

발표_GhiVideo_좌표정합과측점기반구현

GhiVideo 발표 자료

드론 GPS · 영상 · POI 좌표를 영상 위 정확한 위치에 — 그리고 프레임에서
측점(스테이션)으로

작성일: 2026-06-30 · 발표용 요약 문서 슬라이드 구분은 --- 입니다 (Marp/reveal.js 변환 가능). 각 장은 "그림 → 핵심 → 코드 근거" 순.

목차

공간 정합	시간 동기화	표시 안정	측점 전환
지도좌표 → 화면픽셀	영상시각 ↔ 드론프레임	떨림·튐 제거	프레임 → 측점(공간)

1. 우리가 풀어야 했던 문제
2. 입력 데이터 3종 — GPS · 영상 · POI
3. 전체 파이프라인 한 장
4. **핵심 ①** 좌표 정합 — 지도 좌표를 화면 픽셀로 (투영)
5. **핵심 ②** 시간 동기화 — 영상 시각과 드론 프레임 맞추기
6. **핵심 ③** 라벨 안정화 — 떨림·튐 제거
7. **핵심 ④** 프레임 기반 → 측점(스테이션) 기반 전환
8. 정확도 보정 도구 (DEM · 드래그 · 세로화각)
9. 정리 — 무엇을, 어떻게

1. 우리가 풀어야 했던 문제

드론이 철도 노선을 따라 비행하며 찍은 영상 위에, 교량·터널·역사·지장물(POI)의 이름표를 실제 위치에 정확히 띄우고 싶다.



그림 1. 목표 — 흔들리는 드론 영상 위에, 지도 좌표만 가진 POI를 "진짜 그 자리"에 표시.

난이도 4가지 → 그래서 세 가지 정합이 필요합니다.

난이도	무엇이 어려운가	해결 정합
드론 흔들림	yaw/pitch/roll·고도 변화·빠른 이동	공간 + 안정
POI는 지도좌표만	"화면 어디"인지 모름	공간(투영)
시간 어긋남	영상 시각 ↔ 드론 데이터 시각	시간
매끄러움	60fps·떨림 없이	안정

2. 입력 데이터 3종



그림 2. 드론 로그(언제·어디·어느 방향) + 영상 + 지도 POI를 결합.

3. 전체 파이프라인 한 장



그림 3. 시각 t → pose 보간 → 4단계 투영 → 평활 → 렌더. 동시에 GPS는 측점값으로 투영되어 측점바로.

4. 핵심 ① 좌표 정합 — 지도 좌표를 화면 픽셀로

"POI는 위경도만 안다. 그게 **지금 화면 어디에 보이는가?**" — 이게 이 프로그램의 심장. **4 단계 투영**으로 푼다.



그림 4. 4단계 투영 파이프라인. 파일: client/src/utils/geoProjection.ts

4-1. 둥근 지구를 평평하게 (ENU 변환)

지구는 둥글어 거리 계산이 어렵습니다. 노선 한 구간만 잘라 **평평한 모눈종이(미터 단위)** 로 펴니다.

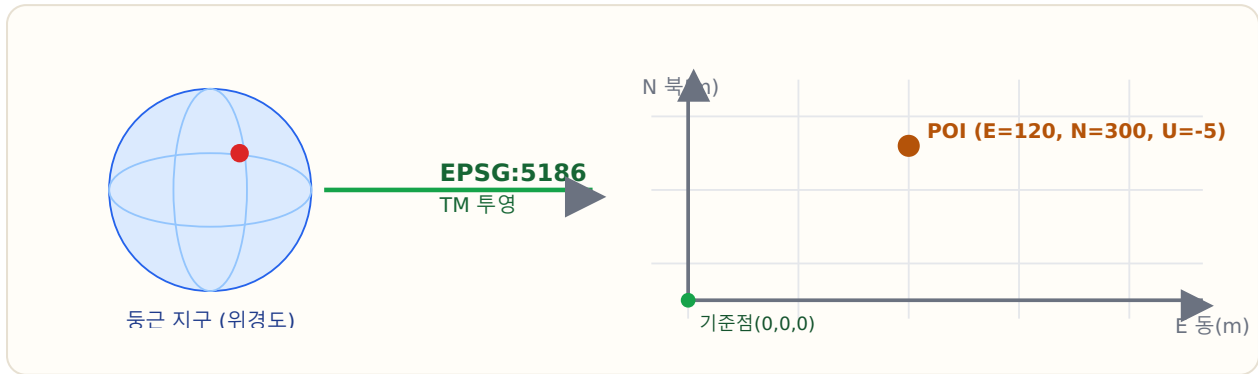


그림 5. E/N = 동·북 몇 m, U = 기준점 대비 상대 높이($\text{alt} - \text{refAlt}$).

```
// geoToEnu (91-97)
const [e, n] = latLonToTM(lat, lon); // 위경도 → 평면 미터
return [e, n, alt - refAlt];
```

4-2. 드론이 보는 방향으로 회전 (카메라 좌표)

세상 좌표(E/N/U)를 드론의 **yaw-pitch-roll** 회전행렬에 곱해 "카메라가 보는 좌표(X_c, Y_c, Z_c)"로 바꿉니다.

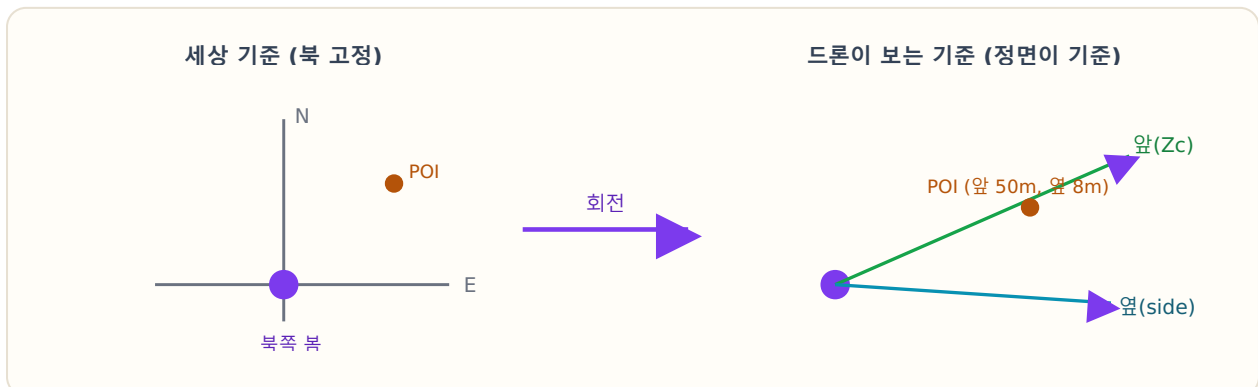


그림 6. 북 기준 좌표를 "드론 정면 기준"으로 회전. 진행방향 거리 fwd(앞)·side(옆)도 함께 산출.

```
// toCameraCoords (301-317)
cc.fwd = relEnu[0]*sy + relEnu[1]*cy; // +면 앞쪽
cc.side = relEnu[0]*cy - relEnu[1]*sy; // +면 오른쪽 → "앞은 멀리, 옆은 가깝게" 비등방 필터 근거
```

4-3. 3D를 납작한 사진으로 (핀홀 카메라 투영)

핵심 한 줄: 깊이(Z_c)로 나눈다 → 멀수록 화면 가운데로 작게. (바늘구멍 사진기 원리)

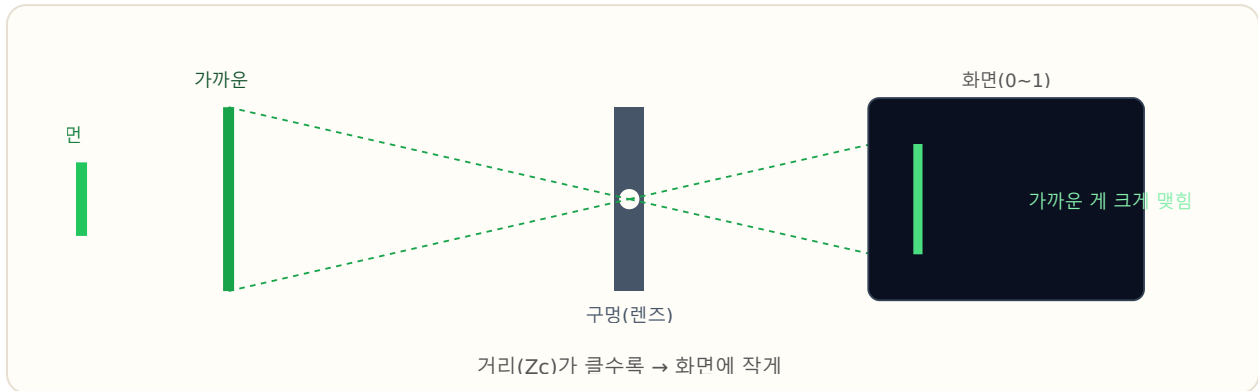


그림 7. $pxRaw = 0.5 + (Xc/Zc) \cdot (f/sensorW)$. 0.5=정중앙, $f/sensor$ =화각.

```
// pixelFromCamera (126-137)
pxRaw = (0.5 + cx0) + (Xc / Zc) * (f / sW); // 가로 0~1
pyRaw = (0.5 + cy0) + (Yc / Zc) * (f / sH); // 세로 0~1
```

4-4. 높이 기준 통일 (지오이드 보정) + 화면 정렬

두 개의 '0층' 문제: 지도 높이(정표고·해발)와 GPS 높이(타원체고)는 기준이 달라 대전 기준 약 25.8m 차이. 안 맞추면 라벨이 위아래로 어긋납니다.

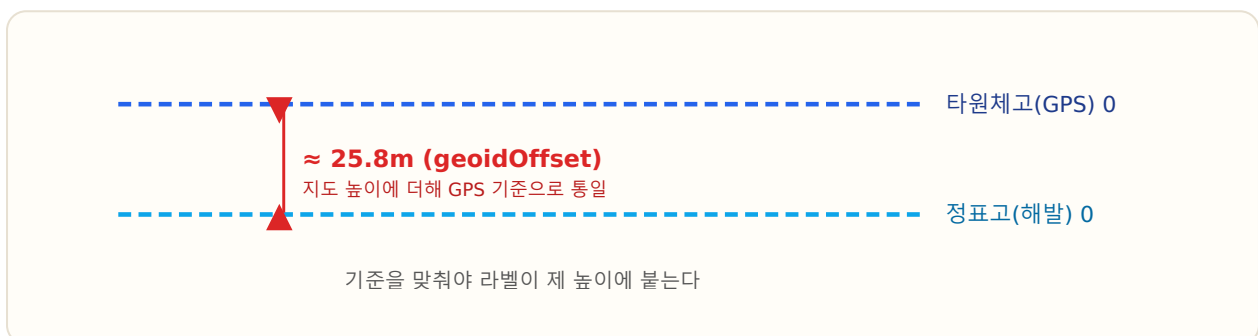


그림 8. 지도 표고 + geoidOffset → GPS와 같은 기준. 마지막에 object-fit:cover 잘림을 반영해 0~1 → 실제 px.

```
geoToEnu(lat, lon, targetAlt + params.geoidOffset, ...); // 높이 기준 통일
// 이후 coverRef(StationOverlay 921-930)로 정규(0~1) → 실제 화면 px 정렬
```

→ 4-1 ~ 4-4를 거치면 POI가 영상 속 진짜 그 자리에 찍힙니다.

5. 핵심 ② 시간 동기화 — 영상 시각 ↔ 드론 프레임

좌표가 맞아도 "지금 영상 시각의 드론 위치" 를 못 잡으면 라벨이 엉뚱한 데 뜹니다.

문제: 브라우저 `video.currentTime` 은 ~250ms 간격으로만 갱신 → 커서·라벨이 뚝뚝 끊김.
 해결: `smoothTimeRef` — 벽시계로 60fps 단조 보간.

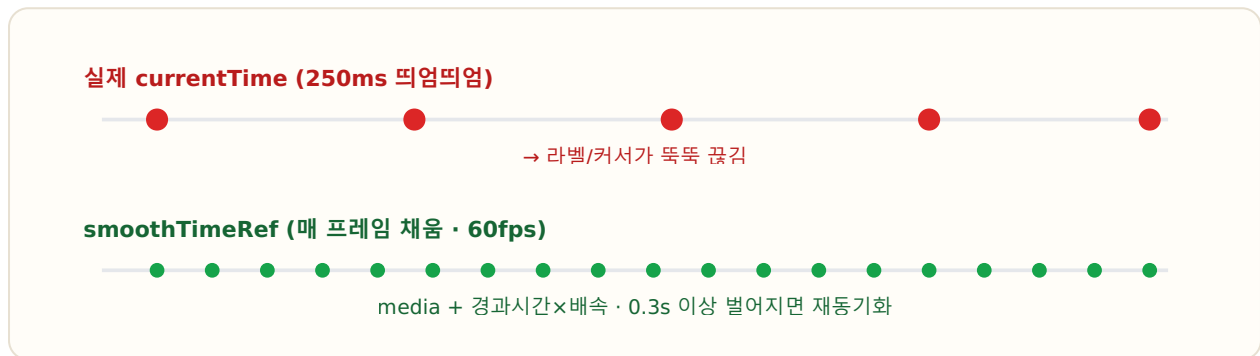


그림 9. 띄엄띄엄한 실제 시각을 60fps로 메워 부드럽게. 이 t로 드론 pose를 보간.

```
// VideoPlayer.tsx 62-117
let est = a.media + ((performance.now() - a.wall)/1000) * rate;
if (real - est > 0.3) est = real; // 재동기화
smoothTimeRef.current = t; // ref로 노출 → 리렌더 없이 매 프레임 읽음
```

6. 핵심 ③ 라벨 안정화 — 떨림·튐 제거

드론은 미세하게 흔들립니다. 그대로 투영하면 라벨이 부르르 떨립니다. → **One Euro** 방식 속도 적응 평활 + 이상치 거부.

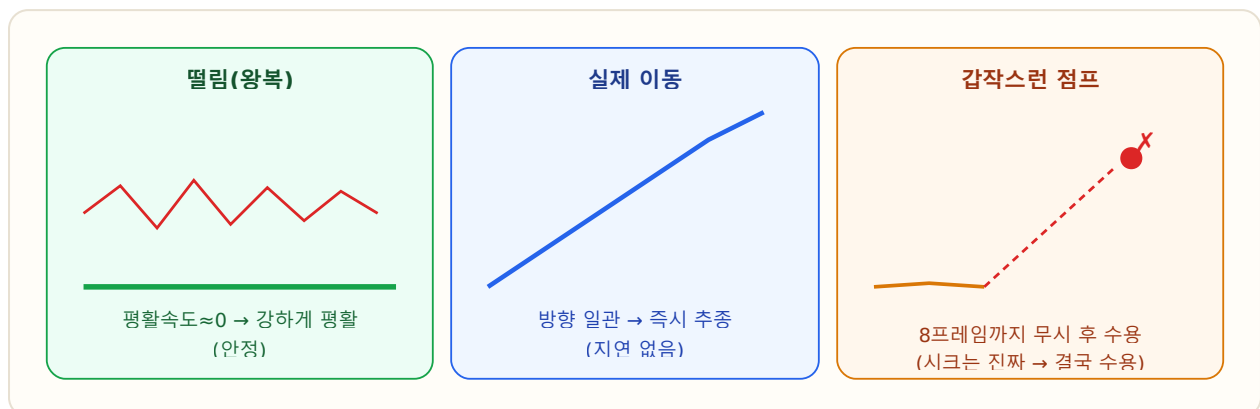


그림 10. 떨림은 죽이고, 진짜 이동은 즉시 따라가고, 튐(점프)은 무시. 세 동작을 한 필터로.

```
// StationOverlay.tsx 48-61
if (d > REJECT_DIST && prev.rej < MAX_REJECT_FRAMES) // 0.12 초과 점프 = 이
  상치
  return { ...prev, rej: prev.rej + 1 }; // 위치 유지
const a = min(maxAlpha, minAlpha + (maxAlpha-minAlpha)*min(1, speed/
  speedRef));
return { x: prev.x + dx*a, y: prev.y + dy*a, ... }; // 속도 클수록 빠르게 추
  중
```

7. 핵심 ④ 프레임 기반 → 측점(스테이션) 기반

7-1. 왜 바꿨나? — 시간축의 한계

철도 현장은 **위치(측점)**로 말합니다: "157K970 지점의 교량". 그런데 시간축은 **드론이 호버(정지)** 하면 무너집니다.

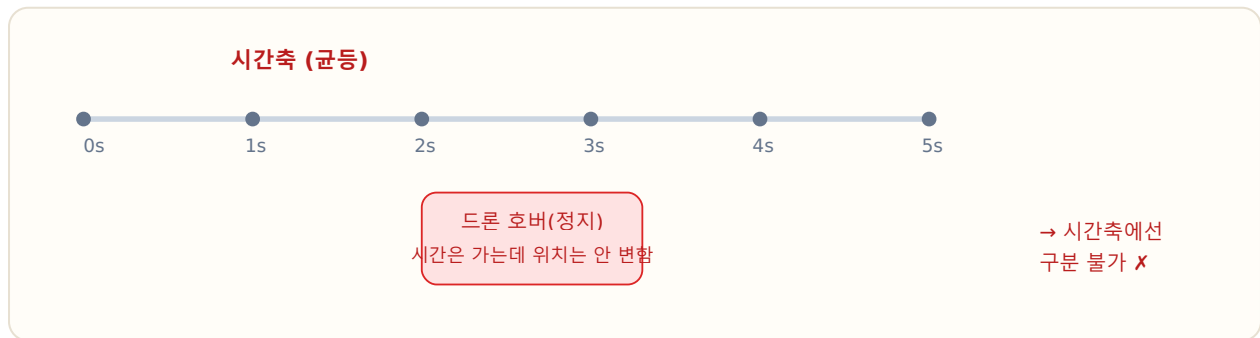


그림 11. 시간 균등축에선 "공중 대기(호버)" 구간을 표현할 수 없다.

→ 그래서 **측점(공간)축**으로 재해석: 드론이 호버하면 측점도 멈춰 있어야 자연스럽게.

7-2. 구현 ① — GPS를 측점값으로 투영

드론 GPS를 측점 폴리라인(선로)에 **수직으로 내려** "측점값(km)"과 "선로 이격(m)"을 구합니다.



그림 12. 측점명 "157K970" → 157970m 환산으로 선로를 만들고, GPS를 수직 투영해 측점값·이격 산출.

```
// chainage.ts projectToChain (35~50)
const t = clamp(((px-a.x)*dx + (py-a.y)*dy) / L2, 0, 1); // 선분 위 투영 비율
bestKm = a.km + (b.km - a.km) * t; // 보간 측점값
```

7-3. 구현 ② — 이동거리축 (핵심 아이디어)

시간축 대신 **드론이 실제로 이동한 거리**를 축으로 씁니다. 측점값 변화량 $|\Delta|$ 을 매 프레임 누적.

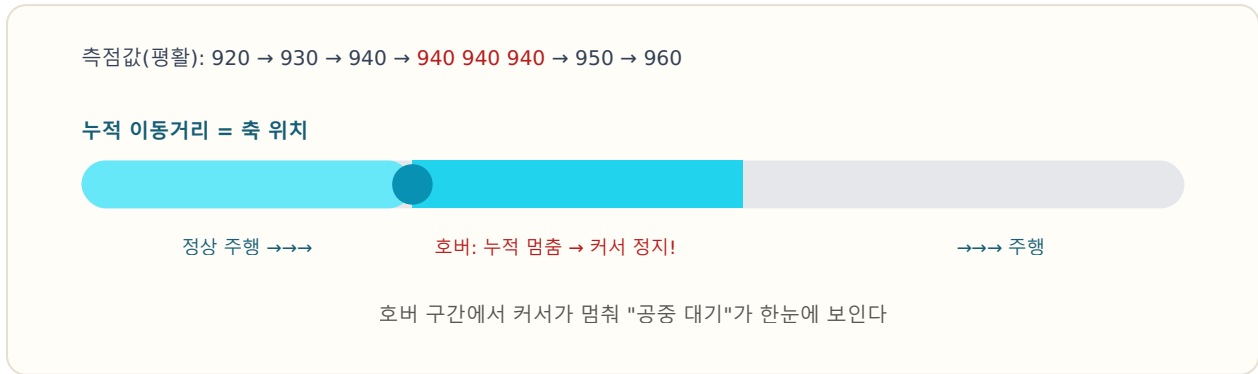


그림 13. $|\Delta\text{측점값}|$ 을 누적해 0~1로 정규화 → 축 위치. 호버는 누적이 멈추므로 커서도 정지.

```
// StationBar.tsx 233-274
sm[i] = (pre[hi]-pre[lo])/(hi-lo);           // 측정값 ±8프레임 평활
if (i>0) cum += Math.abs(sm[i]-sm[i-1]);    // |Δ| 누적 = 실제 이동거리
frac[i] = cum / total;                      // 0~1 정규화 → 축 위치
```

7-4. 측정 기반의 결과 — 하단 측정바

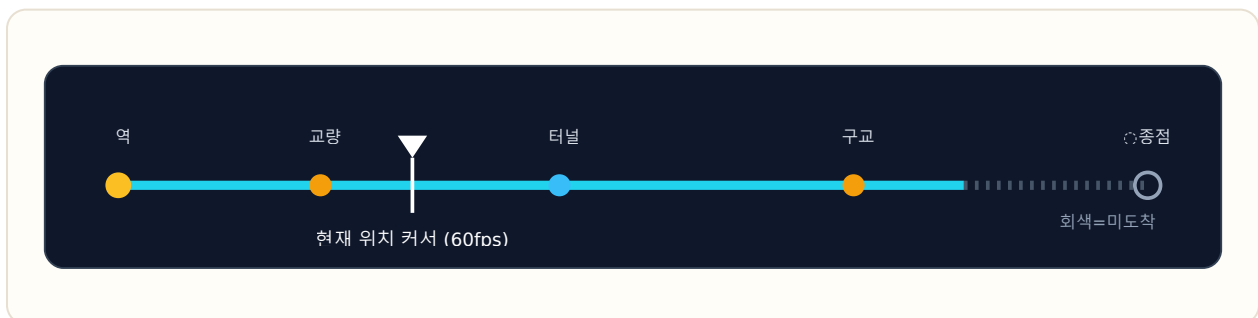


그림 14. 측정·구조물을 공간축에 배치. 영상별 FPS 자동산출로 배치 정밀화 · 종점 미도착은 회색 + 빈 링.

7-5. 프레임 ↔ 측정 — 한 장 비교

구분	프레임 기반	측점 기반 (현재)
축	시간(초)	실제 이동거리/측점(km)
호버	구분 불가	커서 정지로 가시화
현장 용어	"3분 20초"	"157K970 교량"
위치 질의	어려움	GPS→측점 투영으로 즉답
동기화	currentTime	smoothTimeRef + 측정투영

→ 시간축 영상을 공간축(측점)으로 재해석한 것이 이 프로그램의 정체성.

8. 정확도 보정 도구

투영은 카메라 파라미터·표고 가정에 민감 → **사용자가 미세보정**할 수 있게 함. 핵심은 투영의 **역방향**(화면→세계)을 푸는 것.

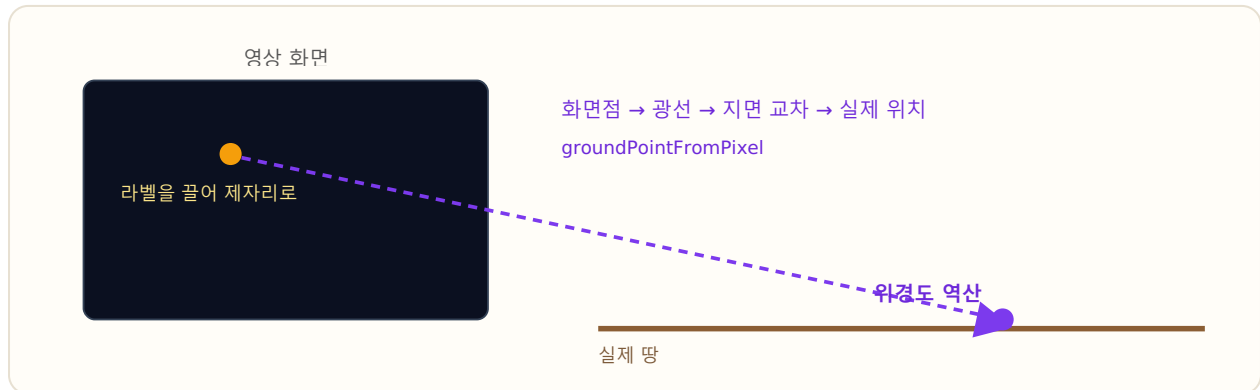


그림 15. "끌어다 맞추기" = 역투영. 끈 화면점을 광선으로 쏘 지면과 만나는 실제 좌표를 복원.

도구	동작
① DEM 자동표고	모든 POI를 실제 지형고도로 (서버 <code>/api/elevation</code> , SRTM 30m)
② 드래그 편집	라벨을 끌면 <code>groundPointFromPixel</code> 로 위경도 역산
③ 세로화각 보정	POI를 실제 위치로 끌면 <code>sensorH</code> 만 역산 자동보정
④ 보정값 저장	<code>poi_overrides.json</code> 내보내기/가져오기

9. 정리 — 무엇을, 어떻게

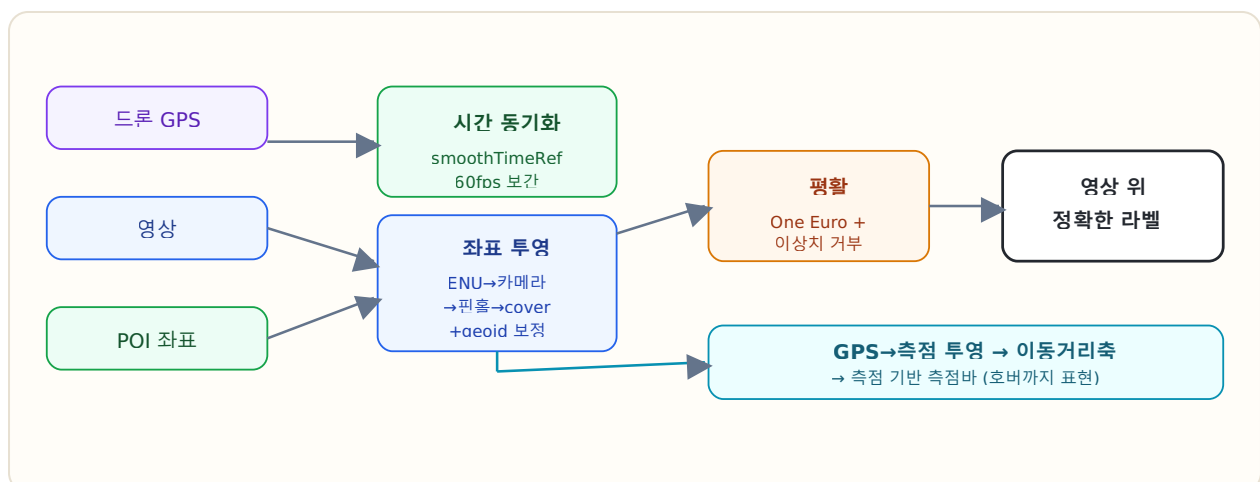


그림 16. 3대 정합(공간·시간·안정) + 패러다임 전환(프레임→측점).

3대 정합 기술

1. **공간**: 4단계 투영(ENU→카메라회전→핀홀→cover) + 지오이드 보정 + 역투영 보정
2. **시간**: smoothTimeRef 60fps 단조보간으로 영상시각↔드론프레임 정합
3. **안정**: One Euro 속도적응 평활 + 이상치 거부

패러다임 전환

- 프레임(시간) 기반 → **측점(공간) 기반**: GPS를 선로에 투영해 측점값화, 이동거리축으로 호버까지 표현

부록 — 핵심 소스 색인

기술	파일	라인
ENU 변환	geoProjection.ts	91–97
카메라 좌표 + fwd/side	geoProjection.ts	301–317
핀홀 투영	geoProjection.ts	126–137
지오이드 보정	geoProjection.ts	53–67, projectPoint
역투영(보정)	geoProjection.ts	149–251
object-fit cover 정렬	StationOverlay.tsx	921–930
라벨 평활/이상치	StationOverlay.tsx	48–61
RAF 60fps 렌더	StationOverlay.tsx	905–1198
smoothTimeRef	VideoPlayer.tsx	62–117
측점 투영(chainage)	chainage.ts	10–64
이동거리축	StationBar.tsx	233–274
종점역 미도착	StationBar.tsx	300–310, 574–586
DEM 표고	StationOverlay.tsx / elevation.ts	1384–1415 / 14–67

상세 구현은 [구현상세_GhiVideo_기술-소스코드매칭.md](#) 참조.