Azure Kinect 기반의 네트워크 서비스를 통한 아바타 제어

오재광, 유지상, 권순철*
광운대학교, *광운대학교
dhworhkd11@kw.ac.kr, jsyoo@kw.ac.kr, ksc0226@kw.ac.kr*

Avatar Control through Networking Service by Using Azure Kinect

Jaekwang Oh , Jisang Yoo and Soonchul Kwon*
Kwangwoon Univ., *Kwangwoon Univ

요 약

본 논문에서는 RGB-D 카메라 Azure Kinect를 통해 획득할 수 있는 깊이와 사람의 관절 정보를 이용하여 유니티 상의 아바타 객체 애니메이터를 생성하고, 유니티 Photon API를 이용하여 네트워킹 서비스를 하는 방법에 관한 내용을 다룬다. Azure Kinect SDK를 통해서 얻은 관절의 정보를 이용해 유니티 상의 humanoid 아바타의 관절과 매칭하고, 실시간의 동작을 입력으로 받아 아바타의 자연스러운 애니메이터를 생성한다. 그러고 나서 Photon 서비스 플랫폼을 이용하여 관절 정보 기반의 정보를 송신하여 네트워킹 서비스를 구현했다. 이 과정 중 초기에 양측의 로컬에서 상대방의 아바타 애니메이터가 실시간으로 동작하지 못하고 눈에 보이는 딜레이가 발생하였다. 이 문제를 송신하는 데이터를 인코딩하여 압축하는 과정을 통해 해결하였고 30fps 속도로 딜레이 없이 동작이 가능해졌다.

Ⅰ. 서론

본 논문에서는 RGB-D 카메라인 Microsoft Azure Kinect[,1,2]에서 획득한 관절 정보를 이용한 아바타의 애니메이터 생성 및 네트워킹 서비스 내용을 서술한다. 관절 정보란 kinect의 내장된 RGB 센서와 IR 센서를 이용해 얻은 깊이 정보를 통해 관절들 각각의 위치 정보를 추출하여 얻은 데이터 집합이다. 관절 정보를 통해 유니티 상의 아바타의 관절 정보를 매칭하여 카메라로 인식한 사람의 동작이 아바타의 동작으로 이루어 애니메이터를 생성한다. 또한, 유니티 안에서의 위와 같은 동작들이 Photon[3]이라는 클라우드 서비스를 통해 여러 로컬에서 동작할 수 있도록 한다. 이 과정 중 가공하지 않은 관절 정보를 실시간으로 통신했을 때 눈에 띄는 딜레이가 발생하였으며, 10fps 이하의 속도에서만 정상 동작하였다. 위 문제를 데이터 압축을 통해서 30fps에서 정상 동작이 가능하도록 만들었다.

Ⅱ. 본론

본 논문에서는 Azure Kinect[1] 카메라를 통한 아바타 동작 및 네트워킹 서비스 방법을 제안한다. 그림 1 은 제안하는 방법에 대한 흐름도를 나타낸다. 제안하는 방법은 사용자 인식, 관절 정보 추출, Photon 클라우드를 통한 데이터 관절 정보 송수신, RPC를 통해 애니메이터를 수행하는 과정으로 이루어져 있다.

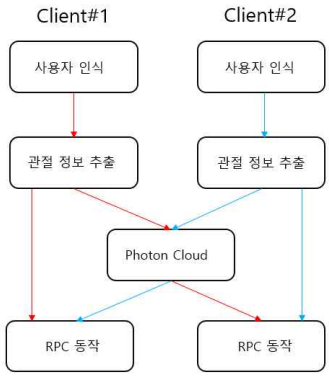

그림 1. 제안하는 방법의 흐름도.

첫 번째로 사용자 인식과 관절 정보 추출의 경우 Microsoft에서 제공하는 Azure Kinect Body Tracking SDK[4]를 사용하여 진행한다. DNN(Deep Neural Network) 모델[5,6]을 통해서 사용자를 인식하고 관절에 대한 정보를 추출할 수 있으며, 그림 2 와 같이 표현할 수 있다.

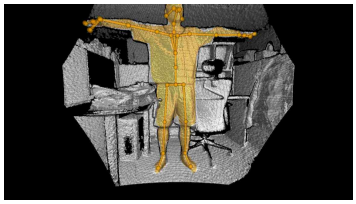

그림 2. 사용자의 관절 정보

Azure kinect를 통해서 얻은 관절 정보를 실시간으로 송수신을 하며 유니티에서 동작을 하기 위해서 Photon[3] 클라우드 서비스를 사용한다. Photon[3]은 유니티를 지원하는 클라우드 기반 서버 플랫폼이다. Photon 클라우드 서비스는 서버의 구조에 따라 크게 3가지 형태로 지원된다. 본 논문은 룸 기반의 릴레이 서버 형식의 PUN(Photon Unity Networking)[7]을 기반으로 진행하였다. 위 과정을 통해 얻은 사용자 ID 정보 및 32개의 관절 정보를 Photon 클라우드에 송수신하여, RPC를 통해서 다른 주소 공간에서도 네트워킹 객체가 동작을 진행할 수 있도록 하였다. 주고받는 데이터 packet의 크기의 경우 사용자 ID 및 32개의 관절 정보를 포함해 1,284byte 크기를 갖는다.

30fps 속도의 RPC 동작에서 눈에 띄는 딜레이가 발생했다. 1,284byte 크기의 packet으로는 10fps 이하의 속도에서만 딜레이 없는 정상적인 동작이 가능했다. 목표로 하는 실시간 동작을 위해서 데이터 packet을 크기를 압축시켰다. 고정되어 변하지 않는 정보 는 한 번의 송수신만 하게 수정하였고, 실수형 자료형을 사용하는 데이터에 대해서 byte 크기가 더 작은 정수형 자료로 인코딩 및 자료형 변환을 통해서 1,284byte의 크기를 890byte 크기로 바꾸어 주고받을 수 있게 하였다.

그림 4. 와 같이 유니티상의 아바타가 Azure kinect[1]를 통해 사용자를 인식하여 사용자와 같은 동작을 할 수 있도록 애니메이터를 생성한다.

그림 3. 유니티상의 아바타 동작

데이터 압축을 통해 트래픽량을 기존보다 월등히 줄였고, 딜레이 또한 시각적으로 확인할 수 없을 정도가 되었다. ping은 20~25ms 정도를 유지하며 30fps에서도 유지 실시간 동작이 가능하다. 아바타의 경우 fbx파일 형식의 데이터를 사용하였으며, 관절 정보가 있는 rigged model에 제한되어 위 동작이 가능하다.

Ⅲ. 결론

본 논문에서는 RGB-D 카메라 Microsoft Azure Kincet에서 획득한 사람의 관절 정보를 통해 유니티상의 아바타 애니메이터 생성 및 네트워킹 서비스를 다루었다. 관절 정보 매칭, rotation 계산을 통해서 아바타의 자연스러운 동작을 생성했다. 기존 가공되지 않았던 데이터 packet의 교환으로 인해 발생 되었던 딜레이를, 데이터 가공을 통해서 약 30%정도의 압축이 가능했다. 30fps 상에서의 Network 객체인 아바타의 동작들이 딜레이 없이 실행 되며, 20~25ms 정도의 ping을 유지할 수 있음을 보였다.

ACKNOWLEDGMENT

이 논문은 2020년도 중소벤처기업부의 기술개발사업 지원에 의한 연구임 (S2949268).

참고 문헌

[1] Microsoft. Azure Kinect DK Hardware Specifications. Available online: https://docs.microsoft.com/en-us/azure/kinect-dk/hardware-specification[accessed May 28, 2021]

[2] Azure kinect body tracking joints. https://docs.microsoft.com/en-us/azure/kinect-dk/body-joints. [accessed May 28, 2021]

[3] https://www.photonengine.com/en-US/sdks#realtime

[4] Microsoft. Azure Kinect Sensor SDK. Available online: https://docs.microsoft.com/en-us/azure/kinect-dk/sensor-sdkdownload [accessed May 28, 2021]

[5] Alp Güler, R.; Neverova, N.; Kokkinos, I. Densepose: Dense human pose estimation in the wild. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18 - 22 June 2018; pp. 7297 - 7306.

[6]Cao, Z.; Hidalgo, G.; Simon, T.; Wei, S.E.; Sheikh, Y. OpenPose: Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21 - 26 July 2017; pp. 7291 - 7299

[7]"Photon Unity Networking". Exit Games https://doc.photonengine.com/en/pun/current/getting-started/pun-intro