

subocr service

Service Overview: SubOCR (<https://subocr.drheck.dev/>) is an end-to-end Japanese burned-in video subtitle (*te/lo/p* / 字幕) extraction, 逐字 逐句 逐行 逐帧 逐字 逐句 逐行 逐帧 annotation, offline **JMdict + JLPT (N5-N1)** morphological dictionary lookup, LLM study-note refinement, and interactive video study platform running on host `ul`, accelerated by the **AMD Radeon AI PRO R9700 (gfx1201, RDNA 4, 32 GB VRAM)** GPU.

1. Architecture & Network Topology

SubOCR is deployed as a containerized FastAPI + ONNX Runtime (ROCm 7.2 MIGraphX) service bound to a dedicated IPv6 ULA address on `eth77`, reverse-proxied through Nginx with Authentik ForwardAuth single sign-on, and paired with the local `ollama-rocm` (`qwen2.5:32b`) LLM engine on the same GPU.

\$\$\text{Video Frame} \rightarrow \text{PP-OCrv5 (MIGraphX FP32)} \rightarrow \text{Qwen2.5-32B (ROCm)} \rightarrow \text{UniDic + JMdict/JLPT Study View}\$\$

- **Public URL:** <https://subocr.drheck.dev/>
- **Dedicated ULA IPv6 Address:** `fd77:7777:5010:0:1337:f1ac:d00d:0c70` (`eth77`, port `8000`)
- **Host Directory:** `/data/docker/subocr/`
- **Databases:**
 - **Application Database:** `/data/docker/subocr/data/subocr.db` (SQLite 3 in WAL mode; stores jobs, timed cues, readings, translations, and custom edits)
 - **Offline Dictionary & JLPT Database:** `/data/docker/subocr/data/jmdict_jlpt.db` (SQLite 3, 63 MB; 218,856 EDRDG JMdict entries, 499,325 indexed kanji/kana forms, and 7,972 JLPT N5-N1 vocabulary entries)
- **Export & Sidecar Storage:**
 - Per-job exports: `/data/docker/subocr/data/jobs/<job_id>/`
 - Automatic sidecar writes next to local videos: `/data/language/Japanese/Video/...` (mounted read-write at `/videos` inside the container)
- **Authentication:** Authentik ForwardAuth (`subocr` ProxyProvider, 30-day token validity) via `/etc/nginx/sites-available/subocr`
- **DNS Resolution:**
 - Local Unbound (`danknasty`, `@10.0.0.1`): `A 10.0.0.100`, `AAAA fd77:7777:5010::1337:f1ac:d00d:aaaa`

2. Two-Stage GPU Processing & Morphological Dictionary Pipeline

When a job is submitted from <https://subocr.drheck.dev/new> (either by selecting a local video under `/data/language/Japanese/Video`, uploading a video file, or providing a YouTube URL fetched via `yt-dlp`), SubOCR executes a two-stage GPU pipeline on the **AMD Radeon AI PRO R9700 (gfx1201)** followed by deterministic morphological and dictionary analysis:

Stage 1: ROCm 7.2 MIGraphX Visual Subtitle OCR (`PP-0CRv5`)

1. Frame Sampling & Spatial Crop:

- OpenCV samples the video at a configurable rate f_s (default 2.0 fps), or frame interval $\Delta t = 0.5 \text{ s}$.
- To isolate burned-in subtitles and ignore top-of-screen variety-show topic banners by default, frames are cropped to the lower subtitle region $y \in [0.58H, 0.98H]$.

2. GPU Inference (`MIGraphXExecutionProvider FP32`):

- Detection (`PP-0CRv5_server_det_infer.onnx`) and recognition (`japanese_PP-0CRv5_mobile_rec_infer.onnx`) run via `onnxruntime-rocm` with `MIGraphXExecutionProvider`.
- Compiled MIGraphX kernel plans (`.mxr`) are cached in `/data/docker/subocr/cache/migraphx` so subsequent container restarts bypass ahead-of-time compilation.
- **Precision Note:** `migraphx_fp16_enable` is explicitly set to `0` (`FP32`) because FP16 causes bounding-box coordinate underflow in `PP-0CRv5` detection on `gfx1201`.

3. Bilingual Line Classification & Spatial Role Separation:

- Detected text boxes within each frame are sorted vertically and classified into Japanese (`ja_lines`) or English (`en_lines`) based on character script ratios.
- Within the lower subtitle band, higher-positioned smaller Japanese commentary boxes ($y < 0.66H$) are separated from the primary bottom speaker dialogue ($y \geq 0.66H$) so simultaneous on-screen editor commentary and speaker subtitles are preserved cleanly.

4. Temporal Deduplication:

- Consecutive frames whose normalized Japanese string similarity satisfies: $\text{sim}(A, B) = \frac{2 \cdot |M|}{|A| + |B|} \geq 0.78$ are merged into a single timed subtitle cue $[t_0, t_1]$, selecting the highest-confidence OCR

recognition frame as the canonical raw text (`ja_raw`) and storing the initial English line (`en_raw`).

Hardware & ROCm Note: Always keep `MIGRAPHX_DISABLE_MIOPEN_FUSION=1` and `migraphx_fp16_enable=0` enabled in `/home/docker-data/.config/containers/systemd/suboccr.container`, and never set `UserNS=keep-id` on the 23.3 GB ROCm image (default rootless user namespace starts in under 2 seconds). Enabling FP16 on `gfx1201` degrades bounding-box detection recall in `PP-OCRv5`.

Stage 2: Contextual LLM Refinement, Kanji-Only Furigana & Offline Dictionary Tagging

1. Whole-Conversation Person Name Identification (`extract_conversation_person_names`):

- Before batched refinement or vocabulary page generation, `qwen2.5:32b` scans the full numbered transcript across the entire video (together with the video title) to identify recurring speakers, cast members, and mentioned individuals.
- Extracted names are stored in `jobs.person_names_json` (mapping each canonical surface form and OCR variant to its reading, English romanization, role description, and `####` tag), automatically repairing visual OCR name typos (e.g., `####` / `####` -> `###`) and preventing person names from being misclassified as common homophones in vocabulary chips.

2. Batched Contextual Refinement (`qwen2.5:32b` on `ollama-rocm`):

- Extracted cues are processed in batches of 15 cues against `http://169.254.1.2:11434/api/chat` (`qwen2.5:32b`, routed via `pasta` `--map-host-loopback` to `127.0.0.1:11434`).
- The LLM receives both the raw Japanese OCR (`ja_raw`) and any burned-in English subtitle (`en_raw`), repairing single-character visual OCR confusions (e.g., katakana `□` / `□`, `□` / `□`, or visually degraded glyphs) while preserving the original spoken phrasing, slang, and dialect.
- If `en_raw` is present and accurate, it is cleaned; if missing or garbled, a natural English translation (`en_text`) is synthesized alongside contextual idiom/slang notes (`vocab`).

3. Punctuation & On-Screen Symbol Restoration (`restore_raw_punctuation`):

- Post-processing compares `ja_raw` and `ja_text` to ensure the LLM never strips on-screen quotation marks (`"..."`, `□...□`, `□...□`), wave dashes (`~`, `□`), exclamation/question marks (`!`, `?`), ellipses (`...`), or counter katakana (`####`, `6###`).
- Example preserved cue: `"#####~!"□<ruby>□<rt>□</rt></ruby>###<ruby>□<rt>□</rt></ruby>#####`

4. Deterministic Kanji-Only Furigana (`fugashi` + `unidic-lite`):

- Morphological tokenization (`fugashi` + `unidic-lite` + `jaconv`) in `generate_kanji_furigana()` strips leading, trailing, and internal okurigana and attaches `<ruby>` annotations **strictly above Kanji characters**, leaving hiragana and katakana unannotated:
- **Example Render:** 日本 語 学 6 月 1 日 開始 — 日 本 語 学 6 月 1 日 開始

5. **Exhaustive Per-Word Tokenization, Grammar Categorization & Offline JMdict / JLPT Lookup** (`analyze_cue_tokens`):

- Every word in every subtitle cue is tokenized by `fugashi` (`MECAB_TAGGER`), lemmatized to its dictionary base form (`orthBase`), and matched against the local read-only `/data/docker/subocr/data/jmdict_jlpt.db` SQLite database and a curated `GRAMMAR_GLOSS_MAP` for Japanese particles and colloquial auxiliaries (`は`/`が`, `を`, `に`, `で`, `から`, Kansai `や`/`が`/`や`, etc.).
- Each token is assigned:
 - **JLPT Level Badge** (`jlpt`): `N5`, `N4`, `N3`, `N2`, `N1`, or `Other`
 - **Part-of-Speech Filter & Chip Tag** (`pos_cat` / `pos_label`): Grouped under the 6 `pos` filter buttons (`名`, `動`, `名詞`, `数`, `接`, `助`) and labeled on each vocabulary chip with its specific Japanese grammatical category (see the complete reference table below)
 - **English Gloss** (`meaning`): Offline JMdict definition or grammatical function

Reference Table: Japanese Part-of-Speech (`pos`) Filter Groups & Vocabulary Chip Tags

Chip Tag (<code>pos_label</code>)	Reading (<code>orthBase</code>)	Filter Button (<code>pos_cat</code>)	English Translation	Grammatical Role & Subtitle Examples
<code>名</code>	名詞	<code>名</code> (Noun)	Common Noun	General nouns and <i>suru</i> -noun stems (e.g., <code>猫</code> , <code>犬</code> , <code>学校</code>)
<code>名詞</code>	名詞	<code>名</code> (Noun)	Proper Noun	Places, countries, people, and brand names (e.g., <code>東京</code> , <code>日本</code> , <code>山崎</code>)
<code>代</code>	代名詞	<code>名</code> (Noun)	Pronoun	Personal and demonstrative pronouns (e.g., <code>私</code> , <code>君</code> , <code>彼</code> , <code>彼女</code> , <code>それ</code>)
<code>数</code>	数詞	<code>名</code> (Noun)	Counter	Numerical counter suffixes (e.g., <code>一つ</code> , <code>二つ</code> , <code>三つ</code> , <code>四つ</code> , <code>五つ</code>)
<code>動</code>	動詞	<code>動</code> (Verb)	Verb	All main verbs (<i>Godan</i> , <i>Ichidan</i> , <i>Suru</i> , <i>Kuru</i>); displayed as <code>conjugated (dictionary form)</code> when inflected (e.g., <code>行く (行く)</code> , <code>食べる (食べる)</code>)
<code>接</code>	接辞	<code>接</code> (Adjective)	<i>i</i> -Adjective	Inflecting adjectives ending in <code>-i</code> (e.g., <code>寒い</code> , <code>暑い</code> , <code>長い</code>)

Chip Tag (pos_label)	Reading ()	Filter Button ()	English Translation	Grammatical Role & Subtitle Examples
な	な な	な (Adjective)	na-Adjective	Adjectival nouns that take な before nouns (e.g., な , な , な)
の	の の	の (Adjective)	Pre-noun Adjectival	Non-inflecting words that directly modify nouns (e.g., の , の , の , の)
は	は は	は (Adverb)	Adverb	Words modifying verbs, adjectives, or clauses (e.g., は , は , は , は)
と	と と	と (Expression)	Expression / Idiom	Multi-word idioms, set phrases, and contextual slang compounds
や	や や	や (Expression)	Interjection	Exclamations, greetings, and conversational fillers (e.g., や , や , や , や)
と	と と	と (Expression)	Conjunction	Sentence/clause connectors (e.g., と , と , と , と)
は	は は	は (Particle)	Particle (General)	General or adverbial/quotative particles (e.g., は , は , は , は)
が	が が	が (Particle)	Case Particle	Marks grammatical case relationships (e.g., が , が , が , が , が , が)
は	は は	は (Particle)	Topic / Binding Particle	Marks topic, contrast, or inclusion (e.g., は , は , は)
は	は は	は (Particle)	Sentence- Ending Particle	Adds tone, question, or emphasis at clause end (e.g., は , は , は , は , は)
は	は は	は (Particle)	Conjunctive Particle	Connects clauses (e.g., は / は , は , は , は , は , は)
は	は は	は (Particle)	Auxiliary Verb	Inflecting verbal/copular endings and colloquial contractions (e.g., は / は , は , は , は , は / は , は , は , は , は)
は	は は	は (Particle)	Prefix	Honorific or intensifying prefixes attached to stems (e.g., は , は , は , は)
は	は は	は (Particle)	Suffix	Meaning-modifying suffixes attached to stems (e.g., は , は , は , は , は)

3. Web Application Layout & Study Controls

SubOCR uses a sharp-cornered (`border-radius: 0`) layout organized into three focused pages:

- **Main OCR Library** (<https://subocr.drheck.dev/>):

- Displays all OCR jobs with status badges, cue counts, video durations, category folders, and a `Source Video ↗` link whenever an original video URL (`youtube_url`) is recorded.
- Provides direct export and download links (`JA .SRT`, `Bilingual .SRT`, `WebVTT .VTT`, `Anki .TSV`, `Study .MD`, `JSON`, and `Save to /data`) on each OCR job card.
- Supports pinning (`★`), category filtering, searching, sorting (`Pinned & Newest`, `Oldest First`, `Title (A-Z)`, `Most Cues`), inline editing (`Organize` drawer for Title, Category, and `Original Video URL`), one-click video recovery (`Re-download Video` via `POST /api/jobs/{job_id}/redownload-video` if a local video file is missing on disk), and job deletion (`Delete`).

- **New Subtitle OCR** (<https://subocr.drheck.dev/new>):

- Dedicated job submission form for local videos under `/data/language/Japanese/Video`, direct file uploads, or YouTube URLs, with optional start/end timestamps (`start_sec`, `end_sec`), frame sampling FPS, category assignment, and an `Original Video URL` field across all tabs (plus automatic YouTube ID extraction from `-<11char_id>.mp4/.mkv` filenames) so source URLs are preserved for backup recovery.

- **Dedicated OCR Study View** (https://subocr.drheck.dev/ocr/{job_id}):

- **Resizable Split-Screen Workspace:** Full-viewport two-column layout with a `52vw` default left video column and an interactive `8px` vertical drag handle (`#split-resizer`) between the left video panel and right subtitle cue panel. Dragging the divider resizes the video player and subtitle list in real time, persists the chosen split width in `localStorage` (`subocr_split_width`), and resets to `52vw` on double-click.
- Compact top bar above the HTML5 video player with `← Library`, `Source Video ↗` link, category badge, and small muted video title. Native `<video controls>` and WebVTT overlays are disabled on the video canvas so burned-in subtitles at the bottom of the frame are never obscured; instead, a dedicated external player control bar (`.player-bar`) sits directly below the video with Play/Pause, an interactive timeline scrubber, `currentTime / duration` readout, volume/mute controls, and fullscreen toggle. If the local video file is ever missing on disk, a recovery banner appears above the player with a `Re-download from Source URL` button (`yt-dlp`).
- Clicking any subtitle card seeks the video to that cue's start timestamp, starts playback, and highlights the active card in high-contrast amber/gold (`#fbbf24` border/pill and `#fef08a` text). Each card header displays the cue index, timestamp span, and `OCR <confidence>%`.
- **Fixed-Width Display & Vocabulary Filter Controls** (under the player controls on the left, persisted in `localStorage` with fixed-width `ON/OFF` labels that never shift button size):
 - **Display Toggles:** `Japanese Subs`, `Furigana`, `English Sub`, `Vocab`, and `Original OCR`
 - **JLPT Level Filters:** `N5`, `N4`, `N3`, `N2`, `N1`, `Other`
 - **(Part-of-Speech) Filters:** `□□` (Noun), `□□` (Verb), `□□□` (Adjective), `□□` (Adverb), `□□` (Expression), `□□` (Particle / Auxiliary)
- **Universal Pre-Generated Vocabulary & Grammar Pages** (<https://subocr.drheck.dev/{category}/{pos}/{level}.html>):

- Clicking any vocabulary box on a subtitle cue card navigates to a universal, pre-generated reference page for that word, phrase, or grammatical particle (e.g., [https://subocr.drheck.dev/\[\]/\[\]_.html](https://subocr.drheck.dev/[]/[]_.html) or [https://subocr.drheck.dev/\[\]/\[\]_.html](https://subocr.drheck.dev/[]/[]_.html)).
- Pages are pre-generated automatically in `/data/docker/subocr/data/vocab_pages/{pos_label}/{slug}.html` at the end of every video analysis job (and updated whenever a subtitle cue is edited).
- Each vocabulary page includes:
 1. **Full Dictionary Senses & JLPT Level:** All matching definitions and readings from `jmdict_jlpt.db` (218,856 JMDict entries + 7,972 JLPT entries + colloquial/dialect grammar map).
 2. **Interactive 300x300px Kanji Stroke Order & `kanji.drheck.dev` Integration:** For any word containing Kanji, embeds an interactive 300x300px KanjiVG + Vivus stroke-order viewer (160x160px per frame in Step-by-Step mode, with compact 5.5px SVG stroke numbers, grid lines, and replay controls) alongside direct links to `https://kanji.drheck.dev/?text=...&mode=animated&duration=327&showNumbers=true&showGrid=true&numberSize=6&boxSize=300&nightMode=true`.
 3. **Complete Verb & i-Adjective Conjugation Tables:** For all verbs (`[]` , `[]` , `[]` , `[]`), *suru*-nouns, and `[]` , renders a full 19-row conjugation table (`[]` , `[]` , `[]` , `[]` , `[]` , `[]` , `[]` , `[]` , `[]` , `[]` , `[]` , `[]` , `[]` , `[]` , `[]` , `[]` , `[]` , `[]`) with `<ruby>` furigana and plain kana readings.
 4. **Token-Highlighted Cross-Video Subtitle Occurrences & Corpus Examples:** Lists every subtitle cue across processed videos where the word appears (clicking any occurrence jumps directly to `/ocr/{job_id}?cue={cue_index}`), paired with offline EDRDG Tanaka Corpus Japanese-English example sentences (92,940 indexed sentences). Morphological token and `<ruby>` span matching (`fugashi.Tagger`) lightly highlights the target word or phrase—including conjugated verb and adjective inflections—using `<mark class="ex-hl">` without matching inside `<rt>` furigana tags.
- **Keyboard Shortcuts** (active whenever focus is outside text inputs):
 - `Space` or `K`: Play / Pause video without scrolling the page
 - `A` or `←`: Jump to previous subtitle cue (preserving current play/pause state)
 - `D` or `→`: Jump to next subtitle cue (preserving current play/pause state)
 - `R`: Replay current subtitle cue from its start timestamp
 - `L`: Toggle single-cue A/B loop mode (`Loop: ON/OFF`)
- **Self-Supporting Per-Cue & Job-Wide Local Re-Analysis (`Re-analyze` , `Edit` , and `Auto-Repair Cues`):**
 - **One-Click `Re-analyze` Button (`POST /api/cues/{cue_id}/reanalyze`):** Every subtitle cue card header includes a **Re-analyze** button next to **Edit** that re-runs local Radeon AI PRO R9700 GPU LLM refinement (`qwen2.5:32b` via `ollama-docker-proxy.socket`) on that single cue with surrounding dialogue context (`cue - 1` and `cue + 1`), quote normalization (`"..."` -> `□...□`), and output sanity

validation (`is_valid_llm_ja`), updating the card, furigana, vocabulary chips, and pre-generated vocabulary pages in place in ~1.5 seconds.

- **Inline Cue Editor & Targeted Recovery Modes:** Clicking **Edit** on any cue card opens an inline editor for Japanese text, Furigana reading, English meaning, and vocabulary notes, with four built-in actions:
 1. **Reset to Raw OCR** (`mode="raw"`): Instantly restores `ja_raw` and re-runs local `JMdict + JLPT` morphological token and furigana analysis.
 2. **Re-analyze Edited JA (AI)** (`mode="custom_ja"`): Preserves your manually corrected Japanese text and uses the local GPU LLM + `JMdict` to regenerate accurate Furigana, English translation, and vocabulary notes.
 3. **Re-OCR Video + AI** (`mode="reocr"`): Re-samples the video frames across `[start_sec - 0.2, end_sec + 0.2]` at a dense `4.0 fps` on the GPU (`PP-OCrv5`) and runs local LLM refinement.
 4. **Save Cue:** Instantly (`< 20 ms`) saves manual field edits and automatically re-runs morphological tokenization (`analyze_cue_tokens`), Kanji-only `<ruby>` furigana generation, symbol-compound recognition (`ox`, `ox`, `x□`), and background vocabulary HTML page regeneration whenever `ja_text` is modified.
- **Job-Wide Auto-Repair Cues** (`POST /api/jobs/{job_id}/repair-cues`): Located in the left control panel, scans all cues in the current video for any truncated or invalid LLM outputs and automatically re-analyzes every affected cue on the local GPU.

4. Operations, Maintenance & Backup / Restore

Container Management

Click to expand Rootless Podman Quadlet & Service Management Commands

```
# Check container status and logs
$ (cd /tmp && sudo -u docker-data HOME=/home/docker-data XDG_RUNTIME_DIR=/run/user/1100
podman ps --filter name=subocr)
$ sudo -u docker-data XDG_RUNTIME_DIR=/run/user/1100 journalctl --user -u subocr.service -
n 100 -f
```

```
# Restart after modifying /data/docker/subocr/app.py or vocab_pages.py
$ sudo systemctl --user -M docker-data@ restart subocr.service

# Rebuild container image if Dockerfile dependencies change
$ (cd /data/docker/subocr && sudo -u docker-data HOME=/home/docker-data
XDG_RUNTIME_DIR=/run/user/1100 TMPDIR=/dev/shm podman build -t localhost/subocr:latest
/data/docker/subocr)
$ sudo systemctl --user -M docker-data@ restart subocr.service
```

Automated System Backup & Restore

Backup Integration: SubOCR is integrated into the standard host `ul` backup and restore suite under `/home/luke/git/backups/ul/scripts/` and runs automatically during `sudo /home/luke/git/backups/ul/scripts/run_all_backups`.

- **Backup Script** (`/home/luke/git/backups/ul/scripts/backup/subocr`):

1. Rotates any existing backup in `/data/backups/services/current/subocr/` to `/data/backups/services/archive/subocr/`.
2. Refreshes `/data/docker/subocr/data/video_sources.json` (recording every job's ID, title, category, `youtube_url`, `source_path`, and `host_path`) and executes a non-blocking SQLite WAL checkpoint (`PRAGMA wal_checkpoint(TRUNCATE);`) on `/data/docker/subocr/data/subocr.db`.
3. Creates a compressed archive (`subocr_data.tar.xz`, ~1.4 MB) of `/data/docker/subocr/` including `app.py`, `vocab_pages.py`, `Dockerfile`, `docker-compose.yml`, `static/`, `data/subocr.db`, `data/video_sources.json`, **all pre-generated .html vocabulary/grammar pages** (`data/vocab_pages/**/*.html`), and all exported job subtitle/study files (`data/jobs/**/*.srt`, `.vtt`, `.anki.tsv`, `.study.md`), while excluding transient model/kernel caches (`cache/`), the static JMdict reference database (`data/jmdict_jlpt.db`), and large raw video files (`*.mp4`, `*.mkv`, `*.webm`, `*.mov`).
4. Generates 10% Reed-Solomon parity protection archives (`subocr_data.tar.xz.par2`) via `par2 create -r10`.

- **Restore Script** (`/home/luke/git/backups/ul/scripts/restore/subocr`):

1. Verifies archive integrity against the `par2` parity data (`par2 verify`).
2. Extracts `subocr_data.tar.xz` into `/data/docker/subocr/`, restoring the SQLite database, all pre-generated vocabulary HTML pages, and all subtitle exports.
3. Restarts the `subocr` container and checks all jobs in `subocr.db`: if any job's video file is missing on disk and has a recorded `youtube_url`, automatically re-downloads the video via `yt-dlp`.

```
# Run a standalone SubOCR backup
$ sudo /home/luke/git/backups/ul/scripts/backup/subocr

# Restore from the latest backup in /data/backups/services/current/subocr/
$ sudo /home/luke/git/backups/ul/scripts/restore/subocr

# Restore from a specific timestamped backup
$ sudo /home/luke/git/backups/ul/scripts/restore/subocr 2026-10-03_1133.09
```

Revision #16

Created 2026-10-03 12:40:38 UTC by luKe

Updated 2026-10-20 03:46:13 UTC by luKe