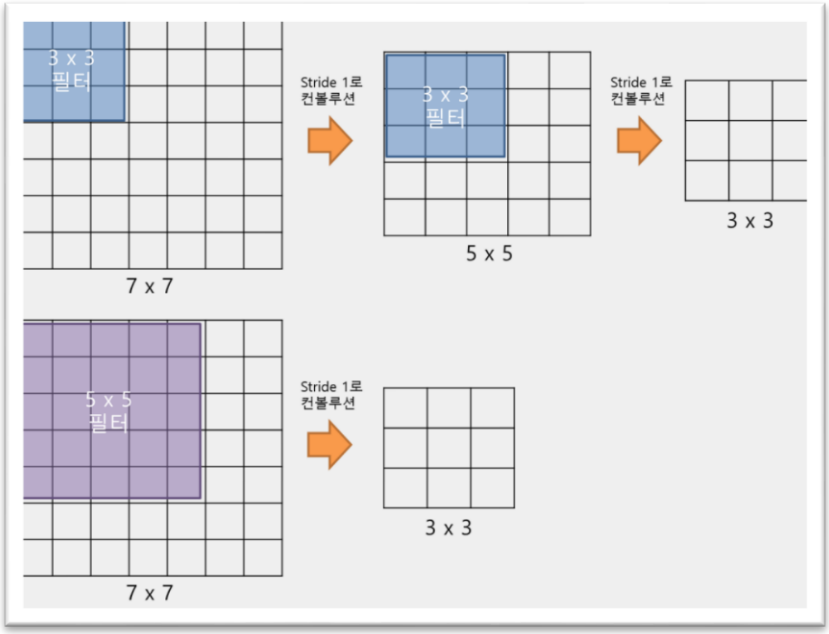
[1] 고성능 딥러닝을 위한 NVIDIA SDK로 CPU전용 플랫폼보다 최대 40배 가속

[2] 추출된 feature를 이미지나 영상에서 표시하는 네트워크

[3] PAF와 같은 사후처리 네트워크

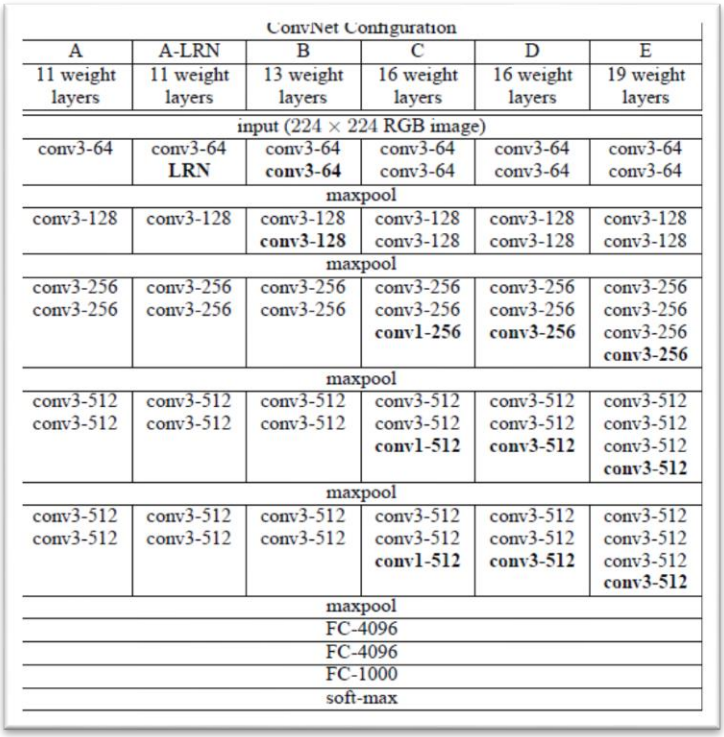
② TinyVGG

우선 VGG는 옥스포드 대학의 연구팀 VGG에 의해 개발된 모델로서, 2014년 이미지넷 이미지 인식 대회에서 준우승을 한 모델이다. 모델은 네트워크의 깊이를 깊게 만드는 것이 성능에 어떤 영향을 미치는지를 확인하고자 만들어졌으며, 필터커널의 사이즈는 가장 작은 3X3이다.



<그림 4 필터커널 사이즈와 깊이 비교>

VGG 연구팀은 논문에서 총 6개의 구조(A, A-LRN, B, C, D, E)를 만들어 성능을 비교했다. 여러 구조를 만든 이유는 기본적으로 깊이의 따른 성능 변화를 비교하기 위함이다. 이중 A 구조가 TinyVGG이다.



ConvNet Configuration

A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

<그림 5 VGG 구조 설명 표 [출처 : VGG연구팀 논문]>

VGG 연구팀은 AlexNet과 VGG-F, VGG-M, VGG-S에서 사용되던 Local Response Normalization(LRN)이 A 구조와 A-LRN 구조의 성능을 비교함으로 성능 향상에 별로 효과가 없다고 실험을 통해 확인했다. 그래서 더 깊은 B, C, D, E 구조에는 LRN을 적용하지 않는다고 논문에서 밝혔다. 또한 깊이가 11층, 13층, 16층, 19층으로 깊어지면서 분류 에러가 감소하는 것을 관찰했다. 하지만 범죄자 추적과정에서 실시간으로 식별해야하는 경우에 속도 부분을 무시할 수 없는 바 VGG의 층의 개수가 늘어날수록 학습속도가 현저히 떨어졌다. 그래서 본 프로그램에서는 에러가 다소 증가하더라도 실시간성을 높이기 위해 11층으로 구성된 TinyVGG를 선택했고 **<그림3>**과 같이 속도가 상당히 상승한 부분을 확인할 수 있었다.

1.3) 프로그램 구조도

<그림 6 프로그램 구조>

프로그램은 먼저 Train data를 실시간 웹캠으로 촬영하고 하이퍼 포즈의 CNN모델을 통해 인체를 특징짓는 18개의 점의 좌표를 추출한다. 이 추출된 좌표로 인체 비율을 계산하고 신원에 대한 정보를 labeling한다. 데이터가 쌓이면 파이썬으로 MLP모델을 사용하여 데이터를 학습시킨다.

그 다음 실시간 촬영을 하면서 Test데이터를 쌓고 Shared memory를 이용하여 실시간으로 저장된 test data를 python에 전송한다. 그리고 학습된 MLP모델로 전송 받은 데이터를 분류하여 촬영된 영상에서 Train data터에 있던 정보와 일치하는 신원을 찾아낸다.

1.4) 프로그램 구동 절차

1.4.1) 하이퍼포즈 세팅

하이퍼포즈를 구동하기 위한 전제 조건은 c++ 17 컴파일러와 Cmake 3.5버전 이상, OpenCV 3.2버전 이상, CUDA 10.2, CuDNN 7, TensorRT 7이 필요하다. 우리는 리눅스 우분투 18.04 LTS 운영체제를 사용했다.

*Ubuntu 18.04에 OpenCV 4.2.0 버전 설치하는 웹사이트

https://webnautes.tistory.com/1186

*Ubuntu 18.04에 NVIDIA Driver, CUDA 10.2, cuDNN 7 설치하는 웹사이트

https://leesh0523.tistory.com/entry/NDIVIA-Driver-CUDA-cuDNN-%EC%84%A4%EC%B9%98%ED%95%98%EA%B8%B0

여기서 주의사항은 CUDA 10.2와 cuDNN 7인지 정확히 확인해야한다.

*Ubuntu 18.04에 TensorRT 설치하기

https://eehoeskrap.tistory.com/302

전제조건을 다 깔았다면 선택 옵션인 gFlag를 깔고 하이퍼포즈를 build한다.

```
# >>> Build gFlags (Optional) from source. Install it if you want to run the examples.
$ wget https://github.com/gflags/gflags/archive/v2.2.2.zip    #gFlag 압축파일을 웹에서 받아온다.
$ unzip v2.2.2.zip #압축파일을 푸는 명령어이다.
$ cd gflags-2.2.2 #압축 푼 디렉토리에 이동하는 명령어이다.
$ mkdir build && cd build #빌드하기 위해 빌드 디렉토리를 만들고 이동해준다.
$ cmake .. -DCMAKE_BUILD_TYPE=Release -DBUILD_SHARED_LIBS=ON && make
#cmake 명령어로 빌드한다.
$ sudo make install #설치해준다.
```

```
# >>> Build HyperPose
$ git clone https://github.com/tensorlayer/hyperpose.git #하이퍼포즈 코드를 가져온다.
$ cd hyperpose #하이퍼포즈 디렉토리로 이동한다.
$ mkdir build && cd build #빌드를 하기 위해 빌드 디렉토리를 만들고 이동해준다.
$ cmake .. -DCMAKE_BUILD_TYPE=Release && make -j
```

위에 전제조건을 다 잘 깔았다면 맨 마지막 줄에서 lbnvinfer 등의 라이브러리가 없다고 에러메세지가 뜰 것이다. 이 라이브러리들은

http://developer.download.nvidia.com/compute/cuda/repos/ubuntu1804/x86_64/

http://developer.download.nvidia.com/compute/machine-learning/repos/ubuntu1604/x86_64/

이 두 사이트에서 deb파일을 다운받고 $ sudo dpkg -i (package_name).deb 명령어로 설치해주면 된다. 여기서 매우 중요한 주의 사항은 CUDA 10.2 버전으로 깔고 모든 패키지의 버전을 동일하게 맞춰줘야 한다.

예를 들면 libnvinfer-plugin-dev_7.0.0-1+cuda10.2_amd64.deb 이 패키지를 깔았다면 libnvonnxparsers7_7.0.0-1+cuda10.2_amd64.deb 이런 식으로 7.0.0-1 버전을 맞춰줘야 충돌이 일어나지 않는다. 그리고 libnvinfer를 깔 때는 libnvinfer와 libnvinfer-dev, libnvinfer-plugin, libnvinfer-plugin-dev를 모두 설치해야한다.

또한 /sbin/ldconfig.real: /usr/local/cuda-10.2/targets/x86_64-linux/lib/libcudnn_cnn_train.so.8 is not a symbolic link

이런 오류가 뜨면 libcudnn_cnn_train.so.8 파일을 지운 뒤 libcudnn_cnn_train.so를 target으로 심

볼릭 링크를 걸어 다시 만들어준다.

예시: ln -s libcudnn_cnn_train.so libcudnn_cnn_train.so.8

그리고 hyperpose 코드를 깃헙에서 가져오면 hyperpose/src 디렉토리에 NvInfer.h, NvInferRuntimeCommon.h, NvInferRuntime.h, NvUtils.h, NvUffParser.h, NvOnnxParser.h, NvOnnxConfig.h, NvInferVersion.h의 헤더파일을 추가한다. 헤더파일의 내용은 https://docs.nvidia.com/deeplearning/tensorrt/api/c_api/_nv_infer_8h_source.html 이곳에서 다운받을 수 있다.

모든 전제조건을 깔고 빌드까지 마쳤다면 하이퍼포즈를 구동해볼 수 있다. 빌드를 성공하면 하이퍼포즈에 있는 example을 실행해보자.

```
sudo apt -y install git cmake build-essential subversion libgflags-dev libopencv-dev
sh scripts / download-test-data.sh              # 예제를 위해 데이터를 설치합니다.
sh scripts / download-tinyvgg-model.sh          # tiny-vgg 모델을 설치합니다.
cmake .. -DCMAKE_BUILD_TYPE = RELEASE && make -j # 빌드 라이브러리 && 예제.
./example.operator_api_batched_images_paf       # 출력 이미지는 빌드 폴더에 있습니다.
```

실행이 다 되면 build 디렉토리에 output_x.png가 저장되어 확인해볼 수 있다.

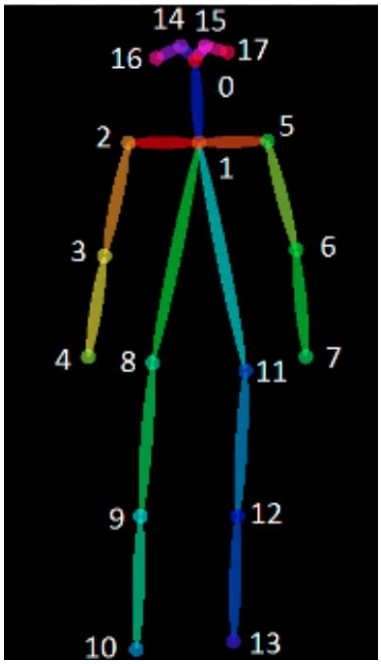
하이퍼포즈에서 실시간 웹캠으로 스켈레톤을 추출해내는 실행파일은 example.operator_api_imshow_paf이다. 컴퓨터에 카메라를 연결한 뒤 ./example.operator_api_imshow_paf -camera 라는 명령어로 실행할 수 있고 Ubuntu 18.04 환경에서는 FPS가 20~50정도 나온다. -camera명령어를 제거하면 DEFINE_string(input_video, "/home/piai/Downloads/20200819_231332.mp4", "The input video path."); 이 코드에 설정된 input video로 재생되니 참고하면 된다. 나의 cpp파일을 빌드하여 hyperpose로 사용하고 싶다면 hyperpose/examples/user_codes에 cpp파일을 넣고 다시 build하면 build 디렉토리에 실행파일이 생긴다.

1.4.2) 하이퍼포즈 스켈레톤 제거

본 프로젝트에서는 목 위쪽에 대한 스켈레톤은 필요 없었기 때문에 빼주었다. 빼는 방법은 operator_api_imshow_paf.example.cpp 파일에서

```
auto poses = parser.process(feature_maps.front()[0], feature_maps.front()[1]);
for (auto&& pose : poses) {
    pose.parts[0].has_value = false;
    pose.parts[14].has_value = false;
    pose.parts[15].has_value = false;
    pose.parts[16].has_value = false;
    pose.parts[17].has_value = false;
}
```

이 부분에 pose.parts[신체부위번호].has_value = false; 로 해주면 사라진다. 신체부위번호는 아래 사진에서 확인할 수 있다.



하이퍼 포즈 신체 부위 번호

1.4.3) 하이퍼포즈 신체 부위에 따른 정적 특징, 동적 특징 추출

다음으로 하이퍼 포즈 신체 부위 번호에 따른 정적 특징과 동적 특징을 추출해내는 함수를 구현하였다.

```
float euclid(float x1, float x2, float y1, float y2) {
    return sqrt(pow(x2-x1, 2) + pow(y2-y1, 2));
}


float cotheta(float x1, float x2, float x3, float y1, float y2, float y3) {
    return acos(((x1-x2)*(x1-x3)+(y1-y2)*(y1-y3))/((euclid(x1, x2, y1, y2)+0.00001)*(euclid(x1 ,x3, y1, y3)+0.00001)))*180/M_PI;
}
```

Euclid는 두 점 사이의 거리를 유클리디안 거리로 계산해주는 함수이고 cotheta는 세 점 사이의 각도를 리턴하는 함수이다.

```
float centerx = 0;
float centery = 0;
float shoulder, pelvis , spine, leg, crotch, left_knee, right_knee, left_pelvis, right_pelvis, stride = 0;
float shoulder_min, pelvis_min , spine_min, leg_min, crotch_max, left_knee_min, right_knee_min, left_pelvis_min, right_pelvis_min, stride_max = 0;
int num = 0;
string str1 ="";
```

글로벌 변수로 정적 특징과 동적 특징 변수들을 선언하고 초기화해준다. 이 변수들의 데이터 값을 받아오고 텍스트 파일에 데이터를 쌓는 과정은 //TensorRT Inference 주석 부분에 있으니 참고하면 된다.

1.4.4)추출한 신체 특징을 shared memory로 파이썬으로 전달

Shared memory와 MLP는 실시간으로 완벽하게 돌아가지 않아 구체적으로 작성하지 않겠다. 하지만 shared memory를 참고한 사이트와 코드를 teams에 올려놓을 예정이니 참고하면 된다.

*python – C++ shared memory 참고 웹사이트

https://speedr00t.tistory.com/484

operator_api_imshow_paf.example.cpp 파일에 shmwriter.cpp를 추가하고 MLP python 파일에 shmreader.py를 추가하여 데이터를 주고 받을 수 있다.

2) 걸음걸이 분석

보행자의 걸음걸이 분석을 통해 보행자의 신원을 확인하는 기술에 대한 근거는 2019년 조호필의 동국대 석사학위논문 『법보행 분석을 통한 범인식별 연구』를 참조하였다. 또한 걸음걸이를 분석함에 있어서 가장 중요한 분석방법에 대해 2017년 대한기계학회에 제출된 유현우와 권기연의 논문 『걸음걸이를 통한 연령 및 성별 분류 방법』을 참조하였다. 신원확인을 위한 분류 작업을 위해 신체 각 부위의 관절을 기준점으로 잡아 정적 특징(Static Feature)과 동적 특징(Dynamic Feature)에 관한 아이디어를 얻었다.

① 보행 분석과 분석 방법

– 분석 방법

조호필의 석사학위논문 '법보행 분석을 통한 범인식별 연구' 에서는 수사현장에서 cctv 영상분석을 가장 기본적인 수사로 정의하며, 영상의 화질이나 인물 식별이 곤란한 경우 등 예상하지 못한 변수가 많은 경우를 위해 걸음걸이 분석을 제시한다. 법보행 분석이란 주로 영상정보처리기기에 촬영된 인물의 걸음걸이 또는 그 특성을 분석하여, 대조 대상 인물과 비교하는 작업을 말한다. 또한 동일인물 식별이 불가한 경우 보행 분석을 통해 대상 인물의 병리적인 특성 등을 분석하여 대상 인물이 범행현장 주변의 인물이라면 주변 의료기관 등을 통해 치료이력 등을 확인으로 용의자 범위를 좁힐 수 있게 하는 등 수사과정에 도움을 줄 수 있다.

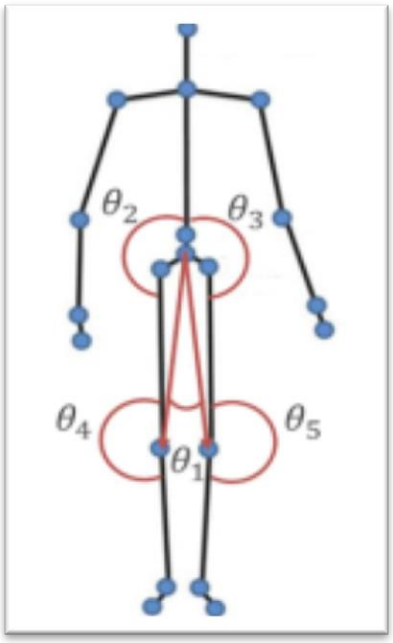
– 분석 방법

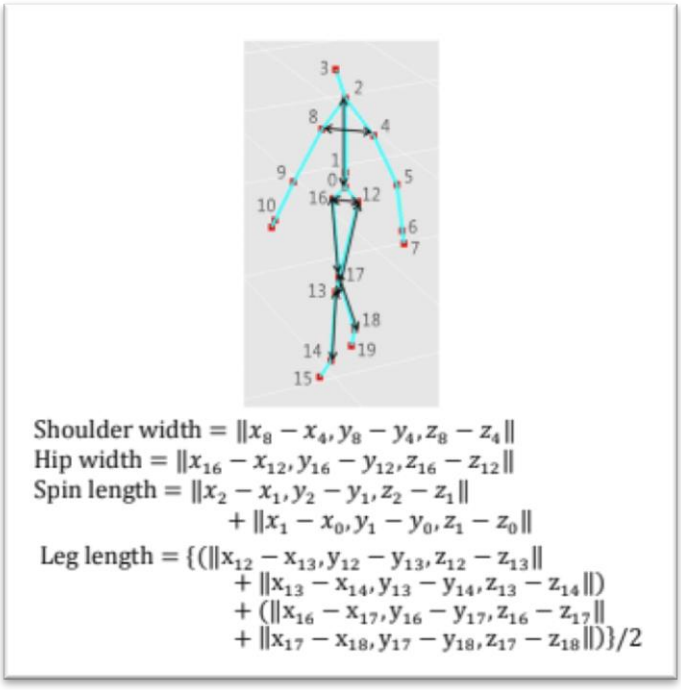
유현우와 권기연의 논문 『걸음걸이를 통한 연령 및 성별 분류 방법』에서는 RGB-D 센서로부터 획득된 골격 모델을 이용한 특징 추출 방법을 제안한다. 이 센서는 초당 30 프레임으로 실시간성 이 보장되며, 이 센서의 깊이 영상으로부터 신체의 총 20 개 관절의 3 차원 좌표를 추정하고 이를 걸음걸이 인식에 사용한다. 걸음걸이를 통해 연령 및 성별을

분류하기 위해 이전 연구를 통해 나타난 연령 및 성별에 따라 유의한 차이가 나는 특징들과 함께 추가적으로 추정된 신체 모델로부터 추출해낼 수 있는 특징을 제안하면 <table 1>과 같다. 여기서 정적 특징(static feature) 은 걸음걸이가 진행되어도 값이 변하지 않는 특징을 말하고, 동적 특징(dynamic feature)은 걸음 걸이가 진행함에 따라 값이 변하는 특징을 말한다. 이 논문에서 걸음걸이 데이터로부터 추출한 특징 벡터를 통한 분석을 제시한 부분은 아래와 같다.

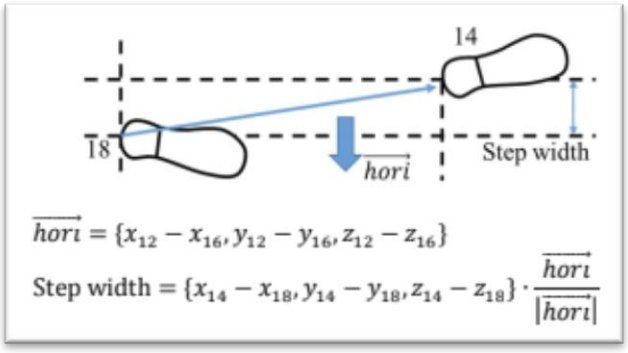
Table 1 Proposed feature

Static feature	Dynamic feature
Shoulder width	Step length
Hip width	Min/max of angle between knees θ_1
Spine length	Min/max of right hip joint angle θ_2
Leg length	Min/max of left hip joint angle θ_3
Step width	Min/max of right knee joint angle θ_4
	Min/max of left knee joint angle θ_5





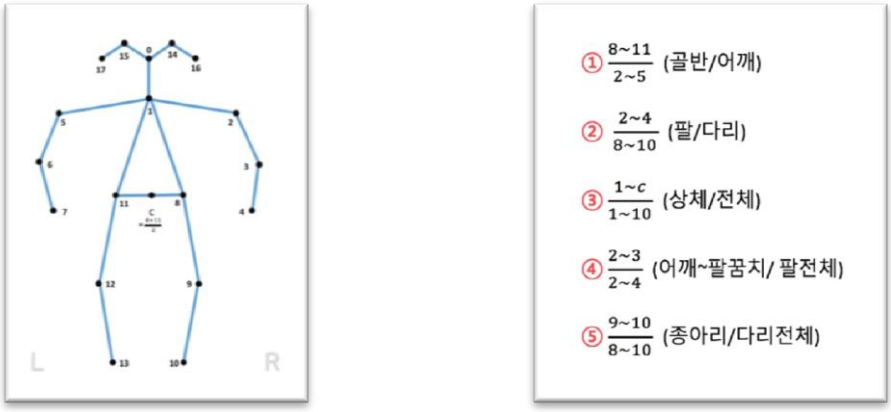
<그림 7 Angles of features>　　<그림 8 Calculation of static features>



<그림 9 Calculation of step width>

ⓐ Static Feature

하이퍼포즈를 통해 입력 받은 총 18개의 신체 좌표 중, 신체 부분간의 길이를 비율로 계산한 5개의 특징 값이다. 위 논문에서는 길이 부분만 feature로 삼지만 논문에서는 RGB-D 센서가 깊이 측정이 가능하여 3좌원 좌표로 출력하기 한다. 또한 카메라로부터 촬영된 인물의 거리에 따라 길이의 정도가 다르게 나타나기 때문에 본 프로젝트에서는 2차원 좌표의 단점을 보완하기 위해 static feature를 '동일한 방향성을 가진 신체부위 간의 길이 비율' 로 재정의하였다.



<그림 10 Static Feature>

위 **<그림 10>**과 같이 인물을 식별하기 위한 정적 특징 5가지 비율을 계산했다. 분모가 0이 되는 현상을 막기 위해 분모에 1e-10씩 더했다. 이 때 연속되지 않은 두 점 사이의 거리는 유클라디안 거리공식을 활용했다.

Ex) ex) 8~10 : 8~9사이 유클리디안 거리+ 9~10사이 유클리디안 거리

ⓑ Dynamic feature

하이퍼포즈를 통해 입력 받은 총 18 개의 신체 좌표 중, 신체부분의 각도와 보폭을 계산한 6 개의 특징 값이다.

<그림 11 Dynamic Feature>

위 논문에서 사용한 Dynamic Feature를 그대로 사용하였다. **<그림 11>**에서 $\theta 1 \sim 5$가 Dynamic Feature 이다. 각각의 각도를 구하기 위해 벡터의 내적 공식을 활용하였다.

4. 참고 문헌

1. 조호필, 2019, "법보행 분석(Forensic Gait Analysis)을 통한 범인식별 연구", 동국대 석사학위논문

2. Taiki Sekii(Konica Minolta, Inc.), 2018, “Pose Proposal Networks”, ECCV 2018

3. 유현우 외 1명, 2017, “걸음걸이 인식을 통한 연령 및 성별 분류 방법”, KCI

4. FaceNet – A Unified Embedding for Face Recognitioni and Clustering

5. MTCNN – Joint Face Detection and Alignment using Multi task Cascaded Convolutional Networks

5. 동료 평가

To. 성수호

From. 김재원

팀뿐만 아니라 반의 분위기를 항상 재밌게 이끌어 주었고 한편으로는 갈등이 있을 때 중간에서 조율도 해주셨던 A반의 플레이메이커였습니다. 꿈을 위해서 과감하게 이번 교육과정에 임하는 것을 보고 도전 정신이 무엇인지 느낄 수 있었습니다. (형의 목표 이루시길 바랍니다!)

From. 김한결

코딩하는 프로젝트 처음 하는데도 코딩을 할 때 논리가 잘 잡혀있는 것과 라이브러리를 수정하며 에러를 만났을 때 해결하는 모습을 보고 놀랐다. 대부분 벽을 만나면 포기하는데 어떻게든 자리를 지키고 끈기 있게 도전하는 모습에 감동했다. 앞으로 꾸준히 목표를 향해 정진해주면 좋겠고 3년뒤 미래의 모습이 기대가 된다.

From. 이예랑

팀의 분위기 메이커 역할을 잘 해줘서 너무 고마웠습니다. 팀원들을 위해 야식을 시켜주고 다 같이 힘내서 할 수 있도록 도와준 고마운 팀원입니다. 비전공자임에도 불구하고 항상 끝까지 남아서 열심히 하는 모습이 멋있었고, 앞으로 꿈꾸는 것을 금방 이룰 것 같은 실력을 가지고 있습니다. 마지막까지 모션인식 때문에 고생해줘서 고마웠습니다.

From. 인지윤

가장 연장자이지만 항상 친구처럼 우리를 대해주어서 너무 고마웠습니다. 사실 나이 차이 때문에 우리들과 친해지는 것이 어려울 수도 있을 텐데, 친근한 모습과 장난도 다 받아주며 분위기 메이커 역할을 톡톡 해냈습니다. 또한, 인공지능 프로젝트 때에는 처음에는 비전공자이니 잘 모르는 것이 당연한데, 시간이 지날수록 전공자같이 잘해주어 대단하다고 느꼈고 감사했습니다. 수호오빠, 존댓말 하는 게 이제는 어색하지만... 감사했습니다!ㅋㅋ

From. 진성희

인공지능 프로젝트를 같이 하면서 나 혼자 했다면 절대 불가능한 일을 같이 해낼 수 있어서 고마워. 새벽까지 의논하고 코드 수정해가면서 일 해보니까 아 이런 사람이 같이 일하고 싶은 동료구나라고 깨달았어. 체력, 능력, 열정, 연륜을 다 갖추고 있어서 많이 배웠고 명확한 꿈이 있다는 게 멋있었어. 축구 데이터 분석가에 대한 오빠의 꿈을 꼭 이루길 바랄게.