[실시간 물체추적을 이용한 타악기 구현]

윤형진, 배수민, 성보람
인하대학교 컴퓨터공학과
yun@inha.edu, bsm8734@naver.com, alsks513@icloud.com

요 약

[요약문 작성. 350 자 내외]

본 문서에서는 실시간 물체 추적 기능을 이용한 타악기 연주 애플리케이션을 제안한다. 제안하는 애플리케이션은 신경망 기반의 물체 검출 기술과 변화점 기반의 물체 추적 기술을 결합한다. 이를 통해, 목표물인 손을 빠르고 정확하게 추적할 수 있어, 실시간 악기 연주에도 무리없이 적용될 수 있다. 물체의 움직임을 각 프레임 간 픽셀의 색과 위치 변화를 통해 검출한다. CNN 기반의 물체 검출기는 물체를 추적하는 도중에 오차가 생기거나, 급격한 변화로 인한 잘못된 추적 경과를 바로잡기 위해 사용되었다.

1. 서론

[개발의 동기]
1/4 페이지

저마다 손 안의 컴퓨터를 가지고 다니는 시대이다. 다양한 무선 디바이스를 통해 다양한 어플리케이션을 시간과 공간의 제약없이 구동할 수 있게 되었다. 그에 따라 다양한 악기도 어플리케이션으로 만들어졌으며, 무거운 악기를 들고 다니지 않아도 손쉽게 음악을 연주할 수 있게 되었다. 그러나 주로 터치패드를 사용하도록 만든 악기 연주 어플리케이션에는 한가지 제약이 존재한다. 바로 3 차원 공간의 악기를 2 차원으로 구현하기 때문에 실제로 악기를 다루는 것과 같은 느낌이 나지 않는다는 것이다. 따라서 우리는, 사용자가 마치 현실에서 악기를 다루는 것처럼 손이나 스틱을 3 차원에서 움직일 수 있게 하기 위해, 화면(터치패드)이 아닌 카메라를 사용하고자 하였다.

[개발의 필요성 및 기대효과]
1/4 페이지

기존의 애플리케이션은 피아노나 드럼 등과 같이 휴대하기 어려운 악기를 장소의 제약 없이 연주할 수 있다는 점에서 이점이 있다. 그러나 기존의 터치 방식 악기 연주 애플리케이션 구현 방법에는 스크린이라는 한가지 제한된 가정이 존재하고, 오직 스크린에만 의존하는 것은 여러 단점을 만든다.

첫 번째는 디바이스 화면 크기에서 오는 제약이다. 사용자는 작은 휴대전화 화면에 한해서 악기 연주를 할 수 있다. 이는 작은 화면 안에서 손가락으로 조작해야 하기 때문에 조작의 제약이 생긴다. 조작의 제약은 원하지 않는 음을 내고, 현실보다 작은 도구들로 인해 사용자의 만족도를 하락시키는 결과를 초래한다.

두 번째는 사용자의 경험을 제한한다는 것이다. 기존의 애플리케이션에서 사용자는 평면에서만 악기를 조작할 수 있다. 그러나 현실에서의 악기를 다루기 위해서는 손을 3 차원적으로 상하, 좌우, 앞뒤로 움직일 수 있어야 한다. 실제로, 애플리케이션으로 구현된 런치패드는 성공을 거두지만, 피아노나 드럼과 같은 악기를 애플리케이션으로 구현한 것은 큰 성공을 거두지 못하는 이유도 여기에 있다. “누른다”라는 개념과 “친다/때린다”라는 개념에서 오는 차이이다. 따라서 드럼과 같은 타악기의 경우는 실제로 3 차원 공간에서 연주가 더 쉽고, 친근하다.

우리는 기기의 카메라와 물체 추적 기술을 활용하여 이러한 악기 연주에 있어 좀 더 재미있고 편안한 사용자 경험을 유도하도록 위와 같은 기술을 개발하고자 하였다.

[관련연구]
1/4 페이지

1. 신경망 기반의 물체검출

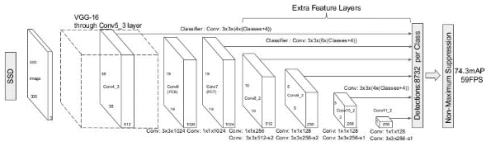
   가. Lending A Hand: Detecting Hands and Recognizing Activities in Complex Egocentric Interactions [1]

   손을 찾아 구별하는 개발을 방법을 제시하고 정확한 픽셀 단위로 손 영역을

만드는 방법을 볼 수 있다. 이를 통해 손 영역을 정확하게 판단할 수 있다.

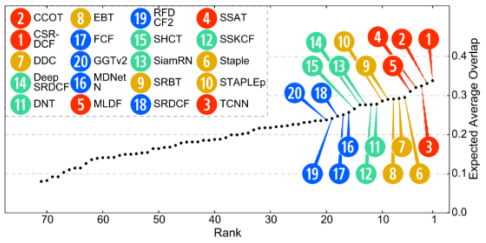
나. SSD : Single Shot Multi Box Detector[2]

하나의 깊은 신경망을 이용하여 영상에서 물체를 감지하는 방법을 제시한다. SSD 는 위치별로 개체가 있을 가능성을 점수로 만들고, 개체의 모양에 맞도록 박스의 사이즈를 조정한다. 기존의 단계를 지우고 네트워크를 캡슐화하여, 간단하다. 또한 정확하며 빠르게 모델을 훈련시킬 수 있다.



[Fig. 1] SSD model

2. OpenCV 를 이용한 물체 추적



[Fig.2] object tracker 비교 그래프 (VOT Challenge 16 제공)

가. CSRT tracker

Discriminative Correlation Filter (with Channel and Spatial Reliability).
정확도가 높으며, 낮은 FPS throughput 을 잘 견딘다.

나. KCF tracker

Kernalized Correlation Filter.
물체가 가려지는 현상등에서 제대로 작동하지 않는다. 빠른 FPS throughput 이 필요할 때 사용되며, 대신 낮은 정확도를 가지고 있다.

다. MOSSE tracker

정확도는 높지 않으나, 매우 빠르다. 오직 빠르기만 중요할 때 사용된다.

[개발하고자 하는 기술 및 개발계획]
1/2 페이지

개발하고자 하는 기술을 다음과 같이 크게 4 단계로 나누어 설명할 수 있다.

1. 타악기는 손이나 막대와 같은 물체로 연주하게 된다. 따라서 악기를 연주하기 전, 연주자가 악기를 연주하는 데 사용할 물체(손)를 컴퓨터가 감지 (detection)할 수 있어야 한다. 이러한 감지에 딥러닝으로 학습한 모델이 사용된다. 이때 중점적으로 확인할 요소는 정확도이다.

2. 악기의 연주는 실시간으로 이루어지기 때문에 시각적인 지연은 사용자가 어플리케이션을 사용하는데 있어, 매우 치명적인 요소가 될 것이다. 따라서 물체 추적에서의 중점적인 요소는 바로 속도이다. 실시간 물체 추적(real-time-object-tracking)을 통해 사용자는 바로바로 자신의 동작에 대하여 소리나 모션등의 피드백을 받아야한다.

3. 위치를 지정하여 해당 위치에 물체가 진입했음을 확인하면, 악기에 해당하는 소리가 나도록 한다. 또한, 물체가 프레임의 지정 영역(악기 영역)에 오래 머무르는 경우가 있는데, 우리가 구현하고자 하는 타악기의 경우에는 손을 올려놓는 것만으로는 소리가 나지 않아야 하기 때문에 이를 주의하여 구현한다.

4. 덧붙여, 단순히 하나의 물체 추적으로 구현이 가능한 북과는 달리, 드럼의 경우는 연주자가 사용하는 막대(손)가 총 2 개이다. 또한, 두 개 이상의 타악기를 연주하고자 할 때도 있을 것이다. 한 번에 하나의 타악기만 연주해야 한다든가, 드럼이나 장구 등의 두 개의 손이 필요한 악기에 대한 제약을 없애기 위해 2 개 이상의 물체에 대해서 다중물체추적(multi-object-tracking)이 가능하도록 구현하고자 한다. 이렇게 추적한 다중의 물체에 대해서 위 1~3 단계의 기술이 구현되도록 한다.

2. 개발내용

2.1 실시간 손의 감지 및 움직임 트레킹

실시간으로 손의 움직임을 감지하기 위하여, 사용해본 방법은 총 네 가지이다.

2.1.1 WebCam Motion Detector

웹 캠의 영상을 이용하여 컬러를 흑백 프레임, 반전 프레임, 임계처리한 프레임 총 네 종류의 프레임을 가지고 움직임을 감지해보았다. [3]

이 방법을 사용한 이유는 비디오는 결국 프레임이라는 이미지의 연속이기 때문에, 정적인 첫번째 프레임과 다른 프레임의 픽셀의 강도 값을 비교하여 두 이미지를 비교하면 움직임을 감지할 수 있지 않을까 라는 생각이 들어 선택하게 되었다. 결과가 생각처럼 나오지는 않았다. 조금만 움직여도 감지해버리는 바람에, 손만 움직였다 생각했지만 화면 전체를 잡거나 사람의 몸 전체를 감지하는 등 정확도가 낮은 한계가 있었다.

비디오는 프레임이라 불리는 그림의 모음으로 취급된다. 여기서 정적이어야 하는 첫 번째 프레임과 다른 프레임을 비교한다. 각 픽셀의 강도 값을 비교하여 두 이미지를 비교한다. 해당 코드를 실행하면 4 개의 창이 나온다. 아래는 각 창에 대한 설명이다.

1. Gray Frame: RGB 이미지에서는 세가지 강도값이 있는 반면에 Gray 이미지에서는 하나의 강도값만 있다. 따라서 grayscale 에서의 intensity 차이를 계산하기가 쉽다. 회색조의 이미지를 약간 흐릿한 효과를 주어 보여준다.
2. Difference Frame: 현재 프레임에 대한 첫번째 프레임의 intensity 차이이다.
3. Threshold Frame: 특정 픽셀의 강도 차이가 지정한 threshold 보다 크면, 해당 픽셀은 흰색으로 나타나고, 지정한 threshold 보다 작으면 해당 픽셀은 검정색으로 나타낸다.
4. Color Frame: 움직이는 물체 주위의 녹색 윤곽선과 함께 컬러 이미지를 볼 수 있다.

정리하자면, webcam 으로 각 시간당 RGB 프레임을 받아 intensity 처리를 쉽게 하기위해 gray scale 로 변환한 이미지에 blur 처리를 하고, 현재 프레임과 처음의 프레임의 차이를 구하여, 변화량이 threshold 를 넘은 부분에 대하여 녹색 윤곽선과 함께 컬러 프레임에 나타낸다.

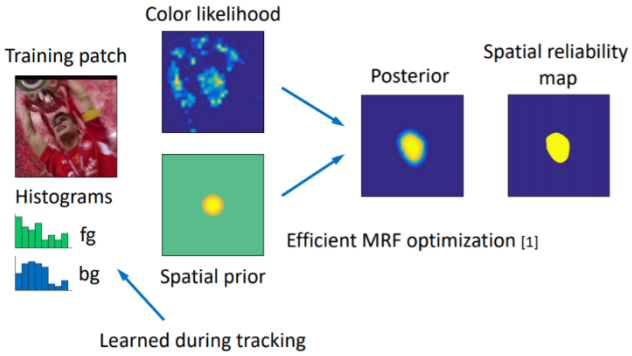
위의 4 단계를 거치며, 배경과 실제 움직이는 물체를 분리하여 화면에 나타낼 수 있다. (녹색 윤곽선을 통해 color frame 에 보여준다.)

2.1.2 CSRT (Discriminative Correlation Filter with Channel and Spatial Reliability, CSRT) [4]

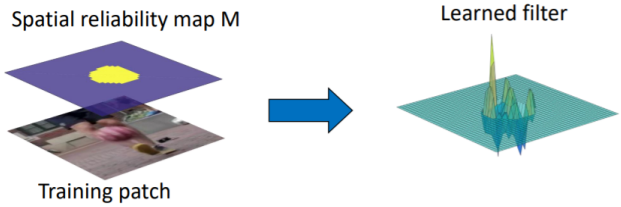
영역을 box 로 지정하여 이 박스 안의 물체를 추적하는 것이다.[5]

이 것은 OpenCV 에서 제공하는 tracker 인 KCF, CSRT, MOSSE 등의 트레커를 사용하였는데, 각 트레커 마다 성능과 속도의 차이를 보였다. 특히 CSRT tracker 는 상대적으로 높은 정확도를 보였으나, 프레임이 16 프레임 안팎 정도로 트레킹하는데 어려움을 겪었고, MOSSE tracker 는 평균적으로 100 프레임 이상의 높은 프레임을 자랑했으나, 저조한 정확도로 인하여 트레킹 하는 물체를 자주 놓쳤다. 이 방법만으로는 실질적인 트레킹을 하는 데 어려울 것이라고 판단하였다. (표 1-2)

DCF tracker 는 기존의 Single-Channel CF tracker 에서 channel reliability 와 spatial reliability 를 도입한 것이다. [6]



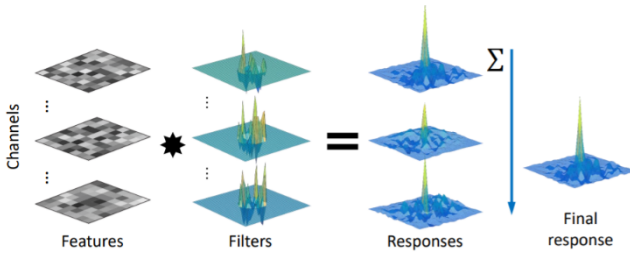
[Fig.3] Spatial Reliability Map 을 만드는 과정



[Fig.4] Spatial Reliability Map 을 통하여 학습된 필터

1. Spatial Reliability Map

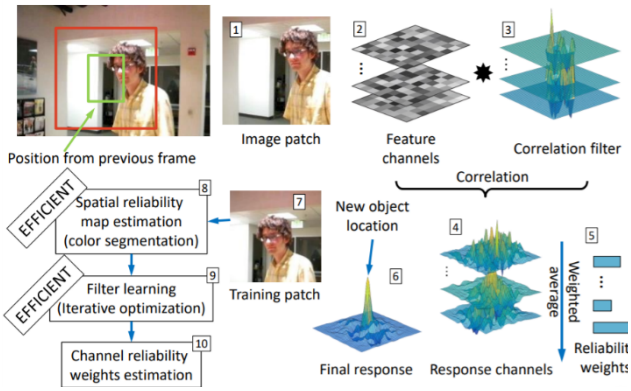
training patch 의 일부분만 학습하는 것이다. (closed-form solution 이 존재하지 않으므로 ADMM optimization 을 사용한다.) training patch 를 컬러 히스토그램으로 만들어 패치 안에 비슷한 색을 가지는 지점을 뽑고, 이를 spatial prior 과 합쳐, posterior 을 생성한다. 이렇게 만든 posterior 지역에서 threshold 를 넘기는 지점을 spatial reliability map 이라고 한다.



[Fig.5] Channel Reliability 를 위해 여러 필터를 사용하는 방법

2. Channel Reliability

다중 채널을 만드는 데에는 HoG 와 ColorNames, CNN 이 사용된다. 여러 channel 의 차별성(response peak)을 반영하여 최대 response 값을 가져온다. 이들의 가중치 평균(reliability weight)을 통해서 resonse channel/ final response 를 만든다. final response 의 max 값을 가리키는 지역이 object 가 존재하는 지역이다.



[Fig.6]. DCF-CSR 의 과정 정리

정리하자면, DCF-CSR 은 다음과 같은 장점을 가지고 있다. [7]

1. 현실적인 학습 예시를 생성하고 큰 변위의 문제도 해결할 수 있다.(검색 범위 확대)
2. 직사각형이 아닌 물체를 추적하므로, 배경이 필터에 포함되지 않게 할 수 있어서 배경의 영향을 줄일 수 있다.
3. 산점도를 감소시킨다.
4. 속도와 정확도 부분에서 성능이 매우 뛰어나다. (occlusion 에서도 물체를 놓치지 않음)

2.1.3 Dynamic Color Tracker

특정 색의 HSV 값을 지정하여 그 색의 물체를 감지하는 모델을 사용하였다. [8]
이 방법을 사용한 이유는 타악기연주에 사용될 채나 손의 색상을 지정하여, 그 물체를 감지한다면 더 좋은 정확도가 나오지 않을까 라는 생각이 들어서 사용하게 되었다. 다음은 모델의 작동원리이다.

1.
- WebCam 으로부터 frame 을 읽어들인다.
- Frame 을 HSV 영역으로 변환한다.

2.
- 위에서 생성된 frame 에서, 감지하고자 하는 물체의 HSV 영역을 찾아 mask 를 생성한다.
- 생성된 mask 에 열림, 닫힘 연산으로 차례로 적용하여 보다 선명한 mask 를 얻어낸다.

3.
- 얻어낸 mask 에서 가장 큰 contour 를 찾아낸다.
- 찾아낸 contour 에 외접하는 원을 생성한다.
- 외접원의 반지름이 임계값 이상일 때, 해당 부분에 강조표시를 한 뒤, 이미지를 출력한다.

실험결과, 적절한 색의 영역을 지정하였을 때 위 방법들 보다 훨씬 높은 정확도를 자랑하였다. 하지만 색을 지정해 주는 방식이 번거롭고, 일반적으로 정확한 색을 맞추는데 시간이 걸리고 빛에 의한 그림자 부분을 인식하지 못하거나 배경과 색이 비슷한 경우 다 같이 인식해버리는 단점이 있다. (표 1-3)

2.1.4 SSD (Single Shot Multi Box Detector, SSD)

손 모양이 학습된 SSD 모델을 이용하여 손을 감지하고 추적하는 방법이다. [9]
SSD 모델은, 기존 detection 에 사용된 base network 에 여러 개의 feature layer 들을 추가한 모델이다. 다른 기존의 방법들과 비교해서 빠르고 정확하다는 장점이 있다. 다음은 사용된 모델의 작동원리이다.

1.
Webcam 으로부터 frame 을 읽어들인다.

2.
학습된 모델을 통해 frame 상의 손의 추정위치와 정확도를 얻는다.

3.
정확도가 임계값 이상이면, 얻어낸 추정위치에 강조표시를 한다.

4.
강조표시를 한 frame 을 출력한다.

2.2 이미지를 이용한 손 추적

2.1 의 방식들의 한계와 문제점을 종합적으로 분석하여, 빠른 프레임 속도와 높은 정확도를 위하여 가장 정확도가 높았던 2.1.4 방식과 다른 방식들을 접목하여 이용할 계획을 해보았다.

1.
첫번째로, 손이 학습된 SSD 모델을 이용하여 초기의 손의 좌표를 받아오고 해당 좌표에 박스를 띄운다.

2.
지정된 좌표 값을 초기좌표로 하여 CSRT Tracker 를 이용하여 손을 추적한다.

3.
손을 추적하던 박스의 좌표가 북 이미지 영역 안에 진입 시 소리가 출력된다.

4.
일정한 시간이 지나거나 reset 버튼(키보드 r 키)을 누르면 초기 상태로 돌아가게끔 한다.

여기서 소리를 불러오고 출력하는 시간(python3 에서 대략 1 초)때문에 화면이 뚝뚝 끊기는 현상이 발생하였다. 이는 실시간으로 영상을 출력해주는 프로그램에겐 치명적인 시간이었다. 이를 multi-threading 방식을 이용하여 소리를 출력하는 쓰레드를 따로 생성하여 병렬적으로 실행하게하여 이를 해결하였다.

이와 같은 방식을 사용하는 이유는, 가장 정확도가 높았던 SSD 모델을 이용하기엔 너무 높은 성능이 요구되었고, 항상 좋은 사양의 기기를 이용하기는 힘들기 때문에 이 것을 최적화할 방법이 없을까 생각하던 중 일정시간에 한번씩 높은 정확도가 나오는 방식으로 갱신해주고, 중간의 시간 동안은 프레임이 잘 나오고 정확도가 약간 떨어지는 방식을 사용하는 건 어떨까 라는 생각에서였다. 이렇게 하면 정확도가 낮아 감지하고 있던 물체를 벗어나는 트레킹 방식들을 보완하면서, 높은 정확도에 높은 사양을 요구하는 방식을 좀더 최적화할 수 있을 것이다.

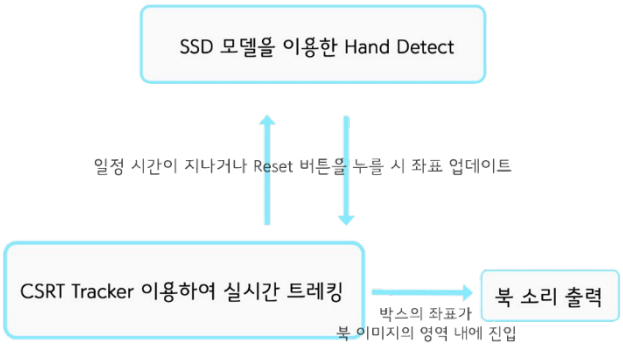


그림 1. 모델구조

3. 실험 결과 및 분석

성능을 향상시키기 위하여 학습된 모델 + openCV object tracker 를 병합하여 사용하였다. 최종적으로 선택한 모델과 앞에서 설명한 다양한 방법들의 장점과 단점을 비교해보았다.

첫번째로 물체를 인식하기 위하여 사용했던 방식인 움직이는 영역을 검출하기 위하여 threshold 와, grayscale, 반전된 이미지 총 세가지의 이미지를 이용하였다. (2.1.1) 결과는 좋지 않았다. 움직이는 물체를 우선 모두 감지하기 때문에 정확도도 떨어지고, 효율적이지 못하였다.

두번째로 OpenCV 의 object tracker 인 csrt tracker 를 사용하는 방식을 사용해보았다.(2.1.2) 정확도는 좋았던 편이지만, 물체를 중간에 놓치거나 다를 개체로 바뀌어야할 때 스스로 인식할 수 없기 때문에 새로 좌표를 지정해주어야 하는 번거로움이 있었다. 초기 좌표를 미리 지정해주고 프로그램을 처음 실행시킬 때 그 영역 내에 손을 고정시키는 방식 또한 크기나 위치를 정확히 맞추기 힘들었기 때문에 좋은 성과를 얻어내기는 힘들었다.

세번째로는 색을 지정하는 방식이었는데,(2.1.3) 색의 영역만 정확히 지정을 해주면 아주 좋은 결과를 얻을 수 있었다. 하지만 밝기나 채도가 바뀐다던지, 비슷한 색이 같이 등장하는 경우에는 원하는 물체를 트레킹하는 데에 어려운 점이 있었다.

마지막으로는 손이 학습된 SSD 모델을 사용한 방식(2.1.4)인데 정확도도 매우 높은 편이였고, 중간에 물체를 놓치거나 다른 물체로 바뀌였을 때에도 좋은 성과를 얻을 수 있었다. 하지만 하드웨어적인 한계 때문에 좋은 성능 자체를 얻기가 힘들었다.

최종적으로 사용한 두번째와 마지막 방법을 병합하여 사용한 경우, 물체를 놓치거나 다른 개체로 바뀌었을 때 다시 좌표를 잡기위해 리셋키를 눌러주어야 하는 번거로움은 있지만, SSD 모델만은 이용한 경우보다 약 2 배가량 높은 프레임을 자랑하였고, 처음에 알아서 손을 인식해주기 때문에 따로 색을 지정해주어야 한다거나 좌표를 지정해주어야하는 번거로움이 훨씬 적었다. 아래의 표는 이를 바탕으로 성능과 속도를 비교한 표이다.

방식	평균 프레임	장점	단점
2.1.1	30fps 이상	좌표를 지정해주지 않아도 자동으로 움직이는 물체 검출	움직이는 모든 물체 추적. 원하는 물체만 따라가는데 한계

2.1.2	10 ~ 15 fps	비교적으로 물체 추적에 있어서 좋은 성능	원하는 물체의 좌표 초기에 지정해 주어야 함
2.1.3	30fps 이상	적당한 색 범위를 안다면 원하는 물체 좌표 추적 잘됨	색 범위를 알아내기 힘듦. 비슷한 색을 같이 인식하거나, 밝기 등 바뀌었을 때 재지정해주어야 함
2.1.4	5 ~ 8 fps	스스로 개체 좌표를 인식가능. 놓치거나 바뀌어도 원활한 추적 가능	너무 낮은 프레임으로 연주에 부적합.
최종 방식	10 ~ 12 fps	스스로 개체 좌표 검출 + 추적	개체를 바꾸거나 놓치는 경우 r 키를 눌러 재지정해주어야함

비교 실험에 사용한 기기의 사양은 CPU 인텔 I3-5005U (2.00GHz), ram 4GB, 그래픽 인텔 HD graphics 5500 (노트북 내장 그래픽카드)로 좋지 않은 사양에 속한다. 최종적으로 사용한 방법은 이러한 저사양 기기에서도 수월하게 실시간으로 연주를 하는 것을 가능하게 해주었다.

4. 결론

초기에 구상하였던 부분 중, single-object tracking 을 통한 타악기 연주까지 성공적으로 구현하였다. 여러 방식의 object detector, tracker 에 대한 특성과 장단점을 파악하고 이들을 서로 접목하여 보완하고자 노력하였다. 정확도와 속도를 모두 놓치지 않기 위해서 여러가지 기술들을 비교하고 접목해보았고, 그 중에서 가장 두가지 면에서 성능이 좋았던 모델을 선택하였다. 이렇게 선택된 방식을 사용 하였을 때, SSD 모델을 이용하였을 때보다 약 2 배정도의 프레임이 향상되었고, csrt tracker 를 이용하였을 때의 단점인, 원하는 시기에 좌표를 초기화하지 못하는 점에 대해서도 성공적으로 개선된 결과를 보여주었다.

결과적으로 초기에 생각한 여러 악기나 여러 사람의 손으로 연주하는 것은 구현하지 못하였으나, 단일 물체에 대한 tracking 과 달리, 다중 물체 추적에서는 한 프레임에서 다른 프레임으로 넘어갈 때, 같은 물체임을 확인해야하므로, 구현에 시간이 더 필요했다.

영상 1. 시연영상

참고문헌

[1] Sven Bambach, Stefan Lee, David J. Crandall, Chen Yu : “Lending A Hand: Detecting Hands and Recognizing Activities in Complex Egocentric Interactions”. In ICCV. (2015)

[2] Wei Liu1, Dragomir Anguelov2, Dumitru Erhan3, Christian Szegedy3, Scott Reed4, Cheng-Yang Fu1, Alexander C. Berg1: “SSD:SingleShotMultiBoxDetector”. In ECCV . Lecture Notes in Computer Science, vol 9905 (2016)

[3] https://www.geeksforgeeks.org/webcam-motion-detector-python/
ujjwal sharma 1, WebCam Motion Detector in Python

[4] https://www.pyimagesearch.com/2018/07/30/opencv-object-tracking/
Adrian Rosebrock, OpenCV Object Tracking

[5] Discriminative Correlation Filter Tracker with Channel and Spatial Reliability 논문,

[6] https://github.com/alanlukezic/csr-dcf (github),

[7] https://www.youtube.com/watch?v=Yl-grwGch_M (YouTube)

[8] https://pastebin.com/NkUTXM8T
anonymous, dynamic color tracking

[9] https://github.com/victordibia/handtracking ‘
Victor Dibia, Real-time Hand-Detector using Neural Networks (SSD)