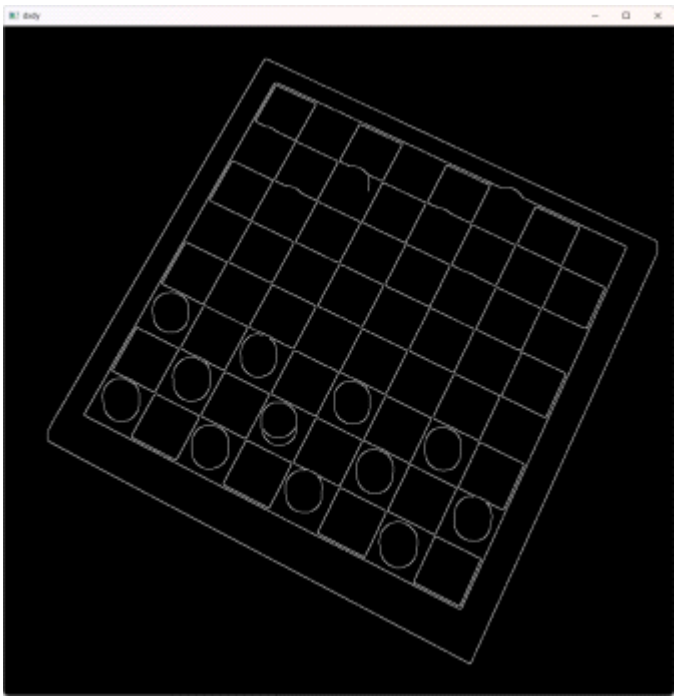
컴퓨터비전 HW01 보고서

컴퓨터학부 20221494 박찬솔

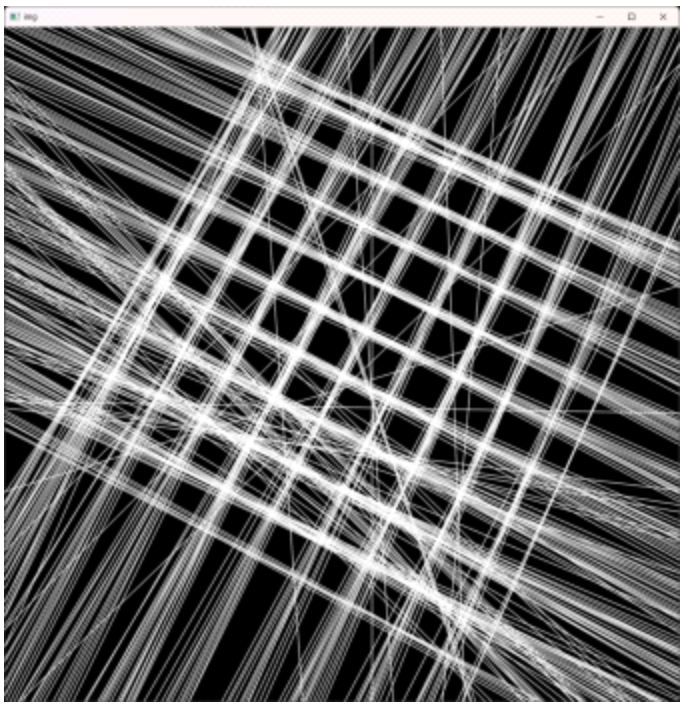
목차

1. 아이디어

Canny로 에지를 찾고, HoughLines를 사용해서 가능한 모든 선을 검출합니다. (이 영상을 dxdy라고 정의합니다.) 에지가 검출된 위치에만 선을 긋지 않고, 길이가 무한한 직선을 긋습니다. (이 영상을 hough라고 정의합니다.)

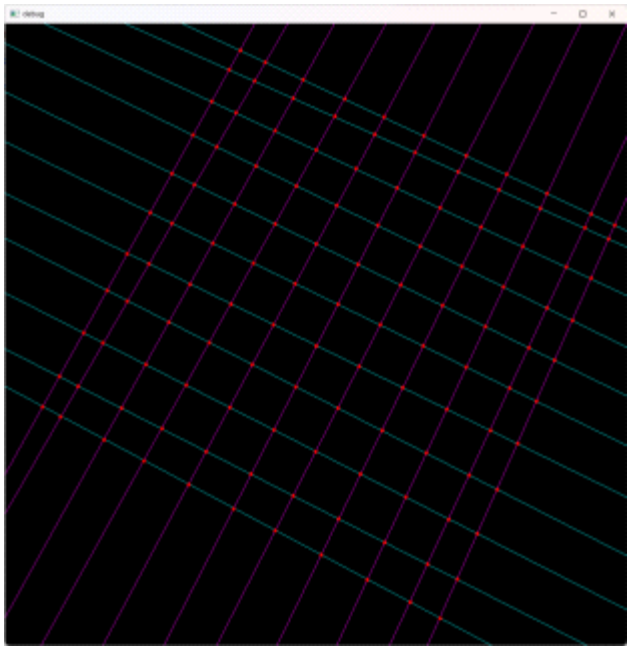


dxdy



hough

hough 영상에는 노이즈, 기물, 또는 판 외부 환경에 의한 직선이 생기며, 같은 성분의 직선도 여러 개 생깁니다. 이를 효과적으로 제거하기 위해서, K-means 알고리즘을 사용하여 직선을 수평 또는 수직으로 나눕니다. (k=2) K-means 알고리즘을 사용한 후, 수평선과 평행한 선들을 X, 수직선과 평행한 선들을 Y라고 정의합니다. 체커판을 구성하는 직선은 서로 평행하여 만나지 않을 것입니다. X, Y에서 같은 그룹에 속하는 직선끼리 교점이 영상 내부에 생긴다면, 그 직선은 올바르지 않은 직선일 가능성이 큽니다. 또한, 교점이 생기지 않더라도(평행한 직선) 충분히 가까우면 그 직선들을 지웁니다.

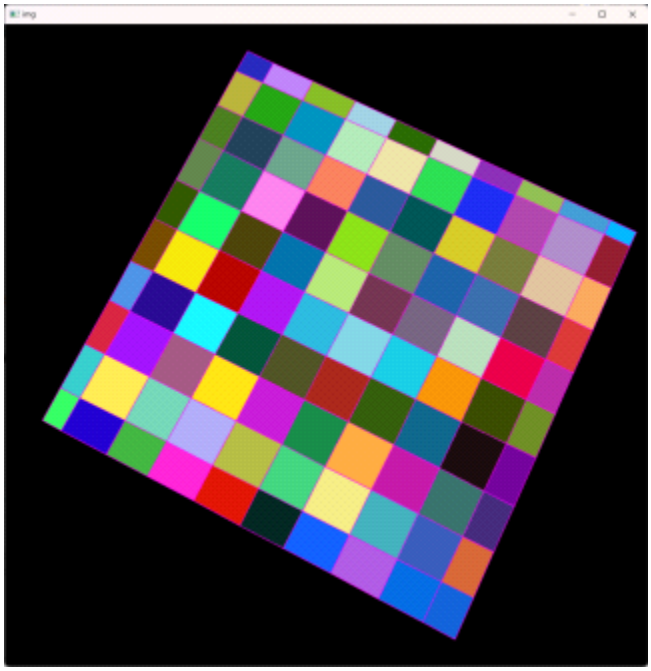


res

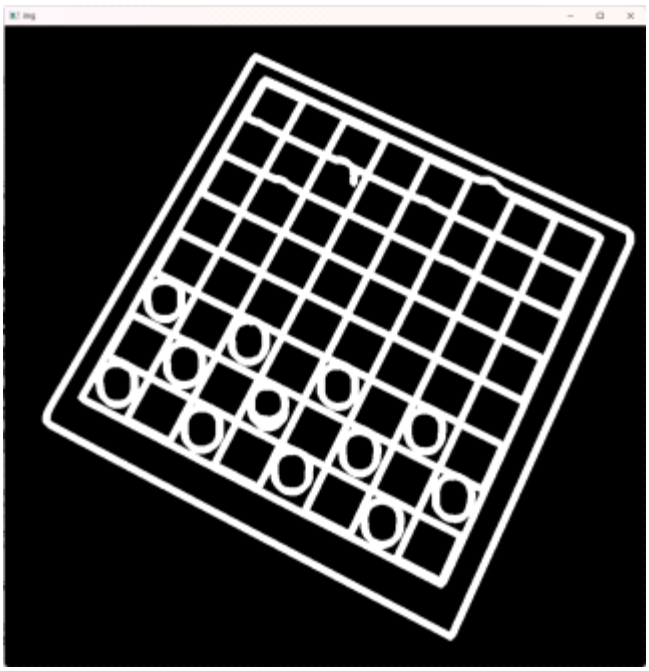
이제 모든 X와 Y에 대해 생기는 교점을 그려봅니다. (이 영상을 res라고 정의합니다.)

잘 구해진 것 같지만, 여전히 체스판 최외각 테두리 등과 만나 생기는 불필요한 교점(선)이 존재합니다. 이는 dxdy 영상을 이용해서 선을 하나씩 지울 예정입니다.

먼저, 인접한 교점을 4개씩 골라 사각형을 만듭니다. 인접한 점 두 개만 고정하면 나머지 두 점은 자동으로 고정됩니다. 따라서, 인접한 서로 다른 두 점에 대해서 사각형을 만드는 작업을 합니다.



판별되는 사각형

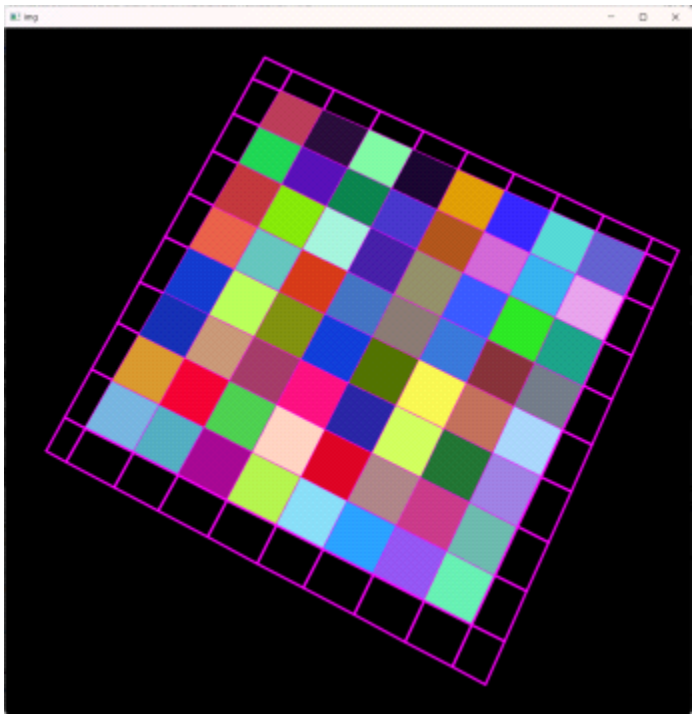


applied dilate operation to dxdy

dxdy 영상에서 + 모양의 kernel를 가지고 dilate(팽창) 연산을 여러 번 수행합니다. res 영상에서 교점들에 의해 생기는 선분과 이 영상을 논리적 AND하면, 원래 그 선분이 있던 것인지 없던 것인데 새로 생긴 것인지 알 수 있습니다. 사각형의 네 선분이 원래 있던 선분인지 아닌지를 구별하여, 원래 있던 선분의 개수를 셉니다. 3개 이상이면 accept하고, 2개 이하이면 우선 reject합니다.

observation. 체커판을 구성하는 선들은 accept되는 사각형의 비율이 상당히 높을 것입니다. 이 비율을 선의 reference rate라고 하겠습니다. 내부에 있는 선들은 최외각 선과 만나는 경우를 빼고는 거의 모든 사각형이 accept됩니다. 또한, 체커판 가장 바깥쪽에 있는 선은 최외각 선과 직접적으로 맞닿으므로 reject된 사각형의 비율이 절반 정도 될 것입니다.

claim. 사각형의 네 선 중 reference rate가 65% 이상인 선이 3개 이상이면 accept합니다.

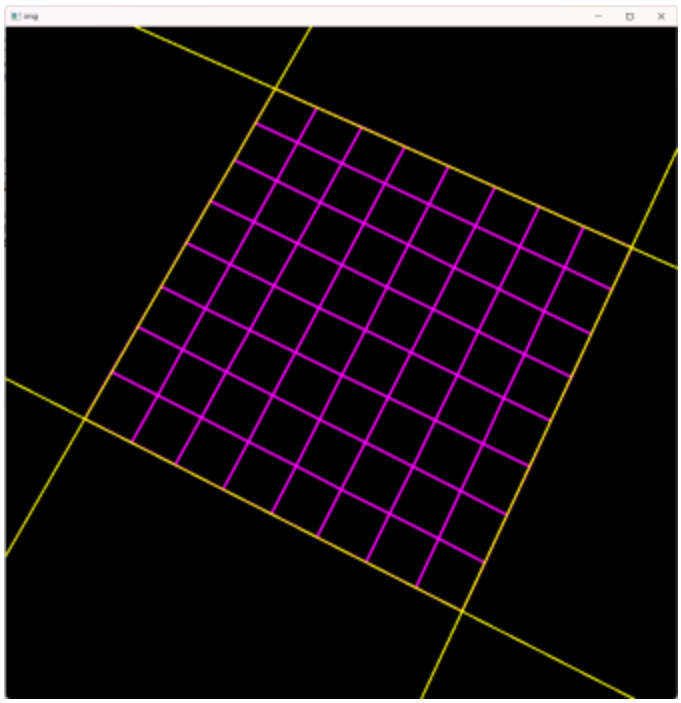


이런 식으로 불필요한 사각형을 지우게 되면 남은 사각형은 옆과 같습니다. 이 남은 사각형들만 체커판으로써 취급할 것입니다.

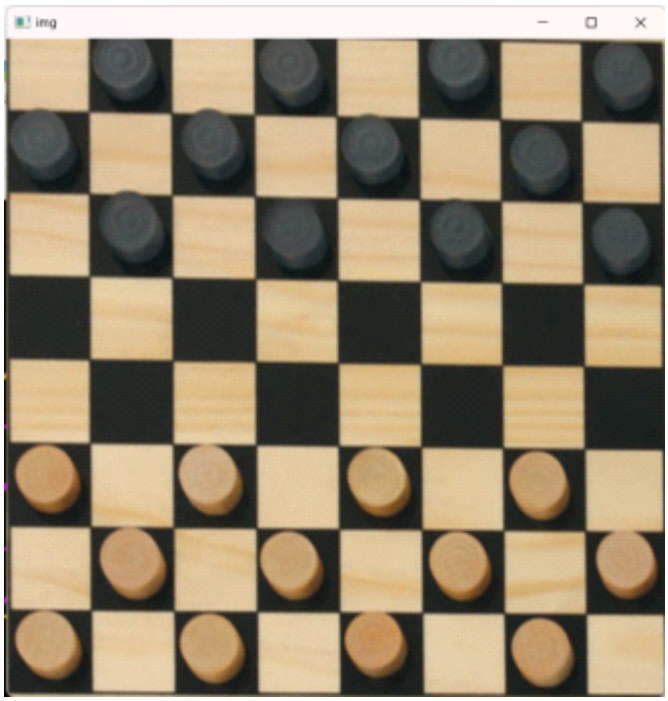
노이즈 등으로 인해 완전히 다른 영역에서도 사각형이 검출될 수 있습니다. Union-Find 알고리즘을 이용해서 실제로 인접한 사각형들끼리 연결하여 컴포넌트를 구성합니다. 컴포넌트 중 사각형의 개수가 가장 많은 것을 체커판으로써 택합니다.

체커판의 테두리는 reference rate와 비슷하게 선이 사각형에 의해 참조되는 개수를 세는 reference count를 이용해 구합니다. 체커판 내부의 선들은 상하좌우 4개의 사각형들에 의해 참조되지만, 체커판 테두리 선들은 상하 또는 좌우로 2개의 사각형에 의해 참조되므로, 참조 횟수가 상대적으로 적을 것입니다.

또한, 참조 횟수의 최댓값은 체커판의 가로와 세로 길이를 동시에 의미하기도 합니다. 한 직선에 참조된 횟수가 n이라면, n개의 사각형이 그 직선을 따라 있었다는 뜻입니다. 이는 가로(혹은 세로)의 길이가 n임을 의미합니다. **[과제 1 풀이]**



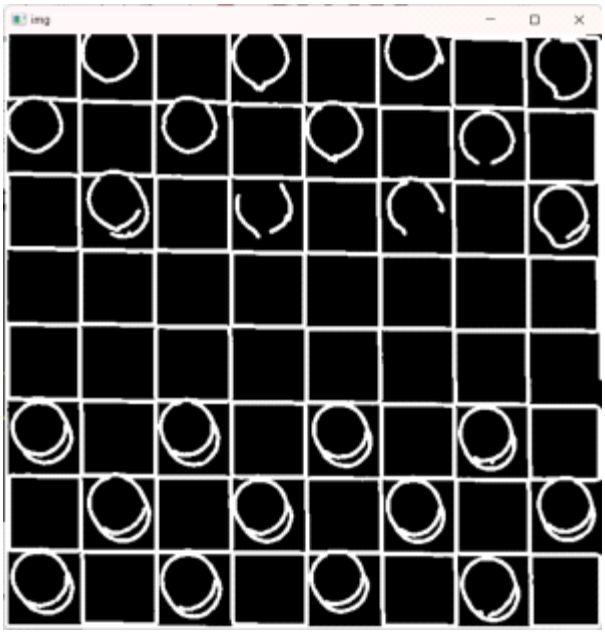
구해진 테두리 및 테두리 교점



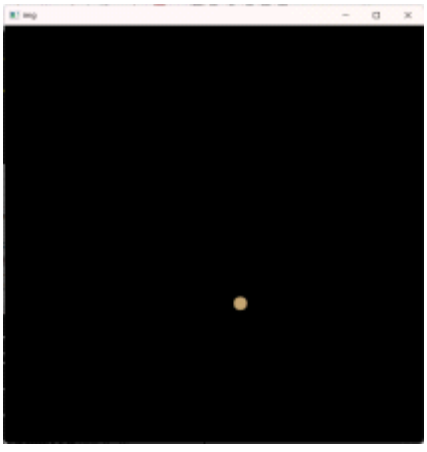
투시 변환 결과

4개의 테두리 선을 가지고 교점을 구합니다. 이 교점을 시계 방향으로 바꿔주는 것도 필요한데, CCW를 사용해서 반시계 방향이면 점을 swap해주는 것으로 시계 방향으로 강제할 수 있습니다. **[과제 3 풀이]**

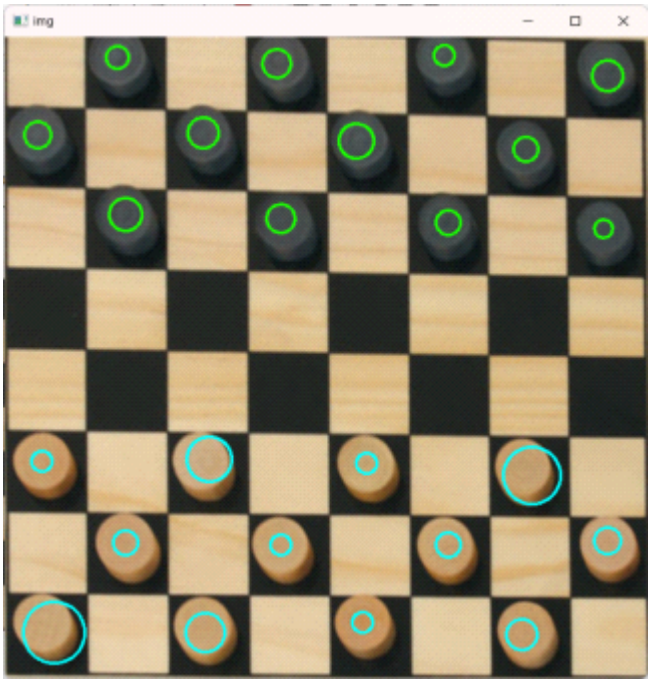
마지막으로 기물의 개수와 색을 판별합니다. 다른 관점(perspective)에서 찍힌 사진을 가지고 HoughCircles를 사용하면 정확도가 매우 낮아집니다. 따라서, Perspective Transform을 수행한 결과 영상에서 HoughCircles를 사용합니다. HoughCircles를 사용하기 전에, Canny와 Threshold(OTSU)를 이용하여 에지로 인식되는 부분을 강조되게 만듭니다.



원으로 검출되는 부분을 칠하고 있는 mask 영상을 생성합니다. 이 영상과 원본 영상(투시 변환한)과 논리 AND 연산을 하면 아래와 같이 기물의 영역만 나옵니다. 이 정보들만 별도로 저장하여 색을 구분합니다.



마지막으로 색을 구분합니다. 색은 HLS 채널에서 밝기 영역만을 사용하며, K-means 알고리즘을 사용하여 가지고 있는 색을 두 종류로 분류합니다. (k=2) 마지막으로, 나뉘어진 두 그룹 중에서 그룹의 중앙값 중 더 큰 것을 밝은 기물로써 취급합니다. **[과제 4 풀이]**

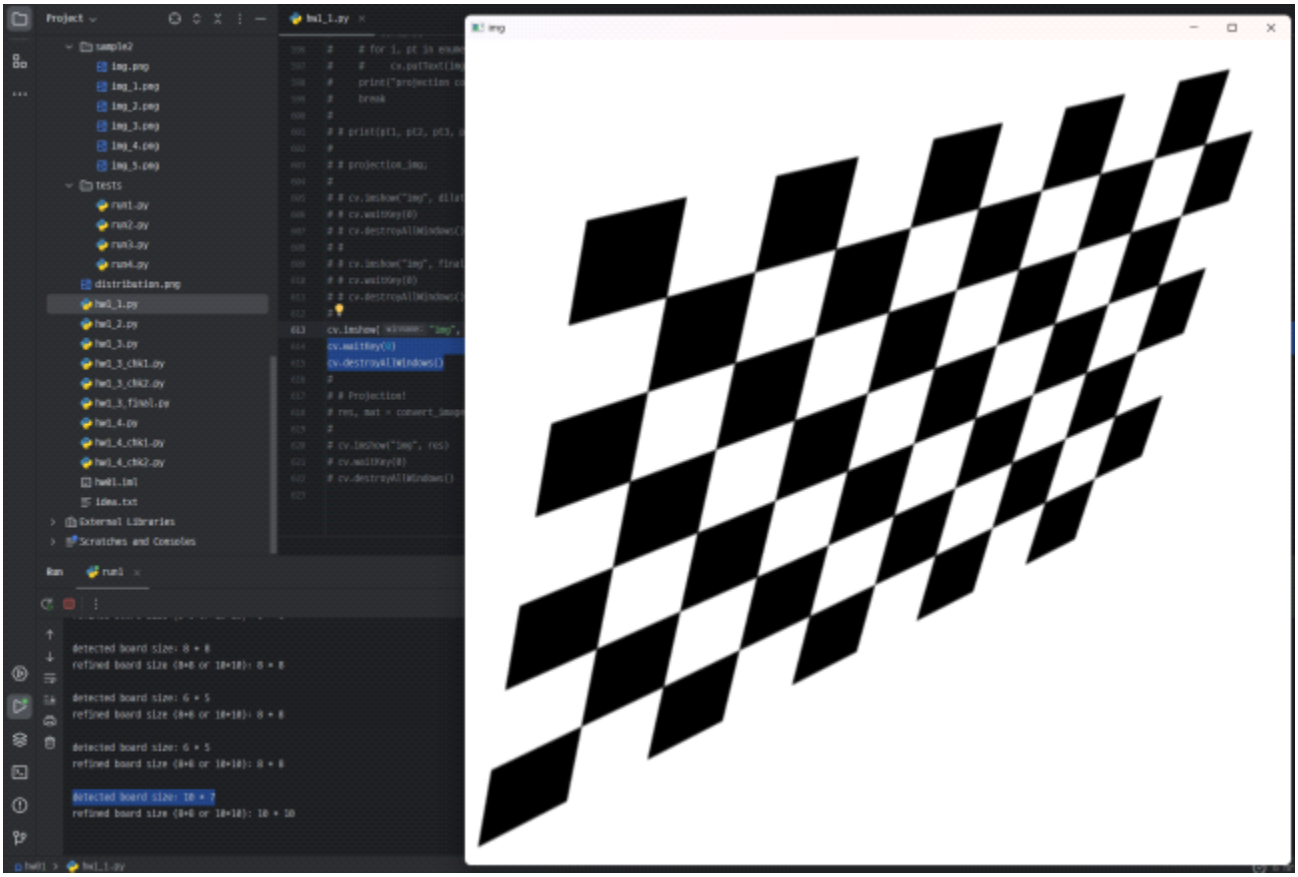


기물에 종류별로 다른 색으로 나타낸 영상

2. 실행 결과

과제 1.

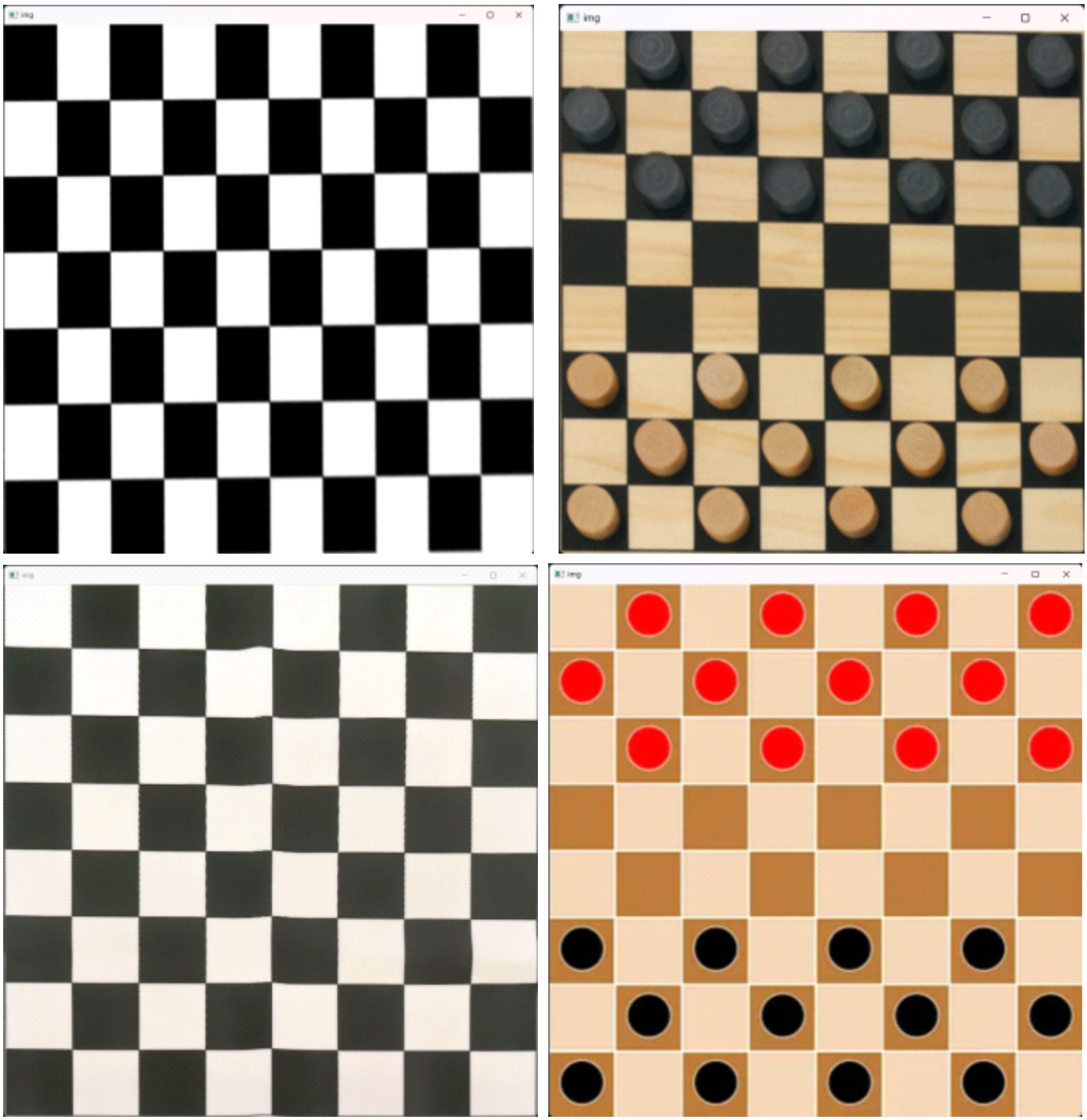
* 명세에 적힌 대로 8 * 8 또는 10 * 10 보드인지 판별하는 경우, 아랫줄에 출력되는 refined board size가 답입니다.
* 주어지는 체커판의 크기가 자유로울 경우, 첫 줄에 출력되는 detected board size가 답입니다.



과제 2.

과제 3.

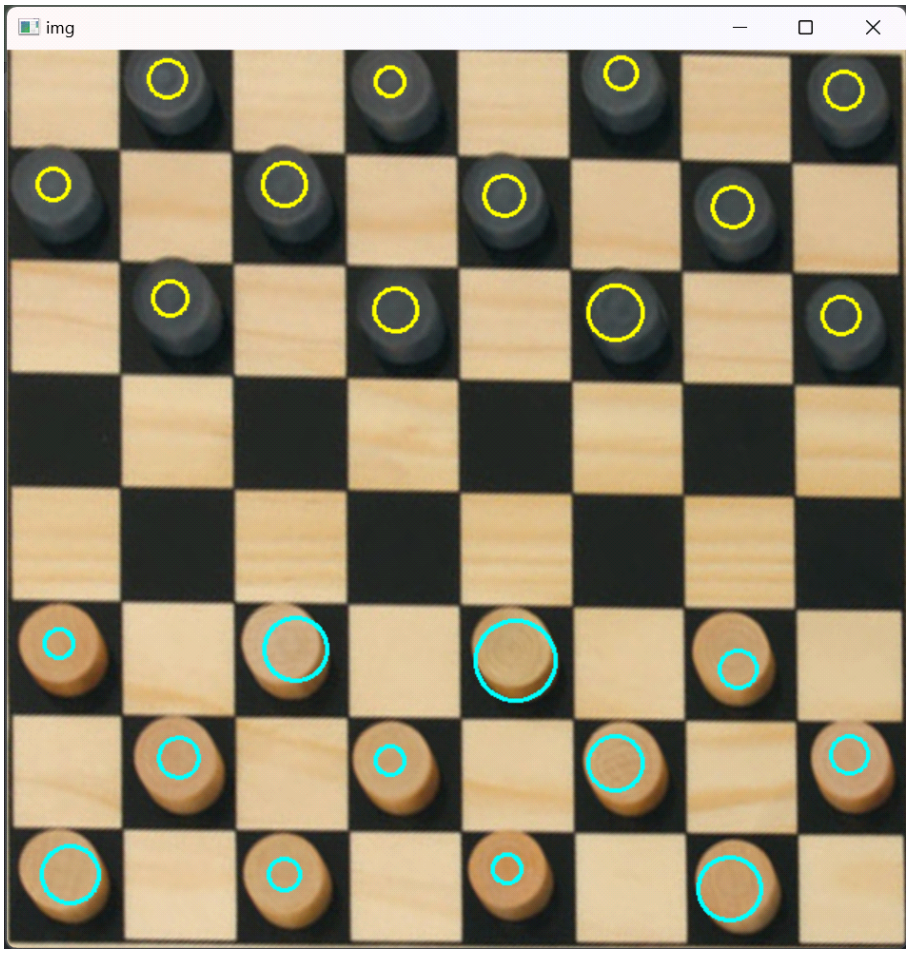
* 과제 2에서 수동으로 찍어야 했던 점의 좌표를 출력하고, 투시 변환을 수행한 결과 영상을 보여줍니다.

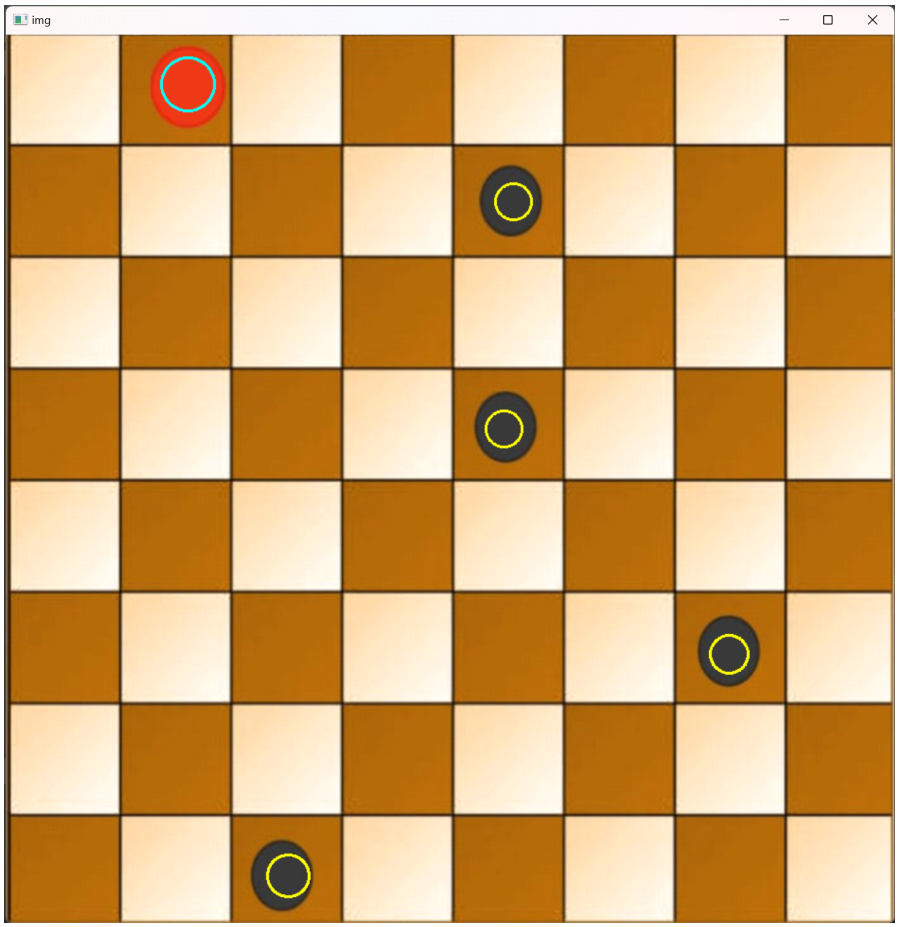


실행 결과 중 일부

과제 4.

* 명세에 제시된 형식대로 밝은 말과 어두운 말의 개수를 출력합니다.





3. 테스트 도구 실행 방법

과제 메인 폴더에서,

- 과제 1 : python3 tests/run1.py
- 과제 2 : python3 tests/run2.py
- 과제 3 : python3 tests/run3.py
- 과제 4 : python3 tests/run4.py

샘플 데이터

- 과제 1, 2, 3 : dataset_13/
- 과제 4 : dataset_4/