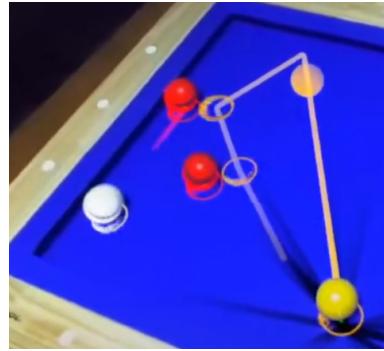


AR 당구

강승우

한국기술교육대학교 전자공학과

2020.10



요약

당구는 재미있는 실내 스포츠이지만, 지금처럼 오락거리가 많은 시대에 초심자가 밭을 블이기에는 다소 진입장벽이 높다. AR 당구는 입문자들이 당구의 높은 진입장벽을 손쉽게 극복하는 데 도움을 줄 수 있도록 기획되었다. 몰입도 높은 VR HMD¹상에서 당구공이 득점 가능한 최적의 경로 및 사용자가 취한 큐의 각도에 대한 피드백을 시각화하고, 여러 시청각 효과를 통한 오락 요소를 도입하여 입문자의 흥미 유발과 당구에 대한 감각 습득을 돋пуска.

AR 당구의 구현은 (1) 영상 처리를 통한 당구대, 당구공, 큐의 3D Pose 획득 (2) 득점 경로를 계산하기 위한 물리 시뮬레이션 구현 (3) 오브젝트의 3D 포즈 및 계산된 득점 경로 등에 대한 시각화의 세 요소를 통해 이루어진다.

(Abstract 내 최종 구현 내용은 추후 기술)

¹Head Mounted Display의 약자; VR 헤드셋과 같이 머리에 장착하여 사용자의 시야 전체를 채우는 형태의 장착형 디스플레이 기기를 일컬음

1 영상 처리

1.1 개요

가상 현실 Virtual Reality 기술은 말 그대로 가상으로 세계를 만들고, 헤드 트래킹 기술을 통해 가상 세계의 카메라 위치를 사용자 눈의 위치와 동기화하여 사실적인 시청각 경험을 부여하는 기술이다.

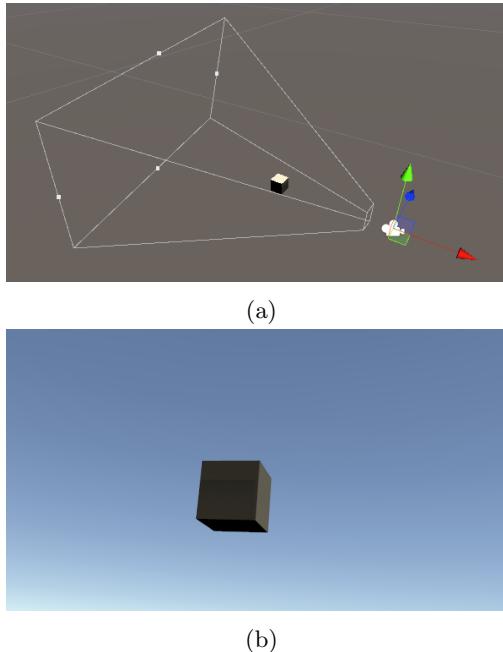


그림 1.1: 가상 세계의 카메라가 사용자의 머리 위치에 따라 동기되어 현실적인 느낌을 준다

가상 현실 어플리케이션은 기존의 전통적인 3D 게임에서 카메라의 제어만을 사용자가 장착한 HMD의 헤드 트래킹 신호에 위임한 것으로, 가상 현실 세계를 만드는 것은 3D 게임을 만드는 것과 다르지 않다.

본 과제에서 구현한 증강 현실 Augmented Reality 어플리케이션 또한 이 개념에서 크게 벗어나지 않는다. 여전히 가상 세계의 좌표와 카메라 위치 동기화 및 오브젝트 배치가 필요하며, 이를 오브젝트를 현실 세계의 영상 위에 덧그린다는 것이 가상 현실과의 유일한 차이점이다.

그러나 증강 현실 어플리케이션이 설득력을 얻기 위

해서는 덧그린 가상 현실 오브젝트의 위치와 크기, 모양 등이 영상의 물체와 보기 좋게 어우러져야 하므로, 영상 처리를 통해 현실 세계의 오브젝트가 가상 세계의 어느 트랜스폼으로 매핑되는지 파악하는 것이 중요하다.

한 번 현실 세계 오브젝트의 가상 세계 좌표를 획득하고 나면, 이후에는 해당 좌표에 가상 오브젝트를 생성하는 것만으로도 증강 현실 영상에 그럴싸한 효과를 그려낼 수 있게 된다. 단, 해당 오브젝트가 지속적으로 움직이고 있는 상태라면 지속적으로 영상 처리를 수행해 해당 오브젝트의 가상 현실 좌표를 업데이트 해주어야 할 것이다.

AR 당구의 영상 처리 또한 이러한 맥락에서 동작한다. 당구대와 당구공의 가상 세계 좌표를 획득하면 게임 엔진은 이 정보를 바탕으로 당구공이나 테이블에 대한 오버레이를 그리거나, 당구공의 최적 타격 경로에 대한 시뮬레이션 등을 수행하고 이 결과를 시각화하는 등 여러 방향으로 활용할 수 있다.

영상 처리 절차에서는 일차적으로 영상 처리를 통해 당구대와 당구공 등 각 물체의 카메라에 대한 3D 포즈를 계산하고, 이를 가상 세계의 좌표로 변환하는 과정을 다룬다.

1.2 당구대 인식

그림 1.2에서 당구대 펠트의 진한 파란색이 매우 뚜렷하다. 색감이 이렇게 강한 물체는 HSV 색공간으로 표현했을 때 hue 영역이 안정적으로 고정되고, saturation이 강해 다른 영역과 쉽게 구별할 수 있다(그림 1.4).

그림 1.4의 hue 이미지의 각 픽셀 값은 그림 1.3의 스펙트럼에 대응한다. 따라서 hue 값이 약 0~25 사이에 분포하는 픽셀을 필터링하면 테이블 영역을 식별할 수 있다. 그러나 saturation이 현저히 낮아 색상을 구별할 수 없는 흰색과 검은색은 hue 값이 단순히 0에 매핑되므로, hue 값만을 사용해 필터링할 경우 잘못된 결과를 얻을 수 있다. 따라서 saturation 값 또한 일정

²휴대성, 작업의 용이성, 경제성 등을 고려하여 작은 크기의 미니 당구 세트를 활용하였다.



그림 1.2: 당구대 세트, VR HMD 및 HMD에 장착된 스테레오 카메라²

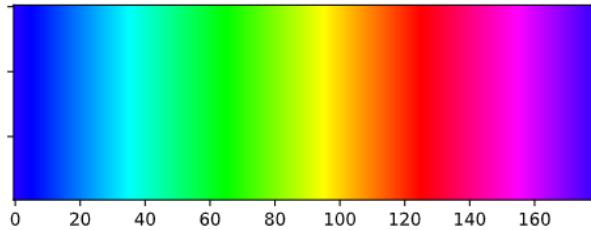


그림 1.3: OpenCV의 HSV 스펙트럼 (8-bit value range)

이상의 값(170~255)을 갖는 픽셀을 선택한다.

Value(밝기)는 당구대 영역을 식별하는 데 큰 도움을 주진 않으므로, 모든 범위를 선택한다.

그림 1.5의 왼쪽 이미지가 위에서 말한 필터링을 적용한 결과이다. ⁴여러 가지 물체가 당구대 위에 올려져 있고, 당구대의 쿠션 밑에 생긴 음영 등으로 인해 당구대의 펠트 전체가 깔끔하게 덮이지는 않았지만, 당구대 영역을 식별하는 것은 가능하다.

그러나 이 시점에서 그림 1.5 (a)의 이진 이미지는 단순한 부울 값 배열로, 이미지만 놓고 봤을 때 프로

³이 때, Hue 채널은 파란색이 0도와 180도 양단에 걸쳐 있어 5° 우측으로 전이shift하였다

⁴OpenCV의 inRange 함수를 사용한다. 함수 inRange는 이미지의 각 채널(여기서는 H, S, V)에 대해 필터링할 값의 범위를 지정하고, 각 픽셀이 해당 범위 내에 속하는지 여부를 검사해 부울bool 값으로 반환한다.

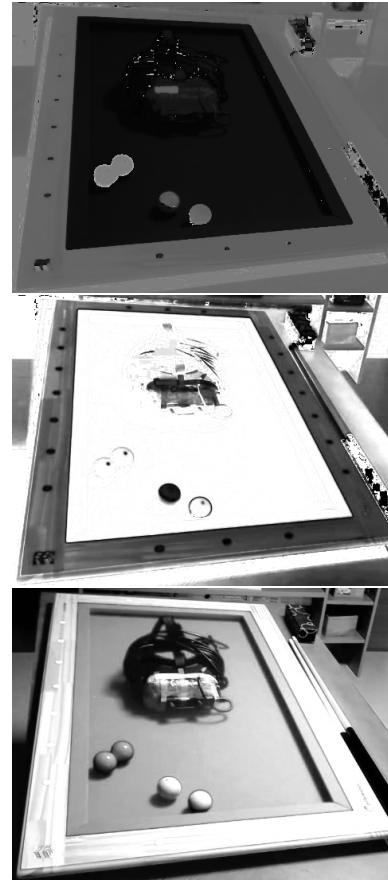


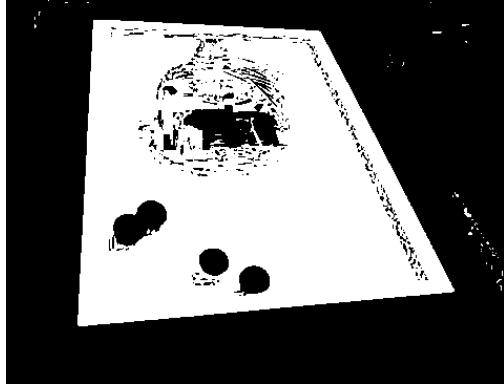
그림 1.4: 그림 1.2의 HSV 색공간 변환 이미지. 순서대로 H³, S, V. 검은색은 0, 흰색은 255 범위

그램은 당구대 영역의 위치는 물론, 존재 여부조차 알 수 없다. 따라서 이미지에서 당구대를 이루는 영역이 어디인지를 찾고 처리하기 쉬운 형태로 표현하기 위해 이미지 상에 존재하는 모든 도형을 찾고, 당구대의 조건에 부합하는 도형을 선택하기로 한다. 여기서는 픽셀 면적이 일정 이상이고, 꼭지점의 개수가 정확히 4개인 도형을 찾는다.

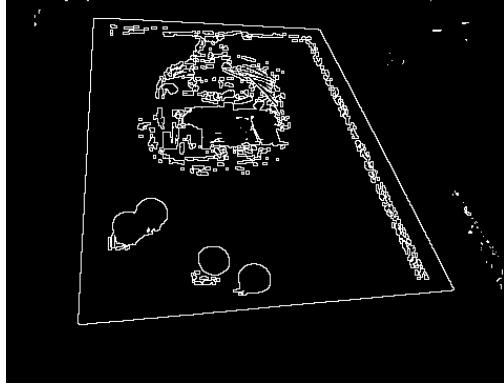
먼저 이미지의 경계선 정보가 필요하다. 이진 이미지에 침식 연산⁵을 적용한 후, 원본 이미지에서 침식된 이미지를 빼는 것으로 각 덩어리chunk의 경계선만을 남길 수 있다. 그림 1.5 (b)가 이에 해당한다.

경계선 이미지에 OpenCV의 findContours 함수를 적

⁵이진 이미지에서, 값이 1인 각각의 픽셀을 인접한 픽셀들과 평가하여, 인접한 픽셀 모두 1이면 1로, 하나라도 0이면 0으로 바꾸는 연산. 노이즈 제거 등에 활용한다



(a) 필터링된 이진 이미지



(b) 검출된 경계선

그림 1.5

용한다.[?] `findContours`는 이진 이미지에 존재하는 모든 닫힌 도형 목록을 찾아 반환하는 함수로, 하나의 도형(`등고선 contour`으로 표현)은 그 도형을 이루는 정점의 좌표 목록으로 구성된다.

이 때 함수가 가장 바깥쪽의 도형만 탐색하게끔 인자를 지정한다. 그림 1.5 (b)의 당구공, HMD, 쿠션의 음영 등으로 인해 생긴 도형들은 모두 가장 바깥쪽의 당구대 영역 내부에 존재하므로 도형 탐색 과정에서 고려하지 않는다. 이로써 당구대 내부에 생기는 어느 정도의 잡음은 무시할 수 있게 된다.

그러나 이렇게 획득한 등고선의 정점 목록을 그대로 평가하기에는 다소 어려움이 있는데, OpenCV의 `findContours` 함수는 두 점 사이의 직선성을 매우 엄격하게 평가하기 때문이다. 즉, 그림 1.5 (b)의 당구대 경계선은 육안으로는 4개의 꼭지점을 잇는 사각형으로 보이지만, 픽셀 단위로 처리될 땐 곧은 직선이 아닌 여러

정점을 갖는 파선으로 평가될 수 있는 것이다.



그림 1.6: 각각의 붉은 점은 정점을, 검은 선은 정점을 있는 간선을 나타낸다

그림 1.6는 이를 잘 나타낸다. 당구대 영역을 구성하는 도형의 한 변은 직선이 아닌 무수한 정점을 지나는 파선으로 평가된다. 우리는 꼭지점의 개수(4개)를 통해 당구대 영역을 찾고자 하므로, 융통성 없게 평가된 등고선의 정점 목록을 직선이 되도록 어느 정도 근사해주어야 한다.

여기에는 OpenCV의 `approxPolyDP`⁶함수를 사용한다. 이 때 `approxPolyDP` 함수에서 근사의 민감도를 나타내는 인자인 ϵ 은 픽셀 단위인데, 많은 경우 당구대의 직선은 10픽셀 이상 벗어나지 않는 깔끔한 형태를 보이므로 $epsilon$ 은 10으로 설정한다. 그 결과 그림 1.7과 같이 테이블 영역의 정점 개수가 4개로 정리되고, 4개의 정점을 이은 도형이 나타내는 영역 또한 테이블 영역에 잘 맞음을 확인할 수 있다.

그러나 위와 같은 방법으로 테이블 영역을 계산하는 경우, 테이블 영역을 침범하는 작은 잡음에도 매우 취약해지는 문제가 발생한다. AR HMD를 착용하고 당구를 즐기는 사용자의 시점에서 큐가 테이블 영역 안으로 들어가거나 입체인 공이 테이블의 경계를 가리는

⁶Ramer-Douglas-Peucker algorithm의 구현체. 시점과 종점을 잇는 간선이 있는 두 정점 사이의 정점 목록 중 가장 거리가 먼 정점이 임계점 ϵ 보다 멀리 있으면 해당 정점을 새로운 종점으로 선택하고, 가까이 있으면 해당 정점을 제거하고 기존의 종점을 시점으로 선택해 새 간선으로 위의 과정을 재귀적으로 반복, 근사를 수행하는 알고리즈다.



그림 1.7: $\epsilon = 10$ 으로 approxPolyDP 함수를 적용한 경우의 시각화

등 테이블의 경계선이 왜곡되는 일은 매우 빈번하게 발생한다(당구대의 일부만이 시야에 들어오는 경우는 뒤에서 다룬다).

그림 1.8은 이렇게 테이블의 경계가 침범당할 때의 처리 과정을 보여주는 이미지로, 그림 1.8 (c)에서 당구대의 침범 당한 경계선을 따라 그림 1.8 (d)의 등고선이 형성되었음을 볼 수 있다. 특히 그림 1.8 (d)의 위쪽, 빨간 공에 의해 침범당한 부분은 공이 내부의 쿠션 음영으로 인한 잡음과 테이블의 외부 경계를 잇는 가교 역할을 해, 더 나쁜 결과를 반환한다.

단순히 정점의 개수가 4개이며, 넓이가 일정 이상이면서 가장 큰 도형을 당구대의 영역으로 판단하는데, 당구대 영역의 도형이 그림 1.8 (d)와 같이 계산되면 이를 당구대 영역으로 인식하지 못하게 된다.

그러나 그림 1.8 (d)에서 검출된 정점 중 가장 바깥쪽의 정점 4개를 이어보면 여전히 내부의 다른 모든 정점을 포함하는 볼록 다각형 convex hull을 획득할 수 있다. 다행히도 OpenCV 라이브러리에는 정점 목록으로부터 볼록 다각형을 계산해내는 함수성 또한 포함되어 있어, 이를 적용한 결과는 그림 1.9과 같다.

이 때 조금이라도 돌출된 부분이 있다면 그림 1.9 (a)와 같이 돌출된 부분을 새로운 정점으로 계산하게 되므로, 정점의 개수를 다시 4개로 근사하기 위해 예의 approxPolyDP 함수를 다시 적용, 그림 1.9 (b)를 얻는다.

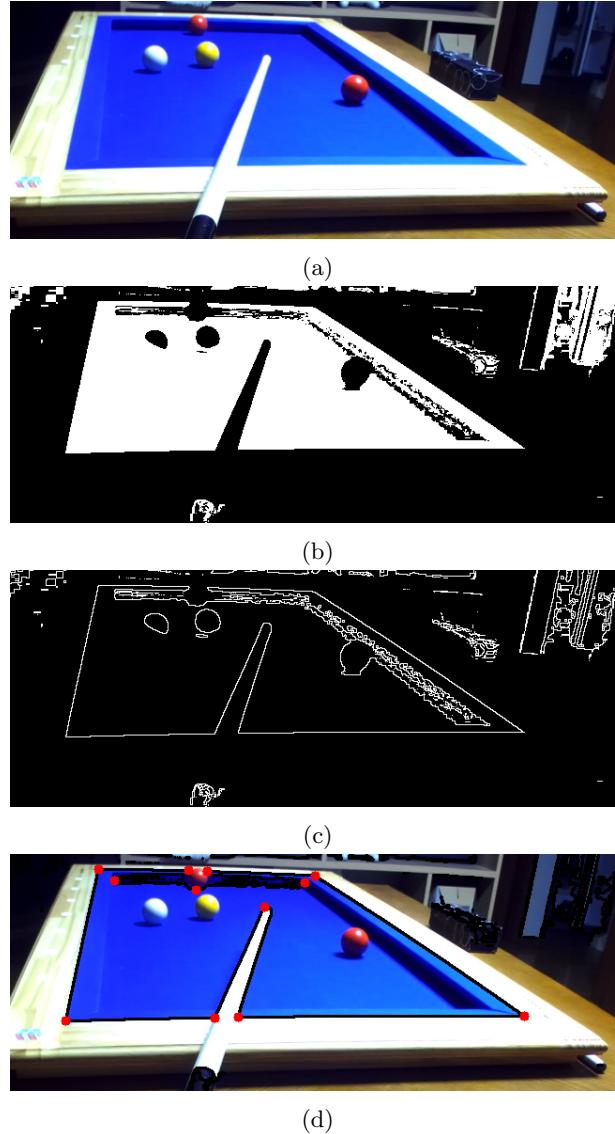


그림 1.8

이렇게 얻어진 당구대 영역의 사각형은(바로 앞에 파랗고 커다란 직사각형 물체를 갖다 놓지 않는 한) 대체로 찾아낸 사각형 중 가장 큰 크기를 갖게 되므로, 사각형을 이루는 네 개의 정점이 당구대의 네 귀퉁이에 해당한다고 안전하게 가정할 수 있다.

당구대의 펠트에 해당하는, 바깥 쿠션 영역의 치수는 $0.96 \times 0.51[m]$ 이다. 만약 카메라의 위치에 있는 같은 크기의 직사각형 판을, 적절하게 회전시키고 이동시키면 화면 상의 테이블 펠트 영역을 정확하게 덮을 수 있을 것이다. 이 때 카메라와 같은 위치(원점)에 존재하



(a)



(b) $\epsilon = 10$ 으로 approxPolyDP 함수 재적용

그림 1.9: convex hull 적용

는 가상의 테이블을 모델이라 하고, 원점에 존재하는 모델로 화면 상의 테이블 영역을 덮기 위해 적용해야 하는 위치와 회전 정도를 테이블의 포즈pose라 한다.

그림 1.10에서와 같이 인덱스가 잘못 지정된 경우를 해소하기 위해, Stereo camera로부터 얻어진 깊이 이미지에서 당구대 후보의 네 점에 대한 깊이 값은 획득하고, 이를 3D 좌표로 변환하여 각 변의 추정 길이를 계산한다.⁷

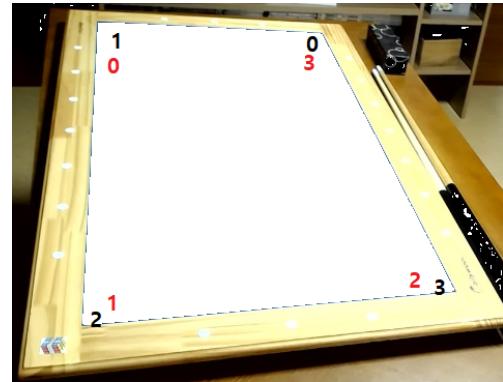


그림 1.10: 인덱스 순서는 위와 같이 어긋날 수 있다.

추정 3D 포인트를 두 개의 인덱스씩 순회하여 짧은 변이 먼저 나타난다면 그대로 두고, 긴 변이 먼저 나타나면 컨투어의 정점 목록을 하나씩 회전시키는 식이다.

정렬된 컨투어의 정점 목록과 모델 공간의 테이블 정점 목록을 인자로 PNP 알고리즘⁸을 적용하면 카메라에 대한 테이블의 relative position과 rotation을 획득할 수 있다. 이를 카메라 월드 트랜스폼으로 변환해 주면 테이블의 월드 위치 및 회전을 획득할 수 있다.

단, 이 경우 테이블의 회전 수치는 180° 회전된 값이 반환될 수 있는데, 이 경우 마지막으로 보고된 테이블의 월드 로테이션과 비교하여 각도의 차이가 175° 이상인 경우 단순히 180°를 제하는 것으로 보정한다.

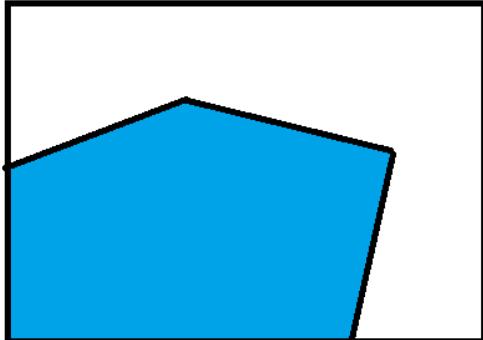
테이블의 일부만 시야에 들어온 경우

테이블의 일부만이 시야 내에 있어 컨투어의 정점 개수가 5개 이상 검출되는 경우에는 PNP 알고리즘을 적용할 정확한 정점의 위치를 파악하기 어렵다. 하지만,

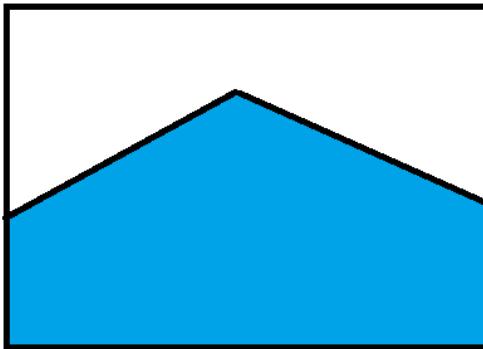
⁷ 깊이 이미지로 얻어진 각 픽셀의 3D 좌표는 값이 거의 10 cm 수준에서 진동하여 신뢰하기 어려우므로, 깊이 이미지는 이 단계에서만 활용하고 이후에는 사용하지 않는다.

⁸ cv::solvePnP 함수, iterative method 적용

적어도 두 개 이상의 코너가 시야 내에 잡혔다면 테이블의 포즈를 추정하는 것이 가능하다.



(a) 위치를 추정할 수 있음



(b) 위치 추정이 불가

그림 1.11: 테이블의 일부분만 시야에 들어오는 경우

임의의 position과 rotation으로 테이블의 모델을 화면에 투영하고, 투영된 컨투어 목록과 검출된 컨투어 목록 사이의 오차를 계산하여 반복적으로 오차를 줄여나가는 방법이다.

그러나 단순히 컨투어 목록을 OpenCV가 제공하는 projection method⁹를 활용하면 일부 정점은 단순히 화면 밖으로 투영되고, 그림 1.12에서와 같이 화면의 경계선에 걸쳐 있는 점들과의 오차를 제대로 계산할 수 없다. 따라서 모델 정점들을 화면에 투사하기 전, 먼저 시야 범위를 구성하는 네 개의 평면에 맞춰 범위를 벗어나는 정점들에 대해 절단을 수행한다.

위 과정을 거쳐 추정 위치 및 회전으로 투사된 정점 목록을 검출된 정점과 비교해 오차를 계산한다. 이 때

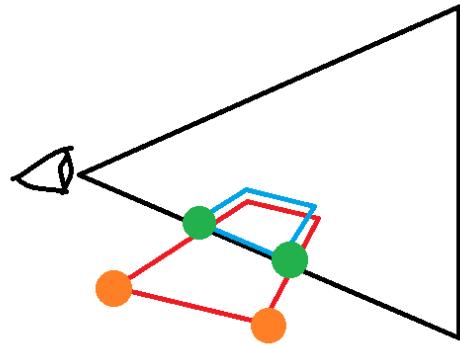


그림 1.12: 모델 정점을 그대로 화면에 투사할 시 일부 정점이 화면에서 이탈한다

투사된 정점 목록의 인덱스를 시계 방향으로 한 번씩 회전시키며, 정점 사이의 거리(오차)의 합 $\sum_{n=0}^N |\vec{P}_{pn} - \vec{P}_{sn}|$ 이 최소치가 되는 트랜스폼을 선택한다.

이후 위와 마찬가지로 카메라에 대한 테이블의 상대 포즈를 월드 좌표로 변환한다.

1.3 당구공 인식

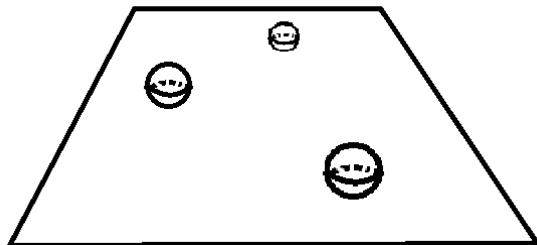


그림 1.13: 당구공의 중점은 항상 한 평면상에 존재한다

당구공은 완전한 구의 형태이며, 따라서 카메라에서 2D 이미지 상 원형의 중점으로 광선(a)을 투사하면 반드시 구의 중점을 지나게 된다. 또, 구의 중점은 반드시 당구대 표면에서 일정한 거리로 떨어진 평면(b) 상에 존재한다.

이 과정은 그림 1.14에 잘 나와있다.

이미 위에서 테이블의 3D 포즈를 계산하였다. 이렇게 계산된 포즈는 테이블의 바깥쪽 쿠션 높이를 기준

⁹cv::projectPoints

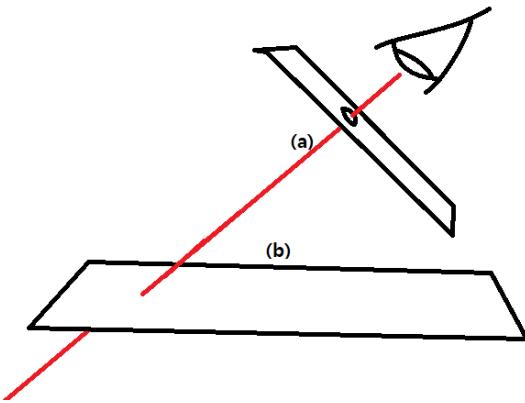


그림 1.14: (b)는 당구공의 중점이 존재할 수 있는 평면이다

으로 한 평면이므로(그림 1.15의 A), 평면 (b)는 테이블 평면을 노멀 반대 방향으로 약간 오프셋하여 손쉽게 계산할 수 있다.

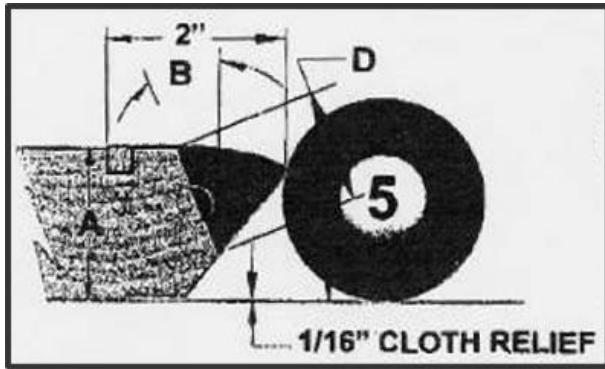


그림 1.15: 당구대 쿠션 높이 예시¹⁰

다음으로 필요한 것은 각 공의 중점을 찾아내는 것이다. 텁-다운으로 이미지를 촬영해 이미지 상에서 공의 크기가 균일하고 공 사이에 페색_{occlusion}이 발생하지 않는 일반적인 당구 보조 프로그램과 달리, AR 당구는 사용자 시점에서 영상 인식을 수행해야 하기 때문에 거리에 따라 구체의 이미지 상 반지름이 다르고, 페색 또한 발생할 수 있다.

카메라로부터의 거리에 따라 변화하는 공 원형의 반

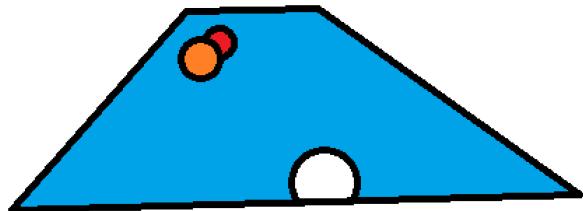


그림 1.16: AR 당구의 당구공 인식은 더 많은 요인을 고려해야 한다

지름에 대응하기 위해 공을 검출하기 위한 커널의 크기를 매 평가시마다 동적으로 조절한다. 이 과정은 알고리즘 1에 서술되어 있듯, 임의의 2D 점에 대한 면 3D 점을 구하고, 카메라 원점 $(0, 0, 0)$ 과 해당 점 사이에 광선을 투사해 평면 (b)과 만나는 지점의 z 값을 찾음으로써 이루어진다.

Algorithm 1: Estimate ball radius scale of given pixel coordinate

$\vec{P}_{ball}(u, v) \leftarrow$ [Input] 2D point of estimated circle center
 $\vec{P}_{ray}(x, y, z) \leftarrow \vec{P}_{ball}$'s 3D coordinate assuming z as 10
 $\vec{P}_{ball3}(x, y, z) \leftarrow$ contact between $(0, 0, 0)$ and \vec{P}_{ray} on the table plane (b)
Result: Evaluation kernel's radius scale is z^{-1}

적합도 평가

일반적으로, AR 당구의 사용자 시점에서 공은 다른 공에 의해 폐색될 수 있다. 즉, 많은 경우 공을 찾는 알고리즘은 오차가 적은 공보다 더 가능성이 높아 보이는 공을 찾게 된다. 이 과정을 직관적으로 수행하기 위해 각 중점 후보를 평가하는 과정에서 오차치가 아닌 적합도를 계산한다.

적합도는 오차 ϵ 에 대한 함수로, 적합도

$$a = \alpha^{-\epsilon^2}$$

로 계산한다(그림 1.17). 이 때 α 는 적합도의 허용치를 결정하는 계수로 여기서는 1.015로 두었다. 이 식에서

¹⁰이미지 출처: <https://www.pooltablefeltcloth.com/cushion-height-guide-for-k-66-k-55.html>

각 경우(픽셀 또는 정점의 비교, 후술)의 적합도 최대
치는 오차가 0인 경우 1이다.

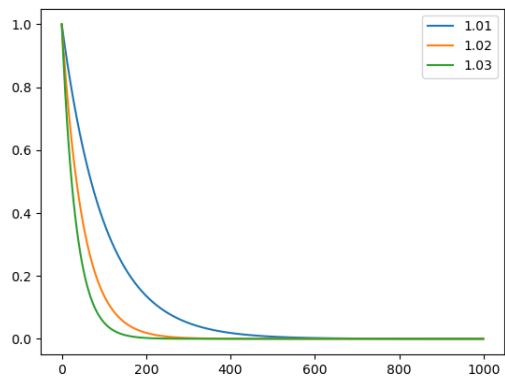


그림 1.17: α 값에 따른 오차에 대한 적합도 그래프

경계 적합도

먼저 이미지의 HSV 색공간 표현에 대해 각 공별로
필터링을 적용,