미래자동차로봇프로그래밍 캡스톤디자인 최종보고서

영상에서의 손 스켈레톤 추출을 활용한 키보드 자판 입력 인식

목 차

Ⅰ. 서 론

1. 연구 목표

본 연구는 다음 세 가지 연구 목표를 가지고 연구를 진행하였다[표 1].

번호	목표
1	손 스켈레톤 인식 오픈소스를 활용하여 영상 속 손 스켈레톤을 추출한다.
2	손으로 자판을 입력한 영상을 처리하여 손가락의 누름 여부를 판단한다.
3	눌러진 키를 판단하여 출력한다.

[표 1] 본 연구의 연구 목표.

먼저, 손 스켈레톤 인식 오픈소스를 활용하여 손 스켈레톤을 영상에서 추출한다. 본 연구에서는 Google의 Mediapipe 에서 제공하는 Hands 솔루션을 활용하게 된다. 둘째로, 찾아낸 스켈레톤의 좌표를 계산하여 손으로 키를 눌렀는지 판단하는 프로그램을 제작한다. 셋째로, 눌러진 키를 올바르게 출력한다. 추가로 분류성능평가지표인 정확도(Precision)와 재현율(Recall)을 구하여 본 연구 결과물의 성능을 평가한다.

2. 연구 배경

2. 1. 선행연구조사

손에 대한 센서 연구를 조사하였다. 손과 관련된 응용 분야는 로봇에서부터 영상 분석까지 많은 분야에 걸쳐서 연구되고 있다. 이 섹션에서는 본 연구에 영향을 준 두 논문에 대해 리뷰를 갖겠다. 첫 번째 논문은 'Sign-to-speech translation using machine-learing-assisted stretchable sensor arrays(Zhihao Zhou 외 15명)'이다. 웨어러블 센서 기기로 손의 신호를 감지하여 어떤 수화를 표현하려 하는지에 대한 연구를 진행하였다. 이 논문에서는 손의 신호를 감지하여 이를 실제 상황과 적용시키는 연구로 본 연구의 방향을 제시한 논문이었다. 두 번째 논문은 '키넥트를 이용한 종이건반 피아노 구현 연구(이정철, 김민성)'이다. 이 논문은 배경검출을 통해 종이 건반의 건반 패턴을 인식하고 손가락의 누름 여부를 OpenNI 라이브러리를 사용하여 판단하는 종이 피아노 시스템을 구축하였다. 키넥트 센서를 이용하여 피아노패턴검출-손영역검출-손가락끝점검출-누름 검출-MIDI제어로 이어지는 절차에 따라 종이 피아노에서 음을 인식하는 연구이므로 본 연구에서 어떤 식으로 알고리즘을 구축할지에 대한 방향성을 제시했다. 처음에는 depth를 감지하는 키넥트 센서를 사용했으나 키넥트 센서의 보급률이 낮아 그보다 많은 사용자를 유입할 수 있는 RGB Camera만을 이용하여 연구를 진행하였다. 초기에는 실제 피아노를 연주하는 영상으로 악보 추출을 제공하는 프로그램을 제작하고자 하였으나 구현 중 음악이라는 출력물의 정확도를 평가할 수 없다는 문제가 발견되었다. 이에 피아노보다 입력 여부가 분명하게 판단되는 컴퓨터 키보드 자판을 대상으로 하는 프로그램을 제작하여 성능을 평가할 수 있도록 하였다.

2. 2. 개발환경 선정

키넥트 센서의 Depth OpenCV와 OpenNI를 좀더 잘 활용하기 위해 관련 커뮤니티가 활발한 Ubuntu 환경에서 연구를 진행하고자 하였으나, Kinect와의 호환성과 확장성, 성능을 생각했을 때 Windows 환경에서 연구를 진행하였다. GPU가 존재하지 않는 컴퓨터 환경에서 개발해야 했기 때문에 필요한 파일을 구글 드라이브에 올리고, Google Colab에서 제공하는 GPU를 활용하여 Colab Notebook으로 개발하였다. Google Colab에는 기본 Python보다 더욱 다양한 모듈을 기본으로 제공하고 있어 개발 시 편리했다.

3. 시사점 및 제한점

새로운 기술을 이용하여 응용을 해보는 시도는 기술 생태계에서 중요하다. 본 연구는 최근에 공개된 Google의 MediaPipe를 이용하여 해당 기술의 확장가능성을 제시하고자 한다. 2020년 1월 Google Research는 경량화된 머신러닝 솔루션을 제공하는 프레임워크인 MediaPipe를 오픈소스로 공개하였다. 해당 솔루션을 이용해 어플리케이션을 제작함으로써 다양한 발전가능성을 제시할 수 있는 시사점이 있다. 또한 기존의 연구는 depth를 이용한 처리를 다루었다면 RGB camera을 사용함으로써 범용성을 높였다. 본 연구는 손의 스켈레톤을 이용하여 키보드 자판의 입력을 올바르게 인식함으로써 키보드 자판만 보이더라도 입력 내용을 유추할 수 있게 한다. 따라서 ATM기 등에서 발생할 수 있는 통장비밀번호 유출의 위험성을 제시한다.

본 연구에서 키보드를 인식하여 위치를 지정하는 것을 구현하였으나 손가락 누름 인식에 더욱 초점을 맞추기 위해 평가에는 사용되지 않았다. 후에 프로그램의 성능을 향상시켜 해당 알고리즘도 추가하면 보다 완성도 있는 프로그램을 제작할 수 있을 것이다.

Ⅱ. 이론적 배경

1. Google MediaPipe

Google에서 개발한 Mediapipe라는 모듈은 커스터마이즈 가능한 라이브 스트리밍 머신러닝 솔루션을 제공하는 Cross-platform 오픈소스이며, Android, iOS, Python, JavaScript에서 구동할 수 있다. 2020년 12월 18일 기준으로 'Face Detection', 'Face Mesh', 'Iris', 'Hands', 'Pose', 'Holistic', 'Hair Segmentation', 'Object Detection', 'Box Tracking', 'Instant Motion Tracking', 'Objectron', 'KNIFT' 등 12여가지의 다양한 솔루션을 제공한다. 본 연구에서는 파이썬 3.7환경에서 Mediapipe 모듈을 설치하여 'Hands'의 손 스켈레톤 추출 모듈을 사용한다. 해당 모듈은 손가락의 각 관절을 점으로 찍어 보여주며, 스켈레톤의 연결선의 특징을 통해 해당 손이 왼손인지 오른손인지 등도 분류해주는 솔루션이다.

[그림 1] Mediapipe Hands

2. Microsoft Azure Kinect DK

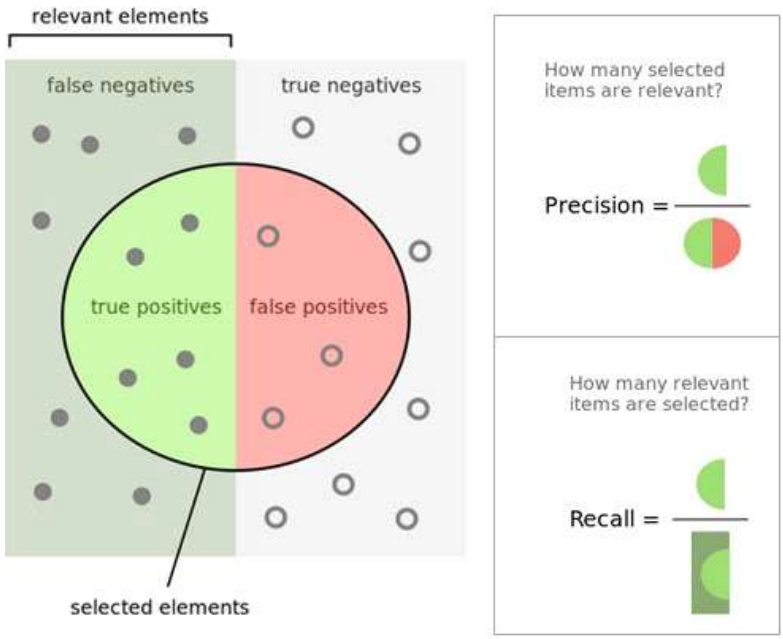
Azure Kinect DK는 Microsoft에서 출시한 키넥트 센서와 관련 소프트웨어 툴킷으로 Azure 소프트웨어와 호환하여 AI기술을 활용한 센싱을 제공한다. Depth camera, RGB camera, IMU sensor, Microphone을 탑재하고 있다. USB를 통해 컴퓨터 등의 디바이스에 직접 연결하여 사용한다.

3. 정확도(Precision)과 재현율(Recall)

분류문제에서, 정확도(Precision)과 재현율(Recall)로 분류성능을 측정할 수 있다. 센서의 인식 결과는 다음과 같이 4가지의 영역으로 구분될 수 있다. True positives, false positives, true negative, false negative가 그것이다. True positives의 뜻은 'A는 1이다'로 판단한 명제(Positive)가 참(True, 실제로 1)일 경우의 집합을 말한다. False positives는 'A는 1이다'로 판단한 명제(Positive)가 거짓(False, 실제로는 0)일 경우의 집합을 말한다. 마찬가지로 True negatives는 'A는 0이다'로 판단한 명제(Negative)가 참(True, 실제로 0)일 경우, False Positives는 'A는 0이다'로 판단한 명제(Negative)가 거짓(False, 실제로는 1)일 경우의 집합을 말한다. 전체 파티클을 이 4가지 영역으로 분류한 후 정확도(Precision)와 재현율(Recall)의 2가지 지표로 성능을 측정한다. 각각의 성능은 이 4가지 집합에 대한 확률로 정의된다.

$$\mathrm{Pr}ecision = \frac{n(TP)}{n(TP)+n(FP)}$$

$$Recall = \frac{n(TP)}{n(FN)+n(TP)}$$



[그림 3] Precision and Recall

Ⅲ. 연구 과정

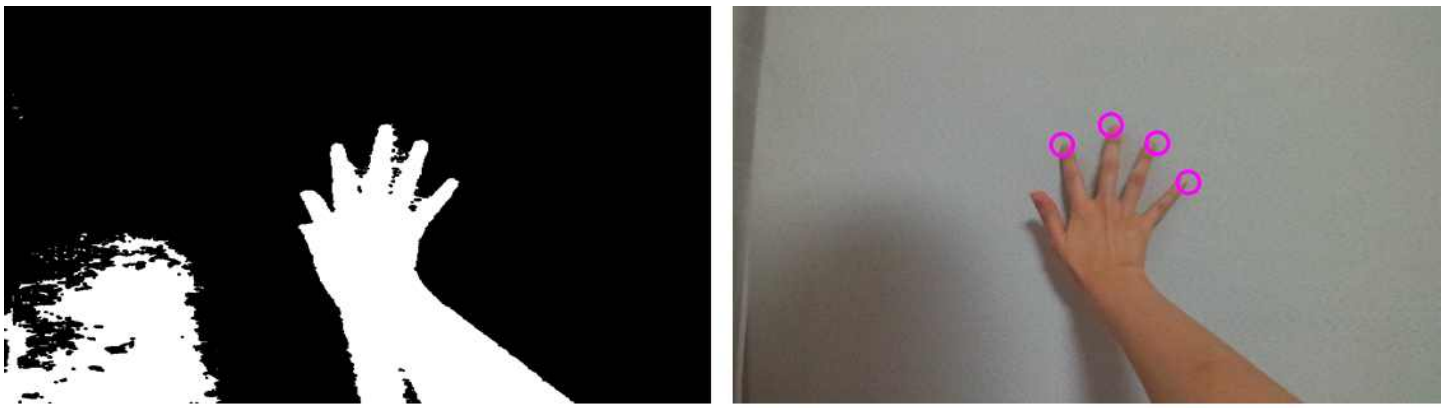
1. 개발 환경

본 연구를 진행하는데에 적용한 개발환경을 HW와 SW로 나누어 설명한다. HW는 CPU로 Intel(R) Core(TM) i5-8250U CPU@1.60SGHz, 1800MHz, 4 cores를 사용하였고 영상 촬영을 위해 Azure Kinect를 사용했다. SW로는 OS로 Microsoft Windows 10, 개발언어로 Python v3.7.8, 외부 모듈로 OpenCV-Python v3.4.11, Mediapipe를 사용했다, 또, Azure Kinect 구동을 위해 Azure Kinect Sensor SDK 내의 k4arecoder.exe (Azure Kiect recorder)와 k4aviewer.exe (Azure Kinect Viewer)를 이용했다.

2. 알고리즘 구현

2. 1. 알고리즘 구현 과정

초기에는 MediapPipe없이 OpenCV만을 이용하여 손 영역을 검출하는 방법에 대하여 조사하였다. 그 방법으로 HSV 색공간에서 손의 색을 건출하여 손을 인식하는 방법을 찾아보았다. 총 7개의 step으로 이루어진 이 알고리즘은 웹나우테스의 github 소스에서 찾을 수 있었다. STEP1은 영상에서 얼굴영역을 회색으로 덮어 없애는 단계, STEP2는 얼굴을 제거한 영상에서 스킨컬러를 감지하는 단계, STEP3는 morphologEx에서 어두운 영역을 삭제하는 단계, STEP4는 바이너리이미지에 컨투어를 감지하는 단계, STEP5는 가장 큰 영역(손)을 선정하여 빨간색 박스를 그리는 단계, STEP6은 convexHull을 계산하고 뾰족한 지점을 찾아 손가락 끝점의 후보군을 선정하는 단계, STEP는 이 점들을 이미지에 그리는 단계이다. 해당 알고리즘을 실행한 결과는 [그림 4] 과 같다.



[그림 4] 스킨컬러 검출을 통한 손가락 끝점 인식 결과. (a) 살구색 영역을 인식하여 하얗게 표시한 바이너리 이미지 (b) 검출한 손가락 끝점을 분홍색 원으로 표시한 결과.

살구색 영역을 검출하여 손가락의 끝점을 인식하는 방식의 특징은 Detect 방식의 알고리즘으로 매우 제한적인 상황에서만 작동한다는 단점이 있었다[그림 5]. 하지만 실제 손가락을 특정한 상태에서 손가락의 끝점을 계산(STEP5의 과정)하는 정밀도는 매우 높았다. 제한적인 상황에서 작동하여서 키보드 자판 위에서 실험할 시 검출할 수 없다는 치명적인 단점이 있어 다른 해결책을 찾았다.

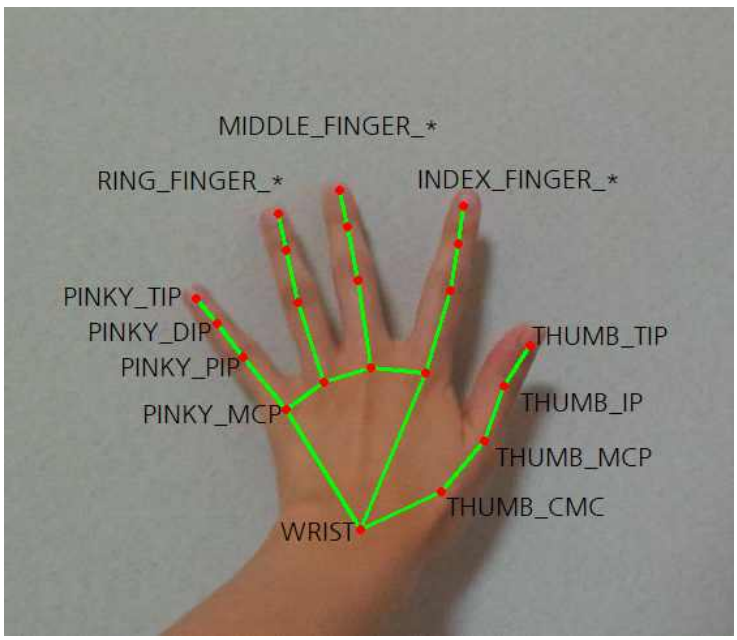
(a) 바이너리 이미지

(b) 결과 이미지

[그림 5] 실제 키보드 입력 영상에서의 스킨컬러 검출을 통한 손가락 끝점 인식 결과. (a) 살구색 영역을 인식하여 하얗게 표시한 바이너리 이미지 (b) 검출한 손가락 끝점을 분홍색 원으로 표시한 결과(결과없음).

Google의 MediaPipe는 연구과제에 대한 높은 적합성을 보이는 해결책이었다. 머신러닝을 이용해 손을 특정하는 것이기 때문에 이전 알고리즘보다 자유로운 환경에서 촬영한 영상에 대해서도 잘 동작하였다.

먼저 본격 적인 개발에 앞서, Mediapipe의 소스코드를 보았다. 손가락의 누름 여부를 판단하기 위해 관절의 각 좌표를 알아내는 것을 알아내야 했다. 역공학(Reverse Engineering)을 통해 소스코드의 생김을 알아보았더니 Landmark의 위치를 상수로 정의하고, 21개의 랜드마크들의 이음선으로 이어진 구조였다. 각 랜드마크 별 인덱스와 명칭은 [그림 6], [표 2]에 나타내었다.



[그림 6] 손의 관절마다 상수로 선언된 번호가 부여됨.

위치	명칭	번호
WRIST	WRIST	0
THUMB	THUMB_CMC	1
	THUMB_MCP	2
	THUMB_IP	3
	THUMB_TIP	4
INDEX_FINGER	INDEX_FINGER_MCP	5
	INDEX_FINGER_PIP	6
	INDEX_FINGER_DIP	7
	INDEX_FINGER_TIP	8
MIDDLE_FINGER	MIDDLE_FINGER_MCP	9
	MIDDLE_FINGER_PIP	10
	MIDDLE_FINGER_DIP	11
	MIDDLE_FINGER_TIP	12
RING_FINGER	RING_FINGER_MCP	13
	RING_FINGER_PIP	14
	RING_FINGER_DIP	15
	RING_FINGER_TIP	16
PINKY	PINKY_TIP	17
	PINKY_DIP	18
	PINKY_PIP	19
	PINKY_MCP	20

[표 2] 각 랜드마크별 값

MediaPipe에서는 drawing_utils.py에 있는 DrawingSpec:draw_landmarks 함수가 랜드마크들을

접근하고 있었다. draw_landmarks 함수는 image, landmark_list, connections, landmark_drawing_spec, connection_drawing_spec을 파라미터로 받아서 process함수를 통해 얻어진 landmark_list(자료형:dictionary)의 idx에 접근하여 해당 pixel에 손 스켈레톤(연결선은 초록색, 랜드마크는 빨간색)을 그리는 함수였다. 나는 이 코드를 landmark의 위치를 받아오는 부분와 해당 위치에 그림을 그리는 함수로 나눈 다음 landmark의 위치를 받아오는 부분의 코드만을 모으고 편집하여 5개의 함수를 정의하였다. 5개의 함수명은 initialize_landmark, draw_landmark2, get_landmark_length, coordinate_particular_landmark, repaint_landmark로 정했다. 각각을 설명하자면, initialize_landmark 함수는 만든 후 21개의 랜드마크들의 위치 리스트를 리턴하게 하는 함수이다. draw_landmark2는 기존의 draw_landmark 함수에서 위치에 접근하는 부분을 뺀 나머지 과정을 처리하는 함수다. get_landmark_length는 두 인덱스를 받아 해당하는 두 랜드마크들 사이의 거리값을 반환하는 함수다. coordinate_particular_landmark함수는 하나의 인덱스를 입력받아 해당 랜드마크의 이미지 좌표를 반환하는 함수다. repaint_landmark함수는 누름 여부를 알 수 있게 그림에 표시하기 위해 만든 함수로, 인덱스를 입력받아 해당 랜드마크의 점을 파란색으로 칠한다.

키보드의 위치를 반환하기 위해 letter라는 함수를 정의했다. 이 함수는 눌렸다고 인식한 지점의 x,y좌표를 저장하고 있는 pressed letter와 x,y 좌표를 저장하고 있는 item, 그리고 키보드의 킷값을 인덱스로 두고 그 위치 값을 값으로 두고 있는 딕셔너리 keyboard를 입력받아 최소 거리를 갖는 키 값을 pressed_letter에 append한다.

```
def letter(pressed_letter, item, keyboard):
  min = 9999
  dist = 0
  for i, value in keyboard.items():
    dist = math.sqrt((value[0]-item[0])**2 + (value[1]-item[1])**2)
    if dist < min:
      min = copy.deepcopy(dist)
      if i == '_1' or i == '_2' or i == '_3' or i == '_4':
        o_letter = '_'
      else: o_letter = i
  pressed_letter.append(o_letter)
  print('!!found!!',o_letter)
```

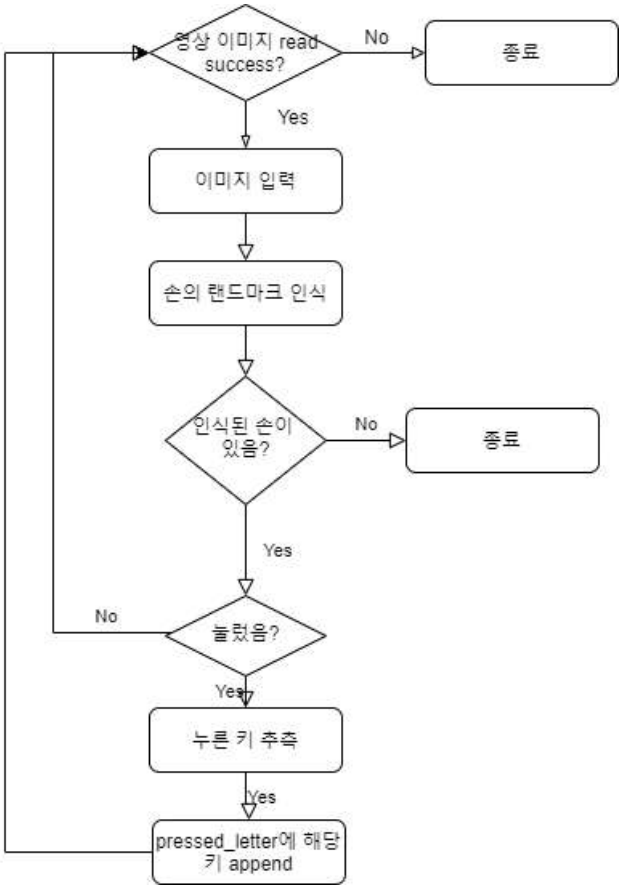
[그림 7] letter 코드

키보드를 검출하기 위해 총 STEP1은 왜곡된 키보드 이미지를 직사각형으로 반듯하게 만들어주는 단계이다. 이때 입력 이미지의 4개의 꼭짓점의 좌표와 출력 이미지의 4개의 꼭짓점의 좌표로 변환매트릭스를 만들어 변환하게 된다. STEP2는 컨투어를 찾는 단계이다. OpenCV의 findContours함수로 컨투어 후보군을 찾고, contourArea함수로 컨투어의 영역크기를 계산한다. STEP3에서는 찾은 컨투어의 크기가 키보드의 자판 하나의 크기 주변 값이면 박스를 생성한다. 모든 자판에 박스가 생성되지 않으므로, 특정한 박스 하나의 키값을 사용자에게 물어보아 대답한 결과를 토대로 모든 키보드 자판의 키의 위치를 찾아낸다. [그림 8]은 결과 이미지이다.

[그림 8] 키보드 자판을 인식한 결과. 3을 기준으로 각 키의 중간 좌표(파란색 점)을 저장함.

2. 2. 최종 알고리즘

최종 알고리즘은 [그림 8]로 나타내어진다. 먼저 동영상의 각 이미지에 대해 시간순으로 읽어들여서 손의 랜드마크를 인식한다. 그 다음 인식된 손에서 눌렀다는 판단(현재 집게 손가락의 길이가 이전 집게 손가락의 길이의 0.955보다 작은 경우)이 되면 키보드 자판에서 어느 키를 눌렀는지 계산하여 해당 킷값을 배열에 저장한다.



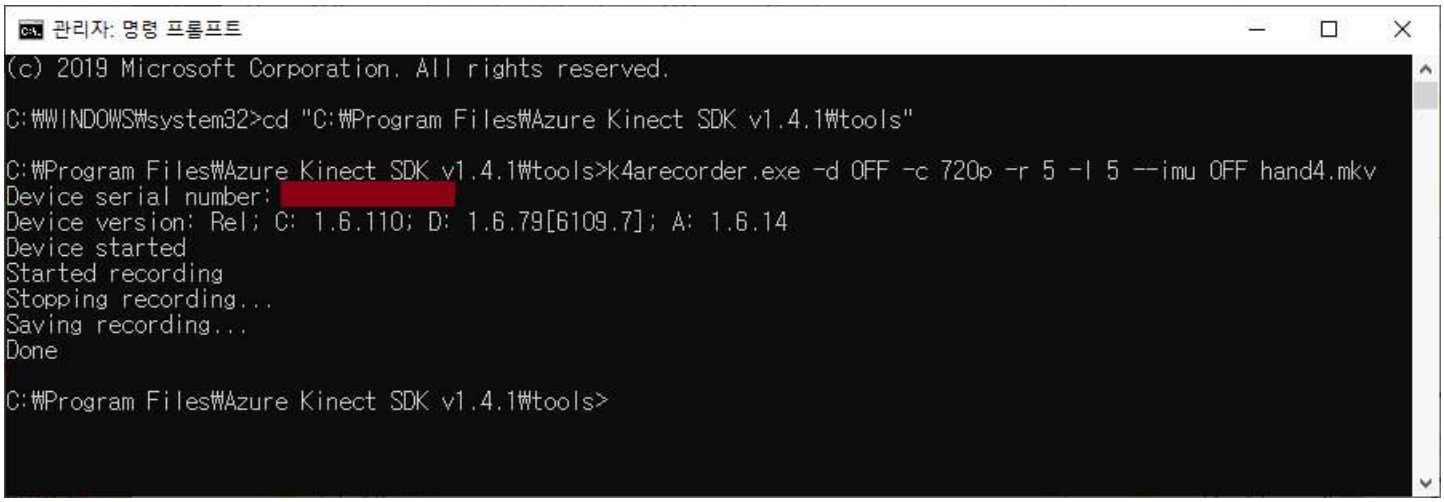
[그림 9] 최종 알고리즘

3. 실험 절차 및 주의사항

3.1 실험 절차

실험 절차는 다음과 같다.

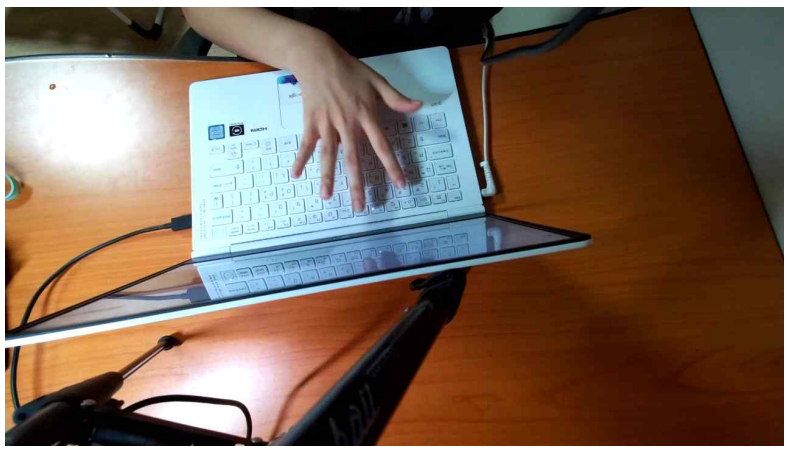
① 삼각대에 Azure Kinect를 적절한 각도와 각도로 설치한다. [그림 11] 참조
② Azure Kinect 연결후 Azure Kinect Viewer (k4aviewer)로 카메라의 앵글을 확인한다. [그림
③ Azure Kinect Recoder(k4arecorder)를 이용해 녹화를 시작하는 데, CMD에 k4arecoder.exe가 위치한 디렉토리로 이동 후, 다음의 명령어 k4arecorder.exe -d OFF -c 720p -r 5 -l [녹화하고자하는시간길이] --imu OFF [동영상이름].mkv 입력하여 녹화한다. [그림 10] 참조
④ 자판 위에 손을 펴고 기다리다가 CMD창에 'Started recoding' 텍스트가 생기면 집게 손가락으로 해당 문장을 입력한다. [그림 10] 참조
⑤ 촬영된 동영상파일(.mkv)를 구글 드라이브에 업로드한다.
⑥ 처리된 이미지를 보면서 [부록 1]과 같이 대상을 분류하여 정리한다.
⑦ 5개의 문장([표 3] 참조)으로 위 과정을 반복하여 최종결과를 도출한다.



[그림 10] 명령어 입력 예시

3. 2. 주의사항

① 카메라를 삼각대 등으로 고정하여 화면이 흔들리지 않아야 한다.
② 1초에 한 글자의 속도로 입력한다.
③ 오른손 집게 손가락을 움직여 입력한다.
④ 전체 손이 다 보여야 인식률이 높기 때문에 전체 손을 다 보이도록 한다. [그림 11] 참조.



[그림 11] 올바른 영상화면의 예. 손 전체가 드러남.

4. 분석 방법

총 5개의 소문자로 이루어진 문장([표 3])을 입력한 영상을 처리하여 프로그램의 Precision과

Recall을 구한다.

번호	문장
S1	the greatest glory in living lies not in never falling, but in rising every time we fall.
S2	the way to get started is to quit talking and begin doing.
S3	your time is limited, so don't waste it living someone else's life.
S4	don't be trapped by dogma – which is living with the results of other people's thinking.
S5	if life were predictable it would cease to be life, and be without flavor.

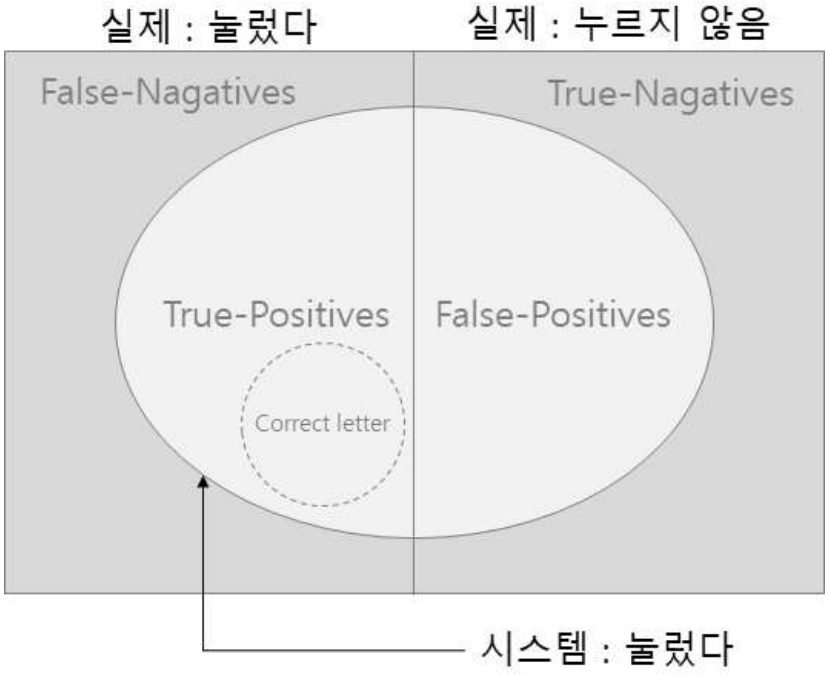
[표 3] 평가에 사용할 5개의 문장. 대/소문자 구별은 하지 않는다.

분석은 총 2개의 Test로 나누어 진행한다. Test 1에서는 눌렀다고 판단할 때 이미지가 보여지며, 그때마다 눌려졌다고 시스템에서 판단한 키 값이 출력된다. 평가자는 이 이미지와 키 값을 참고하여 [표 4]의 어떤 영역에 속하는지를 평가표를 작성한다. 또한 그 결과를 바탕으로 Precision과 Recall를 계산하여 Test1의 결과표를 작성한다. 작정할 때 약어를 사용하도록 한다. False-Negatives는 FN으로, True-Positives는 TP로, True-Negatives는 TN으로, False-Positives는 FP로 축약한다.

	Computer	Real
False-Negatives	X	O
True-Positives	O	O
True-Negatives	X	X
False-Positives	O	X

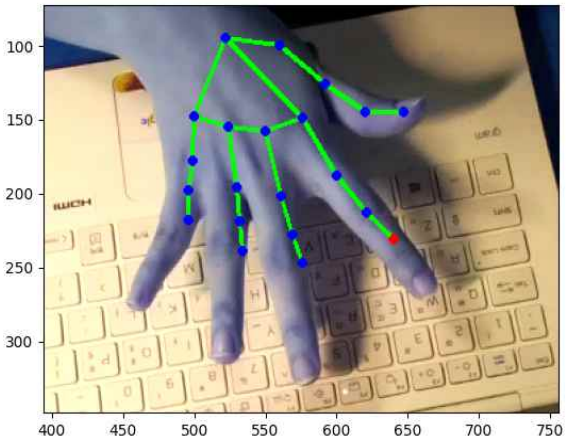
[표 4] 정확도와 재현율 계산을 위한 classification task별 의미. O/X는 누름 여부를 뜻함. O는 눌렸다는 것을 의미하고 X는 눌리지 않았다는 것을 의미함.

Test2는 눌렀다고 시스템이 올바르게 인식한 True Positives 중에 시스템이 눌렀다고 판단한 키 값과 실제 눌린 값을 비교하여 시스템이 정답을 맞췄을 경우 이를 correct로 또다시 영역을 나눈다. 그 후 Correct letter 개수에서 True Positives의 개수를 나눈 값을 Precision으로 계산한다.

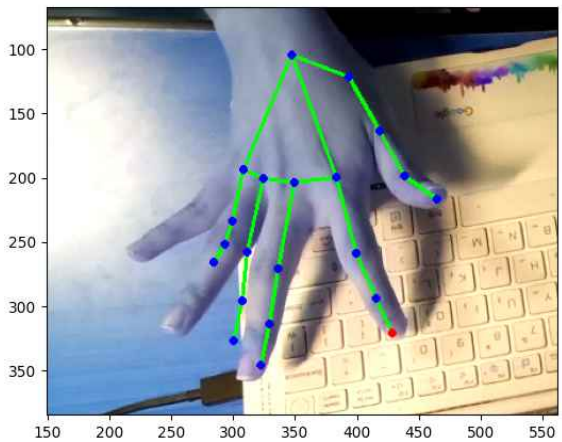


[그림 12] 분류 방법에 대한 그림.

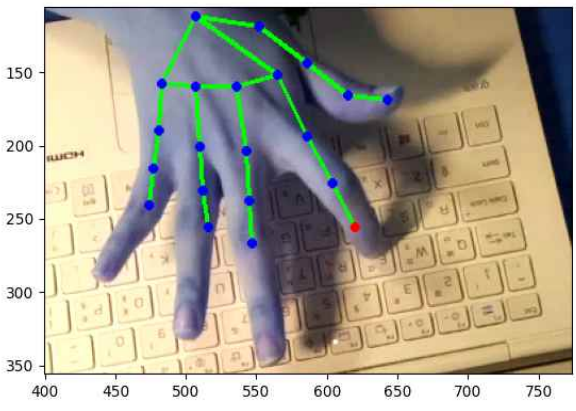
[그림 13]는 False-Positives의 예이다. 그림을 보면 손가락이 자판을 누르고 있지 않다는 것을 볼 수 있다. 따라서 자판을 눌렀다고 하였으나 실제로는 누르지 않았으므로 이 이미지는 False-Positives이다. [그림 15]와 [그림 14]는 모두 True-Positives에 속하는 이미지이지만, [그림 15] 은 실제로 P라는 값을 눌렀고 시스템에서도 P를 눌렀다고 인식하였다. 이는 Test2에서 Correct letter에 해당한다. [그림 14]는 실제로는 E를 눌렀으나 시스템에서는 D라고 인식하여 잘못 판단되었다.



[그림 13] False-Positive의 예. 손가락을 핀 상태로, 실제로는 자판을 누르지 않았지만, 프로그램에서 자판이 눌렸다고 잘못 판별됨.



[그림 14] True-Positive의 첫 번째 예. 실제로 손가락이 구부러져 자판을 눌렀고, 프로그램에서 자판이 눌렸다고 올바르게 판별됨. 실제 누른 키는 'P'이고 프로렸다고 올바르게 판별됨. 실제 누른 키는 'P'를 눌렀다고 올바르게 판단함.



[그림 15] True-Positive의 두 번째 예. 실제로 손가락이 구부러져 자판을 눌렀고, 프로그램에서 자판이 눌렸다고 올바르게 판별됨. 실제 누른 키는 'E'나, 프로그램에서 'D'를 눌렀다고 잘못 판단함.

Ⅳ. 평가 결과

1. 평가 결과

1. 1. Test 1

Test 1의 결과 최종 Precision이 0.55이고 Recall이 0.73으로 Precision보다 Recall의 성능이 훨씬 높다는 것을 알 수 있다.

Sentence	# of letters	T-P	T-N	F-P	F-N	Precision	Recall
S1	89	72	null	29	17	0.712871	0.808989
S2	58	44	null	22	14	0.666667	0.758621
S3	67	52	null	41	15	0.55914	0.776119
S4	88	66	null	76	22	0.464789	0.75
S5	74	43	null	63	32	0.40566	0.573333
TOTAL	376	277	null	231	100	0.545276	0.734748

[표 5] Test 1 결과표

1. 2. Test 2

Test 2의 결과 최종 Precision이 0.86이 나왔다. 이는 제대로 눌렸다고 인식하였을 때 정확도가 신뢰할 만하다는 뜻을 의미한다.

Sentence	T-P	# of Correct letters	Precision
S1	72	69	0.958333
S2	44	37	0.840909
S3	52	45	0.865385
S4	66	54	0.818182
S5	43	32	0.744186
TOTAL	277	237	0.855596

[표 6] Test 2 결과표

2. 평가 결과 분석

2. 1. 성능 하락의 요인

프로그램 실행에 있어 손인식의 정밀도에 평가가 영향을 크게 받는다는 것을 알 수 있다. Test 2에서는 키보드 자판 위치 추정에 대한 Precision을 구하였다. 이 키보드 자판 위치 추정의 값이 Test 1보다 비교적 일정하게 유지된다는 것으로 보아 카메라 화면 흔들림으로 인한 영향은 거의 없으므로 다른 요인인 손인식의 정밀도 문제에 대해 생각할 수 있다. Test 1을 보면 S1을 평가할 때보다 S5를 평가할 때의 정확도가 43%(1-(0.40566/0.712871)) 낮아졌다는 것을 볼 수 있다. 이 말은 실험의 결과가 외부환경 변화에 영향을 많이 받는다는 것으로 해석할 수 있다. 또한 평가 중 출력된 이미지를 보면 실제 손과 손의 스켈레톤 사이의 차이가 크다는 것을 느낄 수 있었다. 즉, 이 프로그램의 성능 저하는 처음에 손의 스켈레톤을 추출할 때 스켈레톤이 올바르게 추출되지 않아 발생한다.

또한, 조명환경을 제대로 통제하지 못하여 이런 결과가 발생했을 수 있다. 영상 촬영은 창가에

있는 책상에서 진행하였는데, 시간에 따라 조도가 달라져 실험 결과에 영향을 주었을 수도 있다.

2. 2. 개선 방안

프로그램의 성능을 높이기 위해 영상 초반에 손을 잘 인식할 수 있도록 기다렸다가 입력을 시작하면 더 좋은 성능을 얻을 수 있을 것으로 기대된다. 또한, 비실시간 영상처리가 아니라 Mediapipe의 특성을 잘 이용하는 실시간 모델을 만들어 개선하여 입력을 하는 도중에 손이 잘 인식되는지, 어떤 글자를 시스템이 인식하는지 확인하며 키보드 자판 입력 인식을 진행하면 실생활에 적용시킬 때 더 탁월해질 것이라 기대된다. 또한 성능을 높이는 과정을 반복한 뒤, 키보드 자판 입력 인식에서 더 나아가 피아노 등 여러 다른 분야에서 이 기술을 적용해보아도 좋을 것이다.

Ⅴ. 요약 및 결론

본 연구에서는 사용자가 입력한 영상의 손 스켈레톤을 추출하여 키보드 자판 누름 여부를 인식하여 주는 프로그램을 개발하였다. 손 스켈레톤 검출을 이용한 프로그램은 수화번역, 피아노 연주 등 활용범위가 넓다. 본 연구는 하나의 활용가능성을 제시하고, MediaPipe를 이용해 개발하여 MediaPipe의 활용 가능성을 검증하는 데 의의가 있었다.

참 고 문 헌

이정철, 김민성. "키넥트를 이용한 종이건반 피아노 구현 연구." *Journal of The Korea Society of Computer and Information*. 2012.

"Sign-to-Speech translation using machine-learning-assisted stretchable sensor arrays." *Nature Electronics*: 2020. 571-578.

Precision and recall. 2020. 12. 18일. <https://en.wikipedia.org/wiki/Precision_and_recall>.

Mediapipe. 2020. 12. 18.일. <https://mediapipe.dev/>.

"Hand Detection using OpenCV HSV Color Space." <*https://github.com/webnautes/nudapeu/blob/master/1378-2*>.

Azure Kinect DK. 2020. 12. 18일. <https://docs.microsoft.com/ko-kr/azure/kinect-dk/>.

부 록

[부록 1] 평가 결과표

S1

T	H	E	_	G	R	E	A	T	E	S	T	_	G	L	O	R	Y	_	I	N	_	L	I	V	I	N	G
G	*	E	B	G	R	E	A	T	E	S	T	G	*	*	*	R	Y	H	I	H	H	L	I	G	I	N	*
TP	FN	FP	FP	FP	TP	TP	TP	TP	TP	FP	TP	FP	FN	FN	FN	FP	TP	FP	TP	FP	FP	TP	TP	FP	TP	TP	FN
		E	_	G						S		_				R		*		N	G			*			
		TP	TP	TP						TP		TP				TP		FN		TP	FP			FN			
																					B						
																					FP						
																					-						
																					TP						
_	L	I	E	S	_	N	O	T	_	I	N	_	N	E	V	E	R	_	F	A	L	L	I	N	G	,	_
*	L	*	E	S	V	N	O	5	G	K	N	_	N	E	R	E	F	G	F	A	L	L	I	N	G	,	*
FN	TP	FN	TP	FP	FP	TP	TP	FP	FP	TP	TP	TP	TP	TP	FP	TP	FP	FP	FP	TP	TP	FP	TP	TP	TP	TP	FN
				S	_			T	_						V		F	_	F			L					
				TP	TP			TP	TP						TP		TP	TP	TP			TP						
B	U	T	_	I	N	_	R	I	S	I	N	G	_	E	V	E	R	Y	_	T	I	M	E	_	W	E	_
*	*	5	G	I	N	_	V	I	S	I	N	G	_	F	5	E	R	Y	_	T	*	M	E	T	*	G	_
FN	FN	FP	FP	TP	TP	TP	FP	TP	TP	TP	TP	TP	TP	FP	FP	TP	TP	TP	TP	TP	FN	TP	TP	FP	FN	TP	TP
		T	_				R							V	*									B			
		TP	TP				TP							FP	FN									FP			
														*										-			
														FN										TP			
F	A	L	L	.																							
F	A	L	L	*																							
TP	TP	TP	TP	FN																							

S2

T	H	E	_	W	A	Y	_	T	O	_	G	E	T	_	S	T	A	R	T	E	D	_	I	S	_	T	O
T	H	F	_	S	Z	Y	Y	T	*	G	G	E	T	_	C	T	*	R	T	*	C	_	I	S	_	T	O
TP	FP	FP	TP	FP	TP	TP	FP	TP	FN	FP	FP	FP	TP	TP	FP	TP	FN	FP	TP	FN	TP	TP	TP	FP	TP	TP	TP
	N	E		Z			H			_	G	D			X			*						S			
	TP	FP		TP			FP			TP	TP	TP			TP			FN						TP			
		E					*																				
		TP					FN																				
_	Q	U	I	T	_	T	A	L	K	I	N	G	_	A	N	D	_	B	E	G	I	N	_	D	O	I	N
O	V	U	*	*	_	F	R	L	K	H	N	G	_	V	N	F	_	G	E	F	F	N	_	D	*	*	N
TP	FP	TP	FN	FN	TP	FP	FP	TP	TP	FP	TP	TP	TP	FP	TP	FP	TP	FP	TP	FP	FP	TP	TP	TP	FN	FN	TP
	*					T	*			*				*		*		B		C	I						
	FN					TP	FN			FN				FN		FN		TP		TP	TP						
G	.																										
G	;																										
TP	FP																										
	.																										
	TP																										

S3

Y	O	U	R	_	T	I	M	E	_	I	S	_	L	I	M	I	T	E	D	,	_	S	O	_	D	O	N
Y TP	I FP	8 FP	R TP	_ TP	T TP	* FN	M TP	F FP	_ TP	E FP	X TP	_ TP	L TP	I TP	K FP	K FP	T TP	D FP	C TP	, TP	B FP	X FP	O TP	H FP	V FP	O TP	N TP
	O TP	8 FP						E FP		X FP					M TP	I TP		* FN			_ TP	X TP		_ TP	V FP		
		* FN						* FN		* FN															D TP		
‘	T	_	W	A	S	T	E	_	I	T	_	L	I	V	I	N	G	_	S	O	M	E	O	N	E	_	E
* FN	U FP	B FP	S TP	A FP	X FP	T TP	R FP	_ TP	I TP	T TP	V FP	P FP	I TP	G FP	I TP	N TP	G TP	_ TP	V FP	O TP	M TP	D FP	O TP	J FP	F FP	_ TP	D FP
	T TP	X FP		Z TP	* FN		D TP				_ TP	* FN		* FN					X TP			* FN		N TP	E FP		E FP
		* FN																							* FN		E TP
L	E	‘	S	_	L	I	F	E	.																		
L FP	E FP	‘ TP	U FP	_ TP	L TP	8 FP	E FP	; FP	. TP																		
L TP	E FP		R FP			I TP	* FN	* FN																			
	E FP		S FP																								
	E TP		* FN																								

S4

D	O	N	‘	T	_	B	E	_	T	R	A	P	P	E	D	_	B	Y	_	D	O	G	M	A	_	-	_
B FP	O TP	M FP	‘ TP	I FP	B FP	B TP	V FP	V FP	T TP	R TP	Z FP	P TP	P TP	U FP	* FN	* FN	B TP	U FP	H FP	D TP	O TP	H FP	M TP	N FP	_ FP	- TP	H FP
D TP		_ FP		T FP	_ FP		D FP	_ FP			Z TP			D FP				H TP	_ TP			B FP		_ FP	_ TP		_ FP
		_ TP		T TP	_ TP		X TP	_ FP						V FP								B TP		* FN			_ FP
								* FN						_ FP													_ FP
														* FN													_ TP
W	H	I	C	H	_	I	S	_	L	I	V	I	N	G	_	W	I	T	H	_	T	H	E	_	R	E	S
_ FP	H TP	N FP	V FP	N TP	J FP	O FP	D FP	_ FP	L TP	I TP	H FP	I TP	H FP	G FP	B FP	W FP	8 FP	T TP	H TP	H FP	5 FP	* FN	R FP	V FP	R FP	E TP	S FP
* FN		I TP	U FP		N FP	* FN	* FN	_ FP			_ FP		N TP	G TP	_ TP	S TP	I TP			N FP	Y FP		* FN	_ TP	F TP		S TP
			* FN		_ FP			_ TP			* FN									* FN	* FN						
					_ TP																						
U	L	T	S	_	O	F	_	O	T	H	E	R	_	P	E	O	P	L	E	‘	S	_	T	H	I	N	K
U FP	P FP	T TP	F FP	_ TP	U FP	V TP	_ TP	O TP	Y FP	Y FP	D TP	F FP	N FP	P TP	R FP	O TP	P TP	L TP	R FP	X FP	H FP	V FP	T TP	H TP	* FN	N TP	K TP
U TP	L TP		X FP		* FN				T TP	H TP		* FN	* FN		F FP				D TP	‘ TP	S FP	* FN					
			X TP												* FN						X FP						
																					* FN						
I	N	G	.																								
8 FP	J FP	G FP	; FP																								
I TP	N TP	G FP	* FN																								
		G TP																									

S5

I	F	_	L	I	F	E	_	W	E	R	E	_	P	R	E	D	I	C	T	A	B	L	E	_	I	T	_
I FP	T FP	B FP	F TP	* FN	_ TP	_ FP	V FP	W TP	E FP	F FP	E FP	_ FP	P TP	R FP	E FP	* FN	9 FP	_ FP	R FP	S FP	H FP	L TP	E FP	_ TP	I TP	V FP	G FP
I TP	V TP	_ FP	* FN			D FP	_ FP		* FN	* FN	F FP	_ TP		F TP	F TP		I TP	_ TP	* FN	* FN	B TP		V FP			T TP	* FN
		_ TP				* FN	I TP				* FN												V FP				
																							* FN				
W	O	U	L	D	_	C	E	A	S	E	_	T	O	_	B	E	_	L	I	F	E	,	_	A	N	D	_
W TP	O TP	8 FP	_ FP	* FN	_ TP	_ FP	F FP	D FP	X TP	D FP	B FP	T TP	O TP	J FP	B TP	E FP	_ FP	L TP	I TP	U FP	B FP	* FN	_ TP	C FP	N TP	D FP	B FP
		Y FP	* FN			* FN	D TP	A FP		D TP	* FN			* FN		F FP	* FN			F FP	B FP			Z FP		C TP	B FP
		D FP						* FN								* FN				D TP	* FN			* FN			* FN
		* FN																									
B	E	_	W	I	T	H	O	U	T	_	F	L	A	V	O	R	.										
B TP	E FP	* FN	W TP	I TP	T TP	H TP	D FP	* FN	T FP	B FP	V FP	P TP	Y FP	* FN	O TP	5 FP	* FN										
	D FP						O TP		T TP	_ TP	V FP		D FP			R TP											
	V FP										* FN		B FP														
	V FP												* FN														
	_ FP																										
	* FN																										