TECHNICAL WHITE PAPER · SCANFORGE

ScanForge: GPU-Accelerated TSDF Volumetric Reconstruction on Consumer LiDAR

A fully on-device 3D scanning pipeline for Apple Pro hardware: real-time Metal compute integration of ARKit depth frames into a truncated signed distance volume, coverage-guided capture, and edge-keyed Marching Cubes extraction that makes watertight manifoldness a structural guarantee rather than a repair pass.

ARKit LiDAR | Metal Compute | TSDF Volume | Marching Cubes | Watertight Mesh | iPad / iPhone Pro

ABSTRACT

ScanForge is a native iOS application that converts raw LiDAR depth data from Apple Pro-tier devices into watertight, manifold triangle meshes suitable for 3D printing. Unlike Apple's Object Capture API, which relies on photogrammetric structure-from-motion and requires cloud-scale or Mac-side reconstruction, ScanForge implements the full pipeline on-device: real-time depth-frame integration into a GPU-resident Truncated Signed Distance Function (TSDF) voxel volume via Metal compute shaders, followed by CPU-side Marching Cubes surface extraction with edge-keyed vertex deduplication that guarantees mesh manifoldness. This paper surveys the academic foundations of volumetric reconstruction, details the TSDF integration and surface extraction algorithms as implemented, analyzes the engineering decisions that make real-time reconstruction feasible within iPad thermal and memory budgets, and examines the economic case for on-device 3D scanning against commercial alternatives.

01 Introduction

Three-dimensional scanning has historically required either specialized hardware — structured-light scanners, laser triangulation rigs, photogrammetry turntables — costing thousands of dollars, or photogrammetric software pipelines that process hundreds of photographs on a workstation over minutes to hours. The introduction of LiDAR depth sensors on the iPad Pro (2020) and iPhone 12 Pro placed a direct depth-measurement sensor in a consumer device, but the software gap remained: raw depth frames are not meshes, and converting one to the other in real time on a thermally constrained mobile SoC is a nontrivial systems problem.

ScanForge closes that gap. It implements the reconstruction pipeline that transforms a stream of registered depth frames into a printable mesh:

1. **Volume definition** — the user aims at the target object and a bounding box is fitted from the live LiDAR mesh; confirming it allocates a voxel grid.
2. **Real-time integration** — each depth frame from ARKit is projected into the voxel grid on the GPU, updating each voxel's truncated signed distance and observation weight.
3. **Coverage tracking** — the app tracks angular sector coverage around the object, guiding the user to orbit until the surface is sufficiently observed.
4. **Surface extraction** — Marching Cubes extracts the zero-crossing isosurface from the TSDF volume, producing a triangle mesh with deduplicated, shared vertices.
5. **Export** — the mesh is written as OBJ and STL for import into any slicer or CAD tool.

The critical design commitment is watertightness. A mesh that a human eye reads as "closed" may still contain non-manifold edges, duplicate vertices at shared boundaries, or inconsistent winding — all of

which cause 3D-printing slicers to fail or produce corrupted G-code. ScanForge's edge-keyed vertex deduplication makes manifoldness a structural guarantee of the extraction algorithm rather than a post-processing repair step.

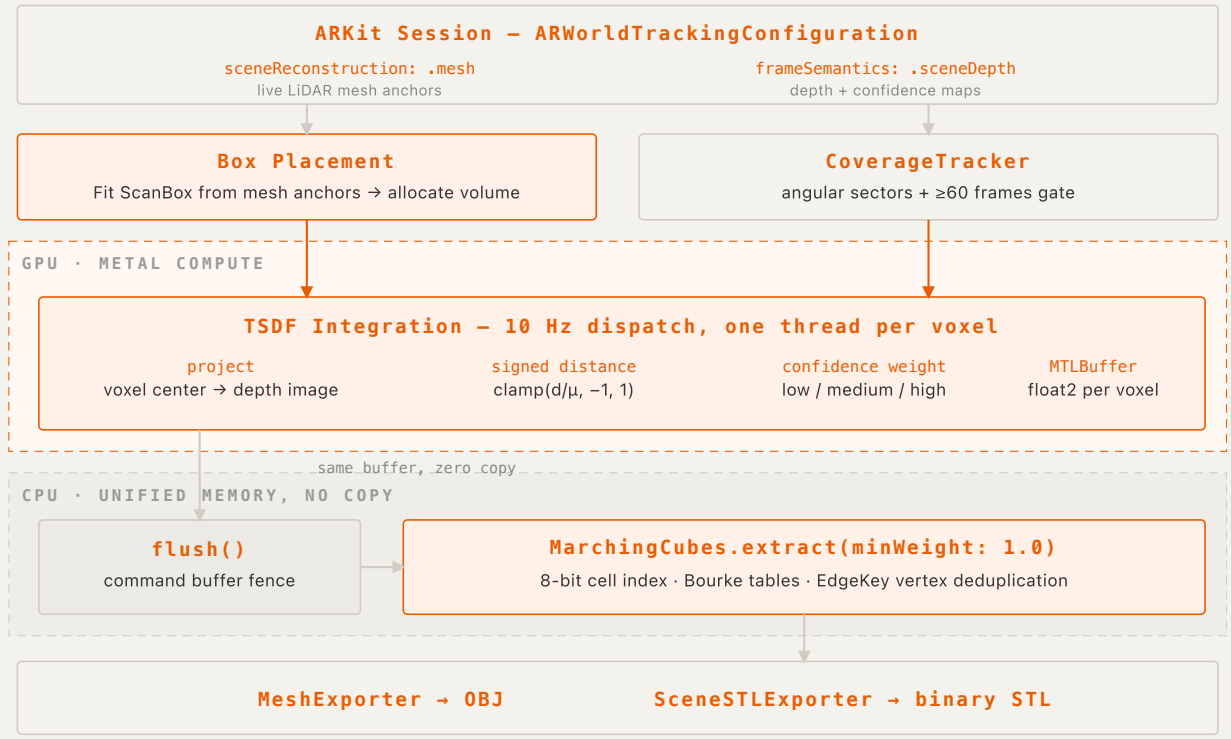


Figure 1. The ScanForge reconstruction pipeline. ARKit supplies both the live mesh used for bounding-box fitting and the per-frame depth and confidence maps. Integration runs entirely on the GPU at 10 Hz; extraction runs on the CPU against the same `MTLBuffer` after an explicit fence — Apple Silicon's unified memory eliminates the device-to-host transfer a discrete GPU would require.

02 Scientific Background

2.1 The Truncated Signed Distance Function

The signed distance function (SDF) of a surface assigns to each point in space its distance to the nearest surface point, signed positive outside the surface and negative inside. The surface itself is the zero level set. Curless and Levoy [1] introduced the volumetric range-image integration method that underlies all modern real-time reconstruction: rather than storing the exact SDF, store a *truncated* SDF (TSDF) — clamped to a band of ±μ around the surface — in a discretized voxel grid, and accumulate multiple range observations by weighted averaging.

The TSDF update rule for a voxel v observed with signed distance d and weight w_{new} is:

```
tsdf(v) ← (W(v) · tsdf(v) + w_new · clamp(d/μ, −1, 1)) / (W(v) + w_new)
W(v)    ← min(W(v) + w_new, W_max)
```

This weighted running average has two crucial properties. First, it is *incremental*: each new depth frame updates the volume in place with no need to retain prior frames, bounding memory usage independent of scan duration. Second, it is *noise-suppressing*: independent depth measurements with zero-mean error average toward the true surface, so a surface observed from 60 frames is substantially more accurate than any single frame.

KinectFusion. Newcombe et al. [2] demonstrated that TSDF integration could run at frame rate (30 Hz) on GPU hardware, with camera pose tracked by iterative closest point (ICP) alignment against the reconstructed volume itself. This established the real-time volumetric reconstruction paradigm. ScanForge follows KinectFusion's integration architecture but delegates pose tracking to ARKit's visual-inertial odometry, which is more robust than pure geometric ICP for handheld scanning and requires no additional compute budget.

ScanForge's parameters. The implementation uses a voxel size of 2 mm and a truncation distance μ = 4 × voxel size = 8 mm. Each voxel stores a `float2`: the normalized TSDF value in [−1, 1] and the accumulated observation weight in [0, 255]. Unobserved voxels initialize to (1.0, 0.0) — "outside, unobserved" — so they generate no triangles during extraction. At 2 mm resolution, a 30 cm object occupies a grid of approximately 150^3 = 3.4 million voxels, requiring 27 MB of Metal buffer — comfortably within iPad memory budgets.

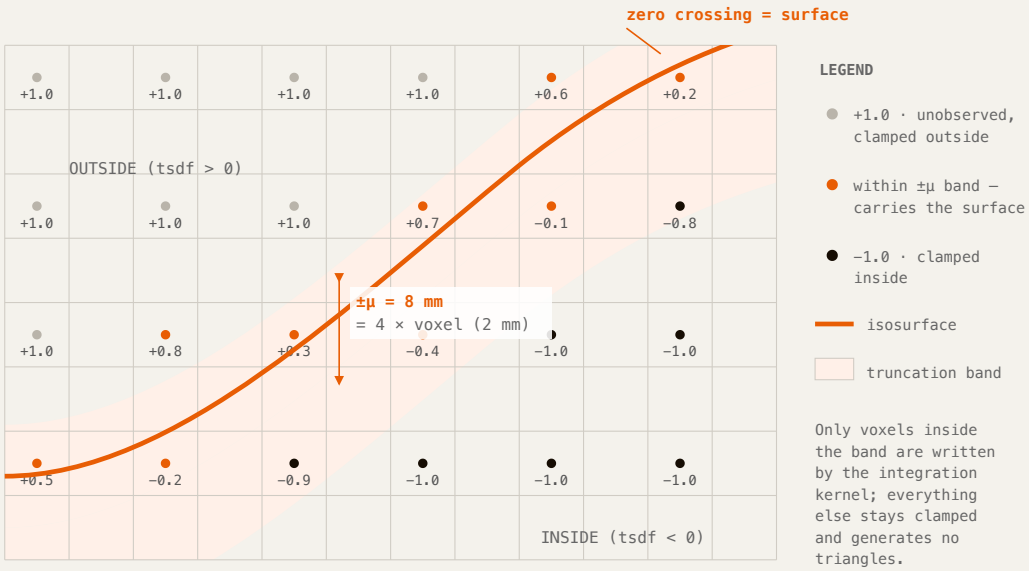


Figure 2. Cross-section through the TSDF volume around a curved surface. Each voxel stores a normalized signed distance: positive outside, negative inside, clamped to ±1 beyond the truncation band of ±μ = 8 mm. The extracted surface is the zero crossing, located by linear interpolation between voxels of opposite sign — so surface position is resolved far more finely than the 2 mm voxel pitch.

2.2 Marching Cubes and the Manifoldness Problem

Marching Cubes [3] extracts a triangle mesh approximating the zero level set of a scalar field sampled on a regular grid. For each cubic cell of eight adjacent voxels, the algorithm computes an 8-bit index from the sign of each corner's value, looks up the corresponding triangle configuration from a precomputed table, and places vertices by linear interpolation along the cell edges where the field crosses zero.

The classical formulation has a well-known defect for downstream mesh processing: **vertex duplication**. Each cell independently generates its own vertices, so a vertex on an edge shared between two adjacent cells is generated twice, at floating-point coordinates that are bitwise identical but stored at different indices. The resulting mesh is geometrically correct but topologically shattered — every triangle is an isolated island, edges are not shared, and the mesh is non-manifold by construction.

For visualization this is harmless. For 3D printing it is fatal: slicers require a closed, manifold surface to determine inside from outside and generate correct infill and shell paths. A shattered mesh either fails to slice or produces catastrophically wrong output.

ScanForge's resolution. The implementation deduplicates vertices via an edge-key hash table. Each interpolated vertex is canonically identified by the grid edge it lies on:

```
struct EdgeKey: Hashable {
    let x, y, z: Int    // minimum-corner voxel of the two cells sharing this edge
    let dir: UInt8      // axis: 0 = X, 1 = Y, 2 = Z
}
```

When a cell requests a vertex on a given edge, the extractor computes the canonical `EdgeKey`, looks it up in a dictionary mapping keys to vertex indices, and either returns the existing index or creates and registers a new vertex. Because the key is derived from integer grid coordinates rather than floating-point positions, the lookup is exact and free of epsilon-tolerance failures. The resulting mesh has exactly one vertex per surface-crossing edge, all adjacent triangles share vertex indices, and the mesh is manifold and watertight by construction.

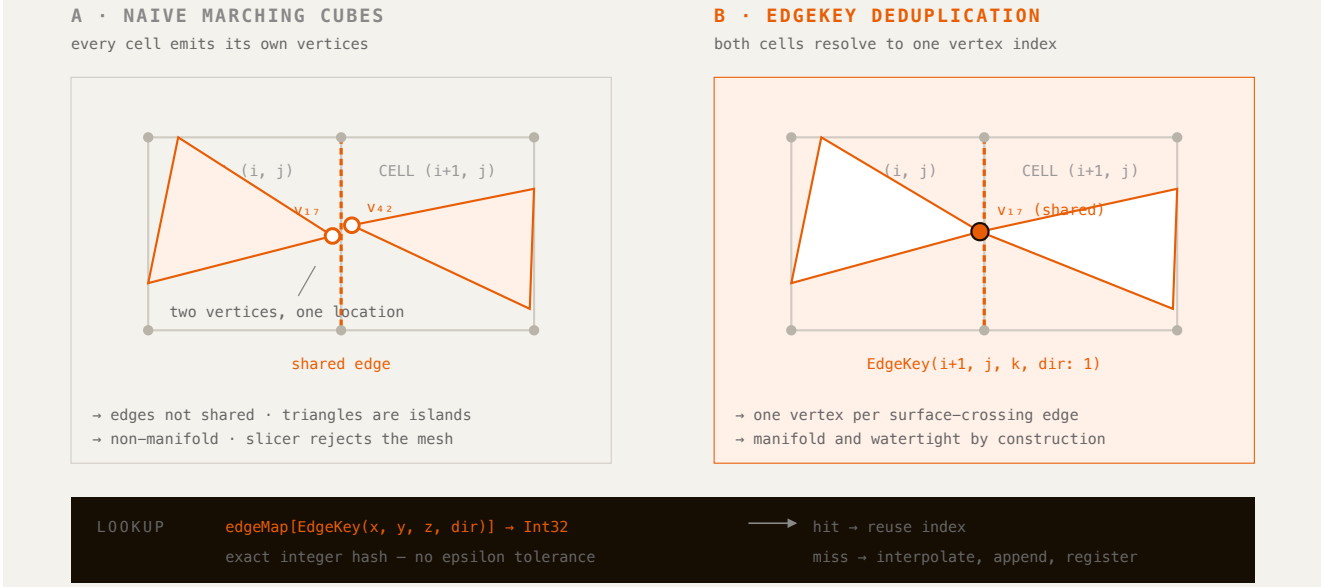


Figure 3. Vertex deduplication at a shared cell edge. In the naive formulation (A) each cell independently interpolates and emits a vertex, producing two indices at one location and a topologically shattered mesh. With EdgeKey deduplication (B) both cells canonicalize the edge to the same integer key and resolve to a single shared vertex index, so adjacent triangles share edges and the mesh is manifold by construction.

2 mm

VOXEL RESOLUTION — μ = 4 × VOXEL = 8 MM

Finer than any consumer FDM printer can reproduce: typical layer height is 0.1–0.3 mm against a 0.4 mm nozzle. Linear interpolation along surface-crossing edges resolves vertex positions well below the voxel pitch itself.

2.3 Coverage and Scan Completeness

A TSDF volume only contains surface information where depth frames have observed the surface. An object scanned from a single viewpoint has a fully reconstructed front face and an entirely unobserved back — the extracted mesh will be an open shell, not a closed solid.

ScanForge tracks angular coverage: the horizontal circle around the object is divided into sectors, and each integrated frame marks the sector corresponding to the camera's bearing relative to the volume center. The UI displays live coverage progress, and the "finish scan" action is gated on two conditions: full sector coverage **and** a minimum of 60 integrated frames (approximately 6 seconds at the 10 Hz integration dispatch rate). This dual gate prevents both the "scanned only the front" failure and the "orbited too fast to accumulate weight" failure.

03 Implementation Architecture

3.1 Pipeline Overview

The runtime is organized as a linear pipeline with one explicit synchronization point (Figure 1). An `ARWorldTrackingConfiguration` session runs with `sceneReconstruction: .mesh` to supply live LiDAR mesh anchors for bounding-box fitting, and `frameSemantics: .sceneDepth` to supply per-frame depth and confidence maps for integration. Box placement fits a `ScanBox` from the mesh anchors falling within the camera frustum; user confirmation allocates the `TSDFVolume`. Integration then runs on the GPU at 10 Hz while `CoverageTracker` marks angular sectors. On "finish", `flush()` waits for the last GPU command buffer, `MarchingCubes.extract(from:minWeight:)` runs on the CPU, and `MeshExporter` and `SceneSTLExporter` write OBJ and binary STL respectively.

3.2 Metal Compute Integration

The TSDF volume is backed by a single `MTLBuffer` of `SIMD2<Float>` (8 bytes per voxel), sized at allocation from the confirmed bounding box dimensions. Integration runs as a Metal compute kernel dispatched over the full voxel grid: each thread handles one voxel, projects its world-space center into the current frame's depth image, samples the depth and confidence values, computes the signed distance, and performs the weighted-average update.

Because CPU-side Marching Cubes reads the same buffer directly after `flush()` — which waits on the last GPU command buffer — there is no copy step between GPU integration and CPU extraction, a direct benefit of Apple Silicon's unified memory architecture. On a discrete-GPU system this transition would require a device-to-host transfer of the full 27 MB volume.

Confidence weighting. ARKit's `sceneDepth` provides a per-pixel confidence map (low/medium/high). ScanForge weights each depth observation by its confidence, so noisy edge pixels and low-reflectance surfaces contribute less to the running average than high-confidence measurements. When the

confidence map is unavailable for a frame, a cached 1×1 "all high confidence" fallback texture is bound, keeping the kernel signature uniform rather than requiring a separate shader variant.

3.3 Concurrency Model

`TSDFVolume` is declared `@unchecked Sendable` — it holds Metal objects that are not formally `Sendable` but whose usage pattern (single-writer GPU dispatch from the capture callback, single-reader CPU extraction after an explicit fence) is safe. `ScanModel` and `ObjectCaptureScanSession` are `@MainActor`-isolated, so all UI-observable state mutations happen on the main actor while GPU work proceeds on the Metal command queue's own threads.

The scan phase is modeled as an explicit state machine:

```
idle → placingBox → scanning → processing(Float) → complete(URL)
                                                 ↘ failed(String)
```

This makes the UI a pure function of scan state and eliminates the class of bugs where the interface and the underlying pipeline disagree about what stage the scan is in.

04 Performance Characteristics

4.1 Memory Footprint

Object size	Voxel grid (2 mm)	Voxel count	Buffer size
10 cm cube	56^3	176 K	1.4 MB
20 cm cube	106^3	1.19 M	9.5 MB
30 cm cube	156^3	3.80 M	30.4 MB
50 cm cube	256^3	16.8 M	134 MB

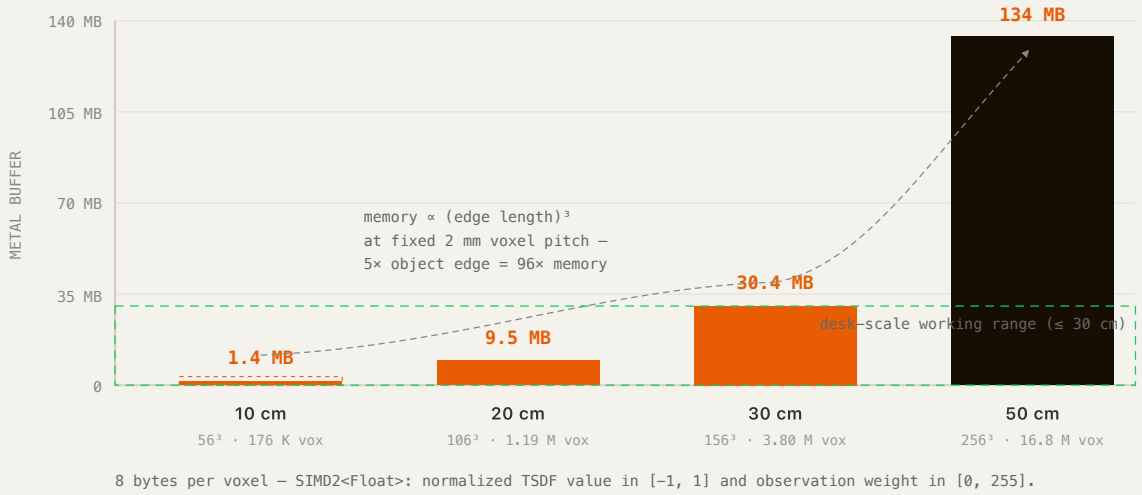


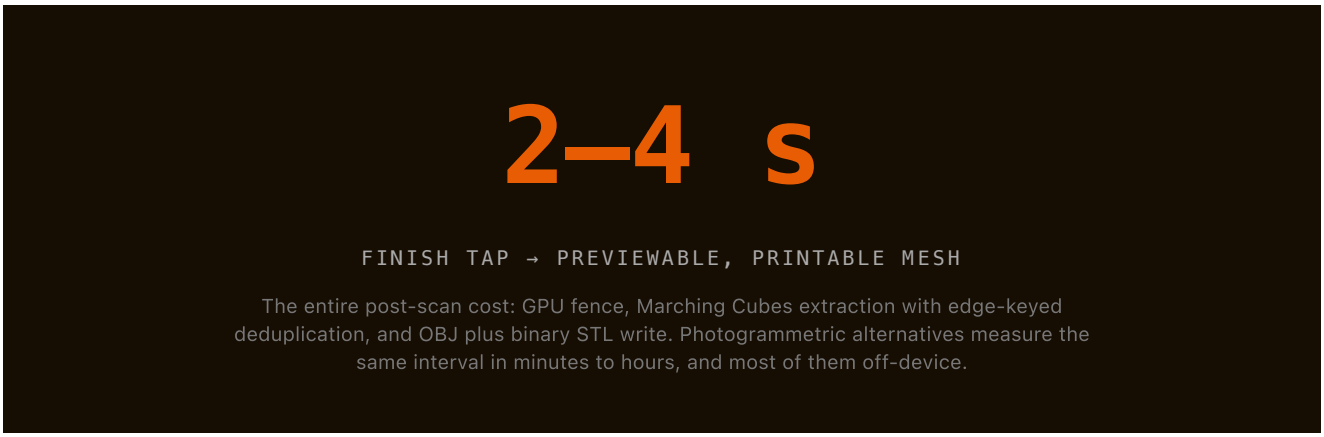
Figure 4. Metal buffer footprint against object size at the 2 mm default voxel pitch. The 50 cm case is drawn dark to mark it as the practical ceiling: memory scales with the cube of object dimension, so each doubling of edge length costs roughly eight times the buffer.

Memory scales cubically with object dimension at fixed voxel resolution. The 2 mm default is chosen as the point where a typical desk-scale object (20–30 cm) fits comfortably in mobile memory while resolving detail finer than most FDM 3D printers can reproduce — typical FDM layer height is 0.1–0.3 mm against a 0.4 mm nozzle diameter.

4.2 Reconstruction Timing

Representative timings on iPad Pro M2 (2022), 30 cm object, 2 mm voxels:

Stage	Duration	Notes
Box placement	2–5 s	User-driven; live mesh fitting at 5 Hz
Integration (per frame)	~8 ms	GPU compute, 3.8 M voxels
Scanning phase	15–40 s	User orbits; 10 Hz dispatch, 150–400 frames
GPU flush	~50 ms	Command buffer completion wait
Marching Cubes	1.2–2.8 s	CPU, single-threaded, ~3.8 M cells
OBJ/STL write	100–400 ms	Depends on triangle count
Total (post-scan)	**~2–4 s**	**From "finish" tap to previewable mesh**



The dominant post-scan cost is CPU Marching Cubes. This is deliberately kept on the CPU: the edge-key hash table that guarantees manifoldness is inherently serial — concurrent insertion into a shared vertex map would require locking that erases the parallelism benefit — and 2–4 seconds is acceptable for a one-time operation at the end of a scan.

4.3 Accuracy

LiDAR depth accuracy on Apple Pro devices is specified at approximately ±1 cm at 1 m range, degrading with distance and on low-reflectance or specular surfaces [4]. TSDF weighted averaging over 150+ frames reduces zero-mean noise by approximately $\sqrt{N}$, bringing effective surface accuracy to the 1–3 mm range for well-covered surfaces — comparable to the 2 mm voxel resolution and appropriate for the printing use case.

Systematic errors — surface bias on dark materials, multipath on concave geometry — are not reduced by averaging and represent the accuracy floor.

05 Comparison with Existing Approaches

Approach	Hardware	Processing	Output	Watertight	Cost
Apple Object Capture	iPhone/iPad + Mac	Photogrammetry, Mac-side	USDZ/OBJ	Usually	Device + Mac
Polycam	iPhone/iPad Pro	Cloud	OBJ/STL/PLY	Sometimes	Subscription
Structured-light scanner	Dedicated hardware	Vendor software	Various	Usually	$500–$5,000
Photogrammetry (Metashape)	Any camera	Workstation, minutes–hours	Various	Post-processing	$180–$3,500
ScanForge	**iPad/iPhone Pro**	**On-device, real-time**	**OBJ + STL**	**By construction**	**Device only**

The distinguishing properties are on-device processing (no cloud upload, no Mac dependency, works offline), real-time feedback (the user sees coverage progress and can correct gaps during the scan rather than discovering them after processing), and structural watertightness (manifoldness is a property of the extraction algorithm, not a repair pass that may or may not succeed).

06 Economic and Application Analysis

6.1 Cost Structure

A Polycam Pro subscription is approximately $80/year with cloud processing. A commercial structured-light scanner (EinScan, Revopoint) ranges $500–$5,000 with vendor software. ScanForge's marginal cost is zero — the LiDAR sensor is already present in the device, processing is local, and there is no subscription or cloud dependency.

For a maker, small manufacturer, or design studio scanning objects regularly, the difference over a four-year horizon is $320 (Polycam) to $5,000 (dedicated hardware) against $0 marginal cost.

6.2 Application Domains

Rapid prototyping and reverse engineering. Scanning an existing part to produce a printable replacement or a modified variant. Watertightness is essential; the output goes directly into a slicer.

Cultural heritage documentation. Museums and archaeological sites benefit from a scanning method that requires no equipment beyond a tablet, works in the field with no network connectivity, and produces meshes suitable for both archival and reproduction.

Medical and prosthetic fitting. Scanning residual limbs or anatomical features for custom prosthetic and orthotic fabrication. On-device processing avoids the HIPAA compliance burden of transmitting patient anatomy to a cloud service.

Education. A physical-to-digital pipeline that a student can operate end-to-end on a single device, with the algorithmic stages — voxel integration, isosurface extraction — visible as real, inspectable steps rather than a black-box cloud service.

6.3 Privacy

Cloud-based scanning services transmit the full geometry of the scanned object — and, unavoidably, whatever appears in the camera frames during scanning — to third-party servers. For scanning in a home, a secure facility, a medical setting, or anywhere with sensitive surroundings, on-device processing is the only architecture that keeps that data local.

07 Limitations and Future Work

Device requirement. ScanForge requires a LiDAR-equipped device: iPad Pro (2020 or later) or iPhone 12 Pro / Pro Max or later. Non-Pro iPhones and standard iPads have no depth sensor and are explicitly rejected at session start with a clear error rather than degrading to a photogrammetric fallback.

Object scale. The 2 mm voxel resolution and cubic memory scaling constrain practical object size to roughly 50 cm on the longest axis before memory pressure becomes significant. Larger objects require either coarser voxels (losing detail) or a spatially-hashed sparse volume (Nießner et al. [5]) — the natural next architectural step.

Surface color. The current pipeline reconstructs geometry only. Vertex colors or a UV-mapped texture atlas from the RGB camera stream would substantially improve the output for visualization use cases, though it is irrelevant for single-material 3D printing.

Marching Cubes ambiguity. The classical Marching Cubes tables contain topologically ambiguous cases that can produce small holes in specific configurations. Marching Tetrahedra or the asymptotic decider [6] resolve these; in practice the ambiguous configurations are rare in TSDF-derived fields with adequate weight, but they are a known theoretical gap.

Multi-threaded extraction. Marching Cubes could be parallelized by partitioning the volume into slabs with per-slab vertex maps merged at boundaries, reducing extraction time from 2–4 s to well under 1 s. This is a straightforward optimization deferred until the sequential timing becomes a user-visible complaint.

08 Conclusion

ScanForge demonstrates that the complete volumetric reconstruction pipeline — real-time GPU TSDF integration, coverage-guided capture, and manifold-guaranteed Marching Cubes extraction — fits within the thermal, memory, and compute budget of a consumer tablet, producing 3D-printable output in seconds rather than the minutes-to-hours of photogrammetric alternatives.

The critical engineering commitment is treating watertightness as a structural property of the extraction algorithm, via edge-keyed vertex deduplication, rather than a post-processing repair. That is what makes the output directly usable by 3D-printing slicers without manual mesh repair. As LiDAR sensors proliferate across the consumer device population, on-device reconstruction of this kind becomes the default expectation rather than a specialized capability.

// References

[1] Curless, B., and Levoy, M., "A Volumetric Method for Building Complex Models from Range Images," SIGGRAPH '96, pp. 303–312, 1996.

[2] Newcombe, R.A., et al., "KinectFusion: Real-Time Dense Surface Mapping and Tracking," IEEE International Symposium on Mixed and Augmented Reality (ISMAR), 2011.

[3] Lorensen, W.E., and Cline, H.E., "Marching Cubes: A High Resolution 3D Surface Construction Algorithm," Computer Graphics (SIGGRAPH '87), vol. 21, no. 4, pp. 163–169, 1987.

[4] Luetzenburg, G., Kroon, A., and Bjørk, A.A., "Evaluation of the Apple iPhone 12 Pro LiDAR for an Application in Geosciences," Scientific Reports, vol. 11, article 22221, 2021.

[5] Nießner, M., et al., "Real-time 3D Reconstruction at Scale using Voxel Hashing," ACM Transactions on Graphics (SIGGRAPH Asia), vol. 32, no. 6, 2013.

[6] Nielson, G.M., and Hamann, B., "The Asymptotic Decider: Resolving the Ambiguity in Marching Cubes," Proceedings of IEEE Visualization '91, pp. 83–91, 1991.

[7] Bourke, P., "Polygonising a Scalar Field," 1994. https://paulbourke.net/geometry/polygonise/

[8] Apple Inc., "ARKit — Scene Reconstruction and Depth," Apple Developer Documentation, 2024. https://developer.apple.com/documentation/arkit

[9] Zhou, Q.-Y., Park, J., and Koltun, V., "Open3D: A Modern Library for 3D Data Processing," arXiv:1801.09847, 2018.

[10] Izadi, S., et al., "KinectFusion: Real-time 3D Reconstruction and Interaction Using a Moving Depth Camera," UIST 2011.