2021년 서울시립대학교 컴퓨터과학부 셈틀제 최종 보고서

프로젝트명	증강현실 당구 프로젝트		
과제팀 이름	DETAIL_AR		
개발기간	2021. 05. 01. ~ 2021. 08. 23.		
분야	안드로이드 애플리케이션, 컴퓨터 비전		
셈틀제 팀 구성원	이름	이상민	배세윤 -
	학번	2018920039	2018920025 -
	연락처	-	-

1. 프로젝트 요약

\

□ **프로젝트 요약**

○ **개발하고자 하는 과제가 어떤 것인지를 알 수 있도록 반 페이지 내외로 요약**

당구 초보를 위한 4구 당구공 길 안내 애플리케이션입니다. 당구공이 배치가 되어있는 모습을 앱 카메라로 촬영하면 알고리즘을 통해 당구공을 어떻게 쳐야 하는지를 증강현실을 통해 표시합니다.

먼저, 4개의 당구공과 당구대의 모든 모서리를 보여줍니다. 정상적으로 인식이 되면 애플리케이션에서 '인식 버튼을 누르세요' 안내 창이 나옵니다. 인식 버튼을 누르면 현재 자신이 쳐야 하는 당구공 색을 기준으로 득점을 할 수 있는 경로를 보여줍니다. 증강현실 애플리케이션이기 때문에 카메라를 자유롭게 이동하며 경로 확인이 가능합니다. 사용자가 원하는 자리에서 안내된 경로를 통해 당구공을 칠 수 있도록 합니다. 다른 경로를 원한다면 경로를 바꾸는 버튼을 통해 경로 변경이 가능합니다. 또 공의 색상을 바꾸는 버튼으로 노란 공, 하얀 공 모든 경우를 고려한 경로를 확인할 수 있습니다.

사용한 기술 스택으로는 Android Studio with NDK와 OpenCV, C++를 이용했습니다. 안드로이드 스튜디오를 통해 카메라 기능 구현 및 앱 개발을 하고, OpenCV를 통해 당구공, 당구대 인식을 하였습니다. 마지막으로, 인식이 완료되면 길을 찾아주는 알고리즘을 C++을 통해 구현하였습니다.

2. 배경

가. 개발 배경

현실에서 당구 게임을 하면 당구공을 어떻게 쳐야 득점할 수 있는지 알기 어려운 경우가 있습니다. 그러다 보니 당구공의 득점 경로를 알려주는 스마트폰 애플리케이션이 있으면 좋겠다고 생각했습니다. 플레이스토어에서 관련 애플리케이션을 찾아봤으나, 만족할 만한 애플리케이션이 없었습니다. 모바일 내 당구 게임은 있었지만 현실의 당구공이 놓인 위치를 게임상에 똑같이 배치하는 것이 어려웠습니다. 또한 시뮬레이션은 할 수 있었지만 어떻게 쳐야 하는지 구체적으로 알 수 없었습니다. 당구대 위 공들을 스마트폰 카메라에 담아 실시간으로 당구공의 길을 찾아준다면 편리할 것이라 생각했고 함께 개발을 시작하였습니다.

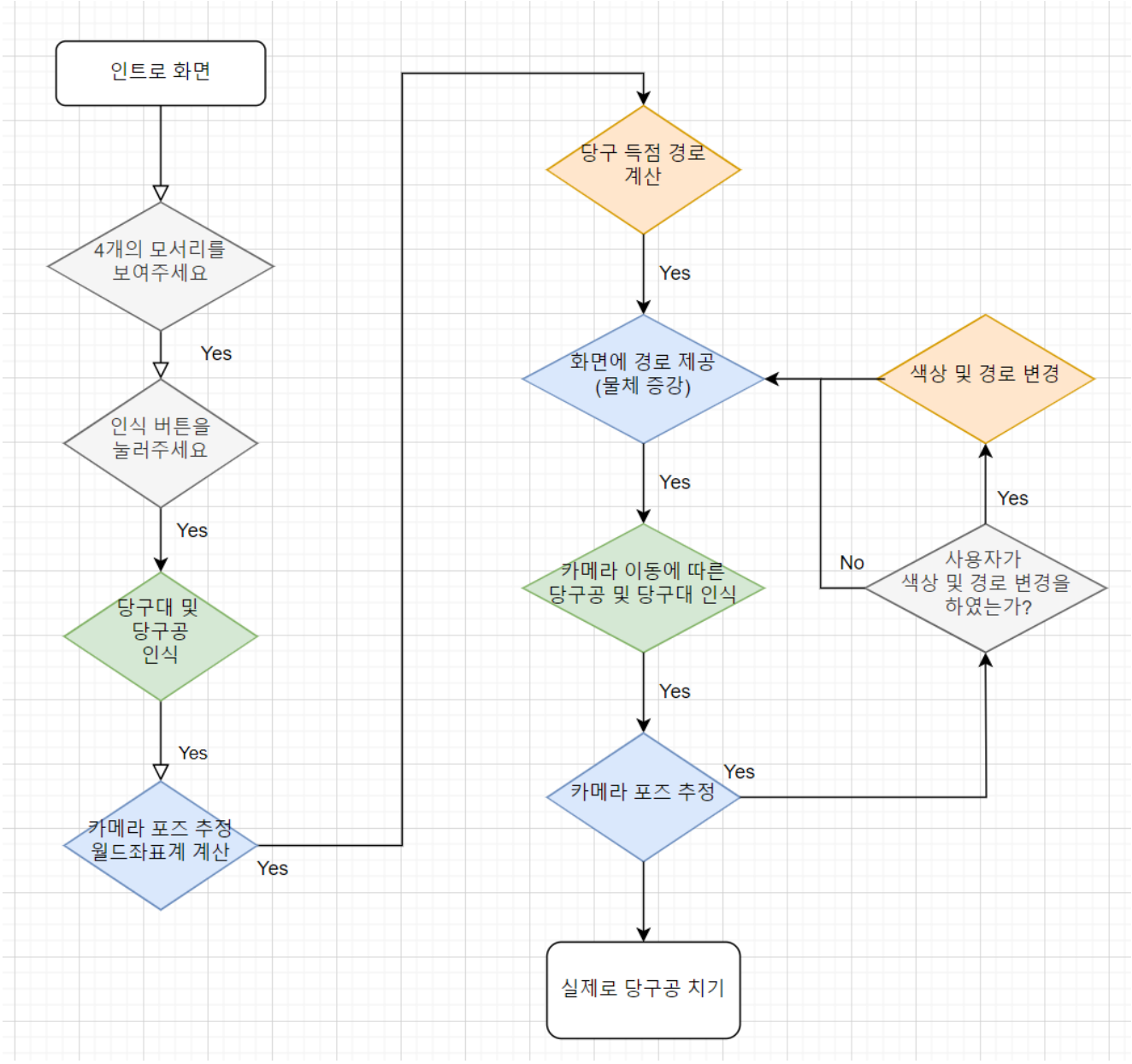
나. 개발 효과

스마트폰 카메라를 이용해 당구 게임을 할 때 당구공의 득점 경로를 실시간으로 확인할 수 있고, 사용자는 경로를 관찰하며 자유롭게 이동할 수 있습니다. 간단히 설치가 가능한 스마트폰 앱과 편리한 앱 사용을 돕는 UI가 만나 남녀노소 누구나 이용할 수 있기 때문에 당구에 대한 진입장벽을 낮춰 줄 것입니다. 또한 당구공의 색상 변경 및 다양한 경로를 표시하는 기능이 구현되어 있습니다. 애플리케이션이 제시한 경로와 자신이 생각한 경로와 비교해 보며 당구에 대한 인사이트를 넓힐 수 있을 것입니다. 해당 애플리케이션을 통해 당구에 대한 흥미를 갖는 사람들이 더 많아질 것이라 기대합니다.

3. 설계

인트로 화면을 거쳐서 핸드폰의 카메라가 켜집니다. 카메라의 해상도는 모든 핸드폰에서 작동할 수 있도록 720 * 480의 해상도로 설정한 상태로 애플리케이션이 작동합니다.
애플리케이션의 작동이 시작이 되면, "4개의 모서리를 보여주세요"라는 토스트 메시지가 나오면서 유저와 상호작용을 시작합니다. 카메라를 통해 당구대의 4개의 모서리를 보여준다면 애플리케이션이 당구대와 당구공의 위치를 인식합니다. 이렇게 당구대와 당구공의 영상에서의 좌표 값을 얻었으면 증강현실을 위한 3차원 기하학적 처리를 거치며 카메라 포즈를 및 당구공의 3D world 좌표를 추정합니다. 계산된 당구공의 3D 좌표 값은 지속적인 증강현실 프레임 업데이트에 쓰이며 득점 경로 계산을 위한 솔루션 클래스에게도 전달됩니다. 솔루션 클래스에서 계산된 경로는 원과 화살표를 통해 화면에 그려지며 카메라의 위치가 움직일 때마다 자연스럽게 물체가 증강됩니다. 사용자는 증강된 득점 경로를 통해 당구 게임을 즐길 수 있으며 언제든 득점 경로와 치고 싶은 당구공의 색을 바꿀 수 있습니다.

알고리즘을 Flow chart로 표현하면 다음과 같습니다. 초록색은 Detection 클래스, 파란색은 3D_Geo_proc 클래스, 주황색은 solution 클래스를 의미하고 회색은 안드로이드에서 발생하는 이벤트를 의미합니다.

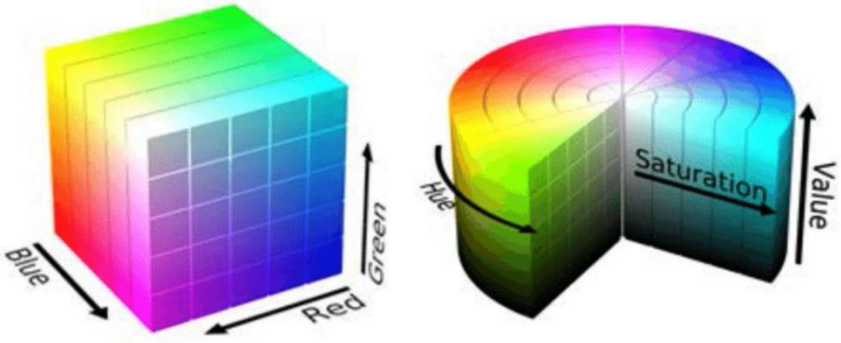


4. 기능

1. Detection

(1) 색상 검출 및 문제점

주어진 영상에서 당구대 및 당구공을 검출하기 위해 RGB 형식으로 된 픽셀을 HSV 형식으로 변환합니다. HSV 공간에서 색을 표현하는 것이 사람이 직관적으로 이해하거나 계산하기 쉽기 때문입니다. 아래 그림은 각각 RGB 색상과 HSV 색상을 나타냅니다.



HSV에서 H가 색상(Hue)을 나타냅니다. 어떤 픽셀의 H 채널 값이 3~35를 나타내면 노란색, 97~123을 나타내면 파란색 등, H 채널 값에 따라 색을 구별하여 인식하는 방식으로 구현하였습니다. 하지만 이런 방식으로 당구대 및 당구공을 검출한다면 주변 환경에 대한 여러 가지 노이즈에 영향을 많이 받게 됩니다. 예를 들어 형광등 빛, 벽지 및 바닥 색, 창문 등 색상 검출만으로 당구대와 당구공을 검출하기 어렵게 만드는 여러 가지 요인이 있습니다. 아래는 노란색을 검출했을 때 True가 되는 픽셀을 흰색으로 표시한 이미지입니다. 노란색 당구공뿐 아니라 벽, 당구봉 등에서 노란색으로 검출되는 것을 확인할 수 있습니다.

이렇게 여러 환경적인 요인을 이겨내는 강건한 검출 알고리즘을 구현하기 위해 당구대 및 당구공 인식에서 다음과 같은 핵심 조건을 이용하였습니다.

당구대 : **허프 라인**을 이용하여 당구 대의 꼭짓점의 좌표를 구합니다.

당구공 : 당구대 내부에 생기는 **구멍**(파란색이 아닌 물체를 의미) 및 구멍의 비율을 이용하여 당구공의 중심 좌표를 구하였습니다.

Detection에서 구한 당구대의 꼭짓점의 좌표와 당구공의 중심좌표는 이후 3D기하학적 처리 부분에서 쓰이게 됩니다.

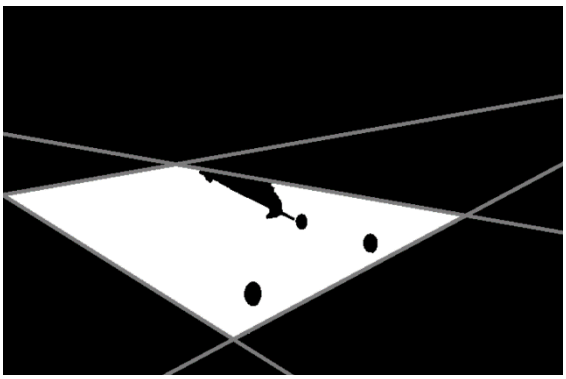
(2) 당구대 인식

당구대로 인식할 조건은 다음과 같습니다.

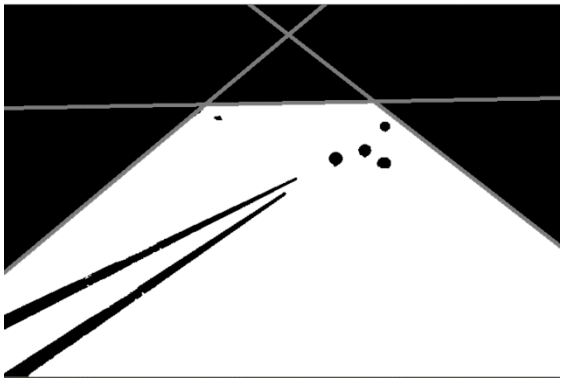
1) 영상에서 보이는 **가장 큰 파란 물체**를 당구대의 후보로 둡니다.

2) 1)에서 검출된 후보에서 일정 개수 이상의 **당구대 꼭짓점의 개수와 당구공의 개수가 검출**되어야 합니다. (이때, 당구대의 코너 개수와 당구공의 개수의 조합이 중요합니다.)

2)의 설명을 덧붙이면, 반듯한 변을 가진 사각형(당구대)과 같은 물체만이 꼭짓점을 가질 수 있습니다. 또한 당구대 위에는 반드시 당구공(흰색, 노란색, 빨간색 두 개)이 있다고 가정합니다. 만약 당구대 위에 당구공이 배치되어 있지 않다면 애플리케이션을 사용해 솔루션을 이유가 없습니다.
실생활에서는 위의 조건을 만족하는 물체를 우연히 발견할 수는 있지만, 당구장에서는 위의 조건을 모두 만족하는 물체는 당구대 이외에는 찾기가 매우 힘듭니다. 따라서 당구대를 검출하기 위한 좋은 제약 조건이 되며 1)과 2)를 만족한 물체에 대하여 허프 라인 검출을 시행합니다.



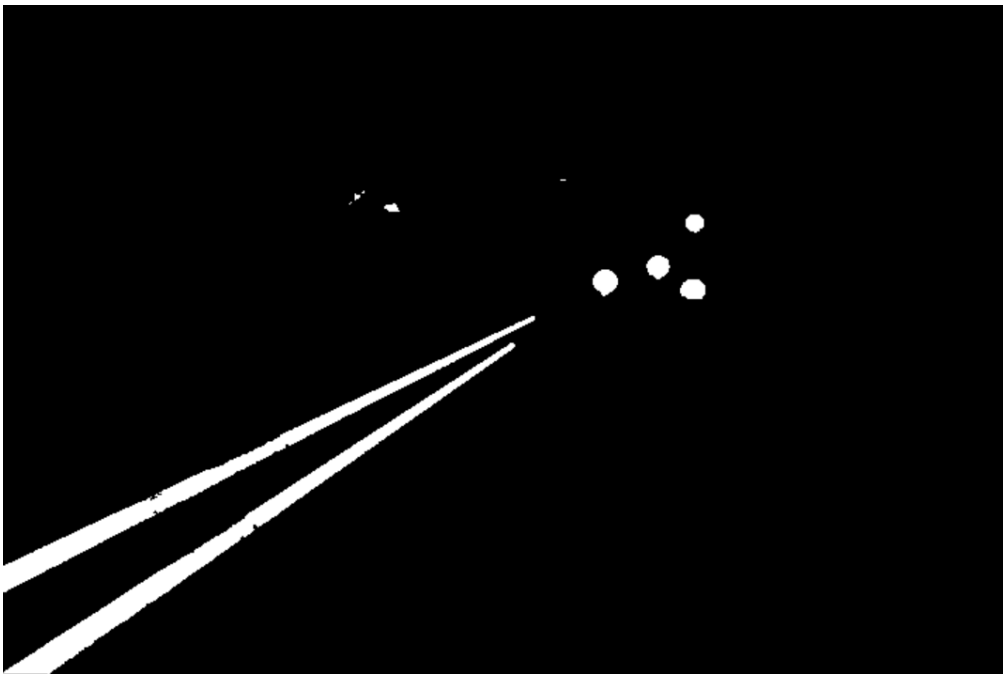
위의 오른쪽 그림에서 1)과 2)의 조건을 만족한 물체를 표시하였으며 허프 라인을 통해 직선을 구한 영상입니다.
허프 라인을 이용해 구한 직선으로 당구공의 코너점을 검출하는 과정에서 어려웠던 점은 다음과 같이 두 가지입니다. 첫 번째로 카메라를 통해서 본 당구대(사각형)의 원근 변환으로 인해 다양한 경우의 수를 고려하여야 했습니다. 예를 들어 4개의 직선으로는 4개, 5개, 6개의 교점을 가지는 상황이 모두 발생할 수 있었습니다. 또한 아래의 왼쪽 그림과 같이 3개의 직선이 3개의 교점을 만들어 그들 중 2개를 골라내야 했습니다. 이를 해결하기 위해 허프 라인 직선이 가지는 각도(세타) 정보와 당구대의 중심에서 각 직선에 내린 수선의 발 사이의 각도를 계산하여 일정 문턱치를 넘은 직선의 교점만이 검출되도록 하여 안정적으로 당구대의 코너를 구할 수 있었습니다.



이어서 두 번째로 어려웠던 점은 위의 오른쪽 그림과 같이 허프 라인을 통해 직선을 구하면 그 후보가 여러 개(노란색 직선)가 생긴다는 점이었습니다. 한 차선에 대해 여러 개의 직선이 검출되는 것을 알 수 있습니다. 당구대도 마찬가지로 한 변에 대해 여러 개의 직선이 검출되는 문제가 있었습니다. 올바른 당구대 코너 검출을 위해 각 변마다 하나의 직선만이 필요합니다. 이를 해결하기 위해 첫 번째와 비슷한 방법으로 각도 차이가 얼마 나지 않는 직선은 같은 변을 나타내는 직선으로 간주하는 등 각 변마다 하나의 직선이 계산될 수 있도록 구현하였습니다.

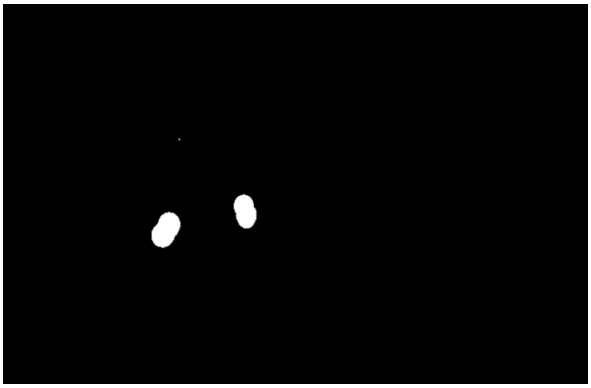
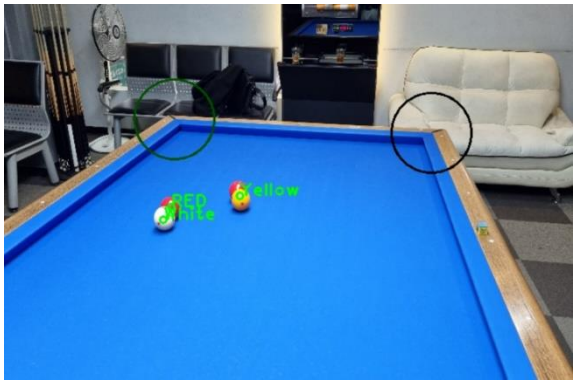
(3) 당구공 인식

당구공은 당구대에 비해 전체 영상에서 차지하는 크기가 상당히 작습니다. 또한 당구대에 비해 빛의 반사 및 그림자의 영향을 많이 받아 온전한 원의 모양을 검출하기가 어려웠습니다. 이를 해결하기 위해 아래 그림과 같이 당구대 검출에서 사용한 영상을 반전시켜 그 구멍이 남도록 하였습니다. 즉 파란색 물체(당구대) 안에서 파란색이 아닌 부분을 나타냅니다. 파란색 이 아닌 모든 물체를 나타내며 이는 흰색, 노란색, 빨간색 당구공을 의미합니다.

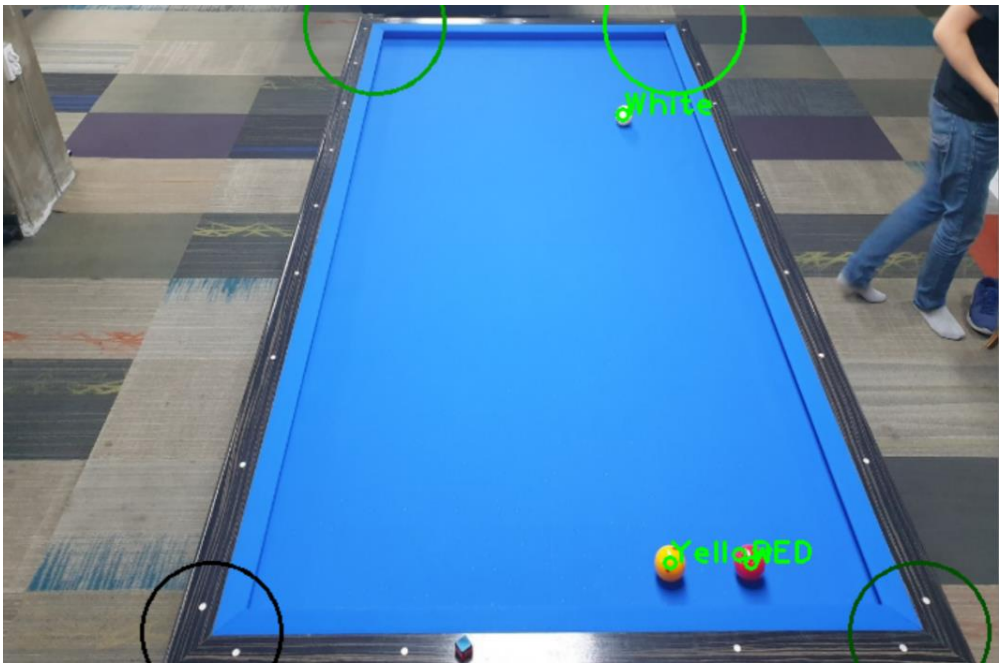


실제 응용에서는 여러 가지 노이즈가 발생하는 경우를 대비했습니다. 동그란 물체를 구하기 위해 어떤 물체를 둘러싼 최소 사각형의 면적이 물체의 면적과 비교해 특정 비율 이상일 때만 공으로 검출되도록 하였습니다. 또한 특정 면적 이하로 너무 작은 물체는 노이즈로 간주하여 공으로 검출되지 않도록 하였습니다.

공 검출에서 어려웠던 점은 공이 겹치는 경우였습니다. 단단한 당구공이 물리적으로 겹칠 수는 없지만 보이는 각도에 따라 언제든지 겹쳐 보일 수 있기 때문에 반드시 해결해야 할 문제였습니다. 이는 K-means 알고리즘을 이용하여 여러 공이 겹쳤을 때 군집의 중심을 찾아서 이를 당구공의 중심 좌표로 두는 방법으로 해결하였습니다. 하지만 빨간 공이 두개 겹친 경우는 겹치는 여부를 알 수 없었기 때문에 해결하지 못하였습니다. 아래의 그림과 같이 흰색 공과 빨간 공이 겹친 경우, 노란 공과 빨간 공이 겹친 경우 각각 공의 중심좌표를 구한 모습입니다.

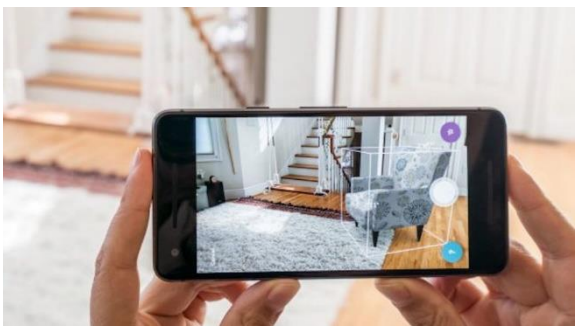
이렇게 해서 최종적으로 아래 그림과 같이 당구대 코너 및 당구공 검출을 완료하였습니다.



2. 3D Geometry processing

(1) 증강현실을 구현하는 방법

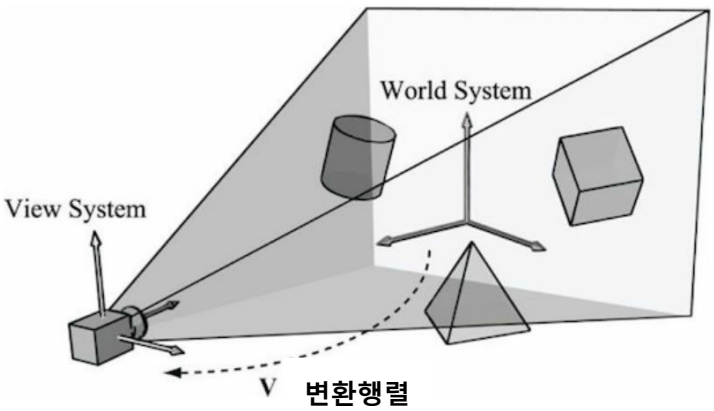
증강현실을 구현하기 위한 방법으로는 대표적으로 두 가지가 있습니다.

왼쪽 그림과 같이 마커 기반 증강현실과 오른쪽과 같이 마커가 없는 증강현실이 있습니다. 저희는 당구대를 일종의 마커로 취급하여 증강현실을 구현한 마커 기반 증강현실이라고 할 수 있습니다.

(2) 3D 기하학적 처리

물체를 증강시키기 위해서는 카메라 좌표와 월드 좌표 간의 **변환 행렬**을 구해야 합니다

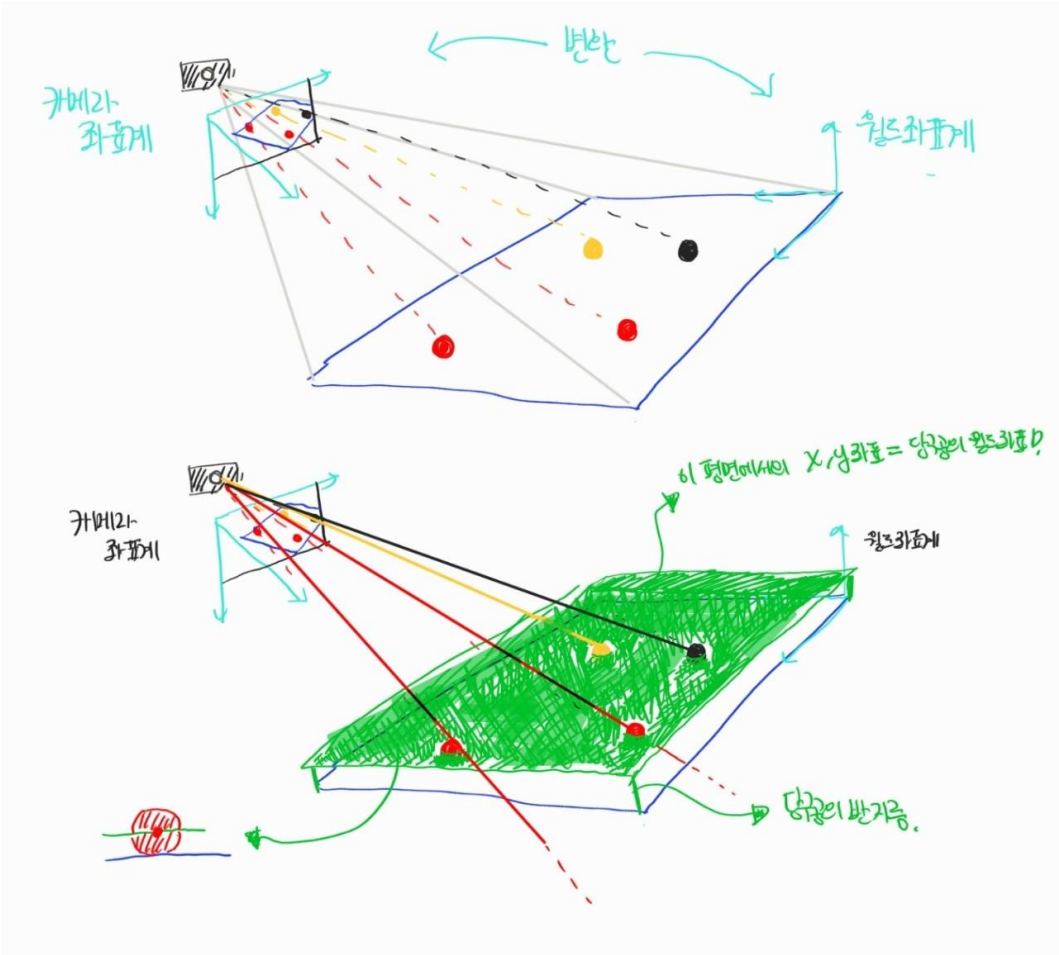


내부 파라미터 행렬 **외부 파라미터 행렬**

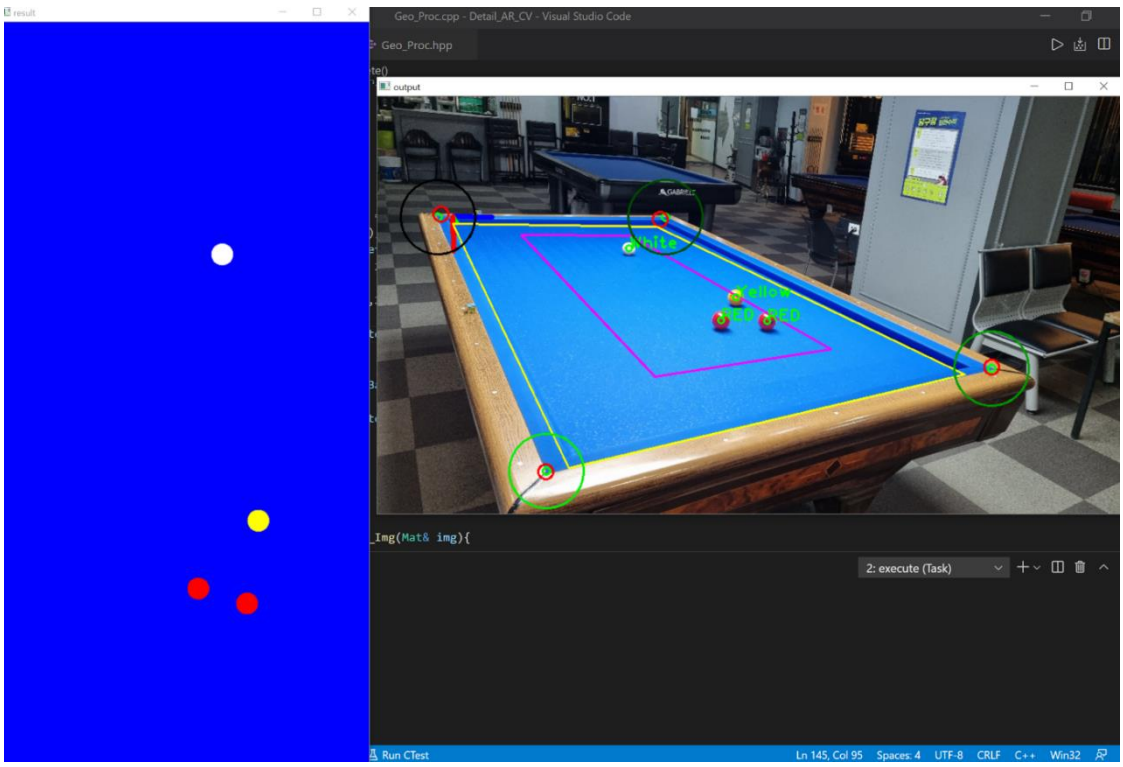
$$s\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

변환 행렬은 카메라 외부 파라미터 행렬, 카메라 내부 파라미터 행렬 두 가지로 쪼개어 타나 낼 수 있습니다. 변환 행렬을 구한다면 우리가 원하는 물체를 입체적으로 영상에 증강되도록 구현할 수 있습니다. 내부 파라미터 행렬은 보통 카메라 켈리브레이션 기법을 이용해서 구하게 됩니다. 하지만 실제로 해본 결과 오토 포커싱 등 여러 가지 문제가 발생하여 카메라 내부 파라미터 행렬을 구하는 것이 어려웠습니다. 따라서 카메라 내부 파라미터 행렬은 눈으로 보았을 때 물체가 자연스럽게 증강되도록 하는 파라미터(초점거리, 주점)를 직접 찾아 대입하여 구하였습니다. 다음으로 카메라 외부 파라미터를 구하기 위해서 월드 좌표의 점 4개와 이미지 좌표의 점 4개를 이용하여 외부 파라미터 행렬의 미지수를 모두 구하였습니다. 이때 사용한 opencv 함수는 solvePnP함수 입니다. 이렇게 변환 행렬을 구하였다면 당구공의 3D 월드 좌표를 구해야 합니다. 즉 당구대의 특정 꼭짓점을 (0,0,0)으로 두고 4개의 당구공이 어디에 위치해 있는지 좌표로 나타내어야 합니다. 이렇게 정확히 구한 당구공의 월드 좌표를 이용해 **당구대 꼭짓점과 당구공의 조합을 이용해 지속적으로 카메라 포즈를 추정하는 변환 행렬을 구할 수 있고, 자연스러운 증강현실 구현이 가능해집니다.**

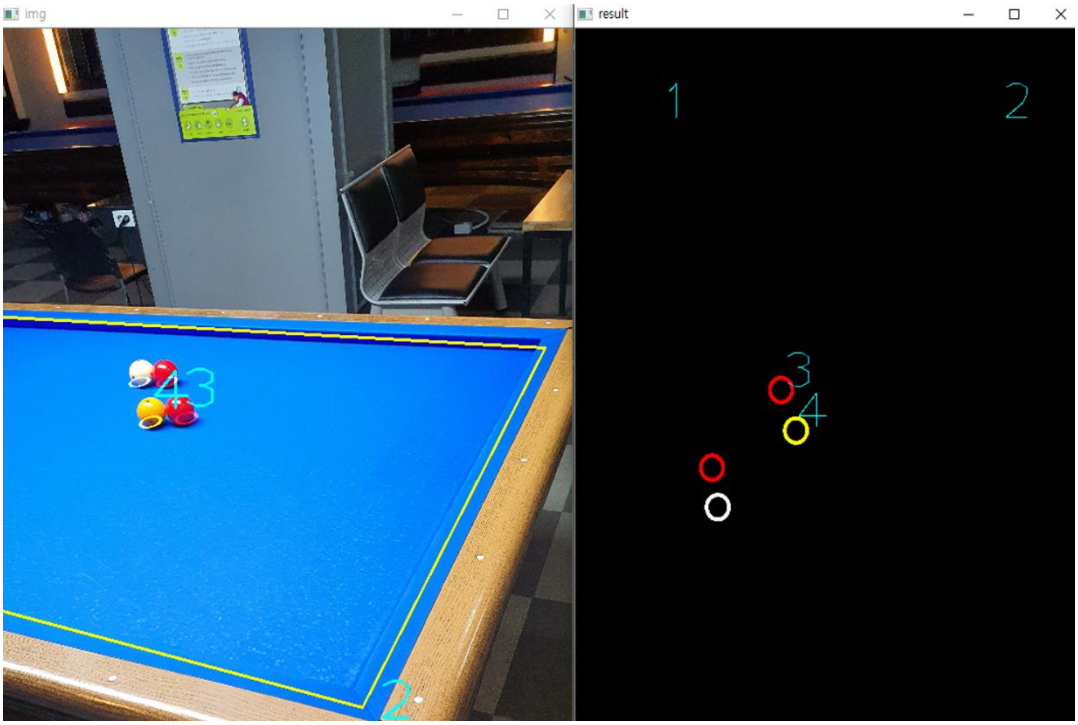
아래의 그림은 당구공의 월드 좌표를 구하는 그림입니다. 아래의 그림과 같이 이미지 위의 당구공을 월드 좌표로 변환하면 **직선과 같은 형태**가 됩니다. 이 직선을 하나의 점으로 결정하기 위해 당구공의 반지름만큼 띄워져 있는 초록색 평면과의 교점을 구하면 됩니다. 정리하면 초록색 평면과 영상의 각각의 당구공을 월드 좌표로 변환하였을 때 나타내는 직선의 교점이 당구공의 중심 좌표가 됩니다.



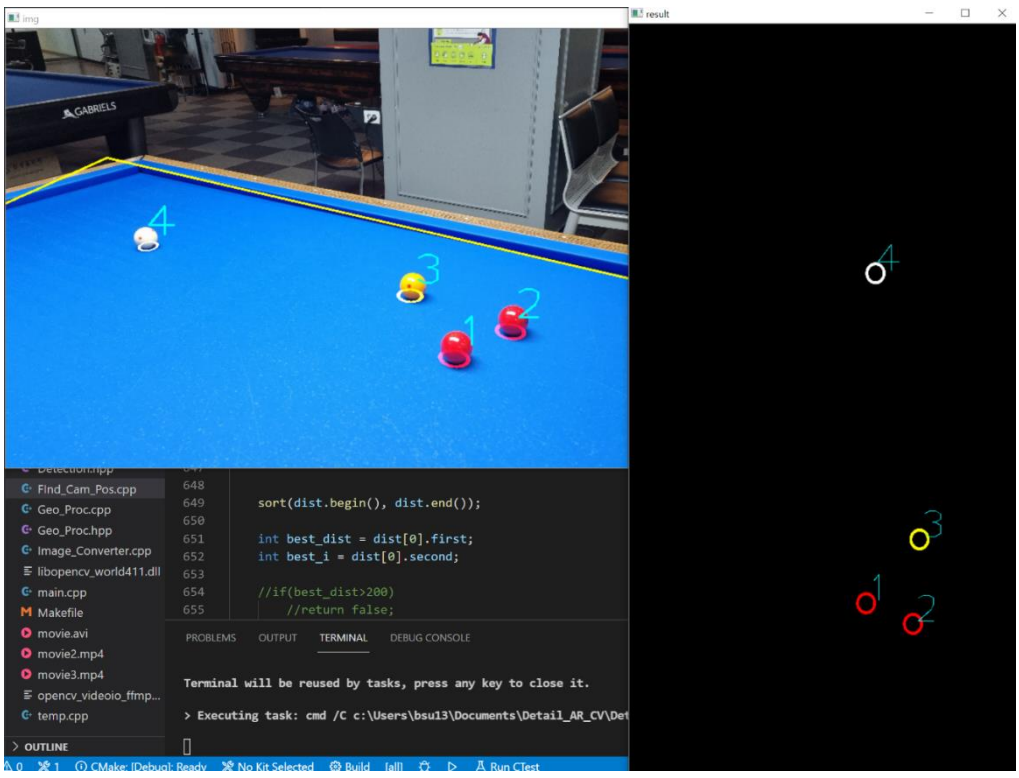
아래의 그림은 당구공의 3D world 좌표를 구한 모습입니다. 여기서 구한 당구공의 3D 좌표는 득점 경로를 계산해 주는 솔루션 클래스에 전달됩니다.



이렇게 해서 당구공의 3D 월드 좌표를 구할 수 있게 되었습니다. 이렇게 구한 당구공의 월드 좌표와 기존에 있던 당구대의 코너를 이용해서 지속적으로 카메라 포즈를 추정할 수 있게 됩니다. 앞서 설명했듯이 카메라 포즈를 추정하기 위해 변환 행렬을 구하려면 영상 좌표와 월드 좌표계에서 각각 4개의 특징점이 있으면 되기 때문입니다. 여기서 당구공과 당구대의 모서리 모두가 특징점의 역할을 하게 됩니다. 예를 들어 아래의 그림은 당구공 두 개와 당구대 코너 2개를 이용하여 변환 행렬을 구할 수 있습니다.



아래의 그림은 당구공 4개만을 이용하여 물체를 증강시킨 그림입니다.



(3) 부드러운 증강현실을 위한 동영상 처리

최종적으로 구현되는 증강현실은 정적인 영상이 아닌 동영상입니다. 물체가 자연스럽게 증강되도록 구현하기 위해 매 프레임마다 물체를 증강시키는 방식이 아닌 특정 조건이 되면 물체를 다시 증강시켰습니다. 예를 들어 사용자가 카메라를 가만히 들고 서 있으면 물체가 매번 업데이트될 필요가 없으므로 전 프레임과 비교하여 특정 움직임 이상이 되면 다시 물체를 증강시키도록 했습니다. 또한 특정 프레임에서는 당구대의 코너와 당구공의 중심 좌표가 일시적으로 구해지지 않는 현상이 있었는데, 이는 이전 프레임의 정보를 이용하여 당구공의 중심 좌표 및 당구대의 코너를 이용하도록 하였습니다.

3. 안드로이드 애플리케이션

카메라의 경우에는 openCV에서 해상도를 이용해서 계산을 하기 위해서 적절한 해상도로 고정할 필요가 있었습니다. OpenCV Camera를 가져와서 받아온 해상도를 테스트를 할 때 대다수 핸드폰에서 전부 작동하기 위해서 720 * 480의 해상도로 고정했습니다.
그 외로는 NDK를 이용해서 C++과 통신하며, 현재 OpenCV에서 어느 정도까지 계산이 되었는지 파악을 하면서 적절한 토스트 메시지를 보여주며 사용자와 소통합니다.

4. 득점 경로 계산 알고리즘

기하학적 처리를 거쳐 계산된 당구대와 당구공 위치를 통해서 적절한 솔루션을 찾아야 합니다.
솔루션은 모든 방향을 탐색하는 완전 탐색을 이용해서 구현을 했습니다. 또한, 해당 애플리케이션은 당구를 많이 접하지 않은 초보자를 위해 나온 애플리케이션입니다. 따라서 당구대의 벽을 많이 이용해서 나오는 솔루션은 적합하지 않다고 판단을 했습니다. 솔루션은 벽을 맞추지 않고 제 1 적구에서 제 2적구로 직접 가는 경로나, 벽을 1번 튕겨서 맞추는 경로 두 경우를 제공합니다.
당구공의 솔루션을 찾을 때 특이사항이 하나 있습니다. 애플리케이션이 완벽하지 않아서 당구공이 당구대 좌표계를 벗어난 경우가 생깁니다. 이럴 때 당구공의 위치는 음수 값을 가지고 있어서, 음수를 가지고 있는 경우 전체적으로 모두 양수의 값을 가지게끔 스케일링을 진행한 후 솔루션을 찾습니다.
마지막으로 솔루션의 경로를 보여주는데, 이때는 스케일링이 되어있는 값을 고려하여 애플리케이션의 적절한 위치에 경로를 보여주면서 해당 경로를 마무리합니다.

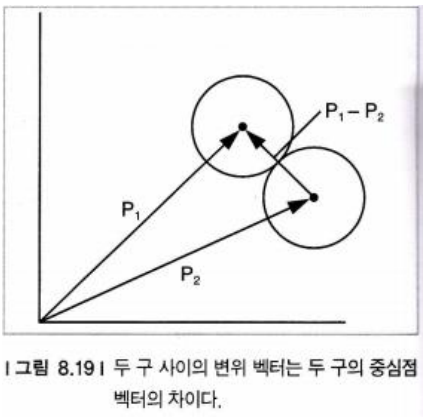
솔루션을 구하는 완전 탐색 과정에서는 당구공들이 움직여야 하고, 충돌해야 합니다. 공은 4개가 있으므로 중복되는 내용이 있으므로 Ball Class를 만들어서 사용했습니다. 해당 클래스에서 들어가는 내용은 다음과 같습니다.

자료 형	변수 명	변수 설명
Point2d	Locate	공의 위치
Point2d	Speed	공의 속도(벡터)
Int	Color	공의 색상 정보
Ball	Other1	다른 공1
Ball	Other2	다른 공2
Ball	Other3	다른 공3
Bool	Collision1	공1과의 충돌 여부
Bool	Collsion2	공2과의 충돌 여부
bool	Collsion3	공3과의 충돌 여부

솔루션의 과정은 다음과 같습니다.
계산된 공의 위치를 받아옵니다. 이때, 받아온 공의 위치는 음의 값을 가질 수 있기 때문에 특정 방향으로 움직여 모든 위치의 값이 양수로 변하게 스케일링을 먼저 진행합니다. 그리고 경로를 탐색하는 완전 탐색 과정이 시작됩니다.

완전 탐색을 시작하면 지금 쳐야 하는 색상의 공을 움직입니다. 예를 들면, 노란 공을 특정 방향으로 움직이기 시작합니다. 움직이면서, 벽과 충돌하거나 공과 충돌할 수 있습니다. 공과 충돌했을 경우 정지해 있던 충돌된 공이 운동량을 가지고 움직이기 시작합니다. 그러다가, 흰색 공과 부딪히거나(정답이 아닌 상황) 빨간 공을 2번 맞추면(정답인 상황)이 되면 해당 속도의 경로 탐색을 그만둡니다.

충돌은 다음과 같습니다.



| 그림 8.19 | 두 구 사이의 변위 벡터는 두 구의 중심점 벡터의 차이다.

그림 1 당구공 충돌

충돌은 당구공의 반지름이 33cm이므로 움직였을 때 두 공의 중심의 거리가 66의 값보다 작았을 경우 충돌이라고 인식합니다. 충돌은 [그림 1]과 같이 P2가 움직이다가 P1과 충돌한다면 P1의 방향은 P1-P2로 운동합니다. P2는 원래 방향에서 (P1 – P2)를 뺀 방향으로 움직입니다. 이때 충돌 전후의 전체 운동량은 같다고 가정하여 내적을 이용해서 구했습니다.

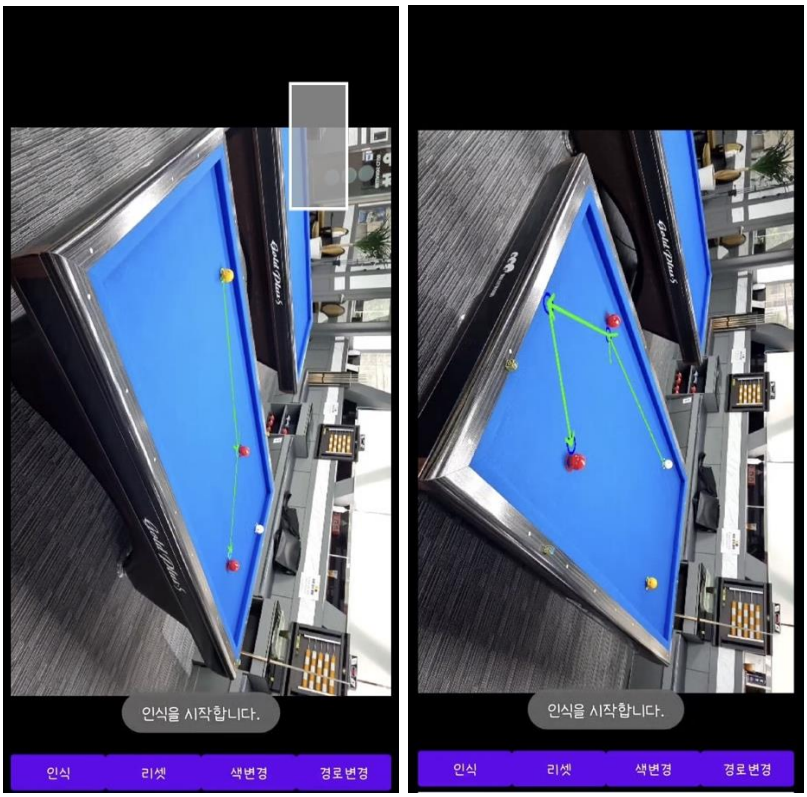
유효한 솔루션을 구하기 위해서는 현재 당구공들이 유효하게 충돌을 했는지 검사를 해야 합니다. 예를 들면, 노란 공을 기준으로 빨간 공 2개를 맞추어도 도중에 흰색 공을 맞추게 된다면 해당 경로를

보여주면 안 됩니다. 그래서 유효성을 검사해야 합니다. 유효성 검사는 어떤 공과 부딪히는지를 나타내는 변수를 사용해서 유효하지 않은 충돌을 걸러냈습니다.

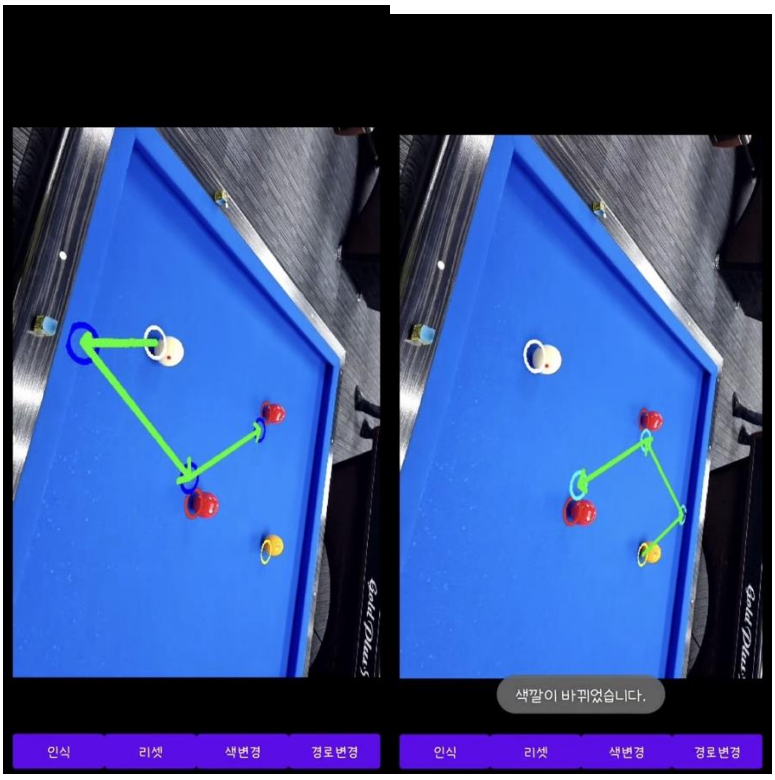
솔루션 도출 과정에서 해당 애플리케이션을 사용할 때 성능을 느리게 만드는 경우가 있었습니다. 처음에는 구현할 때에는 "경로 변경", "색깔 변경" 버튼을 눌렀을 때마다 새롭게 계산을 하려고 했으나, 사용자가 이동할 때마다 계산을 계속하는 이슈가 있어 성능의 문제가 있었습니다. 그래서, 처음 "인식 버튼"을 눌렀을 때 한 번에 노란색과 하얀색 두 공을 기준으로 모든 경로를 저장을 해놓는 방향으로 바꾸었고, 최적화를 진행할 수 있었습니다

5. 결과 및 평가

1. **결과**
   - 벽을 맞추지 않는 경우 & 벽을 1번 맞추는 경우



   - 같은 배치 다른 경로 출력



2. 평가

1) 당구공과 당구대 인식이 매 프레임마다 정확히 가능한가?
2) 계산된 당구공과 당구대의 3D 월드 좌표가 정확한가?
3) 물체를 자연스럽게 증강시킬 수 있는가?
4) 계산된 득점 경로는 정확하며 사용자에게 유익한가?
5) 전체적인 알고리즘의 속도는 어떠 한가?

3. 달성한 목표

1) 대다수의 핸드폰에서 작동을 하는가?
   A. 해상도를 720 * 480으로 고정했기 때문에 다수의 핸드폰의 기기와 호환이 된다.

2) 사용자 층에게 솔루션이 어렵지 않은가?
   A. 벽을 부딪히는 횟수를 최소화 한 경로를 먼저 보여주면서 솔루션의 난이도를 낮추었다. 추가적으로 회전이 있으면 솔루션이 더 예뻐질 수는 있으나 초보자에게 더 어려워질 것이라고 판단했고, 회전을 넣지 않아 솔루션이 직관적으로 보인다. 따라서 초보자가 쉽게 따라 할 수 있을 것이라 생각한다.

3) 실제로 솔루션처럼 공을 칠 수 있는가?
   A. 실제 당구장에서 테스트를 했을 때 똑같은 경로로 칠 수 있었다.

4) 득점 경로를 실시간으로 증강시킬 수 있는가?
   A. 득점 경로 계산은 물체가 증강되기 전 한 번만 시행되도록 구현하였으며 물체 인식 및 3D 기하학적 처리를 최대한 OpenCV가 제공하는 함수로 구현하였기 때문에 문제없이 부드럽게 작동한다

5) 다양한 당구장 환경에서 애플리케이션 사용이 가능한가?
   A. 파란색 당구대이고, 당구대 위에 특별한 물체가 없고, 적당한 조명이 있다면 대부분 강건하게 잘 작동한다.

6) 사용자가 애플리케이션을 쉽게 조작할 수 있는가?
   A. 안내 문구 및 버튼을 추가하여 조작을 편하게 하였다.

4. 추후 방향

실제 애플리케이션을 작동시키면 증강된 물체가 깜박이거나 어느 순간에는 인식이 잘 안되는 경우가 있었습니다. 또한 계산된 득점 경로대로 당구공을 쳐도 득점을 할 수 없는 경우도 있었습니다. OpenCV 영상처리를 통한 물체의 3D 좌표 계산 및 증강현실 구현에서 부족함을 느꼈습니다. 당구공과 당구대 인식을 조금 더 강건하게 되도록 보강해야 합니다. 또한 득점 경로 계산도 인공지능의 강화 학습 기법을 적용하거나 회전 및 타점을 알려주는 옵션까지 넣어 발전시키고 싶습니다.