한국컴퓨터그래픽스학회 Vol. 14, No. 2, P. 9~17

모션 면적을 이용한 버추얼 카메라의 자동 제어 기법

권지용° 이인권

연세대학교 컴퓨터과학과

mage@cs.yonsei.ac.kr°, iklee@cs.yonsei.ac.kr

Automatic Virtual Camera Control Using Motion Area

Ji-yong Kwon° In-Kwon Lee

Department of Computer Science, Yonsei University

요 약

본 논문에서는 캐릭터의 모션을 고려한 카메라의 파라미터를 결정하는 방법을 제안한다. 기본적인 아이디어는 캐릭터의 각각의 링크가 화면상의 평면에 투영되었을 때의 움직임을 면적으로 계산하여 이를 활용하는 것이다. 우리는 이러한 면적을 '모션 면적'이라고 정의하였다. 모션 면적을 이용하면 실시간 혹은 오프라인 상에서 캐릭터의 모션에 대한 적절한 카메라 경로 혹은 고정된 카메라의 위치를 결정할 수 있다. 우리는 실험을 통해서 제안하는 방법으로 만들어진 카메라 경로가 부드럽게 움직임과 동시에 캐릭터의 모션을 보다 역동적으로 보이게끔 한다는 사실을 관측하였다. 또한 제안하는 방법은 카메라의 시선이 장애물과 충돌하지 않도록 하는 제어 기법과 쉽게 합쳐져서 사용될 수 있다. 우리는 또한 제안하는 방법이 역동적으로 움직이는 캐릭터를 포함하는 장면에서의 시각적 품질을 측정하는 도구로써 다른 일반적인 카메라 제어 기법과 함께 쓰일 수 있을 것으로 기대한다.

Abstract

We propose a method to determine camera parameters for character motion, which considers the motion by itself. The basic idea is to approximately compute the area swept by the motion of the character's links that are orthogonally projected onto the image plane, which we call *"Motion Area"*. Using the motion area, we can determine good fixed camera parameters and camera paths for a given character motion in the off-line or real-time camera control. In our experimental results, we demonstrate that our camera path generation algorithms can compute a smooth moving camera path while the camera effectively displays the dynamic features of character motion. Our methods can be easily used in combination with the method for generating occlusion-free camera paths. We expect that our methods can also be utilized by the general camera planning method as one of heuristics for measuring the visual quality of the scenes that include dynamically moving characters.

키워드: 카메라 제어, 카메라 계획, 모션 탐색, 모션 면적

Keywords: camera control, camera planning, motion exploration, motion area

1 서론

대부분의 3D 그래픽스 응용 프로그램들이 가상 카메라 모델을 통해 물체가 어떤 형태로 화면에 보여질지 결정하게 되기 때문에, 3차원 가상환경에서 카메라 제어 문제는 매우 중요하다. 따라서 근래 많은 연구자들이 카메라를 제어하는 기법에 대한 연구를 하였다. 이중 몇몇 기법들은 사용자들로 하여금 직접적으로 카메라를 제어하는 방법들이며, 다른 몇몇 기법들은 자동적으로 카메라를 구성하는 파라미터를 결정하는 기법들이다 [1]. 특히 자동화된 카메라 계획과 주어진 장면의 시각적 품질 측정을 위해 다양한 기법들이 연구되었다.

그러나 우리가 아는 한에서 아직까지 주어진 장면에서의 캐릭터의 움직임을 고려한 카메라 제어 기법은 없었다. 기존에 제안된 기법들은 캐릭터의 움직임보다는 주어진 장면의 의미를 고려하여 사용자가 정의한 제약이나 영화 촬영 기법 등을 이용하였다. 그러나 캐릭터의 움직임은 카메라를 제어하기 위해서 반드시 고려해야 할 요소인데, 그 이유는 다음과 같다. 첫 번째로 카메라의 위치 및 각도에 따라 다양한 형태로 움직임이 보이기 때문이며, 두 번째로 간혹 3D 게임이나 3D 장면 탐색기 등과 같은 프로그램에서는 장면의 시나리오보다는 움직임 그 자체가 훨씬 중요하기 때문이다. 이러한 이유 때문에 본 연구는 '캐릭터의 움직임 만을 고려하여 좋은 카메라 파라미터를 얻을 수 있는 양적인 측정 기법이 존재하는가'라는 질문에서 연구의 동기를 얻었다.

본 논문에서는 이와 같은 질문을 해결하기 위한 방법을 제안한다. 가장 기본적인 아이디어는 캐릭터의 링크가 화면상에 투영되었을 때 움직임이 만들어내는 면적을 측정하는 것으로, 우리는 이러한 면적을 '모션 면적'이라고 칭하였다. 즉 모션 면적의 값이 클수록 더 많은 움직임이 카메라에 담기는 것이다. 달리 말하면 우리는 모션 면적의 값을 이용하여 주어진 카메라 파라미터에서 캐릭터의 움직임이 얼마나 많은지를 측정할 수 있다. 따라서 모션

면적은 캐릭터의 움직임을 포함한 장면에 대하여 카메라 파라미터를 결정하는 데 있어 좋은 요소로 사용될 수 있으며, 또한 캐릭터의 모션 탐색 등과 같은 캐릭터의 움직임 만을 고려해야 하는 경우에 대해 좋은 뷰 포인트를 선택하는 데 이용될 수 있다.

본 논문의 나머지 부분은 다음과 같이 구성되어 있다. 먼저 2절에서 관련 연구들을 살펴보고, 3절에서는 모션 면적에 대해 자세히 소개하겠다. 4절에서 모션 면적을 이용하여 어떻게 카메라를 제어하는지에 대하여 알아본 다음, 5절에서 제안한 방법의 구현 결과를 소개하고 각 방법에 대하여 비교하겠다. 다음으로 6절에서 원근 투영을 이용할 경우의 모션 면적 계산법에 대하여 정리한 뒤, 마지막으로 7절에서 본 논문에서 제안한 방법에 대해 정리하고 결론을 내리고자 한다.

2 관련 연구

Christie 등[2]은 그들의 논문에서 최근의 자동화된 카메라 계획 기법들에 대한 동향을 소개하였다. 그들은 자동화된 가상 카메라 제어 기법은 크게 제어할 카메라가 배치될 장면의 속성을 정하는 방법과 이러한 속성을 이용하여 문제를 해결하는 방법을 기준으로 분류할 수 있다고 설명하였다. 본 논문에서는 시각적 품질의 측정을 위해서 사용된 기준을 바탕으로 관련 연구들을 정리하여 소개하겠다.

많은 연구들이 사용자가 정의한 제약 조건을 만족시키는 방향으로 카메라를 제어하는 기법들을 제안하였다. Blinn[3]은 주어진 장면과 화면 상에 캐릭터가 보여져야 하는 위치 등을 만족하는 저수준의 카메라 파라미터를 계산하기 위해서 대수학적인 풀이 방법을 도입하였다. Drucker와 Zeltzer[4]가 제안한 'CamDroid' 시스템은 여러 가지의 그래픽 환경에 적용할 수 있는 카메라 제어를 가능하도록 하였는데, 먼저 몇몇의 제약 조건을 만족하는 개별적인 장면을 위한 최적의 카메라 위치를 선별한 뒤, 최적화 기법을 이용하여 주어진 장면을 촬영하기 위해 카메라 파라미터를 자동적으로 수정하도록 하였다. Bares 등[5]은 사용자가 요구하는 카메라의 다양한 관측 조건을 만족시키는 해를 구하기 위해 제약조건 해결기(constraint solver)를 이용하였다. 한편 Halper와 Olivier[6]는 적절한 카메라의 위치를 계산하기 위하여 유전자 알고리즘을 적용하였다. Halper 등[1]이 제안한 컴퓨터 게임을 위한 카메라 엔진은 특히 매 프레임 간의 연속성과 제약 조건의 만족 사이의 균형을 해결하는데 주안점을 두었다. Gleicher와 Witkin[7]은 이미지에서의 특징을 가지고 카메라를 제어할 수 있도록 하는 'through-the-lens' 카메라 제어 기법을 제안하였고, Kyung 등[8]이 이를 제약이 있는 비선형 역계산 문제로 설정하여 이들의 방법을 개선하였다. Christie 등[9]은 구조적으로 이진 공간 분리 기법과 비슷한 의미론적 공간 분할 기법을 제안하여 3차원 장면의 시각적 특성을 고려한 같은 조건을 만족하는 해집합을 분리하여 사용하고자 하였다.

한편 많은 수의 영화 촬영 기법 기반의 연구 또한 진행되었다. He 등[10]이 제안한 'Virtual Cinematographer' 시스템이 대표적으로 이러한 영화 촬영 기법을 이용하였다. 이 시스템은 장면과 장면 간의 전환을 유한 상태 기계의 상태와 전이로 바꾸어 구성한 것이다. Christianson 등[11]은 몇 가지의 영화 촬영 기술을 체계화한 선언형 언어를 제안하였다. 한편 Tomlinson 등[12]은 행위 기반의 자동적 가상 영상 촬영 시스템을 제안하였는데, 이들이 제안한 시스템은 카메라를 일종의 인공 생명체로 보고 이 생명체가 동기적인 욕구가 있어서 감성적인 내용을 증강시키기 위해 카메라와 빛을 조절하도록 하였다. Kennedy와 Mercer[13]는 이와 같은 문제를 해결하기 위해 전문가 시스템을 도입하였다. Lin 등[14]은 앞에서 설명한 제약 조건의 개념과 영화 촬영 기술의 개념을 모두 사용하는 실시간 3D 게임을 위한 카메라 모듈을 제안하였다.

몇몇의 연구자들은 좋은 시각적 품질을 측정하고 카메라 파라미터를 얻기 위해 그들이 직접 개발한 방법을 이용하기도 하였다. Gooch 등[15]은 몇 개의 인지심리학 분야에서의 연구 결과로부터 카메라의 시점을 결정하는데 요긴하게 쓰일 수 있는 'Canonical viewpoint'라고 불리는 원칙을 정리하고, 이와 더불어 예술가들이 좋은 시점을 얻기 위해 사용하는 몇 가지 휴리스틱과 혼용하여 카메라를 제어하는 방법을 제안하였다. Sokolov와 Plemenos[16] 주어진 물체의 보여지는 면에서의 곡률을 기반으로 하는 카메라 시점의 품질을 측정하는 기법을 제안하였다. 그들은 이 방법을 이용하여 좋은 카메라 시점을 찾는 한편, 장면 탐색을 위한 좋은 카메라 경로를 찾는 데에도 이용하였다[17]. Lee 등[18]은 이미지 특징점과 개념이 유사한 'Mesh Saliency'라고 하는 개념을 이용하여 이를 좋은 카메라 시점을 찾는데 사용될 수 있음을 밝혔다.

앞에서 언급한 카메라 제어 기법들은 주어진 장면과 상황에 대해 적절한 카메라 파라미터를 자동적으로 생성할 수 있으나, 캐릭터의 움직임에 대한 고려는 특별히 없다. 본 논문은 관절을 가진 캐릭터의 움직임에 대한 시각적 품질을 측정하는 방법을 제안하였다. 제안하는 방법은 시각적 품질을 측정하는 여러 가지 휴리스틱 중 하나로 생각될 수 있으며, 따라서 다른 카메라 계획 기법들과 쉽게 혼용될 수 있다. 더불어서 모션 탐색기와 같은 캐릭터의 움직임만을 고려해야 하는 상황에서는 이러한 측정 기법이 좋은 카메라 시점을 선택할 수 있는 효과적인 기준이 될 수 있다.

3 모션 면적의 측정

모션 면적은 캐릭터의 모든 링크들이 화면 위로 투영하였을 때 화면 위를 지나는 면적으로 정의 된다. 이를 계산하기 위해서는 카메라의 투영 행렬과 캐릭터의 움직임을 이용해야 한다. 본 논문에서는 기본적으로 직교 투영의 경우만을 고려하고 있다. 이는 본 방법이 오직 직교 투영만을 사용하는 응용 프로그램에만 적용할 수 있다는 의미가 아니라, 계산 방법의 단순화를 위한 원근 투영의 근사화로써 직교 투영을 사용한다는 것이다. 이러한 방법은 원근 투영에서 상대적인 모션 면적을 계산할 수 있음을 주목해야 한다. 만일 원근 투영에서 다양한 카메라-목표 간의 거리값과 FOV(field of view)값을 이용할 경우, 카메라-목표 간의 거리값이 작아지고 FOV값이 크면 클수록 당연히 화면에 투영된 절대적인 모션 면적은 커지게 되므로, 원근 투영을 이용한 절대적인 모션 면적은 효과적으로 사용될 수 없다. 직교 투영을 이용할 경우 투영 평면을 결정짓는 요소는 오직 투영 평면의 법선 방향, 즉 뷰 방향이다. 따라서 뷰 방향 벡터 $\mathbf{v}$를 통해 투영 행렬 $\mathbf{P}(\mathbf{v})$을 구하는 식은 $\mathbf{P}(\mathbf{v}) = \begin{bmatrix} \mathbf{v}_1 & \mathbf{v}_2 \end{bmatrix}^T$ 으로 표현될 수 있다. 여기서 $\mathbf{v}_1$과 $\mathbf{v}_2$은 $\mathbf{v}$과 수직으로서 투영 평면의 기저를 이룬다.

캐릭터 C를 두 개의 집합 $\{J, L\}$으로 정의하자. 여기서 J는 캐릭터를 구성하는 N개의 관절들로 이루어진 집합으로써 $\{j_i \mid i \in [1, N]\}$으로 정의되고, L은 캐릭터를 구성하는 각 관절의 연결 구조를 기술하는 집합으로써 $\{(a, b) \mid a, b \in [1, N]\}$과 같이 정의되는데, L의 한 원소 (a, b)는 관절 j_a와 관절 j_b가 서로 캐릭터의 계층 구조상에서 부모와 자식으로 연결되어 링크 (j_a, j_b)를 이루고 있음을 의미한다. 각각의 관절 j_i는 시간 $t \in [0, T]$를 공간상의 한 좌표로 매핑하는 곡선 $j_i(t)$로 생각할 수 있다. 그림 1에서 보는 바와 같이, 링크로 연결된 한 쌍의 커브 $j_a(t)$와 $j_b(t)$, 그리고 특정 시간 t_0과 t_1에서의 링크로부터 얻어지는 선분은 $A_{(a,b)}(t_0, t_1)$만큼의 면적을 가진 곡면 $S_{(a,b)}(t_0, t_1)$을 구성하게 된다. 모션 면적 $A(\mathbf{v}, t_0, t_1)$은 따라서 다음과 같이 $\mathbf{P}(\mathbf{v})$을 기저로 가지는 투영 평면 위에서 곡면 $S_{(a,b)}(t_0, t_1)$들이 만들어내는 면적의 합으로

계산될 수 있다.

$$\mathcal{A}(\mathbf{v}, t_0, t_1) = \frac{1}{t_1 - t_0} \sum_{(a,b) \in L} A^{\mathbf{v}}_{(a,b)}(t_0, t_1). \quad (1)$$

여기서 $A^{\mathbf{v}}_{(a,b)}(t_0, t_1)$는 링크 (j_a, j_b)가 투영행렬 $\mathbf{P}(\mathbf{v})$을 통해 투영된 후 시간 t_0에서 t_1까지 만들어내는 곡면 $S_{(a,b)}(t_0, t_1)$의 면적을 말한다. 이렇게 계산된 모션 면적 값을 시간 간격으로 나누어진 값을 통해 우리는 시간당 평균 모션 면적 값을 계산할 수 있다.

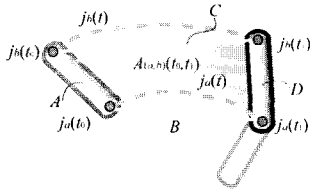

그림 1: 모션 면적의 개념도. 캐릭터의 링크가 투영된 화면 상에서 훑고 지나가는 영역의 면적을 의미한다.

$j_i(t)$가 $\mathbf{P}(\mathbf{v})$을 통해 투영된 곡선을 $p^{\mathbf{v}}_i(t) = \mathbf{P}(\mathbf{v})j_i(t)$ 라고 하자. 만일 $|t_1 - t_0|$의 크기가 충분히 작다면 $A^{\mathbf{v}}_{(a,b)}(t_0, t_1)$의 계산은 그림 1에서 녹색 사각형으로 표시된 바와 같이 네 개의 점 $p^{\mathbf{v}}_a(t_0)$, $p^{\mathbf{v}}_a(t_1)$, $p^{\mathbf{v}}_b(t_0)$, 그리고 $p^{\mathbf{v}}_b(t_1)$이 만들어내는 사각형의 면적으로 간소화될 수 있다. 사각형의 면적은 두 개의 삼각형의 면적으로 바꾸어 계산될 수 있으므로, $A^{\mathbf{v}}_{(a,b)}(t_0, t_1)$는 다음 식과 같이 계산된다.

$$A^{\mathbf{v}}_{(a,b)}(t_0, t_1) = \frac{1}{4}\left(|A^T\mathbf{P}(\mathbf{v})^T\mathbf{X}\mathbf{P}(\mathbf{v})B|^2 + |C^T\mathbf{P}(\mathbf{v})^T\mathbf{X}\mathbf{P}(\mathbf{v})D|^2\right). \quad (2)$$

여기서 $A = p_b(t_0) - p_a(t_0)$, $B = p_a(t_1) - p_a(t_0)$, $C = p_b(t_0) - p_b(t_1)$, $D = p_a(t_1) - p_b(t_1)$, 이고 $\mathbf{X} = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$이다. 또한 삼각형 면적의 절대값 대신 제곱된 면적을 사용한 것은 계산의 편의성을 위한 것임을 명한다. 본 논문에서 실험에 사용한 캐릭터의 모션 데이터는 초당 30개의 샘플로 이루어져 있었기 때문에 우리는 $|t_1 - t_0|$을 $\frac{1}{30}$초로 고정하였다.

삼각형의 면적은 삼각형의 두 변을 이루는 벡터의 외적 크기의 절반으로 계산될 수 있으므로, 식 (2)는 다음과 같이 바뀔 수 있다.

$$A^{\mathbf{v}}_{(a,b)}(t_0, t_1) = \frac{1}{4}\left(|(A \times B) \cdot \mathbf{v}|^2 + |(C \times D) \cdot \mathbf{v}|^2\right). \quad (3)$$

위의 식 (3)는 다음과 같이 뷰 방향 벡터 $\mathbf{v}$에 대한 제곱식으로 간소화될 수 있다.

$$\begin{aligned} A^{\mathbf{v}}_{(a,b)}(t_0, t_1) &= \frac{1}{4}\left(|(A \times B) \cdot \mathbf{v}|^2 + |(C \times D) \cdot \mathbf{v}|^2\right) \\ &= \frac{1}{4}\mathbf{v}^T\left((A \times B)(A \times B)^T + (C \times D)(C \times D)^T\right)\mathbf{v} \\ &= \mathbf{v}^T\mathbf{A}_{(a,b)}(t_0, t_1)\mathbf{v}. \end{aligned} \quad (4)$$

따라서 모션 면적 $\mathcal{A}(\mathbf{v}, t_0, t_1)$ 또한 다음과 같이 제곱식으로 표현될 수 있다.

$$\begin{aligned} \mathcal{A}(\mathbf{v}, t_0, t_1) &= \frac{1}{t_1 - t_0} \sum_{(a,b) \in L} A^{\mathbf{v}}_{(a,b)}(t_0, t_1) \\ &= \frac{1}{t_1 - t_0}\mathbf{v}^T\left(\sum_{(a,b) \in L} \mathbf{A}_{(a,b)}(t_0, t_1)\right)\mathbf{v} \\ &= \frac{1}{t_1 - t_0}\mathbf{v}^T\mathbf{A}(t_0, t_1)\mathbf{v}. \end{aligned} \quad (5)$$

우리는 이렇게 계산된 모션 면적 값이 크면 클수록 화면 상에 투영된 캐릭터의 움직임이 더욱 역동적이 될 것이라는 기대를 할 수 있다. 따라서 모션 면적은 뷰 방향에 대한 화면의 역동성의 품질을 측정하는 도구로 사용될 수 있을 것이다.

4 모션 면적을 이용한 자동적 카메라 제어 방법

본 절에서는 3에서 소개된 모션 면적을 활용하여 카메라를 제어하는 기법들을 소개한다. 본 절에서 제안하는 모든 기법들은 오직 캐릭터의 움직임만을 고려하므로, 주어진 모션의 의미나 영화 촬영 기법 상의 효과를 고려해야하는 상황에서는 적절한 해를 구할 수 없을 가능성이 있다. 그럼에도 제안하는 방법들은 모션 탐색기와 같은 몇몇 응용 프로그램에서 역동적인 뷰 방향을 찾아내는데 효과적으로 사용될 수 있다.

4.1 고정된 카메라의 파라미터 결정

주어진 캐릭터 모션이 일어나는 전체 시간에 대한 모션 면적을 최대화하는 최적화 문제를 해결하면, 캐릭터의 모션을 가장 역동적으로 보이게끔 하는 고정된 카메라의 파라미터를 계산할 수 있다. 식 (6)은 이러한 최적화 문제를 수식으로 표현한 것이다.

$$\text{maximize } O(\mathbf{v}) = \mathbf{v}^T\mathbf{A}(0, T)\mathbf{v} \text{ subject to } \mathbf{v}^T\mathbf{v} = 1. \quad (6)$$

식 (6)은 $\mathbf{A}(0, T)$의 고유값중 가장 큰 고유값을 갖는 고유벡터를 구하는 방법을 통해 쉽게 해결할 수 있다[19].

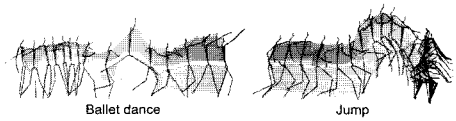

그림 2: 고정된 카메라의 파라미터를 계산한 경우의 예.

그림 3: Swinging motion example.

그림 2는 앞서 설명한 방법을 통해 고정된 카메라의 파라미터를 계산한 두가지 예를 보여준 것이다. 여기서 시간의 흐름은 적색에서 청색으로 변하는 색상의 변화로 표현되었다. 본 그림을 통해서 알 수 있듯이 모션 면적을 최대화하는 방법을 이용하면 캐릭터의 움직임을 관측하는데 유용한 카메라의 뷰 방향을 쉽게 계산할 수 있다.

동적인 장면의 효과적인 카메라 파라미터 계산을 위해서 다른 기준이 사용될 수도 있다. 캐릭터의 바운딩 박스를 이용하는 것이 대표적으로 사용될 수 있는 기준이다[17]. 그러나 화면 상에

바운딩 박스의 면적을 최대화하는 카메라의 뷰 방향을 사용할 경우 주어진 장면에 적절한 카메라 파라미터를 얻지 못할 가능성이 있다. 간단한 예로, 어떤 캐릭터가 그림 3과 같이 검을 횡방향으로 휘두르는 동작을 취한다고 가정했을 때, 만일 바운딩 박스의 투영 면적을 최대화하는 카메라의 뷰 방향을 사용하게 되는 경우, 그림 3의 (a)와 (b)처럼 캐릭터의 전후 방향에 검이 위치하는 경우가 바로 바운딩 박스의 투영 면적을 크게 하므로, 카메라는 캐릭터의 측면을 바라보게 되며 따라서 횡방향으로 휘두르는 동작의 역동성은 떨어지게 된다. 그러나 우리가 제안하는 방법을 이용하면 그림 3의 (c)와 같이 검이 만들어내는 모션 면적을 최대화시키는 뷰 방향을 계산할 수 있다.

4.2 카메라 경로의 오프라인 계산 기법

4.1절에서 소개한 최적화 기법을 작은 단위의 시간 간격마다 적용하면 주어진 모션에 대한 부드러운 카메라 경로를 계산할 수 있다. 이에 대한 가장 간단한 아이디어는 매 프레임 단위로 카메라의 뷰 방향을 결정하는 것이다. 그러나 이와 같이 계산된 카메라 방향을 바로 이용할 경우 카메라의 경로가 떨리는 문제가 발생할 수 있으므로, 우리는 각 프레임에서의 모션 면적 값을 가중치로 사용하여 뷰 방향을 부드럽게 블랜딩하는 방법을 제안한다. 자세한 알고리즘은 그림 4에 소개되어 있다.

$\mathcal{T} \leftarrow$ pre-defined threshold of motion area value
$k \leftarrow$ user-defined smoothness coefficient
$\Delta t \leftarrow$ time interval between two adjacent frames
for each frame t of a given motion sequence,
  $A_t \leftarrow 0$
  $l \leftarrow 0$
  while $A_t < \mathcal{T}$ do
  $t_s \leftarrow t - 2^l \Delta t$
  $t_e \leftarrow t + 2^l \Delta t$
  Find $\mathbf{v}_t^{init}$ that maximizes $O(\mathbf{v}_t^{init}) = \frac{\mathbf{v}^T \mathbf{A}(t_s, t_e)\mathbf{v}}{t_e - t_s}$
  $A_t \leftarrow O(\mathbf{v}_t^{init})$
  $l \leftarrow l + 1$
  end while
end for
for each frame t of a given motion sequence,
  $\mathbf{v}_t =$ Blended direction of all viewing directions
    from $t - k$ to $t + k$ with weight A_i
end for

그림 4: 가중치 합을 이용한 카메라 경로의 오프라인 계산 알고리즘

제안된 알고리즘을 살펴보면, 먼저 매 프레임의 카메라 뷰 방향이 4.1절에서 소개한 최적화 기법을 이용하여 초기화되고 있다. 그러나 캐릭터가 주어진 시간 간격 내에서 거의 움직임이 없을 경우 모션 면적의 값이 거의 0에 가깝기 때문에 구해진 카메라 뷰 방향이 쓸모가 없을 가능성이 커지므로, 미리 정의된 임계값 $\mathcal{T}$를 사용하여 초기에 계산된 뷰 방향이 의미있는 값이 되도록 보장하였다. 즉 계산된 모션 면적의 값이 $\mathcal{T}$보다 작을 경우, 범위 단계값 l을 전보다 증가시킴으로써 더 넓은 시간 간격에서의 모션을 고려하게 된다. 모든 프레임에서의 초기 카메라 뷰 방향을 결정하고 나면, 각 프레임에서의 모션 면적 값을 가중치로 사용하여 초기 카메라 뷰 방향의 평균 방향을 모든 프레임에 대하여 계산하게 된다. 이와 같이 모션 면적 값을 가중치로 사용하게 되면 캐릭터의 모션 면적을 더 크게 할 수 있는 방향을 선택하는 데 유리하다. 사용자에 의해 미리 정의된 계수 k는 계산하고자 하는 프레임을 기준으로 전후로 몇 프레임까지 평균을 낼 것인지를 결정하는 요소로써, 이 계수를 조절함으로써 카메라 뷰 방향의 부드럽게 변하는 정도를 쉽게 조절할 수 있다.

4.3 실시간 카메라 경로 계산 기법

4.2에서 소개한 짧은 시간 간격에 대한 최적화 문제는 고유값 분해법을 통해 매우 빠르게 계산될수 있기 때문에, 이를 이용하면 주어진 캐릭터 모션에 대한 카메라 경로를 실시간에 계산할 수 있다. 그림 5에 이에 대한 의사코드를 기술하였다. 간단히 알고리즘을 살펴보면, 본 방법은 모든 프레임에 대해 새로운 카메라 뷰 방향을 결정하기 보다는 두 프레임 사이의 시간 Δt보다 큰 δt 시간마다 최적화 기법을 통해 새로운 카메라 뷰 방향을 결정한다. 중간의 프레임에 사용될 카메라 뷰 방향은 구선형 보간법(SLERP)[20]을 이용하여 부드럽게 보간되었다.

$\Delta t \leftarrow$ time interval between two adjacent frames
$\delta t \leftarrow$ Fixed update time interval
$t \leftarrow$ Current time
$t_{next} \leftarrow$ Next time for optimizing camera direction
$\omega \leftarrow$ Current blending weight
$\mathbf{q} \leftarrow$ Current camera rotation
$\mathbf{q}' \leftarrow$ Next camera rotation
if $t = t_{next}$,
  $t_{next} \leftarrow t_{next} + \delta t$
  Find $\mathbf{v}$ that maximizes $O(\mathbf{v}) = \mathbf{v}^T \mathbf{A}(t, t_{next})\mathbf{v}$
  $A \leftarrow \mathbf{v}^T \mathbf{A}(t, t_{next})\mathbf{v}$
  $\mathbf{q}' \leftarrow$ Convert $\mathbf{v}$ into a quaternion.
  $\omega \leftarrow \Omega(A)$
else,
  $\mathbf{q} \leftarrow$ SLERP between $\mathbf{q}$ and $\mathbf{q}'$ with a weight ω
  $t \leftarrow t + \Delta t$
end if

그림 5: 실시간 카메라 경로 생성 알고리즘

방금 설명한 알고리즘에서 사용된 $\Omega(A)$는 모션 면적값에 따른 카메라 파라미터의 블랜딩 가중치를 조절하는 함수로서 다음과 같이 정의된다.

$$\Omega(A) = \begin{cases} 0, & A < A_{low} \\ \frac{1}{2}\left(1 - \cos\left(\frac{A - A_{low}}{A_{high} - A_{low}}\pi\right)\right), & A_{low} \le A < A_{high} \\ 1, & A \ge A_{high}. \end{cases}$$

이는 4.2에서 소개한 바와 같이 움직임이 거의 없어 매우 작은 모션 면적값이 계산되는 경우에 최적화를 통해 계산된 뷰 방향이 의미없는 값이 될 수 있기 때문에, 이러한 문제를 해결해 주기 위해 사용되었다.

앞에서 소개한 의사 코드는 카메라의 회전값만을 계산하도록 설계되어 있다. 실제 카메라의 파라미터의 구성 요소인 카메라의 위치, 카메라의 목표 위치, 위방향 벡터를 계산하기 위해서, 우리는 각 프레임 시간에서의 루트 조인트의 위치를 카메라의 목표 위치로 설정하고, 카메라와 목표간의 거리값을 고정하였다. 또한 카메라의 위방향 벡터도 전체 좌표계의 위방향 벡터가 되도록 고정하였다. 앞에서 계산된 카메라 뷰 방향과 고정된 카메라-목표간 거리값을 이용하면 카메라의 위치또한 계산될 수 있다. 루트 조인트의 위치에 대하여 가우시안 필터링 처리를 한 위치값을 카메라 목표 위치로 사용하면 예기치 못했던 카메라의 흔들림을 막을 수 있다.

4.4 카메라 제어를 위한 추가적인 제약 조건

제안된 알고리즘들로 계산된 뷰 방향들은 때때로 너무 높은 각도나 낮은 각도에서 촬영한 장면을 만들어낼 수 있으며, 이는 캐릭터가 바닥에 누워 있는 형태 등의 특별한 경우를 제외하고는 모션을 관측하는데 좋지 못하다. 다음과 같이 최적화 문제를 약간 변형함으로써 이러한 현상을 막을 수 있다.

$$\begin{aligned} \text{maximize} \quad & O(\mathbf{v}) = \mathbf{v}^T \mathbf{A}(t, t_{next})\mathbf{v} - \omega |v_2|^2 \\ \text{subject to} \quad & \mathbf{v}^T \mathbf{v} = 1. \end{aligned}$$

여기서 ω는 사용자가 정의한 가중치 계수이고, v_2는 $\mathbf{v}$의 y축값이다. 따라서 위 식과 같이 v_2값을 최소화 함으로써, 지표면과 좀더 평행한 뷰 방향을 계산할 수 있다. 위 식은 다음과 같이 더 간소화될 수 있다.

$$\begin{aligned} O(\mathbf{v}) &= \mathbf{v}^T \mathbf{A}(t, t_{next})\mathbf{v} - \omega \mathbf{v}^T \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \mathbf{v} \\ &= \mathbf{v}^T \left(\mathbf{A}(t, t_{next}) - \omega \mathbf{H} \right) \mathbf{v}. \end{aligned} \quad (7)$$

식 (7) 또한 제곱식의 형태이기 때문에, 고유값 분해법을 통해 쉽게 최적화시킬 수 있다.

그림 6과 같이 카메라의 위치가 지표면보다 더 아래에 위치하는 경우를 막기 위해서, 우리는 다음과 같은 간단한 형태의 보정 기법을 적용하였다. 보정에 앞서 유지해야 하는 두 가지 기준은 그림 6의 **A**와 **B**에 잘 나타나 있다. **A**는 제안한 방법을 통해 계산된 뷰 방향을 유지시키도록 보정한 위치인 반면, **B**는 미리 정해진 카메라-목표간 거리값을 유지시키도록 보정한 위치이다. 우리는 사용자에 의해 미리 정의된 블랜딩 파라미터 $f \in [0, 1]$를 통해서 두 위치를 보간하여 사용하였다.

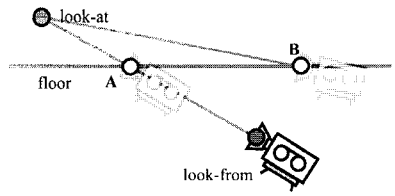

그림 6: 카메라 위치의 바닥 제약 조건을 이용한 보정.

5 결과 및 비교

우리는 제안한 방법의 효용성을 알아보기 위해 몇 개의 모션 캡쳐 데이터에 대해 실험하였다[1]. 실험에 사용된 컴퓨터는 2.13Ghz의 Intel Core2 CPU과 2Gb의 메모리, 그리고 GeForce 7600 GT 그래픽 카드가 장착되어 있었다.

그림 7은 고정된 카메라 파라미터 결정 기법을 사용한 장면과 일반적으로 사용되는 정면 방향, 측면 방향, 조감도 방향 등을 이용한 장면을 비교한 것이다. 여기서 시간 흐름은 그림 2와 같은 방법을 통해 표현되었다. 그림 7의 (c)에 뷰 2번과 5번이나 (e)의 뷰 3번과 같이 일반적으로 사용되는 카메라 방향이 주어진 모션을 제대로 살펴볼 수 없는 장면을 만들어내는 반면, 제안된 알고리즘을 통해 생성된 카메라는 항상 효과적으로 모션 전체를 관찰할 수 있는 장면을 제공한다.

[1]논문의 모든 결과 영상은 visualcomputing.yonsei.ac.kr/personal/jy/camera.avi 에서 확인할 수 있음.

그림 8 오프라인 카메라 경로 계산 기법과 실시간 카메라 경로 계산 기법을 통해 생성된 카메라를 이용한 장면의 스냅샷이다. 우리는 실험에서 임계값 T를 0.02로 설정하였다. 그림 8의 (b)와 (c)의 카포에라 동작이 포착된 것과 그림 8의 (e)와 (f)의 손을 흔드는 동작이 포착된 것을 미루어 보았을 때, 우리가 제안한 방법들이 주어진 동작의 역동적인 특성을 잘 보이도록 하고 있음을 관측할 수 있다. 오프라인 카메라 경로 계산 기법과 실시간 카메라 경로 계산 기법을 이용한 결과들이 거의 흡사한 점은 같은 최적화 수식을 바탕으로 구성되었기 때문으로 해석할 수 있다. 그러나 실제 비디오를 통해 관찰할 경우 오프라인 카메라 경로 계산 기법을 통해 계산된 카메라 경로는 매 프레임마다 뷰 방향을 결정하므로 실시간 기법에 비해 보다 빠르게 주어진 모션에 대한 반응을 보여줌을 확인할 수 있었다. 표 1는 그림 8에 나타낸 각 장면에서의 프레임당 평균 모션 면적값을 계산한 것이다. 제안된 방법들을 통해 찾아낸 뷰 방향들을 이용한 장면에서의 모션 면적값이 보다 높게 나오는 것을 확인할 수 있다.

Fig.8(a)	Fig.8(b)	Fig.8(c)	Fig.8(d)	Fig.8(e)	Fig.8(f)
1.4597	2.5551	2.0944	0.3994	0.5387	0.4657

표 1: 그림 8의 각 장면에서의 프레임 당 평균 모션 면적 값.

	ours	view1	view2	view3	view4	view5
Fig 7(a)	**7**	4	3	5	2	6
Fig 7(b)	**8**	4	3	4	5	3
Fig 7(c)	5	**7**	3	5	6	1
Fig 7(d)	7	3	**7**	1	3	6
Fig 7(e)	**6**	**6**	**6**	3	3	3

표 2: 그림 7의 각 뷰에 대한 설문조사 결과. 굵은 글씨는 가장 높은 점수임을 표시한 것이다.

제안된 방법들을 통해 계산된 카메라 파라미터가 사람들에게 효과적으로 모션의 역동적인 특성을 전달하도록 도움을 주는지 확인하기 위해, 우리는 다음과 같은 설문조사를 수행하였다. 먼저 5개의 모션에 대해 각각의 비디오 묶음을 만들었다. 각각의 비디오 묶음은 6개의 다른 뷰 방향을 사용한 카메라를 이용해 렌더링된 장면으로서, 이중 하나는 본 논문에서 제안한 고정된 카메라 파라미터 계산 기법을 사용한 것이고 나머지는 애니메이터가 직접 선택한 뷰 방향 혹은 일반적으로 많이 사용되는 정면 방향, 측면 방향 등을 사용한 것이다. 우리는 본 논문과 관련된 사전 지식이 없는 성인 남녀 9명을 대상으로 준비된 비디오를 무작위 순서로 보여준 뒤, 각 모션을 가장 잘 역동적으로 보여준 비디오를 3개까지 뽑으라고 지시하였다. 표 2는 투표 결과를 정리한 것으로, 우리의 방법을 이용하여 계산된 뷰 방향이 대체적으로 가장 많은 득표를 하였음을 확인이 가능하다. 따라서 우리는 제안된 알고리즘이 주어진 장면의 모션을 더욱 역동적으로 만들 수 있는 뷰 방향을 선택하는데 적합한 것으로 추측할 수 있다.

제안된 방법은 또한 카메라의 목표물이 장애물에 의해 차폐되는 경우를 피하는 기법들과 쉽게 결합되어 사용될 수 있다. 실험에서 우리는 간단한 형태의 충돌 검출 알고리즘을 이용하여 카메라의 위치를 장애물의 차폐가 없는 영역으로 보정하였다. 실시간 카메라 경로 생성 기법의 경우, 캐릭터가 장애물에 의해 가려지게 되면 카메라의 위치를 장애물의 앞에 오도록 보정하였다. 오프라인 기법의 경우, 먼저 카메라의 경로를 차폐의 고려없이 계산한 후, 차폐가 생기는 경우가 없도록 이 경로를 보정하였다. 급격한 카메라 위치의 변화는 자연스럽지 못한 카메라 경로를 만들게 되므로, 우리는 모션 변위 매핑[21]과 유사한 방식으로 차폐가 발생한 프레임의 이웃 프레임과 함께 부드럽게 보정하였다. 이러한 보정이 더이상 차폐가 생기지 않을 때까지 계속 반복된다. 그림

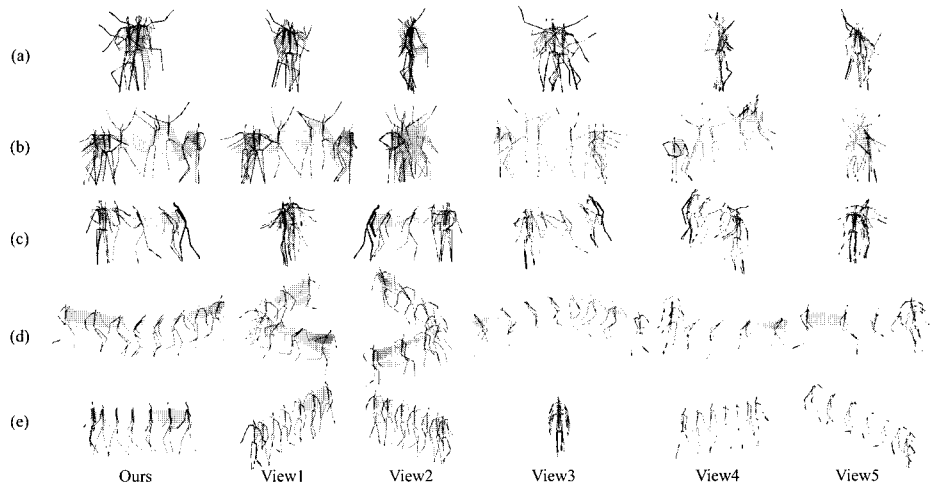

그림 7: 제안된 방법을 통해 계산된 뷰 방향과 타 뷰 방향을 이용한 장면간의 비교. 첫번째 열의 뷰 방향은 제안된 방법을 통해 계산된 것이며, 나머지 장면들의 뷰 방향들은 애니메이터가 직접 선택한 뷰 방향 혹은 일반적으로 사용되는 정면 방향, 측면 방향 등을 이용한 것이다.

9의 (a)는 제안한 오프라인 및 실시간 카메라 경로 생성 알고리즘을 통해 만들어진 카메라 경로가 효과적으로 차폐 회피 보정이 된 모습을 모여주고 있으며, (c)와 (d)에 이에 대한 몇 장의 스냅샷이 보여지고 있다. 그림을 통해 우리는 제안한 방법들이 캐릭터의 동작을 역동적으로 보이게끔 함과 동시에 효과적으로 카메라 차폐를 피하는 것을 확인할 수 있다. 오프라인 카메라 경로에 대한 반복적인 보정 기법 부분을 제외하면, 본 논문에서 제안된 모든 방법들은 실시간에 계산될 수 있을 정도로 충분히 빠르다. 일례로 그림 8에서 사용된 춤 동작은 총 6651개의 프레임으로 이루어져 있으나, 오프라인 카메라 경로 생성 기법을 통해 카메라 경로를 계산하는 데 걸리는 시간은 대략 1초 정도 소모되었다.

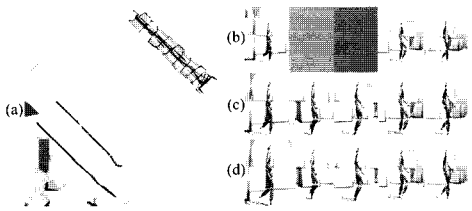

그림 9: 차폐 회피의 예. (a) 예제로 사용된 장면과 카메라 경로를 위에서 본 모습. 적색 곡선은 오프라인 기법을 사용해 생성된 카메라 경로이고, 녹색 곡선은 변위 매핑을 통해 보정된 카메라 경로이며, 청색 곡선은 실시간 기법에 차폐 회피 기법을 더하여 상성된 카메라 경로이다. (b), (c), (d)는 각각의 카메라 경로로부터 생성된 장면의 스냅샷이다.

6 원근 투영을 이용한 모션 면적

앞에서 언급한 모든 방법들은 기본적으로 직교 투영일 경우의 모션 면적을 이용하고 있다. 직교 투영을 이용한 모션 면적 계산은 일반적으로 그래픽스 응용 프로그램에서 많이 사용되는 원근 투영을 이용한 경우를 근사화한 경우로 생각할 수 있으며, 실험 결과 빠른 속도로 문제를 해결하면서도 적절한 해를 제공할 수 있었다. 그러나 이론상 보다 정확한 모션 면적을 계산하기 위해서는 원근 투영을 고려하지 않을 수 없을 것이다. 예를 들어 직교 투영의 경우, $\mathbf{v}$ 뷰 방향과 $-\mathbf{v}$ 뷰 방향은 결과적으로 같은 투영 결과를 생성하므로, 같은 값의 모션 면적을 계산해내게 된다. 그러나 원근 투영을 할 경우 임의의 캐릭터 링크가 만들어낸 궤적이 화면과 가까워질수록 해당되는 모션 면적이 더 커지므로, 보다 정확한 해를 구할 수 있을 것이다. 따라서 본 절에서는 원근 투영을 이용한 모션 면적을 계산하는 방법을 소개하고자 한다.

가장 간단한 형태의 원근 투영은 모든 점들을 $z = 1$ 평면 위에 투영하는 형태의 행렬로 기술될 수 있다. 이러한 형태의 원근 투영을 임의의 점 $\mathbf{p} = [p_x, p_y, p_z]^T$에 적용하게 되면, 투영된 점 $\mathbf{p}^P = \mathbf{p}/p_z = [p_x/p_z, p_y/p_z, 1]^T$ 형태가 된다. 만일 세 점 $\mathbf{a} = [a_x, a_y, a_z]^T, \mathbf{b} = [b_x, b_y, b_z]^T, \mathbf{c} = [c_x, c_y, c_z]^T$가 임의의 삼각형을 이루고 있다면, 3차원 공간 상에서 이 삼각형의 넓이 A는 다음과 같이 계산될 수 있다.

$$A = 0.5\,|\mathbf{a} \times \mathbf{b} + \mathbf{b} \times \mathbf{c} + \mathbf{c} \times \mathbf{a}|\,.$$

이 세 점이 $z = 1$ 평면으로 원근 투영을 했을 경우의 면적은 다음과 같이 계산될 것이다.

$$\begin{aligned} A^P &= 0.5\left|\frac{\mathbf{a}}{a_z} \times \frac{\mathbf{b}}{b_z} + \frac{\mathbf{b}}{b_z} \times \frac{\mathbf{c}}{c_z} + \frac{\mathbf{c}}{c_z} \times \frac{\mathbf{a}}{a_z}\right| \\ &= \frac{|c_z\mathbf{a} \times \mathbf{b} + a_z\mathbf{b} \times \mathbf{c} + b_z\mathbf{c} \times \mathbf{a}|}{2a_zb_zc_z} \\ &= \frac{|c_z(\mathbf{a} \times \mathbf{b})_z + a_z(\mathbf{b} \times \mathbf{c})_z + b_z(\mathbf{c} \times \mathbf{a})_z|}{2a_zb_zc_z}. \end{aligned}$$

여기서 $(\mathbf{a} \times \mathbf{b})_z$는 $\mathbf{a} \times \mathbf{b}$의 z축값을 의미한다. 세 점 $\mathbf{a}, \mathbf{b}, \mathbf{c}$가 모두 $z = 1$ 평면에 투영되었으므로, 삼각형의 넓이를 나타내는 외적의 벡터가 오직 z축값만 갖게 되기 때문에 위의 최종 식과 같이 간소화 할 수 있다.

그림 8: 카메라 경로를 생성한 예제 두 가지. (a), (b), (c)는 카메라 모션에 대해 각각 사용자가 제어한 카메라, 오프라인 카메라 경로 생성 기법, 실시간 카메라 경로 생성 기법을 사용하여 만들어진 장면이다. (d), (e), (f) 또한 같은 식으로 춤 동작에 대하여 만들어진 장면이다.

이제 모션 면적을 계산하기 위해 몇 가지를 가정한다. 먼저 카메라의 뷰 방향은 항상 캐릭터의 루트 관절을 향하도록 하고, 카메라-목표간 거리는 d로 고정되어 있다고 하자. 이러한 가정을 좀더 편하게 하기 위해서, 카메라의 위치는 항상 절대 좌표계의 원점에 위치하도록 하고, 캐릭터의 루트 관절의 위치가 항상 $\mathbf{d} = [0, 0, d_z]^T$의 위치에 오도록 하고, 캐릭터의 회전 변환으로 카메라의 회전을 대신하도록 한다. 또한 캐릭터의 회전 변환의 자유도는 2만 사용함으로써, 카메라의 위방향 벡터를 항상 y축 방향으로 고정시킨다. 이러한 가정을 바탕으로, 세 점 $\mathbf{a}, \mathbf{b}, \mathbf{c}$가 모션 면적을 이루게 되는 한 삼각형으로써 이미 루트 관절의 위치를 0으로 보낸 벡터라고 했을 때, 회전 변환 및 카메라-목표 간 거리 유지를 위한 변환을 적용한 세 점 $\mathbf{a}', \mathbf{b}', \mathbf{c}'$는 다음과 같다.

$$\begin{aligned}
\mathbf{a}' &= \mathbf{R}(u,v)\mathbf{a} + \mathbf{d}, \\
\mathbf{b}' &= \mathbf{R}(u,v)\mathbf{b} + \mathbf{d}, \\
\mathbf{c}' &= \mathbf{R}(u,v)\mathbf{c} + \mathbf{d}, \\
\mathbf{R}(u,v) &= \begin{bmatrix} \cos u & 0 & \sin u \\ \sin v \sin u & \cos v & -\sin v \cos u \\ -\cos v \sin u & \sin v & \cos v \cos u \end{bmatrix}.
\end{aligned}$$

변환된 두 점 $\mathbf{a}', \mathbf{b}'$의 외적을 정리하면,

$$\begin{aligned}
\mathbf{a}' \times \mathbf{b}' &= (\mathbf{R}(u,v)\mathbf{a} + \mathbf{d}) \times (\mathbf{R}(u,v)\mathbf{b} + \mathbf{d}) \\
&= (\mathbf{R}(u,v)\mathbf{a}) \times (\mathbf{R}(u,v)\mathbf{b}) + (\mathbf{R}(u,v)\mathbf{a}) \times \mathbf{d} \\
&\quad + \mathbf{d} \times (\mathbf{R}(u,v)\mathbf{b}) + \mathbf{d} \times \mathbf{d}
\end{aligned}$$

여기서 $\mathbf{d}$가 오직 z축값만 가지고 있으므로, $(\mathbf{a}' \times \mathbf{b}')_z$는 다음과 같이 정리된다.

$$\begin{aligned}
(\mathbf{a}' \times \mathbf{b}')_z &= ((\mathbf{R}(u,v)\mathbf{a}) \times (\mathbf{R}(u,v)\mathbf{b}))_z \\
&= \mathbf{r}_3(u,v)^T(\mathbf{a} \times \mathbf{b}) \\
\mathbf{r}_3(u,v) &= [-\cos v \sin u, \sin v, \cos v \cos u]^T
\end{aligned}$$

따라서 변환된 세 점 $\mathbf{a}', \mathbf{b}', \mathbf{c}'$가 이루는 삼각형이 $z = 1$ 평면에 투영되었을 때의 면적 A'^P은 다음과 같다.

$$\begin{aligned}
A'^P &= \frac{|c'_z(\mathbf{a}' \times \mathbf{b}')_z + a'_z(\mathbf{b}' \times \mathbf{c}')_z + b'_z(\mathbf{c}' \times \mathbf{a}')_z|}{2a'_z b'_z c'_z} \\
&= \frac{|\mathbf{r}_3(u,v)^T \mathbf{A} \mathbf{r}_3(u,v) + \mathbf{B}\mathbf{r}_3(u,v)|}{2(\mathbf{r}_3(u,v)^T\mathbf{a} + d)(\mathbf{r}_3(u,v)^T\mathbf{b} + d)(\mathbf{r}_3(u,v)^T\mathbf{c} + d)}, \\
\mathbf{A} &= \mathbf{c} \cdot (\mathbf{a} \times \mathbf{b})^T + \mathbf{a} \cdot (\mathbf{b} \times \mathbf{c})^T + \mathbf{b} \cdot (\mathbf{c} \times \mathbf{a})^T, \\
\mathbf{B} &= d \cdot (\mathbf{c} \times \mathbf{a} + \mathbf{b} \times \mathbf{c} + \mathbf{c} \times \mathbf{a}).
\end{aligned}$$

위의 식을 살펴보면, 면적 A'^P는 카메라 회전값 u, v에 대한 함수로 볼 수 있다. 절대값 기호를 식에서 제거하기 위해서는 A'^P의 제곱식을 이용하면 된다. 그러나 식의 분모에도 u, v에 대한 식이 포함되어 있으므로, 직교 투영의 경우에서처럼 순수한 형태의 제곱식을 유도할 수는 없다. 따라서 앞에서 소개한 방법대로 모션 면적을 최대화하는 최적화 문제를 통해 최적의 카메라 뷰 방향을 선택할 경우, 직교 투영에 비해 풀기 어려운 비선형 최적화 기법을 이용하여 해결해야 한다. 다만, 위의 식에서 $\mathbf{A}$과 $\mathbf{B}$는 모두 미리 연산이 가능한 값이기 때문에 정리된 식을 이용하면 성능의 향상을 꾀할 수 있다.

앞에서 알아본 바와 같이 원근 투영을 이용한 모션 면적을 계산하기 위해서는 비교적 복잡한 식을 이용해야 한다. 보다 간단한 방법을 통해 원근 투영을 이용한 모션 면적을 계산하는 한 가지 가능한 방법은 Halper 등[1]이 제안한 방법과 유사하게 캐릭터의 링크들의 궤적을 화면상에 렌더링하고 픽셀의 개수를 세는 이미지 정밀도의 알고리즘을 이용하는 것이다. 이러한 방법을 이용하는 경우 모션 면적을 보다 간단한 형태의 구현을 통해 계산할 수

있으나, 한번의 모션 면적을 계산할 때마다 한번의 렌더링 시간을 소모하기 때문에 높은 비용을 소모할 가능성이 크다.

7 결론

본 논문에서는 캐릭터의 움직임이 주어진 카메라 파라미터에 대하여 얼마나 역동적으로 보이는지를 측정하는 방법을 소개하고, 이를 기반으로 캐릭터 모션을 입력으로 하는 몇 가지의 자동 카메라 제어 기법을 제안하였다. 각각의 방법들은 약간씩 다른 카메라 경로를 만들어내므로, 사용자는 응용 프로그램의 필요에 따라 적절한 방법을 선택적으로 이용할 수 있을 것으로 생각된다. 또한 차폐 회피를 위해 제안된 알고리즘은 효과적으로 카메라가 목표로 하는 캐릭터를 차폐물의 방해없이 촬영할 수 있도록 해주는 것을 보장하였다.

모션 면적은 화면 상의 캐릭터 모션을 보는 품질을 측정하는 방법으로서 몇 가지 장점이 있다. 첫번째로, 본 방법은 캐릭터의 관절 구조에 상관없이 적용할 수 있다. 또한 본 방법은 캐릭터가 여러 명일 경우에도 동일한 방법으로 적용이 가능하다. 다만 여러 명의 캐릭터가 등장하는 장면에 대해 제안하는 기법을 효과적으로 사용하기 위해서는 캐릭터가 서로서로를 차폐하게 되는 경우를 막는 방법에 대하여 추가적인 고려가 필요하다. 여러 명의 캐릭터에 대한 모션 면적을 계산하는 것은 계산 비용을 높일 것이나, 우리는 GPU기반의 구현을 통해 제안한 방법의 수행 시간이 효과적으로 개선될 수 있을 것으로 기대한다.

본 논문에서 제안한 방법은 몇 가지의 한계점이 있다. 첫번째로, 4절에서 이미 언급했듯이 제안한 방법들은 캐릭터의 움직임만을 고려하여 카메라 파라미터를 계산하고 있으므로, 주어진 장면이 가지고 있는 의미 혹은 영화 촬영 기법들을 고려하지 않고 있다. 우리는 제안한 방법들과 장면의 의미를 고려하는 다른 방법을 혼용함으로써 캐릭터의 움직임이 있는 장면에 대하여 보다 효과적인 카메라 제어가 가능할 것으로 기대하고 있다.

두번째로, 제안한 방법들은 모두 뷰 방향만을 결정하는데 초점을 맞추고 있으며, 카메라-목표간 거리나 위방향 벡터 등의 다른 파라미터는 모두 미리 정의된 파라미터를 사용하고 있다. 우리는 뷰 방향이 주어진 장면을 다르게 보이게 하는 가장 중요한 요소라고 생각하고 있으나, 보다 좋은 카메라 경로를 위해서는 다른 파라미터들 또한 같이 제어될 수 있는 편이 좋으리라 본다.

논문에서 사용된 차폐 회피 알고리즘은 복잡한 형태의 장애물이 배경으로 주어졌을 때 종종 부자연스러운 결과를 만들 수 있다. 차폐 현상을 막기 위해 반복적으로 카메라-목표간 거리가 수정될 경우 캐릭터가 움직이는 내내 카메라가 매우 짧은 카메라-목표간 거리값을 가지게 되는 경우가 있다. 이러한 문제는 보다 세밀한 형태의 차폐 회피 알고리즘을 적용하여 해결할 수 있을 것으로 기대된다.

8 감사의 글

본 연구는 문화체육관광부 및 정보통신연구진흥원의 IT원천기술 개발사업(2008-F-031-01)의 연구결과로 수행되었음.

참고문헌

[1] N. Halper, R. Helbing, and T. Strothotte, "A camera engine for computer games: Managing the trade-off between constraint satisfaction and frame coherence," in *Proc. Eurographics 2001*, vol. 20, no. 3. Blackwell Publishing, 2001, pp. 174–183.

[2] M. Christie, R. Machap, J.-M. Normand, P. Olivier, and J. Pickering, "Virtual camera planning: A survey," in *SMARTGRAPH '05: Proceedings of the 5th international symposium on Smart graphics*, 2005, pp. 40–52.

[3] J. Blinn, "Where am i? what am i looking at?" *IEEE Comput. Graph. Appl.*, vol. 8, no. 4, pp. 76–81, 1988.

[4] S. M. Drucker and D. Zeltzer, "Camdroid: a system for implementing intelligent camera control," in *SI3D '95: Proceedings of the 1995 symposium on Interactive 3D graphics*. ACM Press, 1995, pp. 139–144.

[5] W. Bares, S. Thainimit, and S. McDermott, "A model for constraint-based camera planning," in *Smart Graphics. Papers from the 2000 AAAI Spring Symposium*, 2000, pp. 84–91.

[6] N. Halper and P. Olivier, "Camplan: A camera planning agent," in *AAAI Workshop on Smart Graphics.*, 2000.

[7] M. Gleicher and A. Witkin, "Through-the-lens camera control," *Computer Graphics*, vol. 26, no. 2, pp. 331–340, 1992.

[8] M.-H. Kyung, M.-S. Kim, and S. J. Hong, "A new approach to through-the-lens camera control," *CVGIP: Graphical Model and Image Processing*, vol. 58, no. 3, pp. 262–285, 1996.

[9] M. Christie and J.-M. Normand, "A semantic space partitioning approach to virtual camera composition," *Computer Graphics Forum*, vol. 24, pp. 247–256, 2005.

[10] L. wei He, M. F. Cohen, and D. H. Salesin, "The virtual cinematographer: a paradigm for automatic real-time camera control and directing," in *Proceedings of ACM SIGGRAPH '96*. ACM Press, 1996, pp. 217–224.

[11] D. B. Christianson, S. E. Anderson, L. wei He, D. Salesin, D. S. Weld, and M. F. Cohen, "Declarative camera control for automatic cinematography," in *AAAI/IAAI, Vol. 1*, 1996, pp. 148–155.

[12] B. Tomlinson, B. Blumberg, and D. Nain, "Expressive autonomous cinematography for interactive virtual environments," in *AGENTS '00: Proceedings of the fourth international conference on Autonomous agents*. ACM Press, 2000, pp. 317–324.

[13] K. Kennedy and R. E. Mercer, "Planning animation cinematography and shot structure to communicate theme and mood," in *Proceedings of the 2nd international symposium on Smart graphics*. ACM Press, 2002, pp. 1–8.

[14] T.-C. Lin, Z.-C. Shih, and Y.-T. Tsai, "Cinematic camera control in 3d computer games," in *SHORT Commication papers proceedings of WSCG '04*. UNION Agency-Science Press, 2004, pp. 289–296.

[15] B. Gooch, E. Reinhard, C. Moulding, and P. Shirley, "Artistic composition for image creation," in *Eurographics Workshop on Rendering*, 2001, pp. 83–88.

[16] D. Sokolov and D. Plemenos, "Viewpoint quality and scene understanding," in *The 6th International Eurographics Symposium on Virtual Reality, Archaeology and Cultural Heritage (VAST '05)*, 2005, pp. 67–73.

[17] D. Sokolov, D. Plemenos, and K. Tamine, "Viewpoint quality and global scene exploration strategies," in *International Conference in Computer Graphics and Applications (GRAPP '06)*, 2006, pp. 184–191.

[18] C. H. Lee, A. Varshney, and D. W. Jacobs, "Mesh saliency," *ACM Trans. Graph.*, vol. 24, no. 3, pp. 659–666, 2005.

[19] D. C. Lay, *Linear Algebra and its Applications (3rd edition)*. Addison-Wesley, 2002.

[20] K. Shoemake, "Animating rotation with quaternion curves," in *Proceedings of ACM SIGGRAPH '85*, 1985, pp. 245–254.

[21] A. Bruderlin and L. Williams, "Motion signal processing," in *Proceedings of ACM SIGGRAPH '95*. ACM Press, 1995, pp. 97–104.

[22] C. B. Phillips, N. I. Badler, and J. Granieri, "Automatic viewing control for 3d direct manipulation," in *SI3D '92: Proceedings of the 1992 symposium on Interactive 3D graphics*. New York, NY, USA: ACM Press, 1992, pp. 71–74.

[23] S. M. Drucker, "Intelligent camera control for graphical environments," 1994.

[24] P. Spellucci, "An sqp method for general nonlinear programs using only equality constrained subproblems," *Mathematical Programming*, vol. 82, pp. 413–448, 1998.