**WebRTC를 적용한 AI 자율주행 영상처리**

안상준\*, 김기현\*, 배재현\*, 이상민\*, 정설영\* , 안병준\*\*

\*경북대학교 IT대학 컴퓨터학부, \*\*팀그릿

**AI Self-driving Video Processing using WebRTC**

**Sangjun Ahn, Kihyun Kim, Jaehyun Bae, Sangmin Lee, Seolyong Jung, Byungjun Ahn**

School of Computer Science and Engineering, Kyungpook National University, Teamgrit

E-mail : twknds@naver.com, snowflower@knu.ac.kr

**요 약**

4차 산업혁명에 들어서며 자율주행 기술과 영상을 보다 빠르게 처리하기 위한 초저지연 기술이 중요성이 높아지고 있다. 또한, 비대면 사회가 도래함으로 인해 지연이 적은 영상처리 기술의 필요성이 더욱 대두되었다. 따라서 본 논문에서는 자율주행기술과 초저지연 영상 스트리밍 기술을 접목하여 스마트 시티에 적용할 수 있는 기술을 미리 구현하고, 본 연구 결과 제한된 임베디드 시스템에서도 자율 주행과 초저지연 영상 스트리밍을 효과적으로 제어할 수 있음을 확인하였다.

**1. 서론**

현대 사회에 이르러 자율주행은 빼놓을 수 없는 기술이다. 또한, 자율주행 기술은 2004년 DARPA Grand Challenge를 시작으로 현재까지 눈부신 발전을 이루고 있다.[1] 특히, ADAS(Advanced Driver Assistance Systems)와 같은 지능형 운전자보조시스템은 탑승자에게 보다 편안함과 안락함, 그리고 안전함을 제공해준다. 또한, COVID-19로 인한 비대면 사회가 도래함에 따라 영상 전송 기술과 영상 전송 속도 또한 중요해지고 있다.

따라서, 본 논문에서는 NVIDIA사의 Nano Developer Kit(이하 Nano)을 기반으로 한 자율주행 자동차를 개발하며, 더하여 WebRTC를 통한 Ultra-low latency video streaming을 바탕으로 자율주행 영상을 웹 브라우저나 모바일을 통해 실시간으로 초저지연으로 볼 수 있게 하고자 한다. 특히, 자율주행 기술 중 가장 필수적인 Lane Detection(차선 인식)과 Object Detection(객체 인식을 구현하고, WebRTC를 이용하여 주행 중인 차량의 상태를 확인할 수 있도록 하는 것을 목표로 한다.

**2. 본론**

제안된 시스템은 Nano 8GB, 라즈베리파이용 CSI카메라, JetBot 등의 하드웨어와 Ubuntu 18.04, Jupyter Notebook 등의 소프트웨어를 기반으로 개발환경을 구축하였다. 표 1을 참조하면 Jetson Nano는 같은 SBC(Single Board Computer)인 Rasberry Pi와 비교해서 CPU성능 면에서는 떨어지나, 병렬 계산이 중요한 경우에는 GPU가 강력한 성능을 내기 때문에 본 프로젝트에서 채택하였다. 아래는 라즈베리파이와 Nano의 주요 제원을 간단하게 표로 비교한 자료이다.

|  |  |  |
| --- | --- | --- |
|  | Raspberry Pi 4 | Jetson Nano 8GB |
| CPU | Quad-core ARM A72 @ 1.5Ghz | Quad-core ARM A57 @ 1.43 Ghz |
| GPU | Broadcom VideoCore VI | 128-core Maxwell |
| Memory | 2,4,8GB LPDDR4 | 8GB 64-bit LPDDR4 25.6GB/s |
| Camera | 2-lane MIPI CSI-2 connector | 1x MIPI CSI-2 connector |
| Price | $35 USD | $99 USD |

표 1.주요 제원 비교

본 프로젝트에서는 Jetbot과 Nano를 결합하여 자율 주행을 수행하는 것이 목표이기 때문에 Jetbot 전용 소프트웨어가 내장되어있는 Jetbot Image를 설치하여 사용하였는데 이는 본래 Nano용 image 파일인 JetPack 4.3을 기반으로 제작된 이미지 파일이다.

딥 러닝으로 구현한 부분은 차선 인식(Lane Detection)과 객체 인식(Object Detection)이다. 모든 학습은 몇 개의 층만 재학습하는 전이 학습(Transfer Learning)을 사용하였다. 본 논문에서 제안하는 자율 주행 방식은 지도학습임을 알린다. 차선 인식에서 사용한 모델은 ResNet18 모델이다.[2] Nano는 한정적인 GPU 자원을 가지고 있기 때문에 비교적 층의 개수가 적으며 성능이 좋기 때문에 본 모델을 선택하였다. 본 모델은 Residual block과 컨볼루션 층, 최대 풀링 층 그리고 전결합층으로 이루어진 모델이다. ResNet 모델은 Residual block 덕분에 기울기가 소실되어 학습이 제대로 되지 않는 기울기 소실 문제가 발생하지 않는다는 강점이 있다. 따라서, 모델의 내부 구조는 변경하지 않았고, 전결합층만 구조에 변경을 주었다. 1024개와 128개의 노드로 설정하였고, 최종 출력 노드 수는 2개로 설정해주었다.

데이터 셋의 구성은 본 프로젝트를 위해 구성한 트랙에서 Nano의 카메라를 통해 직선 코스와 회전하는 구간에 대해서 229장을 수집하였다. 라벨링의 경우, Nano가 주행해야 하는 목표 좌표인 X, Y 좌표를 각각 주었다.

다음으로 객체 인식(Object Detection)을 수행하는 딥 러닝 모델로는 MobileNet을 사용하였다.[3] 본 모델은 임베디드 기기에서 실시간으로 구동하는데 최적화된 딥러닝 모델이다. 기존 컨볼루션 연산을 depthwise separable convolution 연산으로 변경하여 파라미터 개수를 줄였다. 기존 훈련된 가중치에서 본 프로젝트에서 인식해야 하는 사람과 교통 표지판을 고도화하기 위해 추가적으로 훈련을 진행하였다. 표지판과 사람은 각각 120장씩 수집하였다. 또한, 현재 두 가지 클래스에 대해서만 인식을 하기 때문에 전결합층의 최종 노드 수를 2개로 설정하였다.

자율 주행 중인 Nano 주행 영상을 Ultra-low latency video streaming을 하기 위해 WebRTC를 사용했다. 웹 실시간 커뮤니케이션이라고도 하는 WebRTC는 Google, Mozilla 등에서 홍보하는 개방 소스 프로젝트로서 Javascript API를 통해 플러그인 없이 실시간 커뮤니케이션을 가능하게 한다.[4]

기존의 Server-Client 모델에서 비디오 전송은 구조상 지연시간이 발생한다. WebRTC는 P2P 방식을 사용하며, Peer들간 직접적인 통신을 하여 지연시간을 줄일 수 있다. 이러한 P2P 연결을 하기 위해서, WebRTC에서는 Signaling Server가 필요하다. Signaling이란 네트워크 검색, session 생성 및 관리, media 기능 메타데이터 교환과 관련된 프로세스들을 말한다. 이러한 행위는 각 peer들이 서로에 대해 미리 알아야 커뮤니케이션을 시작할 수 있기에 필수적이다.

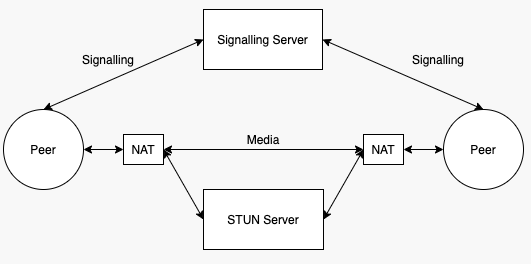


그림 1 WebRTC P2P 구축

그림 1은 WebRTC의 P2P 구축에 대해 간단히 시각화한 자료이다. Signaling Server는 peer들 간 정보를 보내는 데 사용된다. 첫 번째로 offer를 생성하고, 다른 peer들에게 offer를 보낸다. 두 번째로, ICE(Interactive Connection Establishment) candidates handle이 필요하다. WebRTC는 connection을 구축하고 peer를 발견하는 ICE protocol을 사용한다. 세 번째, 다른 peer가 보낸 ICE candidate를 처리해야 한다. Peer는 이 candidate를 받은 후, 해당 candidate를 pool에 추가하면 된다. 그 후, 다른 peer가 offer를 받았을 때, remote description을 설정해야 한다. 또한, initiating peer에 전송되는 answer를 생성해야 한다. 마지막으로, initiating peer는 answer를 받고 Remote Description을 설정하면 WebRTC connection이 성공적으로 구축이 된다. 따라서, Signaling Server 없이 두 peers간 직접 영상을 주고받을 수 있게 된다.

아래 그림 2는 WebRTC의 P2P 연결 과정을 다이어그램으로 시각화한 자료이다.

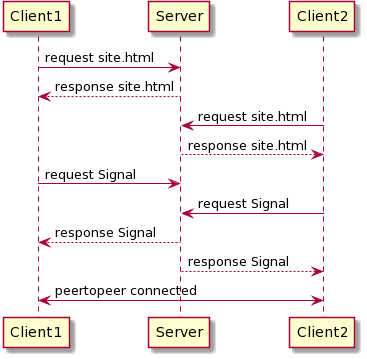


그림 2. WebRTC P2P 시퀀스 다이어그램

그러나, 이러한 과정에서 NAT Issues가 발생한다. NAT은 장치에 로컬 네트워크 내에서 사용할 IP 주소를 제공한다. 따라서, 이 주소는 로컬 네트워크 외부에서는 접근할 수 없다. 이러한 public address가 없다면 peers들은 통신할 수가 없다. 이러한 문제를 해결하기위해 STUN server를 사용했다. Network 정보를 peer들간에 공유하기 전, peer는 STUN server에 IP 주소를 받아올 수 있으며, 이를 통해 public-facing IP address를 가질 수 있게 된다. 그 후, 해당 IP와 port 정보를 연결할 peer에 공유하면 NAT issues를 해결하며 연결할 수 있게 된다.

실제 WebRTC 사용을 위해 AWS(Amazon Web Service) EC2 서비스를 사용하여 배포하도록 한다. Signaling Server는 NodeJS 기반의 프레임워크 express를 통해 구현하였으며, 실제 영상을 처리하고 server에 요청을 보내는 Front-End는 ReactJS를 사용하여 SPA를 구현하여 server의 부하를 줄였다.

**3. 결론**

본 논문에서는 성능에 제한이 많은 임베디드 하드웨어에서의 딥 러닝 모델 구동을 제안하고, 영상처리 전송에서 발생하는 overhead와 latency를 WebRTC를 적용하여 줄임으로써 실시간으로 주행 중인 차량의 상태를 확인 및 제어할 수 있는 방식을 소개한다. 또한, Resnet과 edge device에서 구동하도록 경량화 된 딥러닝 모델인 MobileNet 모델을 사용하여 지도학습으로 자율주행 문제를 해결하였다.

**[참고문헌]**

[1] The DARPA grand challenge – development of an autonomous vehicle, IEEE Intelligent Vehicle Symposium

[2] Deep Residual Learning for Image Recognition, CVPR 2015

[3] MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications, CVPR 2017

[4] WebRTC 기반의 화상회의 시스템과 이기종 단말들간의 상호 연동성 확보를 위한 시스템 구조 설계 및 개발, 한국정보과학회 2017

[5] 3CS, (2016, October 10). WebRTC. [Online], Avail-able, [http://www.3cx.com/global/kr/voip-sip-WebRTC/what-is-WebRTC/](http://www.3cx.com/global/kr/voip-sip-webrtc/what-is-webrtc/).

[6] WebRTC, (2016, October 10). WebRTC, [Online], Available: [https://WebRTC.org](https://webrtc.org)