GrandOMR: Interactive Orchestral Score Recognition and Playback

Zihan Guo 2024011317
Chengwei Liang 2024010681

Computer Vision, Spring 2026

Abstract

We present **GrandOMR**, a hybrid pipeline for Optical Music Recognition (OMR) of scanned printed orchestral scores. Orchestral score OMR is not a single homogeneous problem: staff and system boundary detection, instrument name reading, per-staff note recognition, cross-part consistency repair, and dynamics detection each have distinct input distributions, output spaces, and failure modes. GrandOMR decomposes the task accordingly and matches each subproblem to the most appropriate computer vision method: semantic segmentation (SegNet) for staff and symbol detection, a two-pass vision-language model (Qwen3-VL) for multilingual instrument name identification including Italian, German, and German abbreviation conventions, a Transformer encoder-decoder (TrOMR) for per-staff note recognition, multi-layer rule-based post-processing for cross-part structural repair, and a fine-tuned object detector (YOLO) for dynamics detection. On the OpenScore Orchestra benchmark, GrandOMR achieves a page-level OMR-NED of 0.2854 and a book-level OMR-NED of 0.2968, substantially outperforming homr (0.3210 / 0.5217) and LEGATO (0.6829) while correctly recovering instrument identity, transposition, and dynamics that existing systems discard. Code is available at `https://github.com/2omegaXv/GrandOMR`.

1 Introduction

The International Music Score Library Project (IMSLP) [1] hosts tens of thousands of high-resolution scans of orchestral scores spanning several centuries of repertoire. Converting these images into machine-readable symbolic music — a task known as Optical Music Recognition (OMR) — would enable large-scale musicological analysis, interactive playback, and digital preservation. Yet existing OMR systems fall short in the orchestral setting: rule-based engines such as Audiveris [2] degrade sharply at moderate scan resolution and struggle with printing artifacts; neural staff-level systems such as homr [4] recognise notes accurately but discard all instrument identity; and end-to-end vision-language models such as LEGATO [5] are constrained by output-length limits that preclude full orchestral pages.

The fundamental difficulty is that orchestral score OMR is not a single, homogeneous problem. Staff and system boundary detection is a geometric layout task; instrument name reading must handle multilingual abbreviations and label-to-staff ambiguity; note recognition is a sequence transduction problem with rich rhythmic and pitch relationships; cross-part consistency requires reasoning about musical constraints that are well-defined but vary globally; and dynamics detection amounts to localising a small fixed set of glyph classes amid expressive text. Each subproblem has a different input distribution, output space, and failure mode. A monolithic end-to-end model must simultaneously master all of these, which forces compromises on every front.

We present **GrandOMR**, a hybrid pipeline that decomposes orchestral score OMR into its constituent subproblems and matches each to the most appropriate computer vision method: semantic segmentation (SegNet) for staff and symbol detection, a two-pass vision-language model (VLM) for instrument name identification, a Transformer sequence model (TrOMR) for

per-staff note recognition, rule-based post-processing for cross-part structural repair, and a fine-tuned object detector (YOLO26n) for dynamics detection. This design keeps each component interpretable and controllable, makes it straightforward to replace individual stages as better models become available, and collectively achieves competitive accuracy on full orchestral scores.

The remainder of the paper is organised as follows. Section 2 reviews related work. Section 3 gives a system overview. Section 4 describes each pipeline stage in detail. Section 5 reports experimental results. Sections 6 and 7 discuss limitations and conclude.

2 Related Work

2.1 Rule-based OMR: Audiveris

Audiveris [2] is an open-source OMR engine built around a roughly twenty-step rule-based pipeline that combines staff-line detection, symbol template matching, and a shallow neural classifier for glyph recognition. Its architecture is highly interpretable and produces well-formed MusicXML output for clean engraved scores. However, it imposes stringent image quality requirements: recognition accuracy degrades markedly below approximately 300 DPI, and it is sensitive to printing artifacts, page curvature, and ink bleed-through that are common in IMSLP scans. Audiveris also provides reasonable instrument name extraction from score headers, but it was designed primarily for single-instrument or small-ensemble scores and lacks robust handling of the complex bracket and brace structures found in full orchestral layouts.

2.2 Neural Staff-level OMR: TrOMR / homr

TrOMR [3] is a Transformer-based encoder-decoder that treats per-staff music recognition as a sequence transduction problem: given a cropped staff image, it outputs a linearised token sequence encoding pitch, duration, accidentals, and articulations. By operating at the staff level, TrOMR sidesteps the complexity of multi-staff layout and achieves strong symbol-error rates on standard benchmarks. The homr system [4] augments TrOMR with a UNet-based semantic segmentation front-end that detects staves, noteheads, stems, clefs, and barlines, enabling fully automatic end-to-end recognition from raw page images. However, neither system recovers instrument identity: homr assigns generic part labels and provides no mechanism for cross-page name propagation, making it unsuitable for producing playback-ready orchestral scores.

2.3 Large-scale End-to-end OMR: LEGATO

LEGATO [5] takes a fundamentally different approach, coupling a frozen LLaMA 3.2 vision encoder with an ABC-notation decoder in an end-to-end vision-language model. Trained on a large collection of engraved scores, it achieves state-of-the-art symbol error rates on piano and small-ensemble scores, benefiting from the contextual reasoning capacity of a large pretrained vision backbone. Its primary limitation in the orchestral setting is the output-length constraint inherent to autoregressive decoding: a single page of a full orchestral score can contain hundreds of notes across 20 or more staves, easily exceeding practical token budgets. Additionally, as a single-model system, LEGATO has no explicit mechanism for recovering instrument identities, transposition, or cross-page structural context.

2.4 Positioning

3 System Overview

The pipeline converts a scanned score page into a MusicXML file through five sequential stages. Stage 1 runs SegNet-based semantic segmentation (via the homr framework) to detect staves,

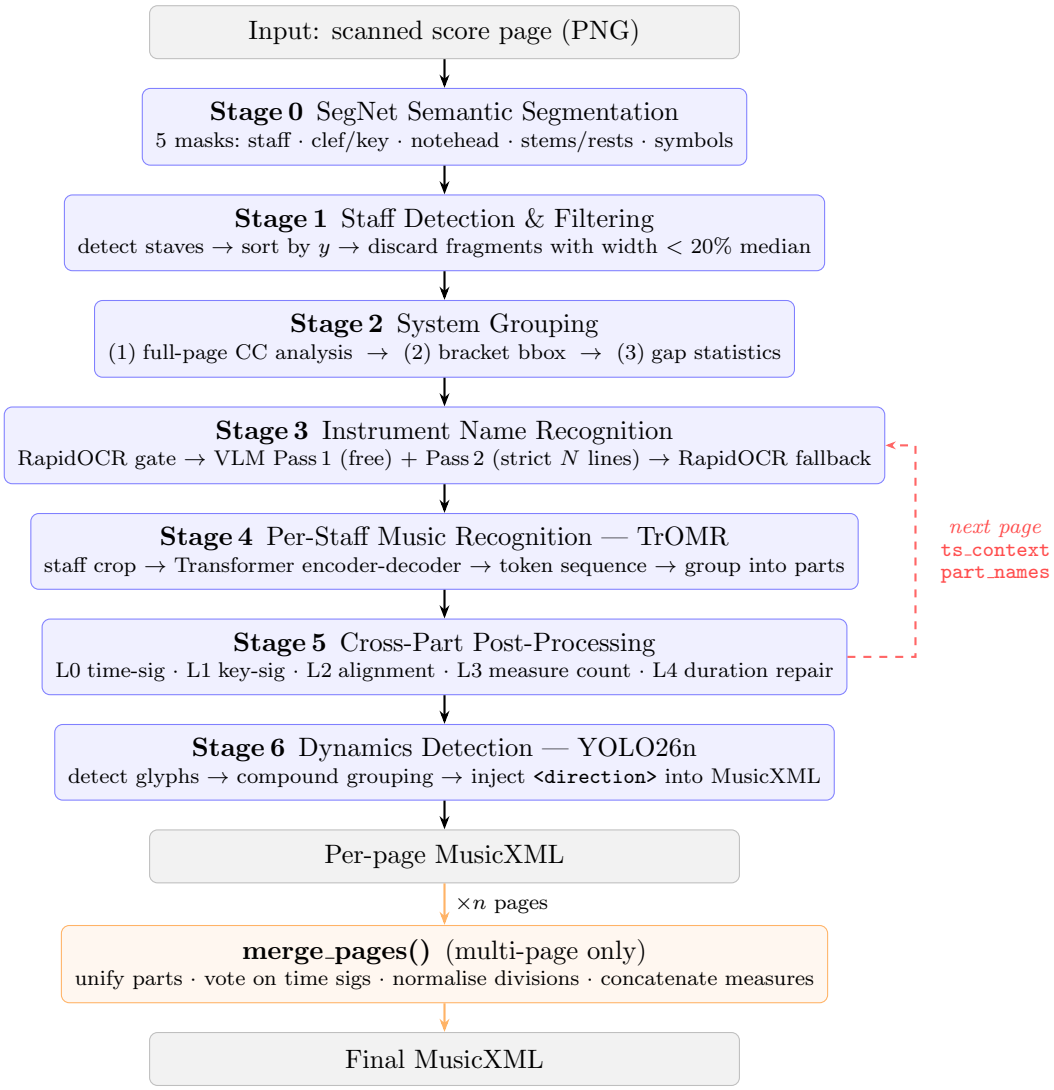

Figure 1: GrandOMR system architecture. Blue stages form the per-page recognition pipeline. The dashed arrow (right) indicates cross-page state passed to the next page: `ts_context` carries the active time/key-signature context into Stage 5; `part_names` propagates the instrument list detected on page 1 into Stage 3. Orange: the multi-page merge step.

	Audiveris	homr	LEGATO	**Ours**
Instrument identity	partial	—	—	✓
Transposing instruments	partial	—	—	✓
Dynamics detection	partial	—	—	✓
Cross-page propagation	—	—	—	✓
Orchestral robustness	—	partial	—	✓

Table 1: Comparison of OMR systems. "Ours" refers to GrandOMR.

noteheads, clef symbols, barlines, and bracket/brace structures, and uses connected-component analysis on the resulting masks to locate system boundaries. Stages 2–3 crop the left margin of each detected system, run a lightweight OCR scan to check for instrument label text, and — when labels are detected — call a two-pass VLM to assign instrument names to staves; names detected on the first page are propagated to subsequent pages via a cross-page override mechanism. Stage 4 crops each individual staff, deskews it, and feeds it to TrOMR, which outputs a linearised symbolic sequence; staves are then grouped into parts according to the detected instrument count. Stage 5 applies a suite of rule-based post-processing layers that correct time signatures (Layer 0), key signatures (Layer 1), and metadata disagreements across parts (Layer 2), normalise measure counts (Layer 3), and repair individual measure durations (Layer 4); transposition elements for non-concert-pitch instruments are injected into the MusicXML at this stage. Stage 6 runs a fine-tuned YOLO26n model to detect dynamics markings and injects `<direction>` elements into the MusicXML output. When multiple pages are processed in sequence, the `ts_context` tuple carries the current time signature, key signature context, and any pending retroactive key-signature fixup across page boundaries.

Subproblem	**Method**	**Rationale**
Staff/symbol segmentation	SegNet	Pixel-level multi-class; regular structure
System boundary detection	CC + gap statistics	Geometric; no semantic understanding needed
Instrument name recognition	VLM (Qwen3-VL)	Open vocabulary, multilingual, context-dependent
Music sequence recognition	TrOMR Transformer	Sequence transduction; complex rhythm
Cross-part consistency	Domain rules	Well-defined constraints; cheaper than retraining
Dynamics detection	YOLO26n (fine-tuned)	Fixed-class local pattern detection

Table 2: Subproblem–method correspondence.

4 Methodology

4.1 Staff and Symbol Segmentation — SegNet (Stage 1)

Staff and system boundary detection is fundamentally a geometric layout problem: the spatial distribution of stave lines is highly regular, and the required decisions ("which pixels belong to a staff line?", "where does one system end and the next begin?") do not require musical understanding. Pixel-level semantic segmentation is therefore a natural fit, and we adopt the

SegNet-based model from the homr framework, which produces five binary masks covering staff lines, clef/key regions, noteheads, stems and rests, and other symbols.

Before system grouping, we apply a narrow-fragment filter: any detected staff whose horizontal extent is less than 20% of the median staff width is discarded, since these arise from SegNet activating on stray ink and would otherwise cause TrOMR to crash during its internal resize.

System boundary detection takes the union of all five segmentation masks and runs connected-component analysis on the full-page binary image. Connected components whose height exceeds one average staff height correspond to the tall bracket or brace structures that delineate a system; their vertical extents are merged with a gap tolerance of one staff height and returned as system intervals. When this primary method yields only a single interval (e.g. for single-system pages or when no brackets are detected), the pipeline falls back first to brace/bracket bounding-box detection from the dedicated brace-dot image, and finally to a gap-statistics heuristic that finds the largest relative jump in the distribution of inter-staff vertical gaps. Each detected system group is then deduplicated: rather than using bounding-box polygon overlap (which falsely fires on adjacent staves with slight SegNet-dilation contact or on staves whose bounding boxes are inflated by stray fragments from a neighbouring stave), we compare stave centre-y coordinates and treat two detections as duplicates only when their centres are within $0.5\times$ the median staff height.

4.2 Instrument Recognition — VLM (Stages 2–3)

Instrument name recognition is an open-vocabulary, context-dependent problem. Orchestral scores use instrument labels from multiple languages: Italian full names such as *Flauto* and *Corno* appear in scores including Bach's Concerto for Four Harpsichords BWV 1065, Mozart's Symphony No. 41 in C major K. 551, and Beethoven's Symphony No. 5 in C minor Op. 67; German abbreviations such as *Kl.* for Clarinet, *Fg.* for Bassoon, and *Pos.* for Trombone — sometimes accompanied by explicit transposition keys (*Kl. in A*, *Trp. B*) — appear in scores including Brahms's Symphony No. 4 in E minor Op. 98 and Mahler's Symphony No. 7 in E minor. A label may also be positioned beside a brace spanning multiple staves. A closed-vocabulary classifier or rule-only OCR approach would require exhaustive hand-crafted patterns for every convention. A vision-language model, by contrast, can interpret visual context, resolve label-to-staff ambiguity by attending to brace geometry, and generalise across conventions without retraining.

The recognition process is OCR-gated: we first run RapidOCR [7] on the left-margin crop of each system, and only invoke the (much more expensive) VLM call when the OCR scan returns at least one non-numeric, non-punctuation text fragment. Pages without visible instrument labels — common for continuation pages in long scores — are handled by cross-page name propagation rather than VLM inference.

When the OCR gate is passed, the VLM (Qwen3-VL-235B [6]) is called in two passes. Pass 1 is free-form: the model receives the full-resolution system crop, a structured hint listing each stave's vertical centre coordinate and the nearest OCR-detected label, and a prompt that asks it to identify instruments from top to bottom with brief reasoning. Pass 2 is a strict formatting pass: it receives the Pass 1 output as additional context and must produce exactly N lines (one per stave) in the form `InstrumentName` or `InstrumentName:Key` for transposing instruments. Up to three retries are attempted; if the VLM still cannot produce N valid lines, the pipeline falls back to pure RapidOCR with per-bracket-group aggregation.

VLM output is normalised through an abbreviation dictionary covering over 100 entries across all standard orchestral instruments in English, German, and Italian, supplemented by a synonym table that maps variant spellings to canonical names (*Klavier* → Piano, standalone *Bass* → Contrabass, *French Horn* → Horn, etc.). Transposition keys are extracted from the colon-separated suffix or a trailing "in X" phrase and normalised (*es* → Eb, *b* → Bb, *as* → Ab).

For multi-page scores, instrument names detected on the first page (or the first page that contains labels) are stored as a reference list. On subsequent pages where OCR detects no instrument labels — common for continuation pages in full orchestral scores — the stored names are used directly without invoking the VLM. When OCR does detect labels on a later page, the VLM runs normally; the stored reference list acts as an override only when its length exceeds the number of detected staves, indicating that some instruments are tacet. In that case, a best-order-preserving subsequence matching procedure selects the subset of reference names that best agrees with both the base instrument types of the detected staves and, when available, the MIDI pitch ranges actually present in the TrOMR output.

4.3 Music Recognition — TrOMR Transformer (Stage 4)

Note recognition on a single staff is fundamentally a sequence transduction task: the input is a two-dimensional image of music notation, and the output is a variable-length sequence of tokens encoding pitches, durations, rests, ties, articulations, and barlines. The dependencies between tokens are non-local — a dotted quarter note carries information about both the note type and its augmentation, a triplet group must be consistent across three tokens, and accidentals within a measure affect subsequent notes — making Transformer encoder-decoder models a natural fit. We use TrOMR [3] as provided by the homr framework, with no modifications to the model weights.

Each detected staff is individually cropped from the preprocessed page image (internally rescaled by SegNet) and passed to TrOMR, which returns a linearised symbolic sequence. Staves are sorted by vertical position and then grouped into parts by dividing the total staff count by the number of detected instruments; each part spans one stave per system. Multi-system pages — where some systems have fewer staves than the first (e.g. reduced orchestration sections) — are handled by processing each system independently and then merging the resulting MusicXML files. When systems differ in stave count, each shorter system is aligned to the largest (reference) system by an order-preserving dynamic-programming procedure that matches staves by clef category and key signature; unmatched reference positions receive rest-filled empty parts, ensuring all systems share the same part structure before merging.

4.4 Structural Post-Processing — Rule-based (Stage 5)

Because TrOMR recognises each staff independently, systematic errors accumulate across parts: the same measure may receive different time signatures in different staves, measure counts may diverge due to barline misdetection, and individual note durations may not sum to the correct bar length. The constraints that govern a correct orchestral score are, however, well-defined: all parts share a single timeline, all staves within a system have the same number of measures, and each measure's total note duration equals the active time signature. Rule-based correction is therefore more appropriate than retraining, since it directly encodes these constraints and is both interpretable and fast.

Layer 0 — Time signature. After TrOMR runs, double barlines (detected as two closely spaced barline bounding boxes, or as a single merged wide-barline box) segment the page into sections. For each section, the pipeline takes a majority vote across all parts to determine the time signature. When TrOMR's vote confidence is below 75% in a post-double-barline section, a VLM is asked to read the time signature from a crop spanning from the double barline to the first note of each sampled staff; the VLM vote overrides TrOMR's reading. Cross-page time-signature context is maintained in the `ts_context` tuple: if the previous page ended without a double barline, the current page inherits the prior time signature; if it ended with one (signalling a section change), TrOMR's reading is trusted. Within a page, continuation segments (those not preceded by a double barline) apply a stricter criterion: a content-based time-signature change

is only accepted when the HOMR symbol reader independently agrees on the same value. This guards against a single part with missed barlines inflating the content-based estimate to an implausible value (e.g. 8/4) while all other parts contribute only whole rests and provide no counter-signal.

Layer 1 — Key signature. Key signatures are propagated by a symmetric rule: in the absence of a double barline since the last system, the previous page's key signature is inherited for all matching parts (matched by instrument name and chromatic transposition); after a double barline, TrOMR's reading at the start of the new section is trusted. A retroactive fixup handles the case where a double barline appears mid-page: the post-double key-signature range is recorded and corrected once the next page's starting key is known from TrOMR.

Layer 2 — Metadata alignment. Residual disagreements in time and key signature within the same measure number across parts are resolved by majority vote, ensuring that all parts carry identical metadata at each measure position.

Layer 3 — Measure count normalisation. The target measure count is the mode across all parts. Parts with too few measures receive appended whole-rest measures; parts with excess measures have their trailing measures removed. Sparse trailing measures (fewer than one third of parts containing notes) are stripped as SegNet/TrOMR recognition artifacts near page boundaries.

Layer 4 — Duration correction. For each measure of each part, position tracking computes the total sounding duration and compares it to the expected duration implied by the time signature. A sequence of increasingly aggressive repair strategies is applied in order: (A) whole-rest duration fixup, (B) uniform scaling when all note durations are off by the same ratio, (B′) note-type–to-duration alignment with backup recalculation, (C) single-note type-change to the adjacent note value, (C′) shrinking or removing a surplus rest, (D) triplet detection and markup, and (E) a proportional fallback scale with rounding correction. A further per-voice pass handles multi-voice measures, and a post-pass marks unlabelled triplet groups.

Transposition. After part names are assigned, the pipeline injects `<transpose>` elements into the first measure of each transposing instrument. The chromatic and diatonic intervals are computed from the key letter (Bb $\rightarrow -2$ semitones, F $\rightarrow -7$ semitones for Horn and English Horn, which always transpose downward). Bass Clarinet additionally receives an octave-change of -1.

4.5 Dynamics Detection — Fine-tuned YOLO26n (Stage 6)

Dynamics markings such as f, p, mf, and sf constitute a fixed, visually distinctive set of glyph classes whose spatial extent is local (a few pixels wide in a typical scan). Object detection is therefore preferable to the VLM used for instrument names: it is faster by an order of magnitude, does not require an API call, and is well-suited to the fixed-vocabulary, bounding-box-output nature of the task.

We fine-tune YOLO26n starting from `yolo26n.pt` (weights pre-trained on COCO) on the DeepScoresV2 dense annotation subset (736 training images, 160 test images) for 100 epochs. The model detects eight classes: `dynamicCrescendoHairpin`, `dynamicDiminuendoHairpin`, `dynamicF`, `dynamicM`, `dynamicP`, `dynamicR`, `dynamicS`, and `dynamicZ`.

Per-class confidence thresholds are set by different criteria reflecting the musical semantics of each symbol. For `dynamicF` and `dynamicP` — the two most common and most consequential dynamics for playback — the threshold is set at the point where precision reaches 0.95, favouring

precision over recall to avoid spurious dynamics. For `dynamicS` (sforzando), which is only meaningful when combined with f or p (to form sf, sfp, etc.), the threshold is set at the F1-optimal point to maximise overall detection at the cost of slightly lower precision.

Post-processing groups spatially adjacent single-letter detections into compound tokens by checking horizontal proximity and vertical alignment, then maps compound tokens to valid MusicXML dynamics: `ff`, `sf`, `mf`, and `fz` all normalise to f; `pp` and `mp` normalise to p. Tokens that mix f and p (such as `fp` or `pf`) and tokens of three or more letters (such as `fff` or `sfz`) are discarded, since they lie outside the MusicXML dynamics vocabulary we target. An OCR pre-scan identifies regions containing expression words (e.g. *cresc*, *dim*, *poco*) and suppresses YOLO detections within those regions, avoiding false matches on the initial letters of Italian expression markings. Detected dynamics are finally injected as `<direction>` elements at the correct measure and note position by mapping detection pixel coordinates back to HOMR coordinate space and consulting the detected barline positions.

4.6 Interactive Localization and MuseScore Plugin (Stage 7)

The MusicXML file is only the final visible product of a much richer recognition pipeline. During recognition, GrandOMR has already computed staff locations, notehead boxes, TrOMR token positions, part assignments, and intermediate symbolic structures. If these signals are discarded after MusicXML generation, we lose information that is useful for inspecting large orchestral scores. There is also a practical gap between the scanned page and the editable score opened in MuseScore: both represent the same music, but MuseScore re-engraves the MusicXML using its own layout engine, so a note in the scan may appear at a different page position, system break, or spacing in MuseScore. For orchestral pages with many staves, manually finding the corresponding MuseScore note is slow and error-prone.

We therefore implement a plugin-oriented interaction layer around GrandOMR. The user can click a note on the original page image in a browser viewer, and the MuseScore plugin automatically selects the corresponding note in the opened score. Once the note is selected, the user can use MuseScore's normal editing and playback tools, for example starting playback from that location. To the best of our knowledge, this is the first plugin that supports note-level localization from a note in the original score image to the corresponding note in MuseScore. We also add convenience features such as browser-side playback control and part-level note localization. A demonstration video of the plugin is included in the supplementary materials. The core technical problem is preserving the connection from image evidence, to generated XML notes, to live MuseScore note objects.

Image note to MusicXML note. GrandOMR inherits HOMR's hybrid architecture: visual geometry such as staff lines and noteheads is extracted by a segmentation network, while each staff is decoded by a TrOMR-style transformer into symbolic note tokens that are later written as MusicXML notes. The TrOMR stage is end-to-end, so it does not directly provide a correspondence between HOMR's detected geometric objects and its output tokens. The missing link is therefore the connection between symbolic notes and their locations in the original image.

We use the attention-derived pixel-level focus point attached to each TrOMR note token as a rough localization cue, and match it to HOMR's detected notehead boxes on the same staff. Single notes can be matched directly by distance. Chords require one extra step: because several notes share the same horizontal position, we first bind the visual chord group and then reassign individual chord tones by the actual y-coordinates of their notehead bounding boxes, using pitch as an additional cue when available. In practice, we found this localization strategy to be reliable and robust enough for interactive use.

MusicXML note to MuseScore note. The second mapping links the generated MusicXML notes to the note objects visible inside MuseScore. Ideally, we would identify MuseScore notes using the same note ids that appear in MusicXML. However, after MuseScore imports a MusicXML file, those ids are not reliably exposed through the plugin API. Another possibility is structural matching, for example locating a note by its part, measure, position inside the measure, and pitch. This is also brittle, because it would require us to reproduce MuseScore's own import and engraving decisions, including how it orders notes, groups chords, and handles voices.

Our solution is to create a temporary tagged MusicXML file. For every clickable note, we inject a hidden lyric as a marker carrying a pseudo id. We then use MuseScore's command-line interface to scan this tagged file automatically, without requiring the user to open it manually. By finding the marker in MuseScore's scan result, we can locate the corresponding MuseScore note directly, without reproducing MuseScore's layout or import logic ourselves.

5 Experiments

5.1 Experimental Setup

We compare against three baselines representing distinct points in the OMR design space. **Audiveris** serves as the rule-based baseline; we use its default configuration and measure its output against ground truth without any post-processing. **homr** (TrOMR + SegNet front-end, no instrument identity) serves as the neural staff-level baseline; its MusicXML output carries generic part labels. **LEGATO** serves as the end-to-end vision-language baseline; pages are processed individually due to its output-length constraint.

5.2 Instrument Classification

Across all evaluated scores — spanning Italian full names (*Flauto*, *Corno*), German full names, and German abbreviations — GrandOMR achieves near-perfect instrument name recognition and correct transposition assignment for all transposing instruments (Clarinet, Bass Clarinet, Horn, Trumpet, English Horn).

5.3 Music Recognition

Benchmark. There is no existing benchmark metric that fully captures all the improvements made by GrandOMR. Many of our changes target practical orchestral usability: part identity, playback-related musical attributes, and score-level consistency. These details are musically important. For example, a wrong instrument label or transposition may look like a small notation error, but it can change the playback result for an entire part. Still, even an incomplete quantitative metric can show part of the improvement, especially when it reflects structural stability across many staves and pages.

We evaluate on the OpenScore Orchestra subset[1], using the `pages_transcribed` configuration, test split, filtered by `corpus == "orchestra"`. The original subset contains 242 full-page orchestra pages from 8 scores. During ground-truth review, we found severe ground-truth errors in three scores and excluded them from the final evaluation: `Beethoven_Op.68_5`, `Beethoven_Op.21_2`, and `Bach_BWV.232_18`. The remaining clean subset contains 66 pages from 5 scores: `Bach_BWV.232_02`, `Bach_BWV.232_09a`, `Bach_BWV.232_12`, `Bach_BWV.232_16`, and `Beethoven_Op.21_4`.

The OpenScore benchmark is originally page-level. To evaluate the score-level behavior that GrandOMR is designed for, we also reconstruct the 5 accepted scores as book-level inputs by grouping their pages back into complete scores. This book-level setting exposes errors that

[1] `https://huggingface.co/datasets/zzsi/openscore`

page-level averaging can hide, such as unstable part alignment and broken measure continuity across pages.

Metric. We use OMR-NED as the main metric. OMR-NED is the metric recommended in TrOMR. It is a normalized edit-distance metric designed for OMR evaluation. It compares the predicted MusicXML with the ground truth after converting the score into an OMR-oriented symbolic representation. In practice, OMR-NED is most sensitive to note-level content and structural alignment, including wrong notes, rhythm notation errors, missing musical structure, and cross-part misalignment. It also counts local symbols such as accidentals, key signatures, dynamics, and articulations, but these usually contribute less to the total score than large note, measure, or staff alignment errors.

	Page-level OMR-NED ↓	**Book-level OMR-NED ↓**
LEGATO	0.6829	—
homr	0.3210	0.5217
GrandOMR	**0.2854**	**0.2968**

Table 3: OMR-NED on the cleaned OpenScore Orchestra subset.

Results. We report homr, GrandOMR, and LEGATO because all three can produce page-level MusicXML on the clean subset. LEGATO is not reported at book level because its native inference interface does not support book-level input.

On page-level OMR-NED, GrandOMR improves over homr from 0.3210 to 0.2854. This is a moderate improvement, which is expected because OMR-NED is dominated by note and structure alignment rather than by every local post-processing target. Nevertheless, the improvement shows that the additional post-processing does not merely add metadata or playback convenience; it also improves the symbolic score under a standard OMR metric.

The book-level result is more informative for our goal. homr degrades substantially from page-level 0.3210 to book-level 0.5217. This does not necessarily mean that each page is recognized worse. Rather, book-level evaluation amplifies cross-page alignment failures: if the part order, number of parts, or measure continuity changes across pages, musicdiff can interpret the resulting mismatch as large insertions or deletions of entire measures or staves. These errors are limited to one page in page-level scoring, but they propagate through the merged score in book-level scoring.

GrandOMR is much more stable in this setting. Its book-level OMR-NED is 0.2968, close to its page-level score and far below homr's book-level result. This suggests that the score-level mechanisms in GrandOMR, including part tracking, page merging, and orchestra-specific post-processing, reduce the cross-page misalignment errors that dominate full-score evaluation. In other words, even though OMR-NED does not fully measure every improvement we made, it still captures the main structural benefit of adapting homr to large orchestral scores.

5.4 End-to-End Comparison with homr / LEGATO

Table 4 summarises a qualitative end-to-end comparison on Mahler Symphony No. 7, page 255 — a full orchestral page with 23 staves spanning woodwinds, brass, percussion, harp, and strings. All four systems were run without ground-truth supervision.

GrandOMR correctly identifies all 23 instruments — including transposing instruments such as Clarinet in E♭, Horn, and Trumpet in F — and produces playback-ready MusicXML with dynamics. Audiveris also detects 23 parts and achieves the closest note count to GrandOMR (318), but its OCR-based name extraction fails on German abbreviations, producing garbled

	GrandOMR (ours)	Audiveris	homr	LEGATO
Recognised parts	**23**	**23**	12	18
Correct names	**23**	0	0	0
Notes extracted	**366**	318	357	60
Instrument identity	✓	—	—	—
Dynamics	✓	—	—	—
Cross-page context	✓	—	—	—

Table 4: End-to-end comparison on Mahler No. 7, page 255 (23 staves). "Correct names" counts parts assigned the right canonical instrument name. Note counts are approximate totals across all parts.

labels such as *m- B E3*, *Pug. 2*, and *Hg]*. homr recovers a comparable note count (357) using the same TrOMR recognition engine, but assigns all parts the generic label "Piano" and provides no instrument identity or dynamics. LEGATO extracts only 60 notes across 13 unnamed voices; the output-length constraint causes the model to terminate generation before covering most staves, leaving 12 of the 13 voices completely empty.

Figure 2 shows the piano rolls for all four systems on the same page.

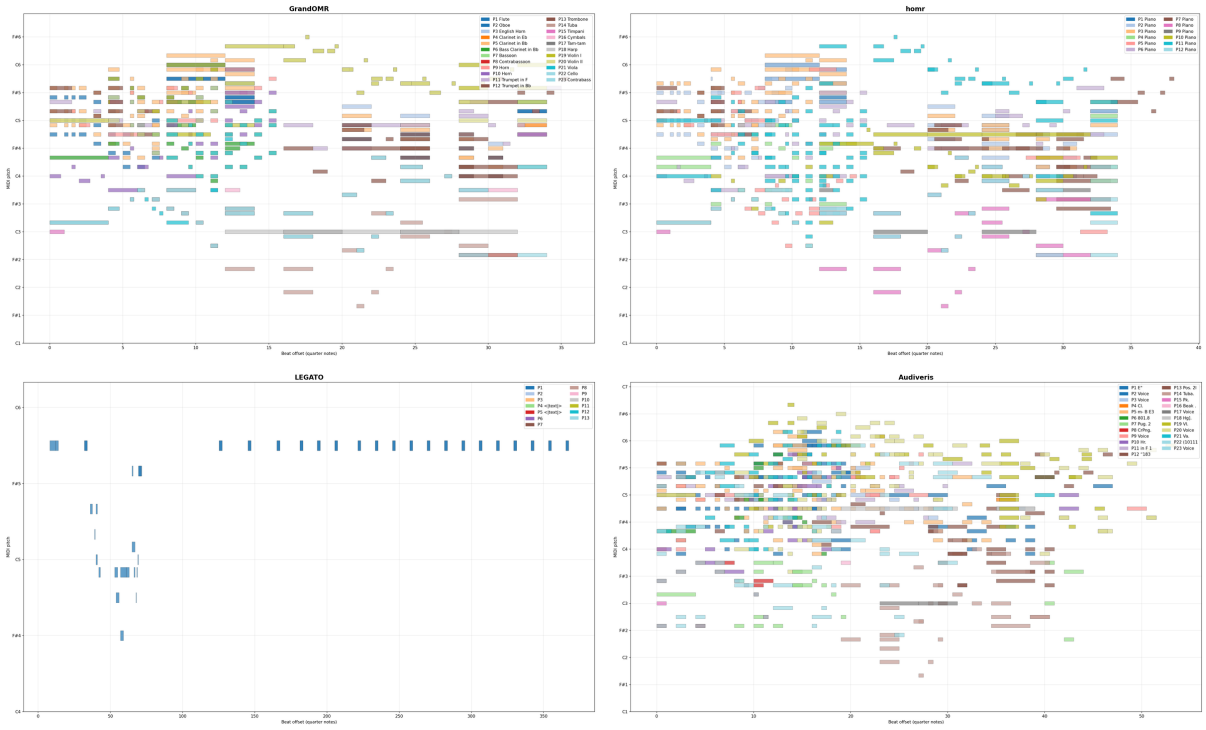

Figure 2: Piano rolls for Mahler No. 7, page 255 (2×2 grid). *Top-left:* GrandOMR (23 named parts, 366 notes). *Top-right:* homr (12 unnamed Piano parts, 357 notes). *Bottom-left:* LEGATO (13 parts, 60 notes — only first voice non-empty). *Bottom-right:* Audiveris (23 parts, 318 notes, names garbled by OCR).

5.5 Dynamics Detection

The per-class confidence thresholds were chosen by inspecting the precision–recall curve on the DeepScoresV2 dense test set. Table 6 summarises the precision and recall achieved at the selected operating point for the three classes with per-class thresholds.

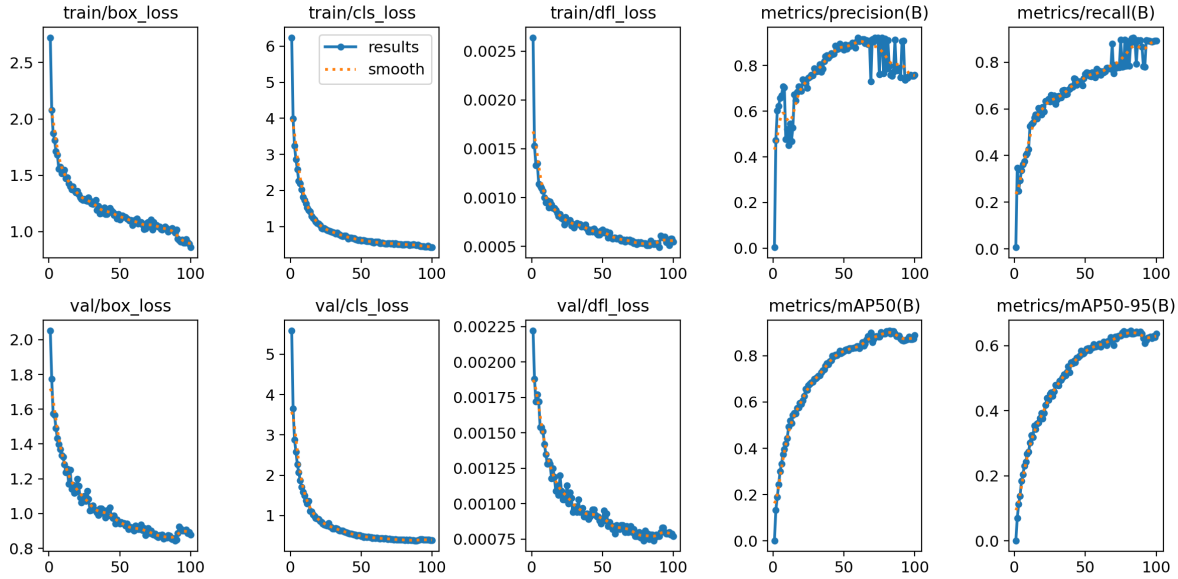

Figure 3: Training curves (100 epochs on DeepScoresV2 dense).

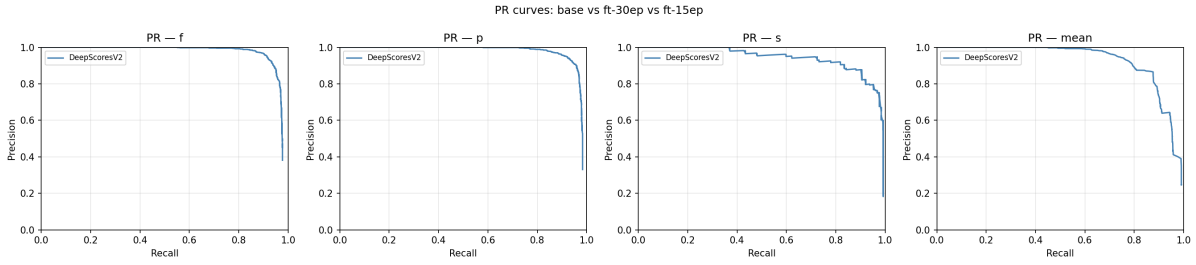

Figure 4: Precision–Recall curves on the DeepScoresV2 dense test set.

	mAP50
All classes	0.908
dynamicF	0.964
dynamicP	0.971
dynamicS	0.943

Table 5: Dynamics detection mAP50 on the DeepScoresV2 dense test set.

Class	Threshold	Strategy	Precision	Recall
dynamicF	0.65	Precision-first ($P \approx 0.95$)	≈ 0.952	≈ 0.875
dynamicP	0.65	Precision-first ($P \approx 0.95$)	≈ 0.952	≈ 0.875
dynamicS	0.31	Recall-first (F1-optimal)	0.877	0.906

Table 6: Confidence threshold selection for the three per-class dynamics classes. `dynamicF` and `dynamicP` use precision-first thresholds to avoid spurious dynamics in playback. `dynamicS` uses the F1-optimal threshold because sforzando markings only occur alongside f or p, so slightly lower precision is acceptable.

5.6 Qualitative Results

Figure 5 shows the piano roll for a single page of Mahler Symphony No. 7 (page 5, 4/4, five measures). GrandOMR correctly identifies all 17 instruments — including transposing instruments (Clarinet in A, Bass Clarinet in A, Horn in F) — and places notes within their expected MIDI pitch ranges. The quality checker reports one residual duration error in Violin II, measure 4 (duration 5.0 instead of 4.0), illustrating the kind of minor inaccuracy that remains after Layer 4 repair.

Figure 6 shows the result after merging pages 5–7. The three-page output contains 23 parts, reflecting additional brass instruments (Trombones, Tuba, Trumpet) that enter on page 6. Cross-page name propagation correctly unifies the part list, and `ts_context` keeps time and key signatures consistent at page boundaries. Minor issues include one Tuba note slightly above its standard range and two additional duration errors in the string section, consistent with TrOMR's known difficulty on fast passage-work.

6 Limitations and Future Work

The primary limitation of GrandOMR is the recognition accuracy of TrOMR, the per-staff sequence model at the core of Stage 4. TrOMR operates independently on each staff, so errors in pitch, duration, or barline placement are common, particularly on complex rhythmic figures, beamed groups, and low-resolution scans. These errors propagate downstream and necessitate the multi-layer rule-based post-processing described in Stage 5 — a substantial engineering effort whose repair heuristics may themselves introduce secondary errors or overcorrect in edge cases. The overall effect is that the accuracy ceiling of the pipeline is set by TrOMR, and the complexity of Stage 5 is a direct consequence of that ceiling.

The most impactful direction for future work is therefore to raise per-staff recognition accuracy. Two complementary approaches are promising. First, a higher-accuracy single-staff model — whether through fine-tuning TrOMR on a larger or more diverse dataset, or adopting a more capable architecture — would reduce the burden on post-processing and improve output quality directly. Second, an end-to-end multi-staff model trained on manually corrected full-system transcriptions could bypass the per-staff decomposition entirely, allowing the model to exploit cross-staff harmonic and rhythmic context that is invisible to a per-staff approach.

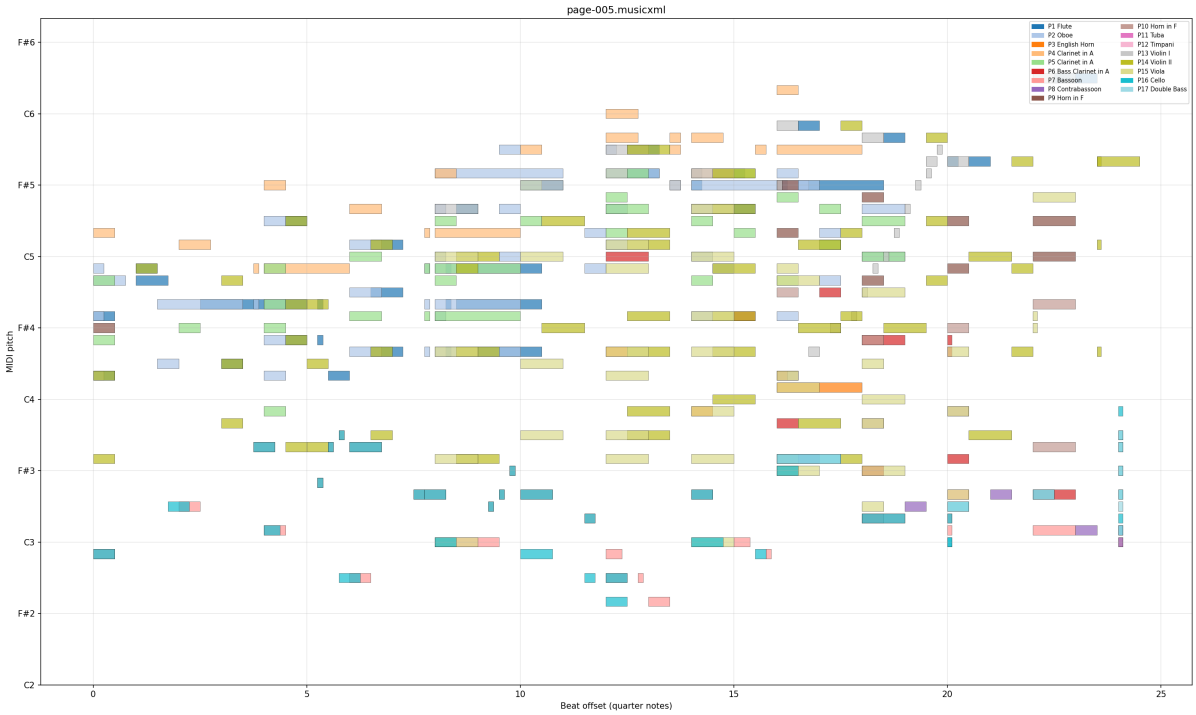

Figure 5: Piano roll for Mahler No. 7, page 5 (single page, 17 parts, 5 measures). Each colour represents one instrument part; block width encodes note duration.

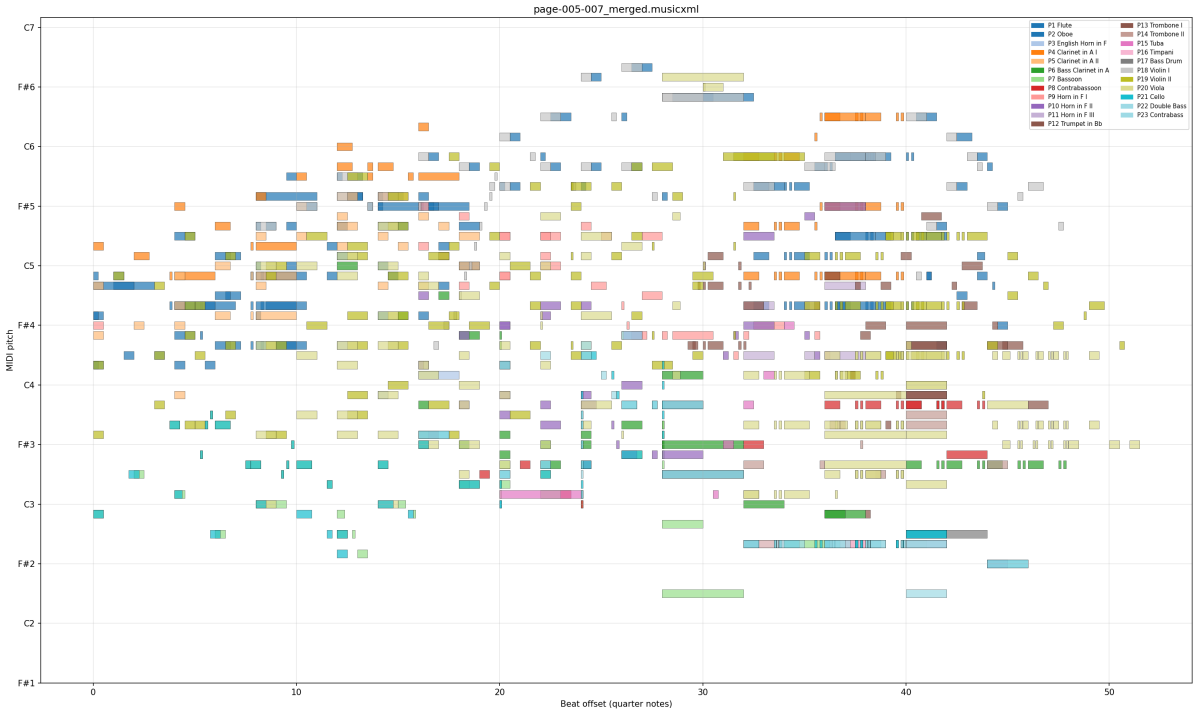

Figure 6: Piano roll for Mahler No. 7, pages 5–7 (multi-page merge, 23 parts, ≈15 measures). New instruments entering on page 6 are correctly appended to the unified part list.

7 Conclusion

We have presented GrandOMR, a hybrid OMR pipeline for full orchestral scores built around a single design principle: decompose the recognition task into its constituent subproblems and match each to the most appropriate computer vision method. Pixel-level semantic segmentation handles the regular geometric structure of stave detection; a two-pass vision-language model handles the open-vocabulary, multilingual challenge of instrument name reading; a Transformer encoder-decoder handles the sequence transduction nature of per-staff note recognition; rule-based post-processing exploits the well-defined consistency constraints shared by all parts of an orchestral score; and a fine-tuned object detector handles the fixed-vocabulary local-pattern nature of dynamics detection. This decomposition yields a pipeline that is more interpretable and controllable than a monolithic end-to-end approach: each component can be inspected, replaced, or improved independently, and domain knowledge can be incorporated directly as rules without retraining. Experiments on IMSLP orchestral scans demonstrate that GrandOMR achieves competitive results on full scores that current end-to-end systems cannot handle, while preserving instrument identity, transposition, dynamics, and cross-page structural context that are essential for interactive playback.

References

[1] International Music Score Library Project (IMSLP). `https://imslp.org`.

[2] Audiveris OMR Engine. `https://github.com/Audiveris/audiveris`.

[3] Z. Li et al., "TrOMR: Transformer-based Polyphonic Optical Music Recognition," *ICASSP*, 2023.

[4] Yoyo et al., "BreezeWhite/oemer: v0.1.7," Zenodo, 2023. *homr*: `https://github.com/liebharc/homr`.

[5] G. Yang et al., "LEGATO: Large-scale End-to-end Generalizable Approach to Typeset OMR," *arXiv:2506.19065*, 2025.

[6] Qwen Team, "Qwen3 Technical Report," Alibaba Cloud, *arXiv:2505.09388*, 2025.

[7] RapidAI, "RapidOCR: A Lightweight, Fast and Accurate OCR Engine," 2022. `https://github.com/RapidAI/RapidOCR`.