

기술명세_GhiVideo_종합기술문서

GhiVideo 종합 기술 명세서

목적: 본 문서는 (1) 특허 출원/발명 신고의 기초 자료와 (2) 후임 개발자 인계·업그레이드 자료를 겸한다. **대상 시스템:** GhiVideo — 시간축이 아니라 노선 위치(측점·체이니지) 를 1차 좌표로 삼아 드론 주행영상을 색인·탐색하고, 프레임별 카메라 자세로 측점·POI·선로중심선을 영상 위에 실시간 정합(geo-registration)하는 철도 주행영상 분석 플레이어. **작성일:** 2026-06-29 · 모든 소스 참조는 `[경로](경로#Lxx)` 형식(클릭 가능).

0. 문서 구성과 읽는 법

장	내용	주 독자
1	시스템 개요·아키텍처·데이터 흐름	전체
2	좌표·투영 정보 파이프라인 (모든 정합의 기반)	전체
3	카메라 투영 & 화각(FOV) 정합	개발/특허
4	측점 표고 정합 (DEM-free) — 발명 B	특허
5	드론 자세 평활/보간 + 오버레이 렌더링	개발
6	측점(체이니지) 시스템 & 스테이션바	개발/특허
7	위치보정 역투영 — 발명 C	특허
8	백엔드 스트리밍/FFmpeg·데이터·상태관리	개발
9	개발 타임라인(주요 의사결정)	전체
10	특허 관점 종합 (인벤토리·후보·청구포인트·선행기술)	특허
11	검증 결과표	전체
12	후임자 인계(데이터흐름·튜닝상수·빌드·업그레이드)	개발
13	한계 / 출원 전 확인사항	전체

1. 시스템 개요 & 아키텍처

1.1 핵심 아이디어

드론으로 촬영한 철도 선로 주행영상의 각 프레임에는 SRT 메타데이터로 **드론 위치(lat/lon/고도)·자세(yaw/pitch/roll)·초점거리**가 기록된다. 한편 철도 GIS에는 **측점·POI·선로 중심선의 WGS84 좌표(+표고)**가 있다. 이 둘을 **핀홀 카메라 투영**으로 결합하면:

1. **순방향 투영** — "이 GIS 지점이 지금 화면 어느 픽셀에 보이는가" → 라벨/중심선/궤적 오버레이.
2. **역투영** — "사용자가 화면에서 끈 픽셀은 어느 지리좌표인가" → POI 위치/표고 보정.
3. **측점 색인** — 드론 GPS를 측점 폴리라인에 투영해 영상 매 시점을 **측점값(체이니지)**으로 환산 → 측점 기반 탐색/스크러빙.

1.2 데이터 흐름

[폴더 선택: base.csv(드론프레임) + building/측점·교량·터널·역사 CSV + route.json + KMZ]

geoData.loadFolderGeoData (인코딩 자동감지·병렬 파싱·KMZ 우선)

▼

[geoStore] frames / stations / pois / structures / centerline / origin / routeMeta

|

→ StationOverlay (영상 위 AR 오버레이)

precompute(가시성/겹침, requestIdleCallback) → RAF(연속포즈 재투영) → Canvas

geoProjection: WGS84→TM→ENU→카메라→핀홀

|

→ StationBar (하단 측점 스크러버)

chainage: 드론GPS→측점폴리라인 수직투영→연속 체이니지

이동거리/측점축, 방향색, 구조물 마커, 측점검색 순환

|

→ VideoPlayer (Video.js + hls.js, 이중 재생경로, 프레임 정확도)

1.3 기술 스택

- **프론트엔드**: React 18 + TypeScript, Video.js 8.23 + @videojs/http-streaming, hls.js 1.6, Zustand(상태), Vite, Canvas 2D(오버레이), proj4(좌표변환), fflate(KMZ 압축해제).
- **백엔드**: Node.js 20 + Express, `child_process.spawn` FFmpeg 래퍼, Range Request 스트리밍, HLS 변환 + SSE 진행률.
- **배포**: PM2(`ghiVideo`, 포트 55000)가 `client/dist` 정적 서빙(빌드 결과물). 소스 변경 시 `npm run build -w client` (Node 20) 후 반영.

2. 좌표·투영 정보 파이프라인 (★ 모든 정합의 기반)

본 절을 **정보**으로 하고, 3~7장은 이를 참조한다. Python 레퍼런스

`advanced_tuner_v2.py` 와 회전행렬·축정렬·정규화 공식이 1:1 동치로 이식되어 있다 ([geoProjection.ts:1-12](#)).

2.1 5단계 변환

- [1] WGS84 (lat, lon, EL정표고)
 - | proj4 EPSG:4326 → EPSG:5186 (한국 중부원점 TM)
- ▼
- [2] 월드 ENU [East(m), North(m), Up(m)] (East=X, North=Y)
 - | rel = target_ENU - (drone_ENU + offset)
- ▼
- [3] 카메라좌표 [Xc, Yc, Zc] ($R_{w2c} = R_{align} \cdot R_{b2w}(\text{yaw}, \text{pitch}, \text{roll})^T$)
 - | 핀홀 + focal/sensor
- ▼
- [4] 정규 이미지 좌표 [px, py] $\in (0\sim1)$ (0.5 = 영상 중심)
 - | object-fit:cover 변환
- ▼
- [5] 화면 px

2.2 단계별 수식·코드

단계 1 — WGS84 → TM ([geoProjection.ts:75-85](#))

```
proj4.defs('EPSG:5186', '+proj=tmerc +lat_0=38 +lon_0=127 +k=1 +x_0=200000  
+y_0=600000 +ellps=GRS80 +units=m +no_defs');  
function latLonToTM(lat, lon) { const [e, n] = _toTM.forward([lon, lat]);  
  return [e, n]; }
```

단계 2 — TM → ENU + 표고 datum 정합 ([geoProjection.ts:91-97](#), [263-271](#)). X/Y는 절대 TM, Z만 기준고도 대비 상대(상대벡터를 빼므로 무관). **핵심: GIS 정표고(EL) + 지오이드고(N) → 타원체고로 변환해 드론 GPS 타원체고와 datum을 맞춘다.**

```
const stEnu = geoToEnu(targetLat, targetLon, targetAlt +  
  (params.geoidOffset ?? 0), ...); // EL + N  
const drEnuAdj = [drEnu[0]+offX, drEnu[1]+offY, drEnu[2]+offZ];  
const relEnu = [stEnu[0]-drEnuAdj[0], stEnu[1]-drEnuAdj[1], stEnu[2]-  
  drEnuAdj[2]];
```

지오이드고 N: 대전지역 $\approx 25.8\text{m}$ (= 84.244 - 16.853 - 41.59, KNGeoid 검증값). 초기 문서엔 25.449m로 기록된 바 있으나 본 명세는 25.8m로 통일(\$11). 지역 상수이므로 타 지역 데이터셋은 재튜닝 필요.

단계 3 — ENU $\rightarrow$ 카메라좌표 ([geoProjection.ts:275-295](#)). $R_{b2w} =$

$R_z(-\text{yaw}) \cdot R_x(\text{pitch}) \cdot R_y(\text{roll})$, $R_{w2c} = R_{\text{align}} \cdot R_{b2w}^T$, $R_{\text{align}} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix}$. 행렬을 명시 생성하지 않고 전치(행=열)·부호치환으로 직접 계산:

```
function applyRw2c(b2w, rel) {
  return {
    Xc: b2w[0][0]*rel[0] + b2w[1][0]*rel[1] + b2w[2][0]*rel[2], // col0
    Yc: -(b2w[0][2]*rel[0] + b2w[1][2]*rel[1] + b2w[2][2]*rel[2]), // -col2
    Zc: b2w[0][1]*rel[0] + b2w[1][1]*rel[1] + b2w[2][1]*rel[2], // col1
  };
}
```

규약: $Z_c > 0$ = 카메라 정면(영상 깊이축), $X_c = \text{우}$, $Y_c = \text{하}$. $Z_c \leq 0$ 은 카메라 뒤 $\rightarrow$ null.

단계 4 — 카메라좌표 $\rightarrow$ 정규 픽셀(핀홀) ([geoProjection.ts:126-137](#)). 영상 중심 = 0.5, 해상도 독립:

```
pxRaw = (0.5 + cx0) + (Xc/Zc) * (f / sensorW); // 가로
pyRaw = (0.5 + cy0) + (Yc/Zc) * (f / sensorH); // 세로
```

단계 5 — 정규 $\rightarrow$ 화면 px (object-fit:cover) ([StationOverlay.tsx:923-933](#)). 영상 크롭 사각형에 맞춰 정합:

```
const s = Math.max(W / vW, H / vH); // cover: 더 큰 배울(넘침=크롭)
dispW = vW * s; dispH = vH * s; offX = (W - dispW)/2; offY = (H - dispH)/2;
const vx = (nx) => offX + nx * dispW; const vy = (ny) => offY + ny * dispH;
```

2.3 두 투영 경로의 의도적 분리

- 렌더(정밀): [geoProjection.ts](#) — proj4 EPSG:5186 TM + 회전행렬 + geoid/포커스 보정. (3~5장)
- 검색(경량): [geoSearch.ts](#) — cos-lat 근사 ENU. "현재 시야 내 POI" 또는 "특정 POI가 보이는 프레임 구간" 산출용.

ENU 월드원점은 첫 축점(없으면 첫 드론프레임, 없으면 중심선 첫 점)([geoSearch.ts:117-128](#)).

3. 카메라 투영 & 화각(FOV) 정합

좌표 변환 파이프라인은 §2 참조. 본 절은 화각·내부표정 보정과 역투영을 다룬다.

3.1 focal ↔ FOV 관계

화각은 focal과 센서 치수로 결정된다(핀홀): $FOV = 2 \cdot \text{atan}(\text{sensor} / (2 \cdot \text{focal}))$ ([StationOverlay.tsx:864](#)). 기본값 `focal=24mm`, `sensorW=36mm`(풀프레임 환산), `sensorH=20.25mm(=36×9/16, 16:9)` ([geoProjection.ts:48-66](#)). 이 hFOV는 나침반/미니맵 시야부채꼴에도 사용.

3.2 fovMode — 세로 화각(sensorH) 역산 보정 (★)

라벨이 멀리/가까이에서 **상하로 어긋날 때**, 사용자가 POI 하나를 영상 속 실제 위치로 드래그하면 그 한 점의 대응으로 세로 화각을 역산한다. **설계 핵심: 가로 focal(f/sensorW)은 건드리지 않고 sensorH만 역산** → 빨간 중심선이 좌우로 들어지지 않으면서 상하 배율만 교정.

투영식 $py = 0.5 + cy_0 + (Y_c/Z_c) \cdot (f/sH)$ 를 sH에 대해 풀면 ([StationOverlay.tsx:1326-1344](#)):

```
const cc = toCameraCoords(drone, drag.lat, drag.lon, poiZ, p, wo);
const b = cc.Yc / cc.Zc;
const v = pos.y - 0.5 - (p.cy0 ?? 0);
const sHNew = (b * p.focalLen) / v; // sH = (Yc/Zc)·f / (py - 0.5 - cy0)
setParam('sensorH', Math.max(6, Math.min(36, sHNew)));
```

영상 세로 중심($v \approx 0$)은 발산 → $|v| < 0.03$ 이면 거부하고 안내.

3.3 핵심 함수

- **toCameraCoords** ([geoProjection.ts:301-317](#)) — GIS점→카메라좌표. 부산물 `distH`(수평 직선거리), `fwd/side`(진행방향 앞/옆 성분, 비등방 거리필터용). `pixelFromCamera`와 분리한 이유: 중심선처럼 선분 단위 근거리 Z_c 클리핑·보간 후 투영해야 하기 때문.
- **pixelFromCamera** ([geoProjection.ts:126-137](#)) — 카메라좌표→정규픽셀(클램프 없음).
- **groundPointFromPixel** ([geoProjection.ts:218-251](#)) — 화면픽셀→지면 lat/lon. 광선-수평면 교차:

```
const t = relUpTarget / dir[2]; // dir = R_w2c^T · ((px-0.5-cx0)·sW/f, (py-0.5-cy0)·sH/f, 1)
if (t <= 0) return null; // 카메라 뒤/위
const E = drEnu[0]+offX + t*dir[0]; const N = drEnu[1]+offY + t*dir[1];
const [lon, lat] = _toTM.inverse([E, N]);
```

- 관련 역투영 2종: `worldFromPixel` (슬랜트거리 유지 lat/lon/z 동시복원, [149-183](#)), `solveZForPixelY` (lat/lon 고정, py에 맞는 z만 1차식 역산, [194-211](#)). → 발명 C(\$7).

3.4 보정값 영속화 (데이터셋별)

- 카메라/내부표정·표시설정: `ghivideo:calib:{baseName}` (로드 1회, 저장 800ms debounce, [StationOverlay.tsx:473-500](#)).
- POI 위치/표고 override(DEM·드래그): `ghivideo:poi ov:{baseName}` (폴더 파일 override 위에 localStorage 병합, [502-524](#)).

3.5 설계 트레이드오프·한계

- 렌즈 왜곡 미보정: 핀홀만 사용(방사/접선 왜곡 계수 없음). 분석 보조 라벨 정합엔 표준 정확도로 충분(가장자리 수 px 오차 가능).
- 클라이언트 순수 계산: proj4만 의존, 서버 왕복 없이 60fps RAF에서 직접 투영.
- focal 분리 보정: 한 점 대응으로 sensorH 한 축만 역산(가로 정합 보존).
- POI 표고 가정: 실측 표고 없으면 최근접 선로 지면표고(또는 "드론고도-이격") → §4·드래그·DEM 보정.

4. 측점 표고 정합 (DEM-free 영상정합) — 발명 B

4.1 무엇을 / 왜

표시할 시설물(POI)은 주소 기반이라 **표고값이 없는** 경우가 대부분이다. 표고가 없으면 부각(수직각) 산출 불가 → 종래는 지면을 평면으로 가정. 그러나 노선은 실제 기복(정표고 41.6m→57.5m)이 있어 평면 가정 시 라벨이 화면 밖/지면 아래로 이탈한다. 실측에서 잘못된 고도(≈ 0) 사용 시 **부각오차 42.7°**.

→ 별도 DEM 취득 없이, **노선을 따라 실측된 측점의 정표고**를 지면고도원으로 삼아 정합한다.

4.2 구성(독립항 필수요소)

1. **측점 표고 수열 보유** — 노선 따라 실측된 복수 측점의 (lat/lon + 정표고 EL). (실데이터: 측점 47개, 41.6→57.5m)
2. **최근접 측점 지면고도 채택** — 표고 미상 대상에 가장 가까운 측점의 정표고를 지면고도로. 프레임 진행에 따라 최근접 측점이 바뀌어 **지면고도가 노선 기복을 따라 갱신**.
3. **지오이드 datum 정합** — 정표고(EL) + 지오이드고(N) → 타원체고 → 드론 GPS와 동일 기준. `rel_up = (targetZ + N) - droneAlt`.
4. **정합 투영** — §2 파이프라인 수행.
5. (종속) **대상별 표고 오프셋(poiZOffset)** — 건물높이/지반차 보정, 측점·중심선엔 미적용.

4.3 검증값

타원체고 = 정표고(EL) + 지오이드고(N≈25.8m, 대전)

측점 157K900, frame0, 수평거리 91.3m

정표고≈0 → 부각 42.7°

정표고 41.5 → 부각 25.1°

정표고+N≈67 → 부각 11.0° (드론 rel_alt 16.85m와 일치)

- **데이터 함정(핵심):** building/01)측점.csv 에 표고 컬럼이 둘 — Z좌표 (로컬 ≈0) / Z좌표_한국 (EPSG:5186 정표고 ≈41.5m). 기존 코드가 Z좌표 (≈0)를 읽어 42.7° 오차 발생 → Z좌표_한국 우선으로 교정([geoData.ts](#) [측점 파서](#)).
- **효과:** 부각오차 42.7°→**11.0°**, 구글어스 DEM 대비 **0.2m 일치**(2160p에서 ~6px). DEM 취득비용 불필요. 공개 DEM(SRTM 30m)보다 측량 측점이 더 정밀.

5. 드론 자세 평활/보간 + 영상 오버레이 렌더링

두 본질 과제: (a) **자세/GPS 노이즈**(단순평균은 회전 뭉갸→지연), (b) **렌더 부드러움 vs 계산 비용**(가시성 판정은 비싸고 60fps는 매 프레임 재투영 요구).

5.1 에지(회전경계) 보존 적응형 이동평균 — `smoothFrame`

[StationOverlay.tsx:577-604](#). 중심 `i` 에서 좌우 최대 `halfWin` (기본 60fr≈2초)까지 평균하되, 중심 yaw와 차이가 `YAW_EDGE=8°` 이내인 인접까지만 창 확장.

```
const YAW_EDGE = 8;
const near = (idx) => Math.abs(((frames[idx].yaw - yc + 540) % 360) - 180)
  <= YAW_EDGE;
while (lo > loMin && near(lo - 1)) lo--; // 회전경계에서 멈춤
while (hi < hiMax && near(hi + 1)) hi++;
// yaw는 원형평균: atan2(Σsin, Σcos)
yaw: Math.atan2(sinYaw / n, cosYaw / n) * 180 / Math.PI,
```

- 직선: 창 풀폭 → 노이즈 강하게 억제. 회전 중: 창 좁아져 **즉시 추종**(지연 없음). 회전 후: **한 프레임씩** 점진 재확장(튐 없음). 과거 '변화율 기반 축소'는 회전 종료 시 한꺼번에 재확장돼 회전 프레임이 다시 섞여 1회 튕던 부작용 → 경계보존으로 교체.

5.2 연속 프레임번호 보간 포즈 — `poseAt`

[StationOverlay.tsx:670-690](#). 실수 프레임번호 `estFrame` 에 대해 양 이웃의 평활포즈를 선형보간(이산 30fps 격자→60fps 연속). yaw는 **최단각** 보간:

```
let dy = b.yaw - a.yaw; dy = ((dy + 540) % 360) - 180; // 최단각
return { ...a, frame: estFrame, lat:L(a.lat,b.lat), ..., yaw: a.yaw +
  dy*frac };
```

O(log n) 이진탐색으로 매 RAF 호출 가능.

5.3 속도적응 EMA + 이상치 거부 — smoothStep (One-Euro 사상)

[StationOverlay.tsx:48-61](#). 느릴 때(떨림)는 강하게 평활, 빠를 때(실제 팬)는 즉시 추종.

```
if (d > REJECT_DIST && prev.rej < MAX_REJECT_FRAMES) // 0.12 초과 점프 = 이상
  치 → 위치 유지

return { x: prev.x, y: prev.y, rej: prev.rej + 1, ... };
const vx = prev.vx + (dx - prev.vx) * SMOOTH_VEL_BETA; // β=0.25 속도평활(떨림
  왕복은 상쇄)
const a = Math.min(maxAlpha, minAlpha + (maxAlpha - minAlpha) * Math.min(1,
  speed/speedRef));
return { x: prev.x + dx*a, y: prev.y + dy*a, ... };
```

- 이상치 거부에 연속거부 한계(MAX_REJECT_FRAMES=8) → 시크/재등장 같은 진짜 큰 이동에 영구 고착 방지.
- 추가로 픽셀 히스테리시스(0.75px 이상일 때만 정수좌표 갱신)로 저속 반올림 깜빡임 제거(1048).

5.4 라벨 사전계산 + 겹침 억제

- **startLabelPrecompute** ([StationOverlay.tsx:692-746](#)): 전 프레임의 가시집합·겹침해소 결과를 Map<frameNum, LabelCache>에 사전계산. 화면좌표가 아니라 월드좌표(lat/lon/z)를 저장 → RAF가 매 프레임 연속포즈로 재투영. requestIdleCallback 200프레임 청크, 취소토큰(precomputeIdRef), 현재 프레임부터 순환 처리(재생 즉시 라벨), 첫 청크 즉시 노출, 재계산 시 옛 맵 복사.
- **겹침 억제** (755-784): 임계 가로 POI_MERGE_X=0.10 /세로 0.035 (비등방, 텍스트가 가로로 긴 특성). 우선순위 ①철도역/역사 최우선 ②근접거리. 진행방향 변형 우선: 상/하행 형제 겹침 시 노선 방향((상)/(하)) 변형을 남김.

5.5 RAF 렌더 루프 & 성능 설계

매 프레임 계산 없이 조회+투영+드로잉만. 라이브 시간은 VideoPlayer smoothTimeRef (단조 벽시계 보간, 일시정지·시크·배속 보정)에서 받음.

작업	비용	배치	빈도
가시성/겹침 판정	높음	precompute	1회(+설정변경 debounce), idle 청크
좌표 투영	중간	RAF	매 프레임

작업	비용	배치	빈도
평활/EMA/히스테리시스	낮음	RAF	매 프레임
팝업 가시집합 setState	리렌더	setInterval 150ms	변화 시만

핵심 = "무엇을 그릴지"(precompute) ↔ "어디에 그릴지"(RAF) 분리로 단일스레드 60fps 달성. 중심선/드론궤적은 `buildLines` 로 매 프레임 직접 투영(근거리 Zc 클리핑·outcode 켜림). 나침반은 별도 $\pm 3fr$ 평활 yaw + 360° 누적 언랩.

6. 측점(체이너지) 시스템 & 스테이션바

6.1 연속 체이너지 수직투영 (핵심 ①)

드론 한 프레임의 위경도를 **측점 폴리라인에 수직 투영**해 100m 양자화가 아닌 **10m 해상도 연속 측점값**을 얻는다(부산물: 선로 수직 이격 offsetM).

- 폴리라인 구축 `buildChainLine` ([chainage.ts:22-32](#)): 측점값 오름차순 정렬, 평면투영 $x=lon \cdot k, y=lat \cdot 111000$ ($k=\cos(lat0) \cdot 111000$).

- 점-선분 투영 `projectToChain` ([chainage.ts:35-50](#)):

```
const t = L2===0?0:Math.max(0,Math.min(1, ((px-a.x)*dx + (py-a.y)*dy)/
  L2));
const d = (px-cx)**2 + (py-cy)**2;
if (d < bestD) { bestD = d; bestKm = a.km + (b.km-a.km)*t; } // 최근접 선
                     분 위 km 보간
```

- **왜 연속 투영인가**: 최근접 측점점(100m 양자화)은 값이 계단처럼 튀고 두 측점 중간에서 전진/후진 전환 시점이 어긋난다. 수직투영은 선분 보간으로 매 프레임 부드럽게 변해 **방향 전환점(극값) 타이밍이 실제 주행과 일치**.

6.2 이중 측점값: 배지용 `km` vs 방향용 `chain`

프레임별 `precompute`([StationBar.tsx:233-246](#)): `km` =최근접 측점점(100m, 배지/폴백), `chain` =연속투영(10m, 방향/축/탐지). `videoFps` 는 실데이터(마지막 프레임/재생시간)에서 유도, 미비 시 `30000/1001` 폴백.

6.3 축 설계: 시간축이 아닌 위치축 (핵심 ②)

선로 점검 영상은 **호버(정지비행)** 와 **전·후진 반복** 특성이 있어 시간축이 부적합하다(호버 시 커서가 위치와 어긋남).

- **이동거리축** ([StationBar.tsx:250-271](#)): 연속 측점값을 $\pm 8fr$ 평활 후 프레임간 **절대 변화량** 누적 (`cum += |sm[i]-sm[i-1]|`)을 정규화

(`frac`, 단조 비감소). 호버=`frac` 정체(커서 정지), 이동=`frac` 증가. 방향은 색 리본으로 직교 분리.

- **측점축**(후속 채택안, [history 2026-06-26 1520](#)): x축을 **측점값 비례**로 두고, 축 범위를 역 POI 첫/끝으로 **노선 고정**(영상 무관 절대 측점) → 어떤 데이터로 재생하든 같은 구조물이 같은 위치/측점에 표시(회차 간 정렬).
- 시간↔축 양방향 변환: `cumFracAtTime` / `timeAtFrac` (또는 `chainAtTime` / `pxAtChain`) 를 이진탐색+보간으로. 커서·구간색·구조물 마커가 **동일 매핑 공유**. 라이브 커서는 React 렌더 없이 RAF에서 CSS 변수(`--pos-px`)만 갱신([771-786](#)).

6.4 방향 판정 & 방향색 트랙

평활 측점값 추세를 극값추적 + 히스테리시스 `HYST=10m`로 방향 전환점 확정([395-458](#)).

`forwardDir = arrChain≥depChain?1:-1`. 정방향=주행 음영, 역방향=청록. 이동거리/측점 축은 구간 `px`가 단조라 `linear-gradient` 1장으로 경량 처리([686-726](#)).

6.5 구조물 마커 — 좌표/측점 이중 탐지 & `kmExists` (핵심 ③)

두 통과 탐지기:

- `pxPassesTo(lat, lon, off)` ([489-534](#)) — 좌표 근접. 임계 `TH=max(100, gmin·3)`.
- `pxPassesAtMileage(targetM)` ([537-559](#)) — 연속 체이닝지가 `TH=stationTolerance`(기본 20m) 이내.

탐지 우선순위(단락평가): ①명시 측점값 ②좌표를 측점선 투영→측점기준(통과순서 정확) ③이름매칭 POI 투영 ④좌표근접 폴백 ... ([612-639](#)).

역사 합집합 보강: 종점/주차장은 측점선 투영이 불안정해 재진입을 놓침(예 대전주차장 162k080). 단락평가 탓에 한 통과를 찾으면 좌표근접이 안 돌아 누락 → **역사에 한해 좌표근접 통과를 합집합으로 합치되 24px 임계 중복제거**([629-634](#)).

`kmExists` (★): 좌표 반경만으로 잡힌 통과는 실제 측점값이 그 위치에 없을 수 있다.

```
const kmExists = stationKmVal == null || Math.abs(p.km - stationKmVal) <= existTol; // existTol = stationTolerance
```

([StationBar.tsx:648](#)) `false` 면 **측점값 라벨·측점검색에서 제외하되 마커(동그라미)는 유지** — "위치 표시"와 "측점 단정"의 분리.

종점 미도착: `endGapPx=min(폭·0.15, (gapM/lenM)·폭)`, `placeUnreachedTerminal` 이 최우측 역사를 트랙 끝에 미도착(속 빈 링)으로 고정([303-308](#), [574-586](#)).

6.6 측정검색 순환 — `handleJumpToMileage`

[StationBar.tsx:850-916](#). 측정값 입력 시 여러 통과를 **통과방향 순환**:

1. 후보 **1순위**: 화면 마커(`structureMarks`) 중 `kmExists!==false` 이고 표시 10m값 일치 → px→`timeAtFrac` 역변환. (좌표근접 통과 포함 → 모든 동그라미 도달)
2. **2순위**: 마커 없으면 chain 연속구간, 최종 전역 최근접 풀백.
3. 정렬 + 0.3s 중복제거 → baseT 기준 다음 통과(끝→처음 순환). `jumpRef` 로 연속검색 판정. Enter 후 입력값 유지, blur 시 삭제([PlaybackControls.tsx](#)).

6.7 타임라인 보조 기술 ([Timeline.tsx](#))

- **동일항목 브래킷 클러스터**: 동명 다통과를 거리 무관 한 묶음으로(드롭+수평 점선) 중앙 라벨 1개([143-172](#)).
- **라벨 2줄 균등분할** `splitStructLabel`: CJK 1.0/기타 0.55폭, 괄호그룹은 통째로 아랫줄([30-49](#)).
- **측점값 라벨 겹침 숨김** `showMileage`: 직전 표시와 `(lastW+w)/2` 미만이면 숨김, `kmExists===false` 는 애초 생략([117-137](#)).
- **아이콘 passed 전환**: upcoming/passed 두 PNG를 opacity로 전환(무깜빡임), 전 상태 사전로 드.

7. 위치보정 역투영 (단일 제스처) — 발명 C

7.1 무엇을 / 왜

지오코딩 POI는 ±수십 m 오차. 2D 화면 이동만으로는 카메라로부터의 거리(깊이)를 결정 못함 (깊이 모호성). 종래는 수평/높이를 따로 조정하거나 깊이맵(LiDAR) 요구. → **단일 드래그로 수평·수직(3D) 동시 보정, 깊이맵 불요.**

7.2 구성(독립항 필수요소)

1. **슬랜트거리 보존** — 드래그 개시 시점의 카메라-객체 슬랜트거리(range)를 고정 → 깊이 모호성 해소 핵심 구속.
2. **역투영** — 드롭 픽셀을 카메라 방향벡터로 환산 후 보존 거리만큼 투사.
3. **월드좌표 복원** — 자세 회전·평행이동 역변환으로 lat/lon/z 복원(지오이드 역환원). 한 번의 드래그로 수평·수직 동시.
4. **영속·재적용** — 복원 좌표를 객체 식별자(title)에 연계해 `<base>_poi_overrides.json` + `ghivideo:poiov:<baseName>` 저장.
5. (종속) **가시영역 한정** — 전방($Z_c > 0$)만 보정.

```

dir_cam ∝ ((px-0.5-cx0)·sW/f, (py-0.5-cy0)·sH/f, 1)
cam = dir_cam/|dir_cam| · range // ★거리 보존(드래그 시작값)
rel = R_w2cT · cam // 회전 역변환
z = droneAlt + offZ + rel.up - N_geoid // 정표고 복원
(lat,lon) = TM-1(droneE+offX+rel.x, droneN+offY+rel.y)

```

변형 — z 전용 역산 `solveZForPixelY`: lat/lon 고정, 드롭 세로위치로 z만 1차식 역산. 왕복 검증 $z=42 \rightarrow py \rightarrow 42.000$.

7.3 효과

2D 드래그 1회로 3D 보정(단계 절반), 역투영 왕복오차 ≈ 0 , 깊이맵 불요, 파일+localStorage 재현성. "depth map 없이 거리보존" 을 명시해 US9188444B2와 구별(\$10).

8. 백엔드 스트리밍/FFmpeg · 데이터 파이프라인 · 상태관리 · 플레이어

8.1 대용량(2GB+) 재생 — 이중 경로

- 로컬(`useVideoPlayer.ts:80-88`): `URL.createObjectURL` Blob URL 직접 재생(업로드/변환 없음), 교체 시 `revokeObjectURL` (누수 방지). **현 운영 주 경로.**
- 서버: Range Request 즉시 재생, 비표준 코덱(HEVC 등)이면 HLS 자동 전환([15-17](#)). (서버/HLS 경로는 보존되나 UI는 `SHOW_TOOLBAR=false` 로 숨김.)

8.2 Range Request ([streaming.ts:52-77](#))

핵심: 한 응답에 전체를 싣지 않음. 끝값 없으면 `start+10MB-1`. `pipeline()` (백프레셔/에러 자동), `highWaterMark:1MB`, `Accept-Ranges:bytes`, path traversal 방어 (`resolveVideoPath`). JS Number($2^{53}-1$)로 4GB+ 안전(BigInt 불요). 멀티레인지 미지원.

8.3 HLS 변환 + SSE ([hls.ts](#))

먹등(디스크에 `index.m3u8` 있으면 즉시 done), 코덱 분기(H.264 `-c copy` 수 초 / 그 외 libx264), `/progress` SSE로 stderr `time=` 파싱 푸시. hls.js 필수: `backBufferLength: 30` (기본 Infinity면 장시간 재생 메모리 누수→크래시), `maxBufferLength 30`, `enableWorker(useVideoPlayer.ts:7-13)`.

8.4 FFmpeg spawn 래퍼 ([ffmpeg.ts](#))

`-y` auto, stderr `time=HH:MM:SS.cc` 정규식 진행률, `exit≠0` 시 stderr 끝 500자 reject. 프레임 추출은 `-accurate_seek -ss {t} -i {f} -frames:v 1` (플래그 순서 = 입력단 정확 시깅, [frame.ts:37-44](#)).

8.5 데이터 파이프라인 (geoData.ts)

<input webkitdirectory> /드래그로 받은 File[] 을 병렬 파싱(Promise.all). 인코딩 자동감지(BOM→UTF-8/EUC-KR), 헤더명 우선+위치 폴백(makeFieldIndexer), 좌표 유효성 게이트(lat 33~39/lon 124~132), KMZ 우선(있으면 CSV POI/구조물 미사용, fflate unzipSync).

소스	파서	대상	매핑
<base>.csv	parseDroneFrames	DroneFrame[]	frame_cnt, lat/lon/alt, yaw/pitch/roll, focal
building/01)측점.csv	parseStations	station + DirectionChange	측점→title, Z좌표_한국(정표고) 우선, 비고→방향전환
<base> POI.csv / 02)지장물.csv	parsePois	poi	category_clean/명칭, lat/lon
02)지장물_역사.csv	parseHistoricStations	역사 구조물	명칭→name(스테이션바 마커)
03)교량/04)터널/06)구조교.csv	parseStructures	bridge/tunnel	구분/시설물명, 연장→lengthM, 시설종별→grade
05)출입문번호.csv	parseAccessDoors	출입문 poi	출입문번호→title
*.kmz/kml	parseKمز	poi+구조물	Placemark + description표→props
route.json	mergeStructures	보정	station/offset/이정 override

중심선은 v2.0에 별도 파일이 없어 측점을 측점값 순 정렬한 폴리라인으로 합성.

8.6 상태관리

- **geoStore**(메모리, persist 없음): frames/stations/pois/structures/centerline/origin/routeMeta. basePois (원본) vs pois (override 적용) 분리로 원본 불변성(geoStore.ts: 96-110).
- **settingsStore**(localStorage ghivideo.settings): 시설종별 필터, 표시 토글들. 커스텀 merge 로 스키마 진화 대비. isGradeVisible 공유 규칙.

8.7 플레이어 통합 & 프레임 정확도

- Video.js `data-vjs-player` + Strict Mode 가드, `controls:false` (측점바로 대체), `fill:true`.
- **FPS 감지**([useFrameStep.ts:21-49](#)): `requestVideoFrameCallback`, 첫 콜백은 시작점만 기록(31fps 오감지 방지), 연속 2회 일치 확정. 스텝은 `1/fps` (누적 금지=드리프트 방지).
- **이원 fps**: 스텝 이동=실측 fps, 프레임번호 표시=29.97 고정(SRT 기준). 의도된 분리.
- **매끄러운 커서**: RAF 단조 벽시계 보간 `smoothTimeRef` (매 프레임, transform) + `smoothTime` (throttle, 배지) 분리.

9. 개발 타임라인 (주요 기술 의사결정)

시기	핵심 결정/구현
2026-04-01	labelMap 사전계산 + requestIdleCallback + RAF, $\pm 10\text{fr}$ 이동평균(yaw sin/cos), 30→60fps 보간
2026-04-02	투영 파이프라인 정립 : EPSG:5186 TM, 핀홀(Rz·Rx·Ry), 타원체고=정표고+지오이드, Cohen-Sutherland 클리핑
2026-06-19	v1→v2 데이터 형상 전환 (실측 폴더), 인코딩 자동감지, 측점 비교→방향전환 추출, 시설종별 필터
2026-06-22	측점 표고 컬럼 교정(Z좌표_한국) → 부각 $42.7^\circ \rightarrow 11^\circ$ (발명 B). 역투영 드래그 보정+override(발명 C 최초). title-키 보간+이상치 거부. 비등방 거리필터
2026-06-23	에지보존 평활(YAW_EDGE=8°). 글로벌+KIPRIS 선행기술 조사 → B·C 신규성 확인
2026-06-24	1클릭 FOV 보정(→sensorH 역산) , object-fit:cover 정렬, 라벨 월드좌표 저장 +RAF 재투영, One-Euro EMA, KMZ 직접 로딩
2026-06-25~26	중점역 누락 수정, 양끝 역 고정, 라벨 2줄분할, 동일항목 브래킷, 거리축→측점축 전환
2026-06-29	좌표/측점 이중탐지+합집합 보강 , 측점값 라벨 겹침숨김, 측점검색 순환+마커기준 전환+kmExists , UI 통합(토글 재생바 이동, 활성색 amber), 보고서 3종 생성, 55000 외부접속

상세 근거: [docs/history/2026-06-22_POI투영오차-개선.md](#)(발명 B·C 1차 출처), [2026-06-26_1520_측점축-전환](#), [2026-06-29_1029_역사-통과-좌표근접-보강](#) 등.

10. 특허 관점 종합

출처: [docs/특허 스테이션기반 주행영상 플레이어.html](#)(기술발굴), [docs/발명신고서 측정표 고정합 역투영보정.html](#)(변리사 제출용).

10.1 기술요소 인벤토리 (E1~E9)

코드	요소	특허 적합성
E1	측점(체이너지) 기반 영상 색인·탐색	후보 A(좁게)
E2	지리객체 실시간 영상 정합(labelMap 사전계산)	공지(큰 틀)
E3	측점 표고 기반 DEM-free 지면고도 추정	후보 B(상)
E4	역투영+슬랜트거리 보존 단일드래그 위치보정	후보 C(상)
E5	비등방 거리필터+소실점 클러터 억제	후보 D(중속)
E6	노이즈 강건 오버레이(이중평활·title짜·이상치거부)	후보 E(중속)
E7	측점 폴리라인→중심선	제외(통상 처리)
E8	이종 CSV 융합(인코딩 자동감지)	제외
E9	비고 텍스트 방향전환 추출	제외

10.2 출원 권고 우선순위

- 1순위: 발명 B + C 결합 — 신규성/진보성 우수, 글로벌+한국 신규성 확인.
- 2순위: 후보 A(좁게) — LRS 비선형 재좌표화 + 회차간 위치앵커 + 다중회차 동기. 한국 선행기술(KR102268318B1·KR100411587B1) 위험 → 좁게 청구. 본 명세의 측정축 노선좌표 고정(\$6.3)·측점검색 순환(\$6.6)·이중탐지 kmExists(\$6.5)가 A의 구현 근거.
- 3순위: D·E 종속항 흡수. 제외: E7·E8·E9.

10.3 청구 포인트 핵심 (B·C)

- 발명 B 독립항: ①측점 실측 정표고 수열 → ②표고미상 대상에 최근점 측정 정표고를 지면고도 채택(노선기복 반영) → ③지오이드고 가산으로 타원체고 변환·datum 정합 → ④정합투영. ※ ②+③ 결합을 한 청구항에(개별요소는 공지).
- 발명 C 독립항: ①드래그 개시 시 슬랜트거리 보존 → ②드롭픽셀 역투영 → ③단일 제스처로 수평·수직 동시 복원 → ④식별자 연계 영속. ※ "depth map 없이 거리보존" 명시(US9188444B2 구별).

10.4 선행기술 (조사 완료분)

- E1-E2 큰 틀은 공지: Esri ArcGIS Pro FMV, US9996976B2, Mandli Roadview, Remote GeoSystems LineVision, ENSCO Virtual Track Walk.
- B·C는 글로벌+한국 신규성 확인. 단, KIPRIS 전문(專門)검색 미수행 — 출원 전 필수.

10.5 신규성 후보 종합 (구현 기반)

1. 드론 SRT pose 기반 실시간 GIS-영상 정합 파이프라인(타원체고/정표고 datum 자동 정합 포함).
2. 단일 대응점 sensorH 역산 인터랙티브 FOV 보정(가로 정합 보존, 중심부 발산 거부).
3. 깊이 모호성 분리형 역투영 3종(슬랜트유지/지면교차/표고 1차역산)을 보정 의도별 선택.
4. 에지(회전경계) 보존 적응형 평활 + 속도적응 EMA(연속거부 한계).
5. 월드좌표 precompute + 연속포즈 재투영 2단 분리로 단일스레드 60fps AR.
6. 누적 이동거리/측점 비례 축 + 방향색 직교 분리 진행바, 노선좌표 고정(회차 정렬).
7. 좌표/측점 이중 탐지 + 합집합 보강 + kmExists 신뢰도 플래그("위치 표시"와 "측점 단정" 분리).

11. 검증 결과표

항목	개선 전	개선 후	근거
부각 오차(측점 157K900, 91.3m)	42.7°	11.0°	\$4.3
지면고도 정확도(측점 vs 구글 DEM)	—	0.2m(~6px@2160p)	\$4.3
역투영 왕복오차 (z=42→py→z)	—	≈0 (42.000)	\$7.2
화면 EMA 지연	1.65s(α=0.01)	~25ms(α=0.4)	\$5.3 / history 06-22
렌더	끊김	60fps(precompute+RAF 분리)	\$5.5
역 재진입 마커	누락	좌표근접 합집합 보강으로 표출	\$6.5

지오이드고 N: 본 명세 25.8m(KNGeoid 검증) 통일. 초기 문서값 25.449m는 각주 처리.

12. 후임자 인계

12.1 데이터 흐름 한눈에

geoStore(frames/stations/structures/routeMeta) → `useEffect` `precompute`(`viewedRef` = 프레임별 km/chain/time, `chainRef` =time/frac/chain) → `pxAtTime` / `pxAtChain` · `barSegments` · `structureMarks` → Timeline 렌더. 모든 `px`는 단일 매핑 (`pxAtTime` / `pxAtChain`)을 통과, 역변환은 `timeAtFrac` / `chainAtTime` 만 사용.

12.2 튜닝 포인트

- 폴더 `route.json`: `stationTolerance` (통과/검색 허용오차, 20m), `direction` / `directionChanges` (방향 변형 필터), `endStationGapMeters` / `lengthKm` (미도착 폭), `startStationName` / `endStationName`.
- 카메라(패널, 데이터셋별 저장): `focalLen`, `sensorW`, `sensorH` (FOV), `yawOffset` / `pitch` / `roll`, `geoidOffset` (지역 N), `poiZOffset`.
- 코드 상수: 평활 `W=8`, `YAW_EDGE=8°`, `HYST=10m`, 좌표통과 `TH=max(100,gmin·3)`, 역사 합집합 `24px`, 검색 중복 `0.3s`, EMA `β=0.25` / `REJECT_DIST=0.12` / `MAX_REJECT_FRAMES=8`.

12.3 빌드/배포 (★ 필수 숙지)

55000은 PM2 `ghiVideo` 가 `client/dist` (빌드 결과)를 정적 서빙한다. 소스만 고치면 화면에 반영 안 됨.

```
export PATH="$HOME/.nvm/versions/node/v20.20.2/bin:$PATH" # 시스템 node v12
→ tsc/vite 실패
npm run build -w client # = tsc && vite build → client/dist 갱신
# 브라우저 Ctrl+Shift+R, 폴더 데이터는 런타임 로드라 폴더 재선택 필요
```

외부(LAN) 접속: netsh portproxy(Windows :55000 → WSL IP:55000) + 방화벽. WSL IP는 재부팅 시 변동.

12.4 업그레이드 포인트

1. 렌즈 왜곡(Brown-Conrady) 추가 → 가장자리 정합.
2. 다점 대응 자동 캘리브레이션(focal·주점·왜곡 번들조정).
3. 지오이드 격자(KNGeoid18) 보간으로 `geoidOffset` 자동화.
4. POI 표고 DEM 자동조회(현 부분지원).
5. 중심선 per-frame 투영 공간색인, `precompute` Web Worker화.
6. `frame.ts` fps 고정(30)→ `getVideoFps`, HLS jobs DB 큐, 멀티 Range 지원.

7. 코드 통합: 측점 파서/포매터(chainage.ts ↔ StationBar.tsx), 투영함수(`projectToChain` ↔ `projectChainage`) 단일화.
-

13. 한계 / 출원 전 확인사항

- KIPRIS 전문검색 미수행 — 출원 전 필수.
 - 지오코딩 수평오차는 데이터 한계(코드 보정은 드래그 수동).
 - `geoidOffset` 지역 상수(대전 25.8m) — 타 지역 재튜닝.
 - VFR/abs_alt 노이즈, 먼 거리 역투영 민감.
 - 미확정 항목(특허문서 `[확인 필요]` 유지): 다중회차 동기화 구현 범위, 위치앵커 주석 데이터모델, 권리귀속, 공개이력.
 - 렌즈 왜곡 미보정(핀홀만) — 가장자리 정확도 트레이드오프.
-

본 문서는 5개 서브시스템(카메라 투영/자세 평활/측점·스테이션바/백엔드·데이터/기존 특허문서) 정밀 조사 결과를 종합한 것이다. 각 소스 참조는 클릭 가능한 파일·라인으로 추적 가능하며, 특허 출원 시 본 문서의 §3~§7·§10과 history 원문을 함께 변리사에게 제공할 것을 권장한다.