Carom Helper
설계 문서

22.10.24 - 22.12.09

소속 | 에스프레소미디어
작성 | 이탁균

목차

I. Ball Detect 모듈

1. 소개

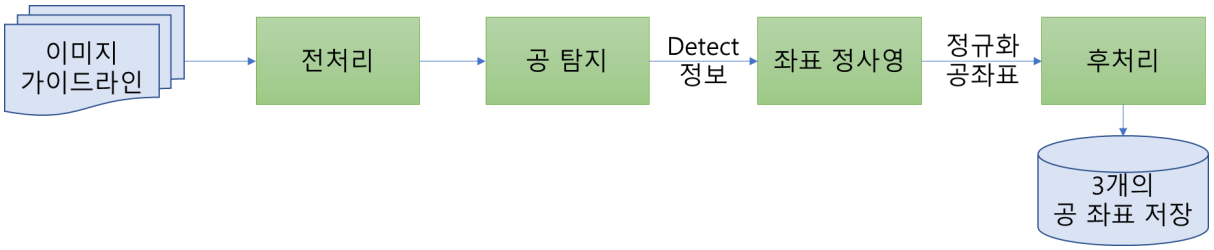
사용자에게 당구대 이미지와 가이드 라인(당구대의 네 모서리)
당구대(나사 안쪽 영역 네 개의 모서리)와 공들(당구대 안쪽 세 개의 공)을 찾아 내기 위한 모듈. 영상의 매 프레임 마다 공들의 위치들의 찾고, 당구대 모서리의 위치를 찾아 낸다.

2. 설계

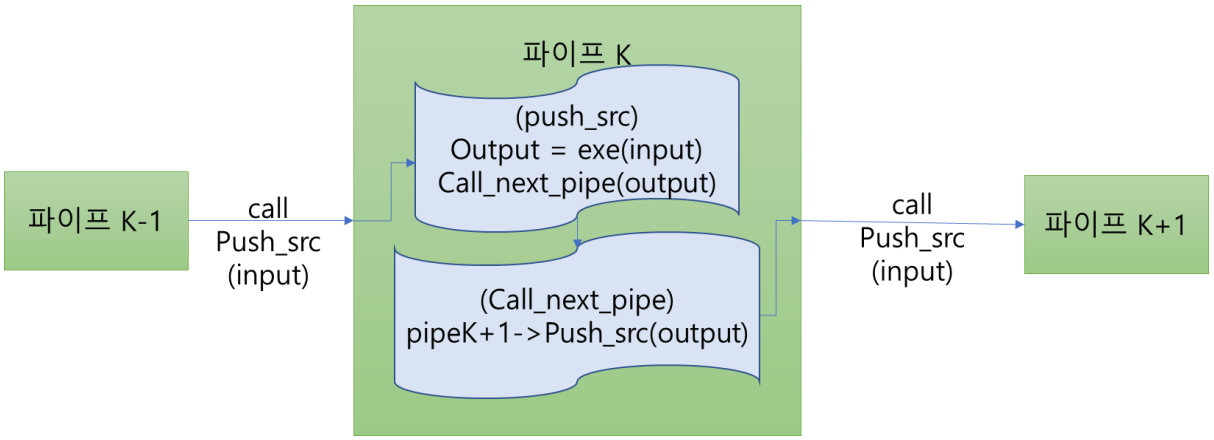
구조 설계

파이프라인 패턴, 옵저버 패턴을 모티브로 구조를 설계했다. 각 기능을 파이프로 제작하여 특정 기능의 변경이나 재사용이 필요할 때 용이하도록 했고 파이프의 입출력을 자동화하기 위해 옵저버 패턴을 추가로 사용했다.
PipeResouce 라는 리소스가 파이프를 지날 때마다 변화가 가해지는 방식으로 파이프라인이 모두 지나면, 지나온 모든 파이프의 연산이 적용 되도록 디자인 했다.

파이프라인



옵저버를 통한 파이프 입출력 자동화



파이프 연결

파이프와 파이프를 연결하기 위해서는 connect_pipe라는 명령어를 통해서 연결해야 한다. 현재 파이프의 다음 파이프가 무엇인지를 설정하고 싶다면,

onePipe.connect_pipe(twoPipe)

와 같은 식으로 파이프를 연결해야 한다. split 계열의 파이프는 뒷 단에 여러개의 파이프가 연결될 수 있는데, 하나씩 파이프를 연결하면, 자동적으로 다른 인덱스가 주어져서 이어진다.

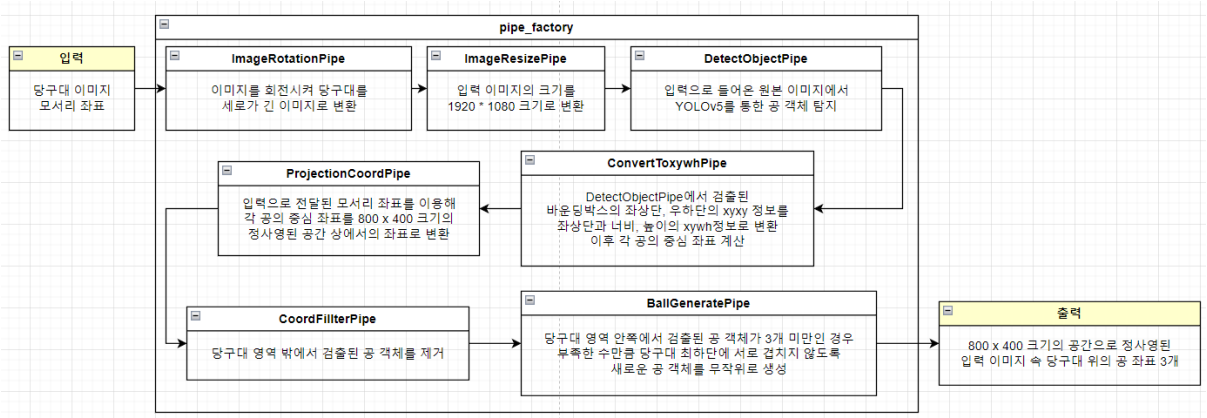
```
for i in range(split_idx_pipe.idx2label.__len__()):
    bag = ResourceBag()
    split_idx_pipe.connect_pipe(bag)
```

파이프 팩토리

파이프의 연결상태를 관리하는 생성자 함수이다. 밑에서 Flow 차트로 설명되는 파이프 설계도에서 나타내는 순서대로 파이프를 연결해 놓았다. 파이프를 생성하고, 해당 파이프를 순서대로 연결 한 뒤에 시작 파이프와 말단 bag을 반환한다.

시작 파이프(DetectObject)에 push_src를 진행하면, 말단 bag에 순차적으로 값이 쌓일 것이다.

Flow 설계

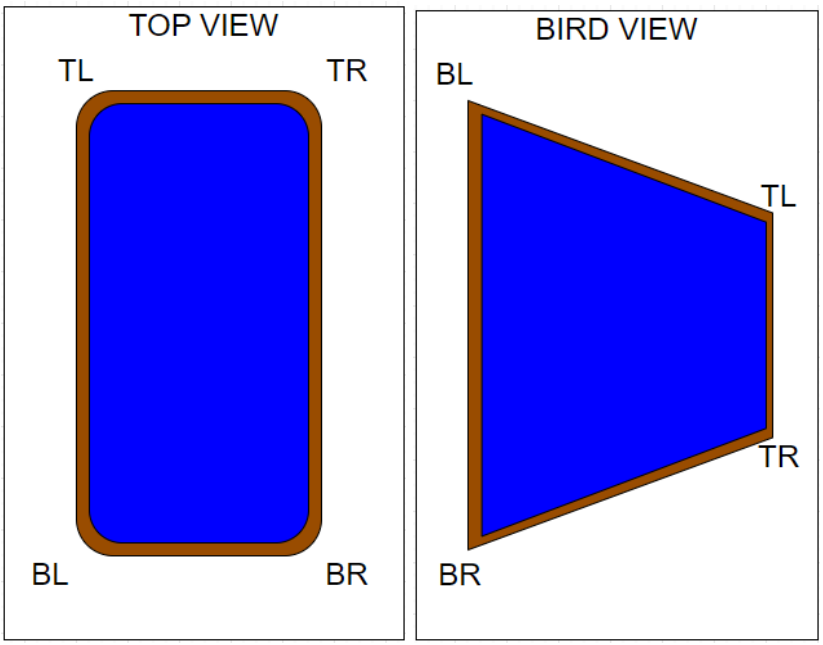


파이프 설계

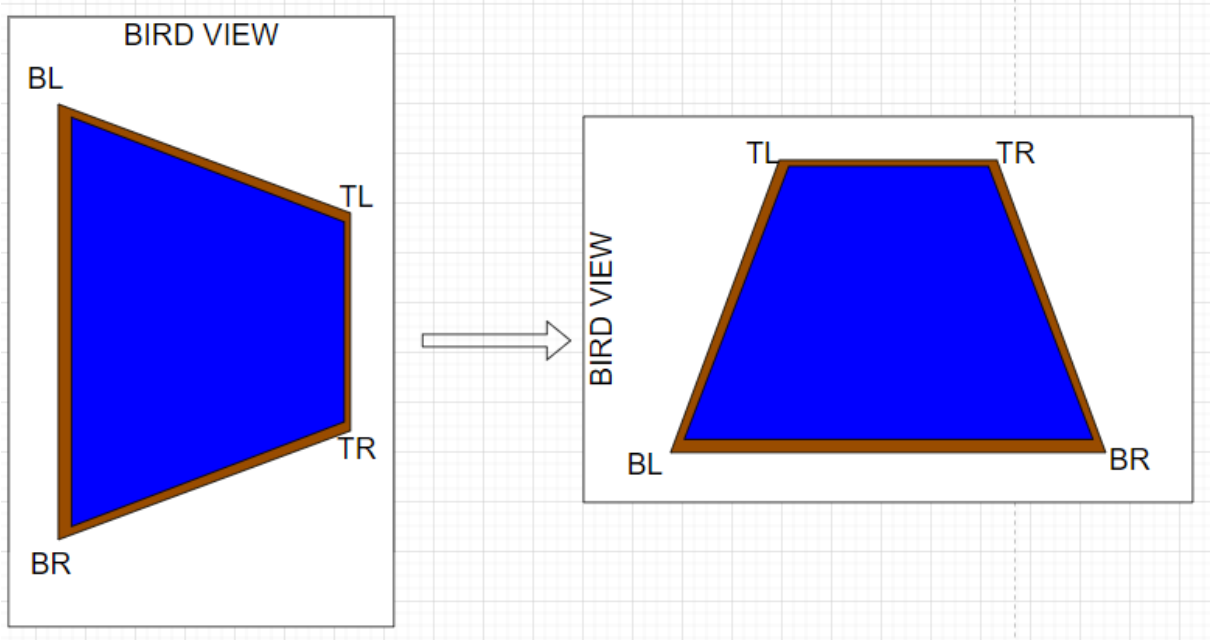
ImageRotationPipe

기능	사용자가 제공한 테이블 모서리 좌표[TopLeft, TopRight, BottomLeft, BottomRight]의 좌표들이 당구대의 위치와 동일한 모양이 될 수 있도록 회전 시켜주는 파이프이다.
이유	탑뷰는 TL,TR,BL,BR의 좌표들이 세로로 잘 배치된 반면에 버드뷰는 가로로 기울여야지 당구대 모양과 매칭된다. 탑뷰, 버드뷰 등등의 모든 뷰를 같은 모양으로 처리해 주기 위한 전처리가 필요했기 때문에 이렇게 진행했다. 모든 이미지는 세로가 길도록 프론트에서 넘어온다. 버드뷰는 카메라를 가로로 눕혀서 세로 방향을 찍는다.

파이프 라인에 입력으로 들어올 사진 형태



ImageRotationPipe에서 작업 이후 버드뷰 이미지의 형태



ImageResizePipe

기능	이미지 사이즈를 특정 크기로 바꾸어 준다. 이미지가 회전했는지 아닌지를 판단해서 너비와 높이를 적당하게 리사이즈 해준다.
이유	사용자 휴대폰 기종에 따라 이미지의 크기가 다르게 들어온다. 하지만 프론트 개발자는 1080*1920 기준의 가이드 라인 절대 좌표로 전달해 주었다. 입력이미지와 가이드라인의 크기(기준)을 통일하기 위해 해당 파이프에서 이미지 사이즈를 조정해 가이드 라인과 크기(기준)을 같게 해준다.

DetectObjectPipe

기능	1. yolov5를 통해 공과 모서리의 xyxy 좌표를 찾는다. 2. 탐지된 객체를 PipeResource 의 dets 변수에 저장한다. 3. SplitCls 를 이용해 공과 모서리를 분리하고 공만 다음 파이프로 전달한다. (모서리는 폐기)
동작	yolo 알고리즘을 활용하여 동작한다. 특별히 Furiosa의 NPU를 사용한다. 그럼으로 Furiosa NPU의 맞는 YOLOv5를 사용한다.

ConvertToxywhPipe

기능	xyxy(top,left / bottom,right)기반의 box 좌표를 xywh(center-x, center-y, width, height)로 변경해 준다.
이유	DetectOjbectPipe의 xywh의 값이 xyxy이지만, 이후 사용되는 값은 공의 중심 값만 사용하기 때문이다.

ProjectionCoordPipe

기능	dets에 담겨있는 1080*1920 기반의 공 좌표(x,y)를 가이드라인을 사용해서 800*400 크기의 정사영된 직사각형 공간으로 이동시킨다.
이유	당구공의 위치가 당구대 상의 좌표로 존재해야 의미가 있다. 입력받은 가이드 라인을 사용해서 좌표를 정사영하면, 정규화 된 공 좌표를 얻을 수 있다. 기울어진 당구대 사진은 모서리를 알수 없기 때문에 사용자가 전달해준 가이드라인을 신뢰한다.

CoordFilterPipe

기능	당구대 영역 밖에서 검출된 공 객체를 제거
이유	당구대 밖에 당구공이 발견되면, 경로를 찾을 때 큰 오류를 발생 시킨다. 해당 오류를 잡기 위해 제거 작업을 진행한다.

BallGeneratePipe

기능	공 개수가 무조건 3개가 될 수 있도록 해준다. 부족하면 생성하고, 많다면 앞에 3개만 결과로 반환한다.
이유	실수로 찾지 못한 공을 아예 못 찾은 것으로 남기기 보다는 사용자가 수정할 수 있도록 임의의 공좌표를 넘겨준다. yolo_ball 학습시 탑뷰 기반으로 학습하여, 큰 공을 찾지 못하는 경우가 많은 것을 발견했다. 하여 버드 뷰일 때, 못 발견했다고 가정하고, 가장 하단에 랜덤으로 생성 하도도록 했다.

3. 예외 처리

DetectObjectPipe

예외	처리
인식하지 못하는 당구공이 있다. -> 당구공의 개수가 3개 미만이다.	1. 버드뷰에서 하단에 붙어 크기가 큰 공을 잘 찾지 못하는 경우가 많이 발생했다. 못 찾는 경우는 모두 하단에 있는 경우라고 가정하고, 공이 겹치지 않도록 생성했다.
당구대 밖에 당구공을 찾았다. -> 당구공의 개수가 3개보다 많다.	1. 정사영을 하고 나면 테이블 밖에 공은 (0,0)~(800,400)의 범위를 나가게 되는데, CoordFilterPipe를 통하여 이러한 경우를 제거 할 수 있다. 2. 모종의 이유로 당구대 내부에 3개의 공을 인식할 수 있는데, 그러한 경우는 CorrdFilterPipe를 통해서 적당한 좌표 3개만 반환 할 수 있도록 한다.

PojectionCoordPipe

예외	처리
가이드라인(TL,TR,BL,BR)이 잘못 들어온 경우	1. 그런 경우는 없다고 가정했다. 2. ImageRotationPipe에서 TL,TR,BL,BR이 제대로 배치되어 있지 않다면, 회전시켜서 제대로 배치시킨다. 하지만 이때 재대로 배배치되어 있지 않다면, 에러를 발생시키도록 코딩을 해 놓았다.
기울어지게 찍힌 사진이 들어올 경우, 전체적으로 공의 좌표 중 y좌표가 작게 나오는 현상 -> 이미지는 측면에서 보는데, 측면에서의 중심 좌표를 그대로 사용하기 때문에 물리적인 공의 위치와는 조금 차이가 있다.	1. 해당 에러는 처리하지 않았다. 2. 프론트 엔드에서 사용자가 직접 수정하도록 하여 해당 문제를 넘기기로 했다.

II. Route Recommend 모듈

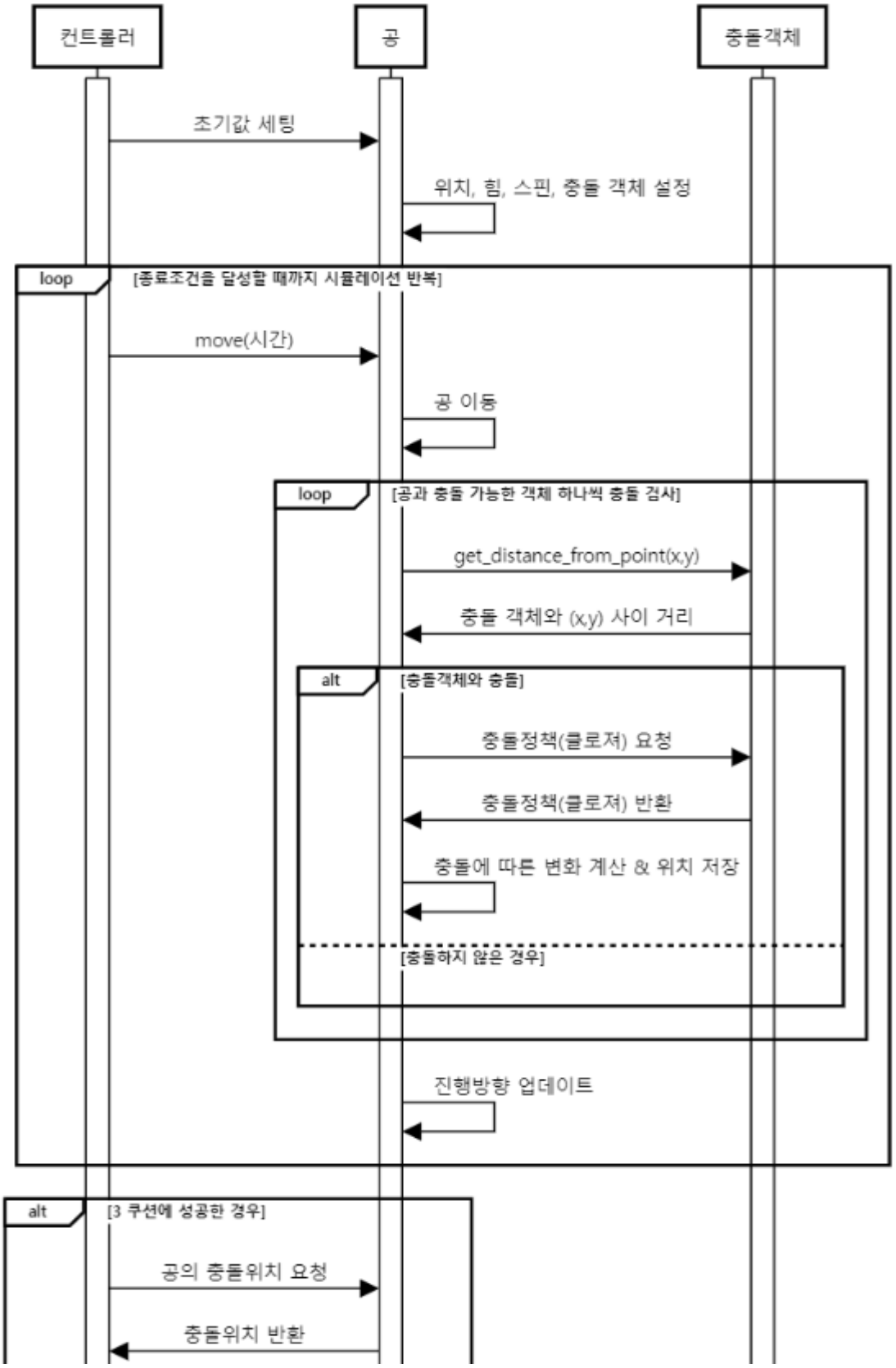
1. 소개

공의 배치와 함께 추천 경로를 요청하면, 추천경로 최대 3개를 반환해 주는 모듈이다.
추천 경로를 반환하기 위해 시뮬레이션 이라는 알고리즘을 사용했다.(하여 Route Recommend 모듈을 시뮬레이션 모듈이라고도 부르겠다.)

시나리오

1. 입력 : 공 배치
2. 초기값 세팅 설정 [두께, 시간, 팁, 힘]
3. [두께, 시간, 팁, 힘]을 [공 방향, 공 상하스핀, 공 좌우스핀, 힘(속력)]로 변환
4. 시뮬레이션 진행
    1) ball.move(time)을 호출하여 공들을 조금 이동시킴
       * time은 공들이 조금 움직일 수 있을 만큼의 시간을 계산해서 넣어준다.
    2) 벽과 다른 공들과 충돌 판정
    3) 충돌로 판정 될시 충돌함수 실행 & 충돌지점 저장
       * 충돌함수는 방향벡터와 충돌객체의 접선을 고려하여 반사벡터를 구한다.
       반사벡터는 공의 방향벡터가 되고, 힘과 스핀이 변형된다.
    4) 진행방향 업데이트
    5) 종료조건 확인 후 조건 만족 시 시뮬레이션 종료 그렇지 않으면, 1)으로 이동
       * 종료조건
       - 모든 공의 속도가 거의 멈춘 것처럼 느릴 때
       - 성공 했을 때(벽에 3쿠션, 1/2적구 충돌)
       - 키스가 날때(1적구와의 두번째 충돌, 1적구와 2적구의 충돌)
5. 종료조건 달성시, 성공/실패 판정
6. 성공시 초기값 [두께, 시간, 팁, 힘]과 공의 충돌지점 반환

흐름도 (특정 공의 입장에서 서술)



2. 설계

위와 같은 개념으로 설계를 하였으나, 충돌정책의 변동성을 고려하지 못하고 설계를 했다. 그렇기 때문에 형태가 좋지 않다. 하여 시뮬레이션을 하는 정책들의 변동에 대해서 강한 형태로 다시 설계하였다. 하지만 코드는 작성하지 못했다. 해당 설계 구조도 simulate_design.pdf로 남겨두겠다.
기존 설계의 내용과 실제 구현의 내용이 맞지 않아서 함수, 변수의 이름도 읽기 힘들것으로 추정합니다. 가능하다면, 제가 다시 구성한 구조도를 토대로 다시 시작하시기를 권장드립니다.