종합설계 설계보고서

**<영상처리를 이용한 객체인식과 2D Mapping>**

20153073 이동빈

20163010 이동인

20163022 이유영

20163040 장현욱

20163044 정범희

제출일: 2021년 6월 14일

**설계보고서 목차**

1. **작품개요**
2. **작품 세부내용**

2.1 경기장 영역 검출

2.2 선수 및 공 인식

2.3 2D 변환행렬 찾기와 2D 이미지로의 변환

2.4 개선점

1. **작품요약 및 기대효과**

**1. 작품개요**

**1.1 설계 필요성**

21세기로 넘어오면서 많은 분야에서 IT 기술을 접목하여 많은 발전이 이루어졌다. 그 중에서도 IT 기술은 많은 사람이 즐기고 열광하는 스포츠와 결합하여 선수 개개인의 데이터 분석을 넘어 리그 전체의 경기를 분석하여 리그 전체의 퀄리티를 높이는 데 도움을 준다. 스포츠와 결합한 최신 IT기술은 경기의 질을 높이는 기능을 넘어서 시청자들이 경기를 더 재밌게 즐길 수 있도록 도움을 주고 있다.

많은 IT 기술 중, 축구 트래킹 시스템은 이미 많은 리그, 구단에서 적용한 기술이다. Bepro11이라는 회사는 실시간으로 축구 경기를 분석하는 프로그램을 개발하여 유럽의 여러 팀들과 계약을 하였고 유비스랩이라는 회사에서는 웨어러블 기기를 이용하여 일반인들이 참여하는 조기축구에서도 데이터를 분석할 수 있도록 도와주는 장비를 개발한다고 한다.

이처럼 축구와 결합한 트래킹 시스템은 프로 축구선수와 유명 구단의 전유물이 아닌 일반인들도 직접 경험하면서 축구를 더욱더 재밌게 즐길 수 있도록 도와주는 기술이 되었다. 따라서, SPOCV 조에서는 이런 축구 분석 프로그램의 기초인 선수 움직임과 경기장 인식을 설계주제로 정하였다.

**1.2 작품 설명**

SPOCV 팀은 앞에서 설명한 설계 필요성을 인식하고 기본적인 축구 분석 프로그램을 만들고자 하였다. 이 축구 분석 프로그램은 고정된 앵글에서 촬영된 축구영상을 입력으로 받아, 경기장 영역 검출, 객체 인식, 변환행렬 찾기, 2D 이미지로의 변환 과정을 통해 축구영상에서 인식한 경기장과 선수들, 심판, 공을 2D 이미지에서 Bird’s Eye View로 나타내고자 한다. 이를 통해, 축구 분석 프로그램의 가장 기본바탕이 되는 선수들의 움직임을 인식하고자 하는 것을 목표로 설계를 진행하였다.

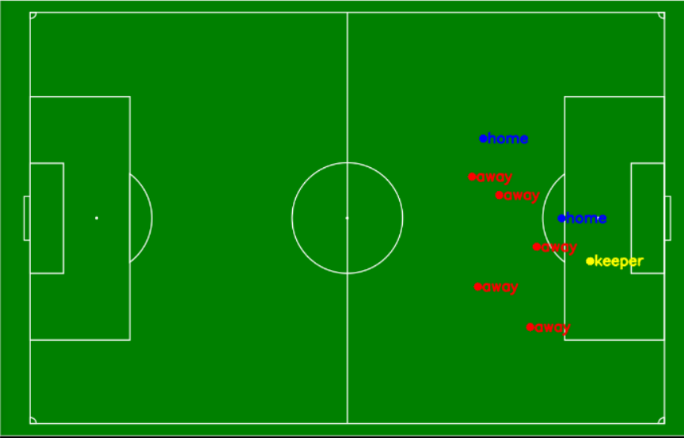


그림 1 (좌) 사용한 원본 축구영상 (우) 원하는 2D 이미지로의 변환 결과

**2. 작품 세부내용**

SPOCV 조의 설계목표인 선수의 움직임을 인식하고 2D 화면으로 Mapping하여 Bird’s Eye View로 나타내기 위해서는 크게 3가지의 단계를 거쳐야 한다. 3가지 단계 중 첫번째로는 경기장 영역을 검출하는 단계, 두번째로는 선수들을 인식하는 단계, 세번째로는 2D Mapping을 하기 위한 변환행렬을 구하고 2D 이미지로 Mapping을 진행하는 단계이다. 경기장 영역을 검출하는 것의 목표는 경기장 외부의 대기좌석, 광고판 등을 제거하기 위한 것이고, 선수들을 인식하는 것의 목표는 영상에서 선수들을 인식하여 인식한 선수들의 위치를 찾는 것이며, 변환행렬을 구하는 것의 목표는 축구영상에서 나오는 경기장으로부터 정보를 얻어 원본영상에서 인식한 것들을 목표로 하는 2D 이미지로 변환하기 위한 행렬을 구하는 것이다. 여기서, 경기장으로부터 얻을 수 있는 정보는 경기장에 그어져 있는 선들의 교점이다.

**2.1 경기장 영역 검출**

앞에서 설명한 경기장 안의 선들의 교점을 얻기 위해서는 이미지에서 얻고자 하는 교점을 가진 선분들의 특징이 잘 나타날 수 있도록 도와주는 전처리과정(마스크 씌우기, 필터적용)을 거쳐야 한다.

전처리과정의 마스크를 씌우는 과정은 경기장 외부의 대기좌석, 광고판, 경기장 내부의 경기를 뛰는 선수들을 삭제해주는 과정이다. 이것은 HSV 공간에서의 순수한 경기장 내부의 이미지만 얻을 수 있는 색의 범위를 지정하여 진행되었다. 마스크를 씌우는 과정 이후에는 영상의 Noise를 제거하기 위하여 MedianBlur 함수를 이용한다.



그림 2 원본 영상에서 마스크를 씌운 결과

**2.2 선수 및 공 인식**

선수 및 공 인식을 진행하기 위해서는 이미 검출한 경기장 영역으로 객체인식 범위를 조정한 뒤, 홈 팀, 원정 팀, 골키퍼, 심판, 공 총 5개의 객체를 구분해야 한다. 따라서, Trackbar를 이용해 각각의 HSV 값의 범위를 구하였다. 다음의 표 1은 초기 Trackbar를 이용하여 찾은 객체 각각의 HSV 임계값 범위이다.

|  |  |  |
| --- | --- | --- |
|  | HSV\_Range\_Lower | HSV\_Range\_Upper |
| 홈팀 | [103, 99, 0] | [179, 191, 184] |
| 원정팀 | [0, 0, 140] | [170, 50, 255] |
| 골키퍼(Away) | [0, 59, 0] | [24, 111, 255] |
| 심판 | [0, 112, 106] | [10, 255, 255] |
| 공 | [0, 0, 0] | [0, 0, 255] |

표 1 Trackbar를 이용하여 찾은 객체들의 HSV 단일 임계값 범위

입력 받은 이미지에 Morphology 작업을 진행하고 FindContours 함수를 이용하여 Contour를 찾는다. 이후, BoundingRect 함수를 통해서 Bounding Box의 범위를 지정하였다. 이전에 추출한 HSV 값을 이용하여 Mask를 씌운 후, CountNonZero 함수를 이용하여 검은색이 아닌 픽셀의 수를 구했다. 이 값을 객체를 인식할 수 있는 민감도로 정의하였고, 객체별 민감도를 조건으로 하여 Bounding Box를 씌우고 객체를 쉽게 구별할 수 있도록 Text를 입력하였다.



그림 3 초기 객체 인식 사진

그림 3은 초기 HSV 단일 임계값 조건으로 검출한 초기 객체인식한 그림이다. 위 그림을 확인하면 두가지의 문제점을 찾을 수 있었다.

첫번째 문제점으로 선수 인식에 비해 공의 인식률이 굉장히 낮았다. 이는 선수에 비해 공의 크기가 매우 작기 때문에 인식이 제대로 되지 않았기 때문이라고 예측하였다. 두번째 문제점으로 모든 객체들의 인식률이 현저히 낮으며 BoundingBox의 겹침 현상도 일어났음을 확인할 수 있었다. 이는 단일 HSV 임계값 설정으로 인해 일어나는 문제점이라고 예측하였다.

첫번째 문제를 해결하기 위해 우리는 그림 4와 같이 공과 선수의 사이즈가 크게 차이 난다는 특징을 이용하여 공을 인식할 때, BoundingRect 함수로 얻은 객체인식의 범위를 선수보다 작은 범위로 조정하였다.



그림 4 (좌) 선수의 Bounding Box 크기 (우) 공의 Bounding Box 크기

두번째 문제를 해결하기 위해 앞서 이용한 Trackbar를 사용하여 다중 임계값 범위를 추출하였다. 표 2는 인식이 잘 되지 않은 특정 객체(원정팀, 공)에 대하여 HSV 임계값 범위를 다양하게 추출한 표이다. 이를 기존의 HSV 임계값 범위에 추가하여 민감도를 향상시켜 객체 인식률을 개선하였다.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | HSV\_Range  \_Lower | HSV\_Range\_  Upper | HSV\_Range\_  Lower\_plus1 | HSV\_Range\_  Upper\_plus1 | HSV\_Range\_  Lower\_plus2 | HSV\_Range\_  Upper\_plus2 |
| 홈팀 | [103, 99, 0] | [179, 191, 184] |  |  |  |  |
| 원정팀 | [0, 0, 140] | [170, 50, 255] | [0, 0, 140] | [155,40,255] | [0.0.202] | [109, 85, 255] |
| 골키퍼(Away) | [0, 59, 0] | [24, 111, 255] |  |  |  |  |
| 심판 | [0, 112, 106] | [10, 255, 255] |  |  |  |  |
| 공 | [0, 0, 0] | [0, 0, 255] | [0,0,140] | [155, 14, 255] |  |  |

표 2 Trackbar를 이용하여 추가한 객체들의 HSV 다중 임계값 범위

그림 5는 초기 객체인식에서 찾은 두가지 문제점을 개선한 그림이다. 초기 객체인식 사진에 비해서 공 인식뿐 아니라 전체적인 인식률이 개선된 것을 확인할 수 있었다. 추가적으로 각각의 객체를 겹침 현상없이 정확하게 구분하여 인식하였고 Bounding Box 범위도 적절히 조절하여 선수의 제스처도 실시간으로 인식하도록 개선하였다.

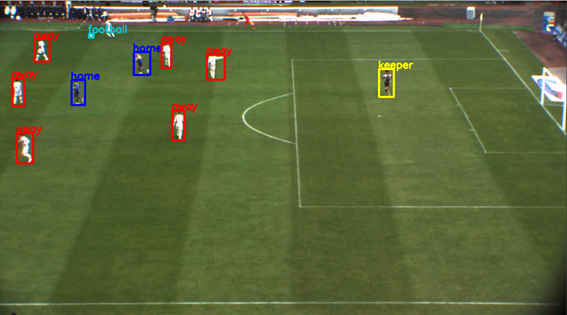


그림 5 인식률 개선 후 선수 인식

**2-3 2D 변환행렬 찾기와 2D 이미지로의 변환**

축구영상을 통해 인식한 선수, 심판, 공을 목표하는 2D 화면으로 나타내기 위해서는 변환행렬을 구해야 한다. 이 변환행렬을 구하는 단계에서 경기장에 존재하는 선을 인식하는 선 인식 과정(2진화, Houghlines 적용), 경기장의 수직, 수평선을 찾는 과정, 선들의 교점을 찾는 과정, 마지막으로 변환행렬을 찾는 과정을 거쳐야 한다.

변환행렬를 찾는 첫번째 과정인 선 인식 과정에서는 필터를 적용한 이미지를 가지고 영상을 AdaptiveThreshold를 적용하여 2진화시킨다. 2진화 이후 얻은 이미지를 가지고 Houghlines를 적용하여 경기장 내의 직선들을 검출한다.

두번째 과정인 경기장의 수직, 수평선을 찾는 과정에서는 Houghlines을 사용하여 얻은 직선들의 Rho, Theta 값들을 살펴보고 교점을 구하고자 하는 경기장에 속한 직선들의 Rho, Theta 값을 찾는다. 찾은 Rho, Theta 값들을 가지고 제어문을 사용해 직선들을 수직, 수평 2가지 그룹으로 나누고 경기장안의 선의 교점을 구한다.



그림 6 (좌) Houghlines를 적용해 얻은 직선 (우) Rho, Theta를 이용해 찾은 수직, 수평선

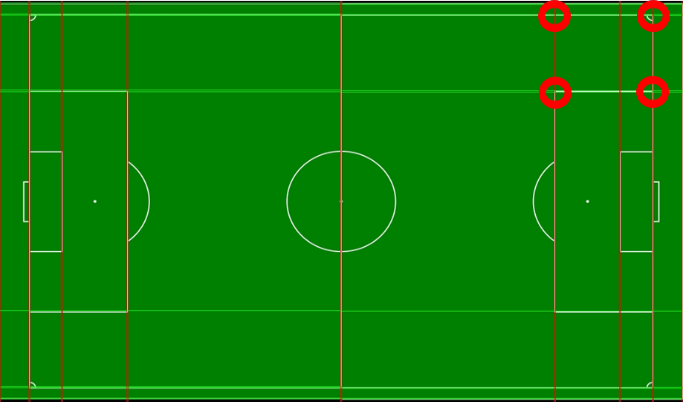


그림 7 영상의 경기장 내에 있는 4개의 교점과 2D 이미지에서의 4개의 교점

마지막 과정은 영상내의 경기장의 교점 중 사용할 4개의 점을 정하고, 목표한 2D 이미지에서 앞서 정한 4개의 점과 대응되는 4개의 점의 위치를 찾아 findHomography 함수를 이용하여 2D mapping에 이용할 변환행렬을 찾는 것이다. 찾은 변환행렬을 가지고 원본영상을 직접 변환해보고 제대로 변환이 되었는지 살펴보는 과정을 거쳤다. 첫번째로 원본 이미지를 변환하기 위한 4개의 점을 정할 때, 그림 7의 오른쪽 그림과 같이 경기장 내에 있는 4개의 점을 이용하고자 하였다

이 4개의 점을 가지고 변환을 해보았을 때, 그림 8의 왼쪽 그림과 같이 경기장의 우측 하단부의 경기장 선이 변환된 영상에 나오지 않은 것을 볼 수 있다. 이것은 우측상단에 모여 있는 점들만 선정하였기 때문이라고 생각하였고, 그림 8의 오른쪽 그림과 같이 경기장 외부에 있는 점들도 활용하여 더 넓은 범위의 4개의 교점을 정하여 변환을 진행하였다.

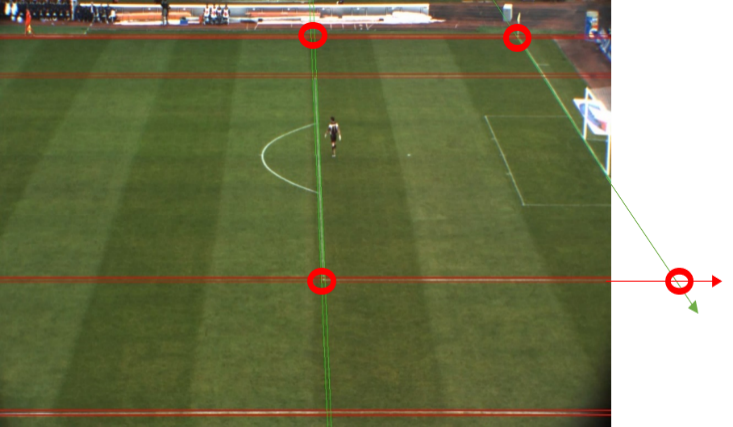


그림 8 (좌) 원본 영상이 변환된 이미지 (우) 새롭게 정한 4개의 교점

더 넓은 범위에서 정한 4개의 교점을 이용해 변환한 결과는 그림 9와 같다. 이것은 변환된 이미지가 2D Mapping 시키려는 그림과 거의 일치하는 것을 확인할 수 있었다. 따라서, 두번째로 시도한 변환행렬을 가지고 2D Mapping을 진행하였다.



그림 9 새로운 교점을 이용해 변환된 경기장

2D 이미지로의 변환 단계에서는 앞서 찾은 변환행렬을 이용하여 목표한 2D 화면에 축구 영상에서 인식한 객체를 점으로 Mapping 시키는 과정이다. 이 과정에서는 2-2의 단계에서 인식한 선수, 심판, 공, 골키퍼의 정보를 통해 선수가 영상 내에 위치한 좌표를 찾는다. 이때, 인식한 객체의 좌표는 선수와 심판은 발 밑, 공은 중심 위치로 정하였다. 이후, 찾은 객체들의 좌표를 변환행렬을 이용하여 계산하고 목표한 2D 화면에 점으로 나타낸다.



그림 10 (좌) 축구 영상에서 객체를 인식한 결과 (우) 목표한 2D 이미지에 Mapping한 결과

**2-4. 개선점**

SPOCV 조에서 목표한 축구영상을 통해 선수들을 인식하고 이를 2D 화면에 나타내는 작품을 만들고 난 후, 2가지의 개선할 문제가 생겼다. 첫번째 문제는 선수들끼리 겹치게 되면 선수 인식과정에서 하나의 Bounding Box로 인식되어 하나의 선수로 인식된다는 점이다. 이것은 한 학기동안 진행된 종합설계 과목에서는 해결하기 어려운 문제로 좀 더 고도화된 트래킹 시스템이 도입되어야 해결할 수 있는 문제이다. 두번째 문제는 축구영상에서 나오는 모든 선수들을 인식하지 못하는 인식률의 문제이다. 우선적으로는 이 문제는 진행한 영상의 화질이 깨끗하지 않았기 때문이라고 생각했다. 따라서, 더 깨끗한 화질의 영상이 사용된다면 객체를 더 잘 인식할 수 있을 것이라고 생각한다.

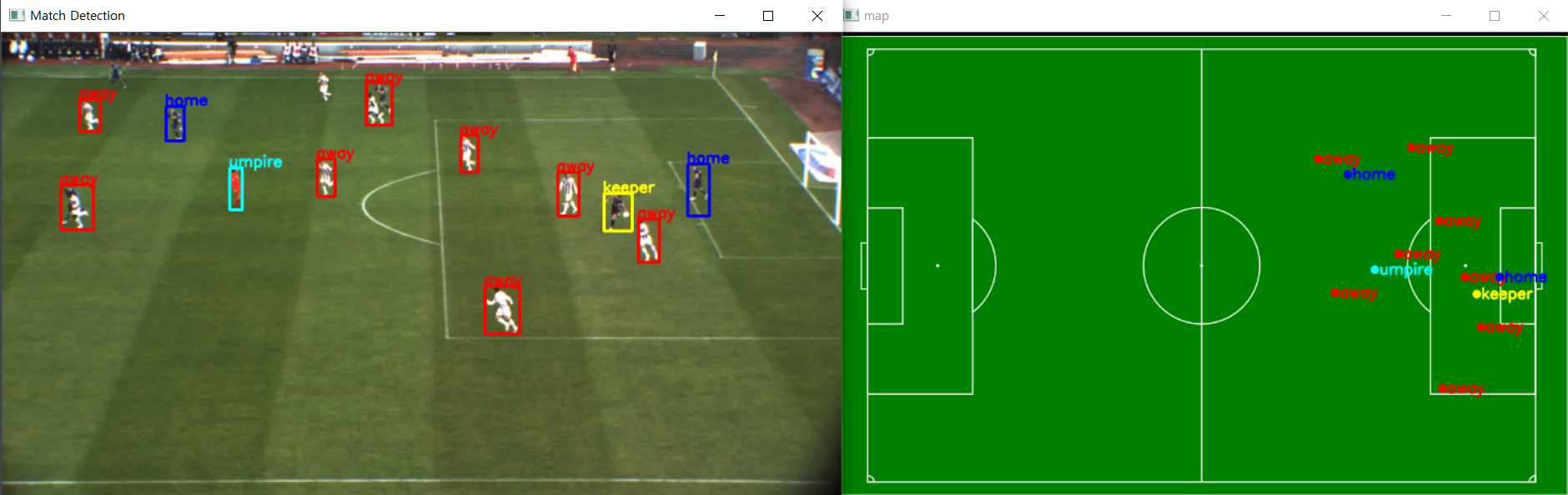
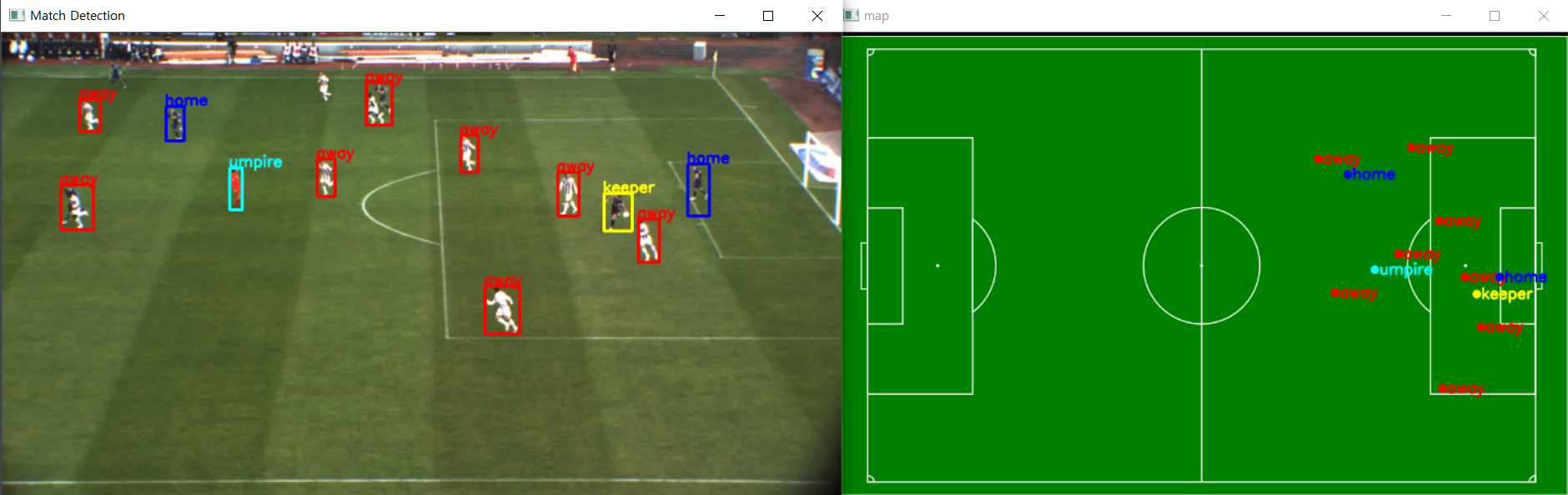


그림 11 (좌) 선수겹침을 나타내는 그림 (우) 선수을 인식하지 못하는 그림

**3. 작품요약 및 기대효과**

SPOCV 조는 축구 영상을 입력으로 받아 2D 화면에 인식한 객체들을 나타내는 것을 목표로 설계를 진행하였다. 설계 목표를 달성하기 위해 경기장 영역 검출, 선수 및 공 인식, 변환행렬을 구하는 과정을 거쳤다. 특히, 선수 및 공 인식 과정에서는 객체인식률을 높이기 위해 단일의 임계값이 아닌 다중 임계값을 사용하여 인식률을 높이고자 하였고, 변환행렬을 구하는 과정에서는 원본 영상을 정확하게 2D화면으로 나타내기 위하여 영상에서 나타난 경기장 내부의 교점이 아닌 영상에서 나타나지 않은 외부의 교점을 추가적으로 이용해 변환행렬을 구하는 노력을 하였다.

SPOCV 조가 만든 작품처럼 경기영상을 2D Mapping하여 2D 이미지 상에 나타내는 이유는 데이터를 통해 선수들의 움직임을 분석하거나 축구의 전술적 측면에서 이용될 수 있기 때문이다. 현재 진행한 프로젝트에서 데이터 분석, 수집, 측정은 아직 불가능하나 이 작업이 기반이 되어 더 발전된 프로젝트를 진행한다면 선수들이 경기 중에 이동한 거리나 속도, 스프린트 횟수 등을 데이터화 할 수 있다고 생각한다. 이러한 데이터를 활용한다면 보다 객관적으로 선수의 능력을 판단할 수 있을 것이다. 추가적으로 선수의 활동반경을 알 수 있는 히트맵을 설계한다면 선수의 전체적인 움직임 데이터를 얻을 수 있다고 생각한다.

이와 같은 활용방안들은 구단들이 경기를 분석하고 더 나은 전략을 세우는데 도움이 될 것이며, 경기를 관람하는 시청자들 역시 이러한 분석프로그램을 통해 보다 쉽고 재미있게 축구경기를 즐길 수 있을 것이다

그림 12 (좌) 선수들이 움직인 거리나 스프린트 속도를 나타내는 프로그램 (우) 축구선수의 히트맵

