

쉬운설명_프레임기반에서_측점기반_플레이어

시간이 아니라 '측점'으로 움직이는 영상 플레이어 (그림 설명서)

이 글은 초등학생도 이해할 수 있게 그림과 이야기로 쓴 설명서예요. "보통 영상 플레이어는 시간(프레임)으로 움직이는데, 우리는 어떻게 **측점(선로 위치)**으로 움직이게 바꿨을까?" 그 비밀을 하나씩 풀어 줄게요. (진짜 코드는 [chainage.ts](#) · [StationBar.tsx](#) 참고)

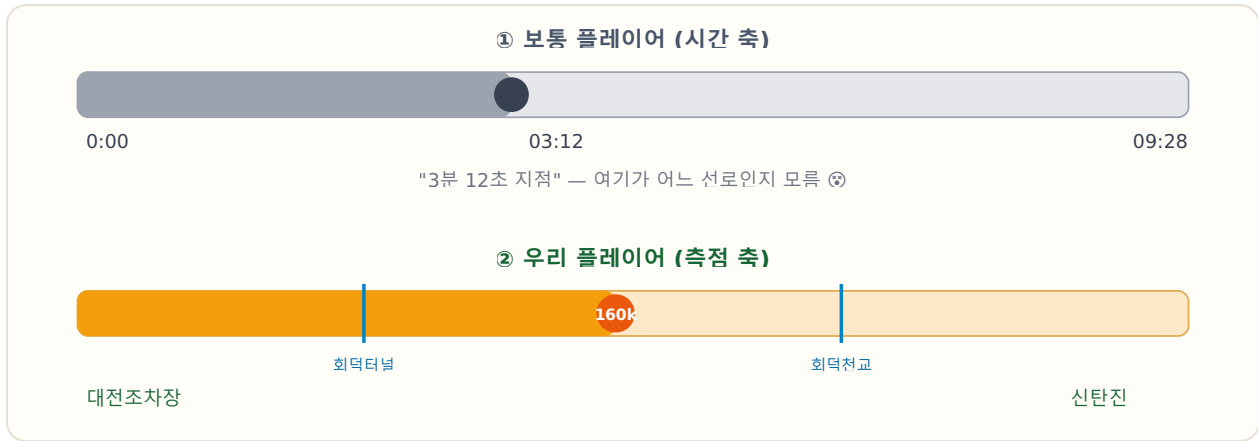
0. 한 문장으로 말하면

"영상의 매 장면마다 드론 GPS로 '지금 몇 측점인지'를 계산해 두어서, 시간(몇 분 몇 초)이 아니라 **측점(160k130 같은 선로 위치)**으로 영상을 찾아보게 만든 기술" 이에요.

비유: 보통 영화는 "32분 10초로 가줘"라고 하죠. 우리 플레이어는 "**회덕터널 앞으로 가줘**"처럼 **장소**로 찾아가요.

1. 보통 플레이어 vs 우리 플레이어

보통 영상 플레이어의 아래 막대는 **시간**(0:00 ~ 끝)이에요. 그런데 철도 점검하는 분들은 "몇 분"이 아니라 "**몇 측점**"으로 이야기해요. 그래서 막대를 **측점 기준**으로 바꿨어요.



시간 축은 "몇 분"만 알려준다. 측점 축은 "지금 어느 선로 위치(측점)", "무슨 구조물 근처"인지 한눈에 보인다.

2. '측점'이 뭐예요?

철도에서 **출발점부터 몇 미터 왔는지**를 나타내는 '거리 이정표'예요. 도로의 **km 표지판**과 똑같아요!

- 160k130 = 출발점에서 **160km + 130m** 지점
- 숫자가 커지면 앞으로 간 것, 작아지면 되돌아온 것

3. 핵심 비밀 — 매 장면마다 '시간'과 '측점'을 짝지어 둔다

비밀은 **드론 GPS**예요. 드론 영상의 **매 프레임(사진 한 장)**에는 그때 드론의 **위도·경도(GPS)**가 기록돼 있어요. 이걸 이용해 프레임마다 두 가지를 계산해 **짝지어** 뒀어요.

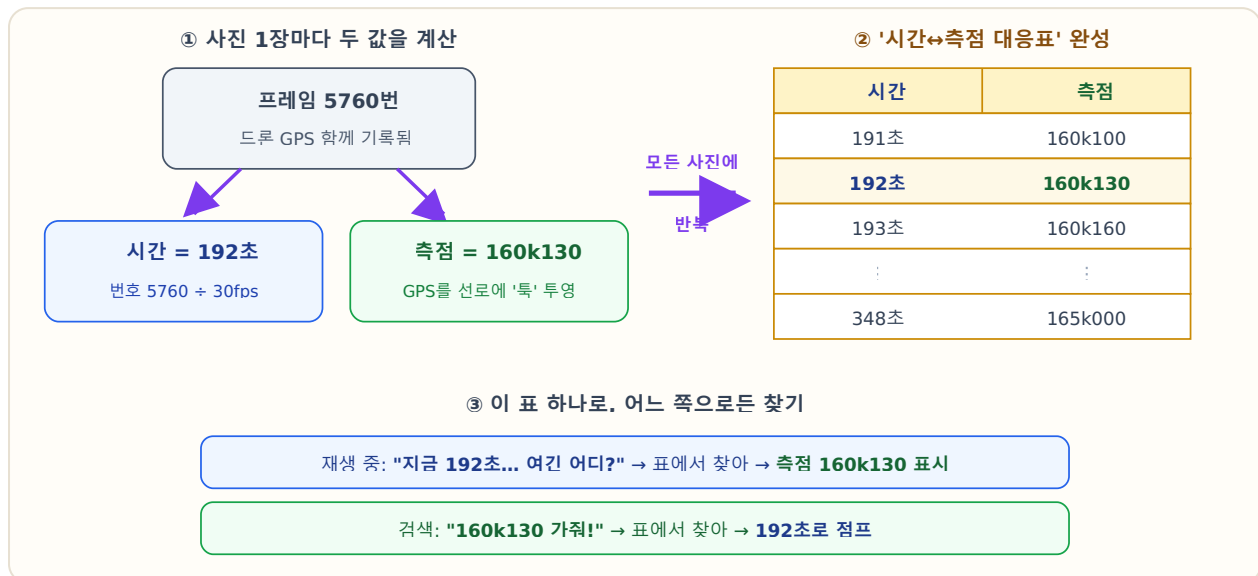
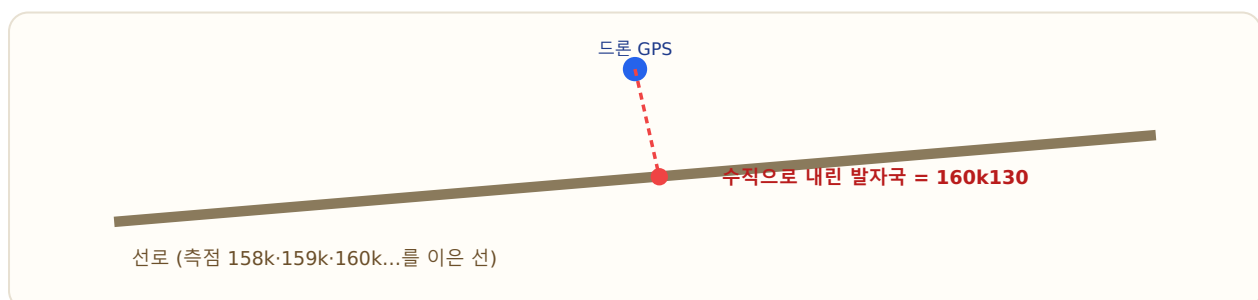


사진 1장마다 (시간, 측점) 짝을 계산해(①) 표로 모아 두면(②), "지금 어디?"와 "거기로 가줘!"를 모두 이 표 하나로 해결한다(③). (코드: StationBar precompute)

GPS를 측점으로 바꾸는 법 — '선로에 수직으로 특'

드론은 선로 바로 위가 아니라 옆에서 비스듬히 날 때도 많아요. 그래서 드론 GPS 점을 **선로(측점들을 이은 선)에 수직으로 특 떨어뜨려**, 그 발자국이 몇 측점인지 읽어요.



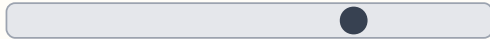
드론이 선로 옆에 있어도, 수직으로 내린 발자국 위치로 '측점값'을 정확히 읽는다. (코드: chainage.ts projectToChain)

4. 재생바의 가로축을 '시간'이 아니라 '실제 이동 거리'로

여기가 특히 똑똑한 부분이에요. 드론이 **멈춰서 맴돌면** 시간은 계속 흐르지만 **위치(측점)는 그대로**예요. 이때 막대 커서가 계속 가면 이상하죠.

그래서 가로축을 "**실제로 이동한 거리**"(측점이 변한 양을 쌓은 값)로 만들었어요. → 멈추면 커서도 멈추고, 움직일 때만 앞으로 가요.

시간 축 (옛 방식)



드론이 멈춰 맴돌아도 커서가 계속 감 🕒

이동거리 축 (새 방식)



멈추면 커서도 멈춤. 움직일 때만 전진 🚶

→ 그래서 커서 위치만 봐도 "드론이 실제로 얼마나 이동했는지"를 알 수 있어요.

가로축 = 축점 변화량을 쌓은 '이동거리'. 전진이든 후진이든 움직이면 커서는 오른쪽으로 간다(방향은 색으로 구분). (코드: cumFracAtTime / timeAtFrac)

5. 방향을 색으로 — 앞으로 가면 주황, 되돌아오면 파랑

드론이 앞으로 가서 축점이 커지면 **주황**, 되돌아와서 축점이 작아지면 **하늘색** 으로 칠해요. 그래서 막대만 봐도 **어디서 앞으로/뒤로 갔는지** 알 수 있어요.

전진(축점 ↑) 주황

후진(축점 ↓) 파랑

다시 전진 주황

색이 바뀌는 지점 = 드론이 방향을 바꾼 곳

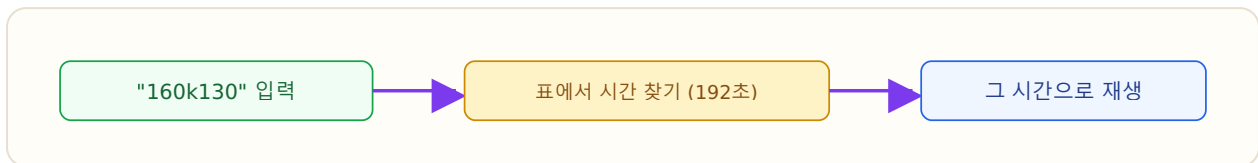
6. 구조물을 '축점 위치'에 딱 배치

교량·터널·역을 각자의 **축점값 위치**에 표시해요. 각 시설물의 실제 축점(예: 회덕터널 160k)과, 드론이 **그 축점을 실제로 지나는 시점**을 맞춰서 막대에 점으로 찍어요. 그래서 "이 구조물이 어디쯤인지"를 막대만 봐도 알 수 있어요.

드론이 같은 곳을 **여러 번 지나가면**(주차장에서 왔다갔다) 그 시설물 마커도 **지날 때마다** 찍혀요.

7. 축점으로 '찾아가기' (검색)

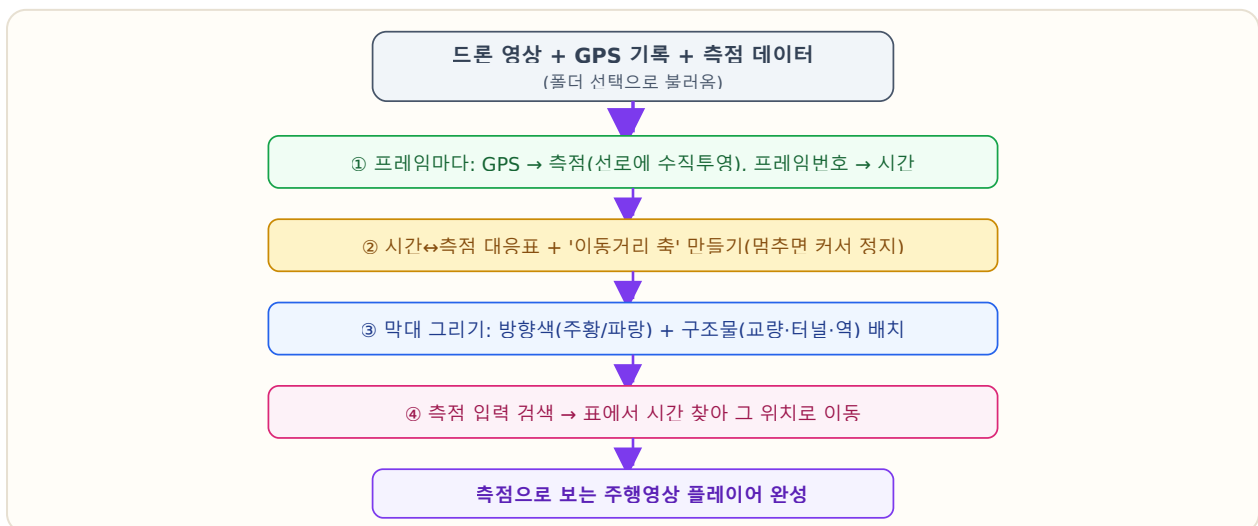
축점값을 입력하면 → **그 축점이 나오는 시간을 표에서 찾아** → **그 시간으로 이동**해요. (3번에서 만든 '시간↔축점 대응표' 덕분!)



여러 번 지난 측점이면 Enter 를 반복해 다음 통과 지점으로 순환 이동. 없는 측점이면 '측점 없음' 안내.

- 실제 측점(드론이 진짜 지난 곳) 으로만 이동해요. 이 영상에 없는 측점이면 엉뚱한 데로 안 가고 "측점 없음" 을 알려줘요.
- 드론이 끝(종점)까지 못 간 경우, 종점 측점은 막대 맨 끝에 '미도착' 으로 표시해요.

8. 전체 흐름 한눈에 (컨베이어 벨트처럼)



단계	하는 일	파일·함수
① GPS→측점	드론 위치를 선로에 수직투영	<code>chainage.ts</code> <code>projectToChain</code>
① 프레임→시간	프레임번호 ÷ fps	<code>StationBar</code> <code>videoFps</code>
② 시간↔측점 대응표	프레임별 사전계산	<code>StationBar</code> <code>precompute</code>
② 이동거리 축	측점 변화량 누적	<code>cumFracAtTime</code> / <code>timeAtFrac</code>
③ 방향색·구조물	측점 증감/측점값 기준	<code>barSegments</code> / <code>structureMarks</code>
④ 측점 검색	측점→시간 역변환	<code>handleJumpToMileage</code>

9. 한 문장 결론

"드론 GPS로 매 장면의 측점을 계산해 '시간↔측점 대응표'를 만들어 두었기 때문에, 시간이 아니라 측점(선로 위치)으로 영상을 찾아보고 이동할 수 있게 된 것" 이에요.

이 덕분에 GhiVideo는 단순 영상이 아니라 "지금 선로 어디를 보고 있는지 아는" 주행영상 플레이어가 됩니다.