Journal of Korea Multimedia Society Vol. 15, No. 6, June 2012(pp. 827-836)

깊이맵 센서를 이용한 3D캐릭터 가상공간 내비게이션 동작 합성 및 제어 방법

성 만 규†

요 약

키넥트의 성공적인 등장 이후 이 센서를 이용하여 사용자의 아바타에 해당하는 3차원 캐릭터의 움직임을 제어하는 많은 인터액티브 콘텐츠가 제작되었다. 하지만, 키넥트의 특성 상 사용자는 키넥트를 정면으로 바라보아야 하며, 모션 또한 제자리에서 수행할 수 있는 동작 정도만으로 국한되었다. 이 단점은 게임에서 가장 중요한 요구기능 중 하나인 가상공간 내비게이션을 수행하지 못하게 하는 근본적인 이유가 되었다. 본 논문은 이와 같은 단점을 해결하기 위한 새로운 방법을 제안한다. 두 단계로 이루어진 본 방법은 첫 번째 단계로서 사용자의 내비게이션 의도를 파악하기 위해 제자리 걷기 동작 제스처인식을 수행한다. 내비게이션 의도가 파악되면, 다음 단계에 현재 제자리 걷기동작을 상체와 하체 모션으로 자동으로 분리한 후, 미리 입력 받은 하체모션캡처 데이터를 현재 캐릭터 속도를 반영하여 수정한 뒤 분리된 원래 하체모션과 자연스럽게 교체한다. 본 논문에서 제안된 알고리즘을 이용하면, 키넥트 센서를 통해 사용자의 상체 모션을 그대로 반영함과 동시에 모션캡처 데이터를 이용하여 하체 동작을 실제 걷는 동작으로 바꾸어주기 때문에 사용자가 조정하는 3차원 캐릭터는 가상공간을 자연스럽게 내비게이션할 수 있다.

3D Character Motion Synthesis and Control Method for Navigating Virtual Environment Using Depth Sensor

Mankyu Sung†

ABSTRACT

After successful advent of Microsoft's Kinect, many interactive contents that control user's 3D avatar motions in realtime have been created. However, due to the Kinect's intrinsic IR projection problem, users are restricted to face the sensor directly forward and to perform all motions in a standing-still position. These constraints are main reasons that make it almost impossible for the 3D character to navigate the virtual environment, which is one of the most required functionalities in games. This paper proposes a new method that makes 3D character navigate the virtual environment with highly realistic motions. First, in order to find out the user's intention of navigating the virtual environment, the method recognizes walking-in-place motion. Second, the algorithm applies the motion splicing technique which segments the upper and the lower motions of character automatically and then switches the lower motion with pre-processed motion capture data naturally. Since the proposed algorithm can synthesize realistic lower-body walking motion while using motion capture data as well as capturing upper body motion on-line puppetry manner, it allows the 3D character to navigate the virtual environment realistically.

Key words: Kinect(키넥트), Motion Synthesis(모션합성), Character Animation(캐릭터 애니메이션), Navigation(내비게이션)

※ 교신저자(Corresponding Author) : 성만규, 주소 : 대전시 서구 둔산동 한마루 아파트 103-506호(302-773), 전화: 053)620-2336, FAX: 053)620-2187, E-mail : mksung@kmu.ac.kr

접수일 : 2012년 3월 12일, 수정일 : 2012년 4월 23일
완료일 : 2012년 6월 13일

† 정회원, 계명대학교 게임모바일콘텐츠학과

1. 서 론

마이크로소프트사는 2010년 11월 자사의 XBOX용 인터페이스 장치인 키넥트를 발표함으로써 새로운 내추럴 사용자 인터페이스를 시도 하였다. 이 장치는 8개월 만에 만개나 팔림으로써 기네스북에 오를 만큼 대성공을 거두었다. 키넥트 센서는 실공간상의 오브젝트와 센서와의 거리 값을 실시간으로 측정하는 깊이맵 센서로서, 구조광(structured light) 방식의 IR(Infrared) 빛을 공간상에 투사한 후, 오브젝트에 반사되어 돌아오는 패턴을 분석하여, 얼마나 원래의 패턴으로부터 이동되어 있는지를 계산함으로써 거리 값을 측정한다[1,2]. 이 센서를 이용하여 취득한 깊이맵 정보를 토대로, 오픈소스 프로젝트인 OpenNI 미들웨어와 키넥트의 원 개발자인 프라임센서사의 드라이버를 이용하면 이 3차원 깊이맵으로부터 15개 조인트의 전역위치와 전역오리엔테이션 값을 구할 수 있으며, 이 데이터를 이용해 3차원 캐릭터를 연동시킬 수 있다[3,4,5].

하지만, 키넥트 센서를 이용하면 사용자의 움직임을 자신의 3차원 캐릭터인 아바타에 그대로 나타낼 수 있는 반면에, 키넥트 센서의 특성 상, 사용자는 항상 센서를 정면으로 마주본 상태에서 일정거리 이상 떨어져서 동작을 취하여야 하며, 되도록 감지영역을 벗어나서는 안 되는 단점이 있다[6,7]. 이 단점은 컴퓨터 게임에서 가장 대표적으로 요구되는 기능인 가상공간 내비게이션을 어렵게 만드는 주요한 원인이 되고 있다[8]. 전통적인 방법으로 사용되는 가상공간 내비게이션은 주로 마우스나 키보드를 통해 제어되는데, 입력제어를 사용자로부터 받으면, 미리 키프레임(key frame)방식을 통해 세밀하게 만들어진 모션 클립들을 자연스럽게 합성(blending)하여 원하는 어느 가상공간에 자신의 캐릭터를 이동시킬 수 있다. 전통적인 이 방법은 구현이 간단한 반면에, 아바타와 사용자의 일체감 측면에서는 단점이 보인다. 반면에, 키넥트와 같은 깊이맵 센서를 캐릭터 애니메이션에 이용하면, 사용자의 동작을 모션캡처한 후 충실하게 아바타에게 표현함으로써 사용자의 동작을 충실히 재현할 수 있는 장점이 있지만, 사용자는 항상 감지영역 안에 존재하여야 하므로, 가상공간 내비게이션을 통해 캐릭터를 이동하는 데 한계가 생긴다[2,6].

본 논문은, 이와 같은 깊이맵 센서를 이용하여 사용자의 움직임을 충실히 반영함과 동시에 장소의 한계를 극복하기 위한 새로운 알고리즘을 제안한다. 두 단계로 이루어진 제안된 알고리즘은 사용자의 가상공간 내비게이션 의도를 실시간으로 인식하기 위한 제자리 걸음 제스처인식(gesture recognition)단계와 모션스플라이싱을 이용한 모션 합성(motion synthesis)단계로 구성되어 있다.

먼저, 제안된 제자리걸음 제스처 인식 단계에서는 3차원 캐릭터의 발이 가상공간의 땅에 닿는 시간 프레임을 제약프레임(constrained frame)으로 정의한 후, 이 제약프레임을 실시간으로 자동 검출한다. 이 때, 땅에 닿는지를 확인하기 위해서는 임계값(threshold) 설정이 필요한데, 본 논문에서는 이를 자동으로 설정하기 위한 새로운 적응적 임계값(adaptive threshold)을 제안한다. 이 방법을 통하면 사용자는 정확한 임계값을 설정하지 않아도, 알고리즘이 스스로 필요에 따라 임계값을 수정함으로써 보다 정확한 검출을 가능케 한다. 검출된 제약프레임은 시공간적 차이를 분석하여 최종적으로 제자리 걷기 동작인지의 여부를 인식한다.

사용자의 내비게이션의도를 파악한 다음 단계로서, 3차원 캐릭터가 가상공간을 자연스럽게 내비게이션 하기 위한 모션합성을 수행한다. 현재 키넥트를 통해 사용자와 연동되어 있는 캐릭터 모션이 제자리 걸음(walking-in-place)이므로, 실제 내비게이션을 위한 걷기모션과는 물리적으로 다른 모션이다. 이와 같은 제자리 걷기모션을 실제 걷기 동작으로 자연스럽게 변환시키기 위해 본 연구는 새로운 모션스플라이싱(motion splicing) 기법을 제안한다.

모션스플라이싱 기술은 사용자의 제자리걸음을 그대로 아바타의 가상공간 내비게이션에 이용하지 못하므로 사용자의 동작을 상체(upper body)와 하체(lower body)로 분리 한 후에, 미리 입력받은 다양한 모션캡처 데이터와의 분석 및 매칭을 통해 상 하체 모션을 결합하여, 이를 3차원 캐릭터에 반영함으로써 자연스럽게 가상공간상을 내비게이션 하는 3차원 캐릭터의 모션을 합성하는 기술이다. 이 기법을 이용하면 상체는 깊이맵 센서의 실시간 동작 데이터를 그대로 이용함으로서, 캐릭터와 사용자의 일체감을 그대로 유지함과 동시에 모션캡처를 통해 미리 저장된 걷기 모션을 하체에 적용함으로서 자연스러운 움

직임 동작을 생성할 수 있다. 그림 1은 제안된 모션합성을 통해 제자리걸음 동작을 자연스러운 걷기 동작으로 변경한 결과를 나타낸다.

제안된 알고리즘은, 깊이맵 센서의 장점인 실시간 모션캡처를 이용하여 상체의 모션을 그대로 캐릭터에 반영함과 동시에, 감지영역의 한계 때문에 가능하지 못한 3차원 캐릭터의 자연스러운 가상공간 내비게이션 모션을 생성할 수 있는 장점이 있으므로, 이를 이용한 다양한 게임에 적용 가능 할 것으로 보인다.

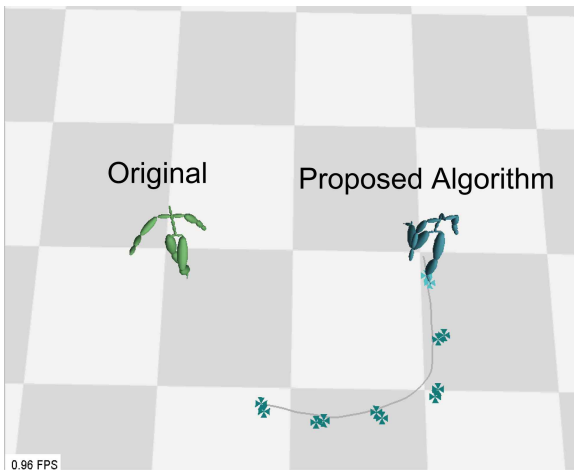

그림 1. 제안된 알고리즘을 이용한 모션 합성 결과(왼쪽: 원 모션, 오른쪽 : 알고리즘 수행결과 모션)

2. 관련연구동향

키넥트의 등장 이후에 키넥트 센서의 원천기술 보유자인 프라임센스사는 오픈소스 프로젝트인 OpenNI와의 협력을 통해 자사의 센서를 OpenNI 미들웨어를 통해 구동하고, 데이터를 받을 수 있도록 드라이버를 공개하였으며, 이를 토대로, 게임을 비롯한 많은 어플리케이션이 개발되어 왔다[9,10,11]. 최근 마이크로소프트사 역시 자사의 SDK공개를 통해 개발자의 다양한 어플리케이션 개발을 유도하고 있다[12].

상용소프트웨어 중에 Activate3D사에서 개발한 Intelligent Character Motion (ICM)기술은 깊이맵 카메라를 이용한 다양한 실시간 캐릭터 제어방법을 제공한다. 이 방법은 제스처 인식을 통해 사용자의도를 파악한 뒤 다양한 시뮬레이션 기법을 통해 가상오브젝트와의 자연스러운 상호작용 동작을 생성할 수 있다. 본 연구와의 차이점으로는, Activate3D는 팔의 움직임을 기준으로 캐릭터의 내비게이션을 제어함에 반하여, 본 논문은 발의 움직임을 기준으로 내비게이션의도를 파악한다는 점에 있다[13]. 본 연구결과의 적용대상인 게임의 경우, 사람의 동작 중 상체동작이 더 중요한 의미를 담고 있는 경우가 많으므로, 본 연구에서는 상체동작을 키넥트와 연동하여 자유도를 주는 것을 채택하였다.

최근에 출시된 Kinect Ultimate Battlefield 3 Simulator의 경우 키넥트를 이용하여 일인칭 슈팅게임에서 다양한 사용자 동작을 실제로 캡처한 후 이를 통해 실제 사용자가 전쟁 상황을 그대로 느끼게끔 하고 있다. 하지만 일인칭 슈팅게임의 경우 사용자의 모션을 삼인칭 시점에서 자세히 나타낼 필요가 없기 때문에, 본 논문의 목적인 자연스럽고 디테일 한 모션합성의 필요성이 덜한 특징이 있다[14]. 그 외에 Dragonball z나 키넥트 Star wars등의 게임 타이틀의 경우 키넥트의 캐릭터 연동기술을 격투장면에 특화 하여 사용하였다.

그 외에 로보틱스 기술 분야에서도 키넥트 기술을 접목하여 휴머노이드 로봇제어에 키넥트를 이용하는 연구 또한 진행되어오고 있다[15].

가상공간상에 자연스럽게 움직이는 캐릭터의 모션을 합성하는 기법은 컴퓨터 애니메이션 연구자들로부터 많은 관심의 대상이 되어 왔다. 모션캡처 데이터를 이용한 데이터 기반(data-driven) 모션 합성 기법들은 대개 모션 블랜딩(blending) 기법을 이용한다[16,17]. 이 방법은 모션 데이터를 사이클 단위로 쪼갠 후 속도 및 방향 값으로 매개화(parameterization)한다. 매개화가 끝난 예제 모션들을 대상으로, 원하는 속도 및 방향에 대해 RBF(Radial Basis Function) 혹은 다양한 가중치값 함수를 통해 가중치를 구한 후 이 가중치를 이용해 모션블랜딩(blending)하는 방법을 취하고 있다[16]. 본 논문에서는 다양한 속도의 하체 모션합성을 위해 RFB기반 모션합성방법을 적용하였다.

이와 더불어, 상체 모션과 합체 모션을 분리한 후에 자연스럽게 섞어 새로운 모션을 생성하는 모션 스플라이싱 기법은 모션 재사용 측면에서 새로운 모션합성 방법으로 제안되었다[18,19]. Heck이 제안한 방법은 상체와 하체를 나눈 후에 하체 모션을 기준으로 상체의 모션을 변경한 방법인 반면, Basten이 제안한 방법은 상・하체 중 어느 모션을 기준으로도

변경할 수 있는 장점이 있다[19]. 이 두 방법 모두, 적절한 원래 오리지널 모션이 아닌 새로운 모션을 상체와 하체에 합체하기 위해서는 두 모션에 대한 시간정규화를 통해 시·공간적인 정렬을 수행하여야 한다. 이 두 방법 모두 미리 캡처되어 있는 모션 데이터를 대상으로 하는 오프라인 방식으로, 본 논문에서 제안하는 온라인 방식에는 그대로 적용할 수 없다. 본 논문에서는 Heck이 제안한 방법을 온라인 형태로 바꾼 새로운 모션 스플라이싱 알고리즘을 제안한다. 이 방법은 제자리걸음을 인식함과 동시에, 속도를 자동으로 검색하고 미리 입력되어 있는 다양한 속도의 걷기 모션 중 적절 한 모션을 선택 후 블랜딩 하여 정확한 속도의 하체 모션을 생성한 후, 이를 키넥트를 통해 실시간으로 연동되고 있는 상체모션과 결합하여 제자리 걷기 모션을 실제적으로 걷기 혹은 뛰는 모션으로 바꾸어 아바타 캐릭터를 내비게이션하도록 하였다.

3. 알고리즘

본 장에서는 본 연구에서 제안한 알고리즘에 대해 자세히 설명한다.

3.1 온라인 캐릭터 연동

3.1.1 조인트 오리엔테이션 계산

깊이맵으로부터 3차원 캐릭터를 연동하기 위하여 계층적인(hierarchical) 캐릭터 스켈레톤 포맷을 이용한다. 상용 3D제작 소프트웨어에서 사용되는 다양한 상용 포맷이 가능하며, 본 연구에서는 바이오비전사의 BVH(Biovision Hierarchy) 포맷을 이용하였고, 사용자의 조인트 오리엔테이션 값을 받기 위해 OpenNI 미들웨어를 사용하였다. OpenNI에서는 15개 조인트에 대한 전역 위치 및 전역 오리엔테이션 값을 제공한다(그림 2). BVH포맷의 계층적 스켈레톤에서는 루트 조인트를 제외하고는 모두 로컬 오리엔테이션(local orientation)을 이용하기 때문에, 전역 오리엔테이션 값을 로컬 값으로 변환할 필요가 있다.

스켈레톤의 포즈 M은 하나의 벡터 $M=\{p^0, q^0, q^1, q^2, q^3, \ldots q^n\}$로 표현할 수 있는데 ($p \in R^3, q \in S^3$(사원수)), p^0는 루트 조인트의 전역 위치를 말하며 q^0는 루트 조인트의 전역 오리엔테이션,

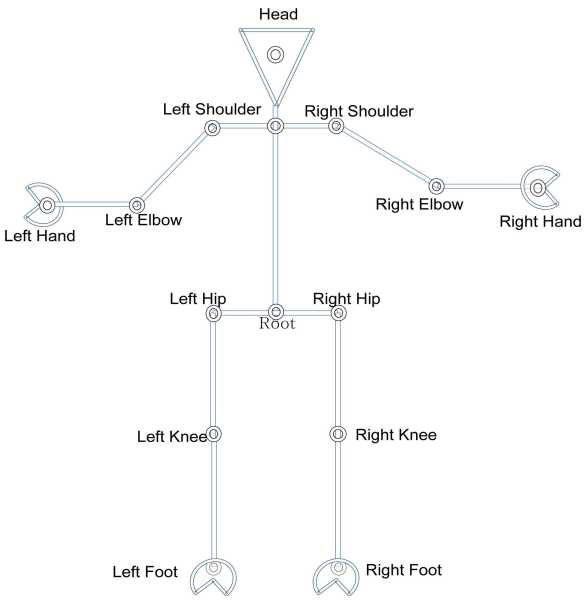

그림 2. 15개 조인트를 가진 계층적 스켈레톤 구조

$q^1 \cdots q^n$은 루트 조인트를 제외한 n개의 조인트의 로컬 오리엔테이션 값을 나타낸다. 특히, p^0는 계층적 구조에서 루트 조인트에 해당하는 값으로서, 가상공간상의 캐릭터의 전체적인 위치를 나타내는 3차원 벡터 값이며, q^0는 가상공간상의 전체적인 캐릭터의 방향을 나타낸다. 만약, 깊이맵 센서를 통해 얻은 n개의 조인트에 대한 오리엔테이션 값을 $Q^0, Q^1, \ldots Q^n$이라고 하고, 캐릭터의 특정 조인트 i의 현재 로컬 오리엔테이션을 q^i, 조인트 i의 부모 조인트 j의 전역 오리엔테이션을 q^j라고 하면,

$$q^i = (q^j)^{-1} \bullet Q^i \quad (1)$$

즉, Q^i와 부모 조인트의 전역 오리엔테이션 값 q^j와의 차이 값을 구한 후(q^{-1}=사원수의 역수), 이 값을 현재 조인트 i의 로컬 오리엔테이션 값으로 설정하면 된다. 루트 조인트의 경우 로컬오리엔테이션과 전역 오리엔테이션에 동일한 값이 들어간다. 이때, OpenNI에서 지원해주는 신뢰값(confidence)을 측정하여 센서로부터 받은 데이터의 정확도를 확인한 후에, 일정 값 이상의 신뢰도를 보일 때에만 설정하도록 하여, 조인트의 꼬임 및 떨림 현상을 줄인다.

3.1.2 캐릭터의 루트 조인트 위치 설정

캐릭터의 루트 조인트의 위치를 나타내는 p^o는 전체적인 캐릭터의 가상공간상의 위치를 나타내는 중요한 정보인데, 본 알고리즘에서는 센서로부터 얻은

루트 조인트 위치정보를 적절하게 스케일링하여 캐릭터가 땅속으로 들어가거나 공중에 떠 있지 않고, 항상 땅 위에 서 있도록 하며, 사용자와 캐릭터가 연동되어 움직일 때도 캐릭터가 항상 적절한 위치로 이동하도록 한다. 즉 캐릭터의 위치인 p^o는 센서 앞에 동작하는 사용자의 깊이맵 상의 전역 위치 값 P^0 정보를 이용하여 설정하되, 현재 연동하고 있는 캐릭터와 사용자의 사이즈 차이를 계산하여 자동으로 스케일링한다. 사용자와 캐릭터의 크기 값 차이를 구하기 위해, 루트 조인트 전역위치와 양 발(Foot) 조인트의 전역 위치 간 거리 값을 이용한다. 키넥트를 통해 얻는 사용자 루트 조인트의 전역위치 및 오른쪽 발 위치와 왼쪽 발 위치를 각각 P^0, P^r, P^l이라고 하고, 가상공간상에서 나타나 있는 캐릭터의 루트 조인트 및 오른쪽, 왼쪽 발 위치를 각각 p^0, p^r, p^l이라고 하면, 스케일 상수 r은 다음의 공식 (2)에 의해 결정된다.

$$r = \frac{(P^0 - P^l)^2 + (P^0 - P^r)^2}{(p^0 - p^l)^2 + (p^0 - p^r)^2} \tag{2}$$

스케일 상수 r을 이용하여, t프레임에 들어오는 깊이맵 상의 사용자 루트 조인트 위치를 P_t^0라고 하고 초기 위치를 P_0^0라고 하면, t프레임에서의 캐릭터 위치인 p_t^0는 식(3)과 같이 계산된다.

$$p_t^0 = p_0^0 + r(P_0^0 - P_t^0) \tag{3}$$

3.2 제약조건 기반 제스처 인식

키넥트 센서를 이용하면 캐릭터는 사용자의 움직임을 실시간으로 따라 하며, 사용자가 공간상을 이동할 때도, 그대로 캐릭터는 가상공간상으로 이동하게 된다. 하지만, 가상공간을 내비게이션 하는 중에 사용자가 감지영역을 벗어나게 되면 조인트의 위치 및 오리엔테이션에 대한 신뢰도 값이 급속도로 떨어지면서 스켈레톤의 꼬임이 발생하게 된다. 본 연구에서는 이를 해결하고자 사용자의 제자리걸음의 의도를 파악한 후, 모션합성을 통해 새로운 내비게이션 모션을 생성하는 방법을 채택한다. 제스처 인식기술은 컴퓨터 비전 및 기계학습 연구자들 사이에 많은 연구가 되어 오고 있다[20]. 하지만 복잡하고, 구현이 어려워 실제적인 사용이 어려운 실정이다. 본 연구에서는 이를 위해 일반적인 제스처 인식이 아닌 본 연구가 목적으로 하는 캐릭터 내비게이션을 위한 새롭고 간단한 제스처 인식방법을 제시한다.

제시되는 알고리즘은 캐릭터의 발이 땅에 닿는 프레임을 제약프레임(constrained frame)으로 정의하고, 자동으로 이와 같은 제약프레임을 찾는 방법을 제안한다. 검출된 제약프레임들은 시공간적 간격 및 순서를 점검하여 최종적으로 제자리걸음임을 인식한다.

제약 프레임을 자동으로 찾는 방법은 두 가지 도메인 상에서 수행할 수 있다. 첫 번째는 그 제약 프레임을 깊이맵 센서를 통해 나오는 데이터를 이용한 센서 도메인 안에서 찾는 방법이고, 두 번째는 이 데이터를 이용해 캐릭터를 연동한 후 캐릭터가 속해져 있는 가상공간 도메인 안에서 찾는 방법이다. 첫 번째 방법에서는 IR(Infrared) 센서가 특성상 조명에 상당히 민감하며 센서의 공간상의 위치에 따라 전혀 다른 데이터가 나오므로, 본 연구에서는 캐릭터 도메인에서 제약프레임을 찾는 두 번째 방법을 선택하였다.

3.2.1 적응적 임계값을 이용한 제약 프레임 검출

현재 프레임이 제약 프레임 즉, 발이 땅에 닿는 프레임인지를 확인하기 위해 제안된 알고리즘은 캐릭터의 발의 위치 및 속도 값을 이용한다. 현재 캐릭터의 왼쪽 발의 3차원 전역 위치를 $p^l(\in R^3)$(이라고 하면, 스칼라 값으로 정의되는 속도 v^l은 다음 공식 (4)에 의해 결정된다.

$$v^l = u\left(\frac{\| p_t^l - p_{t-1}^l \|}{\Delta t}, n\right)$$
$$u(v,n) = rdxl(v,n) \tag{4}$$

이 공식 (4)에서 $\| p_t^l - p_{t-1}^l \|$ 는 t와 t-1프레임사이의 발이 움직인 거리, Δt는 프레임 사이의 시간차이를 의미하며, $rdxl(v,n)$함수는 n번째 숫자에서 반올림되는 값을 구해주는 함수를 의미 한다. 라운드 함수를 쓰는 이유는 값의 민감성을 줄이기 위한 일종의 댐핑(damping)역할을 위해서다. 오른쪽 발의 경우 같은 방식에 의해 계산될 수 있다.

계산된 p^l과 v^l이 동시에 각각 주어진 임계값 $p_{\max}$와 $v_{\max}$안에 동시에 들어오면 해당 프레임을 제약프레임으로 간주한다. 그러므로 정확한 $p_{\max}$와 $v_{\max}$를 설정하는 것은 제약조건 프레임 검출에 상당히 중요한 작업이다. 하지만, 이 값은 어플리케이션에 따라 다르고, 캐릭터마다 세밀한 조정이 필요한 시간적 노력이 요구되므로, 본 연구에서는 이 값을 자동으로

설정하는 적응적 임계값(adaptive threshold)방법을 제시한다. 이 방법은 임의의 초기 값으로 주어진 임계값을 제약조건 판별함수를 실행하면서 매 프레임마다 필요에 따라 알고리즘을 통해 자동으로 수정되는 방식으로 사용자의 세밀한 조정이나 시간적 노력을 덜어줄 수 있다.

이 방법을 가능케 하는 기본 아이디어는 사람이 걷는 동작을 수행 시 반드시 한 발 이상은 땅에 항상 닿아야 한다는 것이다. 그러므로 현재 주어진 임계값을 이용하여 왼발과 오른발에 대한 제약조건 프레임 판별을 수행 후, 두 발 모두 임계값을 벗어나 제약조건이 아닌 것으로 판별이 된다면, 이 임계값은 조정이 되어야 한다는 것을 알 수 있다.

이를 수학적으로 나타내면 다음과 같다, p^l과 p^r을 각각 왼발・오른발의 위치, 그리고 v^l과 v^r을 각각의 속도라고 가정하고, 왼쪽・오른쪽 발에 대한 제약조건 판별 후에 이 두 발 모두 임계값보다 큰 값을 가져 제약조건 프레임이 아닌 걸로 판별된다면 새로운 임계값 $p_{\max}$와 $v_{\max}$는 다음 공식 (5)를 통해 설정된다. 본 연구에서는 δ는 0.1로 설정하였다.

$$\begin{aligned} p_{\max} &= \min(p^r, p^l) + \delta \\ v_{\max} &= \min(v^r, v^l) + \delta \end{aligned} \tag{5}$$

이 방법을 통해 임계값은 자동적으로 필요에 따라 업데이트되며, 이에 따라 제약조건 프레임은 자동으로 판별된다.

3.2.2 제약프레임을 이용한 제자리 걷기 인식

사람의 동작이 제자리 걷기로 판별되기 위해서는 제약프레임의 생성 순서를 확인한 후, 이 제약프레임 사이의 시간적 차이를 확인하여 제스처 인식을 한다. 제약프레임을 시간 축 상에 나타내면 그림 3과 같이 표현할 수 있다.

그림 3에 나타나 있듯이, 사람의 걷기 동작을 포함하는 일반적인 동작 수행 시 제약조건 프레임은 한꺼번에 여러 개의 프레임이 연속적으로 나타나게 된다.

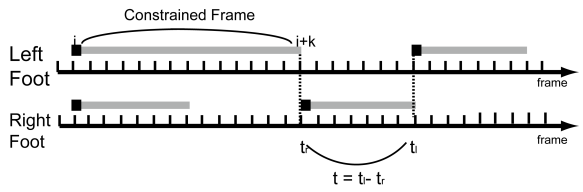

그림 3. 제약조건 프레임

제자리걸음이 이루어지기 위해서는 캐릭터의 양 발이 번갈아 땅에 닿아야 하며, 두 사이의 간격이 일정 시간 미만이어야만 한다. 이 조건을 이용하기 위해 제안된 알고리즘에서는 오른쪽 제약조건 프레임 묶음과 왼쪽 제약조건 프레임 묶음이 일어나는 시작 시간(t_r, t_l)을 기록해 두며, 이 둘 사이의 시간 차이가 t초 차이 미만으로 판정되면 제자리걸음으로 인식하였다. 본 연구에서는 다양한 실험을 통해 t를 2.0초(sec)로 설정하였다.

3.3 모션 스플라이싱

캐릭터의 모션을 생성하기 위해 제약조건 기반 제스처 인식기법을 통해 사용자의 내비게이션의도를 파악한 후, 이 제자리 모션을 일반적인 걷기 모션으로 변환하여 캐릭터에 적용하는 작업을 수행한다. 제자리걸음을 걷기 모션으로 바꾸는 작업은, 상체모션은 깊이맵을 통한 실시간 연동 모션을 그대로 이용하고 하체 모션만을 일반적인 걷기 모션으로 바꾸는 모션 스플라이싱(splicing)을 통해 이루어진다. 그림 4는 이와 같은 모션 스플라이싱 단계를 설명한다. 즉, 깊이맵을 통해 연동된 캐릭터의 원 포즈를 $M = \{p^0, q^0, q^1, q^2, q^3, \ldots q^n\}$ 벡터로 표현한다면, 이 중에 하체에 해당하는 $L = \{p^0, q^0, q^1, \ldots q^k\}$를 모션캡처를 통해 미리 생성해놓은 $\acute{L} = \{p^0, q^0, q^1, \ldots q^k\}$로 교체하여 새로운 포즈 $\acute{M} = \{\acute{L}, q^{k+1}, q^{k+2} \ldots q^n\}$을 생성한다.

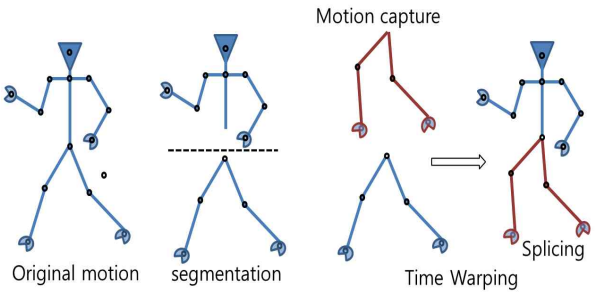

그림 4. 모션스플라이싱

3.3.1 모션캡처 데이터 전처리

본 알고리즘에서는 땅이 발에 닿는 순간 제자리걸음을 인식하기 때문에, 모든 모션 스플라이싱은 한걸음 단위(walking-cycle-motion)로 이루어지도록 하였다. 전처리 단계에서는 현재 연동되고 있는 캐릭터와 동일한 조인트 개수를 가지며 다양한 걷기 속도를

가진 모션캡처 데이터들을 미리 사이클 단위로 쪼개어 메모리상에 한꺼번에 읽어 들인다. 이때 구현의 편의성을 위해 오른발로 시작하는 사이클 걷기 모션과, 왼발로 시작하는 사이클 걷기 모션을 따로 구분하여 저장한다.

3.3.2 모션블랜딩

같은 걸음에도 다양한 속도의 모션이 존재하므로, 모션 스플라이싱을 수행할 때도 제자리걸음의 속도에 정확히 일치하는 걷는 모션을 합체하여야 한다. 만약 속도가 일치하지 않는 하체 모션을 이용할 시 상당히 어색한 걸음걸이가 나타나므로, 본 연구에서는 속도 값(v)을 가중치로 사용한 주기 단위의 예제 모션캡처 데이터 모션 블랜딩기법을 적용한다.

제자리걸음 속도인 v는 프레임 간의 시간적 차이인 Δt와 그 시간 동안 발끝의 움직이는 거리 값인 D를 이용하여 $v = D/\Delta t$ 로 구한다. 주어진 속도 v와 필요로 하는 주기의 유형(오른쪽, 왼쪽 걷기 주기)을 기준으로, n개의 모션에 대해 각각의 가중치(weight)를 계산한다. 본 연구에서는 가중치를 계산하는 방법으로 Sloan이 제안한 산란 데이터 보건법(scattered data interpolation)을 이용한다[16,21]. 이 방법을 통한 가중치 계산 방법은 수식 (6)과 같다.

$$w_m(v) = a_m A(v) + \sum_{j=1}^{n} r_{mj} R_j(v) \tag{6}$$

이 공식에서 $w_m(v)$는 모션 m의 속도 v에 대한 가중치를 나타낸다. $A(v)$와 a_m은 선형 기저 함수(linear basis function) 및 계수, $R_j(v)$ 및 r_{mj}는 방사형기저(radial basis function) 함수 및 계수, 그리고 n은 모션의 개수를 나타낸다. 이 때 예제 모션에 대한 데이터보간(interpolation)이 이루어지기 위해서는 가중치 값을 계산할 때 모션 m 자신이 가진 v에서는 1, 그 외의 모션에 대해서는 0의 가중치를 가져야 한다. 즉, $i = m$인 모션에 대해서는 $w_m(v_i) = 1$ 이어야 하고, 그 외의 $i \neq m$인 모션에 대해서는 $w_m(v_i) = 0$ 을 가져야 한다. a_m과 $R_j(v)$는 Park이 제안한 방법을 각각 이용하여 계산한다. 자세한 구현 사항은 논문[16,21]을 참고한다.

n개의 예제 모션에 대한 가중치를 구한 후, 이 가중치에 따라 모션 블랜딩을 수행하기 전에, 이 예제 모션 사이클을 동일한 시간 축에 정렬하는 작업이 필요하다. 이를 시간 정규화라고 하며, 이렇게 정렬되어 있는 시간을 정규(generic)시간이라고 한다. 주어진 실시간을 f라고 할 때, 정규 시간인 $g(f)$는 수식 (8)에서와 같이 구할 수 있다.

$$g(f) = \left((m-1)\left(\frac{f-k_m}{k_{m+1}-k_m}\right)\right)\frac{1}{n_{k-1}} \tag{8}$$

수식 (8)에서 k_m은 땅에 처음 닿은 프레임을 의미하며 m은 $f > K_m$이 되는 가장 큰 인덱스, 그리고 n_k는 전체 프레임 수를 나타낸다. 이 계산을 통해 모든 시간은 [0 1] 사이로 정규화 된다. 반대로 정규 시간 $g(f)$로부터 원 시간인 f는 다음 수식 (9)에 의해 구할 수 있다.

$$f(g) = ((n_k - 1)g(f) - (m-1)(k_{m+1} - k_m) + k_m \tag{9}$$

이와 같은 시간정규화 과정은 그림 5에 나타나S있다.

정규화 시간 모션을 대상으로 제자리 걷기 모션의 속도인 v를 이용하여, 현재 프레임인 f과 바로 앞 프레임인 $f-1$에서의 정규 시간을 t_f 와 t_{f-1}이라고 하면 $t_f = t_{f-1} + \Delta t_{f-1}$ 로 표현할 수 있다. 이 Δt_{f-1}을 이용하여 각 모션의 새로운 정규시간인 $g(f)$와 대한 원래 시간인 $f(g)$를 구하고, $f(g)$포즈에 대해 식 (6)에서 얻은 가중치 값을 블랜딩하여 새로운 포즈를 합성한다. 즉, 하체의 조인트 $\acute{L} = \{p^0, q^0, q^1, \ldots q^k\}$ 의 각 조인트 앵글 $\dot{q}, 0 \leq i < k$ 는 가중치인 w_m을 이용하여 k개의 사원수 값을 SLERP(Spherical Linear Interpolation)을 연속적으로 수행하여 구한다.

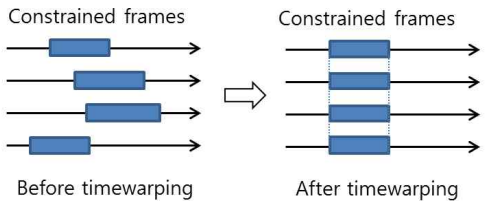

그림 5. 시간정규화 과정

3.3.3 모션변환

사이클 단위로 수행한 예제 모션 블랜딩은 캐릭터의 가상공간상 전역위치와 방향을 나타내는 p^0와 q^0를 제외한 모든 로컬 조인트 오리엔테이션을 대상으로 한다. 캐릭터의 전체적인 위치를 나타내는 p^0값은

제자리걸음의 경우 거의 변화가 없으나, 실제 걸음의 경우, 루트 조인트 위치가 걷기 모션 수행에 따라 변하므로 가상공간상의 실제 위치가 시간 축에 따라 반드시 변화가 있어야 한다. 입력된 모션캡처 데이터의 경우, 데이터 내부에 전역위치인 p^0 데이터가 미리 들어 있기 때문에 이를 현재 가상공간에 맞게끔 2D 변환시킨 후 캐릭터에 적용하여야 한다. 그림 6은 이 단계를 설명한다. 그림 6의 오른쪽에 나타나 있듯이 사이클 단위로 이루어진 하체 모션캡처데이터의 p^0값을 현재 가상공간 아바타 캐릭터의 모션에 2D 변환 후에 적용한다.

이 2D변환에 사용되는 회전 행렬은 현재 센서와 사용자와의 각도차이를 이용해 구한다. 그림 7에 나타나 있듯이 사용자의 루트 조인트 위치와 센서를 잇는 벡터와 현재 캐릭터의 몸체가 가리키고 있는 벡터 사이의 각도 값을 이용하여 2D 회전 행렬을 구한다.

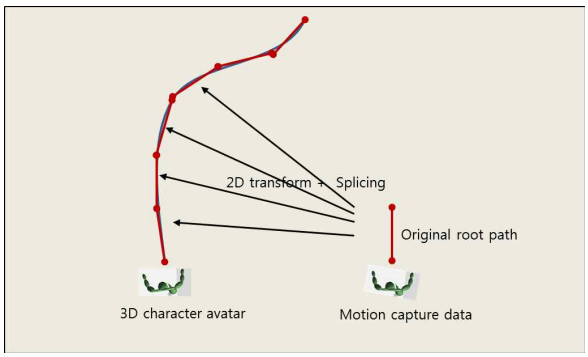

그림 6. 루트 조인트 포지션(p^0) 변환

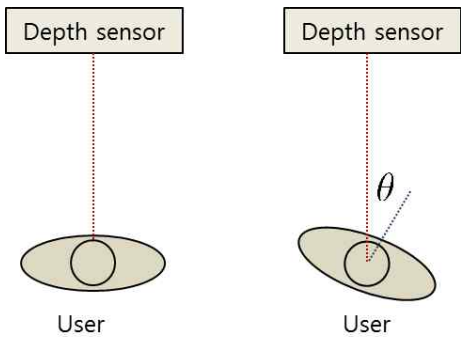

그림 7. 걷기 동작의 회전각 각도 계산

4. 구현 및 적용

알고리즘을 검증하기 위해 그림 8과 같이 마이크로소프트의 키넥트 센서를 이용하여 실시간 캐릭터 연동을 실험하였다. 특히, 본 연구에서 제안한 모션 스플라이싱을 위해 10개의 다양한 속도를 가진 걷기 동작 모션캡처 데이터를 사이클 단위로 나눈 후에, 오른발로 시작하는 사이클 걷기 모션 5개와 왼발로 시작하는 사이클 걷기 모션 5개를 나누었다. 단위시간 동안 발끝의 움직이는 거리를 통한 평균 모션 속도를 계산하면 다음 표 1과 같다.

표 1. 입력 모션 속도

번호	모션 사이클 타입	속도(unit/sec)
1	오른발	7.57
2	오른발	8.05
3	오른발	9.21
4	오른발	10.02
5	오른발	12.21
6	왼발	7.54
7	왼발	8.06
8	왼발	9.22
9	왼발	10.00
10	왼발	12.01

그림 8은 실시간 캐릭터 연동의 모습을 나타낸다. 사용자가 제자리걷기 모션을 취하면 3차원 캐릭터도 사용자의 모션을 그대로 반영한다. 제자리걸음에 대한 제스처 인식을 확인하기 위하여 캐릭터의 발이 땅에 닿는 순간 발 조인트의 색깔을 다르게 표시하였다. 제약조건 파악을 통한 제자리걸음 인식을 캡처한 화면은 그림 9와 같다. 본 실험으로 모션스플라이싱을 통한 하체 모션 합성을 통해 제자리걸음을 실제 걷기 모션으로 바꾼 모습은 그림 10과 그림 11에 나타나 있다. 그림 8과 그림 12에서 나타나 있듯이 상체 부분은 실시간 연동이 계속되고 있으므로 사용자는 걷는 도중에 자유롭게 상체를 사용할 수 있으며 이와 동시에 하체 동작은 모션캡처 데이터를 그대로 이용

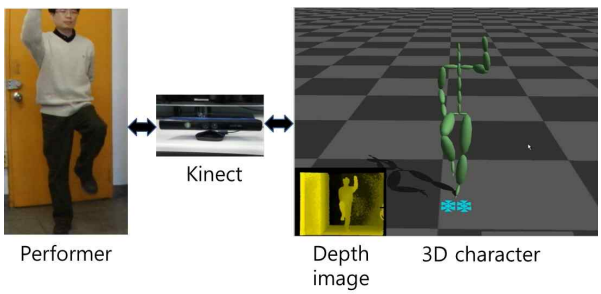

그림 8. 실시간 캐릭터 연동

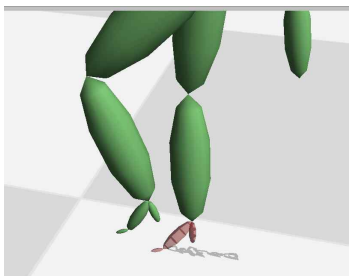

그림 9. 제약조건 파악을 통한 제자리걸음 인식

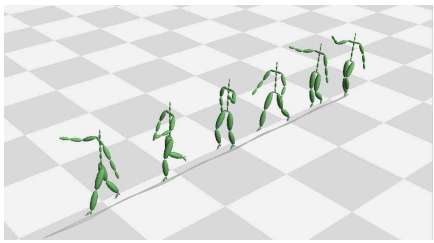

그림 10. 모션 스플라이싱을 통한 가상공간 내비게이션[직선]

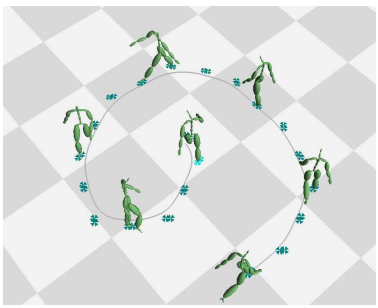

그림 11. 모션 스플라이싱을 통한 가상공간 내비게이션[커브]

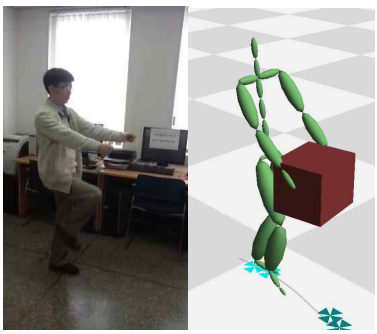

그림 12. 자유로운 상체움직임과 하체 스플라이싱을 통한 내비게이션

하므로 자연스러운 걷기 동작을 생성할 수 있다.

5. 결과 및 토론

본 논문에서는 키넥트와 같은 깊이맵 센서를 이용한 캐릭터 연동 시 불가능했던 가상공간 내비게이션을 가능하게 하기 위한 제스처 인식 및 모션 스플라이싱을 통한 실시간 모션 합성 알고리즘을 제안하였다. 본 알고리즘에서는 사람의 발이 땅에 닿는 프레임을 제약조건 프레임으로 간주하고 이를 자동으로 판별하기 위한 적응적 임계값 방법을 제안하였고, 제자리걸음 모션을 실제 걷기 모션으로 바꾸기 위해 상체와 하체로 모션을 나누고, 하체 모션의 속도를 입력하여 미리 준비된 여러 속도의 모션캡처 하체 모션을 모션 블랜딩하여 사용 하였다. 본 논문에서 제안한 방법은 다양한 실험을 통해, 실제로 제자리걸음을 실제 걸음으로 바꾸어 가상공간을 내비게이션 할 수 있음을 확인하였다. 이러한 방법의 장점으로는 상체 모션은 깊이맵 센서로부터 실시간 연동되므로 자유로운 움직임이 가능하며, 동시에 하체모션은 모션캡처 데이타를 이용해 실시간 합성되므로 자연스럽게 가상공간을 내비게이션하도록 제어할 수 있다.

본 논문에서 제안한 방법 중 적응적 임계값 방법은 두 발 중 한 발은 항상 땅에 닿아 있는 걸로 간주했기 때문에 제자리 걷기 동작에 적합하다. 하지만, 만약 뛰는 동작을 수행 시, 두 발이 땅에 닿지 않는 프레임이 존재하기 때문에 본 알고리즘을 그대로 사용할 수 없는 단점이 있다. 하지만, 알고리즘에 대한 약간의 수정을 통해 두 발의 위치가 일정 높이 이상 땅으로부터 떨어져 있으면 자동으로 뛰는 동작으로 간주할 수 있으므로, 쉽게 문제를 해결할 수 있다. 또한, 본 논문에서 제안한 제스처 인식 방법은 땅에 닿은 제약프레임을 이용하기 때문에 인식자체는 연속적으로(continuous)하지 않고 이산적인(discrete) 특징이 있다. 이를 해결하기 위해 추후 연구에서는 제자리 걷기 속도인 v값을 이용하여 일단 제자리걸음이 시작된 걸로 인식이 되면 v가 일정 속도 범위 안에 있으면 계속 걷기동작으로 간주하도록 바꾸어 좀 더 부드러운 모션 합성을 수행할 수 있을 것으로 본다. 또한 추후에 제안된 방법을 [22]에서 제안한 지능형 가상공간과 결합하여 자연스러운 가상공간과의 상호작용을 수행할 수 있으리라 본다.

참 고 문 헌

[1] Microsoft Kinect Teardown, http://www.ifixit.com/Teardown/Microsoft-Kinect-Teardown/4066/, 2010

[2] T. Carmody, How Motion Detection Works in XBOX Kinect, http://www.wired.com/ gadgetlab/2010/11/tonights-release-xbox- kinect-how-does-it-work, 2010.
[3] PrimeSense, *NITE Algorithms notes*, PrimeSense, 2010.
[4] PrimeSense, *NITE Control Programmer's Guide notes*, PrimeSense, 2010.
[5] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake, "Real-Time Human Pose Recognition in Parts from Single Depth Images," *Proc. of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 1297-1304, 2011.
[6] T. Leyvand, C. Meekhof, Y. Wei, J. Sun, and Baining Guo, "Kinect Identity: Technology and Experience," *IEEE Computer*, Vol.44, Issue 4, pp. 94-96, 2011.
[7] Primesensor datasheet, http://www.primesense.com/en/press-room, 2010
[8] D. Nieuwenhuisen, A. Kamphuis, and M.H. "High Quality Navigation in Computer Games," *Science of Computer Programming: Special Issue on Aspects of Game Programming*, Vol. 67, Issue 1, pp. 91-104, 2007.
[9] Kinect Game list, http://en.wikipedia.org/wiki/List_of_Kinect_games, 2010
[10] Open Kinect, http://openkinect.org, 2010.
[11] OpenNI, *OpenNI™ User's Guide*, 2010.
[12] Microsoft Kinect SDK, http://www.microsoft.com/en-us/kinectforwindows/develop/, 2012.
[13] Activate3D Intelligent Character Motion, http://activate3d.com/, 2012.
[14] Battlefield3, http://www.battlefield.com/battlefield3, Electronic Art, 2012.
[15] Biped fobo, http://www.projectbiped.com/prototypes/fobo, 2011
[16] S.I. Park, H.J. Shin, and S.Y. Shin, S "On-Line Locomotion Generation Based on Motion Blending," *Proc. of ACM SIGGRAPH/ Eurographics Symposium on Computer Animation*, pp. 105-111, 2002.
[17] A. Treuille, Y. Lee, and Z. Popović. "Near-Optimal Character Animation with Continuous Control," *ACM Transactions on Graphics*, Vol.26, Issue 3, 2007.
[18] R. Heck, L. Kovar, and M. Gleicher. "Splicing Upper-Body Actions with Locomotion," *Proc. of Eurographics*, pp. 459-466, 2006.
[19] B.J. H. van Basten and A. Egges "Flexible Splicing of Upper-Body Motion Spaces on Locomotion," *Computer Graphics Forum*, Vol.30, Issue 7, pp. 1963-1971, 2011.
[20] S. Mitra and T. Acharya, "Gesture Recognition: A Survey," *IEEE Trans. on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, Vol.37, Issue 3, pp. 311- 324, 2007.
[21] C. Rose, M. Cohen, and B. Bodenheimer. "Verbs and Adverbs: Multidimensional Motion Interpolation," *IEEE Computer Graphics and Application*, Vol.18. Issue 5, pp. 32-40, 1998
[22] 이윤길, 김성아, 최진원, "가상캐릭터의 자율주행을 위한 지능형 가상 환경 구축," 멀티미디어학회논문지, 제13권, 제3호, pp. 34-42, 2009.

성 만 규

1993년 2월 충남대학교 전산학과 공학사
1995년 2월 충남대학교 전산학과 석사
2005년 12월 University of Wisconsin-Madison 석사
2005년 12월 University of Wisconsin-Madison 박사
1995년 1월-2000년7월 한국전자통신연구원(ETRI) 연구원
2006년 2월-2012년2월 한국전자통신연구원(ETRI) 선임 연구원
2012년 3월-현재 계명대학교 게임모바일콘텐츠학과 교수
관심분야 : 컴퓨터 그래픽스, 컴퓨터 애니메이션, 인터랙티브 콘텐츠, HCI