Perspecta: A Minimalist, Launch-Driven Desktop DICOM Viewer for Targeted Review

Timothy C. Cogan
Independent Researcher
tim@cogan.dev

Abstract

Perspecta is a native desktop DICOM viewer implemented in Rust (using `egui/eframe`), designed to reduce launch overhead when opening and interacting with a specific image, report, or small mammography image group (e.g., 1×2, 1×3, 2×2, or 2×4 layouts). Perspecta is positioned for targeted-review environments that value low-overhead launch, simple integration via a custom URL scheme, and a compact set of high-utility interactions including overlays and measurement. The interface is intentionally minimalist, avoiding button-heavy UI chrome and prioritizing screen real estate for image display. In preliminary local synthetic benchmarks, Perspecta achieved median end-to-end launch latencies of 139 ms for a single image and 221 ms for an eight-image group on a commodity x86_64 system. This paper describes the motivation, core design decisions, implementation architecture, and differentiation of Perspecta from established open-source viewers. Rather than replacing full PACS workflows, Perspecta is designed as a focused complement for deterministic targeted review. Perspecta is a research/open-source prototype and is not cleared, certified, or validated for diagnostic clinical use.

1 Introduction

Many imaging workflows need to open a specific DICOM instance [1] or a small, known set of images with low setup overhead — speed is important to not only time-pressed clinicians but also other professionals (e.g., artificial intelligence researchers) working with medical images. In these cases, broad viewers can introduce setup overhead when their strengths are broad modality coverage, complex navigation, and extensive tooling delivered through a control-heavy user-interface. Perspecta targets low-overhead deterministic opening and immediate interaction.

This paper presents Perspecta as a minimal, launch-driven viewer with a clean interaction surface. It describes the core functionality, how the launch system works (CLI and `perspecta://`), and why a focused scope can be beneficial when integrating DICOM viewing into larger systems.

Perspecta is implemented as a native desktop application in Rust [2] (using `egui/eframe` [3, 4]) and is available as open-source software.[5] This implementation choice is aligned with the project's primary objective of low time-to-image and predictable interaction in targeted review workflows. In this paper, the Rust-based implementation is treated primarily as an architectural decision that supports low launch overhead and a compact runtime footprint, rather than as a blanket claim of superior performance across all viewing scenarios.

Contributions This paper makes four practical contributions:

- A focused problem framing for targeted DICOM review, emphasizing deterministic launch, immediate interaction, and grouped-image workflows.

- A single explicit launch contract (CLI and `perspecta://`) for single-image, grouped-image, and multi-group review workflows.
- An implementation approach for low-overhead rendering and retrieval, including active-group-first loading and background preload of additional groups.
- A workflow-oriented comparison that positions Perspecta as a complement to broader open-source viewers rather than a feature-for-feature replacement.

2 Motivation and Design Goals

Perspecta is shaped by three design goals:

- **Low launch overhead.** Open a single image, structured report, or grouped mammography layout with minimal overhead.
- **Deterministic launch.** Provide a single, explicit launch contract (CLI and URL scheme) that specifies the target image or image group and initial selection, avoiding multi-step viewer setup in common targeted-review workflows.
- **Focused interactions.** Deliver a compact set of high-utility image controls while minimizing UI chrome to prioritize image viewport real estate.

3 System Overview

Perspecta is a native, cross-platform desktop application built in Rust [2] using the `egui/eframe` GUI framework [3, 4]. The viewer exposes three primary review states:

- **Single-image view** for one DICOM instance.
- **Grouped multi-image view** for supported mammography groups of 2, 3, 4, or 8 displayable images (1×2, 1×3, 2×2, and 2×4 layouts). Mammography images are ordered automatically by laterality, view, and study context when available.
- **Report/object view** for Structured Report objects and standalone Parametric Map objects when they are opened directly rather than attached as overlays.

Core interactions include:

- Cine playback for multi-frame images, including synchronized cine across grouped multi-image views.
- Window/level adjustment for monochrome images.
- Zoom with mouse wheel and pan by drag.
- Overlay display and navigation for matching GSPS, Mammography CAD SR, and Parametric Map objects.
- Live distance measurement with DICOM pixel spacing when available, falling back to pixel units otherwise.
- Metadata overlay for quick tag inspection, with a full-field metadata popup for the active object.

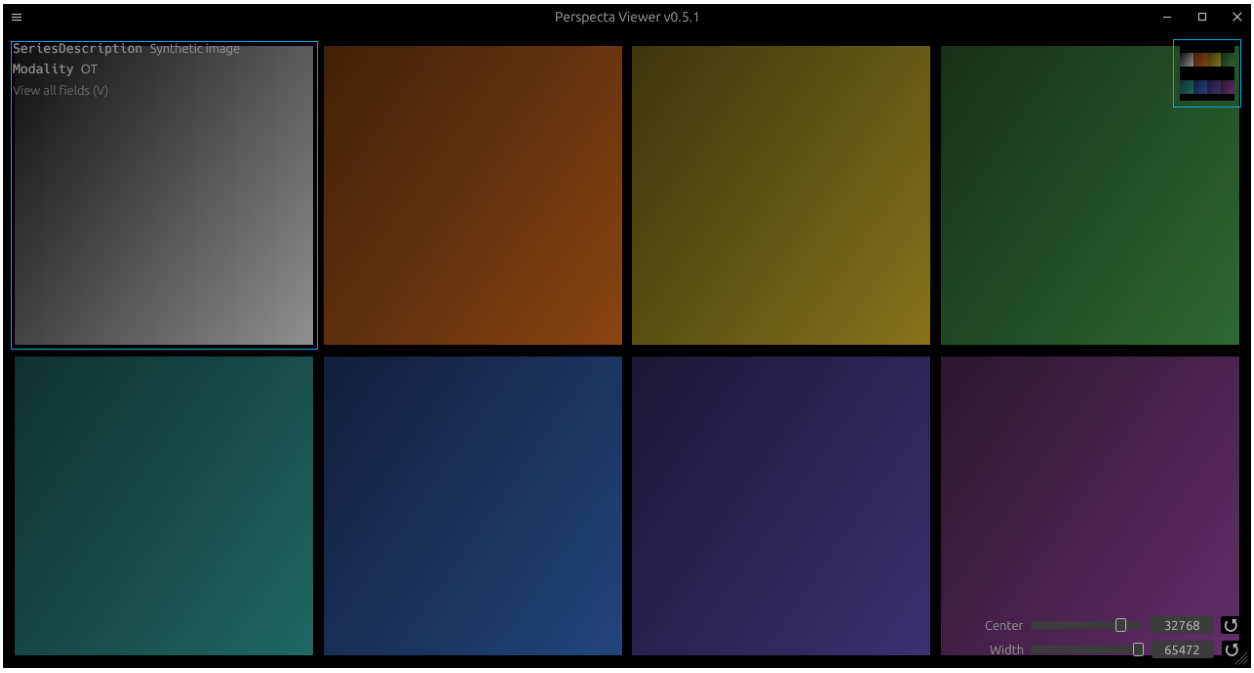

Figure 1: Perspecta opened in a 2×4 grouped layout using generic 1024×1024 synthetic, non-patient DICOM inputs: one grayscale image and seven rainbow-tinted companions.

4 Launch and Integration

Perspecta supports deterministic opening through two mechanisms:

- **CLI** with either one path (single view) or a grouped launch for supported mammography layouts.
- **Custom URL scheme** `perspecta://` that accepts file paths, grouped paths, and DICOMweb parameters.[6]

In this paper, an *image group* denotes one or more displayable images opened as a single review target, and a *multi-group review set* denotes multiple such groups bundled into one launch request with an explicit initial group selection. Supplementary DICOM objects such as GSPS, Structured Reports, Mammography CAD SR, and Parametric Maps can be included with image launches when they are intended to annotate, describe, or augment the selected image content.

Figure 2 summarizes the launch contract as an integration boundary: the external system encodes the target object or image group and the initial selection once, while Perspecta resolves sources, renders the active group, and queues remaining groups without follow-up layout messages.

The URL format allows an external system to select which group opens first, include additional groups in the same launch request, and launch directly from DICOMweb by DICOM study/series/instance identifiers. This makes Perspecta suitable as a focused viewer component for web or desktop systems that want a predictable, explicit launch contract. For example, Perspecta can complement a web-based study list (e.g., OHIF) by opening and visualizing target images while leaving study-list navigation and workflow control to the external system.[7] In integrations where recreating a specific multi-image arrangement involves multiple steps (e.g., import/open actions, layout selection, and manual assignment), Perspecta prioritizes a one-step launch for the intended target image or image group. For mammography workflows, the same contract can encode a multi-

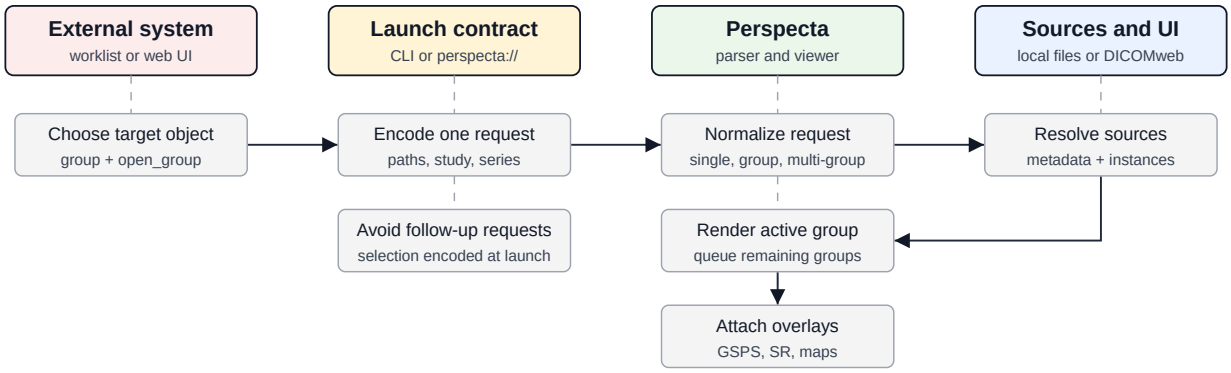

Figure 2: Deterministic launch ladder. The target object/group and initial selection are encoded in a single CLI or `perspecta://` request, reducing cross-process coordination between a worklist or external system and the viewer.

Pattern	**Contract form**	**Purpose**
CLI single target	path	Open one local DICOM instance directly.
CLI grouped target	2, 3, 4, or 8 positional paths	Open a supported local mammography layout (1×2, 1×3, 2×2, or 2×4).
URI local multi-group	`group=... & group=... & open_group=k`	Include multiple local groups, attach supplementary objects where applicable, and choose the initial group explicitly.
URI DICOMweb single	`dicomweb=... & study=...`	Launch from DICOMweb by identifiers.
URI DICOMweb multi-group	`dicomweb=... & study=... & group_series=...`	Build multi-group review sets from DICOMweb series selections, with active-group streaming.

Table 1: Representative deterministic launch patterns supported by Perspecta.

group review set in one launch (e.g., a digital breast tomosynthesis group, a synthetic-view group, and supplementary report or overlay objects) while explicitly selecting which group opens first. For mammography layouts, Perspecta also applies automatic ordering based on laterality, view labels, and study date when applicable. Table 1 summarizes representative deterministic launch patterns.

5 Implementation Highlights

Perspecta is designed to reduce user-visible launch overhead. Figure 3 summarizes the launch-to-render pipeline and where deterministic launch semantics connect to active-group-first loading and background preload behavior.

Key implementation choices include:

- Immediate decode of frame 0 and lazy background preload for multi-frame datasets.
- Object-aware launch preparation that separates displayable images from GSPS, SR/CAD SR, and Parametric Map objects.
- Separately optimized render paths for monochrome and RGB images.
- Simple history cache to allow fast revisiting of recently opened items.

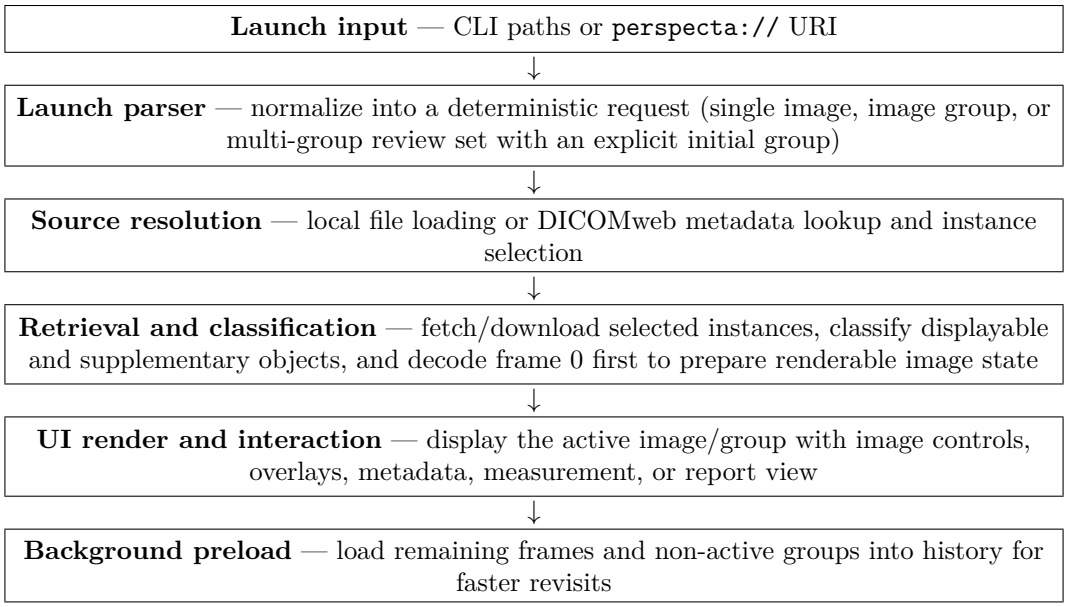

Figure 3: High-level Perspecta launch-to-render pipeline. Deterministic launch parameters specify the initial target, while background preload improves perceived responsiveness for grouped workflows.

For DICOMweb,[6] Perspecta retrieves metadata, selects instances, and downloads them for local rendering. For grouped launches, the active group can be streamed first for immediate visualization while other groups download in parallel. For local launches, preparation runs separately from the UI event loop so that object classification and supplementary overlay attachment do not block interaction. Matching supplementary objects are attached to the corresponding image when references are available: GSPS graphics provide presentation-state overlays, Mammography CAD SR objects can add vector marks and concise finding labels, and Parametric Maps can be rendered as heatmap overlays or opened as standalone images. Structured Report objects can also open in a dedicated text/document view. Perspecta also supports synchronized cine playback across all viewports in a grouped multi-image view (e.g., mammography/DBT layouts), enabling simultaneous frame navigation during targeted review. In other viewers, equivalent multi-viewport cine behavior may require additional layout or synchronization setup, with frame pacing depending on the hardware and software stack. This motivates explicit cine benchmarking in future comparative evaluations.

6 Preliminary Launch Benchmark

As an initial validation of the launch path, Perspecta's end-to-end benchmark harness was run under an Xvfb-backed headless display session on a prebuilt release binary using synthetic local DICOM inputs. The benchmark launches the full application and records user-visible launch latency from process start to first completed UI/render state. The preliminary runs used a commodity x86_64 PC with a 6-core/12-thread CPU and 16 GB RAM, running Ubuntu 24.04.4 LTS; an Xvfb display session; 1024×1024 synthetic viewport images; and 20 measured runs after two warmup runs per scenario. The repository benchmark target builds Perspecta and the benchmark harness with `cargo build --release`; the repository toolchain pins Rust 1.95.0. The reported runs used the following commands:

Scenario	Total	Startup	DICOM load	Render/UI
Local synthetic single image	139 (137–142)	122 (120–123)	17 (15–18)	1 (0–2)
Local synthetic 8-up group	221 (216–226)	127 (124–129)	92 (89–95)	3 (0–4)

Table 2: Preliminary local launch benchmark in ms. Values are median (minimum–maximum) over 20 measured runs after two warmup runs.

Project	Platform / deployment	Primary language(s)	License
Perspecta (this work)	Desktop (cross-platform)	Rust	MIT (permissive)
OHIF Viewer[8, 7]	Web (browser-based)	TypeScript / JavaScript	MIT (permissive)
DWV[9]	Web (browser-based)	JavaScript	GPL-3.0 (strong copyleft)
Weasis[10, 11, 12, 13, 14]	Desktop (cross-platform)	Java	EPL-2.0 or Apache-2.0 (dual license)
3D Slicer[15, 16, 17, 18]	Desktop (cross-platform)	C++ / Python	3D Slicer License (BSD-style, custom)
Horos[19]	Desktop (macOS)	C++ / Objective-C / Objective-C++	LGPL-3.0 (weak copyleft)
Ginkgo CADx[20]	Desktop (cross-platform, archived)	C++ / C	LGPL-3.0 (weak copyleft)

Table 3: Open-source viewer comparison focused on platform/deployment style, implementation language, and license.

```
make benchmark BENCH_IMAGES=1 BENCH_RUNS=20 BENCH_WARMUP=2 \
  BENCH_ROWS=1024 BENCH_COLS=1024 BENCH_TIMEOUT_SECS=15
make benchmark BENCH_IMAGES=8 BENCH_RUNS=20 BENCH_WARMUP=2 \
  BENCH_ROWS=1024 BENCH_COLS=1024 BENCH_TIMEOUT_SECS=15
```

The harness is implemented in `tools/benchmark/src/bin/benchmark_open.rs` and generates Explicit VR Little Endian monochrome mammography objects with 16-bit 1024×1024 pixel data, approximately 2 MiB of pixel data per image. Table 2 reports timing components as median with minimum and maximum observed values.

These measurements are preliminary and are intended only to ground the single-image and grouped local launch paths. Additional scenarios, including DICOMweb launch, cache-state controls, and cine playback, remain future evaluation work. System vendor and product-model details are intentionally omitted because these measurements are not presented as cross-system performance benchmarks.

7 Related Work (Open-Source)

Several open-source viewers listed in Table 3 address complementary clinical and research workflows. This comparison highlights that Perspecta differs not only in workflow scope but also in distribution and integration. The listed projects span permissive licenses (e.g., MIT), weak-copyleft licenses (e.g., LGPL, EPL), and custom BSD-style terms. Although licensing differences are orthogonal to feature breadth, they can matter for downstream integration and redistribution decisions.

Perspecta differs by limiting scope to deterministic targeted opening and a narrow, high-value interaction set. This simplifies integration in systems that only need to open explicit targets rather than browse or manage full studies. In platform terms, Perspecta is intended for cross-platform desktop deployment, unlike Horos, which is specific to macOS.[19]

8 Discussion

A key design benefit of Perspecta is its integration model — a stable, explicit launch contract can enable external systems to open a single image, report, or supported mammography group with minimal cross-process synchronization requirements. In targeted-review scenarios, this can reduce multi-step setup to produce a specific image arrangement. For example, when a worklist UI and a desktop viewer run as separate processes (as in many integrations that launch a general-purpose viewer such as Weasis), keeping selection, layout, and synchronized viewing state aligned across process boundaries can add integration complexity. Encoding the target object/group and initial selection in a single launch command reduces the coordination burden and integration risk. The result is a small, predictable application, suitable for rapid review and embedded workflows.

This design contrasts with broader open-source viewers such as Weasis (Java-based desktop) and OHIF (web/browser-based), which target wider clinical and research workflows and operate under different runtime constraints.[10, 8] Perspecta's current scope is narrower — deterministic launch of a specific image, report, or grouped image set with a small set of high-utility interactions. For that reason, the comparison emphasizes launch semantics, integration model, and workflow focus more than raw feature breadth.

A direct performance comparison with other viewers is an important future step which requires controlled benchmarking (e.g., cold-start and warm-start time-to-first-image, cache state, network conditions, hardware, and dataset characteristics). The native Rust implementation is therefore framed as a design choice motivated by low-overhead execution and deployment simplicity, while reserving stronger comparative performance claims for a dedicated empirical evaluation. Perspecta includes an end-to-end benchmark harness for local single-image and 2×4 launch scenarios, but this tooling is not a substitute for a controlled comparative study.

9 Limitations

A practical limitation (at the time of writing) is ecosystem maturity. While Rust provides a strong systems-programming foundation, the medical imaging/viewer ecosystem in Rust is less mature than long-established C/C++ stacks in some areas for this particular domain (e.g., specialized codecs, legacy integrations, and tooling). This can increase implementation and maintenance effort for production deployments, and may result in short-term performance deficiencies.

Perspecta's launch-driven integration model also introduces operational security and privacy considerations. In particular, `perspecta://` URIs and integration logs may contain sensitive local file paths, DICOM identifiers, or (depending on deployment choices) authentication credentials. Deployments should therefore treat launch URIs, process arguments, and temporary DICOMweb cache contents as sensitive artifacts and apply appropriate access controls and retention policies.

10 Future Work

The preliminary benchmark should be expanded into a controlled evaluation centered on user-visible latency and launch determinism. The most relevant primary metric is image group loading time reported under both cold-start and warm-start conditions.

A reproducible evaluation should report at least:

- **Scenarios** — local / DICOMweb single-image, group, and multi-group launch.
- **Timing endpoints** — process launch start, first image visible, first interaction-ready state, and background groups complete.

- **Cine performance** — effective frames per second, dropped-frame rate, and frame pacing jitter for both single-viewport and synchronized multi-viewport cine.
- **Environment controls** — hardware, operating system, dataset characteristics, cache state, network conditions, and codec/runtime configuration.
- **Reporting** — multiple runs per scenario with median and percentile ranges.

A second future direction is a WebAssembly-based web deployment of Perspecta. Because Perspecta is implemented in Rust, a WASM target could reuse selected core viewer logic while enabling browser-based distribution for environments that prefer zero-install access. Such a version should be evaluated separately, since browser sandboxing, DICOMweb authentication, local-file access, GPU acceleration, and cache behavior impose different constraints than seen in the native desktop viewer.

In comparing with other viewers, benchmarks should emphasize matched workflow tasks such as one-step opening of a specific target layout. Validity threats include network variability, cache warmness, and codec differences across software stacks.

11 Conclusion

This paper has presented a DICOM viewer that is complementary to existing solutions, providing design choices that may be preferable in targeted-review environments where launch determinism and simplicity are priorities. Perspecta demonstrates how a minimalist, launch-driven viewer can support low-overhead, predictable access to DICOM imagery and integrate cleanly with external systems such as PACS systems and worklists. Focusing on a strategic set of essential interactions and deterministic launch semantics, Perspecta provides a practical complement to other open-source viewers, recognizing both the value of existing solutions and difficulty of developing one-size-fits-all viewers.

References

[1] National Electrical Manufacturers Association. NEMA PS3 / ISO 12052, Digital Imaging and Communications in Medicine (DICOM) Standard. National Electrical Manufacturers Association, Rosslyn, VA, USA. Available at `https://www.dicomstandard.org/`.

[2] Rust Project Developers. The Rust Programming Language. `https://www.rust-lang.org/`, 2026. Project website.

[3] egui project. egui (GitHub). `https://github.com/emilk/egui`, 2026. Software repository.

[4] egui project. eframe crate documentation. `https://docs.rs/eframe/latest/eframe/`, 2026. Documentation.

[5] Timothy Cogan. Perspecta (GitHub). `https://github.com/timcogan/perspecta`, 2026. Software repository.

[6] DICOM Standards Committee. DICOM PS3.18: Web Services. NEMA PS3 / ISO 12052, Digital Imaging and Communications in Medicine (DICOM) Standard. Available at `https://www.dicomstandard.org/`.

[7] Open Health Imaging Foundation. OHIF Viewer (GitHub). `https://github.com/OHIF/Viewers`, 2026. Software repository.

[8] Erik Ziegler, Trinity Urban, Danny Brown, James Petts, Steve D Pieper, Rob Lewis, Chris Hafey, and Gordon J Harris. Open health imaging foundation viewer: an extensible open-source framework for building web-based imaging applications to support cancer research. *JCO clinical cancer informatics*, 4:336–345, 2020.

[9] DWV project. DWV (GitHub). `https://github.com/ivmartel/dwv`, 2026. Software repository.

[10] Nicolas Roduit, Francis Klumb, David Bandon, Antoine Geissbuhler, and Osman Ratib. WEASIS: An Open Source Web-based Viewer Aimed for Telemedicine and General Practitioners. In *Radiological Society of North America 2010 Scientific Assembly and Annual Meeting*, Chicago, IL, 2010. Abstract archive.

[11] Weasis. Weasis Documentation: DICOM Import (Query/Retrieve and DICOMWeb nodes). `https://weasis.org/en/tutorials/dicom-import/index.html`, 2026. Documentation.

[12] Weasis. Weasis Documentation: DICOMWeb Import. `https://weasis.org/en/tutorials/dicomweb-config/`, 2026. Documentation.

[13] Weasis. Weasis Documentation: MPR Viewer. `https://weasis.org/en/tutorials/mpr/`, 2026. Documentation.

[14] Weasis Team. Weasis (GitHub). `https://github.com/nroduit/Weasis`, 2026. Software repository.

[15] Andriy Fedorov, Reinhard Beichel, Jayashree Kalpathy-Cramer, Julien Finet, Jean-Christophe Fillion-Robin, Sonia Pujol, Christian Bauer, Dominique Jennings, Fiona Fennessy, Milan Sonka, et al. 3D Slicer as an image computing platform for the Quantitative Imaging Network. *Magnetic resonance imaging*, 30(9):1323–1341, 2012.

[16] 3D Slicer. 3D Slicer Documentation: DICOM Module and Networking. `https://slicer.readthedocs.io/en/latest/user_guide/modules/dicom.html`, 2026. Documentation.

[17] 3D Slicer. Slicer/Slicer (GitHub). `https://github.com/Slicer/Slicer`, 2026. Software repository.

[18] 3D Slicer. 3D Slicer Documentation: About 3D Slicer. `https://slicer.readthedocs.io/en/latest/user_guide/about.html`, 2026. Documentation.

[19] Horos Project. Horos (GitHub). `https://github.com/horosproject/horos`, 2026. Software repository.

[20] Ginkgo CADx. Ginkgo CADx (archived) (GitHub). `https://github.com/gerddie/ginkgocadx`, 2026. Software repository.