# 반구형 스크린 상의 몰입형 AR 탁구 게임

이상경[O], 경동욱, 정기철
숭실대학교, 정보과학대학, 미디어학과, HCI Lab.
{Sangkyung Lee[O], Dongwuk Kyoung, Keechul Jung}@ssu.ac.kr

# Immersive AR Ping-pong Game in Hemispherical Screen

Sangkyung Lee[O], Dongwuk Kyoung, Keechul Jung
HCI Lab., School of Media, College of Information Science, Soongsil Univ.

요 약

체감형 게임의 재미를 극대화 하기 위해서 몰입감있는 게임화면이 제공되어야 한다. 곡면, 반구형 스크린은 일반 평면 스크린을 사용할 때 보다 더 높은 몰입감을 줄 수 있지만 투사시 생기는 왜곡영상 보정에 많은 계산이 요구되는 문제가 있다. 정지된 영상(프리젠테이션 화면)이나 비디오 영상은 메모리 버퍼링을 통해 시스템이 요구하는 속도를 보장할 수 있으나, 사용자와 상호작용이 요구되는 게임에서는 사용자의 반응을 처리한 후 매 프레임 마다 빠른 워핑(warping) 처리를 요구하게 되는데, 렌더링될 화면을 미리 보정할 수가 없기 때문에 시스템 처리속도 저하의 원인이 된다. 본 논문에서는 체감형 게임을 곡면에 투사하는 방법과 GPU를 이용한 빠른 영상 보정을 제안함으로써 게임에서 요구하는 처리 속도를 보장하고 고해상도의 게임화면을 사용할 수 있도록 했다. 웹카메라를 이용하여 스크린과 프로젝터될 영상간의 호모그래피(homographpy)를 정의해서 워핑했고 워핑 연산을 HLSL로 작성하여 GPU를 이용했다. 실험결과로 반구형 스크린에서 몰입감 있는 AR 탁구 게임을 보인다.

## 1. 서 론

곡면 스크린은 평면 스크린에 비해 여러 장점을 가진다. 장점은 평면 스크린에서 생길 수 있는 스팟(spot) 현상을 곡면 스크린을 이용해서 자연스러운 밝기를 가진 화면을 만들 수 있고[1], 몰입감을 주는 디스플레이 환경구축을 가능하게 한다. 이런 장점 때문에 주로 3D 가상세계 시뮬레이터에 사용되거나(Waller Gunnery Trainer)[2], 영화에 적용되었다(cinerama)[3]. 최근에는 가상현실 체험을 위해 더 많은 프로젝터를 이용해서 고해상도 반구형 타일드 디스플레이(tiled display) 연구가 진행되고 있다[4][5].

본 논문에서는 반구형 스크린이 가지는 장점인 몰입감과 실감난 영상을 컴퓨터 게임에 적용하는 방법을 제안한다. 곡면에 투사된 영상의 왜곡을 보정하기 위해 투사되기 전 이미지 워핑(warping)과정이 필요하며, 곡면의 곡률정보를 얻는 방법으로 스테레오 비젼을 이용하거나[4], 카메라 한대로 입력을 받고 호모그래피(homography) 변환을 이용해 보정하는 방식[6]등이 있다. 제안한 시스템에서 사용한 방법은 영상을 m x n등분한 다음 각 부분의 네 개 꼭지점 정보를 이용해서 보정하는 방식을 사용한다.

게임소프트웨어 특성상 사용자의 입력에 대한 빠른 반응을 요구하며, 매 프레임 마다 워핑하는 과정은 초당 프레임 속도를 떨어트려 시스템의 반응 속도를 느리게 하는 요인이 된다. 그래서 최근 게임에서 실시간 쉐이더 처리를 위해 이용되는 GPU를 이용해서 워핑과정을 수행함으로써 게임 소프트웨어에서 요구하는 속도를 보장하고자 한다. 본 논문에서는 증강현실을 이용한 체감형 탁구게임[7] 을 적용시켜 실험했다.

## 2. 반구형 스크린과 시스템 구성

본 실험에서 사용한 반구형 스크린은 리어(rear)프로젝션 방식으로 크기는 직경 1m이다.

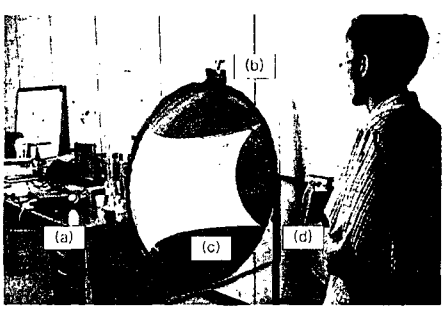



그림 1. 반구형 스크린과 체감형 게임하는 모습

그림 1(a)는 스크린 뒤에 설치된 빔프로젝트, 그림 1(b)는 사용자 동작을 입력 받는 웹카메라, 그림 1(c)반구형 스크린에 투사된 게임화면이며 그림 1(d)는 게임에서 사용되는 라켓이다. 사용자는 이 라켓을 이동시켜 탁구게임을 한다. 탁구채에는 특정한 패턴을 가지는 마커가 부착되어 있고 웹카메라로 사용자가 휘두르는 탁구채의 마커를 인식해서 탁구채의 공간정보를 얻어낸다. 이렇게 얻은 공간 정보를 게임 안 가상 탁구대에 맵핑시켜 탁구공을 친다.

### 3. 왜곡 보정을 위한 워핑 방법

게임화면을 반구형 스크린에 투사하기 위해서는 렌더링된 게임화면을 프리왑(pre-warped)시켜서 곡면에서 발생하는 왜곡을 보정해야 한다. 렌더링된 게임화면은 렌더타겟을 이용해서 하나의 텍스쳐로 처리하게 된다.

시스템에서 곡면 왜곡 보정에 사용한 방법은 하나의 영상을 영상을 n x m으로 나누고 각 영역을 키스톤 보정 한 방식[8]을 이용하여 각 영역에 대한 변환 매트릭스를 구함으로써 전체 영상을 보정한다.

1. 프로젝터-카메라 homography 정의
2. 스크린-카메라 homoraphy 정의
3. 1,2에서 정의된 homography로 프로젝터-스크린 homography 정의
4. 왜곡된 영역안에서 새로 프로젝터될 영역을 정한다.
5. 스크린-이미지 homography 정의
6. 3,5에서 정의된 homography로 프로젝터- 이미지 homography 정의
7. 6에서 정의된 변환 행렬을 이용해서 이미지 워핑수행

그림 2. 키스톤 보정 과정

영상을 평면 스크린에서 키스톤 하기 위한 과정은 그림 2와 같다. 카메라로 스크린에 투사된 이미지의 네 꼭지점과 스크린 네 꼭지점을 찾고 프로젝트된 이미지 좌표와 카메라로 캡쳐된 영상 좌표간의 변환매트릭스 $T$(식 1)와 스크린 좌표와 카메라로 캡쳐된 영상 좌표간의 변환 매트릭스 $C$(식 2)를 구한다. $T$, $C$를 이용해서 프로젝터와 스크린 간의 좌표변환 매트릭스 $P$를 (식3) 구하면 프로젝터 영상이 스크린에 맺히는 좌표를 알 수 있다 (식4). 그림 3은 스크린에 왜곡된 영역 안에 보정될 새로운 사각영역이 지정됨을 보인다.

$$T\times\begin{pmatrix}x_{proj}\\y_{proj}\\1\end{pmatrix}\equiv\begin{pmatrix}x_{cam}\\y_{cam}\\1\end{pmatrix}\quad(1)\qquad C\times\begin{pmatrix}x_{world}\\y_{world}\\1\end{pmatrix}\equiv\begin{pmatrix}x_{cam}\\y_{cam}\\1\end{pmatrix}\qquad(2)$$

$$C^{-1}\times T\times\begin{pmatrix}x_{proj}\\y_{proj}\\1\end{pmatrix}\equiv\begin{pmatrix}x_{world}\\y_{world}\\1\end{pmatrix},\quad P=C^{-1}\times T\qquad(3)$$

$$(p_1^{old},p_2^{old},p_3^{old},p_4^{old})\equiv P\times(p_1^{proj},p_2^{proj},p_3^{proj},p_4^{proj})\qquad(4)$$

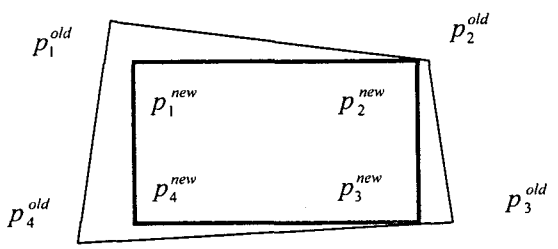


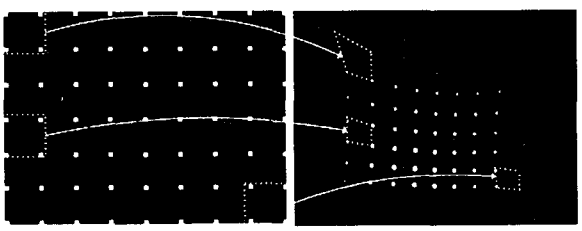

그림 3. 새로운 영역 지정

영상이 보정되어 투사될 새로운 영역과 워핑될 이미지간의 변환 매트릭스 $S$를 구한다 (식5). 최종 워핑 변환행렬 W은 식6 과 같이 구해지며 역행렬 W를 이용해서 백워드(backward mapping) 맵핑 과정을 원영상에 적용함으로써 프리왑된 이미지를 얻는다.

$$P\times\begin{pmatrix}x_{proj}\\y_{proj}\\1\end{pmatrix}\equiv\begin{pmatrix}x_{world}\\y_{world}\\1\end{pmatrix},\quad S\times\begin{pmatrix}x_{image}\\y_{image}\\1\end{pmatrix}\equiv\begin{pmatrix}x_{world}^{new}\\y_{world}^{new}\\1\end{pmatrix}\qquad(5)$$

$$S^{-1}\times P\times\begin{pmatrix}x_{proj}\\y_{proj}\\1\end{pmatrix}\equiv\begin{pmatrix}x_{image}\\y_{image}\\1\end{pmatrix},\quad \mathrm{W}=S^{-1}\times P\qquad(6)$$

곡면에서 비선형 왜곡을 보정하기 위해서는 영상을 영역별로 나누고 각 영역마다 앞서 구한 왜곡보정을 수행한다. 그림4(a)는 곡면 왜곡 정도를 얻기 위해 사전에 뿌려지게 되는 영상이며 그림 4(b)는 곡면에 투사된 모습이다. 그림 4(b) 영상의 점들은 카메라로 입력 되며 이 점들의 위치정보를 이용해서 n x m개 영역의 꼭지점을 구성한다.



(a) (b)

그림 4. 곡면 정보 획득 이미지

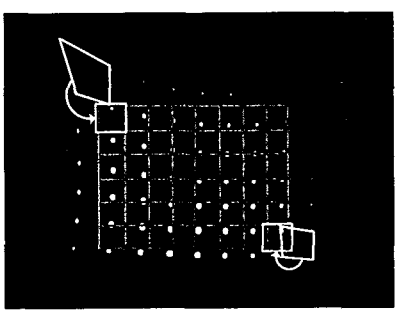


그림 5. 영역별 워핑 수행 모습

그림 5는 반구형 스크린에서 투사된 이미지의 왜곡된 영역안에 보정될 새로운 이미지 영역을 지정한 후 영역별로 워핑이 수행됨을 보인다.

### 4. 실시간성 보장을 위한 GPU 이용

워핑과정 속도는 게임화면의 크기에 따라 달라진다. 게임화면 해상도를 낮게 설정하면 처리 속도를 높일 수 있으나 워핑에서 발생하는 앨리어싱(aliasing) 문제와 사용자에게

선명한 게임화면을 제공 할 수 없다는 문제가 있다. 본 본문에서는 화면 해상도를 크게 유지하면서 게임에서 요구하는 속도를 보장하기 위해서 GPU를 사용했다.[9]

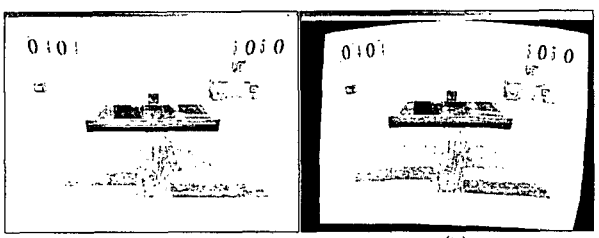
```
float4x4 w; //변환 행렬
texture   tex0; //워핑을 적용하는 게임화면
//기타 설정 생략
float4 PS(
   float4 Diff : COLOR0,
   float2 Tex  : TEXCOORD0) : COLOR
{
   float4 tpos = float4(Tex.x * 1280, Tex.y*1024,1,0);
   tpos=mul(w,tpos);

   Tex.x= (tpos.x / tpos.z);
   Tex.y= (tpos.y / tpos.z);
   Tex.x= Tex.x/1280;
   Tex.y= Tex.y/1024;

    return tex2D(Sampler, Tex);
}
```

그림 6. 워핑을 수행하는 HLSL로 작성된 코드

HLSL를 사용해서 워핑하는 코드를 작성했으며 픽쉘셰이더에서 텍스쳐 좌표를 변환시켜 맵핑한다(그림 6). 원본 텍스쳐의 한 영역을 워핑할 때 변환 행렬W를 상수로 지정하여 픽쉘쉐이더를 거쳐 맵핑된 픽셀값을 반환한다.

## 5. 실험 결과

본 실험에서 CPU는 AMD Athlon XP 2600, 그래픽 카드는 ATI RADEON 9800 pro, 메모리 1GByte를 사용했다.



(a) (b)

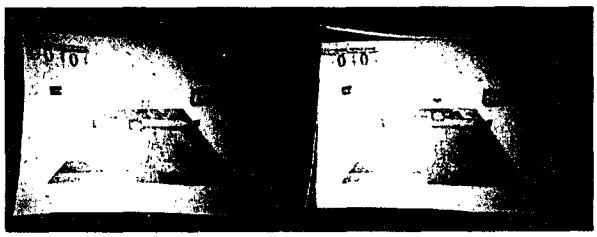
그림 7. 워핑된 게임화면

그림 7(a)는 렌더링된 게임화면이고 그림 7(b)는 게임 소프트웨어에 워핑 모듈을 붙인 후 프리왑된 결과화면이다.

표1. CPU와 GPU 처리 속도 비교.

| | CPU | GPU |
|---|---|---|
| 게임 프레임 수(fps) | 0.97 | 11.64 |

표 1은 본 실험에서 사용한 1280 x 1024 크기의 텍스쳐를 처리한 속도 차이를 나타낸다. 한 장의 게임화면을 변환 매트릭스로 워핑시킬 때 텍스쳐의 좌표를 구하는 곱셈연산을 CPU만 가지고 수행 했을 때 약 621ms정도 걸렸다. 한 장의 텍스쳐(1280 x 1024)를 보정처리 하기 위해서는 1,310,720번의 벡터-매트릭스 곱셈연산이 된다. 게임 프레임 수를 보면 GPU가 월등히 높음을 알 수 있다. 최종 게임화면이 그려지는 단계까지 GPU를 이용할 때 텍스쳐 생성 및 하드웨어간의 데이터 교환으로 지연시간이 포함되어도 결과적으로 본 실험에서 사용한 게임에서 약10배 정도 GPU가 빠른 것을 확인 할 수 있다.



(a) (b)

그림 8. 곡면스크린에 뿌려진 게임화면

그림 8은 반구형 스크린에 투사된 최종 수행 결과를 보인다. 그림 8(a)는 게임 화면이 반구형에 투사되어 왜곡된 화면이고 그림 8(b)는 왜곡을 보정해서 왜곡이 보정된 화면을 나타낸다. 그림 7(a)가 투사되면 그림 8(a)처럼 스크린에 왜곡 되고 프리왑된 그림 7(b)를 투사하면 그림 8(b)과 같은 보정된 영상을 볼 수 있다.

## 6. 결론

곡면에 투사하기 위한 왜곡 보정단계에서 GPU를 사용함으로써 속도향상을 보였다. 앞으로 타일드 디스플레이에서 블렌딩 과정이나 변환 매트릭스간의 보정과정 또한 고속으로 처리할 수 있도록 연구할 예정이다. 또한 GPU를 사용하는 워핑부분은 다른 게임에 적용될 수 있도록 작성되어 있어 다양한 게임에 응용될 것으로 기대된다.

## 참고 문헌


[1]http://www.moesrealm.com/homehteater/screenguid.html

[2]http://www.cineramaadventure.com/trainer.htm

[3]http://www.redballoon.net/~snorwood/book/

[4]Ramesh Raskar, Jeroen van Baar, Thomas Willwacher, Srinivas Rao, "Quadric Transfer for Immersive Curved Display,"Computer Graphics Forum, Vol.23, Issue 3, pp 451-463, Sep.2004

[5]T.Moriya, F.Beniyama, K.Utsugi, T.Minakawa, H.Takead, K. Ando, "Multi-camera and multi-projector based seamless live image display system,"Multimedia Modelling Confference, pp265-272, Jan.2004

[6]Ruigang Yang, David Gotz, Justin Hensley, Herman Towles, Michael S. Brown, "PixelFlex: a reconfigurable multi-projector display system," In Proceedings of the conference on Visualization'01, October 21-26, 2001

[7]이상경, 정기철, "증강현실과 제스쳐를 이용한 탁구 게임"한국게임학회, pp251-255, Jul.2005

[8]Rahul Sukthankar, Rober G. Stockton and Matthew D. Mullin, "Smater Presentations:Exploiting Homography in Camera-Projcetor System,"In Proceedings of International Conference on Computer Vision, Vol. 1, pp.247-253, 2001

[9]Wolfgang F. Engel, "ShaderX2:Introduction and Tutorials with DirectX9.0," WORDWARE Publishing