# Robust Correspondence of CAD Drawings with Partial Optimal Transport

A PROJECT REPORT
SUBMITTED IN PARTIAL FULFILMENT OF THE
REQUIREMENTS FOR THE DEGREE OF
**Master of Technology**
IN
**Faculty of Engineering**

BY

**Valadri Sai Suraj Reddy**


Computer Science and Automation
Indian Institute of Science
Bangalore – 560 012 (INDIA)

June, 2026

# Declaration of Originality

I, **Valadri Sai Suraj Reddy**, with SR No. **04-04-00-10-42-24-1-24297** hereby declare that the material presented in the thesis titled

**Robust Correspondence of CAD Drawings with Partial Optimal Transport**

represents original work carried out by me in the **Department of Computer Science and Automation** at **Indian Institute of Science** during the years **2024–2026**.
With my signature, I certify that:

- I have not manipulated any of the data or results.
- I have not committed any plagiarism of intellectual property. I have clearly indicated and referenced the contributions of others.
- I have explicitly acknowledged all collaborative research and discussions.
- I have understood that any false claim will result in severe disciplinary action.
- I have understood that the work may be screened for any form of academic misconduct.

Date: Monday 29th June, 2026

Student Signature

In my capacity as supervisor of the above-mentioned work, I certify that the above statements are true to the best of my knowledge, and I have carried out due diligence to ensure the originality of the report.

Advisor Name: Prof. Vijay Natarajan

Advisor Signature

DEDICATED TO

*My well-wishers — near and far*

# Acknowledgements

I thank my advisor, Prof. Vijay Natarajan (CSA, IISc), for his guidance and the discussions that shaped this work. I am also grateful to Prof. Raghavendra G.S. (CVIT, IIIT-Hyderabad) and Kedar Patil (PTC Software, Pune) for their guidance.

This work was carried out in the Visualization and Graphics Lab (VGL), Department of Computer Science and Automation, IISc; I thank the Department and the Institute for the research environment and resources. Finally, I thank my family and friends for their constant encouragement.

# Abstract

Establishing accurate correspondences between Computer-Aided Design (CAD) drawings underpins retrieval, design reuse, inspection, and change detection, yet two exports of the same shape rarely agree: they may sit in different frames (translation and scale), overlap only partially when features are added or removed, and subdivide the same geometry differently. We present a training-free pipeline that matches the primitives of one CAD drawing to those of another under all of these differences at once. Each drawing becomes a measured metric–feature space — a per-primitive descriptor unifying lines, circles, arcs, and splines; a contact-graph structure matrix; and length-proportional masses — and tangent-continuous chains are merged into subdivision-invariant super-nodes, so a shape matches whether it was exported as one curve or as many. Correspondence is solved as a Partial Fused Gromov–Wasserstein (pFGW) problem whose transported fraction — the unknown overlap — is selected automatically from the transport-cost curve, then unpacked into interpretable one-to-one and one-to-many matches with colour-coded visualizations for verification. To assess the matcher without the cost and ambiguity of hand-labelling, we further contribute an evaluation framework that synthesizes design-edit pairs whose ground-truth correspondence is known by construction: a vision–language model chooses realistic edits and a deterministic engine executes them exactly, yielding always-valid targets and exact correspondences for objective scoring — and laying the groundwork for a scalable, topology-aware CAD-matching benchmark.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

This chapter motivates the problem of establishing primitive-level correspondences between two CAD drawings, makes the three differences that defeat naive matchers precise, states the single assumption under which we work, situates the thesis among prior approaches, summarizes the contributions of the thesis, and lays out the structure of the remaining chapters.

## 1.1 Motivation and Problem Statement

In the domain of Computer-Aided Design (CAD), the retrieval, alignment, and comparison of models are fundamental to reverse engineering, design reuse, and quality assurance. Unlike unstructured point clouds, CAD drawings are defined by precise geometric *primitives* — straight line segments, circular and elliptical arcs, full circles, and free-form spline curves. A recurring and practically important challenge arises when one must match a clean reference model (which we call the **Source**) against a scan, a revised version, or a user sketch (the **Target**) that contains modifications. Solving this matching problem is the prerequisite for answering questions a designer actually asks: *which* features changed between two revisions, *where* a library part reappears inside a larger assembly, and *whether* two superficially different exports in fact denote the same shape.

The core difficulty is that two exports of supposedly the same shape almost never agree exactly at the level of primitives. We illustrate this with a concrete running example in Figure 1.1. Both the Source and the Target are the *same* mechanical part, reduced to their two-dimensional CAD drawings of geometric primitives. The Source represents a clean ground-truth layout. The Target exhibits a typical "real-world" scenario: it retains the core structure but introduces an extra component (an additional drilled hole) and may displace or omit other features. A naive geometric matcher — one that pairs primitives by nearest position or by similar length — could

easily pair the stray extra hole with one of the genuine holes of the Source, or pair the wrong edges together once the part is shifted or rescaled.

To constitute a *good* match in this context, an algorithm must satisfy two complementary criteria simultaneously:

1. **Geometric fidelity:** it must align primitives by their physical attributes — position, length, and curvature; and

2. **Topological consistency:** it must distinguish valid parts from clutter by analysing their structural context — how each primitive connects to its neighbours.

Neither criterion is sufficient on its own. Geometry alone is fooled by clutter that happens to look locally similar; topology alone cannot tell two structurally identical but geometrically different features apart. The central modelling choice of this thesis — optimal transport with a fused feature-and-structure objective — is precisely a principled way to honour both criteria at once.

## 1.2 Challenges

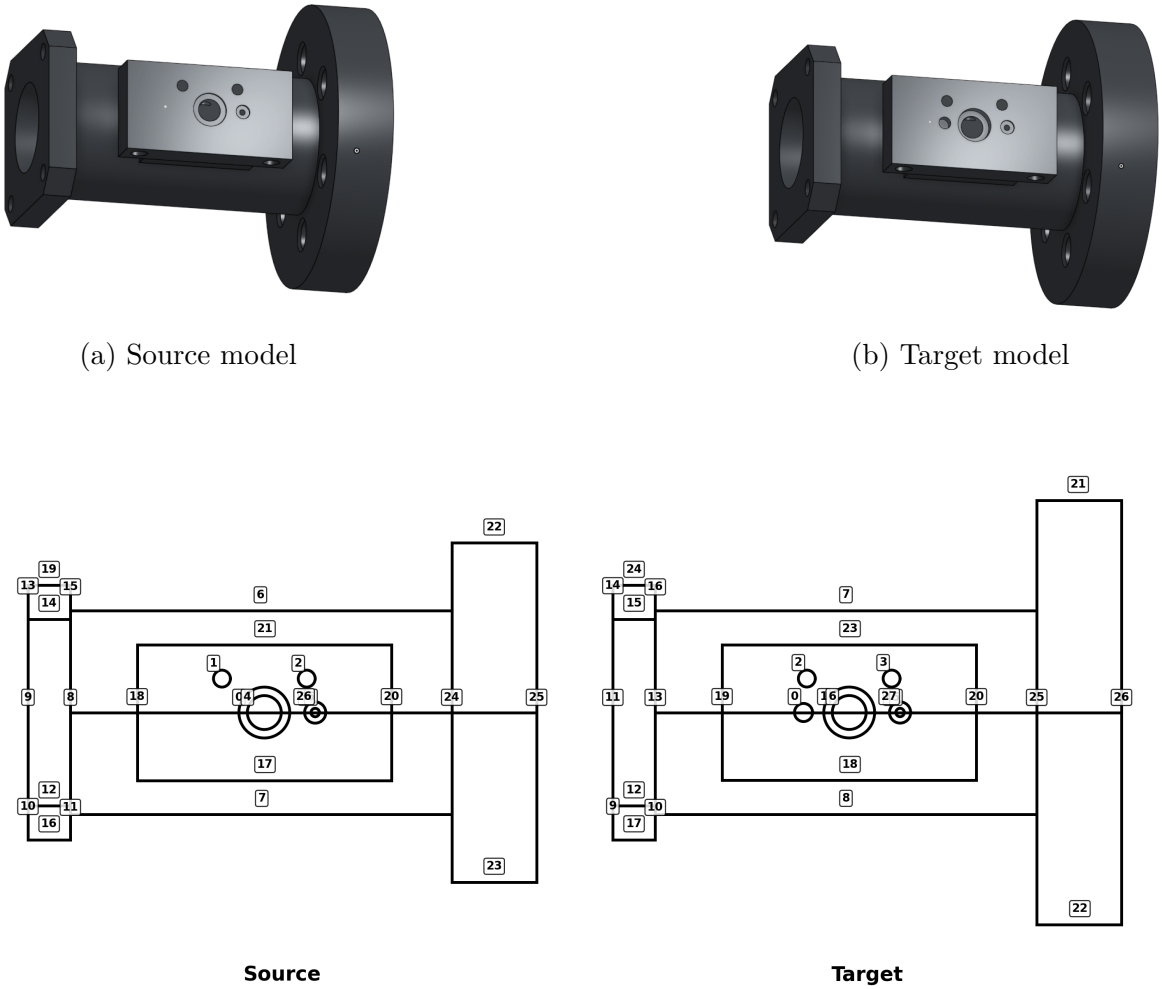
Beyond clutter, a robust matcher must overcome three further differences, each of which arises routinely when two drawings of "the same" shape are produced independently (Figure 1.2):

(i) **Frame differences.** The two drawings need not share a common translation or a uniform scale. The same part may be exported at a different size or placed at a different origin, and a matcher that compares raw coordinates would see every primitive as displaced.

(ii) **Partial overlap.** Either drawing may contain primitives that are absent from the other: a feature deleted in a revision, a hole drilled in the Target, or stray annotation. A matcher that insists on pairing every primitive is forced to invent matches for content that genuinely has no counterpart.

(iii) **Subdivision differences.** The same geometric content may be exported as a single primitive in one drawing and as several tangent-continuous pieces in the other — one long edge versus three collinear segments, or a slot exported as one closed polyline versus four separate arcs and lines. A matcher that pairs primitives one-to-one fails on the mismatched counts alone, before any geometry is even considered.

These three differences are not hypothetical edge cases; they are the normal state of affairs whenever drawings come from different exporters, different revisions, or a human in the loop.

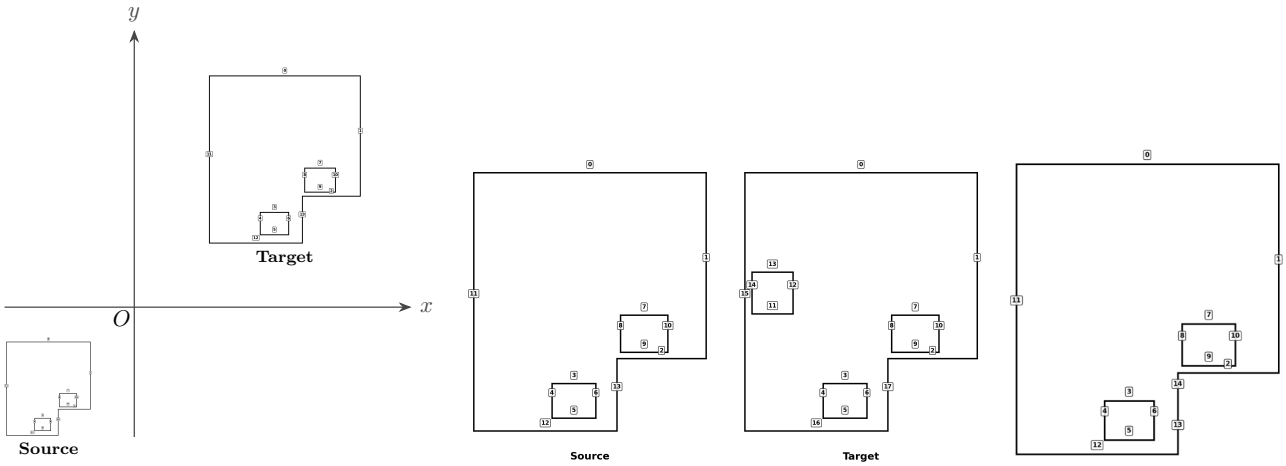



(a) Source model

(b) Target model

(c) The two drawings reduced to numbered geometric primitives

Figure 1.1: **Problem definition.** (Top) Three-dimensional CAD models of the Source and the Target — the same mechanical part. (Bottom) The corresponding two-dimensional CAD drawings, reduced to numbered geometric primitives. The Target retains the shared layout but introduces an extra hole (the small circle near the central bore) as clutter. A correct correspondence must match only the common structure and leave the cluttered Target primitives unmatched.




(a) Frame differences (b) Partial overlap (c) Subdivision

Figure 1.2: **The three challenges a robust matcher must absorb**. **(a) Frame differences:** the same drawing may sit at a different origin and uniform scale, so the raw coordinates differ even though the shape is identical — the coordinate frame $O$–$x$–$y$ is drawn for reference, with one copy near the origin and a translated, enlarged copy of the *same* drawing. **(b) Partial overlap:** either drawing may contain primitives absent from the other; here the Target (right) adds a rectangular loop at upper left (primitives 11–14) that has no counterpart in the Source. **(c) Subdivision:** the same geometric content may be exported as one primitive in one drawing and as several tangent-continuous pieces in the other (here the right edge is split into the collinear segments 14, 13).

The contribution of this thesis is a single pipeline that addresses all three at once, rather than patching each in isolation.

## 1.3 Assumptions

The method operates under a single explicit assumption.

> **Pre-alignment (orientation).** The Source and the Target are assumed to be pre-aligned in orientation by an upstream stage: there is no rotation or reflection between them.

This assumption is mild in practice: it is the normal state of a real CAD workflow, where two revisions of a part are exported in the same standard orthographic views. In our own test set, for instance, the Source and Target are two revisions of the same real exported part that differ in their features — the later revision drills an extra hole around the central bore — yet share a common orientation, exactly as the assumption requires.

This assumption deserves a word of justification, because the method is in fact *partly* rotation-invariant by construction. As we develop in Chapter 2, the structural (Gromov–Wasserstein) term of our objective is theoretically invariant to all rigid motion, including rotation and reflection. The feature term, however, is deliberately *not* rotation-invariant: it compares descriptors that include absolute position, which is a strong and useful cue precisely because most CAD workflows do preserve orientation. Removing only translation and scale — which the method does internally (Section 3.5) — is therefore sufficient under this assumption, which could be relaxed by adding a coarse rotational search on the feature term.

## 1.4 Related Work

Having stated the problem and its challenges, we now situate the thesis among prior approaches to shape and CAD matching. The literature falls into three strands — classical structural shape matching, data-driven CAD representation learning, and optimal transport — and we explain where each falls short of the requirements set out above (Section 1.2) and why a training-free, partial optimal-transport formulation is the right fit. The formal machinery these methods build on is developed in Chapter 2.

### 1.4.1 Structural Shape Matching

Matching shapes by their structure has a long history. Classical methods build medial-axis or shock graphs to capture topology and match them by edit distance [19]. These methods are interpretable and training-free but are tailored to silhouette-like shapes and do not naturally

accommodate the heterogeneous primitive vocabulary (lines, arcs, circles, splines) or the partial overlap that characterizes CAD revisions. They also commit to a single structural abstraction (the skeleton), whereas our setting demands that geometry *and* topology be weighed against each other primitive by primitive.

### 1.4.2 Data-Driven CAD Representation Learning

A more recent line of work applies graph neural networks to Boundary Representations (B-reps) for CAD retrieval and generation [17, 8]. Such models can learn powerful similarity embeddings, but they require large labelled corpora and substantial training, and they typically return a global similarity score or an embedding rather than an explicit, verifiable primitive-to-primitive correspondence. Our goal is the opposite: a training-free method that returns an interpretable transport plan one can inspect and audit, without any learned parameters and without a dependence on the distribution of a training set.

### 1.4.3 Optimal Transport for Matching

Optimal transport (OT) offers an attractive training-free alternative. The Gromov–Wasserstein distance [11] compares metric spaces by their internal pairwise relations and is invariant to isometries; the Fused Gromov–Wasserstein distance [22] augments it with a feature-matching term for attributed graphs. To tolerate the partial overlap and clutter of CAD revisions we adopt Partial Optimal Transport [3], which transports a mass $m < 1$ and thereby ignores outliers. Our use of partial FGW for correspondence is most directly inspired by the recent partial-OT topology-tracking framework of Li et al. [10], which we adapt from scalar-field topology to CAD primitives. All transport solves in this thesis build on the POT library [6], and the drawings are authored in Onshape [12].

## 1.5 Contributions

This thesis develops a complete, training-free pipeline for CAD drawing correspondence and documents the rationale behind every design decision. The principal contributions are the following.

1. **A measured metric–feature representation** of heterogeneous CAD drawings that unifies five primitive types under one geometric descriptor, an analytic per-type geometry layer (exact arc lengths, curvature profiles, and endpoint tangents), and a contact-graph topology (Sections 3.2–3.4).

2. **Subdivision invariance** achieved through two cooperating mechanisms: (a) super-node

merging of tangent-continuous chains, and (b) length-proportional marginals — so that the same shape yields comparable node sets and identical total mass regardless of how finely it is subdivided (Sections 3.3 and 3.6).

3. **A dimensionally consistent, auto-balanced pFGW objective**, in which the feature and structure terms are rendered commensurate by construction and then exactly balanced at a reference transport plan, leaving a single interpretable trade-off knob (Sections 3.7 and 3.8).

4. **Automatic selection of the transported mass** $m$ — the unknown overlap — by a robust jump detector on the transport-cost curve, eliminating the only manual threshold (Section 3.9).

5. **Interpretable correspondence extraction** that unpacks the solved transport plan to the original primitives as one-to-one and one-to-many matches, declares unmatched content, and provides colour-coded visualizations for verification (Section 3.10), with a proof that unpacking preserves the marginals (Proposition 3.7).

6. **An evaluation framework with ground truth known by construction.** To score the matcher objectively without hand-labelling, we contribute a reusable pipeline that synthesizes source-to-target design-edit pairs whose exact correspondence is recorded as a by-product of the edit (Chapter 4). A vision–language model selects realistic edits and a deterministic engine executes the exact geometry, so that every target is valid and its ground truth is exact.

## 1.6 Thesis Outline

The remainder of this thesis is organized as follows.

- **Chapter 2 (Background)** establishes the mathematical foundation: measure networks and couplings, the Wasserstein distance (feature matching), the Gromov–Wasserstein distance (structure matching), their fusion into the Fused Gromov–Wasserstein distance, and the partial relaxation that yields Partial Fused Gromov–Wasserstein.

- **Chapter 3 (Methodology)** is the technical heart of the thesis. It develops the full pipeline stage by stage — the primitive layer, super-node merging, the contact-graph structure space, frame normalization, length-proportional marginals, the feature space and its auto-balancing, the matching engine, automatic mass selection, and correspondence extraction — motivating each non-trivial design decision.

- **Chapter 4 (Evaluation)** presents the ground-truth evaluation framework, the hybrid edit generator, the protocol and metrics, the quantitative and qualitative results, and a discussion of the method's one substantial failure mode.
- **Chapter 5 (Conclusion and Future Work)** summarizes the contributions and results, then looks ahead to the most consequential extension — lifting the evaluation generator from primitives to a design-aware feature/parametric model and scaling the framework into a large public benchmark.

# Chapter 2

# Background

This chapter establishes the mathematical foundation for the matching pipeline. To robustly align CAD drawings we rely on the framework of optimal transport (OT), extend it to compare both geometric features and topological structure, and finally relax it to a *partial* formulation that tolerates content present in one drawing but absent from the other.

## 2.1 Measure Networks and Couplings

To represent a CAD drawing mathematically we use the notion of a *measure network*, in the sense of Chowdhury and Mémoli [5], who introduced measure networks and the network Gromov–Wasserstein distance between them.

**Definition 2.1 (Measure network)** *A measure network is a triple $G = (V, p, W)$ comprising a finite set of nodes $V = \{v_i\}_{i=1}^n$ (each node a vector in $\mathbb{R}^d$ describing a particular primitive instance), a probability mass function $p \in \Delta^{n-1} := \{p \in \mathbb{R}^n_{\geq 0} : \sum_{i=1}^n p_i = 1\}$, and a structural matrix $W \in \mathbb{R}^{n \times n}$ encoding pairwise relationships such as shortest-path hop distances on a graph derived from the drawing.*

A fundamental goal in OT is to find a *coupling* between two such distributions $p \in \Delta^{n-1}$ and $q \in \Delta^{m-1}$. A full coupling is a matrix $C \in \mathcal{C}(p, q)$ defined by

$$\mathcal{C}(p, q) = \Big\{ C \in \mathbb{R}_+^{n \times m} : \; C\mathbf{1}_m = p, \; C^\top \mathbf{1}_n = q \Big\}, \tag{2.1}$$

where the entry $C_{ij}$ is the amount of mass transported from node $i$ of the source to node $j$ of the target, and $\mathbf{1}_n$ denotes the all-ones vector in $\mathbb{R}^n$. The marginal constraints say that node $i$ ships out exactly its mass $p_i$ and node $j$ receives exactly its mass $q_j$.

## 2.2 Integrating Features and Structure

Standard OT can compare objects by either their local features or their global structure. We will need both.

**The Wasserstein distance (feature matching).** When comparing local attributes (such as position or length), we measure the dissimilarity between feature vectors $a_i \in V_1$ and $b_j \in V_2$ by a ground distance $d_{\mathcal{A}}$. The $\mathsf{q}$-Wasserstein distance, a classical construction of optimal transport [23, 13], is

$$d_{W_{\mathsf{q}}}(G_1, G_2) = \min_{C \in \mathcal{C}(p,q)} \Big( \sum_{i,j} d_{\mathcal{A}}(a_i, b_j)^{\mathsf{q}}\, C_{ij} \Big)^{1/\mathsf{q}}. \tag{2.2}$$

For the Euclidean case $\mathsf{q} = 2$ this reads $d_{W_2}(G_1, G_2) = \min_C \big( \sum_{i,j} \|a_i - b_j\|^2 C_{ij} \big)^{1/2}$. The objective is *linear* in the coupling $C$. Its weakness for our problem is visible in Figure 2.1(a): because it compares absolute positions, it pays a large cost whenever the two drawings sit in different frames, even when they describe the same shape.

**The Gromov–Wasserstein distance (structure matching).** CAD parts often appear in different frames, or are deformed so that absolute positions change while internal structure is preserved. To handle this we use the $\mathsf{q}$-Gromov–Wasserstein (GW) distance, which compares the structural matrices $W_1$ and $W_2$ rather than the coordinates directly:

$$d_{GW_{\mathsf{q}}}(G_1, G_2) = \tfrac{1}{2} \min_{C \in \mathcal{C}(p,q)} \Big( \sum_{i,j,k,l} \big| W_1(i,k) - W_2(j,l) \big|^{\mathsf{q}}\, C_{ij} C_{kl} \Big)^{1/\mathsf{q}}. \tag{2.3}$$

The objective is *quadratic* in $C$: it looks at *pairs* of matches at once, charging a cost whenever a pair $(i,k)$ in $G_1$ whose internal relation is $W_1(i,k)$ is matched to a pair $(j,l)$ in $G_2$ whose internal relation $W_2(j,l)$ disagrees. Because only *intra*-space relations enter, the GW distance enjoys a strong invariance, which is the formal reason our method is robust to frame differences (Figure 2.1(b)).

Because the objective in (2.3) depends on the two measure networks only through their structural matrices $W_1, W_2$ and their masses — and never through the coordinates of any node — the Gromov–Wasserstein distance is invariant to isometries: it is unchanged by any transformation that preserves the structural matrices, and it vanishes when the two networks are isometric copies of one another.

In Chapter 3 we take $W$ to be a matrix of contact-graph hop distances, which is itself invariant to translation, rotation, reflection, and (after our normalization) scale. This invariance is then exactly why the structure term of our objective recognizes the same layout across all of

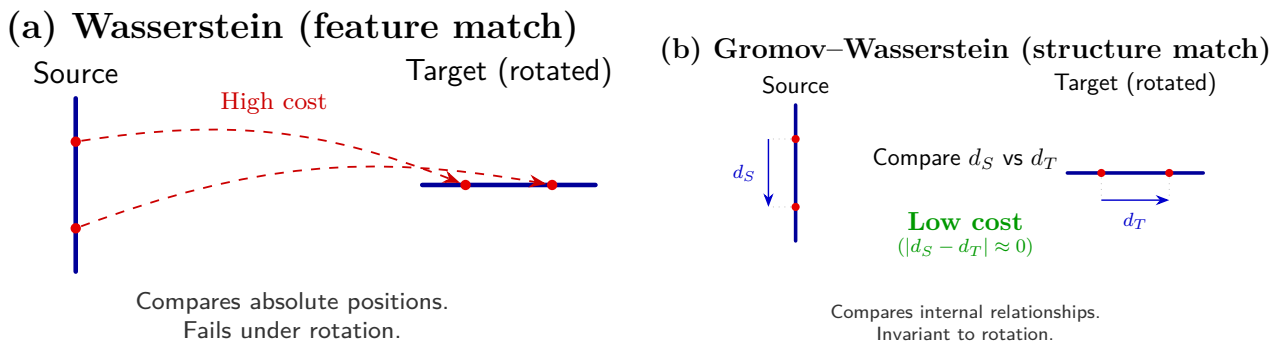



Figure 2.1: **Comparison of transport objectives.** (a) The Wasserstein distance compares features (absolute positions) directly and so pays a high cost when the Target is in a different frame. (b) The Gromov–Wasserstein distance compares the *internal* structure (pairwise distances), and is therefore invariant to rigid motion.

these frame differences.

## 2.3 Fused Gromov–Wasserstein

For our application we need the best of both worlds: matching primitives that look similar (Wasserstein) *and* connect similarly (Gromov–Wasserstein). We therefore adopt the Fused Gromov–Wasserstein (FGW) distance [22], which balances the two terms with a trade-off parameter $\alpha \in [0, 1]$:

$$d_{FGW_q}(G_1, G_2) = \min_{C \in \mathcal{C}(p,q)} \left[ (1-\alpha) \sum_{i,j} d_{\mathcal{A}}(a_i, b_j)^q \, C_{ij} + \alpha \sum_{i,j,k,l} \left| W_1(i,k) - W_2(j,l) \right|^q C_{ij} C_{kl} \right]^{1/q}. \quad (2.4)$$

The trade-off interpolates between the two regimes: $\alpha = 0$ recovers pure feature matching (Wasserstein), $\alpha = 1$ recovers pure structure matching (Gromov–Wasserstein), and intermediate values fuse the two. Our pipeline uses $\alpha = 0.5$; crucially, Chapter 3 renders the two terms commensurate so that $\alpha = 0.5$ is a genuine half-and-half balance rather than an accident of units.

## 2.4 Partial Fused Gromov–Wasserstein

A key challenge in CAD matching is that drawings are only *partially* similar. Standard OT requires the total mass of both distributions to be equal and fully transported, which forces incorrect matches whenever some content has no counterpart (Figure 2.2(a)). To address this we employ *Partial Optimal Transport*: a partial coupling transports only a prescribed fraction

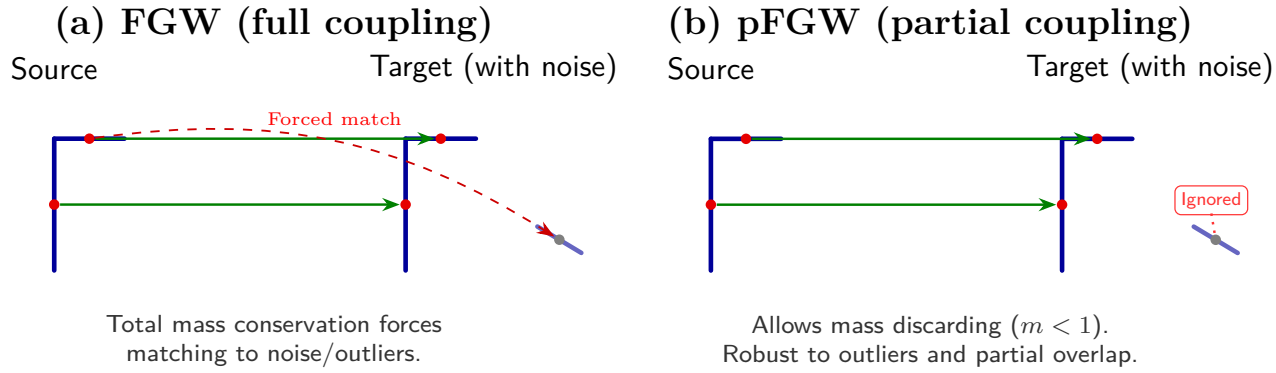



Figure 2.2: **Robustness of partial transport.** (a) Standard FGW forces a full coupling, producing erroneous matches between valid geometry and clutter. (b) Partial FGW (pFGW) selectively matches similar structures while discarding the mass associated with noise.

$m$ of the mass,

$$\mathcal{C}_m(p,q) = \Big\{ C \in \mathbb{R}_+^{n_1 \times n_2} : \; C\mathbf{1}_{n_2} \le p, \; C^\top \mathbf{1}_{n_1} \le q, \; \mathbf{1}_{n_1}^\top C \mathbf{1}_{n_2} = m \Big\}. \tag{2.5}$$

Each node may now ship *at most* its mass and keep the rest at home, while the global constraint pins only the total transported mass to $m \le 1$. Fixing the transported mass with a hard constraint is one of two standard ways to relax balanced transport; the other replaces it with a soft divergence penalty, as in unbalanced optimal transport [4] and its Gromov extension [20]. We adopt the partial form precisely because pinning the transported fraction $m$ explicitly is what later exposes the unknown overlap we wish to recover (Section 3.9). Combining this relaxed feasible set with the FGW objective yields the Partial Fused Gromov–Wasserstein (pFGW) distance [10], which for $\mathsf{q} = 2$ reads

$$d_{pFGW_2}(G_1, G_2) = \min_{C \in \mathcal{C}_m(p,q)} \Big[ (1-\alpha) \sum_{i,j} \|a_i - b_j\|^2 \, C_{ij} + \alpha \sum_{i,j,k,l} \big( W_1(i,k) - W_2(j,l) \big)^2 \, C_{ij} C_{kl} \Big]^{1/2}. \tag{2.6}$$

The minimizer $C$ serves as a probabilistic soft correspondence that is robust to noise and partial overlap: as illustrated in Figure 2.2(b), the optimizer spends its mass budget on the credible matches first and simply declines to ship the mass of primitives that have no good counterpart, instead of force-matching them to garbage. The remainder of this thesis makes every term of (2.6) concrete for CAD data and explains how the overlap fraction $m$ is chosen automatically.

# Chapter 3

# Methodology

This chapter describes the proposed method for establishing correspondences between two CAD drawings. The method combines local geometric similarity with global structural consistency, is invariant to subdivision of primitives, and explicitly supports partial matching when the Target is incomplete or contains clutter. We present the pipeline as a sequence of stages, motivating each design decision, including the intricate per-type formulas, and tolerance arguments.

**A note on notation.** As we specialize the general transport formulation of Chapter 2 to CAD data, two symbols are deliberately renamed to standard usage and to free $C$ for the structure cost. The *coupling*, written $C$ in the background, becomes $\boldsymbol{\pi}$ from here on (so $\pi_{ij}$ is the transported mass); and the intra-drawing *structure matrix*, written $W$ in the background, becomes $\boldsymbol{C}$ (with per-drawing copies $\boldsymbol{C}^{(1)}, \boldsymbol{C}^{(2)}$). The cross-drawing feature cost is written $\boldsymbol{M}$. The trade-off $\alpha$ and the transported fraction $m$ keep their meanings.

## 3.1 Pipeline Overview

The pipeline (Figure 3.1) turns each drawing into a measured metric–feature space $(\boldsymbol{F}, \boldsymbol{C}, \boldsymbol{p})$, computes cross-drawing costs, solves a pFGW problem at an automatically selected mass, and extracts the correspondence. Concretely, a drawing with $n$ primitives is represented by

$$\boldsymbol{F} \in \mathbb{R}^{n\times 10}, \qquad \boldsymbol{C} \in \mathbb{R}^{n\times n}_{\geq 0}, \qquad \boldsymbol{p} \in \Delta^{n-1}, \tag{3.1}$$

where row $\boldsymbol{f}_i$ of $\boldsymbol{F}$ is a geometric descriptor of primitive $i$ (Section 3.7), entry $C_{ik}$ is an intra-drawing structure cost (a normalized graph distance, Section 3.4), and $p_i$ is the probability mass of primitive $i$ (proportional to its arc length, Section 3.6). The correspondence between drawing $A = (\boldsymbol{F}^{(1)}, \boldsymbol{C}^{(1)}, \boldsymbol{p})$ with $n_1$ primitives and $B = (\boldsymbol{F}^{(2)}, \boldsymbol{C}^{(2)}, \boldsymbol{q})$ with $n_2$ primitives is a

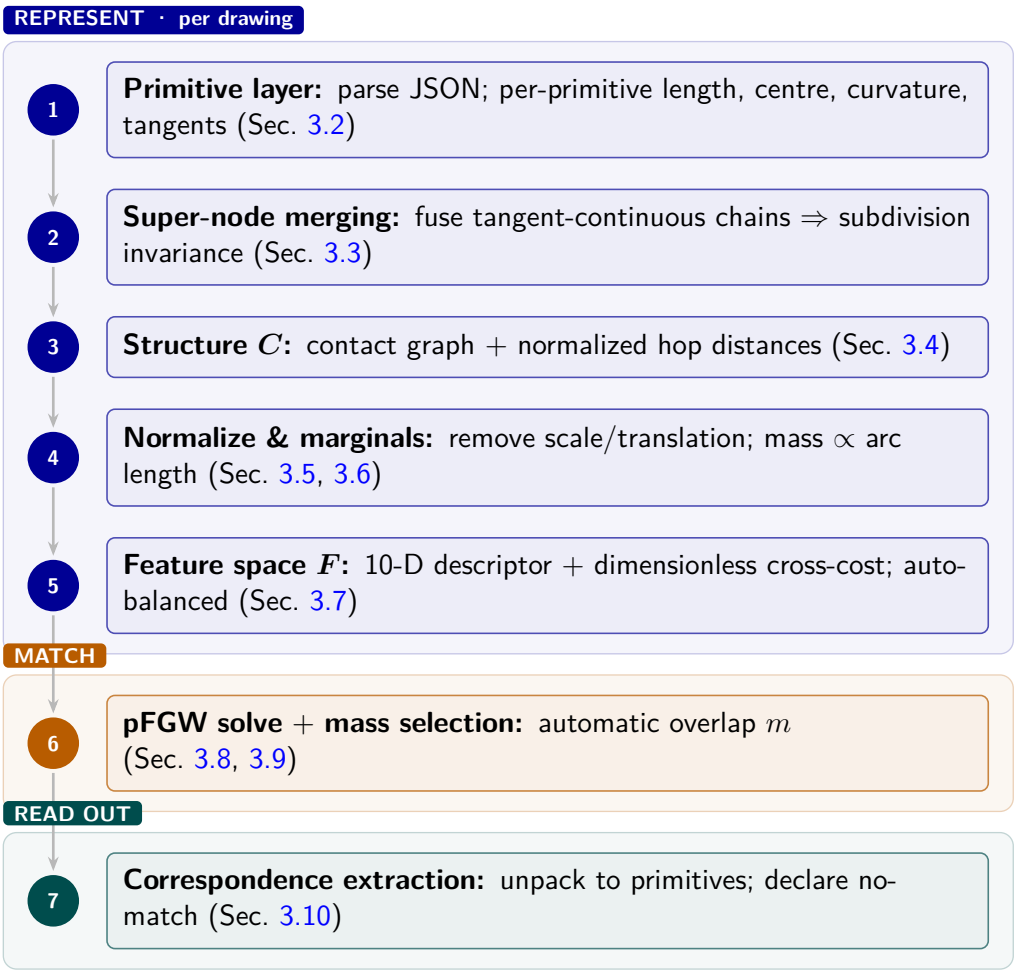



Figure 3.1: **Pipeline overview.** Each drawing is independently lifted to a measured metric–feature space; the two spaces are then matched by partial FGW with automatic mass selection, and interpretable correspondences are extracted from the transport plan.

coupling $\boldsymbol{\pi} \in \mathbb{R}_{\geq 0}^{n_1 \times n_2}$, where $\pi_{ij}$ is the mass shipped from primitive $i$ of $A$ to primitive $j$ of $B$; a large $\pi_{ij}$ means "$i$ corresponds to $j$".

This single modelling choice — optimal transport — purchases all three robustness requirements of Section 1.2 at once. The feature (Wasserstein) term compares descriptors *across* drawings ("this primitive looks like that one"); the structure (Gromov–Wasserstein) term compares pairwise relations *within* each drawing ("these two are two hops apart in $A$, so their matches should be two hops apart in $B$") and is therefore invariant to rigid motion; and the partial relaxation lets primitives without a true counterpart simply not ship their mass instead of being force-matched.

## 3.2 The Primitive Layer: From JSON to Geometry

**What the input looks like.** The inputs are CAD *drawings*: non-isometric two-dimensional views (orthographic projections) of CAD models, each exported as a JSON list of primitives. Although the views are planar, the exporter records every coordinate in full three dimensions as $(x, y, z)$, with the out-of-plane component $z$ held *constant* throughout a drawing. We deliberately keep these coordinates exactly as given and never strip the third one, so every per-primitive point we compute — the centre $\boldsymbol{c}_i \in \mathbb{R}^3$, the samples $\boldsymbol{S}_i \in \mathbb{R}^{K_i \times 3}$, and the endpoints — lives in $\mathbb{R}^3$. Because $z$ is the same for every primitive of a planar drawing, it carries no discriminative information and is removed by the frame normalization of Section 3.5, so the pipeline behaves as a two-dimensional matcher. Crucially, nothing downstream assumes $z$ is constant: the *identical* pipeline applies without modification to genuinely three-dimensional drawings, of which a planar drawing is simply the special case where the third coordinate happens to be fixed.

The pipeline begins by reading the raw JSON and turning every primitive, whatever its type, into a single *uniform record*, so that all later stages can treat primitives identically and never have to special-case the geometry. A drawing contains five kinds of primitive: straight *line segments*, *circular arcs*, *elliptical arcs*, full *circles*, and (possibly) *B-spline / NURBS* curves. Each is reduced to the same set of fields:

- a *type label* $\tau_i \in \{0, 1, 2, 3, 4\}$, a single integer naming the kind of curve;
- a *centre* $\boldsymbol{c}_i \in \mathbb{R}^3$, the point at the middle of the curve measured along its length;
- an *arc length* $\ell_i$, the true length of the curve (not the straight-line distance between its ends);
- $K_i$ *sample points* $\boldsymbol{S}_i \in \mathbb{R}^{K_i \times 3}$, placed at equal arc-length spacing along the curve;
- the curve's *endpoints* (a line has two; a closed curve has none);
- a *curvature profile* $\boldsymbol{\kappa}_i \in \mathbb{R}^{32}$ and its average, the scalar *mean curvature* $\kappa_i$; and
- *unit endpoint tangents*: at each endpoint, the unit vector that points along the curve, away from that endpoint and into the body of the curve.

Each field is explained in turn below, followed by the exact per-type constructions used to compute them.

**The type label is an index, not a number to compute with.** The type label $\tau_i$ is an ordinary integer that merely *names* one of the five primitive types: 0 =line, 1 =circular arc, 2 =elliptical arc, 3 =circle, 4 =spline. The integer is never used in any arithmetic: doing so

would falsely imply that "circle" (3) is closer to "spline" (4) than to "line" (0). Instead it indexes a five-slot one-hot vector $\boldsymbol{e}(\tau_i) \in \{0,1\}^5$ — a vector with a single 1 in the slot for that type and 0 everywhere else — and that one-hot is what the descriptor uses (Lemma 3.1). A merged chain (Section 3.3) is *not* given a separate "composite" type; it inherits the type of its longest member, so no sixth slot is ever needed.

**Mean curvature versus curvature profile.** The record stores two curvature quantities, which answer different questions. The *curvature profile* $\boldsymbol{\kappa}_i \in \mathbb{R}^{32}$ is a *vector*: it records how sharply the curve bends at 32 stations spaced evenly along its length, capturing *where* along the curve the bending happens — a spline that runs straight and then curls looks very different from a uniformly curved arc. The *mean curvature* $\kappa_i \in \mathbb{R}$ is a single number, the average of those 32 values: a one-figure summary of how curved the primitive is overall (a line has $\kappa_i = 0$; a circle of radius $r$ has $\kappa_i = 1/r$). The descriptor uses the scalar $\kappa_i$; the profile is the intermediate it is averaged from.

**Into-curve endpoint tangents, and why circles are special.** At an endpoint of an open curve, the *into-curve unit tangent* is the direction in which one would set off when starting at that endpoint and travelling along the curve — a unit vector pointing *inward*, into the body of the curve. A line from $\boldsymbol{a}$ to $\boldsymbol{b}$ has into-curve tangent $(\boldsymbol{b}-\boldsymbol{a})/\|\boldsymbol{b}-\boldsymbol{a}\|$ at end $\boldsymbol{a}$ and the reverse at end $\boldsymbol{b}$; both point back into the segment. These tangents are what let the merge test of Section 3.3 decide whether two primitives meet smoothly. A full *circle* — and any closed loop — has *no endpoints*, so it stores none. This has two concrete consequences used later: a closed curve can never be merged through the endpoint-sharing test (Section 3.3), and in the contact graph it can only be found adjacent to a neighbour through the *sample*-touch test rather than the endpoint-touch test (Section 3.4) — which is precisely why that sample-touch test exists.

**Sampling density: the count $K_i$ and the spacing $\varepsilon$.** Curved primitives must be represented by points for the graph and curvature computations; the question is how many.

**Definition 3.1 (Sample count and data-driven spacing)** *Each primitive is sampled at equal arc-length steps, with a count*

$$K_i = \mathrm{clip}\Big(\lceil \ell_i/\varepsilon \rceil + 1,\ K_{\min},\ K_{\max}\Big), \qquad K_{\min} = 8,\ K_{\max} = 400, \tag{3.2}$$

*where* $\mathrm{clip}(x,a,b)$ *clamps* $x$ *into* $[a,b]$. *With* $\ell_{\max} = \max_i \ell_i$ *the longest primitive and* $D = \mathrm{diam}(\{\boldsymbol{c}_i\})$ *the diameter of the centre set, the spacing is*

$$\varepsilon = \max\Big(\ell_{\max}/K_{\max},\ 10^{-4} D\Big). \tag{3.3}$$

Reading (3.2) from the inside out, $\lceil \ell_i/\varepsilon \rceil + 1$ is the number of points needed so that neighbouring samples are no more than a distance $\varepsilon$ apart on a curve of length $\ell_i$; the clamp then guarantees at least $K_{\min} = 8$ points (enough to form a meaningful curvature window) and at most $K_{\max} = 400$ points — a hard ceiling, the *sample budget*, that caps the cost of every downstream step. The spacing $\varepsilon$ is never set by hand: the first term of (3.3) is the spacing at which the longest primitive exactly uses up its budget, and the second is a floor against degenerate (near-zero-length) geometry. Because both terms scale with the drawing, $\varepsilon$ adapts automatically to size and is not a fixed magic constant.

**One spacing, two jobs, proved sound.** The same $\varepsilon$ is reused in Section 3.4 as the radius for deciding whether two curves *touch*, and this reuse is not a coincidence but a guarantee.

**Proposition 3.1 (Touch-test soundness)** *Suppose every curve is sampled so that consecutive samples are at most $\varepsilon$ apart in arc length, and that every endpoint is itself a sample. If two curves intersect (their minimum distance is 0), then there exist a sample of one and a sample of the other within Euclidean distance $\varepsilon$ of each other.*

**Proof:** Let $\boldsymbol{x}$ lie on curve $A$ and $\boldsymbol{y}$ on curve $B$ with $\boldsymbol{x} = \boldsymbol{y}$ (an intersection point). Walk along $A$ from $\boldsymbol{x}$ to the nearer of its two flanking samples; the arc-length travelled is at most $\varepsilon/2$, since consecutive samples are at most $\varepsilon$ apart and $\boldsymbol{x}$ lies between two of them (or is itself a sample, giving distance 0). The Euclidean distance is no larger than the arc length, so the nearest sample $\boldsymbol{s}_A$ of $A$ satisfies $\|\boldsymbol{s}_A - \boldsymbol{x}\| \le \varepsilon/2$, and likewise $\|\boldsymbol{s}_B - \boldsymbol{y}\| \le \varepsilon/2$ for the nearest sample of $B$. By the triangle inequality,

$$\|\boldsymbol{s}_A - \boldsymbol{s}_B\| \le \|\boldsymbol{s}_A - \boldsymbol{x}\| + \|\boldsymbol{x} - \boldsymbol{y}\| + \|\boldsymbol{y} - \boldsymbol{s}_B\| \le \tfrac{\varepsilon}{2} + 0 + \tfrac{\varepsilon}{2} = \varepsilon.$$

□ Thus a single $k$-d-tree range query at radius $\varepsilon$ cannot miss a true crossing: sampling density and touch tolerance are one coherent quantity, not two independently tuned knobs.

**The centre as the mid-arc-length point.** For a curve $P(s)$ parametrized by arc length $s \in [0, \ell]$, the centre is the midpoint *of the curve*, $\boldsymbol{c} := P(\ell/2)$, rather than the midpoint of the chord joining its ends or the centroid of an enclosed area. The chord midpoint can land in empty space — for a semicircle it sits well off the arc — whereas $P(\ell/2)$ always lies on the geometry; for a full circle the geometric centre is used. This treats each primitive as a uniform one-dimensional mass spread along its length, the same view used by the marginals (Section 3.6).

**Discrete curvature: the Menger formula.** Curvature at a sampled point is estimated from three consecutive points by the *Menger curvature* [9], the reciprocal of the radius of the

unique circle through them.

**Definition 3.2 (Menger curvature)** *For three non-collinear points* $\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c} \in \mathbb{R}^3$,

$$\kappa(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}) = \frac{2\,\|(\boldsymbol{b}-\boldsymbol{a}) \times (\boldsymbol{c}-\boldsymbol{a})\|}{\|\boldsymbol{a}-\boldsymbol{b}\|\,\|\boldsymbol{b}-\boldsymbol{c}\|\,\|\boldsymbol{c}-\boldsymbol{a}\|}. \tag{3.4}$$

**Proposition 3.2 (Menger curvature is the inverse circumradius)** *The quantity* (3.4) *equals* $1/R$*, where* $R$ *is the radius of the unique circle (the circumcircle) through* $\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}$.

**Proof:** For a triangle with side lengths $|\boldsymbol{a}-\boldsymbol{b}|, |\boldsymbol{b}-\boldsymbol{c}|, |\boldsymbol{c}-\boldsymbol{a}|$ and area $T$, the circumradius is $R = \dfrac{\|\boldsymbol{a}-\boldsymbol{b}\|\,\|\boldsymbol{b}-\boldsymbol{c}\|\,\|\boldsymbol{c}-\boldsymbol{a}\|}{4T}$. The area equals $T = \frac{1}{2}\|(\boldsymbol{b}-\boldsymbol{a}) \times (\boldsymbol{c}-\boldsymbol{a})\|$, since the cross product's magnitude is the area of the parallelogram spanned by the two edge vectors. Substituting,

$$\frac{1}{R} = \frac{4T}{\|\boldsymbol{a}-\boldsymbol{b}\|\,\|\boldsymbol{b}-\boldsymbol{c}\|\,\|\boldsymbol{c}-\boldsymbol{a}\|} = \frac{2\,\|(\boldsymbol{b}-\boldsymbol{a}) \times (\boldsymbol{c}-\boldsymbol{a})\|}{\|\boldsymbol{a}-\boldsymbol{b}\|\,\|\boldsymbol{b}-\boldsymbol{c}\|\,\|\boldsymbol{c}-\boldsymbol{a}\|},$$

which is (3.4). □ The cross-product form (3.4) is preferred over the algebraically equivalent Heron's-formula version because it stays numerically stable on the thin, nearly straight triangles that arise along almost-straight sections. Sliding this three-point window over 32+2 equally spaced points yields the 32 entries of the profile $\boldsymbol{\kappa}_i$, whose mean is the scalar $\kappa_i$.

### 3.2.1 Per-Type Constructions

Four of the five types — lines, circular arcs, elliptical arcs, and circles — are *analytic*: their length, curvature, and endpoint tangents have exact closed-form formulas and so are computed exactly rather than approximated numerically. The fifth (splines/NURBS) has no closed form and is handled by dense evaluation.

**Line.** For endpoints $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{R}^3$, $P(t) = \boldsymbol{a} + t(\boldsymbol{b}-\boldsymbol{a})$, $t \in [0,1]$, with $\ell = \|\boldsymbol{b}-\boldsymbol{a}\|$, centre $\boldsymbol{c} = \frac{1}{2}(\boldsymbol{a}+\boldsymbol{b})$, curvature $\kappa \equiv 0$, and into-curve unit tangents $\pm(\boldsymbol{b}-\boldsymbol{a})/\|\boldsymbol{b}-\boldsymbol{a}\|$ at the two ends.

**Circular arc.** The export supplies a centre $\boldsymbol{o}$, radius $r$, endpoints, and a plane normal $\boldsymbol{n}$. A right-handed orthonormal in-plane basis $(\boldsymbol{u}, \boldsymbol{v})$ is built by Gram–Schmidt against $\boldsymbol{n}$, followed by $\boldsymbol{v} = \boldsymbol{n} \times \boldsymbol{u}$, and the arc is

$$P(\theta) = \boldsymbol{o} + r\cos\theta\,\boldsymbol{u} + r\sin\theta\,\boldsymbol{v}, \tag{3.5}$$

with start/end angles recovered by $\theta = \operatorname{atan2}\Big(\langle \boldsymbol{x}-\boldsymbol{o},\, \boldsymbol{v}\rangle,\ \langle \boldsymbol{x}-\boldsymbol{o},\, \boldsymbol{u}\rangle\Big)$ and sweep $\Delta\theta = \theta_e - \theta_s$ canonicalized into $(0, 2\pi]$ (the normal encodes the sweep direction, with $\Delta\theta = 2\pi$ the full-

circle edge case). Arc length and curvature are exact: $\ell = r\,\Delta\theta$, $\kappa = 1/r$. Differentiating, $dP/d\theta = -\sin\theta\,\boldsymbol{u} + \cos\theta\,\boldsymbol{v}$, already a unit vector since $\boldsymbol{u} \perp \boldsymbol{v}$; the into-curve tangent is $+dP/d\theta$ at the start and $-dP/d\theta$ at the end. (Why analytic tangents matter: Section 3.3.)

**Elliptical arc.** With semi-axes $a, b$ along unit directions $\boldsymbol{u}, \boldsymbol{v}$,

$$P(\theta) = \boldsymbol{o} + a\cos\theta\,\boldsymbol{u} + b\sin\theta\,\boldsymbol{v}, \qquad \|P'(\theta)\| = \sqrt{a^2\sin^2\theta + b^2\cos^2\theta}, \tag{3.6}$$

and the eccentric angle of an endpoint $\boldsymbol{x}$ is recovered by the component-wise-rescaled atan2, $\theta = \operatorname{atan2}\!\Big(\frac{1}{b}\langle \boldsymbol{x}-\boldsymbol{o},\,\boldsymbol{v}\rangle,\ \frac{1}{a}\langle \boldsymbol{x}-\boldsymbol{o},\,\boldsymbol{u}\rangle\Big)$. Let $a_{\max} = \max(a,b)$, $a_{\min} = \min(a,b)$, and define the elliptic parameter $m_e = 1-(a_{\min}/a_{\max})^2$ (the squared eccentricity). With the incomplete elliptic integral of the second kind [1] $E(\varphi \mid m_e) = \int_0^{\varphi}\sqrt{1 - m_e\sin^2 t}\,dt$, two cases arise depending on which axis is major:

$$a \geq b: \quad \|P'(\theta)\| = a\sqrt{1 - m_e\cos^2\theta} \;\Rightarrow\; \ell = a\Big|E(\tfrac{\pi}{2} - \theta_s \mid m_e) - E(\tfrac{\pi}{2} - \theta_s - \Delta\theta \mid m_e)\Big|, \tag{3.7}$$

$$b > a: \quad \|P'(\theta)\| = b\sqrt{1 - m_e\sin^2\theta} \;\Rightarrow\; \ell = b\Big|E(\theta_s + \Delta\theta \mid m_e) - E(\theta_s \mid m_e)\Big|, \tag{3.8}$$

where the first case uses the phase shift $\cos^2\theta = \sin^2(\frac{\pi}{2}-\theta)$ to reach the canonical form of $E$. The closed-form curvature is $\kappa(\theta) = ab/(a^2\sin^2\theta + b^2\cos^2\theta)^{3/2}$, evaluated at the $\theta$-values of the 32 uniform-arc-length stations, and the endpoint tangents come from $P'(\theta) = -a\sin\theta\,\boldsymbol{u} + b\cos\theta\,\boldsymbol{v}$, normalised. Because equal steps in $\theta$ are *not* equal steps in arc length (the spacing varies by up to a factor $a/b$), the curve is first traced as a fine $\theta$-uniform polyline (up to 4096 points), its running chord length accumulated, and then resampled at $K_i$ points genuinely uniform in arc length.

**Circle.** $\ell = 2\pi r$, $\kappa = 1/r$, and no endpoints — which matters for graph construction, since closed curves can only connect to neighbours through the sample-touch test (Definition 3.4).

**Spline / NURBS.** The curve is evaluated to a fine polyline by an exact rational NURBS evaluator [15] when available, falling back to a non-rational B-spline evaluation on the valid knot range and finally to the control polygon. Length and samples come from the polyline; the curvature profile is computed on the *fine* polyline, not the sparse $K_i$ downsamples, because discrete curvature on a sparse polyline would measure the straight interpolating segments rather than the true curve.

## 3.3 Super-Node Merging for Subdivision Invariance

The hardest practical obstacle in CAD matching is that two exports of the *same* shape may chop it into different numbers of pieces. A slot that one drawing exports as a single closed polyline may appear in the other as four separate tangent-continuous arcs and lines. A matcher that pairs primitives one-to-one then fails on the mismatched counts alone, before any geometry is even considered. The fix is a preprocessing pass that, in each drawing independently, *merges every maximal chain of smoothly-joined primitives into one composite "super-node,"* so that both drawings present comparable node sets no matter how finely each was subdivided. "Smoothly joined" here means $G^1$*-continuous*: the pieces meet end-to-end and share a common tangent direction at the join, so there is no corner between them.

**Definition 3.3 ($G^1$ merge test)** *Two open primitives $i, j$ are mergeable when both conditions hold: (a) they share an endpoint, i.e. an end of one lies within a tight weld tolerance $\varepsilon_{\mathrm{w}} = 10^{-6}D$ of an end of the other; and (b) their into-curve unit tangents $\boldsymbol{t}_i, \boldsymbol{t}_j$ at that shared point are collinear to within $\theta_{\mathrm{tol}} = 5°$:*

$$|\langle \boldsymbol{t}_i,\, \boldsymbol{t}_j \rangle| \;\geq\; \cos\theta_{\mathrm{tol}}. \tag{3.9}$$

Here $\langle \boldsymbol{t}_i,\, \boldsymbol{t}_j \rangle$ is the dot product of the two unit tangents, which equals the cosine of the angle between them, so the test asks that this angle be at most 5°. The absolute value is essential: the tangents point *into* their own curves, so at a smooth join they point in opposite directions and $\langle \boldsymbol{t}_i,\, \boldsymbol{t}_j \rangle \approx -1$, whereas a sharp hairpin reversal gives $+1$; both are collinear (the same line, at 0° or 180°), and taking the magnitude accepts the smooth case correctly.

**Two tolerances, three orders of magnitude apart.** Two different distances appear in the test and must not be confused. The *weld tolerance* $\varepsilon_{\mathrm{w}} = 10^{-6}D$ decides whether two endpoints are literally the *same* point; the *sampling spacing* $\varepsilon \approx \ell_{\max}/400$ (Definition 3.1) decides whether two curves merely *pass near* each other. The weld tolerance is far tighter — by about three orders of magnitude — because endpoint coincidence in a CAD export is essentially exact, whereas welding at the much larger sampling radius would glue genuinely distinct but nearby features (hatching, closely spaced dimension lines) into one. The 5° angular tolerance, in turn, absorbs floating-point tangent noise while still rejecting genuine corners. The two tolerances answer different questions and deliberately operate at different scales.

**Why the tangents must be analytic.** The 5° test is only meaningful if the tangents themselves are accurate, and this rules out estimating them from sample chords. On a circular arc the chord from $P(\theta_0)$ to $P(\theta_0 + \delta)$ is parallel to the tangent at the midpoint angle $\theta_0 + \frac{\delta}{2}$, so it makes an angle of exactly $\delta/2$ with the endpoint tangent at $P(\theta_0)$; estimating that tangent

from the first sample chord of a 90° fillet at the sampling floor $K_{\min} = 8$ (seven segments, $\delta = 90°/7 \approx 12.9°$) is therefore already off by $\delta/2 \approx 6.4°$, past the 5° merge tolerance. A chord-based test would therefore accept or reject a genuine $G^1$ join *depending on how densely the curve happened to be sampled*, which is unacceptable. Storing *analytic* endpoint tangents (exact for lines, arcs, and ellipses; fine-polyline chords for the densely evaluated splines) makes the merge decision independent of sampling density.

**Grouping, ordering, and the composite record.** The pairwise decisions are combined into whole chains with a *union–find* (disjoint-set) data structure [21] — a standard tool that groups items into connected clusters: every mergeable pair links its two primitives, and each resulting cluster is one candidate chain. The members of a chain must then be put in order along the curve, which is done by *walking the junction-adjacency graph*: every member of a valid $G^1$ chain has at most two neighbours, so one (1) finds a member with a *free* endpoint (one not shared with any other member; if none exists the chain is a closed loop and the walk starts anywhere), (2) orients the current member's samples to begin at the junction just entered, and (3) hops to the unvisited neighbour whose endpoint set contains the current exit point, repeating; this graph walk is correct for open and closed chains alike. If the walk cannot visit every member — which happens at a junction where three or more pieces meet tangentially — the merge is *refused* and those pieces stay atomic, a deliberately conservative failure mode. A successful super-node then receives an arc length $\ell = \sum_k \ell_k$ (exact additivity), samples and a curvature profile recomputed on the full-resolution walked polyline, a `members` list recording the original primitives (used to unpack the result in Section 3.10), and the *type of its longest member*. We deliberately do *not* introduce a separate "composite" type: a distinct composite label would make the merged chain look maximally different from the same shape exported as a single primitive in the other drawing, charging a full type-mismatch penalty and fighting the very subdivision-invariance the merge is meant to provide. This is also why the type one-hot needs only five slots (Section 3.7), not six.[1]

## 3.4 The Structure Space: Contact Graph and Shortest Paths

The structure matrix $\boldsymbol{C}$ records, for each drawing on its own, *how its primitives are connected* — which ones are neighbours, and how many steps apart any two are. This is the information the Gromov–Wasserstein term compares between drawings, and it is what lets the method

[1] The implementation accordingly uses a five-slot type one-hot, giving the ten-dimensional descriptor of (3.13).

recognize the same layout after translation, rotation, or rescaling. We build $\boldsymbol{C}$ in two steps: first a contact graph, then shortest paths on that graph.

**Definition 3.4 (Contact graph)** *The vertices are the post-merge primitives (the super-nodes). An undirected edge $\{i, j\}$ — meaning "these two physically meet" — is placed whenever at least one of the following holds:*

*(1)* endpoint touch: *an endpoint of $i$ lies within the weld tolerance $\varepsilon_{\mathrm{w}}$ of an endpoint of $j$;*

*(2)* sample touch: *a sample point of $i$ lies within the sampling radius $\varepsilon$ of a sample point of $j$ — this catches curves that cross in the middle and, crucially, closed curves such as circles, which have no endpoints for test (1) to use;*

*(3)* (optional) concentric link: *$i, j$ are circles or arcs with nearly equal radii and nearly coincident analytic centres (off by default).*

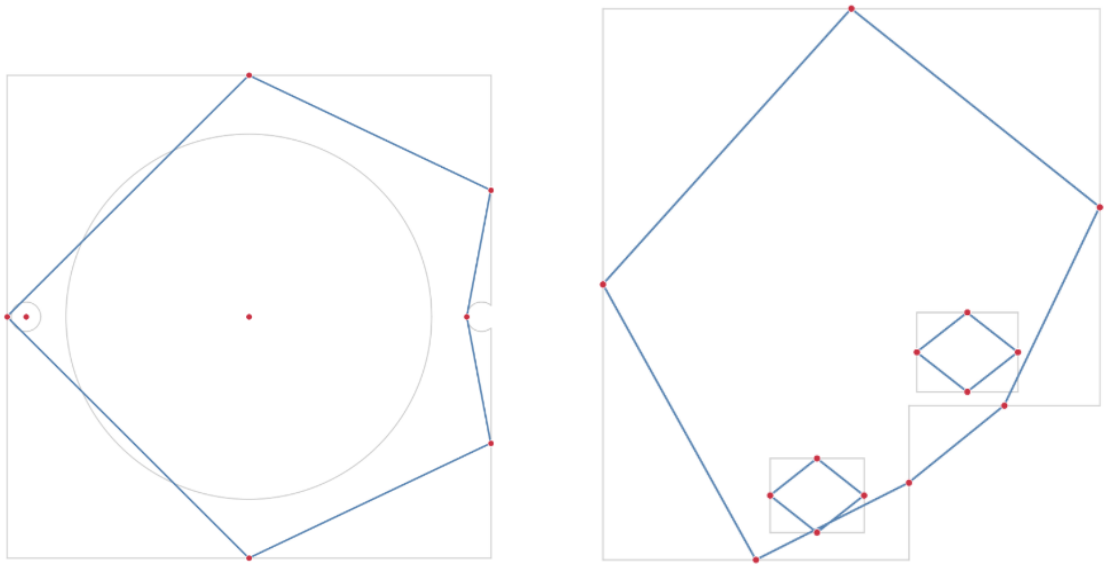
*Every edge has unit weight, i.e. counts as one "hop."*

Checking conditions (1) and (2) naively would compare every sample of every primitive against every other — $O(N^2K^2)$ work, prohibitive now that a single primitive can carry up to $K_{\max} = 400$ samples. Instead all sample points from all primitives are pooled into one cloud, each point tagged with the primitive it belongs to, and a single $k$-d-tree range query [2] finds all point pairs closer than the radius at once, in $O(NK \log NK)$ time. By Proposition 3.1 the sample-touch test at radius $\varepsilon$ cannot miss a true crossing.

**From contacts to hop distances.** On this graph we compute, by breadth-first search, the unweighted shortest-path distance between every pair of primitives — the minimum number of contacts one must cross to get from one to the other — collected in a raw matrix $\widetilde{SP}$. Two primitives in different disconnected components are formally infinitely far apart; rather than store $+\infty$, we replace such entries with a *gentle sentinel* equal to one more than the largest finite distance present: $\widetilde{SP}_{ik} \leftarrow \max\{\widetilde{SP}_{jl} : \widetilde{SP}_{jl} < \infty\} + 1$. Writing $q_{0.95}$ for the 95th percentile of the entries, the matrix is then clipped at that percentile (so a few outliers cannot dominate) and rescaled to $[0, 1]$:

$$SP_{ik} = \frac{\min(\widetilde{SP}_{ik},\, q_{0.95}) - SP_{\min}}{SP_{\max} - SP_{\min}} \in [0, 1], \qquad \boldsymbol{C} = \lambda_{\mathrm{topo}}\, SP, \tag{3.10}$$

where $SP_{\min}, SP_{\max}$ are the smallest and largest values after clipping and $\lambda_{\mathrm{topo}}$ is an overall weight (absorbed automatically by the balancing of Section 3.7, Proposition 3.5). The choice of a "+1" sentinel rather than, say, ten times the maximum is deliberate and interacts with the



(a) Frame with an inscribed circle (b) The two-loop running example

Figure 3.2: **The contact graph.** Each primitive becomes a node placed at its centre (red dot) and an edge (blue) joins two nodes whose primitives physically meet (Definition 3.4); the underlying drawing is shown faintly in grey. In (a) the inscribed circle has no endpoints, so it can only attach to its neighbours through the *sample*-touch test — precisely the case that test exists for. Both drawings are shown rotated to emphasize that the hop distances read off this graph, and hence the structure matrix $\boldsymbol{C}$, are unchanged by rigid motion; only the connectivity matters, not where the drawing sits or how it is oriented.

percentile clip: on a sparse drawing where most pairs are disconnected, a huge sentinel would, after rescaling, crush all the real hop counts toward 0 and wipe out the structure signal; "one hop beyond the farthest real path" keeps the genuine distances spread across most of $[0, 1]$ while still marking disconnection as maximal.

**Why hop count, not Euclidean distance.** We use number-of-hops, not physical distance, for $\boldsymbol{C}$ on purpose. Hop count captures pure *connectivity* — which features are physically adjacent — and is completely unchanged by translation, rotation, reflection, and scale, which is exactly the relational signal the Gromov–Wasserstein term is designed to preserve. The metric (size and position) information is not lost; it is carried separately by the feature term (Section 3.7). Figure 3.2 draws the resulting contact graph on two drawings.

## 3.5 Frame Normalization

The structure term is already blind to rigid motion, but the feature term compares coordinates directly, so any leftover difference in position or size between the two drawings would leak

straight into the feature cost and spoil the match. We therefore put both drawings into a common frame by removing the only two nuisances that may differ between them: overall scale and translation. (Orientation is assumed already aligned, per Section 1.3.)

**Scale.** Each drawing is divided by its own diameter $D = \mathrm{diam}(\{\boldsymbol{c}_i\})$, the largest distance between any two of its centres (computed efficiently from convex-hull vertices [16] when there are many points, since the farthest pair always lies on the hull). After this division every drawing occupies a region of unit diameter, so two copies of the same shape at different sizes become the same size.

**Proposition 3.3 (Scale consistency)** *Dividing every position of a drawing by a scale factor $s$ produces a faithful, merely smaller copy provided each derived quantity is rescaled by the matching power of $s$ dictated by its physical units: positions $\boldsymbol{c}$, lengths $\ell$, and samples $\boldsymbol{S}$ are divided by $s$; curvature $\kappa$ (units of $1/\text{length}$) is multiplied by $s$; and the unit endpoint tangents (pure directions) are unchanged.*

**Proof:** Let $\Phi(\boldsymbol{x}) = \boldsymbol{x}/s$. For a curve $P(t)$ the image is $\tilde{P}(t) = P(t)/s$, so $\tilde{P}'(t) = P'(t)/s$ and arc length $\tilde{\ell} = \int \|\tilde{P}'\|\, dt = \ell/s$; sample points, being values of $\tilde{P}$, scale by $1/s$; the mid-arc point $\tilde{P}(\tilde{\ell}/2) = P(\ell/2)/s$ scales by $1/s$. Curvature is $\kappa = \|P' \times P''\|/\|P'\|^3$; under $\Phi$ both $P', P''$ scale by $1/s$, so the numerator scales by $1/s^2$ and the denominator by $1/s^3$, giving $\tilde{\kappa} = s\,\kappa$. The unit tangent $P'/\|P'\|$ is invariant because numerator and denominator scale identically. These are precisely the stated rules. □

**Translation.** Each drawing is then shifted so that its *length-weighted mean centre*

$$\boldsymbol{\mu} = \frac{\sum_i \ell_i\, \boldsymbol{c}_i}{\sum_i \ell_i} \tag{3.11}$$

sits at the origin. This $\boldsymbol{\mu}$ is the balance point of the drawing when each primitive is weighted by its length — exactly the one-dimensional "mass" the marginals will use (Section 3.6) — so subtracting it lines the two drawings up at a consistent centre without touching their orientation. When the shared content is roughly symmetric, the two drawings' balance points land on corresponding material and the alignment is clean. When a large feature is present in one drawing but deleted from the other, the balance point shifts toward the surviving side and the centring is thrown off slightly — a known stress case we return to in the evaluation (Chapter 4).

## 3.6 The Marginals: Mass = Arc Length

Optimal transport moves *mass*, so each primitive must be assigned a mass.

**Definition 3.5 (Length-weighted marginals)** *A primitive's mass is proportional to its arc length, normalized so that each drawing's masses sum to one:*

$$p_i = \frac{\ell_i}{\sum_{k=1}^{n_1} \ell_k} \in \Delta^{n_1-1}, \qquad q_j = \frac{\ell_j}{\sum_{k=1}^{n_2} \ell_k} \in \Delta^{n_2-1}. \tag{3.12}$$

Length weighting completes the subdivision-invariance story begun by super-node merging.

**Proposition 3.4 (Subdivision invariance of the measure)** *Length weighting is invariant under splitting and merging of primitives: if a primitive of length $\ell$ is replaced by children of lengths $\ell_1, \ldots, \ell_k$ with $\sum_c \ell_c = \ell$, then the total mass carried by that geometric content is unchanged, and the mass of every other primitive is unchanged as well.*

**Proof:** Arc length is additive under subdivision, so the drawing's total length $\sum_k \ell_k$ — the denominator in (3.12) — is unchanged by the split. The children's masses sum to $\sum_c \ell_c / \sum_k \ell_k = \ell / \sum_k \ell_k$, exactly the mass the parent carried; and every other primitive's numerator and the shared denominator are untouched, so its mass is unchanged. □

With *uniform* weights this would fail badly — splitting one line into five equal pieces would quintuple that line's share of the mass and distort the whole transport problem. So Section 3.3 ensures the same shape yields comparable *node sets*, and Proposition 3.4 ensures it yields the same *total mass*; together they make the method indifferent to how each drawing happened to be subdivided.

## 3.7 The Feature Space

This stage builds the feature descriptor $\boldsymbol{f}_i$ for each primitive — the vector the Wasserstein term uses to ask "does primitive $i$ look like primitive $j$?" The guiding principle throughout is *commensurability*: every part of the descriptor must be measured on a comparable numerical scale, so that no single attribute silently dominates the distance merely because of the units it happens to be expressed in.

**Definition 3.6 (The universal descriptor)** *Let $d$ be a common length scale (the common diameter, defined below). Primitive $i$ is encoded as the ten-dimensional vector*

$$\boldsymbol{f}_i = \Big[ \underbrace{\boldsymbol{c}_i}_{\mathbb{R}^3} \;\Big|\; \underbrace{w_\ell\, d\, \tanh\tfrac{\ell_i}{d}}_{\mathbb{R}} \;\Big|\; \underbrace{w_\kappa\, d\, \tanh(\kappa_i d)}_{\mathbb{R}} \;\Big|\; \underbrace{\tfrac{w_{\text{type}} d}{\sqrt{2}}\, \boldsymbol{e}(\tau_i)}_{\mathbb{R}^5} \Big] \in \mathbb{R}^{10}, \tag{3.13}$$

*stacking the 3-D centre, a length feature, a curvature feature, and the 5-D one-hot type vector $\boldsymbol{e}(\tau_i) \in \{0,1\}^5$. The weights default to $w_\ell = 0.6$, $w_\kappa = 0.05$, $w_{\text{type}} = 0.5$.*

The centre block is left as-is, and its entries span a range of about $d$ (after normalization the drawing has unit diameter, so $d \approx 1$). Every other block is engineered so that its contribution is also of order $d$.

- *Length.* Arc length can exceed the diameter — a circle inscribed in the drawing has circumference up to $\pi d$ — so a raw length column could swamp position. The map $x \mapsto d\tanh(x/d)$ squashes length into $[0, d)$ while leaving short primitives almost untouched ($\tanh x \approx x$ for small $x$). The weight $w_\ell = 0.6$ says a full-scale length difference counts for about 0.6 of a diameter of position difference.
- *Curvature.* Curvature has units of 1/length, so it cannot sit next to coordinates directly. Multiplying by $d$ makes $\kappa_i d$ dimensionless, tanh bounds it, and the leading factor $d$ restores length units. The small weight $w_\kappa = 0.05$ reflects that curvature is a comparatively weak, noise-prone cue.
- *Type.* Type is a category, not a quantity with an order, so it is encoded one-hot. The scaling is calibrated by the following lemma.

**Lemma 3.1 (Calibration of the one-hot scale)** *For $\tau \neq \tau'$ the one-hot vectors $\boldsymbol{e}(\tau), \boldsymbol{e}(\tau') \in \{0,1\}^5$ differ in exactly two coordinates, so $\|\boldsymbol{e}(\tau) - \boldsymbol{e}(\tau')\|_2 = \sqrt{2}$. Scaling the block by $s = w_{\text{type}} d/\sqrt{2}$ therefore makes every type mismatch contribute exactly $w_{\text{type}} d$ to the descriptor distance, while identical types contribute 0.*

**Proof:** $\boldsymbol{e}(\tau)$ has a 1 in slot $\tau$ and 0 elsewhere, and $\boldsymbol{e}(\tau')$ a 1 in slot $\tau' \neq \tau$. Their difference has $+1$ in slot $\tau$, $-1$ in slot $\tau'$, and 0 in the remaining three slots, so its Euclidean norm is $\sqrt{1^2 + 1^2} = \sqrt{2}$. Multiplying by $s$ gives $\sqrt{2}\, s = w_{\text{type}} d$. If $\tau = \tau'$ the difference is the zero vector, with norm 0. □ Thus every type mismatch — line versus arc, arc versus circle, all alike — contributes $w_{\text{type}} d$, equivalent to a position error of $w_{\text{type}} = 0.5$ diameters.

**The cross cost, made dimensionless.** The feature cost of matching primitive $i$ of drawing 1 to primitive $j$ of drawing 2 is the distance between their descriptors. Both descriptors are built with the same common diameter $d = \sqrt{d_1 d_2}$, the geometric mean of the two drawings' individual diameters $d_1, d_2$ (each being the quantity $D$ of Section 3.5 measured *after* that section's rescaling, hence $\approx 1$, so $d \approx 1$ as well), and the resulting distance is divided by $d$:

$$M_{ij} = \frac{\|\boldsymbol{f}_i^{(1)} - \boldsymbol{f}_j^{(2)}\|_2}{d} \in \mathbb{R}_{\geq 0}. \tag{3.14}$$

This final division makes $\boldsymbol{M}$ *dimensionless*, so that it speaks the same language as the structure matrix $\boldsymbol{C}$, whose entries already lie in $[0, 1]$. Without it, $\boldsymbol{M}$ would carry raw drawing units and,

at the balanced setting $\alpha = 0.5$, would quietly overwhelm the structure term (on real data the structure term was contributing only about 17% of the objective before this fix). After dividing by $d$, a position error of one full diameter contributes about 1 — the same order as a maximal structural disagreement. We keep $d$ as an explicit symbol rather than writing 1 because it makes the descriptor's units visible and leaves the construction valid when the drawings are not each scaled to unit diameter (then $d \neq 1$ and genuinely rescales the cost).

**Auto-balancing the two terms.** Making both terms dimensionless puts them in the same units, but their typical sizes on a given pair of drawings may still differ. We remove this last imbalance by calibrating at a fixed, neutral reference plan: the *independent coupling* $\boldsymbol{\pi}_0 = \boldsymbol{p}\boldsymbol{q}^\top$, which matches every primitive to every other in proportion to their masses.

**Definition 3.7 (Balance factor)** *With $\circ$ the entrywise (Hadamard) product, define the reference energies*

$$\mathcal{E}_{\text{feat}} = \boldsymbol{p}^\top \boldsymbol{M} \boldsymbol{q}, \tag{3.15}$$

$$\mathcal{E}_{\text{struct}} = \boldsymbol{p}^\top (\boldsymbol{C}^{(1)} \circ \boldsymbol{C}^{(1)}) \boldsymbol{p} + \boldsymbol{q}^\top (\boldsymbol{C}^{(2)} \circ \boldsymbol{C}^{(2)}) \boldsymbol{q} - 2\,(\boldsymbol{p}^\top \boldsymbol{C}^{(1)} \boldsymbol{p})(\boldsymbol{q}^\top \boldsymbol{C}^{(2)} \boldsymbol{q}), \tag{3.16}$$

*and rescale the structure matrices by $s_{\text{bal}} = \sqrt{\mathcal{E}_{\text{feat}} / \mathcal{E}_{\text{struct}}}$, i.e. $\boldsymbol{C}^{(\cdot)} \leftarrow s_{\text{bal}}\, \boldsymbol{C}^{(\cdot)}$.*

**Proposition 3.5 (Exact balance and idempotence)** *Equation (3.16) is the Gromov–Wasserstein square-loss energy (Definition 3.8 below) evaluated at $\boldsymbol{\pi}_0 = \boldsymbol{p}\boldsymbol{q}^\top$. After the rescale of Definition 3.7, the structure energy at $\boldsymbol{\pi}_0$ equals the feature energy at $\boldsymbol{\pi}_0$. The operation is idempotent (re-running it computes $s_{\text{bal}} = 1$), and it absorbs the topology weight $\lambda_{\text{topo}}$ entirely, leaving $\alpha$ as the only trade-off knob.*

**Proof:** At $\boldsymbol{\pi} = \boldsymbol{p}\boldsymbol{q}^\top$ the row and column sums are $\boldsymbol{r} = \boldsymbol{p}$ and $\boldsymbol{c} = \boldsymbol{q}$, so the factorized GW energy of Proposition 3.6 becomes $\boldsymbol{p}^\top(\boldsymbol{C}^{(1)} \circ \boldsymbol{C}^{(1)})\boldsymbol{p} + \boldsymbol{q}^\top(\boldsymbol{C}^{(2)} \circ \boldsymbol{C}^{(2)})\boldsymbol{q} - 2\,\langle \boldsymbol{C}^{(1)},\, \boldsymbol{p}\boldsymbol{q}^\top \boldsymbol{C}^{(2)} \boldsymbol{q}\boldsymbol{p}^\top \rangle$. The cross term simplifies because $\boldsymbol{p}\boldsymbol{q}^\top \boldsymbol{C}^{(2)} \boldsymbol{q}\boldsymbol{p}^\top = (\boldsymbol{q}^\top \boldsymbol{C}^{(2)} \boldsymbol{q})\, \boldsymbol{p}\boldsymbol{p}^\top$, whence $\langle \boldsymbol{C}^{(1)}, (\boldsymbol{q}^\top \boldsymbol{C}^{(2)} \boldsymbol{q}) \boldsymbol{p}\boldsymbol{p}^\top \rangle = (\boldsymbol{p}^\top \boldsymbol{C}^{(1)} \boldsymbol{p})(\boldsymbol{q}^\top \boldsymbol{C}^{(2)} \boldsymbol{q})$, giving exactly (3.16). The square loss is a homogeneous quadratic of degree two in the pair $(\boldsymbol{C}^{(1)}, \boldsymbol{C}^{(2)})$: replacing $\boldsymbol{C} \mapsto s\boldsymbol{C}$ multiplies every term $(C^{(1)}_{ik} - C^{(2)}_{jl})^2$ by $s^2$, so $\mathcal{E}_{\text{struct}}(s\boldsymbol{C}) = s^2 \mathcal{E}_{\text{struct}}(\boldsymbol{C})$. Choosing $s^2 = \mathcal{E}_{\text{feat}} / \mathcal{E}_{\text{struct}}$ makes the rescaled structure energy equal to $\mathcal{E}_{\text{feat}}$. Re-running on the balanced inputs computes $s_{\text{bal}} = \sqrt{\mathcal{E}_{\text{feat}} / \mathcal{E}_{\text{feat}}} = 1$ (idempotence). Finally, since $\boldsymbol{C} = \lambda_{\text{topo}}\, SP$ enters $\mathcal{E}_{\text{struct}}$ with degree two, it appears as $\lambda^2_{\text{topo}}$; thus $s_{\text{bal}} \propto 1/\lambda_{\text{topo}}$ and the product $s_{\text{bal}} \boldsymbol{C}$ is independent of $\lambda_{\text{topo}}$. □

## 3.8 The Matching Engine: Partial FGW

With the feature cost $\boldsymbol{M}$ and the structure matrices $\boldsymbol{C}^{(1)}, \boldsymbol{C}^{(2)}$ in hand, we can write down the quantity the matcher minimizes.

**Definition 3.8 (Fused energy)** *For a coupling $\boldsymbol{\pi} \in \mathbb{R}_{\geq 0}^{n_1 \times n_2}$ and trade-off $\alpha \in [0,1]$,*

$$\mathcal{F}_\alpha(\boldsymbol{\pi}) = (1-\alpha)\sum_{i,j} M_{ij}\pi_{ij} + \alpha \sum_{i,k,j,l} \Big(C^{(1)}_{ik} - C^{(2)}_{jl}\Big)^2 \pi_{ij}\pi_{kl}. \tag{3.17}$$

The two terms penalize the two ways a match can be wrong. The first, feature term is linear in $\boldsymbol{\pi}$: it pays the cost $M_{ij}$ for every unit of mass placed on a pair $(i,j)$, so matching dissimilar-looking primitives is expensive. The second, structure term is quadratic in $\boldsymbol{\pi}$: it pays $(C^{(1)}_{ik} - C^{(2)}_{jl})^2$ whenever a pair $(i,k)$ in drawing 1 is mapped to a pair $(j,l)$ in drawing 2 whose internal hop distance disagrees — enforcing relational consistency using only within-drawing distances, the source of the method's invariance to rigid motion.

**Proposition 3.6 (Square-loss factorization)** *Let $\boldsymbol{r} = \boldsymbol{\pi}\mathbf{1}$ and $\boldsymbol{c} = \boldsymbol{\pi}^\top \mathbf{1}$ be the row and column sums. Then*

$$\sum_{i,k,j,l} \Big(C^{(1)}_{ik} - C^{(2)}_{jl}\Big)^2 \pi_{ij}\pi_{kl} = \boldsymbol{r}^\top \Big(\boldsymbol{C}^{(1)} \circ \boldsymbol{C}^{(1)}\Big)\boldsymbol{r} + \boldsymbol{c}^\top \Big(\boldsymbol{C}^{(2)} \circ \boldsymbol{C}^{(2)}\Big)\boldsymbol{c} - 2\,\langle \boldsymbol{C}^{(1)},\, \boldsymbol{\pi}\,\boldsymbol{C}^{(2)}\boldsymbol{\pi}^\top \rangle, \tag{3.18}$$

*which is computable with a few matrix products at cost $O(n_1^2 n_2 + n_1 n_2^2)$ instead of the naive $O(n_1^2 n_2^2)$ four-index contraction. This factorization of the square-loss Gromov–Wasserstein objective is due to Peyré et al. [14]; we restate and prove it here for completeness.*

**Proof:** Expand $(a-b)^2 = a^2 - 2ab + b^2$ and treat the three sums separately. For the first,

$$\sum_{i,k,j,l} \Big(C^{(1)}_{ik}\Big)^2 \pi_{ij}\pi_{kl} = \sum_{i,k} \Big(C^{(1)}_{ik}\Big)^2 \Big(\sum_j \pi_{ij}\Big)\Big(\sum_l \pi_{kl}\Big) = \sum_{i,k} \Big(C^{(1)}_{ik}\Big)^2 r_i r_k = \boldsymbol{r}^\top \Big(\boldsymbol{C}^{(1)} \circ \boldsymbol{C}^{(1)}\Big)\boldsymbol{r}.$$

The third sum is symmetric, contracting against the column sums to give $\boldsymbol{c}^\top(\boldsymbol{C}^{(2)} \circ \boldsymbol{C}^{(2)})\boldsymbol{c}$. For the cross term, using the symmetry $C^{(2)}_{jl} = C^{(2)}_{lj}$,

$$\sum_{i,k,j,l} C^{(1)}_{ik} C^{(2)}_{jl}\, \pi_{ij}\pi_{kl} = \sum_{i,k} C^{(1)}_{ik} \Big(\sum_{j,l} \pi_{ij} C^{(2)}_{jl} \pi_{kl}\Big) = \sum_{i,k} C^{(1)}_{ik}\, (\boldsymbol{\pi}\boldsymbol{C}^{(2)}\boldsymbol{\pi}^\top)_{ik} = \langle \boldsymbol{C}^{(1)},\, \boldsymbol{\pi}\boldsymbol{C}^{(2)}\boldsymbol{\pi}^\top \rangle.$$

Collecting the three pieces with the factor $-2$ on the cross term gives (3.18). The dominant cost is forming $\boldsymbol{\pi}\boldsymbol{C}^{(2)}\boldsymbol{\pi}^\top$, two dense products of cost $O(n_1 n_2^2)$ and $O(n_1^2 n_2)$. □

**Balanced versus partial constraints.** The remaining question is which couplings $\boldsymbol{\pi}$ are allowed. Standard (balanced) transport requires $\boldsymbol{\pi}\mathbf{1} = \boldsymbol{p}$ and $\boldsymbol{\pi}^\top\mathbf{1} = \boldsymbol{q}$ (the balanced polytope $\Pi(\boldsymbol{p}, \boldsymbol{q})$): every primitive ships out all of its mass and receives its full share. That is exactly wrong under partial overlap: a deleted feature, having no true counterpart, would still be forced to send its mass somewhere, manufacturing a false match. The *partial* polytope instead relaxes the equalities to inequalities and pins only the total transported mass:

$$\Pi^m(\boldsymbol{p}, \boldsymbol{q}) = \{\boldsymbol{\pi} \geq 0 : \ \boldsymbol{\pi}\mathbf{1} \leq \boldsymbol{p}, \ \boldsymbol{\pi}^\top\mathbf{1} \leq \boldsymbol{q}, \ \mathbf{1}^\top\boldsymbol{\pi}\mathbf{1} = m\}. \tag{3.19}$$

Now each primitive may ship at most its mass and keep the rest at home, while only a fraction $m \leq 1$ of the total mass moves at all. The matcher solves, at a given $m$,

$$\boldsymbol{\pi}^\star(m) \in \operatorname*{argmin}_{\boldsymbol{\pi} \in \Pi^m(\boldsymbol{p}, \boldsymbol{q})} \mathcal{F}_\alpha(\boldsymbol{\pi}), \qquad L(m) \coloneqq \mathcal{F}_\alpha\Big(\boldsymbol{\pi}^\star(m)\Big), \tag{3.20}$$

where $L(m)$ is the best achievable cost at that mass — a quantity we study as a function of $m$ in the next section. Given a budget $m$, the optimizer chooses *which* mass to ship, spending it on the best matches first while abandoning the worst-matching primitives; their mass simply stays home. Setting $m = 1$ recovers ordinary FGW.

**Solver.** Problem (3.20) is non-convex; it is solved by a Frank–Wolfe (conditional-gradient) scheme [7] that reaches only a locally optimal plan, so the descending mass sweep of Section 3.9 is run with chained warm starts to keep successive solves in the same basin.

## 3.9 Automatic Selection of the Transported Mass

The fraction $m$ is not a free tuning parameter — it *is* the quantity we most want to recover: the fraction of the two drawings that genuinely overlaps. We therefore read it off the data instead of fixing it by hand. The cue is the best matching cost $L(m)$ from (3.20), watched as we raise the amount of mass $m$ the matcher is required to transport. While $m$ stays below the true overlap, every extra unit of mass still finds a genuine counterpart, so the cost grows only gently. The moment $m$ exceeds the overlap, the matcher runs out of good matches and is forced to ship deleted or cluttered content onto wrong targets, so the cost climbs steeply. The true overlap therefore reveals itself as an "elbow" in the curve, and we keep the largest $m$ just before that first sharp rise — the selected overlap $m^\star$ (Figure 3.3).

In practice we evaluate $L(m)$ at 33 masses spread evenly from 0.20 to 1.00, sweeping from large $m$ down to small and warm-starting each solve from the previous one. Because the underlying problem is non-convex, this warm starting is what keeps the curve well behaved:

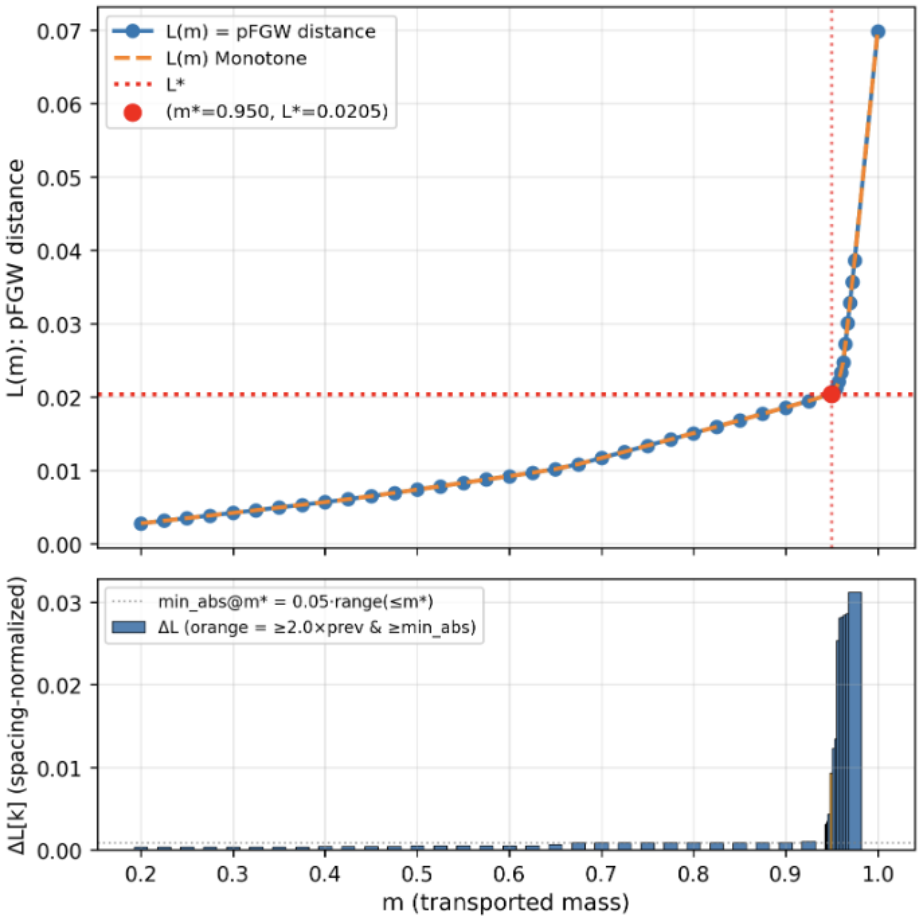



Figure 3.3: **Automatic mass selection.** (Top) The matching cost $L(m)$ against the transported fraction $m$; the marked point is the selected mass $m^\star$, taken just before the curve's first sharp rise. (Bottom) The step-to-step increase of the cost; the highlighted bar is the detected jump.

neighbouring solves stay in the same basin instead of settling into unrelated local optima, so the curve we analyse is smooth and trustworthy rather than jagged with spurious elbows. A light smoothing removes any small residual dips — the true cost can only rise as more mass is shipped — and the elbow is taken as the first place where the curve steepens sharply, roughly where a step is at least twice as steep as anything before it. The grid is then refined around that elbow and the search repeated. If no such rise ever occurs — for instance when the two drawings are identical and the cost stays near zero — the method simply returns $m^\star \to 1$ and ships everything, exactly as it should when the drawings fully overlap.

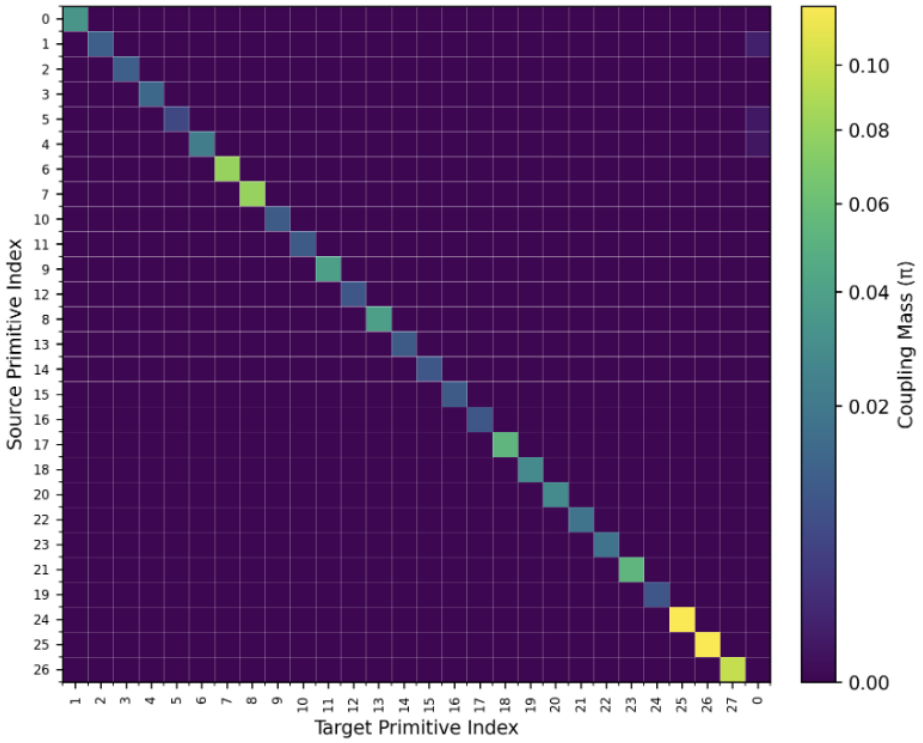



Figure 3.4: **Coupling matrix** for a drawing containing lines, arcs, and a circle. High-intensity entries are strong correspondences; the transport plan is highly sparse, with mass concentrated on a near-diagonal set of credible pairs.

## 3.10 Correspondence Extraction

The solver returns a transport plan $\boldsymbol{\pi}$ defined on *super-nodes* (the merged chains of Section 3.3). Three steps turn this into a human-readable answer: read off the coupling, unpack it back to the original (atomic) primitives, and decide which primitives have no match at all.

**Coupling structure.** Figure 3.4 shows a representative coupling matrix, with source primitives as rows and target primitives as columns and each cell shaded by the mass $\pi_{ij}$. A bright cell means "$i$ corresponds to $j$"; after reordering, the bright cells line up near the diagonal, showing that the shared layout is matched consistently, while an entirely dark row or column flags a primitive that went unmatched. The plan is highly *sparse* — most cells are zero — because the optimizer concentrates the budget on a few credible pairs.

**Unpacking super-nodes.** Because matching was done on super-nodes, a single entry $\pi_{IJ}$ may represent a whole merged chain $I$ matched to a whole chain $J$.

**Definition 3.9 (Length-proportional unpacking)** *With member length shares* $\sigma_I(a) = \ell_a / \sum_{a' \in I} \ell_{a'}$

*and* $\sigma_J(b) = \ell_b / \sum_{b' \in J} \ell_{b'}$*, the atomic coupling is the outer product of shares,*

$$\pi^{\text{atom}}_{ab} = \sum_{I,J} \pi_{IJ}\, \sigma_I(a)\, \sigma_J(b), \tag{3.21}$$

*with terms vanishing unless* $a \in I$ *and* $b \in J$*.*

**Proposition 3.7 (Marginal consistency of unpacking)** *Unpacking preserves the transported mass exactly,* $\mathbf{1}^\top \boldsymbol{\pi}^{\text{atom}} \mathbf{1} = \mathbf{1}^\top \boldsymbol{\pi} \mathbf{1}$*, and respects the unpacked marginals: if the super-node source marginal is unpacked by the same shares,* $p^{\text{atom}}_a = \sigma_I(a)\, p_I$ *for* $a \in I$*, then the atomic row sums obey* $\sum_b \pi^{\text{atom}}_{ab} \le p^{\text{atom}}_a$ *(and symmetrically for columns). When every super-node is a singleton, (3.21) is the identity.*

**Proof:** For each chain, $\sum_{a \in I} \sigma_I(a) = 1$. Total mass: $\mathbf{1}^\top \boldsymbol{\pi}^{\text{atom}} \mathbf{1} = \sum_{I,J} \pi_{IJ} \Big( \sum_{a \in I} \sigma_I(a) \Big) \Big( \sum_{b \in J} \sigma_J(b) \Big) = \sum_{I,J} \pi_{IJ} = \mathbf{1}^\top \boldsymbol{\pi} \mathbf{1}$. For a fixed $a \in I$,

$$\begin{aligned} \sum_b \pi^{\text{atom}}_{ab} &= \sum_J \sum_{b \in J} \pi_{IJ} \sigma_I(a) \sigma_J(b) = \sigma_I(a) \sum_J \pi_{IJ} \Big( \sum_{b \in J} \sigma_J(b) \Big) \\ &= \sigma_I(a) \sum_J \pi_{IJ} = \sigma_I(a)\, (\boldsymbol{\pi}\mathbf{1})_I \le \sigma_I(a)\, p_I = p^{\text{atom}}_a, \end{aligned}$$

using the super-node feasibility $(\boldsymbol{\pi}\mathbf{1})_I \le p_I$; columns are symmetric. If every super-node is a singleton then $\sigma_I(a) = \sigma_J(b) = 1$ and $\pi^{\text{atom}}_{ab} = \pi_{ab}$. □ Splitting by length share is the only choice consistent with the length-weighted masses of Section 3.6: it preserves the marginals exactly, and when every super-node is a single primitive it changes nothing. This is precisely the step where *one-to-many* correspondences appear — a single parent edge in one drawing distributes its mass across the several children it was subdivided into in the other.

**Declaring "no match".** A target primitive $j$ is declared unmatched when the most mass any source sends it is small compared with *its own* mass $q_j$, specifically $\max_i \pi_{ij} \le \frac{1}{2} q_j$. The comparison is made against each primitive's own marginal rather than against one global threshold, and this is important under length weighting: a short primitive carries little mass in total, so even its largest legitimate coupling entry is far below the biggest entry in the whole matrix — a global cutoff would wrongly brand every short primitive unmatched. The per-column rule is scale-free and works under any weighting mode.

**Final visualization.** For interpretability the correspondence is drawn directly on the two drawings: the Source and Target are shown side by side, each Target primitive is annotated with (or coloured by) the Source primitive it matches, and Target primitives that received no

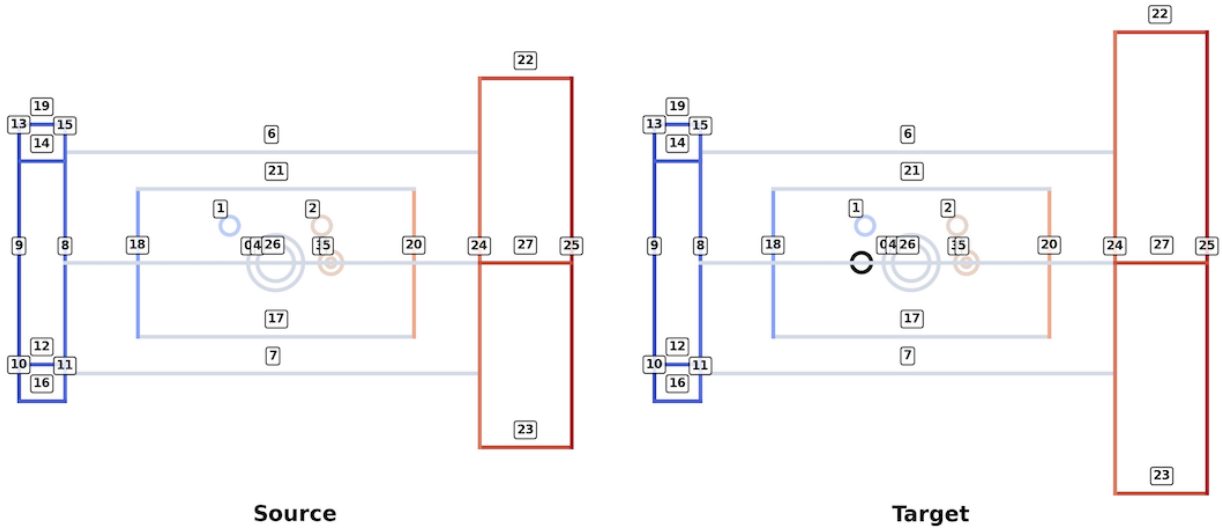



Figure 3.5: **Final correspondence on the running example.** Source (left) and Target (right); each Target primitive is annotated with the index of the Source primitive it matches, and primitives with no dominant match are drawn in black.

accepted mass are drawn in black, marking them as clutter or deletions. Figure 3.5 works an example with lines, arcs, and a circle: the consistent transfer across heterogeneous primitive types confirms a correct correspondence.

## 3.11 The Complete Procedure and Complexity

Having assembled every stage of this chapter into a single procedure, we close with its computational cost.

Let $N = \max(N_s, N_t)$ be the number of primitives after merging, $I$ the number of solver iterations per solve, and $K$ the number of masses on the grid.

**Time:** $O(K \cdot I \cdot N^3)$. The $N^3$ comes from the matrix products that evaluate the structure term (3.18) once per iteration, repeated over $I$ iterations and $K$ grid points; building the geometry and contact graph adds a lower-order $O(NK \log NK)$ from the $k$-d-tree queries.

**Space:** $O(N^2)$, dominated by the dense structure matrices $\boldsymbol{C}^{(1)}, \boldsymbol{C}^{(2)}$, the cross cost $\boldsymbol{M}$, and the coupling $\boldsymbol{\pi}$.

In one sentence: each drawing becomes a discrete measured space of subdivision-invariant supernodes carrying a ten-dimensional descriptor, a length-proportional mass, and contact-graph hop distances; matching is the partial FGW problem with diameter-normalized, auto-balanced

costs; the transported fraction $m$ — the unknown overlap — is found by sweeping $L(m)$ and stopping just before its first statistically drastic rise; and the resulting plan is unpacked to the original primitives and thresholded per column to give the final correspondence.

# Chapter 4

# Evaluation

How does one score a CAD matcher? Hand-labelling the "correct" correspondence between two drawings is laborious and often genuinely ambiguous — especially at split junctions and across deleted features — which makes a hand-annotated test set both expensive and unreliable. This chapter sidesteps that difficulty with a *controlled framework* whose correct answer is known by construction, scores the matcher objectively on six design-edit families, and analyses qualitatively where the method succeeds and where it fails.

## 4.1 An Evaluation Framework with Ground Truth by Construction

We evaluate on a controlled benchmark in which the correct answer is known *by construction*: every Target drawing is produced from a Source by a parametric design edit that records exactly what happened to each primitive — kept, moved, enlarged, removed, or added — so the matcher can be scored objectively rather than judged by eye.

What we contribute here is a *framework*: a reusable, source-agnostic pipeline (generator + scorer + protocol) that turns one drawing into correspondence problems with exact ground truth and emits per-edit metrics for any matcher. We do not yet claim a large *benchmark dataset* — that is obtained by running such a framework at scale (many Sources, randomized parameters, graded difficulty) and freezing the result, which is the immediate next step (Chapter 5). The numbers below therefore characterize per-edit-family behaviour on one representative part. The framework comprises a deterministic edit engine, a renderer, a vision–language decision layer, a matcher adapter, and an objective scorer.

## 4.2 Benchmark Construction

**Source and edit families.** We exercise the framework on a representative mechanical drawing (`Source.json`): an L-bracket of 16 line segments with two interior loops — an upper stepped feature and a lower rectangular pocket. From this one Source we synthesize six Target categories, each chosen to stress one of the robustness requirements of Section 1.2:

- ADD — insert a new loop in free space (tests partial overlap, insertion side);
- LARGER — scale a loop about its centre (tests scale robustness);
- MERGE — translate a loop onto the nearest boundary edge, forming a notch (tests subdivision: the loop fuses with the boundary);
- MOVE — translate a loop to a new interior location (tests structural re-arrangement);
- REMOVE — delete a loop (tests partial overlap, deletion side);
- EVERYTHING — a composition of several of the above at once.

The six Source→Target pairs are shown in Figure 4.2, drawn in the matcher's own convention — plain black strokes with each primitive numbered.

**Hybrid generation: the LLM decides, the engine executes.** Generating these targets poses a dilemma. Our deterministic engine operates *at the level of primitives* — it adds, moves, scales, and merges lines, arcs, and loops directly, and carries no design semantics: it does not represent the part as a feature tree with sketch constraints and driven dimensions, so it cannot itself judge which loop a human would plausibly modify. A purely procedural generator is therefore valid but design-agnostic. (This is a property of *our* primitive-level engine, not of procedural generation in principle: a generator built on a feature/parametric model could be made design-aware and choose edits itself, an extension we return to in Chapter 5.) A purely LLM-driven generator, on the other hand, reasons about plausible edits but, in our experiments, botches the exact coordinates and emits invalid or truncated geometry. (Concretely, both InternVL3.5-14B and a larger Qwen2.5-VL-32B model, even when shown before/after images, reasoned about the edit correctly yet executed the geometry wrongly — one merely deleted a line instead of moving the loop.) We therefore use a *hybrid* generator that separates the two concerns. A vision–language model (InternVL3.5-14B), shown the rendered drawing with its loops labelled, emits only a compact *edit specification* — the design judgment alone: which loop to act on, in which direction, by what scale. A deterministic engine then executes that specification exactly, which guarantees a valid Target and, as a by-product, emits the *ground-truth correspondence*: the identities of every kept, moved, removed, and added primitive, with

one-to-many links recorded wherever an edit splits a primitive. If the model names a feature the engine cannot realize, the engine falls back to a valid default, so every Target is well-formed. In short, the LLM contributes realistic edit *selection* while every precision-critical number stays exact.

**What the model is shown — and what it is *not* trained on.** A point worth making explicit: the vision–language model is *not* fine-tuned or trained on any CAD data. We use the pretrained InternVL3.5-14B with its weights frozen and supply all task knowledge *in context*, through a small set of worked examples (few-shot prompting). The prompt holds exactly six labelled before/after exemplars — one per edit family (ADD, LARGER, MERGE, MOVE, REMOVE, and the composite EVERYTHING) — all built on a single simple Source, a circle inscribed in a square frame with two small attached loops (Figure 4.1). For each exemplar the model sees the rendered Source and Target images together with their JSON and the edit name, so that it learns the *format and intent* of an edit specification rather than any particular geometry; at inference it is shown a held-out test drawing with its loops labelled and asked only to name an edit and its parameters, never to emit coordinates. This training-free, few-shot use is deliberate: it keeps the framework source-agnostic and removes any dependence on a labelled CAD training corpus — a corpus that, were the model trained on it, would also be the obvious way to grow the benchmark itself (Chapter 5).

## 4.3 Protocol and Metrics

**Parameter settings.** Every run uses the single default configuration of Chapter 3, held fixed across all six edits with no per-instance tuning. The feature/structure trade-off is $\alpha = 0.5$, and the descriptor weights are $w_\ell = 0.6$ (length), $w_\kappa = 0.05$ (curvature), and $w_{\text{type}} = 0.5$ (type). Each primitive is sampled with budget $K_{\min} = 8$, $K_{\max} = 400$ at the data-driven spacing $\varepsilon = \max(\ell_{\max}/K_{\max},\, 10^{-4}D)$, which doubles as the sample-touch radius (Proposition 3.1); the endpoint weld tolerance is $\varepsilon_{\text{w}} = 10^{-6}D$ and the $G^1$ merge tolerance is $\theta_{\text{tol}} = 5°$. The structure matrix uses unweighted shortest-path (hop) distances, clipped at the 95th percentile and min–max scaled to $[0, 1]$; its weight $\lambda_{\text{topo}}$ is *not* set by hand, as the auto-balancing of Section 3.7 (Proposition 3.5) fixes the feature/structure scale at the independent coupling, and the common diameter $d = \sqrt{d_1 d_2}$ renders the cross cost $\boldsymbol{M}$ dimensionless. The transported mass $m$ is selected automatically by sweeping the grid $\{0.20, 0.225, \ldots, 1.00\}$ (33 points, $\Delta m = 0.025$, in descending order with warm starts) and stopping just before the cost curve's first sharp rise (Section 3.9), with the grid refined twice around that elbow. Finally, a target primitive $j$ is declared unmatched when its largest incoming mass falls below $\frac{1}{2}q_j$.

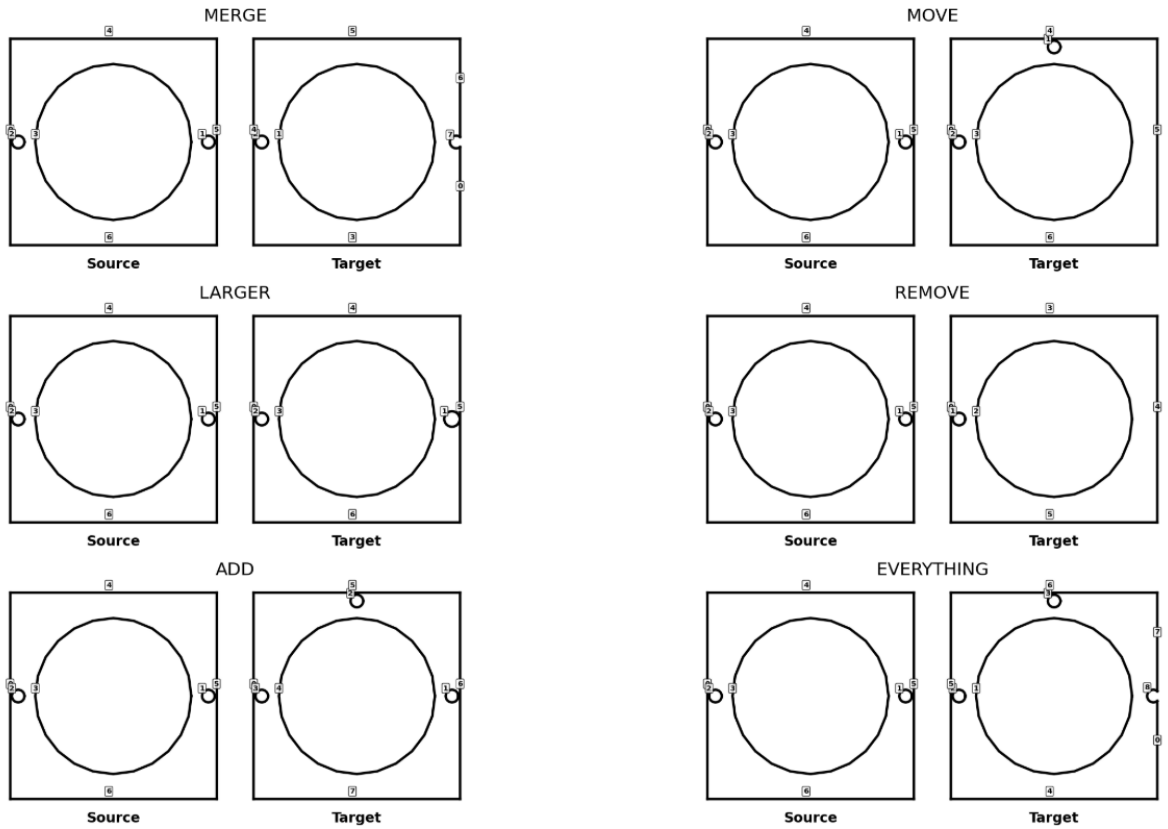



Figure 4.1: **The six few-shot exemplars shown to the vision–language model.** These six labelled before/after pairs are the *entire* supervision the model receives — one per edit family, all built on one simple Source (a circle inscribed in a square frame with two small loops); Source on the left and Target on the right of each pair. They are provided *in context* (the model's weights are never updated); from them the model learns the format and intent of an edit specification, which it then applies to a new, more complex test drawing (Figure 4.2).

**Metrics.** For each Source–Target pair the matcher is run exactly as in Section 3.11, returning an atomic coupling $\boldsymbol{\pi}^{\text{atom}}$ and the auto-selected overlap $m^{\star}$, which are then compared against the recorded ground truth. Because an edit may split one primitive into several, each Source primitive has a *partner set* — the one or more Target primitives that are its legitimate matches — and a match is counted correct if it lands anywhere in that set. We report six numbers:

- **top-1 accuracy** — of the matched Source primitives, the fraction whose single highest-mass Target is a true partner (did the best guess land correctly?);
- **top-3 recall** — the same, but allowing any of the three highest-mass Targets to be a true partner (a more forgiving variant);
- **mass-on-correct** — of all the mass that was transported, the share that landed on true

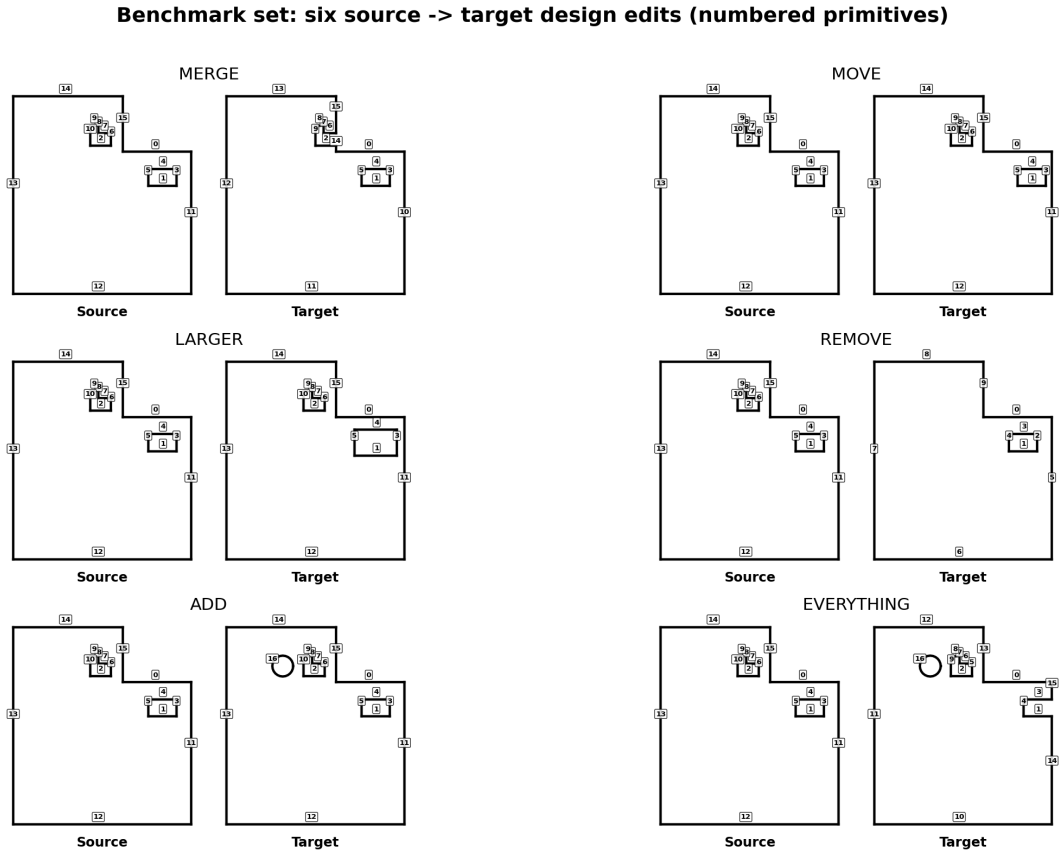



Figure 4.2: **The benchmark set (data).** From one Source, the vision–language model selects an edit and a deterministic engine builds the geometry, recording the exact correspondence. Each panel shows a Source→Target pair (the six edit families: insertion, scaling, merging, translation, deletion, and their composition) in the matcher's own drawing convention — plain black strokes, every primitive numbered. Figure 4.3 shows the matcher's correspondence on these same six pairs in the *identical* panel layout.

partner pairs (how much of the plan is right, by weight);

- **removed-leak** — the share of mass wrongly grabbed by Source primitives that were actually *deleted* (ideal 0);
- **added-leak** — the share of mass wrongly placed on Target primitives that were actually *inserted* (ideal 0);
- $m$**-error** $= |m^\star - f_{\text{kept}}|$ — how far the auto-selected overlap is from the true surviving fraction $f_{\text{kept}}$ (did mass selection estimate the overlap correctly?).

The first three reward correct correspondence (higher is better); the two leaks and $m$-error measure the two ways partial overlap can go wrong (lower is better). Top-1 accuracy and

Table 4.1: **Per-edit correspondence scores** (pFGW matcher, single default configuration). top-3 = top-3 recall; $\text{mass}_c$ = mass-on-correct; leaks and $m$-err: lower is better; all others: higher is better.

| Edit | top-1 | top-3 | $\text{mass}_c$ | rem-leak | add-leak | $m$-err |
|---|---|---|---|---|---|---|
| LARGER | 1.00 | 1.00 | 1.00 | 0.00 | 0.00 | 0.00 |
| ADD | 1.00 | 1.00 | 0.94 | 0.00 | 0.03 | 0.03 |
| MERGE | 1.00 | 1.00 | 0.94 | 0.06 | 0.00 | 0.06 |
| MOVE | 0.81 | 0.88 | 1.00 | 0.00 | 0.00 | 0.20 |
| EVERYTHING | 0.80 | 0.87 | 0.82 | 0.05 | 0.07 | 0.11 |
| REMOVE | 0.50 | 0.70 | 0.31 | 0.38 | 0.00 | 0.38 |
| **Mean** | **0.85** | **0.91** | **0.83** | **0.08** | **0.02** | **0.13** |

top-$k$ recall are the standard accuracy/recall measures used to score feature correspondence and shape retrieval [18, 17], here adapted to the partner sets defined above; the remaining four — mass-on-correct, the two leaks, and $m$-error — are transport-mass-based measures we introduce specifically to exploit the exact ground truth, scoring *how much* of the transported mass lands correctly (and how much leaks onto deleted or inserted content) rather than only whether the single best guess is right. All scoring is objective, since the generator records the exact correspondence.

## 4.4 Quantitative Results

Table 4.1 reports per-edit and mean scores for the pFGW matcher on the six pairs. Averaged over the six edits it reaches top-1 accuracy 0.85, top-3 recall 0.91, and mass-on-correct 0.83, with both leaks near zero. The geometric edits are essentially solved: LARGER is a perfect 1.00, and ADD and MERGE reach top-1 1.00 while withholding nearly all mass from inserted or split content (ADD's added-leak is just 0.03; MERGE's only blemish is a 0.06 removed-leak at the boundary split). MOVE places every transported unit of mass on a correct pair (mass-on-correct 1.00); its lower top-1 of 0.81 is purely a symptom of *under-shipping* the overlap ($m^\star = 0.80$ vs. a true 1.0, hence $m$-error 0.20), since the un-shipped fifth of the mass leaves a few low-mass primitives with no transported match — which top-1, scored over every primitive equally, records as misses even though nothing was matched *wrongly*. The two harder cases are the composite EVERYTHING (top-1 0.80) and, by a wide margin, REMOVE, dissected in Section 4.5.

## 4.5 Qualitative Analysis

We visualise each correspondence in the matcher's own readout convention (Section 3.10). Each Source primitive takes a colour from a left-to-right position gradient, and that colour is *transferred* to whatever Target primitive the coupling matches it to; a Target winning no dominant mass (largest incoming mass below $\frac{1}{2}q_j$) is drawn black. A *correct* match therefore reproduces the Source's gradient on the Target — same colours, same order — so quality is read directly from how faithfully the pattern carries across. Figure 4.3 shows all six edits in the *same* layout as the benchmark set (Figure 4.2), for side-by-side comparison with the data.

For the geometric and additive edits the transfer is essentially exact. In LARGER, despite a 1.5× rescaling of the pocket, diameter normalization (Section 3.5) removes the size change and the structure term anchors each enlarged edge to its original, so the gradient carries across unbroken. MOVE, MERGE, and ADD look much the same — the surviving structure keeps its colours, while inserted or relocated content is absorbed by the partial relaxation, which simply declines to ship mass to it (those Target primitives stay black).

**The deletion failure mode.** REMOVE is the method's one clear weakness (top-1 0.50, removed-leak 0.38, $m$-error 0.38), and its panel in Figure 4.3 shows why: when a whole loop is deleted, its Source primitives have no counterpart in the Target yet are still force-matched onto surviving edges, so the deleted loop's colours reappear on the wrong Target primitives. The cause is *upstream* of the transport solve, in automatic mass selection (Section 3.9): deleting one compact loop removes only a slice of the total mass, spread smoothly across the marginals, so the cost curve $L(m)$ rises gradually instead of with the sharp cliff the jump detector seeks. The detector never fires, $m^\star$ defaults to 1, and at $m = 1$ the partial problem collapses to the balanced one, *forcing* the orphaned primitives to ship — crashing mass-on-correct to 0.31. The predicted overlap ($m^\star = 1.0$ against a true 0.625) confirms this, exactly the asymmetric-deletion case anticipated in Section 3.5. The composite EVERYTHING (top-1 0.80) inherits a milder version, plus a little leakage onto its added loop (added-leak 0.07).

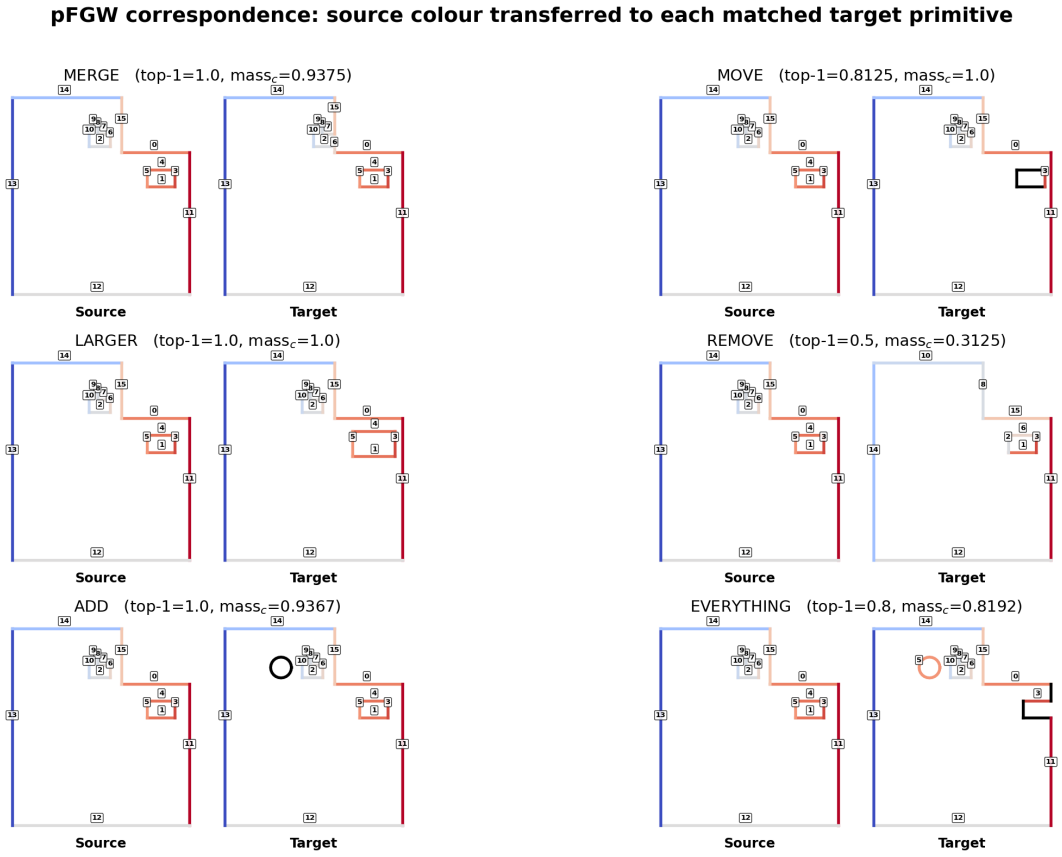



Figure 4.3: **Correspondence results (colour transfer).** The six pFGW matchings, in the same panel layout as the benchmark set of Figure 4.2. Each Source primitive is coloured by a left-to-right position gradient; that colour is transferred to its matched Target primitive, and Target primitives that win no dominant mass are drawn black. A correct match reproduces the Source's colour pattern on the Target: the geometric and additive edits carry the gradient across intact, while REMOVE and EVERYTHING show the residual mis-colourings discussed in the text. Per-panel top-1 and mass-on-correct are annotated.

# Chapter 5

# Conclusion and Future Work

This closing chapter looks ahead to the most consequential extension of the work: making the evaluation framework's generator *design-aware* — built on a feature/parametric model rather than a learned decision layer — so that it emits always-valid pairs with exact ground truth and scales into a large, public benchmark.

## 5.1 Summary

This thesis set out to match the primitives of two CAD drawings that describe the same shape yet disagree at the level of primitives — sitting in different frames, overlapping only partially, and subdividing the same geometry differently. We addressed all three at once with a single, training-free pipeline: each drawing is lifted to a measured metric–feature space of subdivision-invariant super-nodes — a ten-dimensional per-primitive descriptor, length-proportional masses, and a contact-graph structure — and the correspondence is solved as a partial Fused Gromov–Wasserstein problem with diameter-normalized, auto-balanced costs, whose transported fraction (the unknown overlap) is read off the transport-cost curve automatically and unpacked into interpretable one-to-one and one-to-many matches. To score the matcher without hand-labelling, we further contributed an evaluation framework whose ground truth is exact by construction, in which a vision–language model selects realistic edits and a deterministic engine executes them exactly.

On a representative mechanical part the matcher reaches a mean top-1 accuracy of 0.85 (top-3 recall 0.91, mass-on-correct 0.83) across six edit families, with both leaks near zero; the geometric and additive edits (LARGER, ADD, MERGE, and MOVE) are essentially solved. Its one clear weakness is whole-loop *deletion*: automatic mass selection fails to detect the gentle cost-curve elbow left by a compact removal, defaults to full transport, and force-matches

the orphaned primitives (Section 4.5) — the failure mode that most directly motivates the extensions below.

## 5.2 Future Work

The evaluation framework's one remaining dependence on a learned model is its *decision layer*. Because the deterministic engine edits primitives directly and carries no design semantics, it cannot itself judge which edit a designer would plausibly make, so that judgment is outsourced to the vision–language model (Section 4.2). The natural way to remove this dependence — and with it the last source of model-induced bias in the generated pairs — is to lift the generator from the level of primitives to the level of *features*.

A feature/parametric generator would represent each part as a feature tree with sketch constraints (concentric, tangent, symmetric) and driven dimensions — exactly the representation CAD kernels already maintain. On that representation the design judgment becomes intrinsic rather than outsourced: design rules turn first-class (standard hole sizes, minimum wall thickness and clearance, retained symmetry, basic manufacturability), and a *feature grammar* — bosses, ribs, fillets, chamfers, slots, and hole patterns — sampled over feasible ranges replaces the present six hand-written edit families with a far richer vocabulary. Crucially, such a generator would be *procedural and exact*: it yields always-valid targets and exact ground-truth correspondence with no model in the loop at all.

This is also the most direct route from our *framework* to a *dataset*. Driving a design-aware generator over many seed parts, randomized feature parameters, and graded difficulty tiers, then freezing the result, would turn the present per-edit-family characterization into a large, fixed, curated, topology-aware benchmark for apples-to-apples comparison across methods — the scale the framework was built to reach (Chapter 4).

# Bibliography


[1] Milton Abramowitz and Irene A. Stegun. *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*, volume 55 of *Applied Mathematics Series*. National Bureau of Standards / Dover, 1964. 19

[2] Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975. 22

[3] Laetitia Chapel, Mokhtar Z. Alaya, and Gilles Gasso. Partial optimal transport with applications on positive-unlabeled learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pages 2903–2913, 2020. 6

[4] Lénaïc Chizat, Gabriel Peyré, Bernhard Schmitzer, and François-Xavier Vialard. Scaling algorithms for unbalanced optimal transport problems. *Mathematics of Computation*, 87 (314):2563–2609, 2018. doi: 10.1090/mcom/3303. 12

[5] Samir Chowdhury and Facundo Mémoli. The gromov–wasserstein distance between networks and stable network invariants. *Information and Inference: A Journal of the IMA*, 8 (4):757–787, 2019. doi: 10.1093/imaiai/iaz026. 9

[6] Rémi Flamary, Nicolas Courty, et al. Pot: Python optimal transport. *Journal of Machine Learning Research (JMLR)*, 22(78):1–8, 2021. URL http://jmlr.org/papers/v22/20-451.html. 6

[7] Marguerite Frank and Philip Wolfe. An algorithm for quadratic programming. *Naval Research Logistics Quarterly*, 3(1–2):95–110, 1956. 29

[8] Weilin Lai, Tie Xu, Ben Niu, and Hu Wang. Graphbrep: Explicit graph diffusion of b-rep topology for efficient cad generation. *Journal of Computational Design and Engineering*, 13(1):259–274, 2026. doi: 10.1093/jcde/qwaf136. 6

[9] Jean-Christophe Léger. Menger curvature and rectifiability. *Annals of Mathematics*, 149 (3):831–869, 1999. 17

[10] Mingzhe Li, Xinyuan Yan, Lin Yan, Tom Needham, and Bei Wang. Flexible and probabilistic topology tracking with partial optimal transport. *IEEE Transactions on Visualization and Computer Graphics*, 31(10):7951–7969, 2025. doi: 10.1109/TVCG.2025.3561300. 6, 12

[11] Facundo Mémoli. Gromov–wasserstein distances and the metric approach to object matching. *Foundations of Computational Mathematics*, 11(4):417–487, 2011. 6

[12] Onshape Inc. Onshape — the full cloud cad platform. https://www.onshape.com/, 2020. 6

[13] Gabriel Peyré and Marco Cuturi. Computational optimal transport. *Foundations and Trends in Machine Learning*, 11(5–6):355–607, 2019. doi: 10.1561/2200000073. 10

[14] Gabriel Peyré, Marco Cuturi, and Justin Solomon. Gromov–wasserstein averaging of kernel and distance matrices. In *Proceedings of the 33rd International Conference on Machine Learning (ICML)*, volume 48 of *Proceedings of Machine Learning Research*, pages 2664–2672, 2016. 28

[15] Les Piegl and Wayne Tiller. *The NURBS Book*. Springer-Verlag, 2nd edition, 1997. 19

[16] Franco P. Preparata and Michael Ian Shamos. *Computational Geometry: An Introduction*. Springer-Verlag, 1985. 24

[17] Yuhan Quan, Huan Zhao, Jinfeng Yi, and Yuqiang Chen. Self-supervised graph neural network for mechanical cad retrieval. *arXiv preprint arXiv:2406.08863*, 2024. URL https://arxiv.org/abs/2406.08863. 6, 40

[18] Paul-Edouard Sarlin, Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. SuperGlue: Learning feature matching with graph neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4938–4947, 2020. 40

[19] Thomas B. Sebastian, Philip N. Klein, and Benjamin B. Kimia. Recognition of shapes by editing their shock graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 26(5):550–571, 2004. 5

[20] Thibault Séjourné, François-Xavier Vialard, and Gabriel Peyré. The unbalanced Gromov Wasserstein distance: Conquering the quadratic divergence. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 34, pages 8766–8779, 2021. 12

[21] Robert Endre Tarjan. Efficiency of a good but not linear set union algorithm. *Journal of the ACM*, 22(2):215–225, 1975. 21

[22] Titouan Vayer, Laetitia Chapel, Rémi Flamary, Romain Tavenard, and Nicolas Courty. Optimal transport for structured data with application on graphs. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, pages 6275–6284, 2019. 6, 11

[23] Cédric Villani. *Optimal Transport: Old and New*, volume 338 of *Grundlehren der mathematischen Wissenschaften*. Springer, 2009. 10