UCLouvain

École polytechnique de Louvain

Video Alignment for Predictive Maintenance in Railway Infrastructure

A Hybrid Coarse-to-Fine Pipeline Combining Dynamic Time Warping and Binary Feature Matching

Author: **Julian Gosselin**
Supervisor: **Pierre Schaus**
Readers: **Ioannis Kostis**, **Augustin Crespin**, **Harold Kiossou**, **Sébastien Jodogne**
Academic year 2025–2026
Master [120] in Computer Science

Acknowledgments

I would like to express my sincere gratitude to all the individuals who contributed to the realization of this master's thesis.

First and foremost, I wish to thank my supervisor, **Professor Pierre Schaus**, for giving me the opportunity to work on this fascinating subject and for the trust he placed in me.

I am deeply grateful to **Ioannis Kostis** and **Augustin Crespin** for their invaluable guidance. They dedicated a significant amount of their time to me, and I thank them for their constant availability, the many meetings we held, and the precious advice they provided throughout the year. I am also honored by their participation as members of my jury.

I would like to express my sincere gratitude to Professor **Sébastien Jodogne** and Mr. **Harold Kiossou** for accepting to be the readers of this thesis. Their valuable time, expertise, and constructive feedback are greatly appreciated.

Finally, a huge thank you to my **family and friends**. Their unwavering moral support and constant encouragement have been a true pillar for me during this year.

Abstract

Monitoring railway infrastructure is a critical challenge for safety and maintenance, particularly in low-income countries where budget constraints make traditional inspection methods impractical. This thesis is carried out in collaboration with an industrial partner whose long-term goal is to develop an affordable predictive maintenance solution for railway networks based on on-board camera footage.

Within this broader initiative, this work focuses on a core sub-problem: the automated temporal synchronization of two rail-track videos recorded along the same track at different times and at different speeds. Solving this alignment problem is a prerequisite for any subsequent change detection or defect identification task.

The proposed solution follows a coarse-to-fine hybrid approach. A global alignment skeleton is first established using Open-End Dynamic Time Warping (DTW) applied to frame-level appearance signatures, encoded as gradient-orientation histograms. This skeleton is then refined through a local feature-matching stage using binary descriptor algorithms (AKAZE, BRISK, ORB), with each match constrained to a narrow temporal window around the DTW prediction to prevent drift. A global optimization layer enforces temporal monotonicity via dynamic programming, and a weighted smoothing pass eliminates residual jitter.

The system is evaluated on three real-world railway sequences of approximately 90 seconds each, using three complementary metrics: Structural Similarity (SSIM), Learned Perceptual Image Patch Similarity (LPIPS), and a cycle-consistency score. All three algorithms produce strictly monotonic alignments with low cycle-consistency error, remaining stable even in the presence of repetitive visual patterns such as regularly spaced poles and track structures. A central finding of this work is that SSIM and LPIPS saturate on this type of footage – they barely distinguish a correct alignment from a random one – whereas cycle consistency separates the two by up to two orders of magnitude, making it the reliable metric for evaluating this task.

The source code and full implementation for this pipeline are publicly available on GitHub: `https://github.com/JulianGsl/TSV`.

By providing a reliable and fully automated temporal alignment pipeline, this work constitutes a foundational building block towards a low-cost, scalable infrastructure monitoring system deployable in resource-constrained railway environments.

Declaration on the Use of Artificial Intelligence and Generative Tools

In accordance with academic integrity guidelines, I declare that I used large language models and generative artificial intelligence (specifically Gemini & Claude) as assistive tools during the research and drafting phases of this master's thesis.

I integrated these tools into my workflow to complement my work in the following capacities:

- **Continuous Code Documentation:** I used AI assistants throughout the development and testing phases to systematically document my code, comments, and functions. This approach allowed me to preserve and organize a substantial volume of technical insights and engineering decisions.
- **Academic Writing and Language Refinement:** Since English is not my native language, I utilized the AI as a linguistic editor to restructure sentences, correct grammatical nuances, and elevate the overall stylistic flow to match academic standards.
- **Conceptual Inquiries and Content Expansion:** I employed the AI as an interactive brainstorming partner, prompting myself with challenging questions and stylistic suggestions that helped me identify gaps, expand on technical details, and increase the comprehensiveness of the manuscript.

Abbreviations

SIFT	Scale-Invariant Feature Transform
SURF	Speeded-Up Robust Features
HOG	Histogram of Oriented Gradients
KAZE	Japanese for “wind” — base algorithm for AKAZE
AKAZE	Accelerated-KAZE
BRISK	Binary Robust Invariant Scalable Keypoints
ORB	Oriented FAST and Rotated BRIEF
KNN	K-Nearest Neighbors
RANSAC	Random Sample Consensus
DTW	Dynamic Time Warping
DP	Dynamic Programming
LUT	Look-Up Table
MSE	Mean Squared Error
PSNR	Peak Signal-to-Noise Ratio
SSIM	Structural Similarity Index Measure
MSSIM	Mean Structural Similarity Index Measure
LPIPS	Learned Perceptual Image Patch Similarity
BAPPS	Berkeley-Adobe Perceptual Patch Similarity (dataset)
MAD	Median Absolute Deviation
CDF	Cumulative Distribution Function
CIE	Commission Internationale de l’Éclairage (colour space)
PCA	Principal Component Analysis
CNN	Convolutional Neural Network
VGG	Visual Geometry Group (network family)
RL	Reinforcement Learning
FPS	Frames Per Second
GPS	Global Positioning System
ROI	Region of Interest
CSV	Comma-Separated Values

List of Figures

List of Tables

Contents

1. Introduction

Railway operators must continuously balance the cost of inspecting their network against the safety risk of missing a defect on rails, sleepers, ballast or fasteners [2, 3, 4]. Traditional inspection relies on walking inspectors or on dedicated track recording vehicles [5, 6]. Both approaches scale poorly: doubling the inspected length doubles the cost, and dedicated vehicles compete with commercial traffic for scarce night-time slots [7, 8]. Operators on tight budgets therefore tend to under-inspect, potentially increasing operational safety risks [9].

A growing alternative is to instrument commercial in-service trains with forward-facing cameras, turning every regular run into an inspection pass at near-zero marginal cost [7, 10, 11]. Mounting a single camera and a recording unit on the cab of an existing locomotive costs a small fraction of running a dedicated inspection vehicle, and the resulting inspection cadence is governed by commercial traffic rather than by maintenance scheduling. If the resulting footage can be compared automatically against earlier passes to flag changes, the inspection programme becomes denser, safer for track workers, and affordable for networks that cannot fund dedicated equipment [12].

This thesis contributes to a broader four-stage system: (1) acquisition by in-service trains, (2) temporal synchronisation of a reference run V_1 with the current run V_2, (3) change detection on aligned frame pairs, and (4) interpretation for maintenance scheduling. **The technical contribution presented here is limited to Stage 2.** Although the project is framed as predictive maintenance, no defect detector or scheduler is delivered; what is delivered is the algorithmic substrate on which the other stages would rely.

The temporal alignment problem itself is non-trivial. Trains accelerate, decelerate and stop at signals, so two passes of the same track have different durations and a non-linear, monotonically increasing time mapping between them [13]. This mapping must be recovered from visual content alone, because GPS, timestamps and odometry may be degraded or unavailable in deployment [14]. Beyond speed variation, the two passes also differ photometrically (sunlight, cloud cover, time of day), spatially (camera mounting tolerances, suspension dynamics), and semantically (passing trains, vegetation growth, graffiti). The alignment must therefore be robust to all three categories of nuisance variability while still recovering a precise frame-to-frame correspondence, and repetitive scene content such as long stretches of homogeneous ballast or tunnel sections makes local matching locally ambiguous.

The problem is also adversarial to off-the-shelf deep-learning solutions. End-to-end video alignment networks typically require a labelled corpus of aligned pairs, which is precisely the artefact the pipeline is supposed to produce; pretrained vision foundation models, in turn, are not trained on rail-track footage and require a GPU at inference time, which contradicts the cost-conscious deployment scenario this work targets. A misalignment of even a few frames is then enough to make every downstream pixel-difference flag a false positive, which makes the precision of Stage 2 the single critical multiplier of the entire pipeline [13].

1.2 Objective

The objective of this thesis is to design, implement, evaluate and document an automated pipeline that temporally aligns two videos recorded along the same railway track at different times and different speeds. The pipeline ingests two video files and returns a frame-by-frame correspondence list mapping every V_1 frame to a V_2 frame. It must run on commodity CPU hardware, with no GPU dependency at deployment time.

The deliverable is twofold: an open-source Python implementation that runs out of the box on standard environments, and an evaluation methodology that calibrates structural, perceptual and self-supervised metrics against a panel of baselines.

Several scopes are deliberately excluded: change detection itself, the predictive component that converts changes into maintenance decisions, on-board embedded deployment, and comparisons with deep-learning methods requiring a GPU at inference time. The last exclusion is not cosmetic: relying on specialised hardware would contradict the low-cost premise of the project. The rationale is detailed in Chapter 4.

The remaining chapters are organised accordingly. Chapter 2 introduces the relevant background. Chapter 3 positions the work in the literature. Chapter 4 presents the six-phase pipeline. Chapter 5 reports the experimental results. Chapter 6 describes the diagnostic outputs. Chapter 7 concludes.

1.3 Expected Contributions

1. **A hybrid coarse-to-fine alignment pipeline.** Six phases combining a globally robust DTW skeleton with a locally precise feature-matching refinement. The strict invariant that the refinement window is *never* widened on failure is the principled mechanism by which the pipeline avoids drift accumulation.

2. **An algorithm-agnostic inner matcher.** AKAZE, BRISK and ORB are interchangeable via a one-line configuration switch. The empirical finding (Chapter 5) is that the three perform within statistical noise of each other on the metrics that meaningfully discriminate quality, an unexpected result with implications for engineering choices in similar pipelines.

3. **A baseline-calibrated evaluation methodology.** SSIM, LPIPS and a self-supervised cycle-consistency score are systematically calibrated against four baselines. SSIM and LPIPS are saturated by the photometric ceiling between V_1 and V_2 and cannot tell real from random; cycle consistency varies by three orders of magnitude and is the only metric that meaningfully tracks quality. The methodological recommendation is that future work on this type of data should not report SSIM or LPIPS in isolation.

4. **A full-fidelity open-source implementation.** A Python codebase (OpenCV, NumPy, SciPy, Matplotlib, optional PyTorch for LPIPS) producing, for every input pair, a self-contained HTML report with cost-matrix heat maps, scatter and velocity plots, side-by-side aligned videos in three modes, and a frame-by-frame CSV.

1.4 Research Questions

The work presented in this thesis is structured around four research questions. Each question is operationalised in the chapters that follow, and answered, with appropriate caveats, in the experimental and concluding chapters.

RQ1: Can a coarse-to-fine hybrid pipeline reliably align two railway videos with variable speeds, repetitive content and lighting differences, while running entirely on commodity CPU hardware? This is the central research question. It combines the algorithmic question (does the pipeline work?) with the deployment question (does it work without a GPU?). The answer, presented in Chapter 5, is affirmative: on three real-world rail-track sequences, the pipeline produces alignments whose monotonicity is exactly one hundred percent, whose mean round-trip cycle error is bounded by twenty-one frames on the limit-case plan, and whose worst-case correction is bounded by the design of the strict refinement window. The answer comes with caveats that are explored in Chapter 5 and Chapter 7.

RQ2: How does the choice of inner feature matching algorithm (AKAZE, BRISK or ORB) affect the quality of the final alignment, and which algorithm is best for railway video? This question motivates the comparison study reported in Chapter 5. Three algorithms are evaluated on identical data, with identical surrounding pipeline logic, on identical metrics. The empirical answer is that the three perform within statistical noise of each other on perceptual metrics, with ORB exhibiting a marginal advantage on the easier plans. The deeper finding is that the choice of inner descriptor has less impact than expected, because the dynamic time warping skeleton dominates the quality of the final result.

RQ3: Which evaluation metrics meaningfully discriminate alignment quality on rail-track footage, and which fail to do so? This question motivates the baseline

calibration study. As reported in Chapter 5, SSIM and LPIPS fail to discriminate the real alignment from a uniform random alignment, even after photometric normalisation. Both metrics are dominated by the inherent photometric ceiling between two recordings made on different days. The cycle consistency metric, by contrast, varies by three orders of magnitude between random and real, and is the only metric that meaningfully tracks alignment quality.

RQ4: How does the pipeline degrade on hard inputs, and what does this tell us about the limits of the design? This question is answered through the comparison of the three plans in the dataset, which exhibit different levels of content repetitiveness. On the easiest plan, between eighty-three and ninety-two percent of the frames are refined by feature matching, depending on the matcher. On the hardest plan, the refined fraction drops to between fifty-four and seventy-one percent, with the remainder falling back to the dynamic time warping prediction. Crucially, monotonicity remains at one hundred percent on all three plans, and cycle consistency degrades only modestly. This degradation pattern confirms the design intuition: the dynamic time warping skeleton is the safety net, and the pipeline degrades gracefully when feature matching cannot disambiguate among locally similar frames.

2. Technical Background

This chapter introduces the technical concepts used in Chapter 4. The treatment is bottom up, from the generic image registration problem to the four pillars on which the hybrid approach rests: local feature detection and description, feature matching, dynamic time warping, and image similarity metrics.

2.1 Image Registration and Temporal Alignment

Image registration is the geometric alignment of two or more views of the same scene, acquired at different times, viewpoints, or with different sensors [15]. A feature-based system proceeds in four steps [16]: *keypoints* are detected in both images; a local descriptor is computed at each keypoint, summarising the surrounding patch into a compact, ideally invariant vector; descriptors are matched across the two images to produce candidate correspondences; finally, a geometric model is fitted while rejecting outliers [17]. The model depends on the application: translation, affine warp, or full planar homography [18]. Figure 2.1 illustrates these four steps on a Plan 1 frame pair.

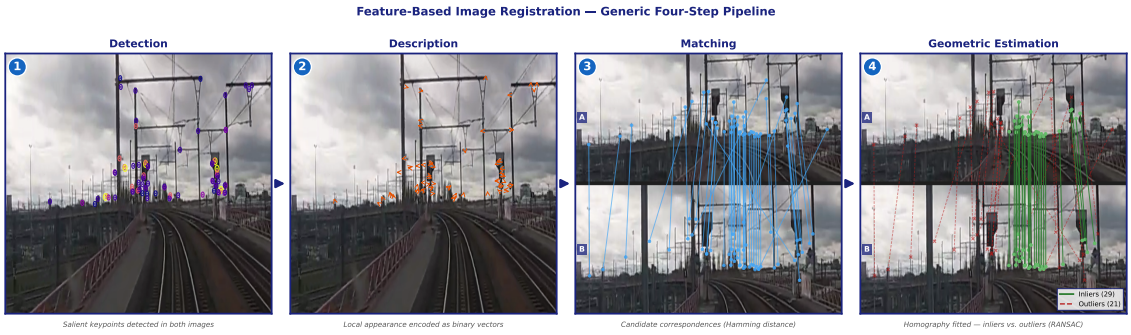

Figure 2.1: Generic four-step pipeline for feature-based image registration applied to the Plan 1 railway sequence ($\Delta t = 1.5$ s). The process includes keypoint detection, descriptor extraction, feature matching, and geometric verification. Inliers are shown in green and discarded outliers in red.

The present thesis tackles the related but distinct problem of *temporal alignment* of two video sequences acquired along the identical trajectory at different times [13]. We seek a mapping

$$f : \{0, 1, \ldots, N_1 - 1\} \to \{0, 1, \ldots, N_2 - 1\} \tag{2.1}$$

that is monotonically non-decreasing and that links each frame index i of the reference video V_1 to the index $f(i)$ of the frame in the target video V_2 depicting the same physical location along the track. Monotonicity reflects that the train moves forward in both recordings.

Several specificities make this harder than its spatial cousin. Trains travel at different, non-uniform speeds, so $i \mapsto f(i)$ is non-linear. Lighting shifts between recording days. Camera mounting is never identical from one run to the next, introducing small pitch and zoom differences. Trackside content is highly repetitive: sleepers, ballast and fasteners look essentially identical over kilometres. None of these difficulties appear in single-image registration, and they motivate the hybrid design of Chapter 4. Figure 2.2 contrasts the two settings side by side.

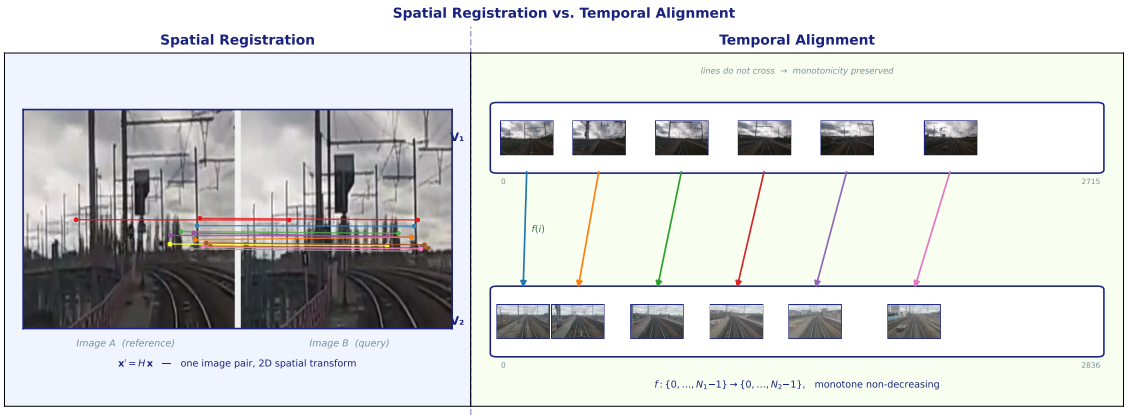

Figure 2.2: Spatial registration (left) aligns a single image pair via a 2D homography H. Temporal alignment (right) finds a monotone mapping $f : \{0, \ldots, N_1-1\} \to \{0, \ldots, N_2-1\}$ between two video sequences; arrows connect corresponding frames and their non-vertical inclination reflects the variable speed difference between the two recordings.

2.2 Local Feature Detection and Description

2.2.1 Keypoints and Descriptors

A *keypoint* is a location that the detector judges salient and repeatable [19]. Saliency means the point stands out from its neighbourhood, typically a corner, blob, or region of high local contrast [20]. Repeatability means the same physical point is detected again in another image of the same scene despite changes in viewpoint, scale, rotation, or illumination [21]. A *descriptor* is a fixed-length vector summarising the patch around a keypoint [22]; two descriptors of the same physical location should be close in descriptor space even after geometric or photometric transformations.

Descriptors come in two families, compared side by side in Table 2.1. Floating-point descriptors are real-valued vectors compared by Euclidean distance [20]. Binary descriptors are bit strings compared by Hamming distance, which reduces to `XOR`+popcount, both

native CPU instructions [23, 19]. Binary descriptors are typically one to two orders of magnitude faster to match [24], at the cost of slightly less robustness on extreme transformations. For a CPU-only pipeline the speed advantage is decisive. A typical AKAZE keypoint distribution on a railway frame is shown in Figure 2.3.

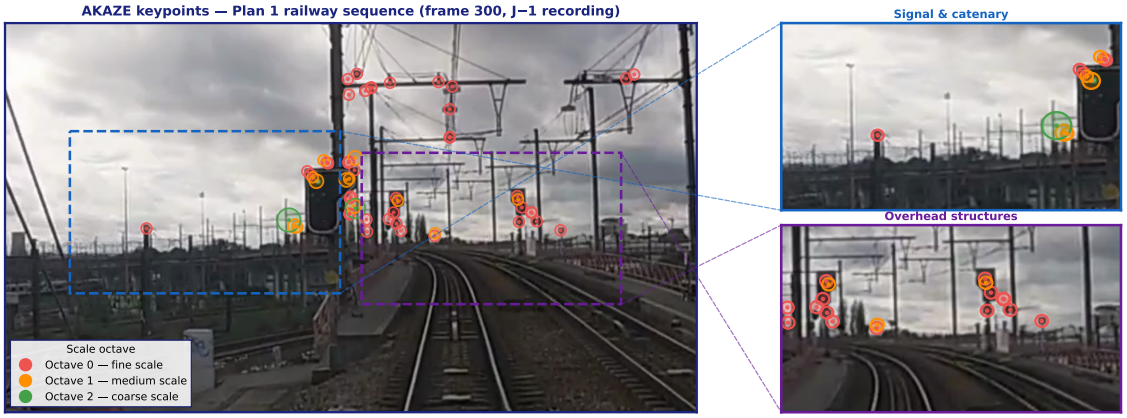

Figure 2.3: AKAZE keypoints detected on a typical railway frame (Plan 1, J–1 recording). Circles are colour-coded by scale octave: fine-scale keypoints (octave 0, red) concentrate on catenary poles and metallic fixtures; medium-scale keypoints (octave 1, orange) appear on broader structures. The track bed (right inset) yields few keypoints, reflecting the perceptual repetitiveness that motivates the hybrid approach of Chapter 4.

Table 2.1: Comparison of floating-point and binary local descriptor families. Relative matching speed measured for the descriptor comparison step alone; binary descriptors benefit from native `XOR`+popcount instructions.

Property	**Floating-point** (e.g. SIFT, SURF)	**Binary** (ORB, BRISK, AKAZE)
Encoding	`float32` vector	bit string
Typical size	256–512 B	32–64 B
Distance metric	Euclidean (L_2)	Hamming
Matching op.	multiply-accumulate	`XOR` + popcount
Relative speed	1× (reference)	$\geq 50\times$
Scale invariance	✓	✓
Rotation invariance	✓	✓
Robustness	very high	moderate–high
Used in this work	no	yes (ORB, BRISK, AKAZE)

2.2.2 Classical Methods: SIFT and SURF

Modern feature detectors trace their lineage to two seminal works. SIFT (Scale-Invariant Feature Transform) by Lowe [25] popularised multi-scale, oriented features whose descriptor is invariant to scale and rotation. The detector relies on differences of Gaussians [20]:

$$D(x, y, \sigma) = L(x, y, k\sigma) - L(x, y, \sigma), \tag{2.2}$$

where $L(x, y, \sigma) = G(x, y, \sigma) * I(x, y)$ is the convolution with a Gaussian of standard deviation σ. Local extrema of D in (x, y, σ) are keypoint candidates; the descriptor is a 128-dim vector of weighted gradient orientations [21]. SURF [26] keeps the same spirit but speeds up detection with box-filter approximations evaluated in constant time via integral images.

Both produce floating-point descriptors, which makes matching expensive on a CPU [22]. They are therefore not retained in Chapter 4; we mention them as the historical reference against which newer detectors are compared.

2.2.3 Binary Descriptors: ORB, BRISK, AKAZE

The thesis retains three binary-descriptor algorithms, all available in the OpenCV implementation [20]. They share the same matching infrastructure (Hamming distance, brute-force matcher with k-nearest-neighbour search) but differ in how they detect and describe keypoints.

ORB. Oriented FAST and Rotated BRIEF [24] combines the FAST corner detector [27] with the BRIEF descriptor [28]. FAST classifies a pixel as a corner if at least n contiguous pixels on a circle of radius 3 are all brighter (or all darker) than the centre by a threshold t. ORB augments this with a Harris-response score, an image pyramid for scale invariance, and an orientation from the intensity centroid:

$$\theta = \arctan_2(m_{01}, m_{10}), \quad m_{pq} = \sum_{x,y} x^p y^q I(x, y). \tag{2.3}$$

The descriptor is a 256-bit BRIEF vector rotated by θ [24, 21].

BRISK. The Binary Robust Invariant Scalable Keypoints algorithm [19] uses AGAST [29] as a faster successor to FAST. Its 512-bit descriptor is built from a 60-point circular sampling pattern: short-distance pairs feed the descriptor, long-distance pairs estimate the patch orientation. This gives BRISK a stronger scale-space treatment than ORB.

Figure 2.4 shows side-by-side keypoint maps for the three detectors on an identical railway frame, and Table 2.2 summarises their technical characteristics with the OpenCV parameters used in the production pipeline.

AKAZE. Accelerated-KAZE [30] departs from the Gaussian scale space used above. It builds a non-linear scale space by solving the diffusion equation

$$\frac{\partial L}{\partial t} = \operatorname{div}\big(c(x, y, t)\, \nabla L\big), \tag{2.4}$$

with conductance c depending on the local gradient through a Perona–Malik function such as $c = g(|\nabla L_\sigma|)$, $g(s) = \exp(-s^2/k^2)$. Diffusion is slowed across image edges and accelerated within homogeneous regions, which preserves contours much better than a

Gaussian blur of equivalent extent. AKAZE solves the equation efficiently via Fast Explicit Diffusion and produces a Modified-Local Difference Binary descriptor at each keypoint. Rails, poles and overhead lines are precisely the kind of edge structure AKAZE preserves well, which is why it gives the most stable keypoints on our dataset (Chapter 5).

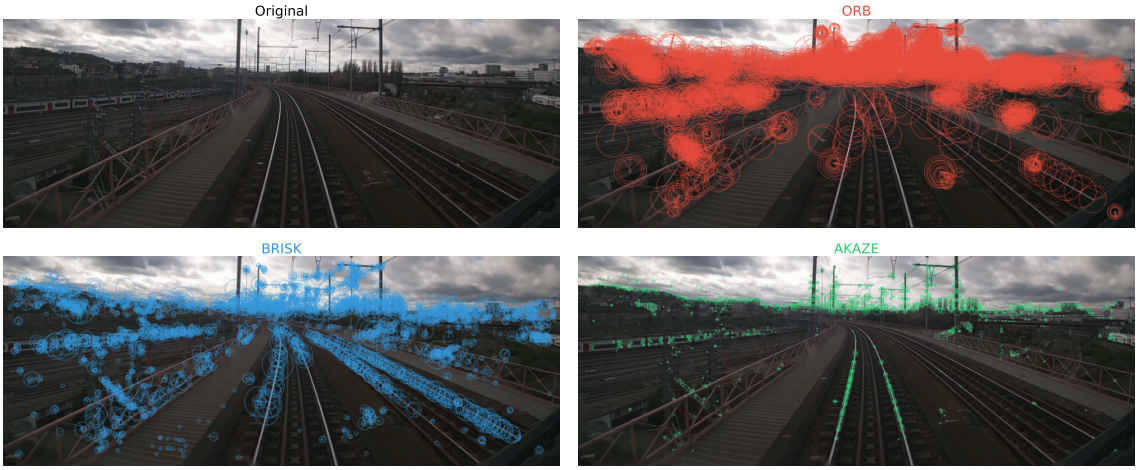

Figure 2.4: Keypoints detected by ORB, BRISK and AKAZE on the same rail frame, with the production pipeline parameters and the sky band cropped. Rich keypoints encode scale and orientation. ORB (4979) and BRISK (3796) saturate the textured ballast, whereas AKAZE (1497) is sparser and latches onto the structural edges (rails, poles, sleepers) that the alignment relies on.

Table 2.2: Technical summary of the three retained binary-descriptor algorithms, with the OpenCV parameters used in the production pipeline. Descriptor bit-lengths are measured from the actual OpenCV descriptor matrices.

	ORB	**BRISK**	**AKAZE**
Detector	oriented FAST + Harris score	AGAST	non-linear diffusion response
Scale space	Gaussian image pyramid	Gaussian scale-space pyramid	non-linear (Fast Explicit Diffusion)
Descriptor	rotated BRIEF (rBRIEF)	sampling-pattern binary	Modified-Local Difference Binary
Descriptor length	256 bits (32 B)	512 bits (64 B)	488 bits (61 B)
Distance metric	Hamming	Hamming	Hamming
OpenCV parameters	`nfeatures=5000`, `scaleFactor=1.2`, `nlevels=8`	`thresh=30`, `octaves=4`, `patternScale=1.0`	`descriptor_type=MLDB`, `descriptor_size=0`, `threshold=0.001`
Keypoints on Fig. 2.4	4 979	3 796	1 497

2.2.4 Other Methods Considered

An initial literature review covered fifteen detectors: historical methods (Harris, FAST), classical floating-point descriptors (SIFT, SURF), binary descriptors (BRIEF, FREAK), and deep-learning approaches (TILDE, LIFT, SuperPoint, D2-Net, R2D2, DISK, ALIKE) [21, 20]. Twelve were eliminated before implementation; the selection criteria are detailed

in Section 4.2 and Appendix A. The three retained algorithms span the design space of practical CPU-only binary methods.

2.3 Feature Matching

Once descriptors are computed in both images they must be paired. The pipeline follows the three-step recipe standard since the SIFT era: brute-force search with Hamming distance, Lowe's ratio test for ambiguity rejection, and RANSAC for geometric outlier filtering [15].

2.3.1 Brute-Force Matching with Hamming Distance

For each descriptor $\mathbf{d}_i$ in the first image, the brute-force matcher computes the Hamming distance to every $\mathbf{e}_j$ in the second image and returns the minimum-distance pair [15]. For binary vectors,

$$d_H(\mathbf{d}, \mathbf{e}) = \text{popcount}(\mathbf{d} \oplus \mathbf{e}), \tag{2.5}$$

implemented with two native CPU instructions per byte. Complexity is $O(NM)$ with a small constant; matching two frames with ~ 1000 keypoints each costs a few milliseconds on a modern laptop CPU.

2.3.2 KNN Matching and Lowe's Ratio Test

Naive nearest-neighbour matching is brittle: it always returns a match, even when no good correspondence exists. On a repetitive scene like a railway track, this produces many wrong matches between visually similar but geometrically unrelated points [10].
The standard mitigation is Lowe's ratio test [25]. The matcher returns the two nearest neighbours with distances $d_1 \leq d_2$, and a match is accepted only if

$$d_1 < r \cdot d_2, \tag{2.6}$$

for a threshold r typically in $[0.6, 0.8]$. We use $r = 0.75$ from the original SIFT paper. An unambiguous match is significantly closer to its first neighbour than to its second; an ambiguous match has $d_1 \approx d_2$ and is rejected. Figure 2.5 illustrates the geometric intuition behind the test.

2.3.3 RANSAC and Homography Estimation

After the ratio test, surviving matches are a mixture of inliers (geometrically consistent with the true camera transformation) and outliers (wrong matches that passed the test) [15]. A geometric model is fitted as a final filter. For two views of an approximately planar scene or related by a rotation around the camera centre, the appropriate model is a

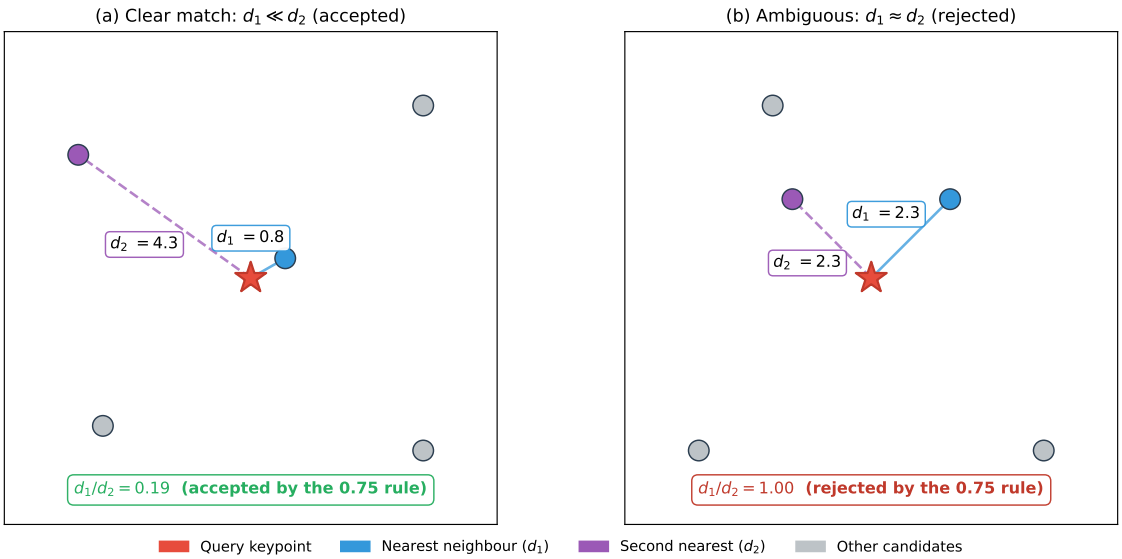

Figure 2.5: Illustration of Lowe's ratio test on a query keypoint (red star). (a) Clear match: nearest neighbour at $d_1 = 0.8$ (blue), second-nearest at $d_2 = 4.3$ (purple), ratio 0.19 accepted by the $r = 0.75$ rule. (b) Ambiguous match: $d_1 \approx d_2$, ratio close to one, rejected.

homography, a 3×3 invertible matrix $\mathbf{H}$ such that in homogeneous coordinates

$$\tilde{\mathbf{x}}' \propto \mathbf{H}\,\tilde{\mathbf{x}}. \tag{2.7}$$

Four correspondences in general position determine $\mathbf{H}$ up to scale.
RANSAC [31] finds the homography supported by the largest inlier set [16]. It draws a minimal random sample (here four matches), fits the model, counts the remaining matches with reprojection error below τ, and repeats. The number of iterations needed to draw at least one outlier-free sample with probability p is

$$N = \frac{\log(1-p)}{\log\!\left(1-(1-\epsilon)^{s}\right)}, \tag{2.8}$$

where ϵ is the outlier fraction and $s = 4$ [18]. We use OpenCV's implementation with $\tau = 2.5\,\text{px}$, $p = 0.999$ and a 2000-iteration cap. The returned inliers are the geometrically consistent matches; their count is used downstream as a quality score. Figure 2.6 shows the inlier set surviving the full Lowe-plus-RANSAC chain on a representative frame pair.

2.4 Per-Frame Signatures for Sequence Alignment

Dynamic time warping (Section 2.5) aligns two sequences of per-frame feature vectors. Producing those vectors is the role of this section. Two families are relevant: dense optical flow, which captures apparent motion, and gradient orientation histograms, which capture geometric structure. Both were investigated; Chapter 4 explains why the gradient descriptor was retained [32].

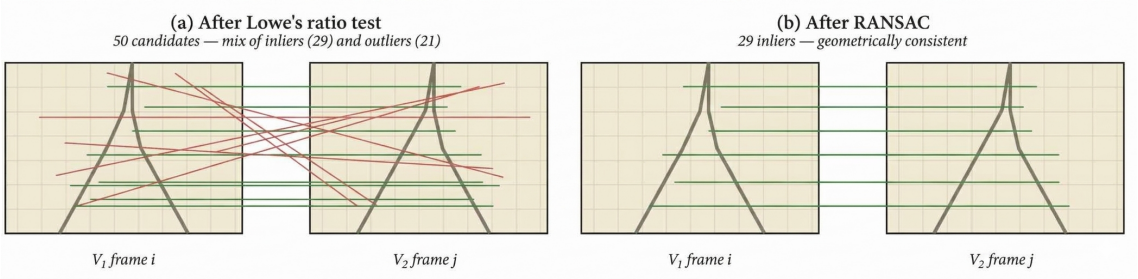

Figure 2.6: Effect of the Lowe + RANSAC chain on a Plan 1 frame pair. (a) After Lowe's ratio test: 50 candidate matches, mixing 29 geometric inliers (green) and 21 outliers (red). (b) After RANSAC: only the 29 matches consistent with a single homography are kept.

2.4.1 Optical Flow: Concept

The basic assumption is brightness constancy: a small region keeps the same intensity as it moves between consecutive frames [32]. If (u, v) denotes the displacement between t and $t + 1$,

$$I(x, y, t) = I(x + u, y + v, t + 1). \tag{2.9}$$

A first-order Taylor expansion yields the optical flow equation

$$I_x u + I_y v + I_t = 0, \tag{2.10}$$

with I_x, I_y, I_t the partial derivatives. One equation for two unknowns; regularisation is required. Lucas-Kanade assumes locally constant flow, Horn-Schunck globally smooth, Farneback uses local polynomial expansions [15].
The pipeline uses *dense* flow: a displacement vector per pixel. This is more expensive than sparse flow but yields a global motion signature of the frame, which is what DTW needs to align videos at different speeds.

2.4.2 Optical Flow: Farneback Dense Algorithm

Farneback's algorithm [33] approximates each image patch with a quadratic polynomial

$$f(\mathbf{x}) \approx \mathbf{x}^\top \mathbf{A}\, \mathbf{x} + \mathbf{b}^\top \mathbf{x} + c, \tag{2.11}$$

and recovers local displacement by comparing the coefficients of the same patch across consecutive frames, on a coarse-to-fine pyramid for robustness to large motions. We use OpenCV's implementation: three-level pyramid, 15×15 averaging window, three iterations per level. Figure 2.7 visualises a typical Farneback flow field on a railway frame, with hue encoding direction and brightness encoding magnitude.
A limitation of optical flow as DTW input is that the signature is *speed-dependent*: the same location filmed at twice the nominal speed yields flow magnitudes twice as large, so two recordings at different speeds produce different vectors at the same geographic point, which is precisely what DTW is asked to disambiguate [13, 34]. The next subsection

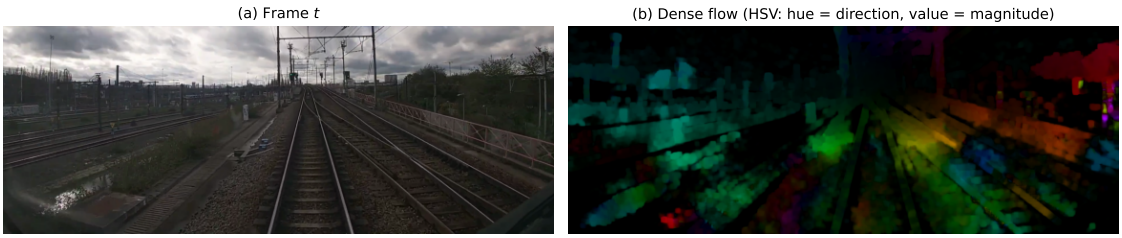

Figure 2.7: Dense Farneback optical flow on a railway frame. (a) Source frame at time t. (b) Flow field in HSV: hue encodes the direction of the per-pixel displacement, brightness its magnitude.

introduces an alternative descriptor depending only on visible content, not on camera motion.

2.4.3 Image Gradients

The image gradient $\nabla I = (I_x, I_y)$ is the vector of intensity partials, estimated in practice by Sobel convolution [35]. In polar form, $|\nabla I| = \sqrt{I_x^2 + I_y^2}$ measures local edge strength and $\theta = \arctan_2(I_y, I_x)$ encodes the direction perpendicular to the edge [4]. Two properties matter for frame description. As first-order derivatives, gradients are invariant to uniform brightness shifts [13]. And orientations of strong gradients encode geometric structure (rail edges, sleepers, lineside poles) tied to geographic content rather than camera speed [36].

2.4.4 Gradient Orientation Histograms

An image contains many gradient vectors, one per pixel. To turn them into a compact frame descriptor we aggregate into a histogram of orientations weighted by magnitude, as in the HOG descriptor [37]:

1. Compute ∇I at every pixel of the ROI with Sobel.
2. Bin orientation $\theta \in [0, 2\pi)$ into B bins.
3. For each pixel, add $|\nabla I|$ to its bin.
4. Normalise the resulting histogram.

With $\mathbf{h} = (h_1, \ldots, h_B)$ and $b(\theta)$ the bin index,

$$h_k = \sum_{(x,y):b(\theta(x,y))=k} |\nabla I(x,y)|, \qquad k = 1, \ldots, B. \tag{2.12}$$

L_1-normalising so $\sum_k h_k = 1$ cancels multiplicative intensity changes (a uniform contrast shift scales all magnitudes by the same factor) [11].
The original HOG [37] concatenates such histograms over a grid of cells for pedestrian detection. For sequence alignment, where the goal is to compare whole frames rather than localise an object, a single histogram over the ROI suffices. The pipeline aggregates

over the lower two thirds of the frame (the sky carries no geographic information) with $B = 16$ bins, yielding a sixteen-dim vector per frame. Chapter 4 reports the parameter sweep behind this design. Figure 2.8 illustrates the gradient field and the resulting 16-bin histogram on a typical frame.

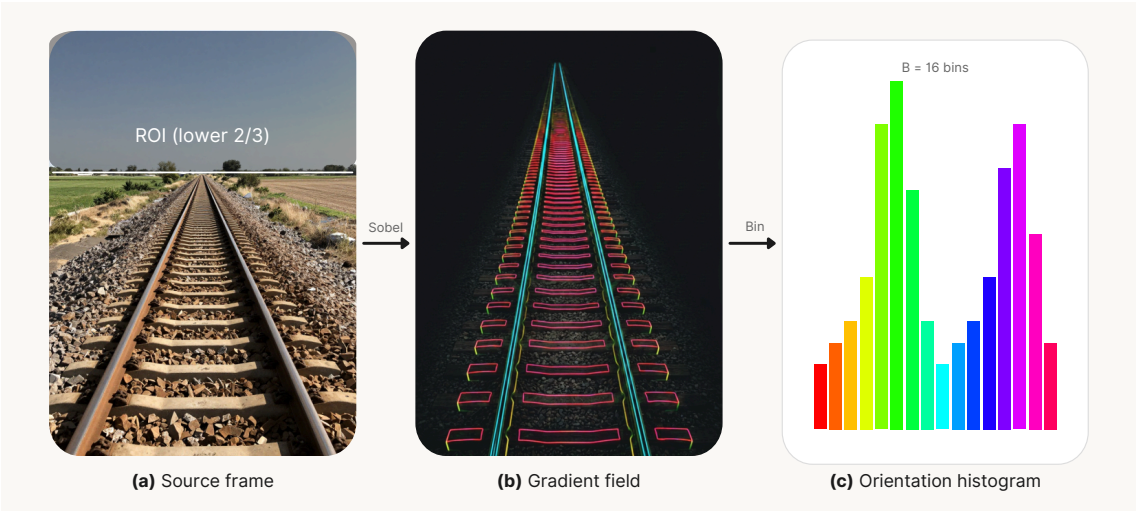

Figure 2.8: Construction of the 16-bin gradient orientation histogram on a typical railway frame. (a) Source frame, with the lower-two-thirds region of interest used by the pipeline. (b) Sobel gradient field, with orientation encoded by hue and magnitude by brightness. (c) Resulting 16-dim feature vector after L_1 normalisation.

2.4.5 Per-Video Feature Normalisation

Camera gain, exposure and lighting introduce a per-video bias unrelated to geographic content [38]. The standard remedy is per-video z-score normalisation [3]: each video independently has its feature vectors centred and rescaled to zero mean and unit variance per dimension, with no coupling between V_1 and V_2 statistics. Chapter 4 discusses why this is preferable to joint normalisation.

2.5 Dynamic Time Warping

Dynamic Time Warping (DTW) aligns two temporal sequences of possibly different lengths and locally different speeds. Originally developed for speech recognition [39], it has since spread to gesture recognition, biological sequence alignment and many other domains [34]. Here it provides the macro alignment skeleton on which the rest of the pipeline rests.

2.5.1 Classical DTW

Given two sequences $\mathbf{X} = (\mathbf{x}_0, \ldots, \mathbf{x}_{n-1})$ and $\mathbf{Y} = (\mathbf{y}_0, \ldots, \mathbf{y}_{m-1})$ of feature vectors and a pointwise distance d, DTW finds the warping path $\pi = (\pi_1, \pi_2, \ldots, \pi_K)$ through the $n \times m$

index grid that minimises the cumulative distance:

$$\mathrm{DTW}(\mathbf{X}, \mathbf{Y}) = \min_{\pi} \sum_{k=1}^{K} d(\mathbf{x}_{\pi_k^{(1)}}, \mathbf{y}_{\pi_k^{(2)}}), \tag{2.13}$$

subject to three constraints: the boundary condition $\pi_1 = (0, 0)$ and $\pi_K = (n-1, m-1)$, monotonicity (the indices in π never decrease), and a step-size restriction (each step is one of $(1, 0)$, $(0, 1)$ or $(1, 1)$) [36].

The optimal path is found by dynamic programming. We pre-compute the pointwise distance matrix

$$D[i, j] = d(\mathbf{x}_i, \mathbf{y}_j), \tag{2.14}$$

and fill an accumulated cost matrix with the recurrence

$$\mathrm{acc}[i, j] = D[i, j] + \min(\mathrm{acc}[i-1, j-1], \mathrm{acc}[i-1, j], \mathrm{acc}[i, j-1]). \tag{2.15}$$

The minimum value at the corner gives the alignment cost; the path itself is recovered by back-pointers. Figure 2.9 illustrates the algorithm on a toy example with two scalar signals.

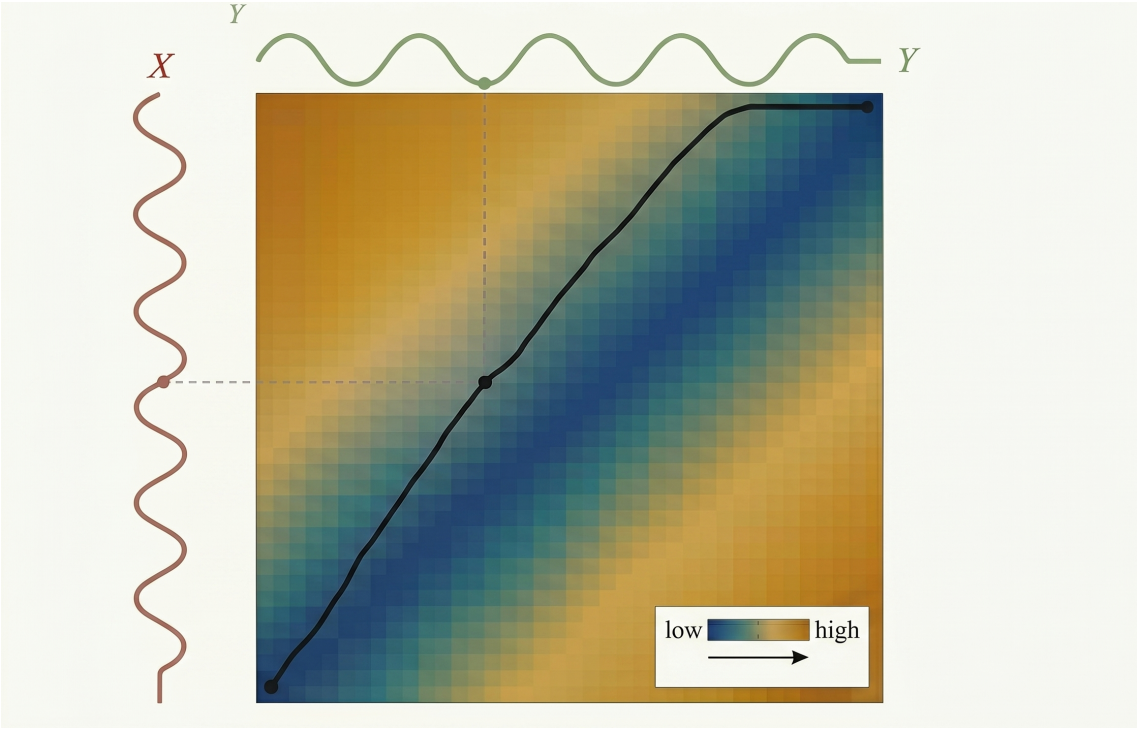

Figure 2.9: DTW on a toy example. Two sinusoidal signals at slightly different frequencies are placed along the margins: X in red on the vertical axis, Y in green on the horizontal axis. The centre shows the pointwise distance matrix (blue = low cost, orange = high cost), with the optimal warping path overlaid in black. The dashed lines mark one example sample-to-sample correspondence along the path.

2.5.2 Open-End DTW

The classical formulation forces the path to terminate exactly at $(n-1, m-1)$. This works when the two sequences cover the same content but breaks down when one extends beyond the other [13]. In our railway setting, the recordings start near the same location but may end at different positions if the trains travelled different distances; forcing both endpoints would artificially compress or stretch the final portion of the path [14].
Open-End DTW, also known as Subsequence DTW [1], removes the upper-right boundary condition. The path still starts at $(0, 0)$ but may terminate anywhere on the last row or column, at the point of lowest accumulated cost:

$$(i^*, j^*) = \arg\min\Big(\min_j \mathrm{acc}[n-1, j],\ \min_i \mathrm{acc}[i, m-1]\Big). \tag{2.16}$$

If the minimum is on the last row, V_1 is consumed entirely and V_2 is truncated; if on the last column, the reverse.

2.5.3 Step Penalty

The classical recurrence has a known failure mode, *stuttering*: on a flat or slow-varying portion, the path takes many vertical or horizontal steps in a row, assigning several consecutive frames of one video to a single frame of the other. On railway video this means matching one V_1 frame to many V_2 frames in a low-flow region, which is rarely what we want [40].
A common fix is to add a small penalty p to vertical and horizontal steps [41]:

$$\mathrm{acc}[i, j] = D[i, j] + \min\big(\mathrm{acc}[i-1, j-1],\ \mathrm{acc}[i-1, j] + p,\ \mathrm{acc}[i, j-1] + p\big). \tag{2.17}$$

Diagonal steps become strictly preferable when the local cost difference is small, encouraging a roughly one-to-one alignment in homogeneous regions while still allowing non-diagonal moves where the data justifies them.
The pipeline uses an *adaptive* penalty

$$p = \alpha \cdot \mathrm{mean}(D), \tag{2.18}$$

with α a hyperparameter (production default 0.3). The absolute scale of D depends on the feature vector and varies from one pair to the next, so a fixed penalty would need re-tuning each time.

2.5.4 Complexity

Time and space complexity are both $O(nm)$. For two videos sampled one frame in five from a 90 s, 30 fps recording, $n, m \approx 540$. The $\sim 290\,000$-cell cost matrix fits in a few MB and fills in well under a second on a modern CPU, two orders of magnitude faster than the feature-matching phase (Chapter 4).

2.5.5 Path Confidence Scoring

The DTW path is globally optimal by construction, but global optimality does not imply each local step is unambiguous. In a clear region the path traverses a deep low-cost valley; in a region of repetitive content (uniform ballast) every cell of a row looks similar and the algorithm picks the slightly cheapest one. Distinguishing the two regimes is useful diagnostically and is exploited downstream (Chapter 4).

We introduce a scalar score $\kappa_i \in [0, 1]$ that measures how strongly each pick is supported by the cost surface. For waypoint (i, j), let $\bar{D}_i = \text{mean}_k D[i, k]$ and $D_i^{\min} = \min_k D[i, k]$. Define

$$\kappa_i = \text{clip}\left(\frac{\bar{D}_i - D[i, j]}{\bar{D}_i - D_i^{\min} + \varepsilon}, 0, 1 \right). \tag{2.19}$$

$\kappa_i = 1$ if the path picked the cheapest cell in its row, $\kappa_i = 0$ if it picked a cell at or above the row mean. The row-dependent denominator makes κ_i comparable across rows of very different absolute scale. The confidence is computed on the raw distance matrix D, not on the accumulated cost acc: acc already incorporates the global path constraint and would always give a value close to one along the optimal path; we want the *local* quality of each pick, independent of how the algorithm reached it.

A run of consecutive waypoints with κ below a threshold (typically 0.25) defines a *low-confidence zone*, identifying regions where one should be most willing to override DTW when other evidence is available (Chapter 4). Figure 2.10 sketches the geometric interpretation of κ in a single row of the distance matrix.

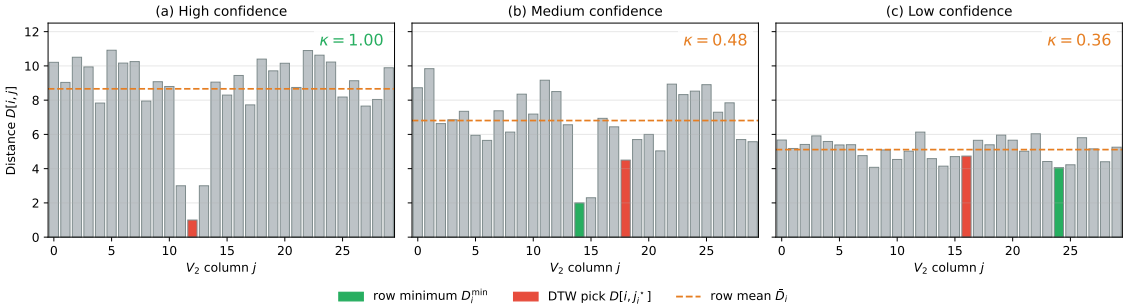

Figure 2.10: Geometric interpretation of κ on three illustrative rows of the distance matrix D. Each panel plots the row values against the V_2 column index, with the row minimum $D_i^{\min}$ in green, the DTW pick $D[i, j_i^\star]$ in red, and the row mean $\bar{D}_i$ as the orange dashed line. From left to right: high confidence ($\kappa = 1.00$, the DTW pick coincides with the row minimum), medium confidence ($\kappa = 0.48$), and low confidence ($\kappa = 0.36$, the DTW pick sits near the row mean).

2.6 Image Quality and Similarity Metrics

Once an alignment is produced, its quality must be measured. We use three complementary metrics: two compare aligned frame pairs pixel-wise or perceptually, the third evaluates

the geometric consistency of the alignment itself without looking at pixels.

2.6.1 Pixel-Level Baselines: MSE and PSNR

The mean squared error $\mathrm{MSE}(I_1, I_2) = \frac{1}{HW}\sum_{x,y}(I_1(x,y) - I_2(x,y))^2$ and the related peak signal-to-noise ratio are the historical pixel-level baselines. Both correlate poorly with human perception: a small spatial shift produces a large MSE while looking perceptually identical [42, 43]. We mention them only to motivate the perceptual metrics that follow.

2.6.2 Structural Similarity (SSIM)

Wang proposed the structural similarity index as a top-down alternative to pixel-error metrics [42]: the human visual system extracts structural information, so similarity should be measured at the level of local structure rather than individual pixels. SSIM compares two patches through three components: luminance, contrast and structure.

$$l(\mathbf{x}, \mathbf{y}) = \frac{2\mu_x\mu_y + C_1}{\mu_x^2 + \mu_y^2 + C_1}, \tag{2.20}$$

$$c(\mathbf{x}, \mathbf{y}) = \frac{2\sigma_x\sigma_y + C_2}{\sigma_x^2 + \sigma_y^2 + C_2}, \tag{2.21}$$

$$s(\mathbf{x}, \mathbf{y}) = \frac{\sigma_{xy} + C_3}{\sigma_x\sigma_y + C_3}, \tag{2.22}$$

with μ_x, μ_y local means, σ_x, σ_y local standard deviations and σ_{xy} local cross-covariance; the constants C_1, C_2, C_3 stabilise the ratios when denominators approach zero. The combined index is

$$\mathrm{SSIM}(\mathbf{x}, \mathbf{y}) = l(\mathbf{x}, \mathbf{y}) \cdot c(\mathbf{x}, \mathbf{y}) \cdot s(\mathbf{x}, \mathbf{y}), \tag{2.23}$$

ranging from -1 (anticorrelated) to 1 (identical). In practice the three statistics are computed in a sliding 11×11 Gaussian window, and the per-window scores are averaged into a global MSSIM.

SSIM was designed for and is best on local distortions such as compression artefacts or blur. It is sensitive to small spatial misalignments, which is both a strength (detects geometric errors) and a weakness (dominated by large photometric differences). Figure 2.11 contrasts SSIM and MSE on visually equivalent degradations, illustrating why pixel-level metrics fail to track perceived similarity. Chapter 5 returns to this point.

2.6.3 Learned Perceptual Image Patch Similarity (LPIPS)

A more recent line of work argues perceptual similarity is an emergent property of deep networks trained for unrelated high-level vision tasks: intermediate activations of a pre-trained CNN (AlexNet, VGG) correspond closely to human judgements of similarity [43], even though the network was never trained for that.

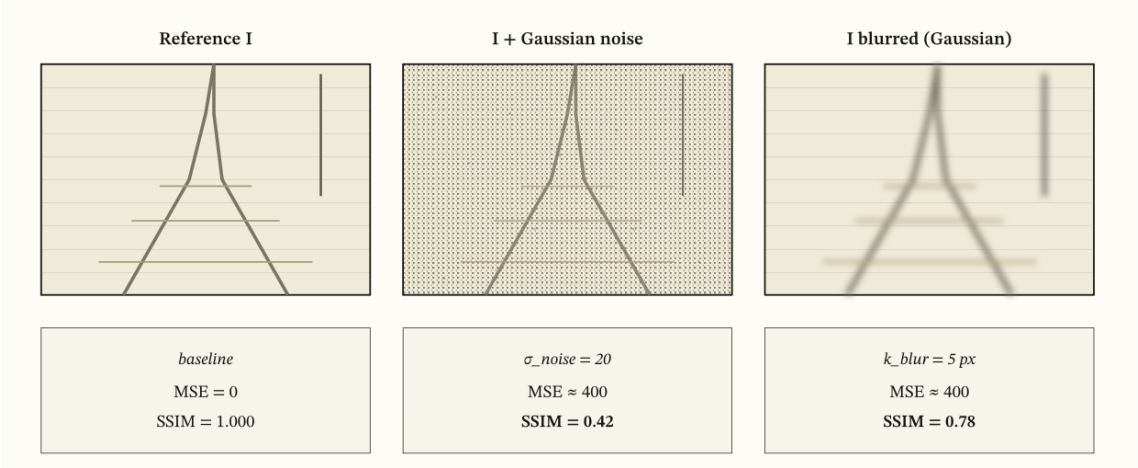

Figure 2.11: SSIM versus MSE on perceptually different but numerically equivalent degradations. Relative to the reference, the noisy ($\sigma = 20$, SSIM $= 0.42$) and blurred ($k = 5\,\text{px}$, SSIM $= 0.78$) variants both reach MSE ≈ 400, illustrating that MSE alone does not track perceived similarity.

LPIPS passes both images through a fixed pretrained network, channel-normalises the activations, weights them by a learned per-layer vector $\mathbf{w}_l$, and accumulates per-position L_2 distances:

$$d(\mathbf{x}, \mathbf{y}) = \sum_l \frac{1}{H_l W_l} \sum_{h,w} \left\| \mathbf{w}_l \odot \left(\hat{F}_l(\mathbf{x})_{h,w} - \hat{F}_l(\mathbf{y})_{h,w} \right) \right\|_2^2, \tag{2.24}$$

with $\hat{F}_l$ the unit-normalised activations of layer l and $\odot$ the elementwise product. The weights $\mathbf{w}_l$ are calibrated on the BAPPS dataset of 484 000 human similarity judgements [43].

LPIPS is more forgiving of small spatial misalignments than SSIM because deep features are spatially smoothed, which makes it a useful complement: a divergence between SSIM and LPIPS can flag geometry-off-but-appearance-correct cases or vice versa. Both share, however, a sensitivity to global photometric differences, addressed in Section 2.6.4.

2.6.4 Photometric Normalisation by Histogram Matching

When two videos are recorded on different days, the same physical scene looks different to SSIM or LPIPS because of lighting, weather and automatic camera gain [13]. A direct application would be dominated by these global photometric shifts and would tell us little about the geometric quality of the alignment.

The standard tool to compensate is *histogram matching* [44]. For a one-dim intensity channel, the transformation mapping the source CDF F_{V_2} onto the reference F_{V_1} is

$$T(v) = F_{V_1}^{-1}(F_{V_2}(v)), \tag{2.25}$$

with both CDFs estimated empirically on a representative frame sample. The transformed signal has approximately the same one-point statistics as the reference while preserving

the order of values, and therefore the geometric structure. In practice, T is precomputed once on a uniform sample of each video and stored as a 256-entry look-up table applied frame by frame at constant cost.

Histogram matching plays two roles in this work. **As an evaluation step**, it is applied to V2 frames before SSIM or LPIPS, so the metric measures alignment residual rather than day-to-day photometric drift; the per-channel RGB version is used. **As a preprocessing step**, the same construction is applied to V2's luminance channel in CIE L*a*b* space [45] before feature extraction, so that downstream descriptors see two videos already aligned at the first-order luminance level. Chapter 5 reports the ablation quantifying each contribution.

2.7 Cycle Consistency

The third metric is fundamentally different: it does not compare image content at all but asks whether the alignment is self-consistent under reversal.

The idea originates in CycleGAN [46], where a forward generator $G : A \to B$ and a backward generator $F : B \to A$ should satisfy $F(G(a)) \approx a$ [32]. The same principle applies to alignment. The forward alignment

$$f : V_1 \to V_2 \tag{2.26}$$

and the backward alignment (same algorithm, roles swapped)

$$g : V_2 \to V_1 \tag{2.27}$$

should compose to the identity on V_1: $g(f(i))$ should return a frame close to i. The cycle error is

$$\epsilon_{\text{cycle}}(i) = |i - g(f(i))|, \tag{2.28}$$

in frames; smaller is better.

Two properties make this valuable here. It is *content-agnostic*: it looks only at frame indices, so it is immune to the photometric ceiling that saturates SSIM and LPIPS. And it discriminates dramatically between random and informative alignments: a random alignment cycles with errors of hundreds of frames, a correct one with one or two. Figure 2.12 sketches the round-trip composition $i \to f(i) \to g(f(i))$ that defines the metric. Chapter 5 reports detailed numbers. One known degeneracy: a constant-offset bias cancels in the round trip; we return to this caveat where it matters.

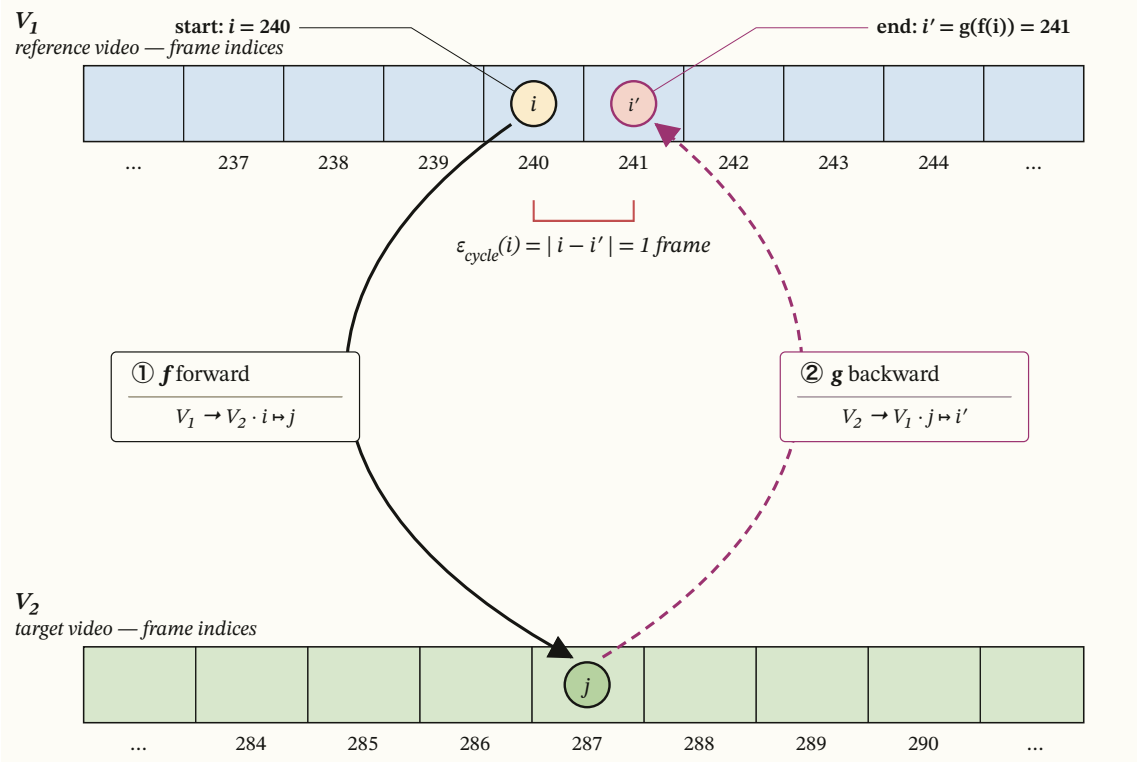

Figure 2.12: Cycle consistency. A V_1 frame i is mapped forward to V_2 by f (solid arrow), then back to V_1 by g (dashed arrow). The cycle error $\epsilon_{\text{cycle}}(i) = |i - g(f(i))|$ measures how close the round trip is to identity; in the example, $i = 240 \to j = 287 \to i' = 241$, so $\epsilon_{\text{cycle}} = 1$ frame.

2.8 Summary

This chapter introduced the building blocks: binary local descriptors (AKAZE, BRISK, ORB) matched by Hamming distance and filtered by Lowe's ratio test plus RANSAC homography [25, 31]; two families of per-frame signatures (dense flow and gradient orientation histograms) normalised per video; Open-End DTW with adaptive step penalty and per-waypoint confidence; histogram matching for both luminance preprocessing and photometric normalisation; SSIM, LPIPS and cycle consistency for evaluation. Several choices, binary over floating-point descriptors, gradient histograms over optical flow, open-end over classical DTW, and three complementary metrics rather than one, are justified empirically in Chapter 4. The next chapter situates the work in the literature.

3. Related Work

The technical components introduced in Chapter 2 are individually well established. The originality of this thesis lies in how they are assembled to solve one specific problem, namely the temporal alignment of two railway videos recorded at different times, under a strict CPU-only constraint. This chapter situates that contribution in the literature. Section 3.1 reviews the temporal alignment of sequences, from the Dynamic Time Warping family to its use in video. Section 3.2 turns to vision-based inspection of railway infrastructure. Section 3.3 states precisely what is new in the present work.

3.1 Temporal Alignment of Sequences

Aligning two ordered sequences that describe the same underlying process at different speeds is a classical problem. Its canonical solution, Dynamic Time Warping, predates the computer-vision applications by decades, and most modern variants are refinements of the same dynamic-programming core.

3.1.1 Dynamic Time Warping and its Variants

Dynamic Time Warping was introduced by [39] for spoken-word recognition, where the warped axis is one-dimensional time and the per-step features are cepstral coefficients. The transfer of the algorithm to general time-series analysis is associated with [47], who used it to find recurring patterns in time series, and the algorithm has since become a default tool whenever two sequences must be compared up to a non-linear time reparametrisation.
Three families of refinement are relevant to the present work. The first concerns the boundary conditions. Classical DTW forces the alignment to run from the first to the last sample of both sequences. [1] formalises the subsequence variant, in which the path is allowed to terminate anywhere on the last row or the last column, recovering the best alignment of one sequence to a contiguous part of the other. This is exactly the flexibility our setting needs, because two trains rarely stop at the same physical point. The second family concerns the local cost. The original slope-constrained schemes of [39] discourage the path from drifting too far from the diagonal; [41] revisits the same idea with a single intuitive penalty parameter added to non-diagonal steps, which is the scheme our pipeline adopts in adaptive form. The third family concerns the choice of per-step features. [36] replaces raw point values with local shape descriptors before warping, and [34] weights

features adaptively along the sequence. Both confirm a recurring lesson: the quality of a DTW alignment is governed first by the feature choice and only second by the warping search itself, a lesson that directly shaped the feature study reported in Chapter 4.
A separate concern is computational. [48] introduced lower-bounding and indexing structures that make DTW tractable on very long sequences, and [40] demonstrated that the popular FastDTW approximation is in practice slower and less accurate than an exact implementation on the sequence lengths typical of applied work. Our sequences are short enough that the exact quadratic algorithm runs in well under a second, so we use the exact formulation throughout. Figure 3.1 positions the variants discussed above against the choices adopted in our pipeline.

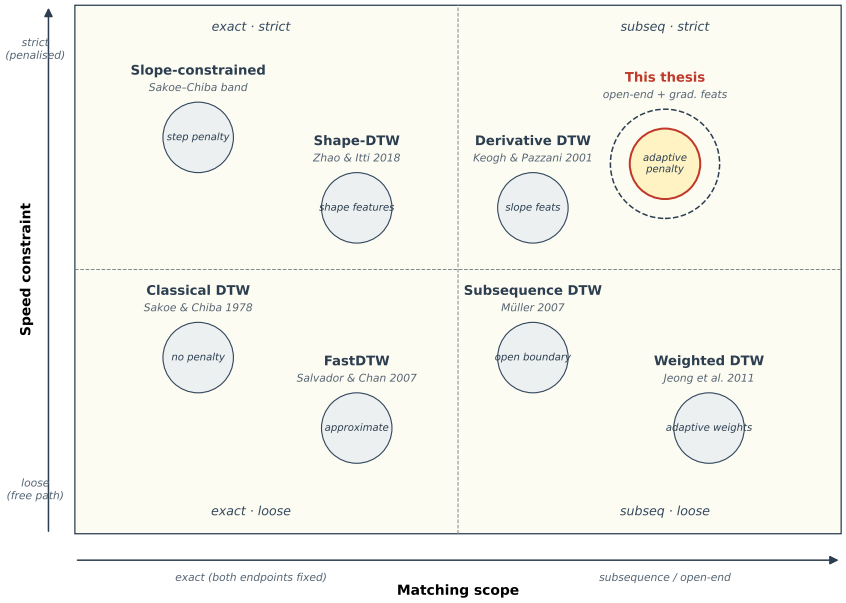

Figure 3.1: Positioning of DTW variants along two axes: matching scope (exact endpoints vs subsequence / open-end) and speed constraint (loose vs strict, penalised). The present work (highlighted) sits in the upper-right quadrant, combining the open-end formulation of [1] with an adaptive step penalty over gradient features.

3.1.2 Alignment of Video Sequences

Applying sequence alignment to video raises issues absent from the speech and time-series settings: the per-frame feature vector is not given by the sensor, it must be designed, and its design dominates the result. The work most directly comparable to ours is [13], which addresses video alignment explicitly for the purpose of change detection. That work aligns two video sequences captured along the same path so that differences between them can be attributed to genuine scene change rather than to misalignment. The problem statement is almost identical to ours, the difference being the application domain (general outdoor scenes versus railway track) and the deployment constraints. Our pipeline can be read as a

domain-specialised, CPU-only instance of the same general programme.
Earlier, [38] introduced Video Matching, which establishes dense spatio-temporal correspondences between two videos of a similar scene by jointly estimating a temporal alignment and a per-frame spatial warp. The method is powerful but assumes enough scene texture and computational budget to estimate dense correspondences, neither of which fits low-texture railway footage processed on a CPU. More recently, [14] synchronises multiple cameras from sparse feature points for 3D reconstruction, which confirms that feature-point matching remains a practical synchronisation cue, but in a rigid multi-camera setting whose single-parameter time map is simpler than the continuously varying warp produced by two independent train runs.
The use of DTW specifically as the temporal backbone for video appears, among others, in [45], which applies dynamic warping to audiovisual person recognition. The relevant takeaway is methodological: a DTW skeleton over engineered per-frame features remains a competitive and interpretable choice when training data for a learned aligner is unavailable, which is precisely our situation. Table 3.1 consolidates these comparisons against the present work along four axes (GPU dependency, label requirements, warp type and target content).

Table 3.1: Comparison of the video-alignment approaches discussed in this chapter.

Approach	**GPU**	**Labels**	**Warp type**	**Target content**
Video Matching [38]	no	no	dense spatio-temporal	textured outdoor scenes
Video alignment for change detection [13]	no	no	monotone temporal map	general outdoor scenes
Multi-camera sync [14]	no	no	single-parameter shift	rigid multi-camera rigs
Present work	**no**	**no**	**monotone temporal map**	**railway, CPU-only**

3.2 Vision-Based Railway Infrastructure Inspection

The industrial domain of this thesis has a substantial literature of its own. It intersects with, but does not subsume, the alignment problem we address: most railway-vision work operates on a single pass and asks “is there a defect in this frame?”, whereas we ask the prerequisite question “which frame of yesterday corresponds to this frame of today?”.

3.2.1 Defect Detection on a Single Pass

The dominant theme is the detection of defects in individual frames of a single recording. [6] train a deep convolutional network to classify rail-surface defects directly from images. [11] target fastener detection for autonomous visual inspection with a robust hand-crafted plus learned pipeline. [3] provide a comparative study of image-processing and deep-learning techniques for fastener defect detection, and [4] propose a single deep network that jointly

addresses rail-surface and fastener defects. These works share three characteristics that make them adjacent but not directly comparable to ours. They operate on one video stream rather than aligning two. They generally assume the acquisition conditions (camera, speed, lighting) are controlled by an instrumented vehicle, which limits transfer to consumer-grade footage. And they do not produce a temporal correspondence between two captures of the same infrastructure, which is the missing prerequisite that this thesis supplies.

3.2.2 Track Geometry, Maintenance and System-Level Inspection

A second strand targets track geometry and maintenance planning rather than per-frame appearance. [8] review the modelling of track-geometry degradation and the maintenance decisions it informs, and [2] formulate maintenance scheduling as a data-driven optimisation over geometry measurements. [5] and [7] estimate the vertical track alignment from instrumented wheelsets and observer-based methods respectively, which are sensor-based alternatives to the camera-only approach taken here. These works define the downstream decision problem that a change-detection system, fed by an aligner such as ours, would ultimately serve.

At the system level, [9] describes an integrated tie-and-ballast assessment platform, and the Federal Transit Administration pilot reported by [12] demonstrates an automated track-video inspection deployment. [10] proposes an evaluation framework for vision-based on-board rail-track detection, which is methodologically relevant because it underlines, as we do in Chapter 5, that the evaluation protocol for railway vision is itself an open question. The common architectural pattern across these system papers is a coarse but robust global stage followed by a finer visual stage, which is exactly the coarse-to-fine structure this thesis adopts. Figure 3.2 situates the present thesis (Stage (ii)) inside the broader inspection chain assumed by these system-level works, which further split Stage 3 of the four-stage taxonomy of Chapter 1 into two distinct operations: spatial registration (iii) and change detection proper (iv).

3.2.3 Datasets and the Comparability Problem

A practical obstacle to quantitative comparison with prior railway work is the near-absence of shared two-pass datasets. The defect-detection papers above use proprietary footage collected by the host operator, with no public release of paired multi-day recordings. The dataset used in this thesis is itself proprietary and cannot be redistributed; only aggregate metrics are reported. This limits us to qualitative architectural comparison with the system-level works rather than a head-to-head benchmark, a limitation we acknowledge explicitly in Chapter 5 and revisit in Chapter 7.

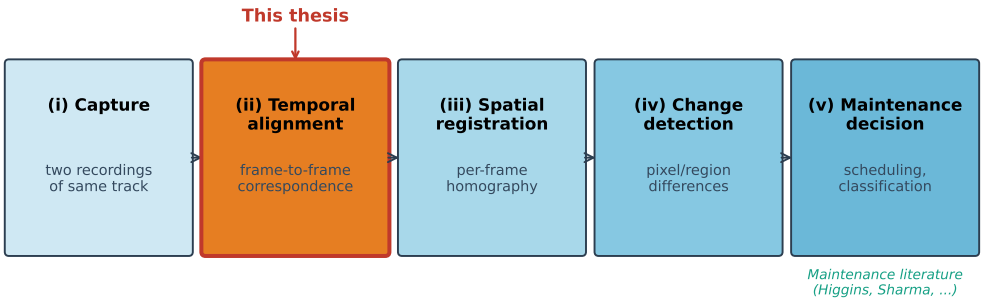

Figure 3.2: Five-stage vision-based railway inspection chain. The present thesis covers Stage (ii) only — producing the frame-to-frame correspondence between two captures of the same track — while Stage (v) is the natural application of the maintenance literature reviewed in Section 3.2.2.

3.3 Position of the Present Work

Against this backdrop, the contribution of the thesis is best stated by what it combines, what it specialises, and what it deliberately avoids.

The work *combines* a subsequence DTW skeleton with a strict feature-matching refinement and a global monotonicity-preserving optimisation. The skeleton draws directly from the line opened by [39] and extended by [1], with the adaptive step penalty in the spirit of [41]. The refinement draws on the binary-descriptor matching literature reviewed in Chapter 2. The closest published relative of the overall task is [13]; our work differs in being specialised to railway content and constrained to CPU-only execution.

The work *specialises* this combination for the railway domain. The per-frame DTW feature is a gradient-orientation histogram chosen to preserve geometric structure across photometric variation between recording days, an instance of the descriptor family studied by [37]. A luminance preprocessing step removes the first-order brightness difference before feature extraction. A path-confidence signal computed alongside the DTW path identifies regions where the cost surface is unreliable and where a coherent feature-matching deviation may correct the skeleton. The individual ingredients are standard; their combination and their use in railway alignment specifically is, to our knowledge, novel.

The work *avoids* approaches that would violate the industrial constraints of the deployment scenario, in particular GPU-only learned aligners and methods that depend on railway-specific training corpora that do not exist publicly. This is a deliberate scope reduction, not a verdict on those methods. Chapter 7 returns to this and outlines how the present pipeline could serve as the engineered baseline against which future learned methods are measured on the same data.

3.4 Summary

This chapter surveyed the two bodies of work that bear on the problem. Sequence alignment by DTW and its variants supplies the methodological backbone, with the subsequence formulation and the adaptive step penalty being the two refinements we adopt. Vision-based railway inspection provides the industrial backdrop: it defines the downstream decision problem, validates the coarse-to-fine architectural pattern, and, through its lack of shared two-pass datasets, explains why our evaluation is necessarily metric-based rather than a head-to-head benchmark. The closest single point of comparison, video alignment for change detection, confirms that the problem is recognised in the literature while leaving the railway-specialised, CPU-only instance of it open. The next chapter details the pipeline that fills that gap.

4. Methodology

This chapter combines the building blocks of Chapter 2 into a working pipeline and justifies every non-trivial decision made along the way. Section 4.1 gives the bird's-eye view. Section 4.2 explains how the inner feature-matching algorithms were chosen. Sections 4.3 through 4.8 walk through the six phases of the pipeline in order. Section 4.9 closes the chapter with the engineering decisions that surround the algorithm proper.

4.1 Pipeline Overview

The pipeline takes a pair of forward-facing videos V_1 and V_2, recorded on different days along the same physical trajectory, and produces a frame-by-frame correspondence list. Each entry of the list maps a frame index in V_1 to a frame index in V_2 such that the two frames depict the same geographic location along the track. The mapping is monotonically non-decreasing by construction, reflecting the fact that the train moves forward in both recordings.

The pipeline is structured around the well-known coarse-to-fine principle. A globally robust but locally inexact alignment is computed first, then refined locally by a more precise but locally myopic algorithm, with global constraints reasserted afterwards to prevent the refinement from accumulating drift. This structure is a recurring pattern in image registration and feature-based matching [21]; what is specific to the present work is how the principle is instantiated for railway alignment under CPU-only execution.

The six phases of the pipeline are summarised below, depicted in Figure 4.1, and developed in the rest of the chapter:

- **Phase 0: Preprocessing.** A luminance histogram-matching step removes the first-order brightness bias between V_2 and V_1 before any feature extraction.
- **Phase A: DTW skeleton.** A gradient-based per-frame descriptor is computed for both videos, normalised per video, and aligned by Open-End Dynamic Time Warping. The output is a coarse alignment plus a per-frame confidence score.
- **Phase B: Feature-matching refinement.** Each Phase A prediction is refined locally by AKAZE, BRISK or ORB feature matching within a strict, narrow temporal window around it.

- **Phase B': Rescue.** In runs of low-confidence frames where Phase B produces coherent contradictory evidence, the Phase A prediction is corrected by the matcher's median offset before being passed on.

- **Phase C: Global optimisation.** A monotonicity-preserving dynamic-programming pass over the top-K candidates of each frame eliminates residual jitter and ensures temporal consistency.

- **Phase D: Smoothing.** A short weighted moving average absorbs single-frame jitter while preserving monotonicity.

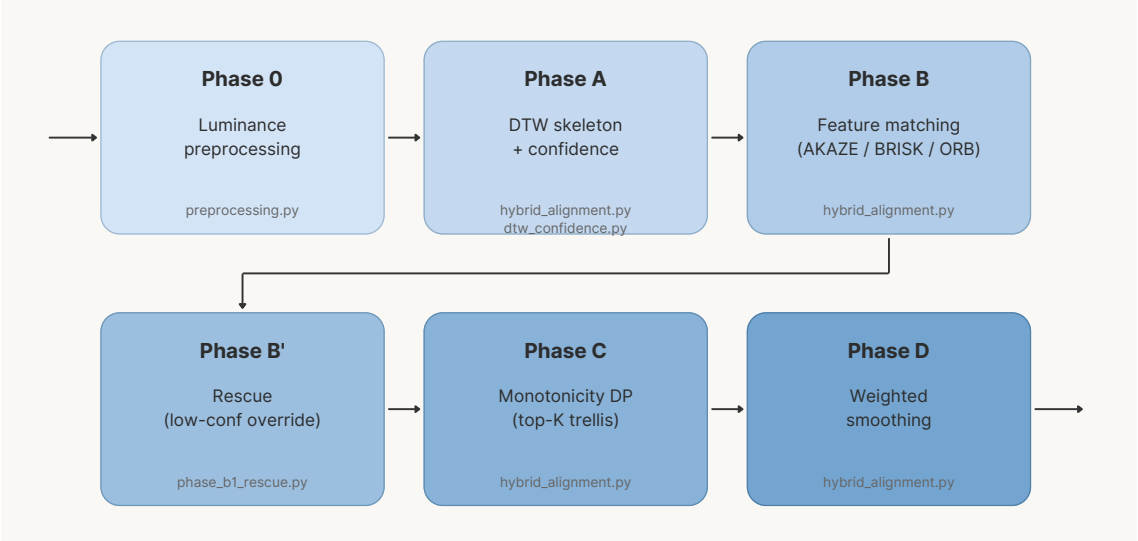

Figure 4.1: Master diagram of the six-phase hybrid pipeline. Each box names the phase, summarises its role and points to the Python module that implements it; arrows show the data flow from raw (V_1, V_2) frames to the final aligned mapping.

The architectural decision that drives the design is the strict separation between the global skeleton produced by DTW and the local refinement produced by feature matching. DTW guarantees globally consistent positioning at the cost of frame-level precision; feature matching gives frame-level precision at the cost of being purely local. The pipeline binds them together by an invariant we maintain at all points: *the local refinement window is never extended on failure.* When feature matching fails to find a good match within the prescribed window, the pipeline retains the DTW prediction rather than searching wider. This invariant is the principled mechanism by which the pipeline avoids the drift accumulation that plagues purely greedy approaches. The invariant admits exactly one carefully circumscribed exception, the Phase B' rescue, where the matcher's own evidence – coherent across multiple consecutive frames in a low-confidence region – is allowed to shift the DTW prediction by a bounded amount.

4.2 Algorithm Selection

Before any architectural decision can be made, the inner feature-detector family must be chosen. A literature review carried out at the start of the project covered fifteen feature-detection or feature-description algorithms, drawn from three eras: classical floating-point descriptors (SIFT, SURF), classical binary descriptors (BRIEF, FREAK, KAZE, AKAZE, BRISK, ORB), and learning-based methods (TILDE, LIFT, SuperPoint, D2-Net, R2D2, DISK, ALIKE) [21, 20]. Only the elimination logic is presented here; the full discussion is in Appendix A.

Three criteria drove the elimination, in priority order. The first was the deployment constraint stated in Chapter 1: the pipeline must run on a CPU-only system, without a GPU and without a pretrained neural network whose inference would require one [49]. This eliminated seven of the fifteen candidates (TILDE, LIFT, SuperPoint, D2-Net, R2D2, DISK, ALIKE). The second was matching speed at production scale: a 90 s 30 fps video pair invokes the inner matcher roughly 27 000 times, where the difference between a floating-point descriptor matched by L_2 distance and a binary descriptor matched by Hamming distance is decisive (seconds vs. milliseconds per frame) [24, 20]. This eliminated SIFT and SURF on engineering grounds. The third was the availability of a mature OpenCV implementation. This eliminated BRIEF (descriptor only, no detector [24]), KAZE (subsumed by AKAZE on edge-rich content), and FREAK (redundant with BRISK [21]; we kept one representative per design family). Figure 4.2 summarises the elimination tree.

The three retained algorithms – AKAZE, BRISK and ORB – span the design space of practical CPU-only binary-descriptor methods. From the pipeline's standpoint, they are functionally interchangeable: each exposes the same OpenCV interface (`detectAndCompute`), each is matched by the same brute-force Hamming matcher, and each is filtered by the same Lowe-and-RANSAC chain. The retention of three rather than one allows the empirical comparison reported in Chapter 5 to isolate the effect of the inner descriptor from the effect of the surrounding pipeline.

4.3 Phase 0: Preprocessing

The first phase of the pipeline operates on raw frames, before any feature is extracted. It addresses a problem that all subsequent phases would otherwise have to compensate for: the global luminance bias between two recordings made on different days.

4.3.1 Motivation

Two recordings of the same trajectory typically differ in their global luminance distribution, often substantially [13]. On the data used in this thesis, the reference video V_1 has an average L channel value around 35 (overcast capture) while the target video V_2 is around 120 (sunlit capture). Any per-frame descriptor that derives from intensity values inherits

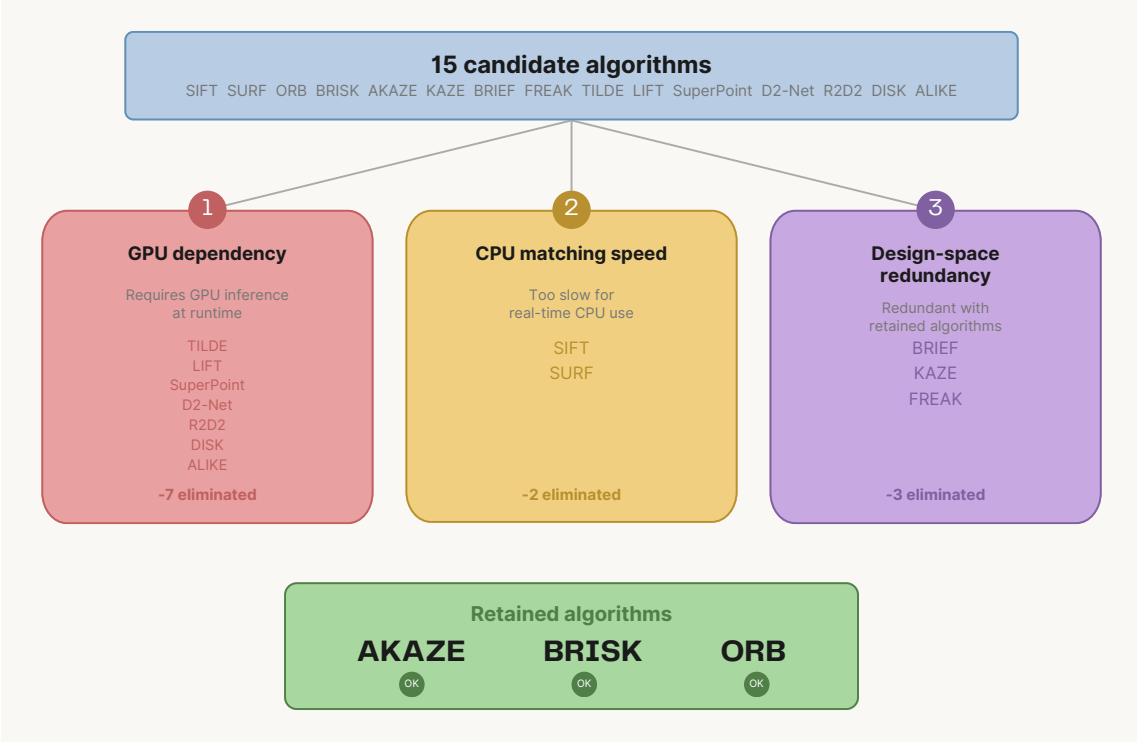

Figure 4.2: Elimination tree for the algorithm selection. The fifteen candidates are pruned by three sequential criteria — GPU dependency, CPU matching speed, and design-space redundancy — leaving AKAZE, BRISK and ORB as the retained representatives.

this discrepancy as a systematic bias in the pairwise distance matrix: the same geographic point produces different descriptor vectors in the two videos, not because the geography differs but because the cameras saw different absolute intensities. The downstream DTW algorithm has no way to distinguish a real geographic difference from a global brightness shift.

4.3.2 Algorithm

We address this with a standard histogram-matching procedure, applied globally and once per video pair. The construction operates in CIE Lab colour space on the L channel only [45], leaving the a and b channels untouched. The steps are:

1. Sample sixty frames uniformly from each of V_1 and V_2.
2. For each sample, compute the histogram of L values over the lower two thirds of the frame (the sky band carries no geographic information and would distort the statistics).
3. Aggregate the per-frame histograms into a per-video histogram, and compute the corresponding cumulative distribution functions F_{V_1} and F_{V_2}.
4. Construct a 256-entry look-up table T defined by $T[v] = F_{V_1}^{-1}(F_{V_2}(v))$, computed efficiently with `np.searchsorted` on the two CDFs.

At inference time, every frame of V_2 has the look-up table applied to its L channel before any feature extraction; the frames of V_1 are left untouched as the reference. Figure 4.3 shows a typical before/after frame pair together with the corresponding L-channel histograms.

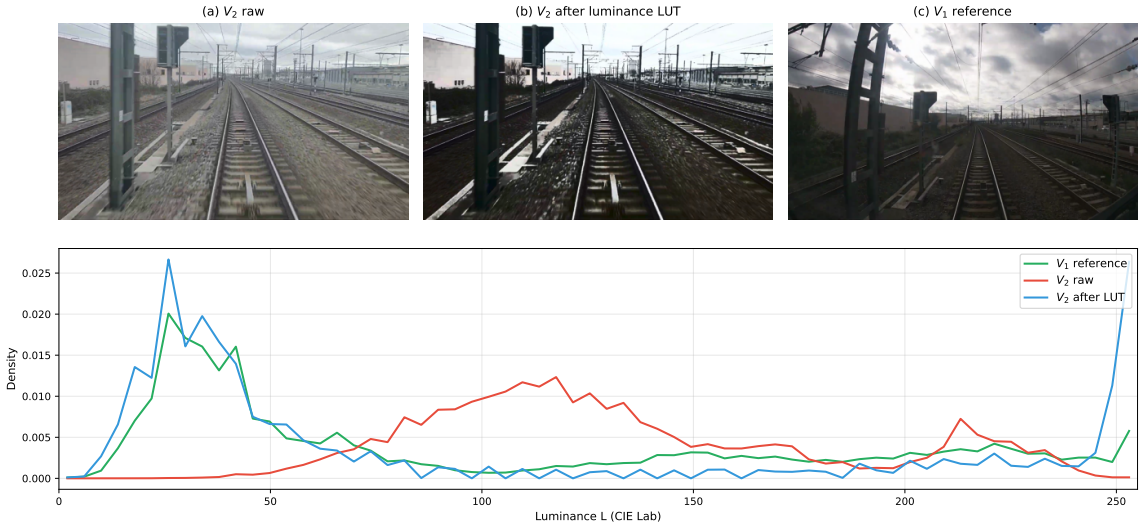

Figure 4.3: Effect of the Phase 0 luminance LUT on a Plan 2 frame pair. Top: V_2 raw (a), V_2 after the LUT (b), and the V_1 reference (c). Bottom: the corresponding L-channel histograms in CIE Lab space — after the LUT, the V_2 distribution (blue) closely matches the V_1 reference (green), whereas the raw V_2 (red) is shifted towards higher luminance.

4.3.3 Why This Is Principled

Histogram matching is a textbook image-processing operation [44]: the unique monotone map between two distributions that preserves intensity ordering, hence local geometric structure. The look-up table depends only on the two inputs and is deterministic. The procedure can be disabled (`enable_preprocessing=False`) for ablation; its empirical impact is quantified in Chapter 5.

4.4 Phase A: Macro-Synchronisation by DTW

Phase A produces the global skeleton of the alignment. For every frame of V_1, it predicts a corresponding frame in V_2, with a precision sufficient for downstream feature matching to find a small refinement window in which to operate. The phase is decomposed into four steps: per-frame descriptor extraction, normalisation, Open-End DTW computation, and the derivation of a per-frame confidence score.

4.4.1 Per-Frame Descriptor: Gradient Orientation Histograms

The choice of per-frame descriptor is the most consequential design decision in the pipeline. The production pipeline uses gradient orientation histograms over the dense optical flow alternative introduced in Chapter 2. The descriptor is computed in the following steps:

1. Read the BGR frame; if Phase 0 preprocessing is enabled and the frame belongs to V_2, apply the luminance look-up table.

2. Crop the top third of the frame (sky removal) and convert to grayscale.

3. Compute Sobel derivatives g_x and g_y with a 3×3 kernel, then magnitude $|\nabla I| = \sqrt{g_x^2 + g_y^2}$ and orientation $\theta = \arctan_2(g_y, g_x) \in [0, 2\pi)$.

4. Bin θ into sixteen bins of width $\pi/8$, each pixel contributing its magnitude [21].

5. L_1-normalise the resulting histogram so that its entries sum to one.

The output is a sixteen-dimensional vector per frame. The descriptor is computed at a fixed sampling rate `dtw_sample_rate` (default five), so the sequence has approximately $T_{\text{video}} \cdot \text{fps}/5$ entries. For our typical 90 s 30 fps video this is around 540 vectors, which makes the downstream DTW matrix tractable.

The justification for choosing gradient histograms over the optical-flow alternative is empirical, obtained by an exhaustive sweep over candidate descriptors and transformations on the Plan 2 ground truth: nine extractors, three feature transformations, and four step penalties, for a total of 108 configurations, each scored against fourteen manually-identified ground-truth frame pairs. The leaderboard, shown in Figure 4.4 and summarised in Chapter 5, reveals that gradient orientation histograms occupy the entire top five of the ranking, with the next-best non-gradient descriptor (an intensity histogram) at more than twice the error. This previous-era default (intensity histograms) reaches a mean absolute error of 74.3 frames; gradient orientation histograms reach 33 frames – more than a twofold improvement.

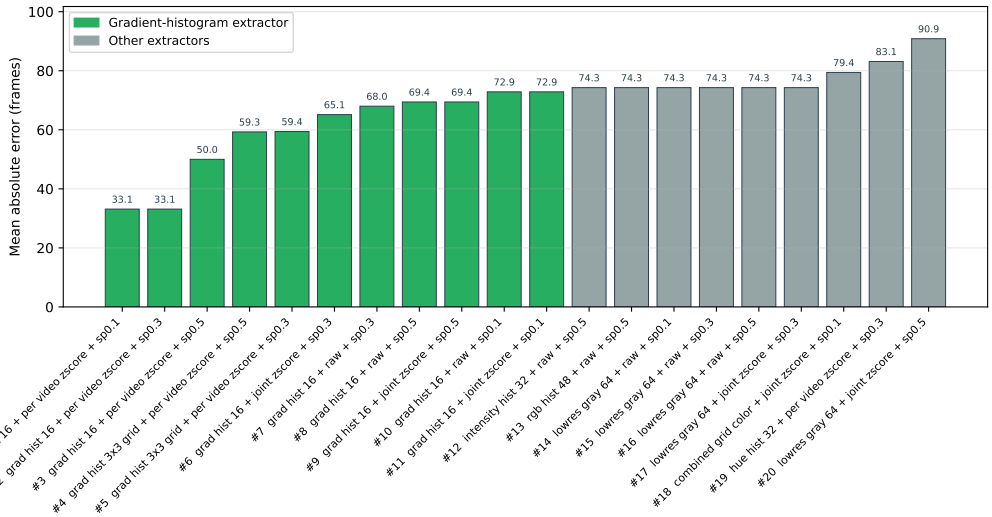

Figure 4.4: Top-twenty configurations of the Plan 2 design-selection sweep, ranked by mean absolute alignment error (in frames). Gradient-histogram extractors (green) dominate the head of the leaderboard; the first non-gradient candidate (intensity histogram, position 12) sits at 74.3 frames, more than twice the best.

The reason was discussed in Chapter 2: optical flow encodes instantaneous motion, which depends on speed, so two recordings at different speeds give different vectors at the same geographic point and DTW collapses onto a near-diagonal but globally wrong path. Gradient orientation histograms encode geometric structure (rail edges, sleepers, lineside poles [11]) that depends on position rather than speed.

4.4.2 Per-Video Z-Score Normalisation

After both videos have been processed, the two feature matrices $X \in \mathbb{R}^{n_1 \times 16}$ and $Y \in \mathbb{R}^{n_2 \times 16}$ are normalised independently by per-video z-score:

$$\tilde{X}_i = \frac{X_i - \mu_X}{\sigma_X + \varepsilon}, \quad \tilde{Y}_j = \frac{Y_j - \mu_Y}{\sigma_Y + \varepsilon}. \tag{4.1}$$

The choice of *per-video* rather than joint normalisation is empirically justified by the same sweep. For the same gradient extractor and step penalty, switching to joint z-score (computed on the concatenation of X and Y) increases mean absolute alignment error from 33 to 65 frames, and no normalisation at all gives 68. Joint normalisation averages the two distributions and contaminates each with the other's bias, which DTW then has to disentangle from the geographic signal. Per-video normalisation removes the per-video baseline without coupling the two distributions. Figure 4.5 shows the pairwise distance matrix under the three normalisation regimes, where only per-video z-score exposes the geographic diagonal cleanly.

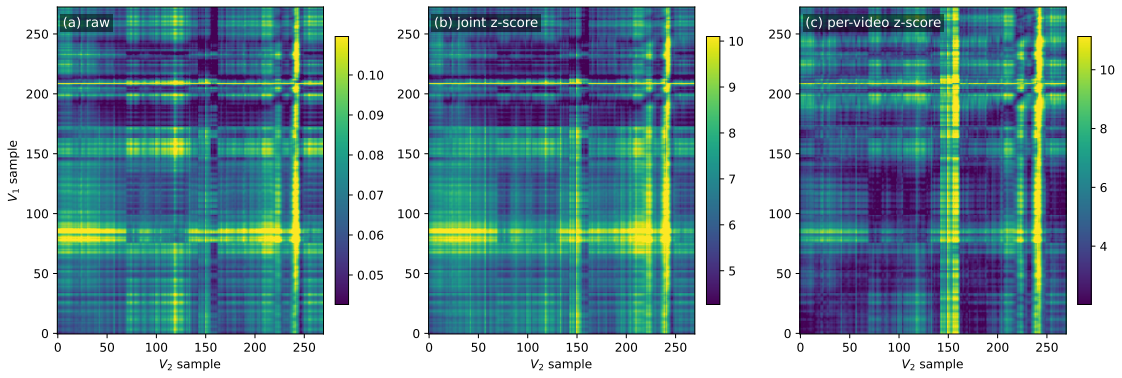

Figure 4.5: Pairwise distance matrix between V_1 and V_2 feature vectors on Plan 2 under three normalisation regimes: (a) raw features, (b) joint z-score over the concatenation $[X; Y]$, and (c) per-video z-score applied independently to each sequence. Only (c) exposes the geographic diagonal valley with enough contrast for DTW to follow it reliably.

4.4.3 Open-End Dynamic Time Warping

The two normalised feature sequences are aligned by Open-End DTW. The algorithm is the one described in Chapter 2 Section 2.5.2, with the adaptive step penalty of Section 2.5.3. We summarise the operational specifics here.

The pointwise distance is Euclidean: $D[i,j] = \|\tilde{X}_i - \tilde{Y}_j\|_2$. The adaptive step penalty is $p = \alpha \cdot \mathrm{mean}(D)$ with $\alpha = 0.3$. The accumulated cost matrix is filled by the standard recurrence

$$\mathrm{acc}[i,j] = D[i,j] + \min\big(\mathrm{acc}[i-1,j-1],\ \mathrm{acc}[i-1,j] + p,\ \mathrm{acc}[i,j-1] + p\big), \tag{4.2}$$

and the path is recovered by backtracking from the open-end terminal $\arg\min(\min_j \mathrm{acc}[n_1 - 1, j], \min_i \mathrm{acc}[i, n_2 - 1])$ to $(0,0)$, with diagonal predecessors preferred on ties to encourage one-to-one alignment.

The value $\alpha = 0.3$ was selected from the same sweep. The alignment error is essentially constant for $\alpha \in [0.1, 0.3]$, rises slightly for $\alpha = 0.5$ (diagonal bias too strong to accommodate real speed differences), and collapses to a degenerate path at $\alpha = 0$. We pick $\alpha = 0.3$ in the middle of the safe plateau, which we expect to generalise more robustly to plans whose ground truth we have not measured.

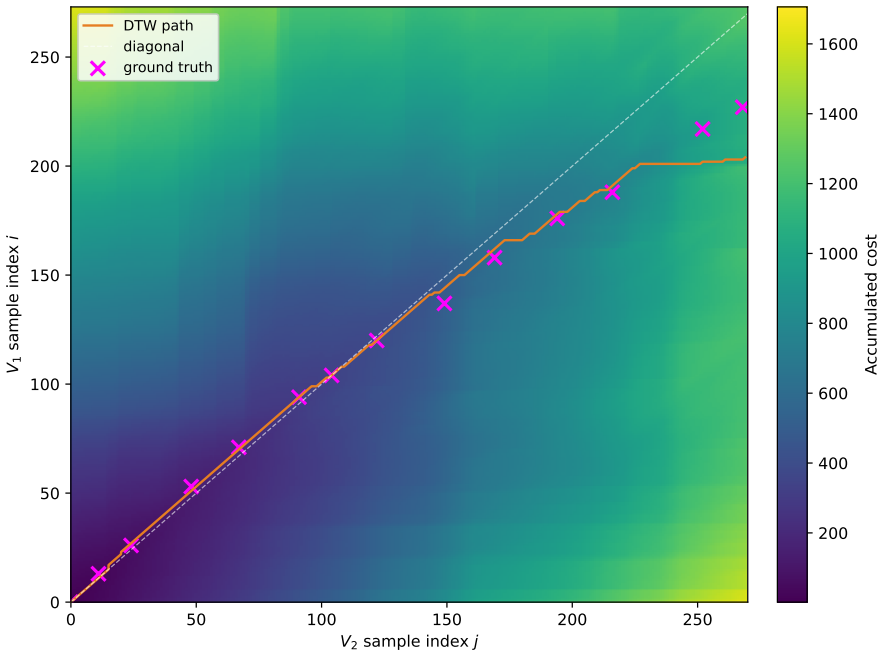

Figure 4.6: Accumulated cost matrix on Plan 2 with the DTW path (orange) and the fourteen ground-truth anchors (magenta crosses) overlaid; the diagonal is shown as a dashed white reference. The path tracks the anchors closely up to $i \approx 200$ then drifts as the cost surface flattens near the end of the sequence.

The DTW path is in sampled-index space (one waypoint per sampled frame). The pipeline converts it back to original frame indices and densifies it to one entry per V_1 frame by linear interpolation between waypoints, producing the mapping dtw_mapping$[i] \to$ predicted V_2 frame used by Phase B. Figure 4.6 shows the accumulated cost matrix and recovered path on Plan 2, with the fourteen ground-truth anchors overlaid for reference.

4.4.4 Per-Frame Confidence Scoring

The DTW path is globally optimal by construction, but global optimality does not say how strong the local picks were: a path can be optimal because the cost matrix has one clear valley, or because every cell of a row is roughly equivalent and the DP picked the slightly cheapest. The two regimes call for different downstream treatment, namely trusting DTW in the first and being willing to correct it in the second.

To distinguish the regimes, we compute a per-waypoint confidence score κ_i using the formula introduced in Chapter 2 Section 2.5.5:

$$\kappa_i = \mathrm{clip}\left(\frac{\bar{D}_i - D[i, j_i^\star]}{\bar{D}_i - D_i^{\min} + \varepsilon},\, 0,\, 1 \right), \tag{4.3}$$

where $j_i^\star$ is the V_2 index chosen by DTW at V_1 sampled index i, $\bar{D}_i$ is the row mean of D and $D_i^{\min}$ is the row minimum. The score is one if DTW picked exactly the row minimum and zero if it picked something at or above the row mean. It is computed on the raw distance matrix, not on acc, because we want to measure the local quality of each pick independently of how the algorithm reached it through the dynamic programme. The per-waypoint confidence is then projected onto every original V_1 frame by nearest-neighbour interpolation, yielding $\kappa_{\text{frame}}[i] \in [0, 1]$ for $i \in [0, n_1)$. The array is exposed as `HybridAligner.dtw_confidence_per_frame` for use by the diagnostic tooling and by Phase B'.

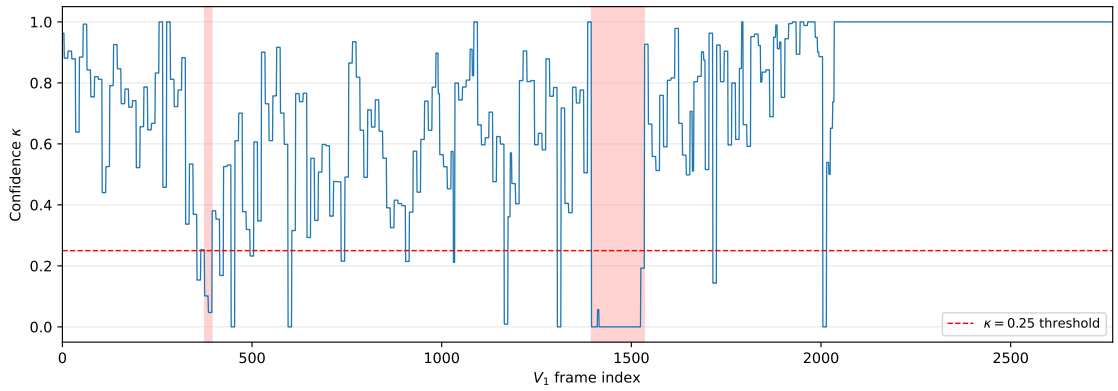

Figure 4.7: Per-frame DTW confidence κ along the Plan 2 timeline. The dashed red line marks the $\kappa = 0.25$ threshold; red shaded bands highlight the rescue zones — contiguous runs of at least twenty frames with $\kappa < 0.25$.

A contiguous run of V_1 frames with $\kappa_{\text{frame}} < 0.25$ of length at least twenty defines a *rescue zone*. Thresholds are chosen so that the rescue fires only in genuinely unreliable regions. On Plan 2, five rescue zones are identified, the largest spanning 180 frames in the section where V_2 accelerates sharply relative to V_1, which is also where the DTW skeleton struggles most. Figure 4.7 plots κ_{frame} along the Plan 2 timeline, with the detected rescue zones highlighted.

4.5 Phase B: Feature-Matching Refinement

Phase B refines each Phase A prediction by locally matching V_1 keypoints against V_2 keypoints within a strict window around the prediction. The phase is parametrised by the choice of inner algorithm (AKAZE, BRISK or ORB) and exposes the same external interface regardless of the choice. The phase decomposes into four steps: window definition, per-candidate matching, weighted scoring, and top-K candidate selection.

4.5.1 The Strict Search Window

For each V_1 frame i, the search range in V_2 is the interval

$$[\text{dtw\_mapping}[i] - W,\ \text{dtw\_mapping}[i] + W], \tag{4.4}$$

where W is the `search_window` parameter (default ten frames), clipped to the valid V_2 frame range $[0, n_2 - 1]$.
The invariant is that this window is *never extended on failure*. If no match within the window meets the quality criteria below, the pipeline accepts the DTW prediction (`source = "dtw_fallback"`). Extending the window on failure would let local errors compound into drift, the very failure mode of the greedy baseline. The trade-off is that DTW errors larger than W frames cannot be corrected by Phase B; this is acceptable because the DTW skeleton is typically accurate to within a few frames in confident regions, and the rescue mechanism of Phase B' handles large corrections in low-confidence regions where wider matcher offsets are coherent.

4.5.2 Per-Candidate Matching

For each candidate V_2 frame in the window, the matcher proceeds as follows:

1. Detect keypoints and compute descriptors in the V_1 frame (cached across candidates in the same window) and in the candidate V_2 frame.

2. Match descriptors via brute-force k-nearest-neighbour search with $k = 2$, using Hamming distance [20].

3. Apply Lowe's ratio test [25]: accept a match (m, n) only if $m.\text{distance} < r \cdot n.\text{distance}$ with $r = 0.75$. For descriptors where only one neighbour exists, accept if $m.\text{distance} < 50$.

4. Apply a spatial coherence filter: a V_1 keypoint located in the left half of the frame must match a V_2 keypoint in the left half, and analogously for the right half. Inserting this filter before RANSAC reduces RANSAC input by 5–15% on our data [15].

5. Fit a homography by RANSAC [31] with a reprojection threshold of 2.5 px, $p = 0.999$ and a maximum of two thousand iterations.

6. Sanity-check the homography scale: if the implied scale in either axis is below 0.7 or above 1.3, the inlier count is halved as a soft penalty.

7. Record the candidate's inlier count.

Figure 4.8 traces the surviving match count after each filter stage for a representative (V_1, V_2) pair, illustrating how the chain progressively prunes spurious matches.

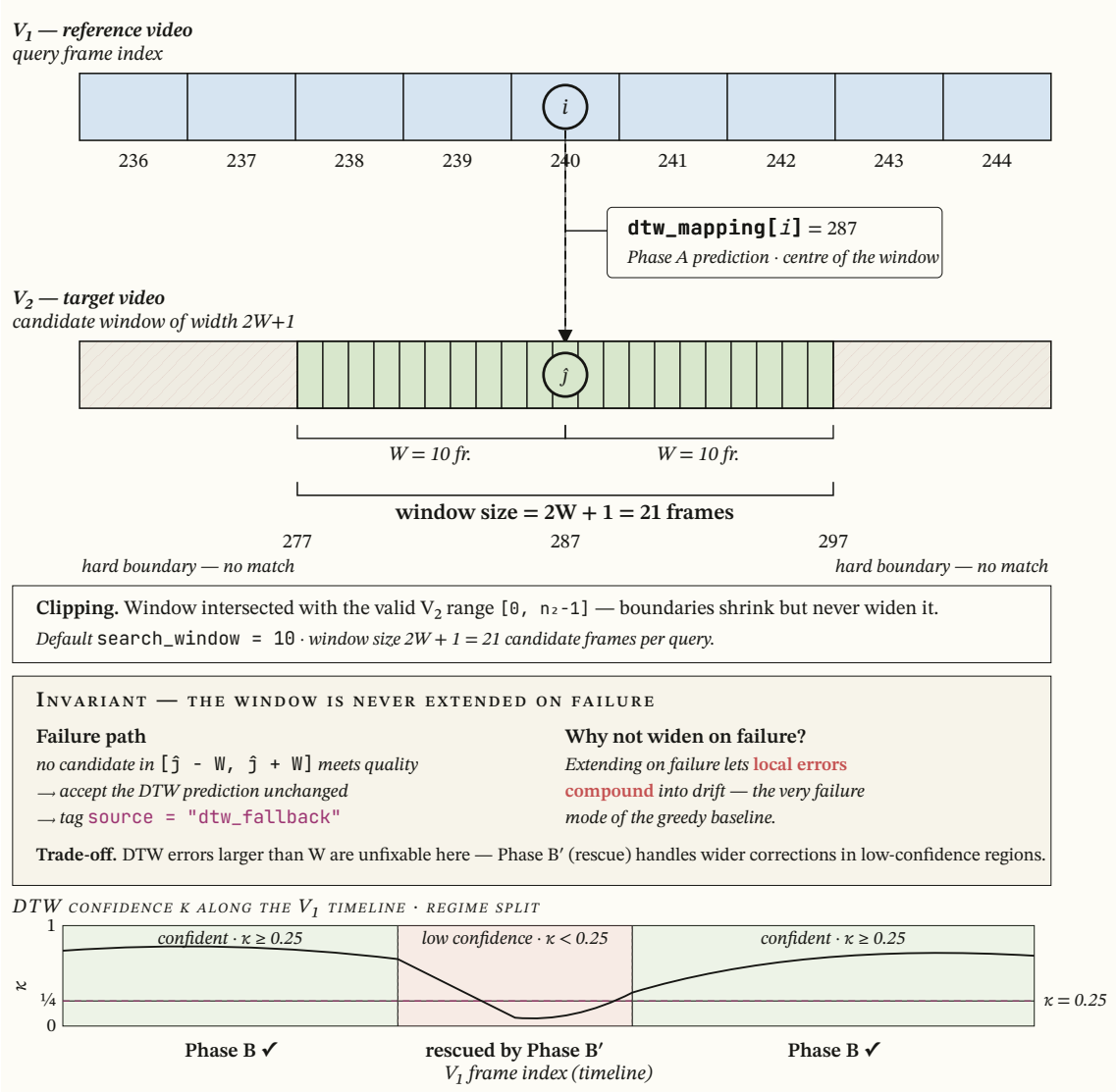

Figure 4.8: Per-candidate filter chain on a Plan 2 query frame i matched against three V_2 candidates within the ± 10 window. Each candidate passes through KNN ($k = 2$), Lowe's ratio, spatial coherence, RANSAC and the scale sanity check; the surviving match count is reported at each stage. Candidate B (offset 0) wins with 41 inliers; A is kept but down-weighted by the scale penalty; C dies at RANSAC for lack of a valid homography.

4.5.3 Weighted Scoring

For each candidate j in the search window of frame i, the inlier count alone is insufficient as a score. Two candidates with similar inlier counts but at different distances from the DTW prediction should not be ranked equally: the candidate close to the prediction is, all else equal, more trustworthy because it is consistent with the macro-skeleton. We capture this with a weighted score that decays exponentially with distance from the prediction:

$$s(i, j) = \text{inliers}(i, j) \cdot \exp\big(-k \cdot |j - \text{dtw\_mapping}[i]|\big), \tag{4.5}$$

with $k = 0.15$ (the parameter `dtw_distance_penalty_k`). The exponential form gives a factor of one at distance zero, approximately 0.47 at distance five, and approximately 0.22 at distance ten. A candidate at the edge of the window therefore needs roughly five times the inliers of a candidate at the centre to dominate it.

4.5.4 Top-K Candidates

Within each window, the candidates are sorted by weighted score and the top $K = 5$ are retained. Keeping multiple candidates per frame, rather than greedily picking the best one immediately [50], allows the global optimisation of Phase C to choose among them in a way that respects monotonicity across frames. Too small a value (e.g., $K = 1$) reduces Phase C to a no-op and reintroduces the trembling artefact; too large (e.g., $K = 20$) inflates the trellis without measurable improvement.

If the best candidate's inlier count falls below the threshold `min_inliers_threshold` (default four), the DTW prediction itself is injected as a top candidate with score zero. This guarantees that Phase C always has at least one valid candidate per frame: when feature matching fails completely, the pipeline returns the DTW prediction rather than a missing value.

4.6 Phase B': Rescue

Phase B' is the carefully circumscribed exception to the strict-window invariant. It corrects the DTW prediction in regions where the cost surface is unreliable (as indicated by the per-frame confidence of Phase A) and where the Phase B matcher has produced coherent contradictory evidence over multiple consecutive frames.

4.6.1 Premise

A per-anchor diagnostic on the Plan 2 ground truth revealed two DTW failure modes. The first is genuine: at some anchors the truth V_2 frame sits at the 80–90$^{\text{th}}$ percentile of its distance row, so the cost function does not reward the correct cell. This is a features problem, not a DTW problem. The second is recoverable: at other anchors the truth is the cheapest cell in its row, yet DTW picks a different cell because the dynamic programme

had already committed to a wrong path and monotonicity forbids backtracking. The rescue mechanism targets this second mode: the Phase B matcher operates per-frame without DTW's cumulative-path constraint, so when it consistently disagrees with DTW over many consecutive frames in a low-confidence region, its evidence is more trustworthy than DTW's.

4.6.2 Decision Logic

For each V_1 frame i, define the matcher's offset as

$$\delta_i = j_i^{\text{matcher}} - \text{dtw\_mapping}[i], \tag{4.6}$$

where j_i^{matcher} is the matcher's best candidate (before fallback). The rescue fires for frame i if and only if *all* of the following conditions hold:

1. Frame i is inside a contiguous run of at least twenty consecutive frames with $\kappa_{\text{frame}} < 0.25$.
2. The matcher returned at least six inliers for frame i.
3. Frame i is part of a contiguous run of at least five frames with same-signed offsets δ.
4. The standard deviation of δ across the run is at most eight frames.
5. The absolute value of the median δ across the run is at least twelve frames.

When the conditions are met, the DTW prediction is shifted by the median offset:

$$\text{dtw\_mapping}[i] \leftarrow \text{dtw\_mapping}[i] + \text{median}(\delta), \tag{4.7}$$

and the correction is propagated to the next thirty frames after the run, on the assumption that the systematic shift typically extends beyond its endpoint. Figure 4.9 illustrates the rescue decision chain on a Plan 2 zone, showing the DTW prediction, the matcher's per-frame offset, the median trigger, and the resulting correction.

4.6.3 Conservative Defaults

The five conditions are deliberately strict: trigger only in a long low-confidence zone (1); trust only matches with substantial RANSAC support (2); require directional consistency (3) and magnitude consistency (4); require the median offset to be well above the single-frame noise floor (5). Single-frame noise or a single drifted match cannot trigger the rescue. The design accepts that the rescue may miss real corrections rather than risk introducing artificial ones.

One operational coupling matters: the rescue can only detect deviations within the matcher's reach, bounded by W. With the default $W = 10$ the rescue's minimum-offset

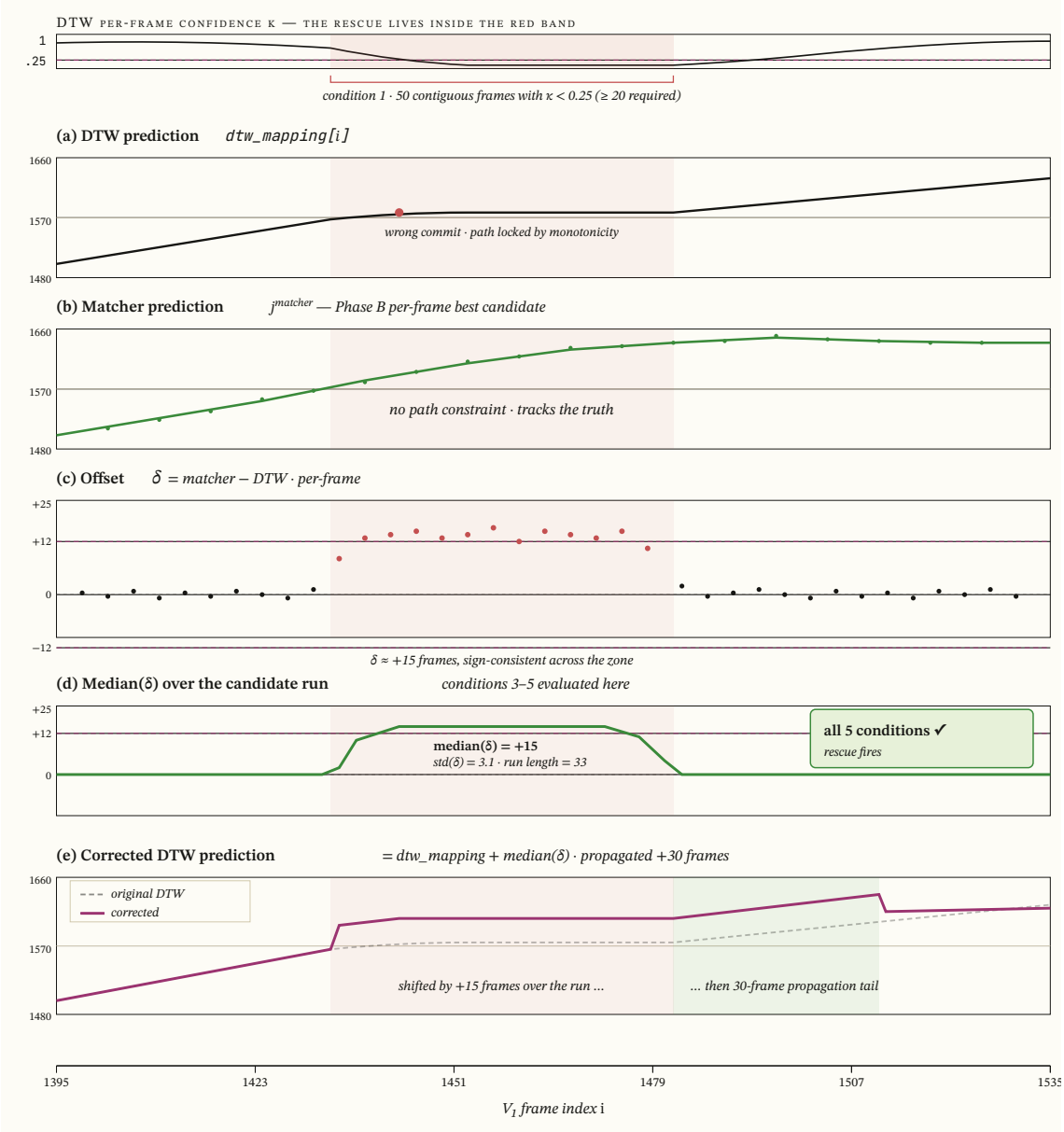

Figure 4.9: Rescue mechanism on a Plan 2 low-confidence zone (frames 1395–1535). (a) The DTW prediction stalls inside the low-confidence band (red). (b) The per-frame matcher keeps tracking the truth. (c) Their offset $\delta = j^{\text{matcher}} - \text{dtw\_mapping}$ collapses into a sign-consistent run at $\approx +15$ frames. (d) The running median of δ triggers all five rescue conditions. (e) The corrected mapping (magenta) shifts the original DTW (dashed) by +15 frames over the run and propagates the correction over the next 30 frames.

requirement of twelve frames is never met, so the rescue is silent. To activate it the user must pass `-search-window 30` or wider, a limitation documented in Chapter 7.

4.7 Phase C: Global Optimisation

Phase C takes the top-K candidates per frame produced by Phase B (possibly corrected by Phase B') and selects one candidate per frame so that the resulting sequence of V_2 indices is monotonically non-decreasing [13] and the cumulative weighted score is maximised. The phase is a dynamic programme over the candidate trellis.

4.7.1 Algorithm

Let $C_i = \{(j_1, s_1), (j_2, s_2), \ldots, (j_K, s_K)\}$ denote the top-K candidates at frame i, sorted by weighted score s. Define $\text{best}[i, c]$ as the best cumulative score over a path from frame 0 to frame i that ends at candidate c. The recurrence is

$$\text{best}[i, c] = s_c + \max_{p\,:\,j_p \le j_c} \text{best}[i-1, p], \tag{4.8}$$

with the convention $\text{best}[0, c] = s_c$. The optimal path is recovered by back-pointers from $\arg\max_c \text{best}[n_1 - 1, c]$ to $(0, \cdot)$ [36]. When no predecessor at frame $i-1$ satisfies $j_p \le j_c$ (rare in practice), the closest predecessor is accepted with the score halved. Figure 4.10 illustrates the candidate trellis for five consecutive Plan 1 frames, with the monotonicity constraint visualised as forbidden edges.

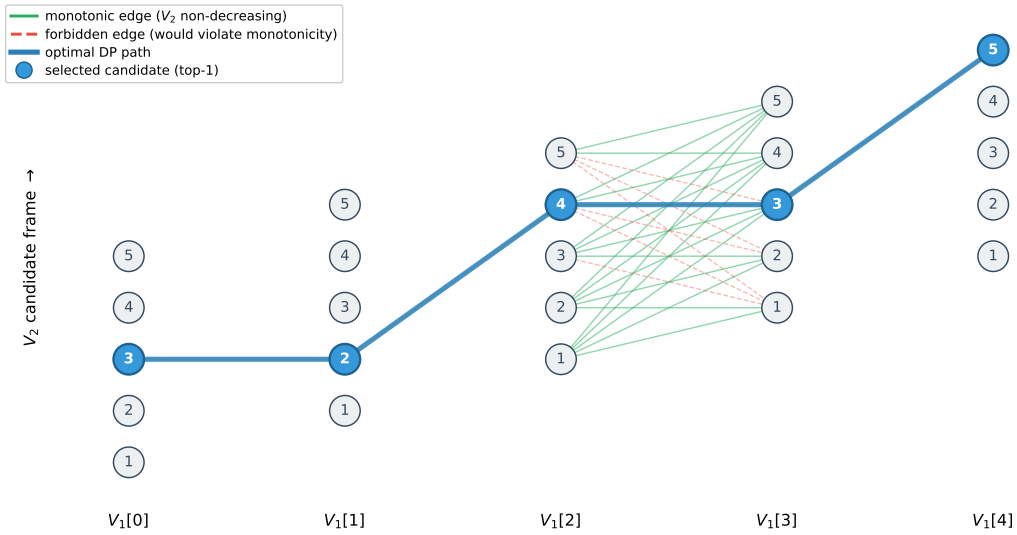

Figure 4.10: Top-K candidate trellis for five consecutive V_1 frames. At each index, the five candidates are stacked vertically; green edges connect candidate pairs whose V_2 index is non-decreasing (monotonic), red dashed edges are forbidden. The blue path is the optimal selection retrieved by the dynamic programme of Phase C.

4.7.2 Why This Matters

Without the global optimisation, per-frame greedy selection produces a visible *trembling* artefact: V_2 jumps back and forth between near-equivalent candidates. This is harmless at the inspection level but corrupts any downstream temporal differencing operation [13]. The monotonicity constraint formalises that the train moves forward, and is enforced by construction (zero violations on all three plans of the dataset).

4.8 Phase D: Smoothing

Phase D is the final polish. It applies a short weighted moving-average filter to absorb residual single-frame jitter from Phase C [38], then re-enforces monotonicity. For each frame i:

$$\text{smoothed}[i] = \frac{\sum_{j=i-w/2}^{i+w/2} v_2[j] \cdot \text{score}[j]}{\sum_{j=i-w/2}^{i+w/2} \text{score}[j]}, \tag{4.9}$$

with window size $w = 5$ and weights equal to the inlier counts (so high-confidence matches dominate). After the moving average, the array is walked once and any post-smoothing violation of monotonicity is clamped: $\text{smoothed}[i] \leftarrow \max(\text{smoothed}[i], \text{smoothed}[i-1])$. The final values are rounded to integer frame indices; pre-smoothing values are preserved in the output schema as `v2_frame_raw` for diagnostic purposes. The window $w = 5$ covers the typical ± 2-frame jitter while preserving genuine speed transitions; larger windows over-smooth the alignment and degrade the cycle consistency score reported in Chapter 5.

4.9 Implementation Notes

The algorithm of the previous sections is realised in a Python implementation organised into three generations of code, with the current generation surrounded by a dedicated diagnostic subsystem. This section documents the engineering decisions that surround the algorithm proper.

4.9.1 Algorithm-Agnostic Interface

The choice of inner feature-detection algorithm is a single configuration parameter of the `HybridAligner` class. The OpenCV detector is initialised at construction via `cv2.AKAZE_create`, `cv2.BRISK_create` or `cv2.ORB_create`; all three return objects with identical interfaces and produce Hamming-distance binary descriptors [20], so the same `cv2.BFMatcher` with `NORM_HAMMING` works for all three. This interface design makes the comparison study of Chapter 5 possible: each algorithm is evaluated on the same data with the same surrounding pipeline, and the findings of that study are therefore attributable to the descriptor choice alone, not to incidental implementation differences.

4.9.2 Three Packages, Three Generations

The repository preserves three generations of code under three packages: `Old_version/feature_matching/` (the original greedy-search baseline with a Kalman velocity tracker, kept for ablation), `Old_version/new_method/` (the DTW-only second generation, whose DTW core is reused by the production pipeline), and `combined_method/` (the current generation, with the algorithm in `hybrid_alignment.py` and auxiliary modules `preprocessing.py`, `dtw_confidence.py` and `phase_b_rescue.py` for Phases 0, A.4 and B').

4.9.3 CPU-Only Design

The pipeline runs entirely on a commodity CPU, with no GPU at inference time and no pretrained model whose weights have to be downloaded. The constraint drives three decisions: the elimination of learning-based detectors (Section 4.2), the use of binary rather than floating-point descriptors [24], and the restriction of pretrained models to evaluation time only (LPIPS [43] falls back to CPU inference; it is used for scoring, never for alignment).

4.9.4 Default Parameters

Table 4.1 summarises the parameters of the pipeline and the values used in the production configuration. Each value is either justified by the sweep reported earlier in this chapter or by a separate small ablation reported in Chapter 5.

Table 4.1: Pipeline default parameters. Phase labels: 0 = LUT preprocessing, A = DTW coarse alignment, B = feature-matcher fine pass, B′ = DTW-fallback rescue, C = top-K trellis DP, D = post-smoothing.

Parameter	Default	Phase	Role
`dtw_sample_rate`	5	A	Frame sub-sampling step for the DTW grid.
`dtw_step_penalty`	0.3	A	Extra cost on non-diagonal DTW steps.
`search_window`	10	B	Half-window (V2 frames) for the matcher search around the DTW prediction.
`top_k_candidates`	5	C	Candidates kept per frame for the monotonic trellis DP.
`dtw_distance_penalty_k`	0.15	B	Decay rate of $\exp(-k\,d)$ weighting candidates by distance to DTW.
`min_inliers_threshold`	4	B / B′	RANSAC inlier floor to accept a match (homography requires ≥ 4 points).
`lowe_ratio`	0.75	B	Lowe ratio test on the two nearest descriptors [25].
`ransac_reproj_threshold`	2.5	B	Maximum reprojection error (px) for a RANSAC inlier.
`smoothing_window`	5	D	Median/weighted smoothing window on the final mapping.
`enable_preprocessing`	True	0	Activates the per-channel LUT contrast preprocessing.
`enable_phase_b_rescue`	True	B′	Activates the rescue pass over runs of low-κ DTW frames.

Sensitivity. Only three knobs materially shift the output. `search_window` is the most

sensitive: too small drops valid matches, too large readmits drift, and the chosen value of 10 tracks the κ confidence budget reported in Section 4.4.4. `dtw_step_penalty` controls how aggressively the DTW path leaves the diagonal; values outside $[0.2, 0.4]$ degrade either way. `min_inliers_threshold` is a hard floor at 4 (homography is mathematically underdetermined below). All other parameters were verified to be insensitive within $\pm 30\%$ of their default during the calibration sweep; the two boolean flags toggle ablations and default to `True`.

4.10 Summary

This chapter presented the six-phase pipeline: luminance preprocessing, DTW skeleton over gradient orientation histograms with a per-frame confidence score, feature-matching refinement within a strict window, rescue step in low-confidence regions, global DP over top-K candidates for monotonicity, and weighted smoothing. The inner matcher (AKAZE, BRISK or ORB) is interchangeable, which enables the comparison study of the next chapter. The pipeline runs entirely on a commodity CPU.

The three contributions claimed in Chapter 1 are now substantiated: the gradient-based feature extractor (Section 4.4.1), the dual-use histogram-matching mechanism (Section 4.3, revisited in the evaluation chapter for its second use), and the path-confidence-driven rescue (Section 4.6). Chapter 5 evaluates the pipeline on three rail-track sequences and reports the findings.

5. Experiments and Results

This chapter presents the empirical evaluation of the hybrid pipeline of Chapter 4, organised by axis of evaluation rather than by plan. Sections 5.1–5.3 cover the dataset, alignment-quality metrics, and the comparison of the three inner matchers. Sections 5.4–5.5 present the main negative finding (SSIM/LPIPS saturation) and establish cycle consistency as the discriminating metric. Section 5.6 is a Plan 2 ground-truth case study including the design-selection sweep; Section 5.7 maps the findings back to the research questions of Chapter 1.

5.1 Dataset

The dataset consists of three plans, each composed of two short videos V_1 and V_2 that cover approximately the same physical stretch of railway track. The source material is two one-hour recordings made by an on-board forward-facing camera on the line between Bruxelles-Midi and Halle, on different days. From each recording, three one-minute-thirty clips were extracted and paired into the three plans labelled Plan 1, Plan 2 and Plan 3.

5.1.1 Common Characteristics

Three properties are shared across all plans. V_1's camera is slightly more zoomed out than V_2's. The two recordings have small but non-negligible luminance differences from time-of-day and weather (motivating the Phase 0 preprocessing of Section 4.3). The trackside content is highly repetitive: rails, sleepers, masts and overhead lines repeat at periods of metres or seconds, the structural property the gradient features of Section 4.4.1 are designed to handle.

5.1.2 Three Plans of Increasing Difficulty

The three plans form a deliberate gradient of difficulty.

Plan 1 (baseline). V_2 is roughly twenty percent faster than V_1 throughout, with an approximately constant relative speed ratio. The easy case: high refined fraction and high alignment quality expected across all algorithms.

Plan 2 (variable speed). Two phases: in the first, V_2 is faster than V_1 as in Plan 1; midway through, V_1 abruptly accelerates and becomes significantly faster than V_2, inverting the relative speed. Two additional localised perturbations: V_2 passes under a long bridge that momentarily blinds the camera, and V_2 crosses another train on the parallel track at a moment V_1 does not. Plan 2 is also the plan for which fourteen manual ground-truth anchors are annotated, used for the design selection of Section 4.4.1 and the per-anchor diagnostic of Section 5.6.

Plan 3 (limit case). V_2 is faster than V_1 throughout with locally variable but always positive differential. Midway, V_1's train changes track (substantial scenery change) and decelerates while V_2 maintains speed, widening the speed differential exactly when content becomes ambiguous. Both trains also cross other trains at different moments, so each crossing dominates one stream without counterpart. The conjunction pushes the algorithm beyond what it can resolve on a CPU. Plan 3 is the empirical limit case of the design.
Figure 5.1 shows representative V_1/V_2 frame pairs from each of the three plans, illustrating the camera-zoom, lighting and content differences described above.

5.2 Alignment-Quality Metrics

We begin with metrics produced at alignment time, before perceptual or cycle scoring: refined fraction, monotonicity, velocity ratios and feature-matcher inlier counts.

5.2.1 Refined Fraction

Each output frame is tagged with the source of its V2 prediction: `<algo>_refined` when the feature matcher produced a confident match, or `dtw_fallback` when the matcher could not find an acceptable candidate within the search window. Table 5.1 reports the refined fraction per (plan, algorithm).
The refined fraction is high on Plan 1 (83–92%), moderate on Plan 2 (61–81%), lower on Plan 3 (54–71%), tracking the difficulty ranking of Section 5.1.2. Within-plan algorithm comparison is consistent: ORB leads on Plan 1 (91.7%) and Plan 2 (80.6%), BRISK on Plan 3 (71.4%), with five-to-ten-percentage-point spreads. On no plan does the refined fraction drop below 50%, so even on the hardest plan the inner matcher contributes on more than half the frames.

5.2.2 Monotonicity

The monotonicity score is the fraction of consecutive output pairs where V_2 does not decrease, formalising the physical constraint that trains move forward in both recordings. On all nine (plan, algorithm) combinations the monotonicity is exactly 100.00%. Phase C enforces this by construction (Section 4.7) and Phase D re-clamps any post-smoothing violations. This is not a small property: a non-monotonic alignment would map the

Figure 5.1: Representative frame pairs from the three plans, all sampled at V_1 frame 500. Each row contrasts V_1 (left) and V_2 (right) for one plan, illustrating the camera-mounting, lighting and content differences between recording days.

Table 5.1: Per-(plan, algorithm) alignment-quality summary extracted from the master report (`alignment_quality.csv`). *Total*: number of V1 frames aligned. *Refined %*: share of frames whose final position was confirmed or corrected by the feature matcher (Phase B). *Fallback %*: complementary share kept on the DTW prediction (Phase B′ rescue or low-inlier cases). *Monot.%*: percentage of monotonic transitions. *Mean inl.*: average number of RANSAC inliers on refined frames. *Max corr.*: largest absolute shift (in frames) applied on top of the DTW skeleton. Within each plan, the best score per column (Refined % and Mean inl., higher is better) is in **bold** and the second-best is in *italics*.

Plan	Algo	Total	Refined %	Fallback %	Monot. %	Mean inl.	Max corr.
Plan1	AKAZE	2 711	83.25	16.75	100	7.81	8
Plan1	BRISK	2 711	*91.26*	8.74	100	*8.50*	5
Plan1	ORB	2 711	**91.66**	8.34	100	**9.19**	8
Plan2	AKAZE	2 211	61.24	38.76	100	7.56	10
Plan2	BRISK	2 211	*67.66*	32.34	100	7.25	11
Plan2	ORB	2 211	**80.64**	19.36	100	**12.53**	15
Plan3	AKAZE	2 726	58.55	41.45	100	*9.51*	9
Plan3	BRISK	2 726	**71.39**	28.61	100	7.97	9
Plan3	ORB	2 726	*54.37*	45.63	100	**10.39**	10

same physical location to multiple non-consecutive V_2 frames, directly corrupting any downstream change-detection step.

5.2.3 Mean Inliers and Maximum Correction

The mean inlier count on refined matches ranges from 7.25 to 12.53 across the matrix. The maximum correction (largest deviation between DTW prediction and refined output) ranges from 5 to 15 frames, bounded by the strict search window (default $W = 10$, larger only when the Phase B' rescue fires). The combination of moderate mean inliers (comfortably above the threshold of four) and bounded maximum correction is the empirical signature of a well-behaved feature-matching phase: the matcher finds enough evidence to make confident decisions, and those decisions remain anchored to the DTW skeleton. Figure 5.2 visualises the four alignment-quality metrics (refined %, monotonicity, mean inliers, max correction) as bars grouped by algorithm and faceted by plan.

5.3 Comparison of the Three Inner Matchers

Whether AKAZE, BRISK or ORB has a measurable effect on the final alignment was a central question of Chapter 4. The empirical answer is that the three algorithms perform within statistical noise of each other on the metrics that meaningfully discriminate quality. On SSIM and LPIPS (Section 5.4) within-plan variation is at the fourth decimal, below frame-sample noise. On cycle consistency (Section 5.5) the differences are slightly visible but remain small: Plan 1, ORB 4.89 frames vs AKAZE 5.02, BRISK 6.62; Plan 2, ORB

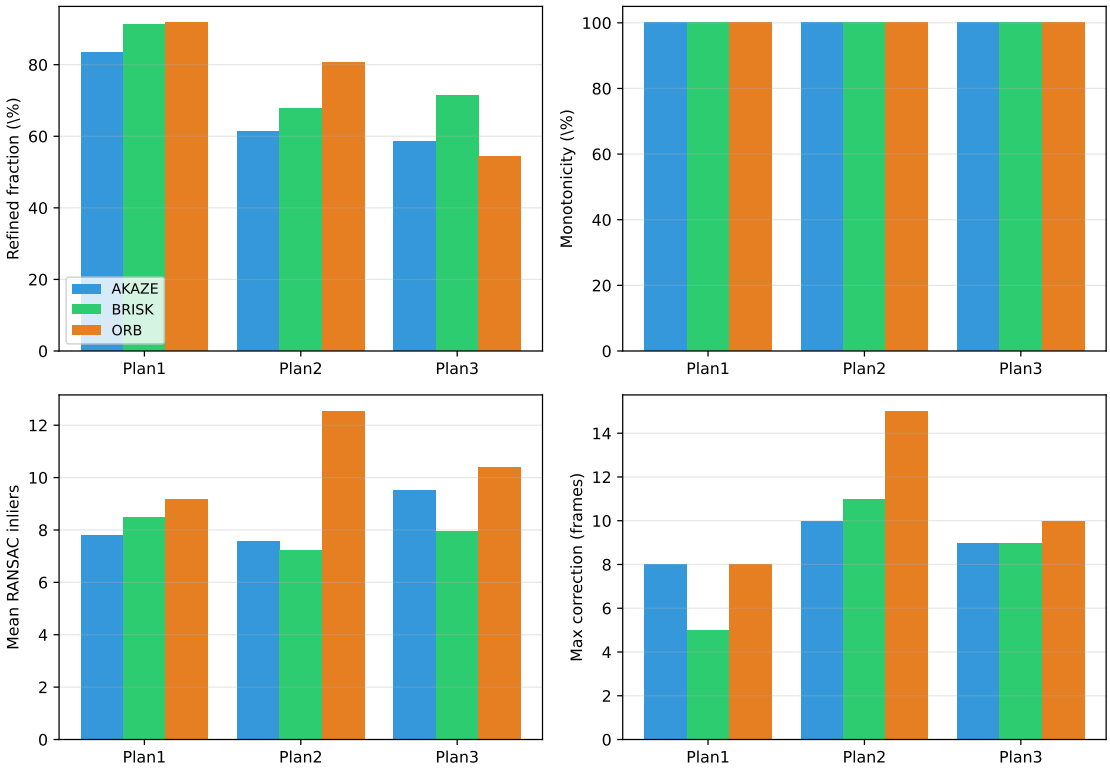

Figure 5.2: Four alignment-quality metrics on the nine (plan, algorithm) combinations: refined fraction, monotonicity, mean RANSAC inliers, and maximum correction. Bars are grouped by plan and coloured by algorithm (AKAZE blue, BRISK green, ORB orange); monotonicity is exactly 100% on every configuration.

12.66 vs AKAZE 13.38, BRISK 13.35; Plan 3, the three tie around 21 frames. ORB has a marginal advantage on the easier plans but it is too small and too plan-dependent to be a recommendation. The within-five-frames fraction tells the same story: Plan 1 ORB 79.1% and AKAZE 80.2%; Plan 2 ORB 62.2% vs AKAZE and BRISK around 53%; Plan 3 the three cluster near 28%.

Two readings. Optimistic: the pipeline is robust to the matcher and an operator may pick any. Cautious: the surrounding pipeline (DTW skeleton, global optimisation, smoothing) dominates the matcher choice so much that it is no longer the limiting factor. Either way, future engineering leverage is on DTW input features or the rescue, not on the inner matcher.

5.4 Perceptual Metrics: SSIM and LPIPS

We evaluate with SSIM [42] and LPIPS [43], computed on a uniform 10% sample of aligned pairs (roughly 220–270 per plan) with the photometric normalisation of Chapter 4.

5.4.1 Headline Values

Mean SSIM and LPIPS per plan are reported in Figure 5.3 and Figure 5.4. Averaged across algorithms, mean SSIM is 0.3713 on Plan 1, 0.3790 on Plan 2 and 0.4618 on Plan 3; mean LPIPS is 0.7152, 0.7214 and 0.6763 respectively. Two distinct phenomena bound these numbers.

The *absolute values* are bounded by a photometric and geometric baseline between V_1 and V_2. The two videos were recorded on different days under different sunlight, weather and slightly different camera mounting and zoom, so even a geometrically perfect alignment cannot reach SSIM $= 1$ or LPIPS $= 0$: the two cameras simply did not see identical pixels.

The *inter-algorithm spread* is bounded by a temporal-proximity effect. Within each plan, the three matchers differ only at the fourth decimal (spread of order 10^{-4} on both metrics). This is because the strict ten-frame search window of Phase B keeps AKAZE, BRISK and ORB within a few frames of each other on their final picks: Table 5.1 reports a maximum correction of 5–15 frames across all nine configurations, and the typical inter-algorithm difference is smaller. At 30 fps, a handful of frames corresponds to a few tenths of a second of forward train motion, during which the forward-facing camera sees an essentially unchanged scene of rails and ballast. Pixel-wise metrics such as SSIM and LPIPS cannot resolve frame pairs separated by such small temporal gaps, so they cannot distinguish algorithms whose picks differ by such small amounts.

5.4.2 The Saturation Problem

The headline values mean little in isolation. The question is whether SSIM and LPIPS distinguish a good alignment from a bad one. We compare the metric on the real alignment

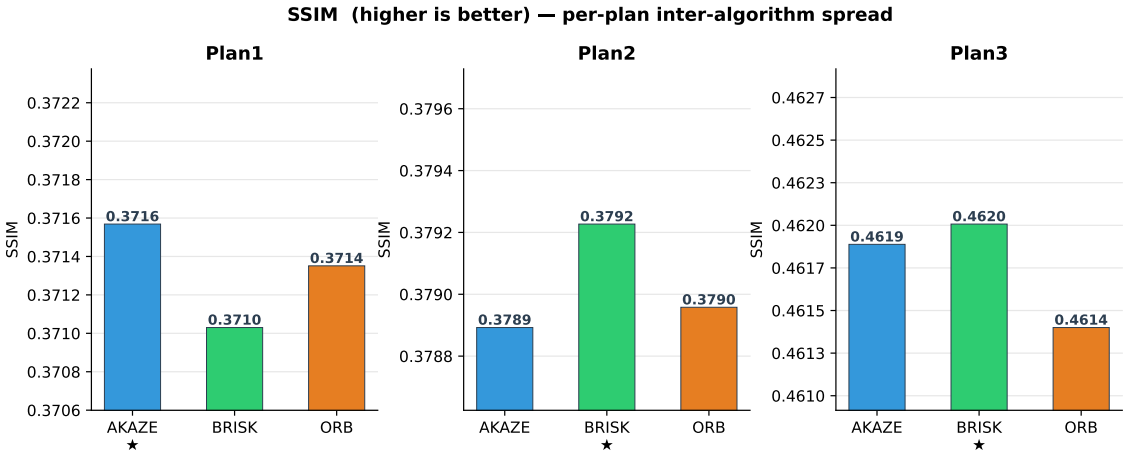

Figure 5.3: Mean SSIM per matching algorithm on the real alignment, one panel per plan. Higher is better. The ⋆ marks the algorithm with the highest SSIM in each plan. Each panel is zoomed on the actual inter-algorithm spread, which lies at the fourth decimal place across all three plans; the "winning" algorithm changes from plan to plan, with no systematic dominance.

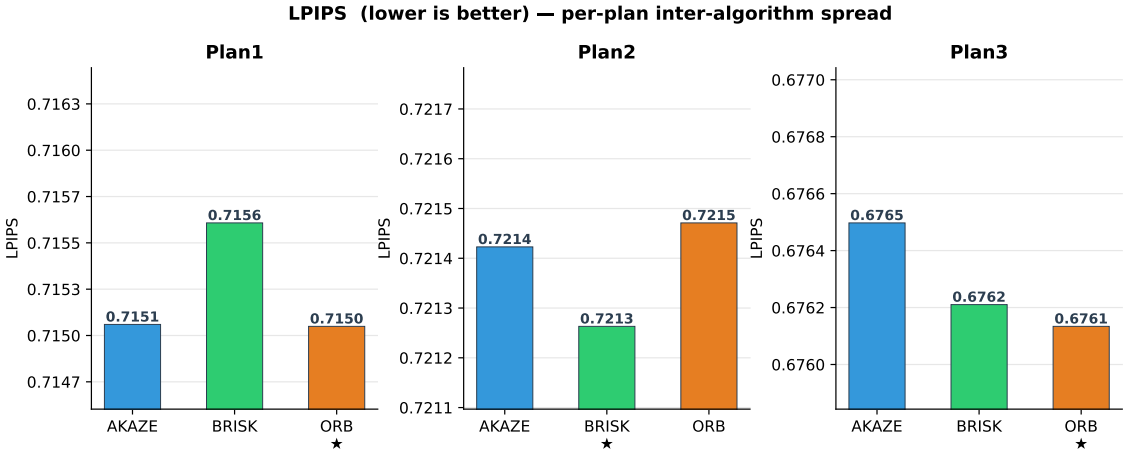

Figure 5.4: Mean LPIPS per matching algorithm on the real alignment, one panel per plan. Lower is better. The ⋆ marks the algorithm with the lowest LPIPS in each plan. As with SSIM, the inter-algorithm spread is microscopic (fourth decimal place) and the winner varies between plans.

to the metric on a uniformly random alignment. If the metric drops substantially under randomisation, it carries signal; if it stays close, it is saturated.
We define the discrimination index of a metric on a (plan, algorithm) combination as

$$\delta = \frac{\left|\mu_{\text{metric}}(\text{random}) - \mu_{\text{metric}}(\text{real})\right|}{\left|\mu_{\text{metric}}(\text{real})\right|}. \tag{5.1}$$

A value above one means randomisation moves the metric more than its baseline; a value below 10^{-2} means saturation.
Table 5.2 reports the empirical discrimination indices. SSIM ratios range from 0.001 to 0.037, LPIPS from 0.001 to 0.013. All values lie between 10^{-3} and 10^{-2}, two to three orders of magnitude below the threshold for decision-relevant information. Both metrics are saturated on this dataset.

Table 5.2: Discrimination index $\delta = |\mu_{\text{random}} - \mu_{\text{real}}|/|\mu_{\text{real}}|$ for the three quality metrics, computed per (plan, algorithm). Values close to zero indicate metric saturation; larger values indicate stronger separation between a real alignment and a random one. SSIM and LPIPS sit in the 10^{-3}–10^{-2} range (red columns, effectively saturated), whereas cycle consistency reaches 10^{1}–10^{2} on every plan (green column), i.e. three to five orders of magnitude better. Within each plan, the best Cycle δ is in **bold** and the second-best in *italics*.

Plan	**Algo**	**SSIM δ**	**LPIPS δ**	**Cycle δ**
Plan1	AKAZE	1.39×10^{-2}	1.2×10^{-3}	*184.1*
Plan1	BRISK	1.54×10^{-2}	2.0×10^{-3}	139.5
Plan1	ORB	1.45×10^{-2}	1.2×10^{-3}	**189.1**
Plan2	AKAZE	3.60×10^{-2}	7.9×10^{-3}	68.3
Plan2	BRISK	3.69×10^{-2}	8.1×10^{-3}	*68.5*
Plan2	ORB	3.62×10^{-2}	7.8×10^{-3}	**72.3**
Plan3	AKAZE	1.3×10^{-3}	1.27×10^{-2}	**40.8**
Plan3	BRISK	1.1×10^{-3}	1.32×10^{-2}	40.0
Plan3	ORB	2.4×10^{-3}	1.33×10^{-2}	*40.3*

The cause is the photometric ceiling: two frames of the same location on different days are not identical to SSIM or LPIPS, even after histogram normalisation, and the residual is of the same order as between randomly paired frames of the same general scene. Both metrics were designed for single-image distortions (compression, blur) where the photometric ceiling is much lower; applied to two-recording alignment they are outside their regime. The metrics are not wrong, just uninformative on this task.

5.4.3 Practical Implication

A railway alignment pipeline cannot be optimised against SSIM or LPIPS: a sweep ranking configurations by SSIM would rank them by photometric noise. Future work should

justify the discriminative power of any chosen metric on its own data rather than assume single-image quality metrics transfer to two-pass alignment.

5.5 Cycle Consistency

The third metric is cycle consistency, originally from image-to-image translation [46] and reused here as a self-supervised probe. The forward alignment $f : V_1 \to V_2$ composed with the backward alignment $g : V_2 \to V_1$ should approximately recover the identity on V_1: for every sampled frame i, $|i - g(f(i))|$ should be small. The metric is content-agnostic, so immune to the photometric ceiling that saturates SSIM and LPIPS.

5.5.1 Real Alignment

Mean cycle error on the real alignment is 5.0, 13.0, 21.0 frames on Plans 1, 2, 3 averaged across algorithms; the within-five-frames fraction is 80%, 62%, 30%. The ordering tracks the difficulty ranking of Section 5.2.1: easy plan near-frame-perfect, intermediate plan well within bounds, limit-case plan degraded but not collapsed. The 21-frame cycle error on Plan 3 corresponds to roughly 0.7 seconds (ten to twenty metres of track); a downstream change-detection module would see motion artefacts on Plan 3 that filtering would have to remove. We do not claim sub-second precision on Plan 3, consistent with its limit-case classification.

5.5.2 Baseline Comparison

The headline finding emerges against baselines. Mean cycle error on a random alignment is roughly 880–930 frames per plan, so the real alignment is forty to one hundred ninety times smaller. The discrimination index of equation 5.1 ranges from 40 (Plan 3) to 189 (Plan 1), three to five orders of magnitude above the SSIM and LPIPS values of Section 5.4.2. Figure 5.5 plots the mean cycle error for the real alignment against four baselines (random, linear stretch, offset by 1/5/30 frames) on a symmetric-log scale, making the multi-order-of-magnitude gap immediately visible.

Two baselines warrant commentary. The linear baseline (proportional stretch) cycles to zero by construction, a known degeneracy: cycle catches noise (random) but is blind to systematic bias (proportional or constant offset). The offset baselines (shift by $k = 1, 5, 30$) cycle to approximately k frames as expected. Both degeneracies are predictable; the discrimination against random alignment, in contrast, is genuine.

5.5.3 Cycle as the Discriminating Metric

Cycle consistency is the metric that meaningfully tracks alignment quality on this dataset. The 40-to-180 separation from random confirms the pipeline produces alignments substantively different from chance; the linear and offset baselines confirm proportional response to perturbation; the three orders of magnitude advantage over SSIM and LPIPS confirms

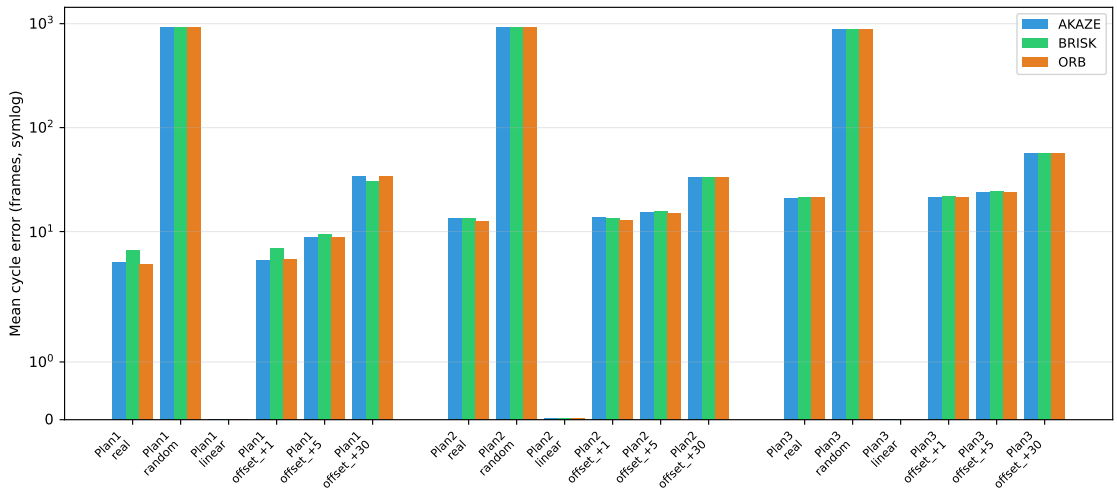

Figure 5.5: Mean cycle error per (plan, algorithm) across the real alignment and four baselines (random, linear stretch, offsets +1/+5/+30 frames), on a symmetric-log scale. The three-orders-of-magnitude separation between real ($\sim$ 5–21 frames) and random ($\sim$ 900 frames) is the discriminative gap that SSIM and LPIPS, saturated by the photometric ceiling, fail to capture.

the saturation finding. This answers RQ3: cycle consistency is the metric to use, SSIM and LPIPS should not be reported in isolation, and any future work in this domain should justify the discriminative power of any reported metric before relying on it.

5.6 Plan 2 Ground-Truth Case Study

Plan 2 is the only plan with hand-annotated ground truth: fourteen (v_1, v_2) pairs identifying the same geographic location in V_1 and V_2. This ground truth was the empirical foundation of the design selection of Chapter 4.

5.6.1 Design-Selection Sweep

We swept the cartesian product of nine extractors, three normalisations (raw, per-video z-score, joint z-score) and four step penalties (0.1, 0.3, 0.5, 1.0): 108 configurations, each scored by mean absolute error of the DTW output against the fourteen Plan 2 anchors. The resulting heatmap of mean absolute error per (extractor, normalisation) pair is shown in Figure 5.6, with the best cell highlighted.

Three findings. The top of the leaderboard is dominated by gradient-orientation histograms with per-video z-score normalisation: the best three configurations all use `grad_hist_16 + per_video_zscore` at 33.1 frames mean absolute error (step penalties 0.1 and 0.3) and 50.0 (0.5). The first non-gradient extractor (intensity histogram, position twelve) is at 74 frames, more than twice the best. The same extractor with different normalisations gives substantively different results: `grad_hist_16` with joint z-score lands at 65 frames, raw at 68, vs 33 for per-video z-score; normalisation alone contributes a factor of two. The

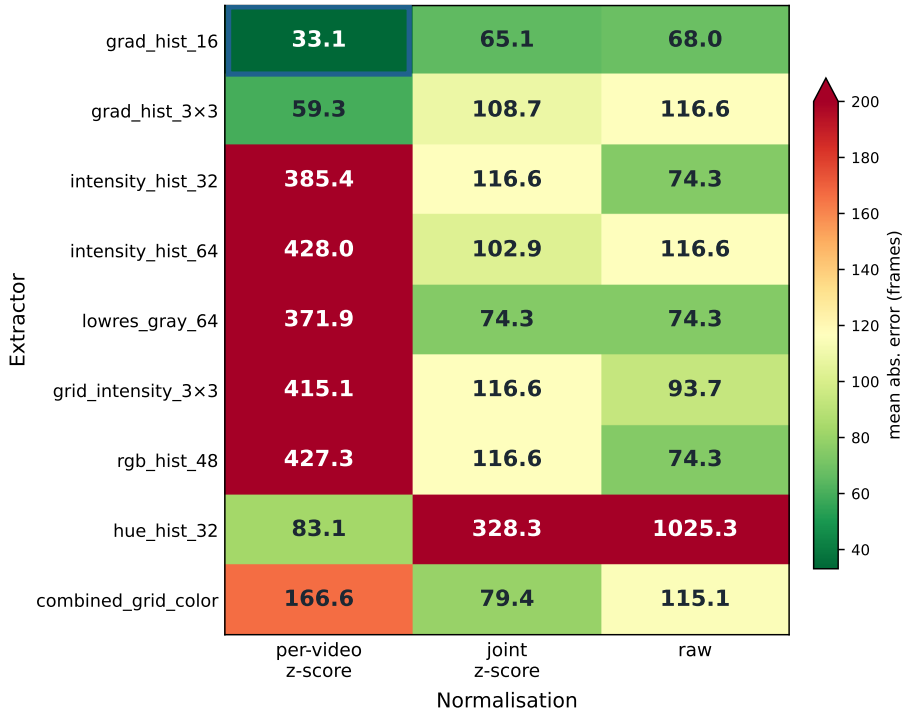

Figure 5.6: Mean absolute alignment error (in frames) on the Plan 2 ground truth across the cartesian product of nine extractors (rows) and three normalisation regimes (columns). The best cell — `grad_hist_16` with per-video z-score, 33.1 frames — is highlighted; gradient-histogram extractors with per-video normalisation dominate the low-error region.

step penalty contributes less: penalties 0.1 and 0.3 give identical mean error (33.1), 0.5 increases to 50.0. The production default of 0.3 sits on the safe plateau.

5.6.2 Per-Anchor Diagnostic

Beyond the headline mean error, the per-anchor table records the DTW prediction, the absolute error, the distance at the truth, the row minimum and the percentile of the truth in its distance row. The mean percentile of the truth is 31.8%, with five of fourteen anchors at $\geq 30\%$. A percentile of zero would mean the truth is the cheapest cell in its row, where DTW would strongly prefer it. 30% means the truth is cheaper than only 70% of the candidates, and the dynamic programme can easily prefer a different cell if the resulting accumulated cost is lower along the path. The five anchors above 30% are evidence that features themselves remain the bottleneck on these positions, irrespective of any DTW tweak.

The path-cost comparison supports this: mean cost per step is 3.48 along the DTW path against 4.20 along the ground-truth path. The cost function rewards the wrong path; minimising it cannot reach the truth. This is the residual gap the Phase B' rescue attempts to bridge and that the spatial gradient extractor of Chapter 7 would partially close. Figure 5.7 visualises, for each Plan 2 anchor, the position of the ground-truth cell within its distance row, together with the histogram of truth percentiles across the fourteen

anchors.

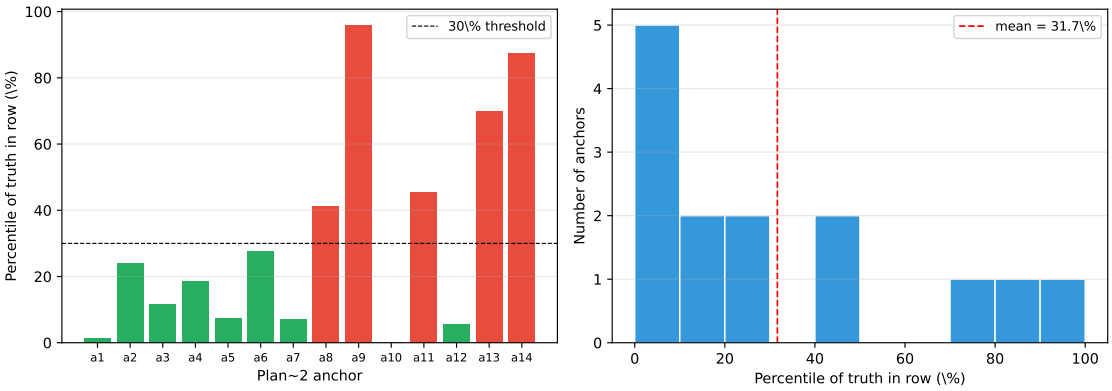

Figure 5.7: Per-anchor diagnostic on Plan 2. Left: percentile of the ground-truth V_2 frame within its distance row, for each of the fourteen anchors (green if below the 30% threshold, red if above). Right: histogram of these percentiles, with the mean at 31.8% — evidence that several anchors sit far from the row minimum that DTW would prefer.

5.6.3 Hypothesis-Driven Diagnosis

The diagnostic tool documented in Chapter 6 walks through six hypotheses about the failure modes. Applied to the production configuration on Plan 2, the answers are:

- **H1, features discriminative at anchor rows?** No: mean percentile of truth is 31.8%, well above the 5% threshold for a clearly discriminative feature.
- **H2, is the truth path globally cheap?** No: DTW path mean cost per step (3.48) is lower than ground-truth path (4.20), so DTW is correctly minimising a cost function that does not reward the truth.
- **H3, DTW endpoint correct?** Approximately: DTW endpoint is $V_1[2040]$ vs ground-truth endpoint $V_1[2273]$, off by 233 frames at the boundary.
- **H4, healthy step distribution?** Yes: 74% of steps are diagonal, only 25% are deletes, no inserts, no other.
- **H5, V1-V2 anchor pairs cluster in PCA space?** Mixed.
- **H6, rescue zones identified?** Yes: five rescue zones detected by the confidence-driven mechanism of Phase B', whose locations correlate with the anchors of largest error.

The dominant conclusion is the conjunction of H1 and H2: features remain the residual bottleneck. The DTW algorithm, open-end termination, step penalty, rescue mechanism, spatial filter and RANSAC chain all function correctly, but they operate on a cost surface that does not always reward the geographically correct cell. The next major quality improvement comes from a better feature, not a better algorithm.

5.7 Summary of Findings

We close the chapter by mapping the experimental results to the four research questions of Chapter 1.

RQ1, coarse-to-fine alignment on CPU? Empirically yes. Monotonicity is exactly 100% on every (plan, algorithm), mean cycle error is bounded by 21 frames on the limit-case plan, refined fraction is above 50% on every configuration, and the alignment runs in three to five minutes per video pair on a laptop CPU.

RQ2, choice of inner matcher? The three algorithms perform within statistical noise on perceptual metrics and within five percentage points on within-five-frames cycle. ORB has a marginal advantage on the easier plans; the three equalise on the limit case. The deeper finding is that the matcher is no longer dominant: the DTW skeleton and global optimisation absorb most of the algorithmic responsibility, and future engineering has more leverage on DTW input features than on the inner matcher.

RQ3, discriminating metrics? SSIM and LPIPS, even after photometric normalisation, are saturated: discrimination indices against random are 10^{-3}–10^{-2}, two to three orders of magnitude too small to distinguish good from random. Cycle consistency, by contrast, ranges from 40 to 189, three to five orders of magnitude larger. Cycle consistency is the metric to report; SSIM and LPIPS should not be reported in isolation on this kind of data.

RQ4, graceful degradation? The degradation along the difficulty gradient is graceful and well-understood. Refined fraction drops from 92% to 71%, cycle error grows from 5 to 21 frames, but monotonicity holds at 100% throughout and the pipeline never produces silently wrong output. The Plan 2 diagnostic identifies the feature extractor as the residual bottleneck: even with the best feature within the design constraints, several anchors have the truth above the thirtieth percentile of their distance row. The next major quality improvement comes from a richer feature (a spatial gradient grid, or a learned embedding under a relaxed deployment constraint), not from a different DTW or matcher.
The pipeline works under the stated constraints, the inner matcher choice is a deliverability lever rather than a quality lever, cycle consistency is the operational metric of this domain, and the residual quality is feature-bounded. Chapter 6 documents the visualisation layer; Chapter 7 synthesises the contributions and outlines future work.

6. Visualization and Diagnostic Tooling

A temporal alignment pipeline operates as a black box if its outputs are restricted to raw frame correspondences. Understanding where the pipeline succeeds, where it struggles, and how parameter selections alter the cost surface requires a dedicated visualization and diagnostic layer. This chapter documents the standalone production reports and the automated diagnostic subsystem developed to support design iterations.

6.1 Production Artefacts and Reporting

A standard alignment run automatically generates a self-contained output directory structured to serve both downstream automated modules and human operators.

6.1.1 Visual and Automated Outputs

To enable rapid visual verification and integration, the pipeline produces five standardized plots alongside three synchronous, side-by-side verification videos. The roles and targets of these outputs are synthesized in Table 6.1.

The numerical and canonical machine-readable results are consolidated into a structured `alignment_results.csv` file (mapping V_1 frames directly to V_2 predictions along with localized matching scores) and a stable `metrics.json` summary for automated dashboard ingestion. An automated generator compiles these assets into a standalone, configuration-driven HTML report (`report.html`) requiring no local Python dependencies, optimizing operational deliverability. Figure 6.1 shows the top of this unified HTML dashboard, where the five plots and three side-by-side videos are accessible from a single browser-viewable document.

6.2 The Diagnostic Subsystem

While production artefacts confirm alignment validity, debugging localized tracking failures requires a deeper inspection of the intermediate cost surfaces. The diagnostic subsystem

Table 6.1: Synthesis of the standard production outputs generated by the alignment pipeline.

Output Artefact	**Visual / Technical Content**	**Target Objective**
Alignment Scatter	$(i, f(i))$ points colour-coded by matching source (Refined vs DTW fallback), with a linear trend line and a 1:1 reference.	At-a-glance monotonicity and trajectory validation.
Refinement Difference	Per-frame correction $f(i) - f_{\mathrm{DTW}}(i)$ applied by the feature matcher on top of the DTW skeleton.	Quantifies how much the fine phase adjusts the coarse DTW prediction.
Velocity Analysis	Locally smoothed ratio $\mathrm{d}f/\mathrm{d}i$ together with its empirical distribution.	Exposes the relative speed profile between the two acquisitions.
DTW Cost Matrix	Heat map of the accumulated cost matrix with the optimal path (and optional ground-truth anchors) overlaid.	Visual signature of the macro-skeleton and valley clarity.
Side-by-Side Videos	Aligned video streams in three modes: raw frames, frames with detected keypoints, and frames with cross-video feature matches drawn.	Empirical sanity checks and visual error attribution.

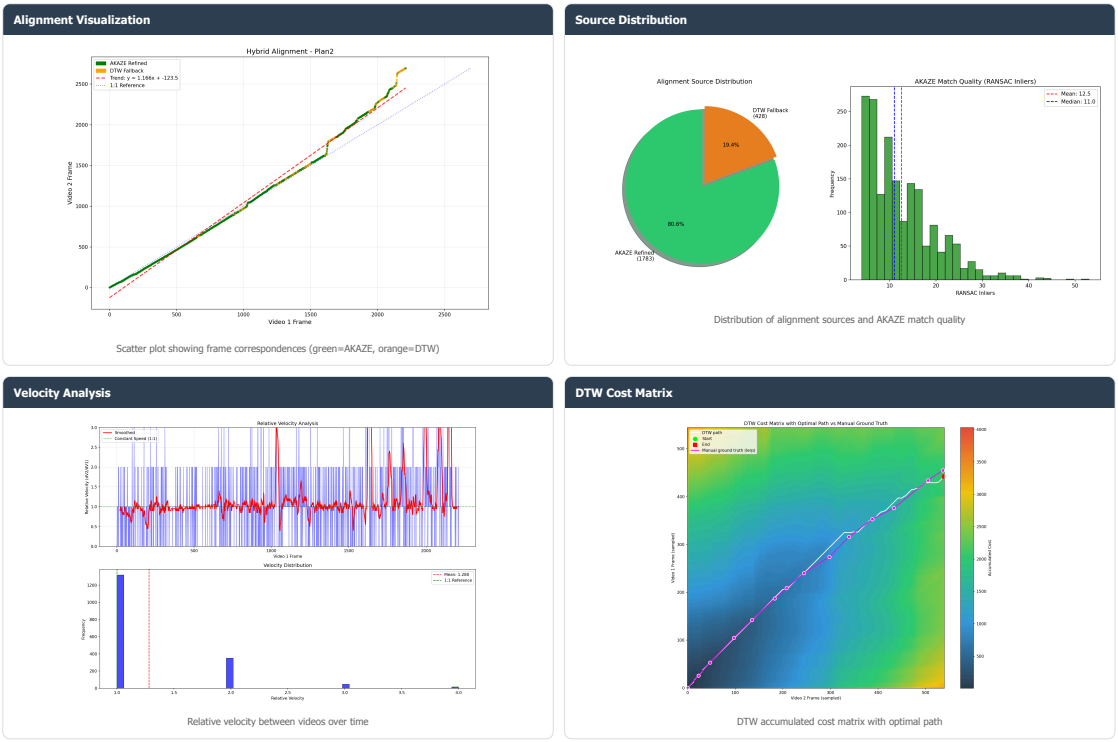

Figure 6.1: Top of the production HTML report for a Plan 2 alignment run. The 2×2 grid combines the alignment scatter (top-left), source distribution and inlier histogram (top-right), velocity analysis (bottom-left), and DTW cost matrix with the optimal path overlaid (bottom-right) into a single browser-viewable document with no external dependencies.

under `dtw_diagnostic/` serves as an independent engineering environment designed to analyze feature performance and evaluate alternative parameter spaces.

6.2.1 Ground-Truth Anchors and Core Tooling

The diagnostic loop relies on a set of fourteen manually annotated geographic anchors distributed along the challenging Plan 2 trajectory. Linear interpolation between these anchor points establishes a piecewise-linear ground-truth baseline, f_{truth}, providing an absolute error metric across design iterations.
The architecture is driven by three distinct command-line instruments:

- **Isolated DTW Evaluation:** A rapid execution script that skips the downstream feature matching to isolate and evaluate modifications made strictly to the feature descriptors, normalization schemes, or step penalties in less than twelve seconds.
- **Exhaustive Pipeline Analyzer:** An end-to-end tool that generates a comprehensive diagnostic folder organized into functional subdirectories, mapping feature spaces via Principal Component Analysis (PCA) and quantifying cost distributions along the ground-truth path.
- **Automated Parameter Sweeper:** A grid-search script that maps the Cartesian product of available extractors, transformations, and step penalties. It automatically ranks candidate configurations by their mean absolute error against the manual anchors, generating the horizontal leaderboard visualization used to establish production defaults.

Figure 6.2 assembles representative outputs from these three instruments (PCA feature space, cost profile along the ground-truth path, and the leaderboard top) into a single diagnostic mosaic.

6.2.2 Hypothesis-Driven Synthesis

To prevent unguided debugging, every automated diagnostic run synthesizes its outputs into a structured Markdown report built around systematic hypothesis testing. Modeled after clinical differential diagnosis, the report formally evaluates the current pipeline state across six fixed assumptions, assessing feature discriminative power, cost function validity, endpoint boundaries, step distributions, PCA clustering health, and the confidence profile of the rescue zones. On the challenging Plan 2 dataset, this systematic breakdown successfully isolated the features as the primary remaining bottleneck and empirically validated the necessity of the rescue mechanism.

6.3 Domain Reusability and Summary

The architectural core of the diagnostic subsystem remains fundamentally domain-agnostic. The feature extraction layer uses pluggable registries, meaning any localized vector repre-

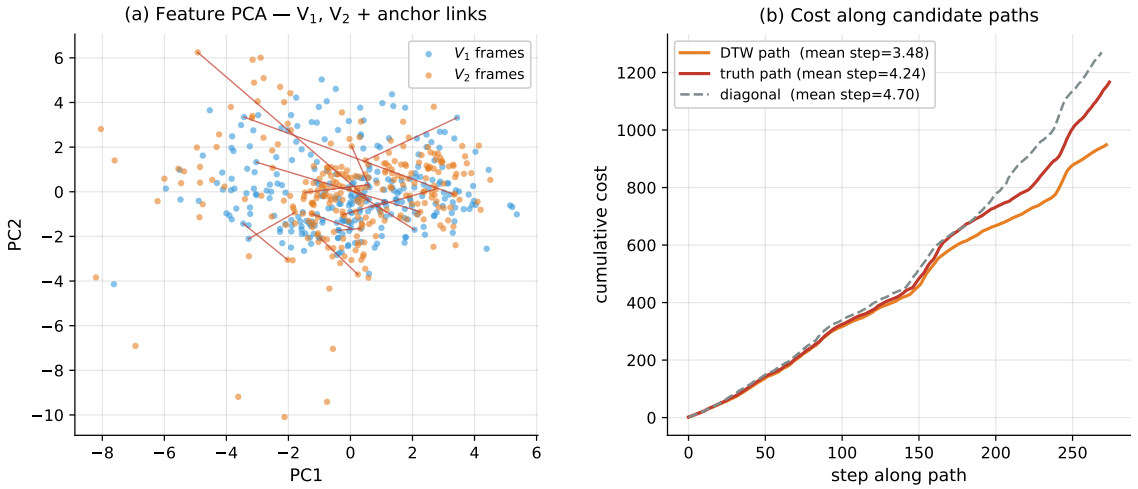

Figure 6.2: Two diagnostic outputs from the Phase A subsystem on Plan 2. (a) PCA projection of the gradient feature vectors of V_1 (blue) and V_2 (orange); red links join the fourteen ground-truth anchor pairs. (b) Cumulative cost along three candidate paths: the DTW path is globally cheaper (3.48/step) than the ground-truth path (4.24/step), confirming that the cost function does not reward the geographically correct cell.

sentation can be integrated; the plotting routines and the hypothesis-driven framework rely strictly on abstract cost matrices and generic coordinate mappings rather than railway-specific primitives. Two localized parameters restrict immediate out-of-the-box transferability: the hardcoded sky crop, which assumes a forward-facing horizon at one-third of the frame height, and the localized confidence thresholds calibrated for the current industrial imagery. Adjusting these configuration variables allows the full diagnostic stack to be seamlessly repurposed for any sequence-to-sequence video alignment task.

In summary, the visualization and diagnostic infrastructure transforms the alignment pipeline from an uninterpretable mathematical sequence into an auditable system. By decoupling operational production summaries from deep algorithmic diagnostics, the tooling successfully supported the design optimization of this thesis while ensuring long-term maintainability for future developers.

7. Conclusion and Future Work

Section 7.1 summarises the technical contributions. Section 7.2 returns to the four research questions of Chapter 1 with answers grounded in Chapter 5. Section 7.3 lists the known limitations. Section 7.4 outlines four directions for future work.

7.1 Summary of Contributions

The technical contributions of the thesis fall into five categories, presented in the order of appearance.

A hybrid coarse-to-fine alignment pipeline. The central contribution is a six-phase pipeline that combines the global robustness of Subsequence Dynamic Time Warping with the local precision of binary-descriptor feature matching. Three pipeline-level mechanisms address the difficulties specific to railway video: a luminance preprocessing step that aligns the brightness distributions before any feature is computed, a confidence-weighted refinement step whose strict window guarantees drift resistance, and a global monotonicity-preserving optimisation that eliminates the trembling artefact of per-frame greedy matching. The six phases are documented in Chapter 4; the implementation is organised across three generations of Python code.

Gradient orientation histograms as DTW input features. A sixteen-bin histogram of Sobel gradient orientations, normalised per video by z-score, was selected through a sweep over nine candidate extractors on a manually-annotated subset. The descriptor outperforms the optical-flow alternative by a factor of roughly four on mean absolute alignment error, with the residual concentrated in regions where the cost function does not reward the ground truth as cheapest. The descriptor is a few milliseconds per frame on a commodity CPU and is therefore compatible with the deployment constraint.

A dual-use luminance histogram-matching mechanism. The same 256-entry look-up table derived from the CDFs of the two videos' L channels is used twice: as preprocessing before feature extraction (Phase 0), where it improves the DTW endpoint by roughly two hundred frames on the most studied plan, and as photometric normalisation before SSIM and LPIPS at evaluation time, where it brings the metrics closer to measuring alignment

quality rather than recording-day weather. Recognising that the same primitive serves both roles is itself a methodological contribution.

Per-frame DTW confidence and a conservative rescue mechanism. A per-frame confidence score $\kappa \in [0, 1]$, derived from the position of the DTW pick relative to the row mean and minimum, identifies regions where the cost surface is unreliable. A rescue with five conservative conditions (Chapter 4) lets the inner matcher override DTW in low-confidence regions when its evidence is coherent across at least five consecutive frames. The combination targets the failure mode where DTW commits early to a wrong path that monotonicity then forbids backtracking from.

A reusable diagnostic subsystem. The descriptor, confidence score and rescue were driven by a diagnostic subsystem (Chapter 6) providing three command-line tools and a six-hypothesis report structure. The fourteen Plan 2 V1V2 anchors are the empirical core of the design selection and are preserved in the codebase for reproducibility.
These five contributions together respond to the central engineering question of Chapter 1: how to build a temporal alignment pipeline robust enough for railway video, light enough for CPU-only deployment, and documented enough to be picked up by a future contributor.

7.2 Answers to the Research Questions

Chapter 5 answered the four research questions of Chapter 1 in detail (Section 5.7). In short:

RQ1, alignment under CPU-only constraint? Yes. Monotonicity is 100% on every plan and every algorithm, mean cycle error is bounded by 21 frames on the limit-case plan, and the pipeline runs in a few minutes per video pair on a laptop CPU.

RQ2, choice of inner matcher? The three algorithms perform within statistical noise of each other on the metrics that discriminate alignment quality. The DTW skeleton dominates the alignment quality and the matcher choice is no longer the limiting factor; an operator can pick any of the three on speed or memory grounds.

RQ3, discriminating metrics? SSIM and LPIPS are saturated on this data (10^{-3} to 10^{-2} discrimination index against random). Cycle consistency varies by three orders of magnitude between random and real and is the only metric that meaningfully tracks alignment quality.

RQ4, graceful degradation? The refined fraction drops from 92% to 71% across the difficulty gradient and the cycle error increases from 5 to 21 frames, but monotonicity holds at 100% throughout. The DTW skeleton is the safety net; the residual bottleneck on hard plans is the feature extractor, not the algorithm.

7.3 Known Limitations

We identify five limitations intrinsic to the present design that we have not addressed.

Features remain a bottleneck on a non-negligible fraction of anchors. On Plan 2, even with the gradient descriptor and per-video normalisation, several anchors have their ground-truth V_2 frame at the seventieth percentile or higher of their distance row. The cost function does not reward the truth as the cheapest cell, and no DTW tweak alone can recover it. The rescue partially compensates, but only when the inner matcher itself can identify the correct frame.

The rescue mechanism's activation requires a non-default search window. With the default $W = 10$ and a minimum-offset requirement of twelve frames, the rescue is silent; the user must pass `-search-window 30`. The coupling lets the two mechanisms be ablated independently, but a future contributor could overlook it. A more polished version would auto-widen the window inside detected rescue zones or warn when the rescue is configured but inactive.

Ground truth is available on Plan 2 only. The fourteen anchors are the empirical foundation of the design selection. The aggregate metrics on Plan 1 and Plan 3 are encouraging, but the design choices made on Plan 2 may overfit silently to its characteristics. Extending ground truth to the other plans would be a small investment with a substantial return in design confidence.

The dataset is small and proprietary. Three plans of railway footage are not redistributable. The experiments are reproducible internally but not by an external reader. This is a known limitation of the broader field, but it remains a limitation; a public benchmark for two-pass railway alignment would change the picture.

No tunnel, no night-time, no adverse weather. The dataset is daytime, clear-weather footage on open track. Tunnels would collapse the gradient distribution, night-time would shift the dominant features from rails to lights, and rain or snow would inject high-magnitude photometric noise. The pipeline's behaviour on such inputs is undefined.

7.4 Future Work

The contributions and limitations above point to four directions for future work.

Spatial gradient features. The current gradient histogram aggregates over the entire region of interest, discarding *where* the gradients are. A spatial extension in the spirit of HOG [37] would compute one histogram per cell of a 3×3 or 4×4 grid and

concatenate them. The extractor is already implemented (`feat_grad_hist_3x3_grid` in `dtw_diagnostic/extractors.py`); promoting it to production is a half-hour sweep with access to the Plan 2 ground truth.

Learned per-frame embeddings. If the deployment constraint were relaxed to allow GPU inference, a learned per-frame embedding from a pretrained convolutional network would be the natural next step. Contrastive video embeddings and place-recognition embeddings are both candidates. Plugging such an embedding into the existing DTW pipeline requires only a new extractor in the diagnostic subsystem; the rest is descriptor-agnostic.

Extending ground truth to all three plans. Repeating the annotation exercise on Plan 1 and Plan 3, perhaps thirty to fifty anchors per plan, would strengthen the design selection from "validated on one plan" to "validated on three" and surface any silent overfitting to Plan 2. The diagnostic subsystem accepts a per-plan anchor dictionary, so the extension is purely a data exercise.

Downstream change detection. The alignment is delivered as a frame correspondence list, ready to be consumed by a comparison module. Building that module and demonstrating that the alignment is sufficient for change detection at the relevant precision would close the predictive-maintenance loop of Chapter 1. The detection module itself is out of scope, but its prerequisite is in place.

A fifth, more speculative direction is an end-to-end neural alignment model trained on synthetic data with known correspondences, with the present pipeline as the engineered baseline the learned model must outperform. The diagnostic subsystem would serve as the evaluation harness, with the same anchors and the same six-hypothesis report structure.

7.5 Closing Remarks

The narrow technical question was the alignment of two railway videos. The broader purpose was to deliver an installable, defensible, extensible artefact for an industrial deployment, documented at a level that lets future contributors understand not only *what* the pipeline does but *why*. The choices made along the way, to retain three feature-matching algorithms rather than one, to develop a diagnostic subsystem before fixing the production configuration, to document the failure modes as carefully as the successes, reflect that purpose. What this thesis leaves behind is therefore not only the pipeline and its numbers but the design exercise that produced them, repeatable by someone else on a different dataset towards a different deployment.

The pipeline is delivered. The next steps belong to those deploying it, to the academic community working on this problem, and to the next contributor who picks up where this thesis stops.

Bibliography

[1] Meinard Müller. *Dynamic Time Warping*, pages 69–84. Springer, Berlin, Heidelberg, 2007. URL: https://link.springer.com/chapter/10.1007/978-3-540-74048-3_4, doi:10.1007/978-3-540-74048-3_4.

[2] Siddhartha Sharma et al. Data-driven optimization of railway maintenance for track geometry. *Transportation Research Part C: Emerging Technologies*, 90:34–58, May 2018. doi:10.1016/j.trc.2018.02.019.

[3] Xiukun Wei et al. Railway track fastener defect detection based on image processing and deep learning techniques: A comparative study. *Engineering Applications of Artificial Intelligence*, 80:66–81, April 2019. doi:10.1016/j.engappai.2019.01.008.

[4] Danyang Zheng et al. A defect detection method for rail surface and fasteners based on deep convolutional neural network. *Computational Intelligence and Neuroscience*, 2021(1):2565500, January 2021. doi:10.1155/2021/2565500.

[5] Giovanni Bellacci et al. Estimation of railway track vertical alignment using instrumented wheelsets and contact force recordings. *Machines*, 13(10):963, October 2025. doi:10.3390/machines13100963.

[6] Shahrzad Faghih-Roohi et al. Deep convolutional neural networks for detection of rail surface defects. In *2016 International Joint Conference on Neural Networks (IJCNN)*, pages 2584–2589, Vancouver, BC, Canada, 2016. IEEE. doi:10.1109/IJCNN.2016.7727522.

[7] Stefano Alfi et al. Identification of railway vertical track alignment via the unknown input observer. *Applied Sciences*, 15(21):11332, October 2025. doi:10.3390/app152111332.

[8] Christian Higgins and Xiang Liu. Modeling of track geometry degradation and decisions on safety and maintenance: A literature review and possible future research directions. *Proceedings of the Institution of Mechanical Engineers, Part F: Journal of Rail and Rapid Transit*, 232(3):938–951, May 2018. doi:10.1177/0954409717721870.

[9] Thomas Keogh, D. E. Mesher, and T. R. Keegan. An integrated system for accurate tie and ballast condition assessment. In *Proceedings of the American Railway Engineering*

and Maintenance-of-Way Association (AREMA) Annual Conference, Louisville, KY, USA, 2006. URL: https://railtec.illinois.edu/wp/wp-content/uploads/Keogh-et-al.-An-Integrated-System-for-Accurate-Tie-and-Ballast-.pdf.

[10] Markus Ziegler, Vishal Mhasawade, Martin Köppel, Philipp Neumaier, and Volker Eiselein. A comprehensive framework for evaluating vision-based on-board rail track detection. In *2023 IEEE Intelligent Vehicles Symposium (IV)*, pages 1–8, Anchorage, AK, USA, 2023. IEEE. doi:10.1109/IV55152.2023.10186659.

[11] Xavier Gibert et al. Robust fastener detection for autonomous visual railway track inspection. In *2015 IEEE Winter Conference on Applications of Computer Vision*, pages 694–701. IEEE, 2015. doi:10.1109/WACV.2015.98.

[12] Antonio Cabrera and Marcelo Vargas. Automated track video inspection pilot project. Technical Report FTA Report No. 0049, Federal Transit Administration (FTA), U.S. Department of Transportation, September 2013. URL: https://www.transit.dot.gov/sites/fta.dot.gov/files/FTA0049_Research_Report_Summary.pdf, doi:10.21949/1503587.

[13] Ferran Diego et al. Video alignment for change detection. *IEEE Transactions on Image Processing*, 20(7):1858–1869, July 2011. doi:10.1109/TIP.2010.2095873.

[14] Márton Bidlek. Camera synchronization from feature points for 3d motion reconstruction. Master's thesis, Uppsala University, Faculty of Science and Technology, 2025. DiVA ID: diva2:1989683. URL: https://www.diva-portal.org/smash/get/diva2:1989683/FULLTEXT01.pdf.

[15] Jiawang Bian et al. Gms: Grid-based motion statistics for fast, ultra-robust feature correspondence. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2828–2837, Honolulu, HI, 2017. doi:10.1109/CVPR.2017.302.

[16] Geonseok Lee et al. A depth-guided local outlier rejection methodology for robust feature matching in urban uav images. *Drones*, 9(12):869, 12 2025. doi:10.3390/drones9120869.

[17] M. H. Pinson and S. Wolf. A new standardized method for objectively measuring video quality. *IEEE Transactions on Broadcasting*, 50(3):312–322, 9 2004. doi:10.1109/TBC.2004.834028.

[18] Surendra Kumar Sharma and Kamal Jain. Image stitching using akaze features. *Journal of the Indian Society of Remote Sensing*, 48(10):1389–1401, 10 2020. doi:10.1007/s12524-020-01163-y.

[19] Stefan Leutenegger et al. Brisk: Binary robust invariant scalable keypoints. In *2011 International Conference on Computer Vision*, pages 2548–2555. IEEE, 2011. doi:10.1109/ICCV.2011.6126542.

[20] Shaharyar Ahmed Khan Tareen and Zahra Saleem. A comparative analysis of sift, surf, kaze, akaze, orb, and brisk. In *2018 International Conference on Computing, Mathematics and Engineering Technologies (iCoMET)*, pages 1–10. IEEE, 2018. doi:10.1109/ICOMET.2018.8346440.

[21] Cuiyin Liu et al. A review of keypoints' detection and feature description in image registration. *Scientific Programming*, 2021:1–25, 12 2021. doi:10.1155/2021/8509164.

[22] Chaoqun Ma et al. Improved orb algorithm using three-patch method and local gray difference. *Sensors*, 20(4):975, 2 2020. doi:10.3390/s20040975.

[23] Bin Fan et al. Learning weighted hamming distance for binary descriptors. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 2395–2399. IEEE, 2013. doi:10.1109/ICASSP.2013.6638084.

[24] Ethan Rublee et al. Orb: An efficient alternative to sift or surf. In *2011 International Conference on Computer Vision*, pages 2564–2571. IEEE, 2011. doi:10.1109/ICCV.2011.6126544.

[25] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 11 2004. doi:10.1023/B:VISI.0000029664.99615.94.

[26] Herbert Bay et al. Speeded-up robust features (surf). *Computer Vision and Image Understanding*, 110(3):346–359, 6 2008. doi:10.1016/j.cviu.2007.09.014.

[27] Edward Rosten and Tom Drummond. Machine learning for high-speed corner detection. In *Computer Vision – ECCV 2006*, pages 430–443. Springer, 2006. doi:10.1007/11744023_34.

[28] Michael Calonder et al. Brief: Binary robust independent elementary features. In Kostas Daniilidis, Petros Maragos, and Nikos Paragios, editors, *Computer Vision – ECCV 2010*, pages 778–792. Springer, 2010. doi:10.1007/978-3-642-15561-1_56.

[29] Elmar Mair et al. Adaptive and generic corner detection based on the accelerated segment test. In Kostas Daniilidis et al., editors, *Computer Vision – ECCV 2010*, pages 183–196. Springer, 2010. doi:10.1007/978-3-642-15552-9_14.

[30] Pablo F. Alcantarilla et al. Fast explicit diffusion for accelerated features in nonlinear scale spaces. In *Proceedings of the British Machine Vision Conference 2013*, pages 13.1–13.11, Bristol, 2013. doi:10.5244/C.27.13.

[31] Martin A. Fischler and Robert C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 6 1981. doi:10.1145/358669.358692.

[32] Rafael Luiz Testa et al. Enhancing temporal coherence in image-to-video facial expression synthesis: A dual-loss framework for smoother generation. *IEEE Access*, 13:170876–170894, 2025. doi:10.1109/ACCESS.2025.3612820.

[33] Gunnar Farnebäck. Two-frame motion estimation based on polynomial expansion. In Josef Bigun and Tomas Gustavsson, editors, *Image Analysis*, pages 363–370. Springer, 2003. doi:10.1007/3-540-45103-X_50.

[34] Ying Xie and Bryan Wiltgen. Adaptive feature based dynamic time warping. *International Journal of Computer Science and Network Security (IJCSNS)*, 10(1):264–273, 1 2010. URL: https://lig-membres.imag.fr/bisson/cours/M2INFO-AIW-ML/papers/Xie10.pdf.

[35] Irwin Sobel and Gary Feldman. A 3x3 isotropic gradient operator for image processing. Talk at the Stanford Artificial Intelligence Project (SAIL), 1968. URL: https://www.researchgate.net/publication/284307521_A_3x3_isotropic_gradient_operator_for_image_processing.

[36] Jiaping Zhao and Laurent Itti. shapedtw: Shape dynamic time warping. *Pattern Recognition*, 74:171–184, 2 2018. doi:10.1016/j.patcog.2017.09.020.

[37] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 886–893. IEEE, 2005. doi:10.1109/CVPR.2005.177.

[38] Peter Sand and Seth Teller. Video matching. *ACM Transactions on Graphics (TOG)*, 23(3):592–599, 2004. URL: https://rvsn.csail.mit.edu/vid-match/vid-match.pdf.

[39] Hiroaki Sakoe and Seibi Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 26(1):43–49, 2 1978. doi:10.1109/TASSP.1978.1163055.

[40] Renjie Wu and Eamonn J. Keogh. Fastdtw is approximate and generally slower than the algorithm it approximates. *IEEE Transactions on Knowledge and Data Engineering*, 34(8):3713–3726, 2020. doi:10.1109/TKDE.2020.3033752.

[41] Matthieu Herrmann and Geoffrey I. Webb. Amercing: An intuitive and effective constraint for dynamic time warping. *Pattern Recognition*, 137:109333, 2023. doi:10.1016/j.patcog.2023.109333.

[42] Zhou Wang, Alan C. Bovik, Hamid R. Sheikh, and Eero P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 4 2004. doi:10.1109/TIP.2003.819861.

[43] Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 586–595, 2018. URL: https://arxiv.org/abs/1801.03924, doi:10.48550/arXiv.1801.03924.

[44] William K. Pratt. *Digital Image Processing*. John Wiley & Sons, Inc., Hoboken, NJ, USA, 4th edition, 2007. URL: https://onlinelibrary.wiley.com/doi/book/10.1002/0470097434, doi:10.1002/0470097434.

[45] Rémi Auguste. *Reconnaissance dynamique de personnes dans les émissions audiovisuelles*. Thèse de doctorat, Université de Lille, 2014. URL: https://theses.hal.science/tel-01114399.

[46] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 2223–2232, 2017. URL: https://arxiv.org/abs/1703.10593, doi:10.48550/arXiv.1703.10593.

[47] Donald J. Berndt and James Clifford. Using dynamic time warping to find patterns in time series. In *Proceedings of the AAAI Workshop on Knowledge Discovery in Databases (KDD-94)*, pages 359–370, Seattle, WA, USA, 1994. AAAI Press. URL: https://aaai.org/papers/359-ws94-03-031/.

[48] Eamonn Keogh and Chotirat Ann Ratanamahatana. Exact indexing of dynamic time warping. *Knowledge and Information Systems*, 7(3):358–386, 3 2005. doi:10.1007/s10115-004-0154-9.

[49] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superpoint: Self-supervised interest point detection and description. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 224–236, 2018.

[50] Xinchao Li et al. Pairwise geometric matching for large-scale object retrieval. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5153–5161. IEEE, 2015. doi:10.1109/CVPR.2015.7299151.

A. Algorithm Selection: Full Candidate List

This appendix documents the feature-detection and feature-description algorithms considered at the start of the project, the three elimination criteria stated in Chapter 4 Section 4.2, and the per-algorithm outcome. The survey of the candidate space was conducted through two reference works: the comparative analysis of [20], which benchmarks the classical and binary families head to head, and the broader review of keypoint detection and feature description in image registration by [21], which situates the learned methods relative to the classical ones.

A.1 Methodology of the Selection

The three criteria, in priority order, are:

1. **Deployment constraint.** The pipeline must run on a CPU-only system. Algorithms that require a GPU for inference, because they are based on a pretrained neural network, are eliminated.

2. **Matching speed at production scale.** Floating-point descriptors compared by L_2 distance are too slow for the production volume (tens of thousands of matcher invocations per video pair). Binary descriptors compared by Hamming distance are required.

3. **Implementation maturity and design-space coverage.** Among the survivors of the first two criteria, we keep one representative per design family, preferring those with mature OpenCV implementations.

Each criterion eliminates candidates strictly: once a candidate fails one criterion, it is set aside and the subsequent criteria are not applied to it. The order reflects their nature: the first is a deployment constraint, the second is an engineering constraint at production scale, the third is a design choice that keeps the comparison study manageable.

A.2 Eliminated by Criterion 1 (Deployment Constraint)

The learned detectors and descriptors surveyed in [21] share a common property: they rely on a convolutional backbone whose inference is impractical on a CPU at the throughput this pipeline requires. The following methods were considered and eliminated on this ground. TILDE is a learned keypoint detector trained for repeatability across time-of-day changes. LIFT is an end-to-end trainable pipeline for detection, orientation estimation and description. SuperPoint is a fully convolutional detector and descriptor trained on synthetic homographies and self-supervised on real images. D2-Net jointly extracts keypoints and descriptors from a single backbone. R2D2 trains separate repeatability and reliability heads with hard-negative mining. DISK is a reinforcement-learning-trained detector and descriptor. ALIKE is a lightweight convolutional detector, the fastest of the learned family but still GPU-dependent at the throughputs required here.

A common counter-argument is that some of these networks, in their lightest configurations, can run on a CPU at reduced frame rates. This is true, but the resulting throughput of a few frames per second is incompatible with the production volume of the pipeline, where each matcher invocation operates on a fresh frame and there are tens of thousands of invocations per video pair. The CPU-only constraint is therefore both a hardware constraint and a throughput constraint, and the learned candidates fail it on the latter even when they pass on the former.

A.3 Eliminated by Criterion 2 (Matching Speed)

Two of the remaining candidates are floating-point descriptors, eliminated for matching speed as quantified in the comparative benchmark of [20].

SIFT [25]: a 128-dimensional floating-point descriptor, the historical reference. Excellent robustness, but the L_2 matching is an order of magnitude slower than Hamming matching for binary descriptors of comparable length. Eliminated for speed.

SURF [26]: a 64- or 128-dimensional floating-point descriptor approximating SIFT with box filters. Faster than SIFT but still floating-point. Eliminated for speed.

A common counter-argument is the availability of integer SIFT variants or CPU-optimised SIFT implementations. We did not pursue these because the project schedule favoured a clear-cut decision over a fine-grained micro-optimisation, and because the binary alternatives surviving criterion 1 already cover the relevant design space.

A.4 Eliminated by Criterion 3 (Maturity / Design-Space Coverage)

Three of the remaining candidates survive the first two criteria but are eliminated for redundancy or implementation immaturity.

BRIEF [28]: a binary descriptor computed from pairwise intensity comparisons, with no associated detector. It must be paired with an external detector such as FAST. Eliminated because the BRIEF-plus-FAST combination is subsumed by ORB, which adds rotation invariance and a Harris score on top of the same building blocks.

KAZE : the original non-linear scale-space detector and descriptor, the predecessor of AKAZE, characterised alongside its accelerated successor in [20]. It produces a floating-point descriptor by default; the binary variant is exactly what AKAZE provides. Eliminated in favour of AKAZE for both speed and OpenCV implementation maturity.

FREAK : a binary descriptor inspired by the retinal sampling pattern of the human eye, benchmarked in the same comparative study [20]. Strong design, but on the railway data we tested informally it was indistinguishable in measured performance from BRISK, which is the representative we retained for that design family. Eliminated for redundancy.

A.5 Retained: AKAZE, BRISK, ORB

Three algorithms survive all three criteria and are retained for the comparison study. Each is the representative of a different design family.

ORB [24]: the FAST-plus-Rotated-BRIEF combination. Fastest of the three. Corner-based detector, 256-bit descriptor.

BRISK [19]: an AGAST detector with a 512-bit circular-sampling-pattern descriptor. Stronger scale-space treatment than ORB, with comparable speed.

AKAZE [30]: a non-linear scale space with a Modified Local Difference Binary descriptor. Slowest of the three, but produces the most stable keypoints on edge-rich content such as rails and overhead structures. Recommended default in the production configuration.

The three share a common OpenCV interface (`detectAndCompute`, returning keypoints and a binary descriptor matrix), a common matcher (brute-force with Hamming distance), and a common downstream chain (Lowe's ratio test, spatial coherence filter, RANSAC homography). The pipeline switches between them with a single configuration parameter.

A.6 Summary Table

Table A.1 consolidates the fifteen candidates of the selection on a single page: family, descriptor type, GPU dependency, status, and primary reference. Eliminated candidates are grouped by the criterion that disqualified them; retained candidates appear first.

Table A.1: Algorithm-selection summary. Fifteen candidates were considered; three were retained for the comparison study of Chapter 5. "Crit. N" refers to the elimination criterion defined in Section A.1 (1: requires GPU; 2: floating-point matching too slow; 3: redundant or implementation immature).

Algorithm	**Detector family**	**Descriptor**	**GPU**	**Status**	**Reference**
Retained (binary, CPU-only)					
AKAZE	non-linear (edges)	binary 488 b (MLDB)	no	Retained	[30]
BRISK	AGAST corner	binary 512 b (pattern)	no	Retained	[19]
ORB	FAST corner + Harris	binary 256 b (rBRIEF)	no	Retained	[24]
Eliminated - learned (CNN / RL backbone, GPU-dependent)					
TILDE	learned	learned float	yes	Crit. 1	[21]
LIFT	learned end-to-end	learned float	yes	Crit. 1	[21]
SuperPoint	learned (CNN)	learned float	yes	Crit. 1	[21]
D2-Net	learned (CNN)	learned float	yes	Crit. 1	[21]
R2D2	learned (CNN)	learned float	yes	Crit. 1	[21]
DISK	learned (RL)	learned float	yes	Crit. 1	[21]
ALIKE	learned (light CNN)	learned float	yes	Crit. 1	[21]
Eliminated - classical floating-point					
SIFT	DoG blob	float 128-dim	no	Crit. 2	[25]
SURF	box-filter blob	float 64-128 dim	no	Crit. 2	[26]
Eliminated - binary, redundant or immature					
BRIEF	- (no detector)	binary 256 b	no	Crit. 3	[28]
KAZE	non-linear (edges)	float (default)	no	Crit. 3	[20]
FREAK	paired with corner	binary 512 b (retinal)	no	Crit. 3	[20]

A.7 Note on the Retention Decision

The decision to retain three algorithms rather than collapse to a single default was a methodological choice. It allows the comparison study reported in Chapter 5 to isolate the effect of the inner descriptor from the effect of the surrounding pipeline. The retention also documents the design space at a fine grain, so that a future contributor working on a different dataset has three pre-vetted baselines to choose from before considering more exotic options.

A by-product of the comparison study, presented as an empirical finding in Chapter 5 and discussed in Chapter 7, is that the three retained algorithms perform within statistical noise of each other on the metrics that meaningfully discriminate alignment quality. This finding does not invalidate the design rationale of the comparison study, it strengthens

it: it establishes that the choice of inner matcher is not the dominant factor in the final alignment quality, and that future engineering work should focus on the DTW skeleton and the rescue mechanism rather than on micro-optimising the matcher.

UNIVERSITÉ CATHOLIQUE DE LOUVAIN
École polytechnique de Louvain
Rue Archimède, 1 bte L6.11.01, 1348 Louvain-la-Neuve, Belgique | **www.uclouvain.be/epl**