지능형 로봇

팀 프로젝트

최종 보고서

1. 프로젝트 소개

프로젝트 명: 당구의 신

당구는 정확성과 전략이 중요한 스포츠로, 특히 4 구에서는 최적의 경로를 아는 것이 중요하다. 본 프로젝트는 컴퓨터 비전 기술과 임베디드 보드를 활용하여 당구 4 구 게임의 최적 해를 계산하는 프로그램 제작을 목표로 하였다. 카메라로 당구대를 촬영하여 당구대와 각 색깔 별 공의 위치를 인식하고, 점수를 획득하기 위해 공을 쳐야 할 최적의 방향을 실시간으로 영상에 표시하는 방식으로 구현하였다.

4 구의 규칙

2 명의 Player 가 각각 Yellow 공, White 공을 갖고, 자신의 공을 쳐서 Red 2 개를 맞추면 득점이다. 만약 상대의 공을 맞추거나 아무 공도 맞추지 못하면 실점한다.

2. 구현 과정

작동 흐름

1. 당구대와 당구공을 인식한다.
2. 당구대의 꼭짓점과 당구공의 2D pixel 좌표를 당구대 평면을 Z=0으로 하는 3D world 좌표계로 변환한다.
3. 변환한 좌표를 이용하여 공을 칠 방향을 계산하고 저장한다.
4. 칠 방향을 시각화한다.
5. 키를 누를 때까지 1,2,4단계를 반복한다. 키를 누르는 경우 공을 칠 방향을 재계산하여 반영한다.

당구대 인식

다음과 같은 순서로 당구대를 인식하였다.

1. 실시간 frame에서 당구대 색상에 해당하는 부분을 cv2.inRange() 함수를 통해서 탐지한다.
2. 1번에서 당구공으로 인해 탐지된 영역에 구멍이 발생하는데, 이 부분을 Morphology 닫기 연산을 통해 구멍을 채운다. 이 단계에서 Morphology 연산을 하면서 noise도 제거된다.
3. Morphology 침식 연산과 cv2.substract() 함수를 사용하여 탐지한 영역의 Edge

부분만 추출한다.

4. 추출한 Edge에서 4개의 직선을 추출한다. 이때 Hough 변환을 사용하여 각 직선의 (r, θ)를 얻을 수 있다. Hough 연산만 할 경우 noise로 인하여 한 직선에 대해 여러 직선을 인식할 수 있는데, 이때 당구대의 색상으로 탐지한 영역의 중심 좌표를 활용하여 중심과 각 모서리 별 가장 가까운 직선만 사용한다. 이렇게 총 4개의 직선을 얻을 수 있다.
5. 구한 4개의 직선 간의 교점을 계산하고 이 좌표를 당구대의 꼭짓점으로 사용한다. 4개의 직선에 대해 최대 6개의 교점이 발생할 수 있다. 이 부분에 대해서는 이미지 내에 있는 교점만 사용하도록 구현하였다.

당구공 인식

다음과 같은 순서로 당구공을 인식하였다.

1. 3 classes로 라벨링된 당구공 이미지들을 약 300장 생성하여 Pretrain된 YOLOv8 모델에 학습하였다.
2. 매 frame마다 YOLO 모델을 적용한다.
3. 탐지한 공들의 bounding box의 중심 좌표를 최종 공의 좌표로 사용한다. 이때 한 객체에 대해 bounding box가 여러 개 생길 수 있는데, 이 부분을 Non maximum suppression 기법을 활용하여 중복되는 bounding box를 제거하였다. 또한 탐지된 당구대의 꼭짓점이 4개이고, 탐지된 공의 중심 좌표가 당구대 내에 위치하며, Red 2개 Yellow 1개, White 1개를 만족하는 경우에만 해를 계산하도록 하였다.

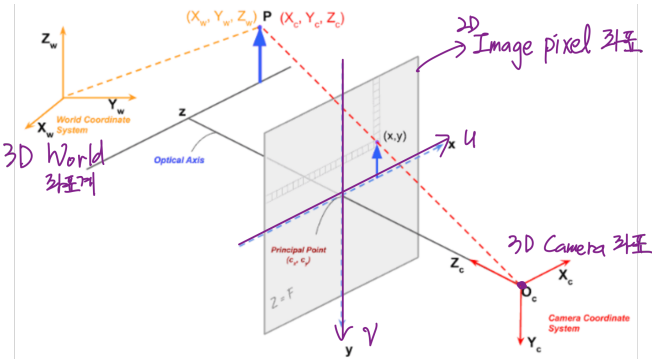
좌표 변환

$$z_c \times \underbrace{\begin{pmatrix} u \\ v \\ 1 \end{pmatrix}}_{\text{2D pixel 좌표}} = \begin{pmatrix} \\ K \\ (3 \times 3) \end{pmatrix} \begin{pmatrix} \\ R \\ (3 \times 3) \end{pmatrix} \underbrace{\begin{pmatrix} x_w \\ y_w \\ z_w \end{pmatrix}}_{\text{3D world 좌표}} + \begin{pmatrix} \\ K \\ (3 \times 3) \end{pmatrix} \begin{pmatrix} t_x \\ t_y \\ t_z \end{pmatrix}$$

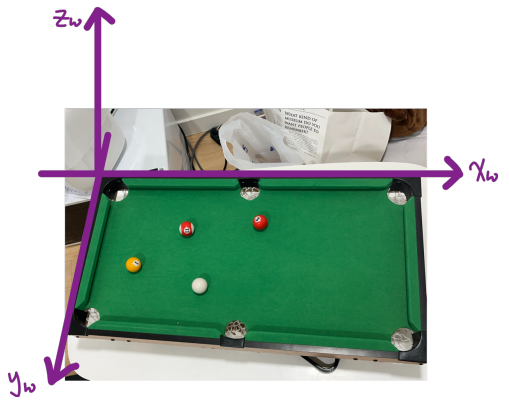
위의 수식을 활용하여 2D pixel 좌표 <-> 3D world 좌표 간 변환이 가능하다. 두 좌표계 간 변환을 하기 위해 2D pixel 좌표계, 3D camera 좌표계, 3D world 좌표계 총 3개의 좌표계가 필요하다. 또한 수식에서 필요한 parameter는 Intrinsic matrix K, Rotation matrix R, Translation vector t가 있다.

우선 2D pixel 좌표계는 이미지의 가로를 u축, 세로를 v축으로 하는 좌표계이다. 인식된 당구공의 좌표계는 이 2D pixel 좌표계의 값을 얻게 된다.

3D camera 좌표계는 (x_c, y_c, z_c)의 3차원 값을 표현하며, 카메라를 기준으로 한 좌표계를 나타낸다. x_c는 카메라를 기준으로 왼쪽에서 오른쪽 방향의 좌표계, y_c는 카메라를 기준으로 아래에서 위 방향의 좌표계, z_c는 카메라에서 물체까지의 거리를 나타내는 좌표계이다.



마지막으로 3D world 좌표계는 현실을 기준으로 좌표계이며, (x_w, y_w, z_w)로 표현한다. 당구대의 평면을 $z = 0$으로 하여 진행하였다. 당구대의 크기가 600mm * 300mm이기 때문에 x_w는 0~600, y_w는 0~300, z_w는 0의 값을 갖는다.

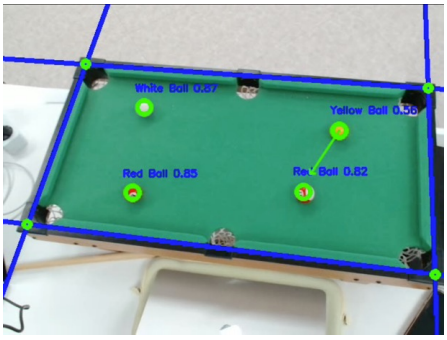


카메라 Calibration이 추가적으로 필요한데, 과정은 카메라 고유의 parameter를 추정하는 과정이다. 이 과정을 통해 카메라의 초점 거리, 광학 중심, 왜곡 계수를 얻을 수 있으며 최종적으로 사용하는 값은 Intrinsic matrix(3 * 3)과 Distortion matrix(5 * 1)이다.

Intrinsic matrix와 Distortion matrix, 당구대 꼭짓점의 2D pixel 좌표, 3D world 좌표를

cv2.solveePnP() 함수를 통해 Rotation vector(3 *1)와 Translation vector(3 * 1)를 얻을 수 있다. 이렇게 얻은 Rotation vector를 cv2.Rodrigues() 함수를 통해 Rotation matrix로 변환할 수 있다.

이렇게 얻은 parameter들을 사용하여 위 수식을 적용할 수 있다. 해 계산을 위해 당구공의 2D pixel 좌표를 3D world 좌표로 변환한다. 그리고 해 계산을 통해 공을 칠 방향을 얻고 Yellow를 기준으로 칠 방향 위의 한 점을 선택한다. 이 점은 3D world 좌표계로 표현되는데, 이를 2D pixel 좌표로 변환 후 Yellow의 2D pixel 좌표와 연결하여 실시간 동영상에 시각화한다.



최적의 각도 도출

- **물리 엔진 구현:** 당구 4구의 해를 계산하기 위해 물리 엔진을 설계하였으며, 이 엔진은 공의 움직임, 충돌 조건, 마찰력 등을 고려하여 최적의 각도를 도출하는 역할을 수행한다.
- **당구대 초기화:** 먼저 2D로 변환된 각 공의 초기 위치를 입력으로 받아 당구대 환경을 초기화한다.
- **타겟 공에 대한 히팅 가능한 각도 계산:** 노란색 공으로 빨간 공 두 공을 맞출 수 있는 범위를 각각 계산한다. 계산 과정은 아래와 같다.
    1. 목표 각도 계산: $\theta = arctan\frac{y_t - y_c}{x_t - x_c}$
    2. 히팅 가능 범위 계산: $\alpha = arcsin\frac{2R}{d}$
    3. 최종 히팅 가능한 각도 범위: $[\theta - \alpha, \theta + \alpha]$

- **물리 시뮬레이션:** 가능한 각도에 대해 공의 이동 경로를 시뮬레이션하고 결과를 저장해 놓는다. 시뮬레이션을 위해 사용한 공식들은 아래와 같다.
    - 속도와 위치 업데이트
        - $v = v * \mu$
        - $p = p + v * dt$
    - 공과 벽의 충돌
        - $v = -v * e$
    - 공과 공의 충돌

운동량 보존

- $m_1v_1 + m_2v_2 = m_1v_1' + m_2v_2'$

비탄성 충돌

- $v_{2,normal}' - v_{1,normal}' = -e(v_{2,normal}' - v_{1,normal}')$

접선 방향 속도 보존

- $v_{1,tagnet}' = v_{1,tangent}',\ v_{2,tagnet}' = v_{2,tangent}'$

- **우선순위가 가장 높은 각도 도출:** 저장한 각도들 중에서 아래의 우선순위가 가장 높은 각도를 도출한다. 사람이 치기 자연스러운 각도를 도출하도록 우선순위를 선정하였다.

1. 성공 여부
2. 첫 충돌이 빨간 공인지
3. 충돌 횟수가 몇 번인지

- **성능 분석**: 매 번 무작위로 공의 초기 위치들을 설정하고 시뮬레이션을 진행하는 테스트를 1000번 진행하였다. 테스트 결과 대략 86.1% 정도의 성공 확률을 보였고, 평균 계산 시간은 대략 2.136초 정도 되었다.

3.프로젝트의 잠재적 확장성 및 개선 사항

- **다양한 경로 탐색**: 점수를 내기 위한 최적 경로가 여러 개일 수 있다. 여러 경로 중에서 사용자가 선택을 할 수 있도록 다양한 경로를 사용자에게 제공하는 기능을 구현할 수 있다.
- **다양한 게임 모드 지원**: 이 프로젝트는 당구 4 구에 특화된 알고리즘을 개발하였지만, 추후 3 구나 포켓볼의 다른 당구 게임에도 적용할 수 있는 알고리즘을 개발할 수 있다.
- **더욱 정교한 물리 모델링**: 현재의 경로 계산은 공의 단순한 이동과 충돌을 기반으로 하고 있지만, 추후에는 공의 회전을 고려한 정밀한 모델링을 추가하여 더 현실감 있는 최적 경로 계획 기능을 개발할 수 있다.
- **휴머노이드 로봇**: 당구를 실제로 칠 수 있는 휴머노이드 로봇에서 이 프로그램을 구동하여 당구를 직접 칠 수 있는 로봇을 개발할 수 있다.

4. 회고

처음에는 당구대와 당구공을 인식하는 과정에서 Perspective Transform을 이용하여 당구대의 꼭짓점을 찾은 후, 이 그림을 당구대 크기에 맞게 변환한 후 YOLO 모델을 적용

하여 당구공을 인식하려 하였다. 하지만 Perspective Transform을 적용하는 과정에서 이미지가 과도하게 왜곡되어 이 방법으로 구현하지 못하였다. 하지만, 실제로 구현한 방법을 공부하면서 수업 시간에서 다루었던 Rotation matrix, Translation matrix, Homogeneous 좌표계 등을 적용하여 배울 점이 많았다.

최적의 경로를 도출하는 과정에서 강화학습을 시도하였지만, 성공 확률이 40%정도에 그치었다. 시뮬레이션 구성과 보상 설계를 적절하게 구현하지 못한 것 같다. 특히 보상 함수 설계를 어떻게 정의하느냐에 따라 모델이 학습하는 방향이 크게 달라졌다. 처음에는 목표 공과의 충돌 성공 여부를 보상으로 설정하였지만, 충돌 이후의 경로 평가나 충돌 횟수 최소화와 같은 세부 조건을 반영하지 못해 학습이 제대로 이루어지지 않았다. 또한, 에피소드마다 공의 초기 위치가 무작위로 설정되었기 때문에 학습 데이터의 일관성이 부족했고, 이를 개선하지 못한 점도 성능 저하의 원인으로 작용했다. 강화 학습에 대한 충분한 이해를 바탕으로 시뮬레이션 환경을 정교하게 구성하고 보상 함수의 세부 조건을 개선하여, 향후에는 강화 학습을 통해 최적의 경로를 도출하도록 구현할 것이다.