**Carom**

**API 문서**

22.07.14 - 22.08.29

소속 **|** 광운대학교

작성 **|** 김민지, 이종윤, 이탁균

(주) 에스프레소 미디어

**목차**

**I. 개**[**요**](#_n6xuoulby2gi) **3**

**II.** [**Detect 파트**](#_oxu4v8czg3ie) **4**

[1. DetectObjectPipe](#_rs5q8quyv7mg) 4

[2. FindEdgePipe](#_kcw6semwgvs3) 5

[3. CheckDetectPipe](#_dkn0x5dlzonw) 6

[4. ConvertToxywhPipe](#_ydbisevrui81) 7

[5. DetectIndexPipe](#_l2h0x3c5dg33) 8

**III.** [**Tracking 파트**](#_i1jdffynttcv) **9**

[1. 클래스 설명](#_4uepsmlz37w7) 9

[2. bagreader.py](#_hffwgoialt1u) 11

[3. balltracker.py](#_7pwc4uc63dqe) 12

[4. curve.py](#_f67b50y3qyof) 14

[5. cvtCoord.py](#_ofz0y8cxi5i1) 16

[6. pointmodify.py](#_tfn0qub0tze7) 17

[7. predict.py](#_2j2tjx1wluyq) 22

[8. ballpredictor.py](#_l964ovdt5nrv) 24

# 

# **개요**

Carom(가명)은 탑 뷰의 당구 영상을 입력 받아 공의 이동 경로와 충돌 이벤트를 출력하는

Detect파트와 Tracking 파트로 분리되어 있다.

Detect 파트에서 각 프레임 별 당구공(이하 공) 3개와 테이블 아스트로 안쪽 코너(이하 모서리) 4개를 탐지하고 Track 파트에서 누락된 경로와 충돌 이벤트를 추론한다.

# **Detect 파트**

## 1. DetectObjectPipe

**기능**

yolov5 를 통해 4개의 모서리와 3개의 공을 탐지하는 파이프 클래스이다. 4개의 모서리는 서로 구분되지 않고 3개의 공도 서로 구분되지 않는다.

**입력**

| 변수명 | 타입 | 설명 | 예시 |
| --- | --- | --- | --- |
| im | 이미지 | 영상을 프레임 단위로 나눈 이미지를 yolo에서 detect를 위해 규격을 조정한 버전  사이즈를 32의 배수로 하기 위해 패딩값이 더해져 있음 |  |

**출력**

| 변수명 | 타입 | 설명 | 예시 |
| --- | --- | --- | --- |
| dets | 리스트 | 객체의 정보를 담고 있는 딕셔너리에 대한 리스트  (해당 딕셔너리는 프레임번호, 바운딩박스 좌상단의 x, y, 바운딩박스 우하단의 x, y, 신뢰도, 객체종류로 구성됨) | [{“frame”: 1, “x”: 0, “y”:0, “w”:10, “h”:10, “conf”: 0.9, “cls”: 0 }, {“frame”: 1, “x”: 10, “y”:10, “w”:20, “h”:20, “conf”: 0.9, “cls”: 0}] |

## 2. FindEdgePipe

**기능**

프레임 별로 탐지된 모서리의 좌표를 한 영상에 대한 모서리로 결정하는 클래스이다.

**입력**

| 변수명 | 타입 | 설명 | 예시 |
| --- | --- | --- | --- |
| im0 | 이미지 | 영상을 프레임 단위로 자른 이미지 |  |
| dets | 리스트 | 탐지된 모서리 객체의 정보를 담고 있는 딕셔너리에 대한 리스트  (해당 딕셔너리는 프레임번호, 바운딩박스 좌상단의 x, y, 바운딩박스 우하단의 x, y, 신뢰도, 객체종류로 구성됨) | [{“frame”: 1, “x”: 0, “y”:0, “w”:10, “h”:10, “conf”: 0.9, “cls”: 0 }, {“frame”: 1, “x”: 10, “y”:10, “w”:20, “h”:20, “conf”: 0.9, “cls”: 0}] |

**출력**

| 변수명 | 타입 | 설명 | 예시 |
| --- | --- | --- | --- |
| result | 딕셔너리 | 당구대 모서리 4개의 좌표   * TL - 좌상단(Top Left) * TR - 우상단(Top Right) * BL - 좌하단(Bottom Left) * BR - 우하단(Bottom Right) | {“TL”: (0,0),  “TR”: (800,0),  “BL”: (0,400),  “BR”: (800,400)} |

## 3. CheckDetectPipe

**기능**

공에 대한 탐지를 정상적으로 진행했는지 확인하는 클래스이다. 한 영상에서 3개 미만이나 3개를 초과하여 공을 탐지한 프레임이 10개 이상이면 비정상적인 영상의 삽입으로 판단한다.

**입력**

| 변수명 | 타입 | 설명 | 예시 |
| --- | --- | --- | --- |
| dets | 리스트 | 탐지된 공 객체의 정보를 담고 있는 딕셔너리에 대한 리스트  (해당 딕셔너리는 프레임번호, 바운딩박스 좌상단의 x, y, 바운딩박스 우하단의 x, y, 신뢰도, 객체종류로 구성됨) | [{“frame”: 1, “x”: 0, “y”:0, “w”:10, “h”:10, “conf”: 0.9, “cls”: 0 }, {“frame”: 1, “x”: 10, “y”:10, “w”:20, “h”:20, “conf”: 0.9, “cls”: 0}  ] |

**출력**

정상적인 영상으로 판단 되면 입력을 그대로 출력하고, 비정상적인 영상으로 판단 되면 NotEnoughDetectError 를 발생시킴

## 4. ConvertToxywhPipe

**기능**

DetectObjectPipe에서 xyxy로 넘겨준 좌표에 대해서 xywh로 변경해 주는 클레스 이다.

**입력**

| 변수명 | 타입 | 설명 | 예시 |
| --- | --- | --- | --- |
| dets | 리스트 | 탐지된 공 객체의 정보를 담고 있는 딕셔너리에 대한 리스트  (해당 딕셔너리는 프레임번호, 바운딩박스 좌상단의 x, y, 바운딩박스 우하단의 x, y, 신뢰도, 객체종류로 구성됨) | [{“frame”: 1, “x”: 0, “y”:0, “w”:10, “h”:10, “conf”: 0.9, “cls”: 0 }, {“frame”: 1, “x”: 10, “y”:10, “w”:20, “h”:20, “conf”: 0.9, “cls”: 0}  ] |

**출력**

| 변수명 | 타입 | 설명 | 예시 |
| --- | --- | --- | --- |
| result | 딕셔너리 | 탐지된 공 객체의 정보를 담고 있는 딕셔너리에 대한 리스트  (해당 딕셔너리는 프레임번호, 바운딩박스 **center-x, center-y, width, height**, 신뢰도, 객체종류로 구성됨) | [{“frame”: 1, “x”: 5, “y”5, “w”:10, “h”:10, “conf”: 0.9, “cls”: 0 }, {“frame”: 1, “x”: 15, “y”:15, “w”10, “h”:10, “conf”: 0.9, “cls”: 0}  ] |

## 5. DetectIndexPipe

**기능**

DetectObjectPipe에서 구분하지 않았던 공을 같은 공에 대해 같은 id를 부여하여 구분짓는 클래스이다. StrongSort를 사용했다.

**입력**

| 변수명 | 타입 | 설명 | 예시 |
| --- | --- | --- | --- |
| im0 | 이미지 | 영상을 프레임 단위로 자른 원본 이미지 |  |
| dets | 리스트 | 탐지된 공 객체의 정보를 담고 있는 딕셔너리에 대한 리스트  (해당 딕셔너리는 프레임번호, 바운딩박스 center-x, center-y, width, height, 신뢰도, 객체종류로 구성됨) | [{“frame”: 1, “x”: 5, “y”5, “w”:10, “h”:10, “conf”: 0.9, “cls”: 0 }, {“frame”: 1, “x”: 15, “y”:15, “w”10, “h”:10, “conf”: 0.9, “cls”: 0}] |

**출력**

| 변수명 | 타입 | 설명 | 예시 |
| --- | --- | --- | --- |
| result | 딕셔너리 | 탐지된 공 객체의 정보를 담고 있는 딕셔너리에 대한 리스트  (해당 딕셔너리는 프레임번호, 바운딩박스 center-x, center-y, width, height, 신뢰도, 객체종류, id로 구성됨) | [{“frame”: 1, “x”: 5, “y”5, “w”:10, “h”:10, “conf”: 0.9, “cls”: 0, “id”:1 }, {“frame”: 1, “x”: 15, “y”:15, “w”10, “h”:10, “conf”: 0.9, “cls”: 0, “id”:0}] |

# 

# Tracking 파트

## 1. 클래스 설명

| Ball Class | | | |
| --- | --- | --- | --- |
| 공 하나에 대한 정보들을 담은 클래스 | | | |
| 멤버 변수 | | | |
| 변수명 | 타입 | 설명 | |
| ball\_list | list | detector에서 전달된 공의 좌표 정보들이 저장된 리스트, 프레임 순서대로 Binfo 객체들이 저장됨 | |
| modified\_  ball\_  list | list | modify\_soft를 통해 검출이 안된 프레임의 좌표를 추론하여 생성하고 modify\_hard를 통해 생성된 공의 좌표 정보들이 저장된 리스트, 프레임 순서대로 Binfo 객체들이 저장됨 | |
| id | int | 객체를 구분하는 id, 1~4는 각각 위/왼쪽/아래/오른쪽 쿠션, 5는 큐볼, 6은 제 1목적구, 7은 제 2목적구 | |
| lalst\_  event\_  obj | int | 마지막으로 충돌한 객체의 id | |
| event\_  frame | list | 충돌이 발생했던 프레임 번호를 저장한 리스트 | |
| last\_  csh\_  dist | float | 마지막으로 쿠션과 충돌했을 때 해당 쿠션까지의 거리 | |
| 멤버 함수 | | | |
| 함수명 | | 설명 | |
| insert\_obj(self, obj) | | ball\_list에 Binfo 객체인 obj를 저장하는 메소드 | |
| check\_move(self, index) | | 해당 인덱스의 프레임에서 공의 이동 여부를 판단하는 메소드. 두 프레임 전의 좌표와 비교하여 2.24(≒√5)픽셀 이상 차이가 나면 이동으로 판단  이동 판단의 기준 값인 2.24를 고정값으로 사용하고 있어 영상의 크기나 영상을 촬영한 카메라와 당구대의 거리 등에 따라 결과가 달라질 수 있음 | |
| add\_move(self) | | detector에서 입력으로 받은 공의 좌표 정보들에 각 프레임에서 공의 이동 여부를 추가하는 메소드 | |
| insert\_predict(self, fr, x, y, index) | | 입력으로 받은 정보들을 보정된 리스트의 특정 인덱스에 저장하는 메소드 | |
| find\_index(self, tar\_frame) | | 보정된 리스트에서 입력 받은 프레임이 몇 번째 인덱스에 있는지 리턴하는 메소드 | |
| add\_event(self, obj1\_id, obj2\_id, event\_frame, dist) | | 입력 받은 객체 1의 id가 해당 객체와 일치하는 정보라면 마지막으로 충돌한 객체, 이벤트가 발생한 프레임, 쿠션과의 충돌이었다면 마지막으로 쿠션과 충돌했을 때의 거리를 갱신하는 메소드 | |
| update\_csh\_event(self, fr\_old, fr\_new) | | 기존의 쿠션과 충돌했던 프레임 번호를 보정된 프레임 번호로 갱신하는 메소드 | |
| delete\_event(self, obj1\_id) | | 마지막으로 충돌했던 프레임 번호를 지우는 메소드 | |
| print(self) | | 보정된 리스트의 프레임, 좌표, 이동 여부를 출력하는 디버깅용 메소드 | |

## 

| Binfo Class | | | |
| --- | --- | --- | --- |
| 공의 좌표 정보를 저장한 클래스 | | | |
| 멤버 변수 | | | |
| 변수명 | 타입 | 설명 | |
| frame | int | 프레임 번호 | |
| x | float | x좌표 | |
| y | float | y좌표 | |
| move | bool | 해당 프레임에 공이 움직이고 있었는지 여부. 움직였다면 참, 움직이지 않았다면 거짓 | |

## 2. bagreader.py

**기능**

1. detector에서 전달 받은 ball\_bags에서 정보를 읽어와 큐볼, 제 1목적구, 제 2목적구로 구분하여 저장

2. detector에서 전달 받은 edges에서 정보를 읽어와 당구대 모서리의 좌표에서 각 쿠션에 해당하는 두 모서리의 좌표들로 저장

3. Ball클래스에서 각각의 공을 id에 맞게 분류한 후 find\_cue()를 통해 큐볼과 목적구로 구분

4. 우상단, 좌상단, 좌하단, 우하단의 모서리 좌표를 [(우상단, 좌상단), (좌상단, 좌하단), (좌하단, 우하단), (우하단, 우상단)]으로 이루어진 cshline 리스트로 변환

**입력**

| 함수명 | 타입 | 설명 | |
| --- | --- | --- | --- |
| ball\_bags | 리스트 | Detect 모듈로부터 전달 받은 ball\_bag 리스트들이 저장된 리스트   * ball\_bags는 ball\_bag 리스트들이 저장된 리스트 * ball\_bag 리스트는 ball 딕셔너리가 저장된 리스트 * ball 딕셔너리는 frame, x, y, w, h, id, conf 값을 키값으로 가지는 공의 정보가 저장된 딕셔너리 | |
| edges | 딕셔너리 | FindEdgePipe로부터 전달받은 Edge값이 저장된 딕셔너리 | |

**출력**

| 변수명 | 타입 | 설명 | |
| --- | --- | --- | --- |
| cue | Ball Class | 큐 볼에 대한 Ball Class | |
| tar1 | Ball Class | 제1적구에 대한 Ball Class | |
| tar2 | Ball Class | 제2적구에 대한 Ball Class | |
| cshline | 리스트 | 우상단, 좌상단, 좌하단, 우하단의 모서리 좌표를 [(우상단, 좌상단), (좌상단, 좌하단), (좌하단, 우하단), (우하단, 우상단)]으로 변환한 것 | |

## 3. balltracker.py

**기능**

tracking 관련 연산 함수를 정의한 파일이다.

**내부함수**

| 변수명 | | 설명 | |
| --- | --- | --- | --- |
| get\_vector(b\_curr, b\_prev) | | Binfo 객체를 입력받아 방향 벡터를 반환 | |
| norm(vector) | | 벡터를 입력받아 단위 벡터를 반환 | |
| get\_degree(vector\_in, vector\_out) | | 두 벡터가 이루는 사이 각을 반환 | |
| find\_ini(ball\_list) | | ball\_list에서 공이 최초로 움직이기 시작하는 인덱스를 반환 | |
| point\_dist(obj1, obj2) | | Binfo 객체를 입력받아 두 객체 사이의 거리를 반환 | |
| get\_dist(p1, p2) | | 두 좌표를 입력받아 좌표 사이의 거리를 반환 | |
| line\_dist(b, line) | | Binfo 객체 b와 선분을 이루는 두 좌표 line을 입력받아 b에서 line까지의 최단거리를 반환 | |
| line\_dist\_point(b, line) | | 좌표 b와 선분을 이루는 두 좌표 line를 입력받아 b에서 line까지의 최단거리를 반환 | |
| find\_cue(b1, b2, b3) | | Ball 객체 세 개를 입력받아 최초로 움직인 프레임을 비교하여 큐볼, 제 1목적구, 제 2목적구 순으로 정렬하여 반환 | |
| is\_cue(b1, b2) | | * b1이 b2보다 최초로 움직인 프레임이 빠른 경우 참을 반환 * 최초로 움직인 프레임이 같은 경우 두 공의 중심을 이은 방향으로 튕겨나간 공을 목적구로 판단 | |
| make\_line(ball) | | * 공의 좌표들을 통해 전체 이동경로를 그린 ball\_line을 반환 * 공의 좌표들을 event\_frame을 기준으로 구간을 나누고 각 구간별로 make\_curve로 곡선을 이루는 연속된 점의 좌표들을 구하여 전체 이동경로에 대한 경로를 구함 | |

## 4. curve.py

**기능**

이동 경로가 표시된 점들을 하나의 선으로 바꾸기 위한 함수들을 정의하는 파일이다.

**내부함수**

| 함수명 | | 설명 | |
| --- | --- | --- | --- |
| make\_curve(ball\_list) | | * 충돌 지점을 기준으로 나뉘어진 구간을 입력으로 받고 해당 구간의 좌표들을 근접하게 지나는 연속된 점들의 집합을 반환 * 구간의 시작점에서 끝점 방향으로의 방향벡터를 계산 * T\_R\_origin()을 통해 시작점이 원점에 위치하도록 구간 전체를 이동시키고 방향 벡터가 x축의 양의 방향과 일치하도록 회전된 좌표의 리스트를 받음 * 변환된 좌표에서 curve\_fit2() 함수로 2차함수를 근사하고 구간 내의 각 x좌표에 대해서 2차함수로 근사된 y좌표를 구해 리스트로 저장 * 곡선을 이루는 점들의 좌표를 R\_T\_reverse() 함수를 통해 다시 원래의 위치로 회전 및 이동시킨 좌표로 변환하여 반환 * 곡선의 저장된 좌표들은 x좌표 값에 따라 오름차순으로 저장되며 구간에서의 공이 이동한 순서와는 일치하지 않음 | |
| curve\_fit(xs, ys) | | 동일한 인덱스끼리 매칭되는 x 좌표 리스트와 y 좌표 리스트를 입력으로 받고 해당 좌표들을 근접하게 지나는 2차 함수를 근사하여 계수들을 리턴 | |
| curve\_fit2(xs, ys) | | * curve\_fit() 함수에서 x, y 좌표 리스트를 입력으로 받아 근사한 2차함수는 시작점과 끝점을 지나지 않을 수 있기 때문에 양 끝점과 구간 안의 특정한 점을 잡아 세 점을 지나는 2차함수를 다시 근사 * 구간 안의 특정한 점은 2차함수 곡선에서 시작점과 끝점에서의 기울기의 평균에 해당하는 기울기를 가지는 지점 | |
| calc\_y(x, coefficients) | | x좌표와 근사로 얻은 2차함수의 계수를 입력으로 받아 x 좌표에 해당하는 y 좌표를 반환 | |
| calc\_y\_prime(x, coefficients) | | x 좌표와 근사로 얻은 2차함수의 계수를 입력으로 받아 x 좌표에 해당하는 2차함수 위의 점에서의 기울기를 반환 | |
| get\_middle\_point(xs, coefficients) | | 구간의 x 좌표들과 근사로 얻은 2차함수의 계수를 입력으로 받아 시작점과 끝점에서의 기울기를 계산하고 2차함수 위에서 그 기울기들의 평균에 해당하는 기울기를 가지는 점의 x좌표를 반환 | |
| T\_R\_origin(ball\_list, start, theta) | | 구간의 좌표들을 시작점이 원점에 위치하고 시작점에서 끝점으로의 방향 벡터가 x좌표의 양의 방향과 일치하도록 이동 후 회전 시킨 좌표들로 변환 | |
| R\_T\_reverse(ball\_list, start, theta) | | T\_R\_origin에서의 과정을 역순으로 행하여 구간의 좌표들을 다시 원래의 위치로 변환 | |

## 

## 5. cvtCoord.py

**기능**

1. 영상을 당구대 나사 안쪽 부분만 보이도록 변환한다
2. 모든 좌표들을 800x400으로 정규화 한다

**내부함수**

| 함수명 | | 설명 | |
| --- | --- | --- | --- |
| make\_pers(cshline) | | * cshline을 입력받아 cshline 좌표 내부의 영역을 800x400 크기의 영역으로 변환하는 변환행렬을 계산하여 반환 * cshline에서 네 모서리이 좌표를 얻고 모서리 내부의 영역을 cv2.getPerspectiveTransform()을 이용해 800x400 크기로 변환하는 변환행렬을 계산 | |
| cvtCoord(x, y, pers) | | x, y 좌표를 입력받아 800x400 크기로 변환한 영역에서 해당 좌표에 대응하는 변환된 좌표를 반환 | |
| cvtball(ball, pers) | | Ball 클래스를 입력받아 modified\_ball\_list의 좌표들을 모두 800x400으로 변환된 영역의 좌표로 변환 | |
| cvtline(line, pers) | | make\_curve()를 통해 만들어진 곡선의 리스트를 800x400으로 변환된 영역의 좌표로 변환 | |
| cvtimgs(imgs, pers) | | 이미지에서 당구대 내부에 해당하는 영역만 분리하여 800x400 크기로 변환 | |
| cvtcsh(cshList, pers) | | 쿠션의 좌표들을 800x400으로 변환된 영역의 좌표로 변환 | |

## 6. pointmodify.py

**기능**

각 점들을 보정하기 위한 함수들을 정의하는 파일이다.

**내부함수**

| 함수명 | | 설명 | |
| --- | --- | --- | --- |
| modify\_soft(ball\_list) | | * detector에서 전달받은 좌표들 중 공을 검출하지 못한 프레임에서의 좌표를 추정하여 생성 * find\_collision\_soft()를 통해 충돌이 발생할 가능성이 있는 모든 지점을 구함 * find\_collision\_soft()에서 구한 지점들을 기준으로 구간을 나누고 각 구간에서 프레임 번호를 비교해 공이 검출되지 않은 프레임이 있는지 검사 * modify\_miss()를 통해 검출하지 못한 프레임에서의 좌표들을 생성 * 각각의 구간들을 합쳐 누락이 보정된 전체 구간을 반환 | |
| modify\_miss(fixed, index, miss\_len) | | * 구간의 좌표와 구간에서 누락이 발생한 지점의 인덱스 번호, 누락된 프레임의 개수를 입력으로 받고 누락된 프레임에서의 좌표를 추정하여 생성 * start\_point는 누락이 발생하기 직전 프레임의 좌표 * end\_point는 누락이 발생한 직후 프레임의 좌표 * curve\_fit2() 함수로 구간의 경로를 2차함수로 근사 * start\_point의 x좌표와 end\_point의 x좌표 사이에서 누락된 프레임의 개수만큼 n개의 x 좌표들을 구하고 2차함수에 대입해 누락된 프레임에서의 x, y 좌표를 생성 * n개의 x좌표는 start\_point의 x좌표에서부터 start\_point 직전 프레임에서 start\_pont으로의 x의 변화량만큼 일정하게 x를 변화시킨 지점들 * 새로 생성된 좌표들에서 이동 중인지 여부를 판단하여 move 값 지정     보정 전    보정 후   * 좌표의 누락 프레임 전후로 공이 이동상태에서 정지상태로 변한 경우 정지상태의 좌표를 넘어가지 않도록 제한하였음 * 좌표의 누락 프레임 전후로 공이 정지상태에서 이동상태로 변한 경우 end\_point직후 프레임에서 end\_point으로의 x의 변화량만큼 x를 일정하게 변화시켜 역순으로 좌표를 생성하고 마찬가지로 정지상태의 좌표를 넘어가지 않도록 제한하였음 | |
| modify\_hard(cue, tar1, tar2, event\_list, cshline, ball\_radius) | | * event\_predict()를 통해 추정한 충돌 지점들을 실제 충돌 지점에 더 가깝도록 새로운 충돌 지점을 생성하여 갱신 * 충돌이 발생한 프레임을 기준으로 구간을 나누기 때문에 모든 구간을 검사하도록 각 Ball 객체들의 event\_frame에 공이 검출된 마지막 프레임을 추가 * 각각의 공에 대해서 쿠션과 충돌한 모든 지점들에 대해 modify\_csh\_point()를 통하여 충돌 좌표를 보정 * 큐볼과 목적구가 최초로 충돌하는 지점에 대해 modify\_hit\_point()를 통해 충돌 좌표를 보정 | |
| modify\_csh\_point(ball, event\_list, hit\_frame, cshline, ball\_radius) | | * 공과 쿠션이 충돌하는 지점을 실제 충돌 지점에 가깝도록 보정 * 쿠션과 가장 가까웠던 프레임을 기준으로 전후 두 프레임 씩 총 5프레임 동안 해당 쿠션 이외의 객체와 충돌이 있었는지 검사 * 충돌이 없었다면   + 5프레임의 좌표들 중 연속된 4 프레임의 좌표를 고르고 getCross()를 통해 보정된 좌표를 구하고 나머지 한 쌍의 연속된 4 프레임의 좌표들로도 getCross()를 통해 보정된 좌표를 구함   + 각 보정된 좌표에 대해서 충돌한 쿠션이 위나 아래쪽 쿠션이라면 보정 지점의 x 좌표가 오차 범위 이내인지, 왼쪽이나 오른쪽 쿠션이라면 보정 지점의 y 좌표가 오차범위 이내인지 검사   + 보정된 좌표가 기존의 좌표들로부터 떨어진 거리에 비례해 충돌한 프레임을 추정하고 보정된 좌표로 충돌 정보를 갱신          * 충돌이 있었다면   + 충돌이 없었던 연속된 두 좌표를 지나는 직선을 통해 들어오거나 나가는 방향을 파악   + 해당 방향의 직선에서 쿠션과의 거리가 ball\_radius에 해당하는 지점의 좌표를 보정된 좌표로 충돌 정보를 갱신   + 충돌이 없었던 연속된 두 좌표는 쿠션과 가장 가까웠던 프레임을 포함하지 않는 두 좌표와 들어오는 방향의 두 좌표를 우선함 | |
| modify\_hit\_point(cue, tar, event\_list, cshline) | | * 큐볼과 목적구가 충돌하는 지점을 실제 충돌 지점에 가깝도록 보정 * 큐볼이 목적구와 충돌하기 전 목적구를 향해 움직인 방향의 직선과 목적구가 튕겨나간 방향의 직선을 구하고 두 직선의 교점을 구함 * 보정된 충돌 지점이 오차범위 내에 존재하는지 검사하고 기존 충돌 지점보다 더 가까우면 쿠션 충돌 정보를 갱신 * 보정된 충돌 지점 전후로 쿠션과의 충돌이 있을 경우 쿠션 충돌 정보를 갱신        * 큐볼이 움직인 직후 목적구와 충돌한 경우 큐볼이 목적구와 맞고 튕긴 방향의 직선과 목적구가 튕긴 방향의 직선의 교점으로 보정하였음 | |

## 7. predict.py

**기능**

충돌 지점을 추론하는 함수들을 정의한 파일이다

**내부함수**

| 함수명 | | 설명 | |
| --- | --- | --- | --- |
| find\_collision\_soft(fixed\_ball\_list) | | 공이 움직이기 시작한 시점부터 collision\_check\_soft()를 통해 충돌할 가능성이 있는 지점으로 판단되는 지점들을 col\_list에 저장하여 반환 | |
| collision\_check\_soft(b\_post, b\_curr, b\_prev) | | 이전 프레임과 비교하여 진행방향이 15’ 이상 바뀐 경우 충돌할 가능성이 있는 것으로 판단하여 True를 반환 | |
| event\_predict(cue, tar1, tar2, cshline, ball\_margin, csh\_margin) | | * modify\_soft()를 통해 누락된 프레임에 대해 보정을 거친 공의 좌표들에서 충돌이 있었던 프레임의 충돌 정보를 모아 반환 * 충돌은 큐볼이 움직인 이후부터 발생하므로 last\_event\_idx를 큐볼이 움직인 시점으로 초기화 * nearest\_csh()와 nearest\_b()를 통해 last\_event\_idx 이후 각 유형의 충돌에 대해 발생할 수 있는 가장 빠른 충돌 정보들을 모아 possible\_event\_list에 저장 * possible\_event\_list를 충돌이 발생한 프레임 기준 오름차순으로 정렬 * last\_event\_idx 이후로 가장 먼저 충돌이 발생하는 프레임으로 last\_event\_idx를 갱신하고 충돌 정보를 event\_list에 저장 * 해당 프레임에 여러 번의 충돌이 있었다면 그 충돌 정보들을 모두 저장 * event\_list에 저장한 충돌 정보에 따라 각 공 객체들의 last\_event\_obj와 last\_csh\_dist를 갱신 * 다시 nearest\_csh()와 nearest\_b()로 예상 충돌 정보들을 얻고 가장 빠른 충돌 정보들만 저장하는 과정을 반복 | |
| nearest\_csh(ball, cshline, last\_event\_idx, csh\_marign) | | * 공이 last\_event\_idx 이후로 가장 먼저 쿠션과 충돌하는 프레임의 충돌 정보를 반환 * prev\_moved 리스트에는 순서대로 이전 프레임에서 위쪽 / 왼쪽 / 아래쪽 / 오른쪽 쿠션까지의 거리변화가 저장됨 * last\_event\_idx 이후의 좌표들을 하나씩 확인하면서 각 쿠션에 대해 이전 프레임과의 거리 변화를 비교 * 공이 쿠션과 margin 이내의 거리에서 가까워지다가 멀어지는 지점이 발생하면 해당 지점을 충돌 지점으로 판단하여 충돌 정보를 반환 * 공이 마지막으로 충돌한 객체가 동일한 방향의 쿠션인 경우 마지막으로 충돌한 쿠션과의 거리를 비교하여 더 가까웠던 충돌 지점을 저장하도록 예외처리하였음 * margin은 csh\_margin을 최소값으로 하고 쿠션에 가까워지는 정도에 따라 늘어남 | |
| nearest\_b(cue, tar, last\_event\_idx, ball\_margin) | | * 큐볼이 last\_event\_idx 이후로 가장 먼저 목적구와 충돌하는 프레임의 충돌 정보를 반환 * prev\_moved는 이전 프레임에서 목적구까지의 거리변화가 저장됨 * last\_event\_idx 이후의 좌표들을 하나씩 확인하면서 큐볼과 목적구의 거리 변화를 비교 * 큐볼과 목적구가 margin 이내의 거리에서 가까워지다가 멀어지는 지점이 발생하면 해당 지점을 충돌 지점으로 판단하여 충돌 정보를 반환 * 목적구가 정지상태인 경우 거리 변화에 상관없이 목적구가 움직이기 시작할 때의 큐볼의 위치를 반환 * 큐볼과 목적구 중 정지상태의 공이 있다면 이동 중인 공의 현재 프레임의 좌표와 이전 프레임의 좌표를 이은 선분에서 정지한 공까지의 최단거리로 거리를 비교함 * 목적구가 정지한 상태에서는 margin을 1.4배 크게 계산함 | |

## 8. ballpredictor.py

**기능**

Tracking 모듈의 메인함수 역할을 하는 파일이다.

**설명**

* detector에서 검출한 공과 모서리, 영상의 가로 길이, 영상 이미지를 입력을 받고 공의 충돌정보와 공의 이동경로가 그려진 영상 이미지를 반환
* ball\_margin은 공끼리 충돌할 때의 margin, csh\_margin은 공과 쿠션이 충돌할 때의 margin, ball\_radius는 쿠션 충돌 지점을 보정하여 생성될 좌표가 쿠션으로부터 떨어진 거리로 공의 반지름을 의미
* bag\_reader()를 통해 큐볼과 목적구를 구분하고 모서리의 좌표를 각 쿠션을 이루는 두 좌표쌍으로 변환
* modify\_soft()를 통해 각 공에 대해서 검출이 안된 프레임의 공의 좌표를 생성
* event\_predict()에서 충돌이 발생한 지점을
* modify\_hard()에서 충돌이 발생한 지점을 실제 충돌 지점에 가깝도록 보정
* make\_line()으로 공의 전체 경로를 선으로 형성
* 영상 이미지와 좌표들을 당구대 내부 영역만 분리하여 800x400 크기로 변환
* 변환된 이미지에 쿠션의 범위와 각 공의 이동경로를 그림
* event\_list에서 출력 형식에 맞는 문자열로 변환
* 충돌 정보를 저장한 event\_str과 각 공의 이동 경로가 그려진 이미지 draw\_imgs를 반환