RADIANCEKIT

Benutzerhandbuch

Photorealistische 3D-Rekonstruktion
via Gaussian Splatting

Version 1.5.0 · macOS 26.0+ · Mai 2026

BJOERN KINDLER · KINDLER-DEV.DE

INHALT

Übersicht

Einleitung — Was du wissen solltest ... 3
Was ist RadianceKit? ... 3
Was ist Gaussian Splatting? ... 3
Kapitel 1 — Menüleiste ... 5
File-Menü ... 6
Mode-Menü ... 9
Training-Menü ... 11
Viewport-Menü ... 14
Export-Menü ... 20
Help-Menü ... 25
Hinweis: Cmd-Z im Edit-Menü ... 29
Tastatur-Kurzbefehle in der Übersicht ... 30
Kapitel 2 — Inspector (Expert View) ... 31
Look-Sektion (L1–L5) ... 34
Presets-Sektion (I1–I11) ... 37
Trainingskonfiguration-Sektion (I12–I22) ... 43
Enhancements-Sektion (I26–I29, I42–I44) ... 49
Metriken-Sektion (I30–I38) ... 56
Verlust-Diagramm-Sektion (I39–I41) ... 62
Wann nach Inspector greifen? ... 65
Kapitel 3 — Einstellungen ... 67
General-Tab ... 68
AI-Helpers-Tab ... 73
Inspector-Spiegel-Settings ... 76
Wann was? ... 77
Kapitel 4 — Aux-Fenster ... 78
User Guide (W1–W4) ... 79
Keyboard Shortcuts (W5–W6) ... 82
Manage Storage (W7–W12) ... 84
Pareto Dashboard (W13–W22) ... 88
Holdout Analysis (W23–W29) ... 95
BayesOpt Console (W30–W39) ... 100
Hauptfenster: Verlustverlauf und Gaussian-Count (I39–I41, Querverweis) ... 108
Faustregel-Box ... 109
Kapitel 6 — Trainings-Konfiguration ... 110
Iteration (T1–T2) ... 112
Learning Rates (T3–T10) ... 114

Densification — Classic (T11–T16) 121
Loss (T17–T20) 126
SH-Degree-Progression (T21) 130
Performance (T22–T25) 131
Diagnose und Punktwolken-Vorbereitung (T26–T30) 133
Regularisierung (T31–T37) 136
Refinement (T38–T44) 139
Sky-Dome (T45–T48) 144
Adam + LR-Schedule (T49–T55) 146
Post-Processing + Apple AI (T56–T60) 150
MCMC-Densification (T61–T73) 152
Mip-Splatting (Q1.5) (T74–T76) 159
Adaptive Densification (Q5) (T77–T79) 161
Curriculum (Q6) (T80–T81) 162
Statische Presets (TP1–TP9) 163
Methode: 166
Welches Feld wofür? (Cheat-Sheet) 167
Gefährliche Felder 168
Kapitel 7 — Eingebaute Qualitäts-Presets 169
Wann welches Preset? 179
Schnell-Vergleich 180
Eigene Presets 182
Kapitel 8 — Export-Formate 183
Welches Format wann? 197
Schnell-Vergleich 198
Kapitel 9 — SfM-Backends 199
Welches Backend wann? 205
Schnell-Vergleich 205
Kapitel 10 — Einsteiger-Modus 206
Z1 — Import (Bilder & Preset wählen) 206
Z2 — Verarbeitung (SfM + Training) 215
Z3 — Vorschau (3D-Modell drehen) 222
Z4 — Export (Format wählen & speichern) 226
Wechsel zu Expert-Modus 231
Häufige Fragen 232

LESEHILFE

So liest du dieses Handbuch

Jeder Eintrag im Handbuch folgt demselben Schema. Auf der linken Seite stehen die Bedienpfade und technischen Details, rechts in einem warmen Seitenkasten findest du immer die einfache Erklärung. Kleine Icons am Zeilenanfang verraten dir auf einen Blick, welche Art von Information jetzt folgt.

DIE VIER ICONS

Wo finde ich das? Der konkrete Klickpfad durch die App — Menüleiste, Inspektor-Sektion oder Einsteiger-Modus-Schritt. Auch die zugehörigen Tastatur-Kurzbefehle stehen hier. Das Icon ist ein Karten-Pin und zeigt: Hier sitzt die Funktion in der Benutzeroberfläche.

Details. Standardwerte, Wertebereiche und Code-Pfade. Begegnet dir vor allem bei den Trainingseinstellungen, die kein Menü-Eintrag sind, sondern Zahlen-Parameter. Das Icon zeigt eine kleine Spezifikations-Karte.

Technisch. Was die Funktion intern tut, welche Parameter wirken, worauf sie reagiert und welche Nebenwirkungen sie hat. Für Leser, die verstehen wollen, was im Hintergrund passiert. Das Icon ist ein Schieberegler-Block und steht symbolisch für die Stellschrauben unter der Haube.

Einfach gesagt. Die Kernaussage in klaren Worten — ohne Fachsprache, ohne Code. Lies diesen Abschnitt zuerst, wenn du nur schnell wissen willst, wofür eine Funktion da ist und wann du sie brauchst. Das Icon ist eine Sprechblase und steht für „auf den Punkt gebracht„. Diese Spalte ist immer in einem warmen Sand-Ton hinterlegt, damit das Auge sie sofort findet.

KAPITEL-FARBEN

Jedes Kapitel hat eine eigene Akzentfarbe, die du an der ID-Marke (zum Beispiel M1) links neben jedem Eintragstitel und an den kleinen Icons davor wiedererkennst. Beim Blättern siehst du so sofort, in welchem Kapitel du gerade bist.

1 Menüleiste · 2 Inspektor · 3 Settings · 4 Hilfsfenster · 6 Training

7 Vorlagen · 8 Exporte · 9 SfM · 10 Einsteiger

TIPPS ZUR NAVIGATION

Schnell-Start. Wenn dich nur die Bedienung interessiert, springe direkt zu **Kapitel 10 — Einsteiger-Modus**. Das ist die geführte Variante mit vier Schritten und braucht keinerlei Vorwissen.

Tieferer Einstieg. Kapitel 2 — Inspektor und **Kapitel 7 — Vorlagen** erklären die Bedienelemente und voreingestellten Qualitäts-Profile, die du im Experten-Modus zur Verfügung hast.

Nachschlagen. Inhaltsverzeichnis und PDF-Volltextsuche helfen, eine bestimmte Funktion zu finden. Du musst das Handbuch nicht von vorn nach hinten lesen.

KAPITEL

Einleitung — Was du wissen solltest

Was ist RadianceKit?

RadianceKit ist eine native macOS-App, die aus einer Reihe normaler Fotos oder einem Video eine begehbare 3D-Rekonstruktion macht. Die Eingabe sind beispielsweise 50 bis 500 Aufnahmen, die du um ein Objekt herum, durch einen Raum oder über eine Landschaft gemacht hast. Die Ausgabe ist eine sogenannte Gaussian-Splatting-Szene — ein 3D-Modell, das du am Mac in Echtzeit aus jeder Perspektive ansehen kannst, das sich exportieren und auf Webseiten einbetten lässt und das in den Hauptaspekten photoreal aussieht.

Die App läuft komplett lokal auf deinem Mac — es werden keine Bilder in die Cloud hochgeladen, kein Login wird verlangt, kein Abo. Sie nutzt die GPU deines Apple-Silicon-Macs (M-Serie) intensiv: ein vollständiges Training kann je nach Szene und Preset zwischen zwei Minuten und mehreren Stunden dauern. Während der Berechnung kannst du am Mac ganz normal weiterarbeiten, RadianceKit läuft im Hintergrund weiter und meldet sich, wenn das Ergebnis fertig ist.

Es gibt zwei Bedien-Modi: der *Einsteiger-Modus* (Simple Mode) führt dich in vier Schritten durch den Workflow Import → Vorlagen wählen → Training → Export. Der *Experten-Modus* (Expert Mode) öffnet einen großen Inspector mit allen Stellschrauben, einem Live-Vorschau-Fenster und Diagnose-Charts. Du kannst jederzeit zwischen den Modi wechseln; die Daten in der Szene bleiben dabei erhalten.

Was ist Gaussian Splatting?

Gaussian Splatting (oft kurz *3DGS* oder einfach *Splatting*) ist eine relativ neue Methode für photorealistische 3D-Darstellung, vorgestellt 2023 in einem Paper aus Graz und INRIA. Die Idee: statt eine Szene als klassisches Polygonnetz (Dreiecke) oder als Voxel-Grid zu modellieren, wird sie aus Millionen kleiner, weicher 3D-Wölkchen zusammengesetzt — jede einzelne Wolke ist eine 3D-Gaußverteilung (daher der Name) mit eigener Position, Größe, Form, Farbe und Durchsichtigkeit. Diese Wölkchen werden so trainiert, dass sie aus allen Blickwinkeln deiner Eingabe-Fotos zusammen das richtige Bild ergeben.

Praktisch heißt das: Gaussian Splatting kann Reflexionen, Glanzlichter, weiches Laub, Haare oder Vorhänge so darstellen, wie klassisches 3D-Modellieren es nicht oder nur mit immensem Aufwand kann. Im Gegenzug ist das Ergebnis kein editierbares 3D-Modell im klassischen Sinn — du kannst nicht einfach eine einzelne Wand verschieben oder eine Vase umsetzen. Es ist eher eine *gefrorene Aufnahme* des Raumes, durch die du dich frei bewegen kannst. Für viele Anwendungen — Architektur- Visualisierung, Produkt-Präsentation, virtuelle Touren, Forensik, Kulturerbe — ist genau das die richtige Stärke.

Damit aus den Eingabe-Bildern eine 3D-Szene wird, sind zwei Schritte nötig. Zuerst rechnet die App über ein Verfahren namens *Structure-from-Motion (SfM)* aus, wo deine Kamera bei jedem Foto gestanden hat. Dabei entsteht nebenbei eine grobe Punkt-Wolke der Szene. Dann startet das eigentliche Gaussian-Splatting-Training: ausgehend von dieser groben Wolke werden die Millionen 3D-Wölkchen schrittweise verteilt, vergrößert, verfeinert und in Position und Farbe nachjustiert, bis sie aus allen Eingabe-Blickwinkeln das passende Bild ergeben.

Du musst von beidem nichts wissen, um RadianceKit zu benutzen. Der Einsteiger-Modus blendet diese Schritte komplett aus. Aber wenn du verstehen möchtest, was die Diagnose-Zahlen im Experten-Modus (Iteration, Loss, Gaussians, SSIM …) bedeuten oder warum manche Szenen schöner werden als andere, dann findest du in den späteren Kapiteln des Handbuchs die Antworten.

KAPITEL

Kapitel 1 — Menüleiste

Die Menüleiste von RadianceKit gliedert alle Funktionen, die nicht unmittelbar im Hauptfenster oder im Inspector liegen. Das sind in erster Linie Aktionen, die auf die ganze Szene wirken (Öffnen, Speichern, Neues Projekt), das Training steuern (Start, Pause, Fortsetzen), den Viewport bedienen (Auto-Rotation, Screenshot, Hintergrundfarbe) und Exporte in verschiedene 3D- und Medien-Formate auslösen. Dazu kommen Sprungpunkte zu allen Hilfsfenstern (User Guide, Pareto Dashboard, Holdout Analysis, BayesOpt Console).

Tastatur-Kurzbefehle stehen jeweils rechts vom Menüeintrag. Konventionen: `⌘` bedeutet die Command-Taste (Apfel-Taste), `⇧` ist Shift, `⌥` ist Option (Alt) und `^` ist Control. Beispiel: `⇧⌘T` steht für Shift+Command+T. Alle hier dokumentierten Kurzbefehle sind über `Help → Keyboard Shortcuts` (`⌘/`) zusätzlich in einem eigenen Übersichtsfenster aufgelistet.

Die folgenden 42 Einträge sind in der Reihenfolge der Inventur (M1–M42) dokumentiert, gruppiert nach dem zugehörigen Top-Level-Menü. Alle Einträge wurden gegen den aktuellen Code-Stand in (Zeilen 175–477) verifiziert. Keine Einträge sind entfernt oder gegenüber der Inventur überholt; ein neuer Edit-Menü-Eintrag (Cmd-Z für „Remove Image„) wird durch das System-NSUndoManager-Framework aufgenommen und erscheint daher nicht im RadianceKitApp-Code (siehe Hinweis am Ende des Kapitels).

File-Menü

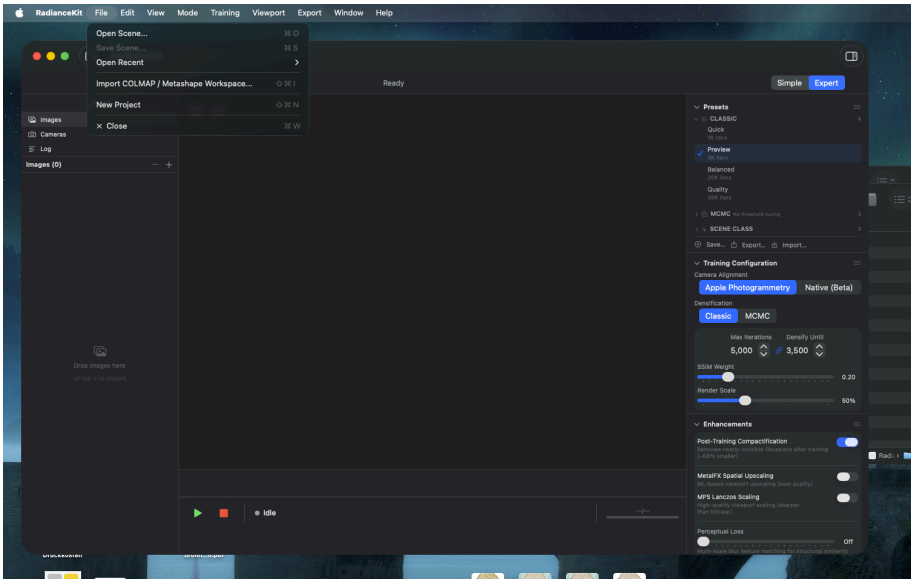

Abbildung 1: Ablage-Menü aufgeklappt — Einträge M1 bis M6

Das File-Menü ersetzt Apples Standard-„New Window„-Eintrag durch projektspezifische Aktionen. Es umfasst Szenen laden/speichern, eine dynamische Recent-Liste, den Workspace-Import sowie das harte Zurücksetzen auf einen Leerzustand.

M1 File > Open Scene…

WO

Menüleiste → File → Open Scene… (⌘O).

TECHNISCH

Öffnet einen Datei-Dialog für die Formate `RadianceScene`-Bundle, `.ply`, `.splat` und `.spz`. Single-Selection, kann sowohl Dateien als auch Verzeichnisse zeigen (für das Bundle-Format). Nach erfolgreicher Auswahl wird der Pfad in die Recent-Liste eingetragen und die Szene asynchron geladen — die vorherige wird ersetzt und die Trainings-Pipeline mit dem geladenen Stand initialisiert. PLY/SPZ/Splat-Dateien werden über die jeweiligen Format-Loader gelesen; das `.radiancescene`-Bundle ist ein Verzeichnis mit Manifest, Cloud-Snapshot und SfM-Resultaten.

EINFACH GESAGT

So lädst du eine bereits trainierte Szene wieder in die App. Funktioniert mit dem RadianceKit-eigenen Format und mit den Standardformaten PLY, SPLAT und SPZ, die andere Splatting-Programme erzeugen. Verwende das, wenn du z. B. eine Szene über Nacht trainiert hast und am nächsten Tag weitermachen oder exportieren willst. Beim Öffnen wird der bisherige Stand im Hauptfenster ersetzt — speichere also vorher, wenn dir die aktuelle Szene noch wichtig ist. Der Pfad landet automatisch in „Open Recent„ (M3), damit du beim nächsten Mal schneller dran kommst.

M2 File > Save Scene…

WO

Menüleiste → File → Save Scene… (⌘S).

TECHNISCH

Öffnet einen Datei-Speichern-Dialog mit dem Content-Type `RadianceScene`-Bundle und vorausgefülltem Dateinamen `scene.radiancescene`. Schreibt ein Verzeichnis-Paket mit `manifest.json`, der serialisierten Gaussian-Cloud (PLY-Snapshot) und einem Dump des SfM-Resultats, sodass nach dem Wiederöffnen auch das Continue-Training funktioniert. Der Eintrag ist deaktiviert, solange noch keine Gaussians existieren. Speichert nicht in den Training-Logs-Pfad, sondern dorthin, wo der Speichern-Dialog zeigt — typischerweise unter `~/Documents/`.

EINFACH GESAGT

Speichert deine aktuelle Szene als Datei (genauer: als Paket-Ordner, der wie eine Datei aussieht). Erst danach kannst du diese Szene später mit „Open Scene…„ (M1) wieder öffnen. Im Paket landen sowohl die Gaussian-Cloud als auch das SfM-Resultat, sodass du auch Continue-Training (M12–M14) später noch anhängen kannst. Solange du noch kein Training abgeschlossen hast, ist der Eintrag ausgegraut. Der Standardname ist `scene.radiancescene` — du kannst aber im Save-Dialog einen eigenen Namen vergeben.

M3 File > Open Recent > [Szenennamen]

WO

Menüleiste → File → Open Recent → (Liste).

TECHNISCH

Dynamisches Submenü, das aus einer Liste der zuletzt geöffneten Pfade (in den Einstellungen gespeichert) generiert wird. Jeder Listen-Eintrag wird mit dem Dateinamen benannt und bei Klick geladen. Wenn die Liste leer ist, erscheint stattdessen das deaktivierte Label „No Recent Scenes„. Apple-typisch hält die Liste die N zuletzt geöffneten Szenen — die Begrenzung findet beim Schreiben in die Einstellungen statt und nicht im Menü-Builder selbst.

EINFACH GESAGT

Hier siehst du die zuletzt geöffneten Szenen und kannst mit einem Klick wieder reinspringen, ohne über den Datei-Dialog gehen zu müssen. Wenn du gerade erst angefangen hast, ist die Liste leer und steht grau im Menü. Jede Szene, die du über „Open Scene…„ (M1) öffnest, landet automatisch in dieser Liste. Wenn dir die Liste irgendwann zu voll wird oder du sie aus Datenschutzgründen leeren willst, nutze „Clear Recent“ (M4).

M4 File > Open Recent > Clear Recent

WO

Menüleiste → File → Open Recent → Clear Recent.

TECHNISCH

Leert die Recent-Liste in den Einstellungen. Wirkt sofort, ohne Bestätigungsdialog. Der Eintrag erscheint nur dann im Submenü, wenn überhaupt Einträge in der Recent-Liste vorhanden sind (er steht unter einem Divider nach den Pfaden).

EINFACH GESAGT

Löscht die Liste der zuletzt geöffneten Szenen. Praktisch, wenn du mit einem Test-Datensatz herumgespielt hast und die Pfade nicht mehr sehen willst. Die Szenen-Dateien selbst werden dabei nicht gelöscht — nur die Verknüpfung im Menü. Die Aktion wirkt sofort, ohne Rückfrage; danach erscheint im Submenü „No Recent Scenes„. Der Eintrag taucht nur dann auf, wenn überhaupt Szenen in der Liste sind — bei leerer Liste ist er nicht sichtbar.

M5 File > Import COLMAP / Metashape Workspace…

WO

Menüleiste → File → Import COLMAP / Metashape Workspace… (⇧⌘I).

TECHNISCH

Öffnet einen Ordner-Picker. Erwartet einen Ordner mit dem COLMAP-Workspace-Layout (z. B. `sparse/0/cameras.{bin,txt}` plus `images/`). Nach Auswahl wird eine Vorprüfung des Workspace durchgeführt — diese erkennt die drei Layouts (`sparse/0/`, `sparse/`, Root) und ob die Rekonstruction binär (`cameras.bin`) oder als ETH3D-Text (`cameras.txt`) vorliegt. Bei Erfolg wird der Workspace importiert; andernfalls erscheint nur eine Warnung im App-Log. Siehe auch Kapitel 9 „SfM-Backends„, Q6 für die volle Pipeline-Logik.

EINFACH GESAGT

Wenn du Metashape, COLMAP, RealityCapture oder eine ähnliche Software für die Kamerarekonstruktion benutzt und einen Export hast, lädst du den Ordner hier rein. RadianceKit überspringt dann die SfM-Stufe und startet direkt mit dem Training — das spart bei großen Szenen Stunden. Drag-and-Drop auf das Hauptfenster funktioniert genauso. Erwartet wird ein Ordner mit COLMAP-Layout (also `sparse/0/` mit `cameras.*` plus `images/`-Ordner). Mehr zu den unterstützten Layouts und Workflows steht in Kapitel 9 „SfM-Backends„.

M6 File > New Project

WO

Menüleiste → File → New Project (⇧⌘N).

TECHNISCH

Prüft, ob ungesicherte Arbeit vorhanden ist. Falls ja, erscheint ein Bestätigungsdialog, bevor irgendetwas verloren geht. Wenn nichts zu speichern ist, läuft das Zurücksetzen direkt — es leert importierte Bilder, das SfM-Resultat, die Gaussian-Cloud, Training-State und alle abhängigen UI-Indikatoren. Achtung: Eine vom Nutzer angelegte Preset-Bibliothek bleibt erhalten, weil sie in den App-Einstellungen und nicht im Projektstand liegt.

EINFACH GESAGT

Setzt alles zurück auf einen leeren Start — als ob du die App gerade frisch geöffnet hättest. Wenn du noch ungesicherte Arbeit hast, fragt die App vorher nach. Verwende das, wenn du mit einer ganz anderen Szene anfangen willst. Importierte Bilder, SfM-Resultat, Gaussian-Cloud und Trainings-Zustand werden komplett geleert. Deine eigenen Presets bleiben aber erhalten, weil die in den App-Einstellungen liegen und nicht zur Szene gehören.

Mode-Menü

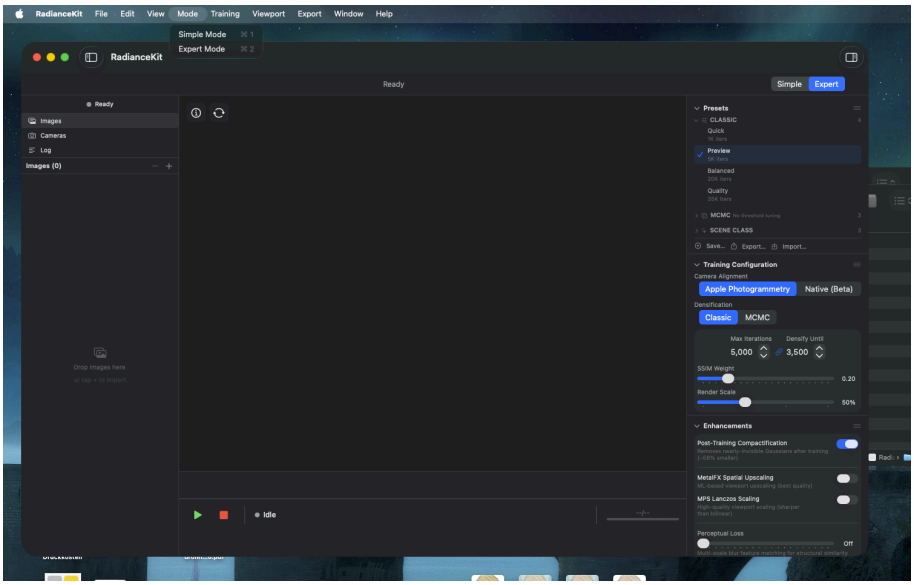

Abbildung 2: Modus-Menü mit Simple- und Expert-Mode-Schalter

Zwei einfache Schalter zwischen dem geführten Simple Mode (Wizard-artig, 4 Schritte) und dem vollen Expert Mode (klassisches Inspector-Layout mit allen Reglern).

M7 Mode > Simple Mode

WO

Menüleiste → Mode → Simple Mode (⌘1).

TECHNISCH

Schaltet den App-Zustand auf den Simple Mode um. Der Hauptbereich der App zeigt dann den geführten Workflow statt des Expert-Layouts. Der Mode-Zustand wird in den Einstellungen gespeichert (siehe S1 „Default Mode„ in Kapitel 3 Settings).

EINFACH GESAGT

Schaltet auf die Schritt-für-Schritt-Variante um, bei der dich die App durch Importieren, Verarbeiten, Vorschau und Export führt. Empfohlen, wenn du gerade erst anfängst oder schnell ein Ergebnis brauchst. Die meisten Detail-Regler sind ausgeblendet — du arbeitest mit sinnvollen Voreinstellungen. Wenn du später tiefer einsteigen willst, wechsle einfach in den Expert Mode (M8). Welcher Modus beim App-Start aktiv ist, kannst du in den Einstellungen (Kapitel 3, S1) festlegen.

M8 Mode > Expert Mode

WO

Menüleiste → Mode → Expert Mode (⌘2).

TECHNISCH

Schaltet den App-Zustand auf den Expert Mode. Damit erscheint das volle Inspector-Layout mit allen Sections (Presets, TrainingConfig, Enhancements, Metrics, LossChart, ProjectNavigator). Im Expert Mode sind sämtliche Training-Parameter, COLMAP-Picker, Mid-Compact-Toggles und Diagnostics zugänglich. Auch der Live-Preview funktioniert nur in diesem Modus.

EINFACH GESAGT

Schaltet in die Vollansicht mit allen Reglern. Hier siehst du Loss-Charts in Echtzeit, kannst alle Parameter feinjustieren und mehrere Vergleichs-Configs parallel über Presets verwalten. Empfohlen, wenn du verstehen willst, was das Training intern macht, oder wenn du gezielt experimentieren möchtest. Auch die Live-Preview, COLMAP-Picker und die Diagnostics sind nur hier zugänglich. Wenn du dich überfordert fühlst, geh über M7 zurück in den Simple Mode — deine Szene bleibt dabei erhalten.

Training-Menü

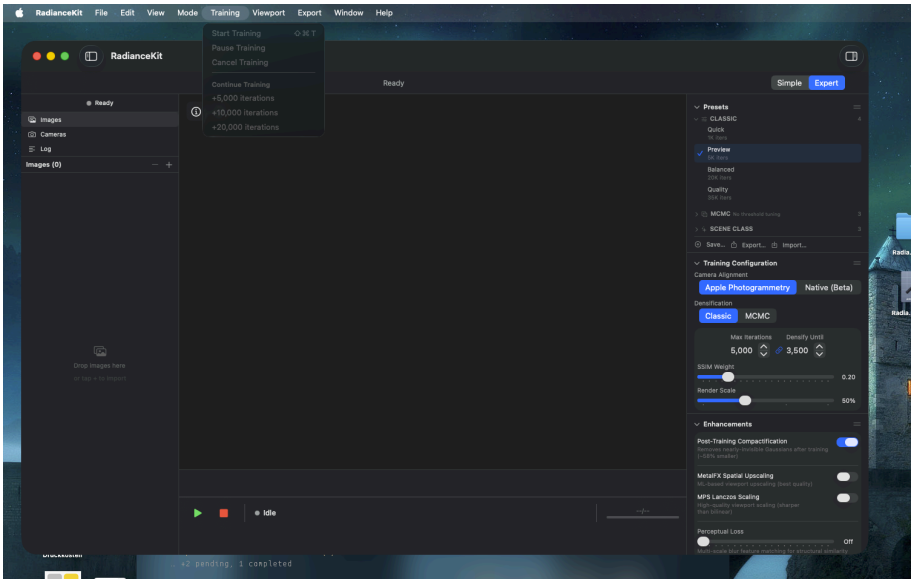

Abbildung 3: Training-Menü mit Continue-Submenü — Einträge M9 bis M14

Vier Aktionen rund um den Trainings-Lauf: starten, pausieren, abbrechen und um eine vorgegebene Iterationszahl verlängern. Alle drei Continue-Einträge sind über IAP-gegated (in der Free-Trial-Version nicht klickbar).

M9 Training > Start Training

WO

Menüleiste → Training → Start Training (⇧⌘T).

TECHNISCH

Startet die Trainings-Pipeline asynchron. Voraussetzung: ein SfM-Resultat liegt vor und es läuft gerade keine andere Pipeline. Beide Bedingungen blockieren den Eintrag, falls nicht erfüllt. Beim Start werden die aktuellen Konfigurations-Werte gelesen, ein neuer JSONL-Log unter `~/Documents/RadianceKit/Logs/training_YYYY-MM-DD_HHmmss.jsonl` angelegt, und je nach Strategie-Wahl der klassische oder der MCMC-Pfad gefahren. Der Trainings-Zustand wechselt von „idle„ auf „training“.

EINFACH GESAGT

Drückt den großen grünen Knopf — sobald du Fotos importiert hast und die Kamera-Rekonstruktion durch ist, beginnt damit das eigentliche Gaussian-Splatting-Training. Lass die App laufen; je nach Preset zwischen 1 Minute (Quick) und mehreren Stunden (MCMC Quality). Der Eintrag bleibt grau, solange noch kein SfM-Ergebnis vorliegt oder gerade eine andere Pipeline läuft. Jeder Lauf schreibt nebenbei ein Log nach `~/Documents/RadianceKit/Logs/`, das du später über das Pareto Dashboard (M40) auswerten kannst.

M10 Training > Pause Training

WO

Menüleiste → Training → Pause Training.

TECHNISCH

Pausiert das laufende Training. Wird nur freigeschaltet, wenn der Trainings-Zustand „training„ ist. Pausieren stoppt den Iterations-Loop am nächsten Sicherheits-Sync-Point, behält den vollen GPU-State (Gaussian-Buffers, Optimizer-Moments, Scheduler-Position) und schaltet auf „paused“. Resume erfolgt über erneutes Drücken (der Eintrag-Titel ist statisch — die App wechselt aber zwischen Pause/Resume in der eigentlichen Logik). Pausierte Trainings überleben kein App-Quit; in dem Fall stattdessen die Szene speichern und später per Continue-Training-Eintrag (M12–M14) erweitern.

EINFACH GESAGT

Hält das Training kurz an, ohne den Fortschritt zu verlieren. Praktisch, wenn du den Computer kurz für etwas Wichtigeres brauchst. Nochmal klicken setzt fort. Funktioniert nicht über App-Neustarts hinweg — wenn du wirklich später weitermachen willst, beende das Training mit Cancel (M11), speichere die Szene mit Save Scene (M2) und nutze danach Continue Training (M12–M14). Während der Pause ruht die GPU vollständig; der Speicher bleibt aber belegt.

M11 Training > Cancel Training

WO

Menüleiste → Training → Cancel Training.

TECHNISCH

Bricht das laufende Training ab. Aktiv, wenn der Trainings-Zustand nicht „idle„ ist. Setzt den Cancel-Flag in der Trainings-Engine, was den Iterations-Loop am nächsten Sync-Point sauber beendet, den finalen Summary-Eintrag mit ins JSONL-Log schreibt und den Zustand auf „idle“ zurücksetzt. Die bisher trainierte Cloud bleibt erhalten (kann gespeichert oder exportiert werden), wird aber als „cancelled„ markiert.

EINFACH GESAGT

Bricht das laufende Training endgültig ab. Der bisherige Stand bleibt — wenn du also nach ein paar tausend Iterationen schon ein vorzeigbares Ergebnis hast, kannst du es danach trotzdem exportieren. Wenn du nur kurz unterbrechen willst, nutze stattdessen Pause (M10). Im Trainings-Log wird der Lauf als „cancelled„ markiert, der finale Loss-Wert wird trotzdem mit weggeschrieben. Eine abgebrochene Szene kannst du auch über Continue Training (M12–M14) später fortsetzen, solange die App nicht zwischendurch beendet wurde.

M12 Training > Continue Training > +5 000 iterations

WO

Menüleiste → Training → Continue Training → +5,000 iterations.

TECHNISCH

Setzt das Training um 5 000 Iterationen fort. Aktiv, wenn ein abgeschlossenes Training fortsetzbar ist und die Vollversion freigeschaltet ist. Die Fortsetzbarkeit gilt, wenn ein abgeschlossenes Training existiert und der volle Optimizer-State noch im Speicher steckt. Beim Continue werden die Adam-Momente und der LR-Scheduler weitergeführt, sodass die Fortsetzung sich verhält wie ein durchgehender 25K-/45K-/60K-Lauf statt eines Neustarts. Der JSONL-Log bekommt einen neuen Config-Eintrag mit dem inkrementellen Setup. Nur in der Vollversion verfügbar.

EINFACH GESAGT

Hängt 5 000 weitere Trainings-Schritte an. Verwende das, wenn das Ergebnis nach dem ersten Lauf nahe dran, aber noch nicht ganz scharf ist. Funktioniert nur in der bezahlten Vollversion. Im Unterschied zu einem komplett neuen Lauf bleibt der Optimizer-Zustand erhalten, sodass die Fortsetzung sich anfühlt wie ein durchgängiger Lauf. Wenn du mehr als 5 000 Schritte brauchst, nimm gleich M13 (+10 000) oder M14 (+20 000).

M13 Training > Continue Training > +10 000 iterations

WO

Menüleiste → Training → Continue Training → +10,000 iterations.

TECHNISCH

Identisch zu M12, aber mit 10 000 zusätzlichen Iterationen. Gleiche Vorbedingungen, gleicher LR-Scheduler-Pfad. Empfohlen, wenn das initiale Training mit einem Mid-Tier-Preset gefahren wurde und du eine signifikante Qualitätssteigerung sehen willst, ohne den Lauf komplett neu zu starten.

EINFACH GESAGT

Verlängert das Training um 10 000 Schritte — der mittlere der drei verfügbaren Continue-Werte. Gute Wahl, wenn der erste Lauf zwar okay war, du aber noch klar besser werden willst. Wie auch M12 und M14 wird der Lernraten-Verlauf nahtlos fortgesetzt, statt neu zu starten. Nur in der Vollversion verfügbar.

M14 Training > Continue Training > +20 000 iterations

WO

Menüleiste → Training → Continue Training → +20,000 iterations.

TECHNISCH

Identisch zu M12 / M13, aber mit 20 000 zusätzlichen Iterationen. Der größte vorgegebene Continue-Sprung. Bei MCMC-Trainings ist das oft das, was den Unterschied zwischen „passt„ und „benchmark-tauglich“ macht; bei Classic ab 35–40K kommt erfahrungsgemäß wenig dazu.

EINFACH GESAGT

Hängt 20 000 zusätzliche Trainings-Schritte an, der maximale Continue-Wert. Verwende das, wenn du wirklich das letzte Quäntchen Qualität rausholen willst. Bei klassischem Training nach 40 000 Schritten bringt das oft nicht mehr viel — bei MCMC dagegen lohnt es sich häufig, weil dort die Konvergenz langsamer einsetzt. Rechne je nach Szene mit deutlicher Mehrlaufzeit. Wie M12 und M13 ist auch dieser Eintrag nur in der Vollversion verfügbar.

Viewport-Menü

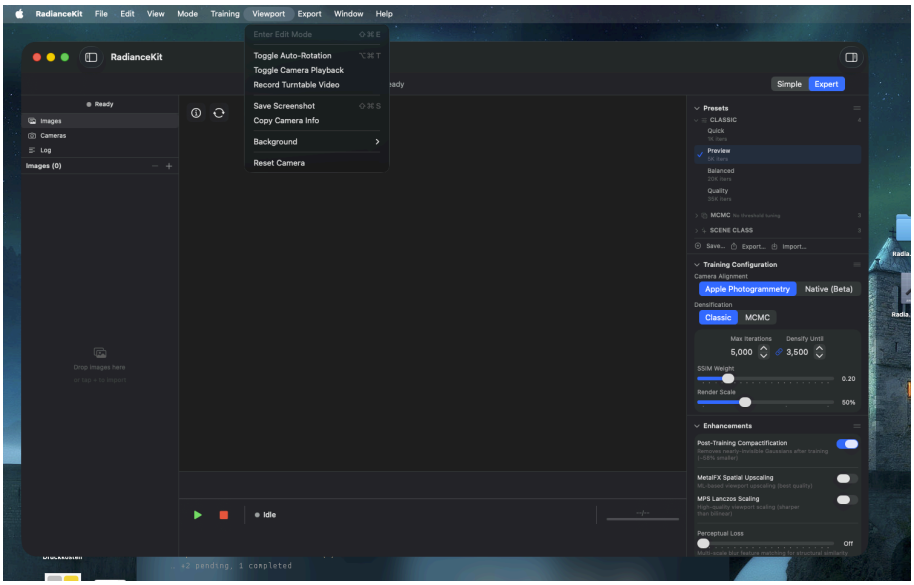

Abbildung 4: Ansichtsfenster-Menü mit Edit-Mode, Kamera-Steuerung und Hintergrund-Submenü

Steuert den 3D-Viewport: Edit-Mode für Gaussian-Selektion und Cleanup, Kamera-Steuerung (Auto-Rotation, Playback, Recording), Screenshot, Hintergrundfarbe und Reset.

M15 Viewport > Enter/Exit Edit Mode

WO

Menüleiste → Viewport → Enter Edit Mode (oder „Exit Edit Mode„, je nach Zustand). ⇧⌘E.

TECHNISCH

Der Eintrag-Titel ist dynamisch und zeigt je nach Zustand „Exit Edit Mode„ oder „Enter Edit Mode“. Beim Drücken wird der Edit-Mode auf dem Viewport-Renderer umgeschaltet. Beim Verlassen des Edit Modes wird zusätzlich die aktuelle Auswahl zurückgesetzt. Der Edit Mode aktiviert die Klick-Selection auf Gaussians, die Box-Selection und das Löschen markierter Gaussians (siehe Editor-Bereich der UI). Deaktiviert solange kein Viewport-Renderer angeschlossen ist.

EINFACH GESAGT

Schaltet zwischen normaler 3D-Ansicht und einem Bearbeitungsmodus um, in dem du einzelne Gaussians markieren und löschen kannst (etwa Floater oder Ausreißer im Hintergrund). Beim Verlassen wird die Auswahl automatisch zurückgesetzt. Der Eintrag bleibt grau, solange noch keine Szene im Viewport sichtbar ist. Die Beschriftung wechselt je nach Zustand zwischen „Enter Edit Mode„ und „Exit Edit Mode“ — du siehst also immer, in welchem Modus du gerade bist.

M16 Viewport > Toggle Auto-Rotation

WO

Menüleiste → Viewport → Toggle Auto-Rotation (⌘⌥T).

TECHNISCH

Schaltet die kontinuierliche Rotation der Viewport-Kamera um eine vertikale Achse durch das Szenen-Zentrum an oder aus. Die Achse und Geschwindigkeit kommen aus der Kamera-Steuerungs-Konfiguration. Auto-Rotation ist ein reiner Viewport-Effekt und beeinflusst weder Training noch Recording — wenn du parallel den Turntable-Video-Recorder benutzt (M18), liefert die Auto-Rotation aber genau den Pfad, den der Recorder einfängt.

EINFACH GESAGT

Dreht die Kamera dauernd langsam um deine Szene, damit du sie von allen Seiten sehen kannst, ohne mit der Maus zu ziehen. Nochmal klicken stoppt die Rotation. Praktisch beim Begutachten fertig trainierter Szenen oder als Hintergrund-Animation für eine Live-Demo. Wenn du parallel ein Video aufnimmst (M18), liefert die Auto-Rotation genau die Bewegung, die der Recorder einfängt.

M17 Viewport > Toggle Camera Playback

WO

Menüleiste → Viewport → Toggle Camera Playback.

TECHNISCH

Schaltet den Kamera-Pfad-Playback um. Wenn ein aufgezeichneter Kamera-Pfad existiert (z. B. aus einem vorhergehenden Recording oder weil eine `transforms.json` geladen wurde), läuft der Pfad ab — die Viewport-Kamera bewegt sich also nicht mehr nach Maus-/Trackpad-Eingaben, sondern reproduziert die Trajektorie Frame für Frame. Erneutes Drücken pausiert den Playback.

EINFACH GESAGT

Lässt eine vorher aufgezeichnete oder importierte Kamerafahrt ablaufen. So kannst du den Originalpfad nachvollziehen, mit dem die Szene aufgenommen wurde, oder eine geplante Orbit-Bewegung vor dem Video-Export prüfen. Während der Playback läuft, sind Maus- und Trackpad-Eingaben deaktiviert — die Kamera folgt strikt dem Pfad. Erneutes Klicken pausiert die Wiedergabe. Wenn du keinen Kamerapfad geladen oder aufgezeichnet hast, passiert nichts.

M18 Viewport > Record Turntable Video

WO

Menüleiste → Viewport → Record Turntable Video.

TECHNISCH

Schaltet die Viewport-Aufnahme um. Beim ersten Drücken startet eine Frame-Aufzeichnung in einen temporären Pfad; beim zweiten Drücken wird die Aufnahme beendet, encodiert und in einen MP4-Pfad geschrieben (Pfad wird über einen Speichern-Dialog abgefragt). Im Unterschied zum Export → Media → Orbit Video (M31), das einen festen 360°-Pfad bei einer einstellbaren Dauer erzeugt, nimmt der Turntable-Recorder *live* das auf, was du im Viewport siehst — du kannst also auch eine manuelle Kamerafahrt aufnehmen.

EINFACH GESAGT

Nimmt direkt im Viewport ein Video auf. Egal ob die Kamera automatisch dreht oder ob du sie selbst mit der Maus bewegst — alles, was du siehst, wird in eine MP4-Datei gespeichert. Im Unterschied zum „Orbit Video„-Export (M31) gibst du selbst die Kamerafahrt vor. Erstes Klicken startet die Aufnahme, zweites Klicken beendet sie und fragt dich nach dem Speicherort. Praktisch, wenn du z. B. einen bestimmten Detail-Schwenk zeigen willst, der mit der starren Orbit-Bewegung nicht möglich wäre.

M19 Viewport > Save Screenshot

WO

Menüleiste → Viewport → Save Screenshot (⇧⌘S).

TECHNISCH

Erfasst einen einzelnen Viewport-Frame in voller Render-Auflösung (also nicht das Fenster-Pixel-Layout, sondern den vollen Render-Target-Inhalt) als PNG-Datei. Der Pfad wird über einen Speichern-Dialog abgefragt. Hintergrundfarbe (M21–M23) wird mit eingebrannt. MetalFX-/MPS-Upscaling-Einstellungen aus den Enhancements (siehe I27/I28) wirken sich aus, wenn aktiv — der Screenshot zeigt also den hochskalierten Output.

EINFACH GESAGT

Speichert einen Schnappschuss deiner aktuellen 3D-Ansicht als PNG-Bild. Praktisch für Marketing-Material oder einen schnellen Vergleich. Beachte: Der Hintergrund ist Teil des Bildes — falls du Transparenz brauchst, exportiere lieber eine Szenen-Datei. Die Auflösung entspricht dem internen Render-Target, nicht deiner Fenstergröße — das Bild ist also oft schärfer, als es im Fenster aussieht. Etwaige Upscaling-Einstellungen (Inspector → Enhancements) fließen ebenfalls mit ein.

M20 Viewport > Copy Camera Info

WO

Menüleiste → Viewport → Copy Camera Info.

TECHNISCH

Liest die aktuelle Viewport-Kamera-Pose (Position, Look-At-Punkt, Up-Vektor) und die FOV-Werte aus der Kamera-Steuerung und schreibt sie als Mehrzeilen-Text in die Zwischenablage. Format ist menschenlesbar (label = value je Zeile), nicht JSON. Praktisch, um eine spezifische Ansicht zu Debug-Zwecken zu reproduzieren oder mit dem Support zu teilen.

EINFACH GESAGT

Kopiert die aktuelle Kamera-Position und Blickrichtung als Text in die Zwischenablage. Wenn du z. B. einem Mitentwickler zeigen willst, von wo aus eine Stelle in der Szene komisch aussieht, fügst du den Text einfach in eine Mail oder ein Chat-Fenster ein. Das Format ist menschenlesbar (eine Zeile pro Wert), kein JSON. Hauptsächlich für Bug-Reports oder Support-Anfragen gedacht.

M21 Viewport > Background > Dark Gray

WO

Menüleiste → Viewport → Background → Dark Gray.

TECHNISCH

Setzt die Viewport-Hintergrundfarbe auf ein dunkles Grau (RGB 0.1/0.1/0.1). Der Renderer verwendet diese Farbe als Hintergrund, vor dem die Gaussians composited werden. Die Default-Farbe bei App-Start steuert die Settings-Option S3 „Default Viewport Background„.

EINFACH GESAGT

Färbt den Hintergrund des 3D-Viewports dunkelgrau. Standardwahl für die meisten Szenen — bietet einen guten Kontrast zu hellen wie dunklen Gaussians, ohne dass das Auge sich an einer reinen Schwarz- oder Weißfläche festkrallt. Die Farbe wird auch in Screenshots (M19) und Orbit-Videos (M31) übernommen. Wenn dir Dark Gray zu unspektakulär ist, probier zum Vergleich auch Black (M22) oder White (M23). Welche Farbe beim App-Start aktiv ist, kannst du in den Einstellungen (S3) festlegen.

M22 Viewport > Background > Black

WO

Menüleiste → Viewport → Background → Black.

TECHNISCH

Setzt die Viewport-Hintergrundfarbe auf reines Schwarz (RGB 0/0/0). Hilft, falls die Szene viele helle Floater hat und du sie identifizieren willst, oder für Marketing-Material mit dunklem Look-and-Feel.

EINFACH GESAGT

Schwarzer Hintergrund. Gut für sehr helle Szenen oder wenn du in den Edit Mode hineinschauen willst und kleine helle Gaussians (Floater) suchst, die im Grau untergehen. Auch ideal für Marketing-Material mit dunklem, dramatischem Look. Die Farbe wird in Screenshots und Orbit-Videos eingebrannt — wenn du Transparenz für einen späteren Composit brauchst, ist Schwarz die schlechteste Wahl. Für dunkle Floater wechsle nochmal in die andere Richtung auf White (M23).

M23 Viewport > Background > White

WO

Menüleiste → Viewport → Background → White.

TECHNISCH

Setzt die Viewport-Hintergrundfarbe auf reines Weiß (RGB 1/1/1). Nützlich, wenn die Szene überwiegend dunkle Inhalte hat und du dunkle Floater (typisches Outdoor-Hintergrund-Rauschen) sehen willst.

EINFACH GESAGT

Weißer Hintergrund. Praktisch, wenn das Motiv hell-auf-dunkel besser zur Geltung kommt, oder um dunkle Ausreißer zu finden, die du danach im Edit Mode (M15) entfernen kannst. Bei Outdoor-Szenen ist Weiß oft nützlicher als Schwarz, weil die typischen Outdoor-Floater eher dunkel sind. Wie bei den anderen Hintergrund-Optionen wird die Farbe in Screenshots und Videos übernommen.

M24 Viewport > Reset Camera

WO

Menüleiste → Viewport → Reset Camera.

TECHNISCH

Setzt die Viewport-Kamera zurück, verlässt die Training-Camera-Ansicht und stoppt die Auto-Rotation. Damit ist die Kamera zurück auf der initialen Position (typisch: vor der Szene, leicht von oben blickend), die Auto-Rotation ist aus, und falls der Renderer gerade die Training-Camera (eine der SfM-Posen) anzeigte, geht er zur Free-Camera zurück.

EINFACH GESAGT

Bringt die Viewport-Kamera in die Startposition zurück. Wenn du dich beim Rumdrehen verirrt hast oder die Szene aus dem Bild geschoben hast — einmal hier klicken und du siehst wieder, was du sehen solltest. Schaltet gleichzeitig die Auto-Rotation aus, falls die gerade läuft, und kehrt aus einer eingefrorenen Training-Kamera in die freie Ansicht zurück. So bekommst du in jedem Fall einen sauberen Neustart der Ansicht.

Export-Menü

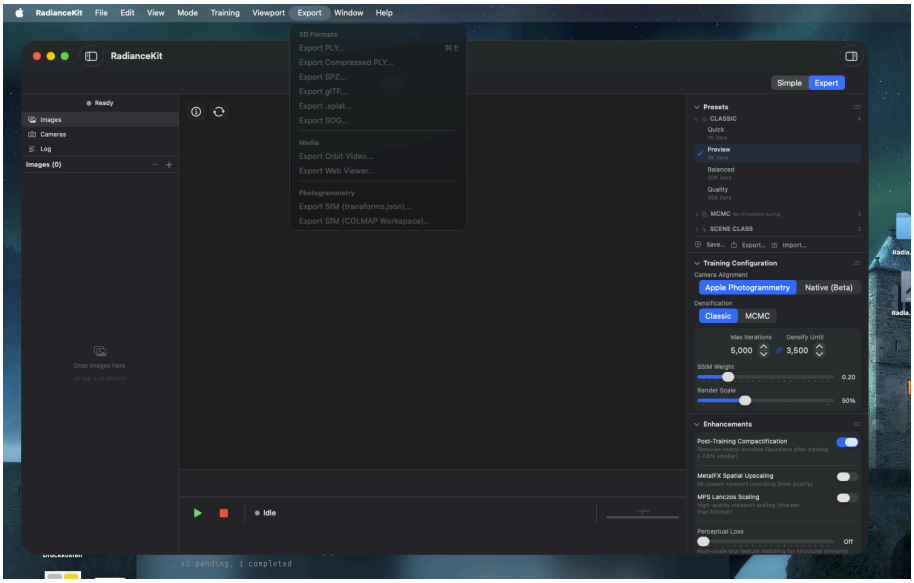

Abbildung 5: Exportieren-Menü mit drei Submenü-Gruppen — 3D Formats, Media und Photogrammetry

Acht Export-Ziele plus zwei Photogrammetry-Exporte, gruppiert in drei Sections (3D Formats, Media, Photogrammetry). Die ersten sechs werden über eine gemeinsame Helper-Routine gebaut, die jeweils einen Speichern-Dialog öffnet und den Export am Format-Katalog registriert. Die Photogrammetry-Einträge haben individuelle Logik. Alle Photogrammetry- und manche 3D-Exporte sind nur in der Vollversion verfügbar.

M25 Export > 3D Formats > Export PLY...

WO

Menüleiste → Export → 3D Formats → PLY (⌘E).

TECHNISCH

Öffnet einen Speichern-Dialog mit Vorgabe-Dateinamen `gaussians.ply`. Bei OK wird die aktuelle Gaussian-Cloud ins standardisierte ASCII/Binary-PLY-Format geschrieben — kompatibel mit SuperSplat, PolyCam, PlayCanvas und allen gängigen 3DGS-Viewern. Volle SH-Koeffizienten, volle Präzision (Float32 pro Feld). Datei-Größe oft mehrere hundert MB bei ≥ 500K Gaussians.

EINFACH GESAGT

Speichert deine 3D-Szene als Standard-PLY-Datei. Das ist das universellste Format — fast jede Software kann das laden, von SuperSplat über PolyCam bis PlayCanvas. Die Dateien werden allerdings groß, oft mehrere hundert Megabyte. Verwende PLY, wenn du in voller Qualität weitermachen oder archivieren willst. Wenn du die Szene per Web teilen willst, schau dir lieber SPZ (M27) oder Compressed PLY (M26) an — die sind wesentlich kleiner.

M26 Export > 3D Formats > Export Compressed PLY...

WO

Menüleiste → Export → 3D Formats → Compressed PLY.

TECHNISCH

Schreibt die Gaussian-Cloud im Compressed-PLY-Format mit Custom-Quantisierung der Position-, Scale-, Rotation- und SH-Felder. 5–10× kleinere Dateien als das unkomprimierte PLY (M25) bei minimalen visuellen Verlusten. Kompatibel mit SuperSplat (das den Compressed-PLY-Standard liest) und PlayCanvas. Standard-Dateiname `gaussians_compressed.ply`.

EINFACH GESAGT

Wie das normale PLY, aber 5–10 Mal kleiner. Die Qualität bleibt fast gleich. Verwende das, wenn du die Datei online teilen willst oder per E-Mail verschickst. Funktioniert direkt mit SuperSplat und PlayCanvas. Wenn dein Zielsystem aber noch kleinere Dateien braucht (Mobile, Browser-Demos), nimm stattdessen SPZ (M27) — das ist nochmal aggressiver komprimiert. Für volle Bearbeitungs-Qualität nimm das unkomprimierte PLY (M25).

M27 Export > 3D Formats > Export SPZ...

WO

Menüleiste → Export → 3D Formats → SPZ.

TECHNISCH

Schreibt die Gaussian-Cloud im SPZ-Format — das von Niantic veröffentlichte komprimierte Splat-Format mit aggressiver Quantisierung (~90 % kleiner als unkomprimiertes PLY). Vor allem für Web-Viewer und mobile Apps optimiert. Kompatibel mit Niantic Splatt3R, gsplat.js und dem Niantic-Browser-Viewer.

EINFACH GESAGT

Eines der kleinsten Formate. Etwa 10× kleiner als ein normales PLY. Verwende das vor allem, wenn du die Szene in einem Browser zeigen oder per Handy-App betrachten willst. Für maximale Qualität ist PLY die bessere Wahl. SPZ ist von Niantic entwickelt und funktioniert direkt mit gsplat.js, Splatt3R und dem Niantic-Web-Viewer. Wegen der starken Komprimierung kannst du SPZ-Dateien nicht mehr ohne Weiteres weitertrainieren — für Bearbeitung nimm PLY.

M28 Export > 3D Formats > Export glTF...

WO

Menüleiste → Export → 3D Formats → glTF.

TECHNISCH

Schreibt eine `.glb`-Datei (Binary-glTF) mit der KHR_gaussian_splatting-Extension. Standard-konform, geeignet für Pipelines, die glTF-Engines wie Babylon.js oder Three.js verwenden und die KHR_gaussian_splatting-Extension implementieren.

EINFACH GESAGT

Speichert die Szene im glTF-Format, das viele 3D-Programme und Web-Engines verstehen — vorausgesetzt, sie unterstützen die Gaussian-Splatting-Erweiterung. Wenn du eine spezifische 3D-Pipeline hast (z. B. Three.js oder Babylon.js), die das versteht, ist das dein Format. Die Datei kommt als binäres `.glb` raus — ein einzelnes Paket, das alles enthält. Für klassische Splatting-Workflows ist meistens PLY oder SPZ die bessere Wahl, weil mehr Tools die direkt verstehen.

M29 Export > 3D Formats > Export .splat...

WO

Menüleiste → Export → 3D Formats → .splat.

TECHNISCH

Schreibt das Antimatter15-`.splat`-Format — fixed-size 32 Bytes pro Gaussian (Position als 3× Float32, Scale als 3× Float32, Rotation als 4× Uint8 normalisierte Quaternion, RGB+Opacity als 4× Uint8). Keine SH-Koeffizienten höher als DC. Kleinste Datei mit Browser-Direkt-Kompatibilität. Für gsplat.js und antimatter15s Online-Demo-Viewer.

EINFACH GESAGT

Das simpelste Web-Viewer-Format. Klein und sofort in jedem Browser anzeigbar. Verliert aber die Detail-Beleuchtung (höhere SH-Koeffizienten gehen verloren — der Splat sieht aus jedem Blickwinkel gleich aus, statt auf das Licht zu reagieren). Für maximale Web-Performance gut, für Foto-Realismus eher SPZ oder PLY. Funktioniert mit dem antimatter15-Online-Viewer und gsplat.js. Jede Gaussian belegt fix 32 Bytes, was das Format simpel und kompatibel macht — aber eben um den Preis der Detailtiefe.

M30 Export > 3D Formats > Export SOG...

WO

Menüleiste → Export → 3D Formats → SOG.

TECHNISCH

Schreibt die Gaussian-Cloud im SOG-Format. SOG („Self-Organizing Gaussian„) ist das PlayCanvas-Format mit Texture-Atlas-Layout und WebP-Komprimierung der quantisierten Daten. Skaliert mit 15–20× besserem Größenverhältnis als PLY. Der Export ruft intern `cwebp` als externes Werkzeug auf — daher in der Sandbox-Variante (App Store) potenziell eingeschränkt.

EINFACH GESAGT

Sehr kleines Format für PlayCanvas-Workflows. Etwa 15–20 Mal kleiner als PLY, weil die Daten in ein Texture-Atlas-Layout gepackt und WebP-komprimiert werden. Wenn du keinen PlayCanvas-Workflow hast, ist SPZ oder Compressed PLY meist die bessere Wahl. Der Export ruft intern `cwebp` als externes Werkzeug auf — in der App-Store-Version (Sandbox) kann dieser Schritt eingeschränkt sein.

M31 Export > Media > Export Orbit Video...

WO

Menüleiste → Export → Media → Orbit Video.

TECHNISCH

Rendert einen 360°-Orbit um das Szenen-Zentrum und encodiert ihn als MP4 (H.264) oder MOV (HEVC, je nach System-Default). Im Unterschied zu M18 (Live-Recording) ist der Pfad hier fest vorgegeben — Dauer wird in den Settings bzw. im Simple-Mode-Export-Step gewählt.

EINFACH GESAGT

Erzeugt automatisch ein Drehvideo um deine Szene. Kein manuelles Bewegen nötig. Gut für Social Media oder eine schnelle Demo. Wenn du die Kamera selbst steuern willst, nutze stattdessen Record Turntable Video (M18). Der Pfad ist fest: ein voller 360°-Orbit um das Szenen-Zentrum, Dauer wählst du in den Einstellungen oder im Simple-Mode-Export-Step. Das Video wird je nach System als H.264-MP4 oder HEVC-MOV ausgegeben.

M32 Export > Media > Export Web Viewer…

WO

Menüleiste → Export → Media → Web Viewer.

TECHNISCH

Verpackt einen Standalone-HTML-Viewer (gsplat.js-basiert) plus die Gaussian-Daten Base64-kodiert in eine einzige `.html`-Datei. Diese Datei läuft offline in jedem modernen Browser — keine Server-Abhängigkeiten, keine externen URLs. Datei-Größe ist etwa Faktor 1.3 größer als die SPZ-Variante (wegen Base64-Overhead).

EINFACH GESAGT

Speichert deine Szene als selbststartende Webseite. Doppelklick auf die HTML-Datei → Browser öffnet sich → fertige interaktive 3D-Szene. Funktioniert ohne Internet, lässt sich per Mail verschicken, ist die einfachste Art, das Ergebnis mit Freunden oder Kunden zu teilen. Die Datei enthält den kompletten gsplat.js-Viewer und die Gaussian-Daten in einem einzigen Dokument — nichts wird aus dem Web nachgeladen. Datei-Größe ist etwa ein Drittel größer als ein SPZ-Export, dafür brauchst du beim Empfänger keine zusätzliche Software.

M33 Export > Photogrammetry > Export SfM (transforms.json)…

WO

Menüleiste → Export → Photogrammetry → Export SfM (transforms.json).

TECHNISCH

Eigener Export-Pfad (nicht über die gemeinsame Helper-Routine), weil keine Gaussian-Cloud, sondern das SfM-Resultat exportiert wird. Öffnet einen Speichern-Dialog mit `transforms.json` als Vorgabe und Content-Type `json`. Bei OK wird eine nerfstudio-kompatible `transforms.json` mit Kamera-Intrinsics, Posen (als 4×4-Matrix in NeRF-Konvention) und Frame-Pfaden geschrieben. Hilfetext im UI weist darauf hin, dass die Trainings-Bilder als Geschwister-Ordner `images/` mitkopiert werden müssen. Aktiv nur, wenn ein SfM-Resultat vorliegt und die Vollversion freigeschaltet ist.

EINFACH GESAGT

Wenn du das SfM-Ergebnis in einer anderen Software wie nerfstudio, Brush, gsplat oder OpenSplat weiterverwenden willst, exportierst du hier die Kamera-Positionen. Lege deine Trainings-Bilder zusätzlich in einen `images/`-Ordner neben der `transforms.json`-Datei — sonst kann das Zielprogramm die Bilder nicht zuordnen. Der Eintrag ist ausgegraut, solange noch kein SfM-Ergebnis existiert, und in der Free-Trial-Version gesperrt. Für den COLMAP-Workspace-Workflow nimm stattdessen M34.

M34 Export > Photogrammetry > Export SfM (COLMAP Workspace)…

WO

Menüleiste → Export → Photogrammetry → Export SfM (COLMAP Workspace).

TECHNISCH

Öffnet einen Speichern-Dialog mit Vorgabe-Name `colmap_workspace` (ohne Extension, weil es ein Ordner ist). Schreibt einen Standard-COLMAP-Workspace mit `sparse/0/cameras.bin`, `images.bin`, `points3D.bin`. Erlaubt es, eine in RadianceKit gerechnete oder importierte SfM-Rekonstruktion in andere Tools wie Postshot, Nerfstudio oder Meshroom zu öffnen, oder bei einem A/B-Re-Run als bereits-gerechnete Eingabe in RadianceKit selbst (über M5) wieder zu laden — spart Rechenzeit. Aktiv nur, wenn ein SfM-Resultat vorliegt und die Vollversion freigeschaltet ist.

EINFACH GESAGT

Wie M33, aber im COLMAP-Format statt nerfstudio. Wenn du Postshot, Meshroom, Nerfstudio oder ein anderes Tool mit COLMAP-Workflow benutzt, ist das dein Export. Ein praktischer Nebeneffekt: Du kannst diesen Ordner später per M5 in RadianceKit zurückladen und dir die SfM-Rechenzeit beim nächsten Lauf sparen — gerade bei großen Szenen ein Zeitgewinn von Stunden. Wie M33 nur verfügbar, wenn ein SfM-Ergebnis vorliegt, und in der Free-Trial-Version gesperrt.

Help-Menü

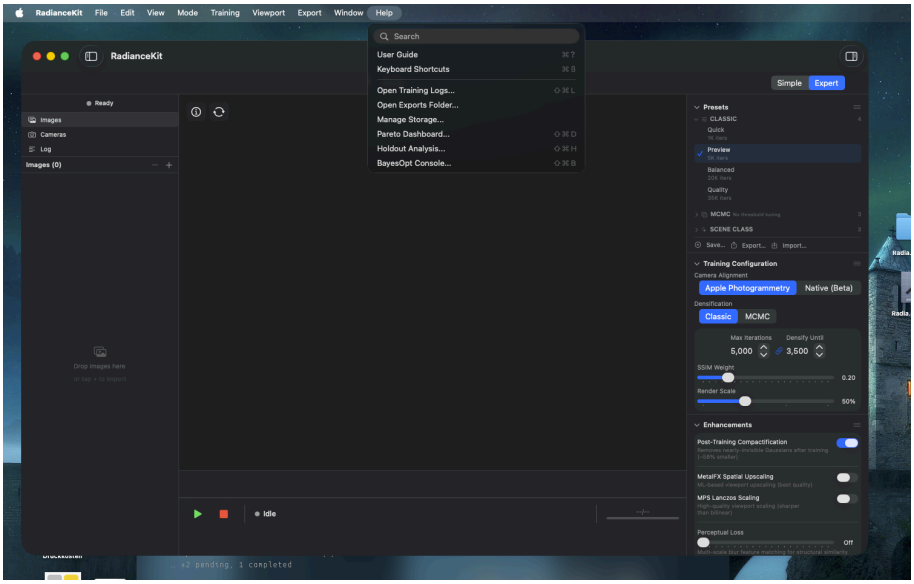

Abbildung 6: Hilfe-Menü mit Dokumentations-, Ordner- und Analyse-Einträgen

Sieben Einträge: zwei Dokumentations-Fenster (User Guide, Keyboard Shortcuts), drei Ordner-Shortcuts (Training Logs, Exports, Storage), und drei Analyse-Fenster (Pareto Dashboard, Holdout Analysis, BayesOpt Console). Apple-typisch erscheint das Help-Menü ganz rechts. Das Standard-Help-Menü wird komplett durch die RadianceKit-eigene Variante ersetzt.

M35 Help > User Guide

WO

Menüleiste → Help → User Guide (⌘?).

TECHNISCH

Öffnet das User-Guide-Fenster. Es zeigt eine Navigation mit Themen-Sidebar und Scroll-Detailbereich bei Default-Größe 860×640. Die Inhalte sind statisch hinterlegt (nicht aus Markdown geparsed).

EINFACH GESAGT

Öffnet die App-interne Anleitung. Wenn du nicht alles in diesem Manual nachlesen willst, findest du dort die wichtigsten Schritte direkt im Programm. Die Anleitung ist als eigenes Fenster mit Themen-Sidebar aufgebaut — du kannst also gezielt zu einzelnen Themen springen. Die Inhalte sind kürzer als dieses Handbuch und konzentrieren sich auf die häufigsten Workflows.

M36 Help > Keyboard Shortcuts

WO

Menüleiste → Help → Keyboard Shortcuts (⌘/).

TECHNISCH

Öffnet das Keyboard-Shortcuts-Fenster — ein einfaches Scroll-Layout mit allen App-Kurzbefehlen, gruppiert nach Top-Level-Menü. Default-Größe 440×560. Inhalte sind ebenfalls statisch hinterlegt.

EINFACH GESAGT

Öffnet ein Fenster mit der kompletten Liste aller Tastatur-Kurzbefehle. Wenn du dir z. B. nicht merken kannst, mit welcher Taste man das Training startet, schaust du da rein. Eine Zusammenfassung steht auch am Ende dieses Kapitels. Die Liste ist nach Top-Level-Menü gruppiert, sodass du schnell zum richtigen Bereich springst. Hilfreich, wenn du dich gerade vom Maus- auf den Tastatur-Stil umstellst.

M37 Help > Open Training Logs...

WO

Menüleiste → Help → Open Training Logs... (⇧⌘L).

TECHNISCH

Berechnet den Log-Ordner als `~/Documents/RadianceKit/Logs`, legt ihn falls nötig an und öffnet ihn in Finder. Jeder Trainings-Lauf schreibt eine eigene JSONL-Datei `training_YYYY-MM-DD_HHmmss.jsonl` dorthin.

EINFACH GESAGT

Öffnet im Finder den Ordner mit allen bisherigen Trainings-Protokollen. Wenn etwas schief gelaufen ist oder du nachsehen willst, wann genau das Training auf welchen Wert konvergierte, findest du das hier in JSONL-Dateien. Pro Trainings-Lauf wird genau eine Datei mit Zeitstempel angelegt — die kannst du auch in andere Tools einlesen oder per Mail an den Support schicken. Wenn du eine grafische Auswertung willst, ist das Pareto Dashboard (M40) der bessere Einstieg.

M38 Help > Open Exports Folder...

WO

Menüleiste → Help → Open Exports Folder...

TECHNISCH

Analog zu M37, aber mit `~/Documents/RadianceKit/Exports`. Wird beim ersten Auto-Test-Lauf oder beim ersten Klick angelegt; danach landen dort die Standardpfade aller Auto-Test-Exporte (z. B. `autotest_<timestamp>.ply`). Manuell über den Speichern-Dialog ausgewählte Exporte gehen NICHT zwingend hier rein, sondern wohin der Nutzer das speichert — daher ist dieser Ordner vor allem für Auto-Tests interessant.

EINFACH GESAGT

Öffnet den Ordner, in dem die App ihre eigenen Exporte ablegt (vor allem Auto-Test-Läufe). Wenn du einen Export manuell mit dem Speichern-Dialog woanders hingelegt hast, ist er dort und nicht in diesem Ordner. Praktisch zum Aufräumen oder um nachzuschauen, wieviel Platz frühere Test-Exporte belegen. Wenn du einen kompletten Überblick inklusive Logs und Szenen-Bundles brauchst, nimm stattdessen Manage Storage (M39).

M39 Help > Manage Storage…

WO

Menüleiste → Help → Manage Storage…

TECHNISCH

Öffnet den Storage-Browser (siehe Kapitel 4 Auxiliary Windows, IDs W7–W12). Listet alle persistierten Szenen, Trainings-Logs, Exporte und Caches im `~/Documents/RadianceKit/`-Ordner mit Größe, ermöglicht Reveal-in-Finder und Move-to-Trash pro Eintrag.

EINFACH GESAGT

Öffnet einen Fenster-Browser, der dir zeigt, wieviel Platz RadianceKit auf deiner Platte belegt — pro Szene, Log und Export. Du kannst direkt einzelne Sachen löschen, ohne in Finder gehen zu müssen. Praktisch nach längerer Nutzung, wenn die Festplatte voll wird — frühere Logs und Auto-Test-Exporte können sich auf mehrere Gigabyte summieren. Per Reveal-in-Finder kommst du jederzeit auch zur klassischen Ansicht.

M40 Help > Pareto Dashboard…

WO

Menüleiste → Help → Pareto Dashboard… (⇧⌘D).

TECHNISCH

Öffnet das Pareto-Dashboard (siehe Kapitel 4, IDs W13–W22). Das Dashboard lädt alle JSONL-Trainings-Logs aus `~/Documents/RadianceKit/Logs/`, ordnet sie nach Szene und Preset und zeichnet einen Pareto-Scatter-Plot (Standard: Loss vs Gaussians, optional Loss vs Wallclock oder PSNR vs Iterationen).

EINFACH GESAGT

Öffnet eine Übersicht aller bisherigen Trainings-Läufe als Diagramm. Du siehst sofort, welcher Lauf die beste Balance aus Qualität und Größe geliefert hat. Praktisch, wenn du verschiedene Presets miteinander vergleichen willst. Standardmäßig zeigt das Diagramm Loss gegen Gaussian-Anzahl — du kannst aber auch nach Wallclock-Zeit oder PSNR umschalten. Die Daten kommen aus den JSONL-Trainings-Logs (M37); je mehr Läufe du hast, desto aussagekräftiger wird die Auswertung.

M41 Help > Holdout Analysis…

WO

Menüleiste → Help → Holdout Analysis… (⇧⌘H).

TECHNISCH

Öffnet das Holdout-Analyse-Fenster (siehe Kapitel 4, IDs W23–W29). Lädt eine `transforms.json`, zeichnet die Kameras als 3D-Globe und erlaubt Train/Test-Fold-Splits (angular oder linear, 2–8 Folds). Output ist eine `fold_assignment.json`, die das Training in den jeweiligen Trainings-Configs als Test-Set verwenden kann.

EINFACH GESAGT

Hilft dir, deine Kamera-Aufnahmen in Trainings- und Test-Sets zu zerlegen — damit du objektiv messen kannst, wie gut deine Szene ist (auf Bildern, die das Training nicht gesehen hat). Eher ein Forschungs- und Benchmark-Werkzeug. Die Kameras werden als 3D-Globe dargestellt; du kannst zwischen 2 und 8 Folds wählen, entweder gleichmäßig im Winkel oder linear über die Reihenfolge. Das Resultat ist eine kleine JSON-Datei, die das Training dann als Test-Set verwendet.

M42 Help > BayesOpt Console…

WO

Menüleiste → Help → BayesOpt Console… (⇧⌘B).

TECHNISCH

Öffnet die BayesOpt-Konsole (siehe Kapitel 4, IDs W30–W39). Lädt vordefinierte Such-Räume (z. B. „MCMC scale-reg + opacity-reg + ssim„), führt Bayesian-Optimization-Trials asynchron aus und zeigt Konvergenzkurve und Trial-Log live.

EINFACH GESAGT

Eine eingebaute Auto-Tuner-Konsole. Statt manuell verschiedene Parameter durchzuprobieren, kann die App das selbst über Nacht laufen lassen und dir am Ende die besten Werte für deine Szene vorschlagen. Sehr fortgeschrittenes Werkzeug — für die meisten Workflows ist ein gutes Preset (siehe Kapitel 7) ausreichend. Du wählst einen vordefinierten Such-Raum (z. B. „MCMC scale-reg + opacity-reg + ssim„), und siehst live die Konvergenzkurve sowie das Trial-Log. Plane je nach Setup mehrere Stunden bis Tage ein.

Hinweis: Cmd-Z im Edit-Menü

Seit Mai 2026 unterstützt der Project Navigator im Expert Mode das Löschen importierter Bilder per Minus-Button oder Backspace-Taste, und das Rückgängigmachen via `Cmd-Z`. Diese Cmd-Z-Aktion erscheint im macOS-Edit-Menü (das von SwiftUI bereitgestellt wird) als „Undo Remove Image„, solange ein gelöschtes Bild noch wiederherstell-

bar ist. Sie wird über das standard-konforme -System registriert, nicht in ; daher gibt es keinen eigenen M-ID-Eintrag in der Inventur.

Tastatur-Kurzbefehle in der Übersicht

Menü-Eintrag	Kurzbefehl
File > Open Scene…	⌘O
File > Save Scene…	⌘S
File > Import COLMAP / Metashape Workspace…	⇧⌘I
File > New Project	⇧⌘N
Mode > Simple Mode	⌘1
Mode > Expert Mode	⌘2
Training > Start Training	⇧⌘T
Viewport > Enter/Exit Edit Mode	⇧⌘E
Viewport > Toggle Auto-Rotation	⌘⌥T
Viewport > Save Screenshot	⇧⌘S
Export > 3D Formats > PLY	⌘E
Help > User Guide	⌘?
Help > Keyboard Shortcuts	⌘/
Help > Open Training Logs…	⇧⌘L
Help > Pareto Dashboard…	⇧⌘D
Help > Holdout Analysis…	⇧⌘H
Help > BayesOpt Console…	⇧⌘B

Edit-Menü (system-bereitgestellt, im Expert Mode bei aktiver Project-Navigator-Auswahl):

Aktion	Kurzbefehl
Undo Remove Image	⌘Z
Remove Selected Image	Backspace / Delete

KAPITEL

Kapitel 2 — Inspector (Expert View)

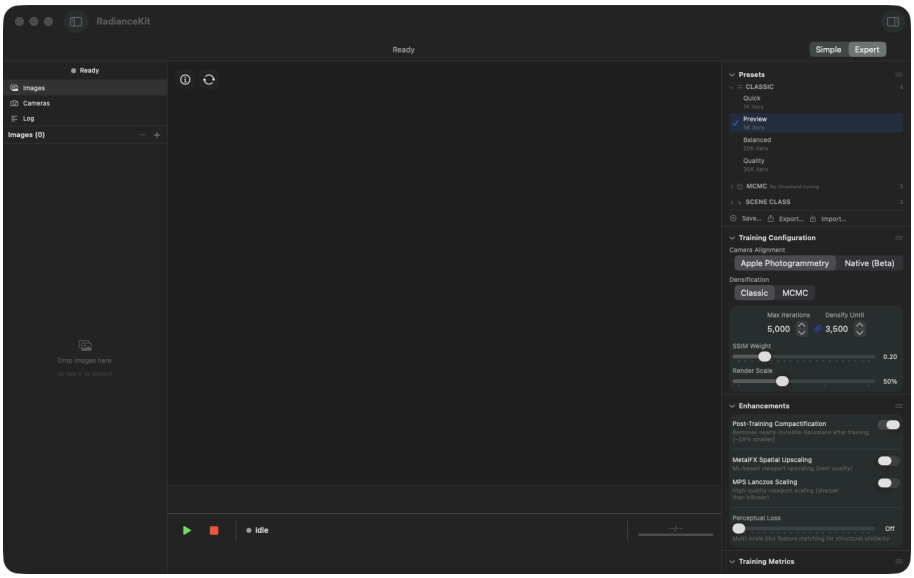

Abbildung 7: Expert-Modus leer — Project Navigator links (Images 0, Cameras, Log), Empty-Viewport mittig, Inspector rechts mit Presets/Training Configuration/Enhancements/Training Metrics-Sektionen

Leerer Inspector vor Import: Linke Sidebar zeigt Images-Counter 0 und Drop-Hint „Drop images here / or tap + to import„. Inspector rechts ist voll funktional, aber Presets sind nur informativ (kein aktives Training). Default-Preset „Preview“ (5K iters) ist markiert. Camera-Alignment auf Apple Photogrammetry, Densification Classic, SSIM Weight 0.20, Render Scale 50 %. Empty-States in Training Metrics („Start training to see live metrics„) und Loss History („Loss curve will appear during training“).

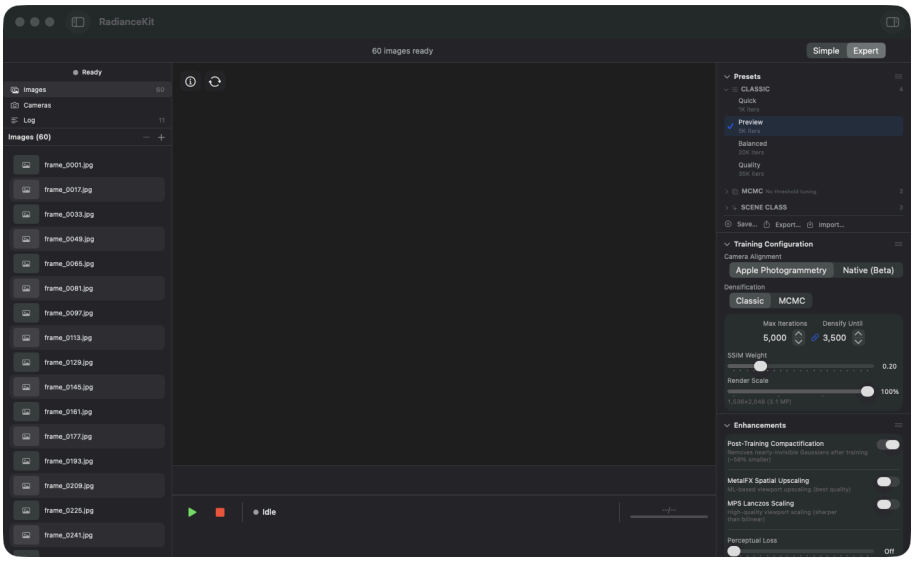

Abbildung 8: Inspector mit 60 flowers-Bildern geladen — Image-Sidebar zeigt erste Dateinamen frame_0001.jpg ff, Header „60 images ready„

Inspector nach Import: Header-Status „60 images ready„. Image-Sidebar listet alle 60 importierten Frames (`frame_0001.jpg` bis `frame_0945.jpg` , jeder 16. Frame des 960-Cam-Bouquet-Datasets als Subset für schnelle Iterationen). Auto-Render-Scale-Logik prüft die Bild-Auflösung (1536×2048 = 3.1 MP) und passt Render Scale entsprechend an. Play-Button (grün, unten links) ist jetzt aktiv und startet das Training mit dem aktiven Preset.

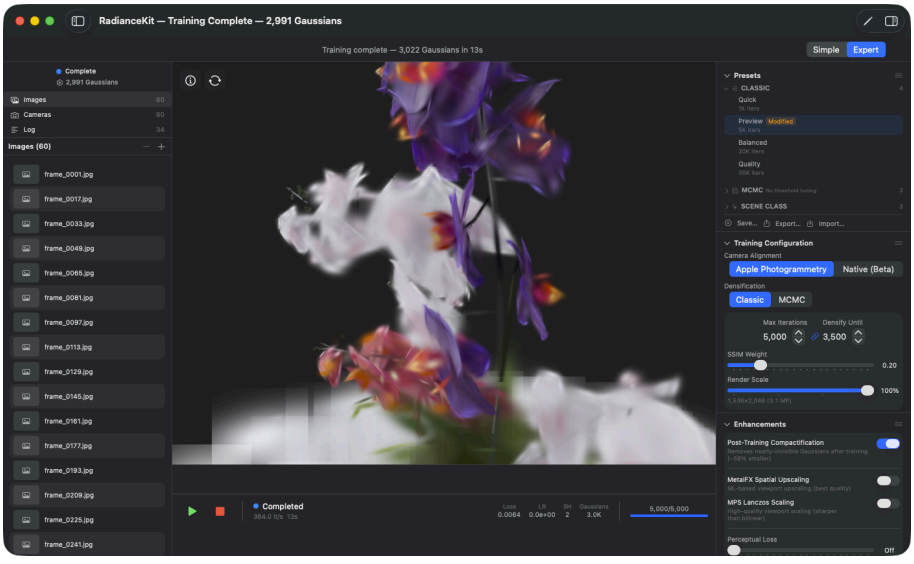

Abbildung 9: Inspector mid-training — Live-Viewport zeigt flowers-Bouquet-Rekonstruktion, Metrik-Bar unten (Loss / LR / Gaussian-Count / Iterationen), Preset-Card „Preview„ mit „Modified"-Badge falls Parameter angepasst

Inspector während Training: Titel-Bar zeigt globalen Fortschritt „RadianceKit — Training NN %„. Viewport rendert die laufende Gaussian-Rekonstruktion in Echtzeit (alle 50 Iterationen aktualisiert — Live-Preview-Intervall einstellbar in Settings → General → Training → Live Preview). Metrik-Bar unter dem Viewport: aktueller Loss, Learning Rate, Gaussian-Count und Iterationen-Counter (z.B. 1,600/5,000 bei Preview-Preset).

Inspector-Preset-Card „Preview“ trägt „Modified„-Badge, sobald irgendein Parameter vom Built-in-Default abweicht. Sidebar „Log“ sammelt SfM- und Training-Stage-Events.

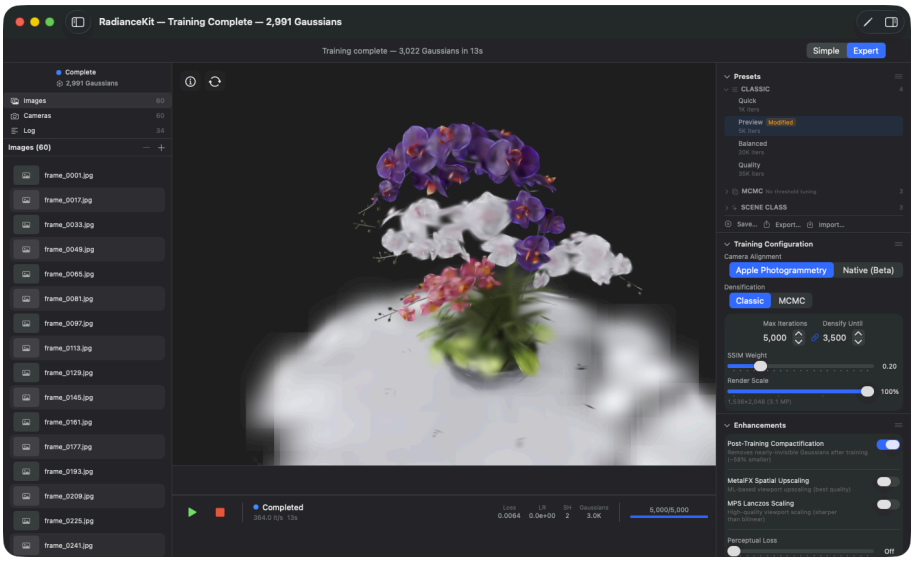

Abbildung 10: Inspector nach Training-Abschluss — Viewport zeigt fertige flowers-Bouquet-Rekonstruktion (2,991 Gaussians nach 5K Iterationen in 13s), Titel-Bar „Training Complete — 2,991 Gaussians„

Inspector nach Training: Titel-Bar zeigt finale Gaussian-Anzahl (hier 2 991 — sehr kompakt, weil die synthetische Blender-Bouquet-Szene auf hellem Hintergrund einfache Geometrie hat). Viewport zeigt die fertige Punktwolke — orbitale Drag-Navigation aktiv (rotiert um den Szenen-Mittelpunkt). Training-Metrik-Sektion ist jetzt mit Final-Werten gefüllt, Loss-History-Chart zeigt den Verlauf der gesamten 5 000 Iterationen. Export-Sektion unten ist jetzt aktiv (alle Format-Buttons enabled).

Der Inspector ist die rechte Seitenleiste im Expert Mode (⌘2). Er bündelt sämtliche trainingsrelevanten Parameter in sieben einklappbaren Sektionen. Die Default-Reihenfolge von oben nach unten beim ersten Start ist: Look, Presets, Trainingskonfiguration, Metriken, Verlust-Diagramm, Enhancements und Export. Die „Look„-Sektion (post-Training-Bildanpassungen) ist die reale UI-Umbenennung der früheren „Finishing“-Sektion — ihr interner Enum-`rawValue` bleibt aus Persistenz-Gründen „Finishing„, die angezeigte Überschrift heißt „Look“. Jede Sektion lässt sich per Klick auf den Header zuklappen, die Reihenfolge per Drag-and-Drop neu anordnen (`InspectorView.swift:81–97`). **Beim ersten Start sind alle sieben Sektionen eingeklappt** (`inspectorCollapsedSections` defaultet auf `Set(InspectorSection.allCases)`); der App-State speichert die Klapp- und Reihenfolge-Präferenzen danach über App-Starts hinweg.

Eine Reihe von Bedienelementen aus dem Inspector taucht in fast identischer Form auch in den Einstellungen (Kapitel 3) auf — typischerweise SfM-Backend, Sky-Masking und ähnliche Defaults. Die Trennung ist bewusst: Die Einstellungen liefern die App-globale Vorlage für neu angelegte Projekte, der Inspector überschreibt diese Werte für das aktuell offene Projekt. Wer einmal die Bedienlogik der einen Seite kennt, kann die andere blind benutzen.

Die linke Spalte im Expert Mode — der Project Navigator — gehört nicht zum Inspector, ist aber sein direkter Nachbar. Dort lassen sich importierte Bilder per Klick auswählen, mit Leertaste in Quick-Look ansehen und über den Minus-Button oder die Entfernen-

Taste löschen (mit Cmd-Z zum Rückgängigmachen). Der Inspector folgt der aktuellen Sidebar-Auswahl mit kontextspezifischen Detail-Informationen, die sieben Hauptsektionen bleiben aber immer verfügbar.

Look-Sektion (L1–L5)

Die Look-Sektion (interner `rawValue` weiterhin „Finishing„) ist die oberste Inspector-Sektion und sammelt die **post-Training** -Bildanpassungen an einem Ort. Alle Regler arbeiten **nicht destruktiv**: Jeder Slider wendet den `FinishingPass` erneut auf einen unveränderten Pristine-Snapshot (Original-DC-Farbe, -Opacity, -Skalierung) an — die Anpassung ist damit **idempotent**, nicht kumulativ. Das Ergebnis erscheint **live im Viewport** (WYSIWYG, exakt so wie der spätere Export) und wird in **jeden Export eingebacken**. Die Sektion ist erst **nach Abschluss eines Trainings-Laufs** verfügbar (vorher steht „Available after a training run completes.“); ihre Werte werden **bei jedem neuen Training zurückgesetzt**. Solange ein Export läuft, sind alle Regler **gesperrt** — ein Lock-Hinweis „Locked while exporting — the file uses the current settings.„ erscheint und die GroupBox ist disabled.

L1 Saturation-Slider

WO

Inspector → Look-Sektion → GroupBox → Saturation.

TECHNISCH

Slider 0.5–1.2, Anzeige zweistellig (z.B. „1.00”). Skaliert die SH-DC-Chroma jedes Splats um den Luminanz-Wert: 1.0 = unverändert, < 1.0 = entsättigt (Farbe zur Graustufe gezogen), > 1.0 = kräftiger. Mathematisch wird die DC-Farbe aus dem Pristine-Snapshot zurückgerechnet (`desaturateDC`), sodass wiederholtes Schieben nicht aufsummiert. Wurde auf DJI-Drohnen-Material validiert (Pensford-Viadukt), das tendenziell überzeichnet — der Drohnen-Default liegt bei 0.82. Wirkt nur auf die Farb-Basis (SH-Grad 0), höhere SH-Koeffizienten bleiben unberührt.

EINFACH GESAGT

Wie kräftig die Farben des fertigen Splats sind. 1.00 lässt alles wie trainiert, Werte darunter ziehen die Farbe Richtung Grau — gut für Drohnen- oder Video-Material, das oft übersättigt rauskommt. Werte über 1.0 machen es kräftiger. Du kannst beliebig hin- und herschieben, ohne dass sich etwas „aufschaukelt„, weil die App immer vom unveränderten Original-Stand neu rechnet. Live im Viewport sichtbar und genau so im Export.

L2 Splat length-Slider

WO

Inspector → Look-Sektion → GroupBox → Splat length.

TECHNISCH

Slider 0.3–1.0, Anzeige zweistellig. Zieht die drei Skalierungs-Achsen jedes Gaussians im Log-Raum zu ihrem Mittelwert hin (`shortenScale` , Faktor `alpha`): 1.0 = unverändert, kleinere Werte machen längliche „Nadel„-Splats runder, 0 wären reine Kugeln. Greift nadel-artige, überdehnte Splats an, ohne die Gesamt-Größe zu verändern, und reduziert dadurch typische „Konfetti"-Artefakte. Vom Pristine-Snapshot (Original-Log-Skalierung) aus angewendet, daher idempotent. Kommutiert mit Splat size (L3), weil beides im Log-Raum arbeitet.

EINFACH GESAGT

Macht überlange, splittrige Splats runder. 1.00 lässt die Form wie trainiert, niedrigere Werte stauchen die langgezogenen „Nadeln„ zu rundlicheren Klecksen — das beruhigt körnige, von Konfetti-Artefakten geplagte Rekonstruktionen. Die Gesamt-Größe bleibt gleich, es geht nur um die Länglichkeit. Lässt sich gefahrlos mit Splat size (L3) kombinieren.

L3 Splat size-Slider

WO

Inspector → Look-Sektion → GroupBox → Splat size.

TECHNISCH

Slider 0.5–2.0, Anzeige zweistellig. Skaliert jeden Gaussian auf **allen** drei Achsen uniform (`sizeScale`): 1.0 = unverändert, < 1.0 = kleiner/dichter/schärfer, > 1.0 = größer/„fluffiger„ (füllt Lücken zwischen den Splats). Da die Skalierungen im Log-Raum liegen, wird die Multiplikation als additiver `log(factor)` -Offset realisiert — das kommutiert mit Splat length (L2), weil ein konstanter Offset die Abweichung-vom-Mittelwert unangetastet lässt. Vom Pristine-Snapshot aus, also idempotent. Neu in dieser Version.

EINFACH GESAGT

Skaliert alle Splats gleichmäßig größer oder kleiner. 1.00 ist der trainierte Zustand, Werte darunter machen die Punktwolke enger und schärfer, Werte darüber decken Lücken zwischen den Splats zu (wirkt weicher/„fluffiger„). Praktisch, um eine löchrige Rekonstruktion optisch zu schließen oder umgekehrt mehr Detail freizulegen. Verträgt sich problemlos mit Splat length (L2) — beide Regler beeinflussen sich nicht gegenseitig.

L4 Fade far region (mit Sub-Slidern)

WO

Inspector → Look-Sektion → GroupBox → Toggle „Fade far region„ plus die Sub-Slider „Fade start ×radius“ und „Fade floor„.

TECHNISCH

Toggle, der einen radialen Opacity-Abfall mit der Distanz vom Kamera-Schwerpunkt aktiviert — die schwach beobachteten „Far-Konfetti„ im Hintergrund werden ausgeblendet. **Nur für Orbit-Aufnahmen**: Der Toggle ist disabled, wenn `finishingContext.fadeEligible` false ist (lineare Flüge, zu wenige oder degenerierte Kameras); dann erscheint statt der Sub-Slider der Hinweis „Far-fade applies only to orbit captures (not this scene).“ Die Eignung wird per Azimut-Abdeckung der Kamera-Positionen ermittelt (ein Orbit umkreist den Schwerpunkt und füllt viele Kompass-Sektoren, ein Linearflug nur ~2). Zwei Sub-Slider steuern die Geometrie: **Fade start ×radius** (1.0–3.0) setzt den Innenradius als Vielfaches des Orbit-Radius, innerhalb dessen volle Opacity gilt; **Fade floor** (0.0–1.0) ist der Opacity-Faktor weit jenseits des Fade-Radius. Wichtig: Der Fade **überspringt den Sky-Dome-Bereich** (die frozen Gaussians der Indizes [0, frozenCount)), damit die absichtliche Hintergrund-Kuppel nicht mitgedimmt wird.

EINFACH GESAGT

Blendet die schwammigen Reste am äußeren Szenen-Rand aus — genau die „Far-Konfetti„-Klümpchen, die bei Rundum-Aufnahmen weit hinten schweben. Funktioniert nur bei echten Orbit-/Umkreisungs-Aufnahmen; bei geraden Drohnen-Flügen oder zu wenigen Kameras ist der Schalter ausgegraut und ein Hinweis erklärt warum. Ist er aktiv, kommen zwei Feinregler dazu: „Fade start ×radius“ legt fest, ab welcher Entfernung (als Vielfaches des Umkreis-Radius) das Ausblenden beginnt, „Fade floor„ wie stark die fernen Splats am Ende noch sichtbar bleiben (0 = ganz weg, 1 = unverändert). Ein bewusst rekonstruierter Sky-Dome (I44) wird dabei nie angefasst — der Himmel bleibt erhalten.

L5 Reset finishing-Button

WO

Inspector → Look-Sektion → GroupBox → „Reset finishing„ (unten, kleiner Button).

TECHNISCH

Setzt alle Look-Settings auf die Defaults zurück (`FinishingPass.Settings()` = Saturation 1.0, Fade aus, Splat length 1.0, Splat size 1.0) und löst sofort ein erneutes Finishing aus, sodass der Viewport auf den unveränderten trainierten Zustand zurückspringt. `controlSize(.small)`. Da der ganze Look-Stack idempotent vom Pristine-Snapshot aus rechnet, ist „zurück auf Default„ exakt der ursprüngliche Trainings-Output — kein Qualitätsverlust durch mehrfaches Hin und Her. Wie alle Regler der Sektion während eines laufenden Exports gesperrt.

EINFACH GESAGT

Stellt mit einem Klick alle Look-Regler auf Standard zurück (Saturation 1.00, Fade aus, beide Splat-Slider auf 1.00) — der Viewport zeigt danach wieder exakt das frisch trainierte Ergebnis. Praktisch, wenn du dich verspielt hast und sauber von vorne anfangen willst. Weil die App immer vom Original-Stand rechnet, gibt's dabei keinen Qualitätsverlust. Während ein Export läuft, ist der Button (wie die Slider) gesperrt.

Presets-Sektion (I1–I11)

Die Presets-Sektion ist der schnellste Weg, eine getestete Konfiguration anzuwenden. Built-in-Presets (Capture Class, Classic, MCMC, Hybrid) liefern reproduzierbare Startpunkte aus 560+ dokumentierten Experimenten; eigene Presets lassen sich speichern, exportieren, importieren und teilen. Die Liste ist nach Kategorien gruppiert (Capture Class, Classic, MCMC, Hybrid, Custom) und mehr als eine Kategorie kann gleichzeitig aufgeklappt sein. Über den Kontextmenü-Mechanismus (Rechtsklick auf eine Zeile) sind Export, Duplizieren und — bei eigenen Presets — Löschen erreichbar.

I1 Save…-Button

WO

Inspector → Presets-Sektion → Save…-Button (Action-Leiste unten).

TECHNISCH

Öffnet ein Popover mit Textfeld und Save-/Cancel-Buttons. Der aktuelle TrainingConfig-Zustand wird als neues benutzerdefiniertes Preset persistiert (JSON-codiert, App-übergreifend gespeichert). Der Save-Vorgang kopiert alle 81 Trainings-Parameter plus die aktuelle Densification-Strategie. Das Preset landet automatisch in der Kategorie Custom, unabhängig davon, von welchem Built-in-Preset es abgeleitet wurde. Leere Namen und reine Whitespace-Eingaben werden verworfen. Schon existierende Namen werden nicht abgewiesen — jedes Preset hat eine eigene interne ID, doppelte Namen sind technisch erlaubt, aber praktisch verwirrend.

EINFACH GESAGT

Sichert deine aktuelle Konfiguration als wiederverwendbares Preset. Drück den Button, gib im Popover einen Namen ein und klick Save — alle 81 Parameter inklusive Densification-Strategie landen unter dem gewählten Namen in der Custom-Kategorie. Brauchst du, wenn du dir Mühe gegeben hast und nicht beim nächsten Projekt wieder von vorne fummeln willst. Besonders praktisch für wiederkehrende Setups wie „Drohne 4K„ oder „Indoor schnell“. Doppelte Namen sind technisch erlaubt, aber praktisch verwirrend — nimm lieber etwas Sprechendes.

I2 Preset Name TextField

WO

Save-Popover → Textfeld „Preset Name„.

TECHNISCH

Einfaches Textfeld mit gerundetem Rahmen, breiter Form. Der Wert wird beim Klick auf den Save-Button als Preset-Name übernommen. Keine Längenbegrenzung im UI, aber der gespeicherte Name muss JSON-codierbar und in den UI-Listen darstellbar sein — Emoji und Umlaute funktionieren. Der Inhalt wird beim Öffnen des Popovers automatisch auf einen leeren String zurückgesetzt. Der Save-Button bleibt disabled, solange das Feld nach Trim leer ist. Es gibt kein Auto-Suggest und keine Vorbelegung mit dem Namen des gerade aktiven Presets.

EINFACH GESAGT

Hier tippst du den Namen für dein Preset ein. Wähle was Sprechendes wie „Drohne 4K 30fps„ oder „Innenraum schnell“ — das hilft dir später beim Wiederfinden in der Custom-Kategorie. Emoji und Umlaute sind erlaubt, eine harte Längenbegrenzung gibt’s nicht. Solange das Feld leer ist oder nur aus Leerzeichen besteht, bleibt der Save-Button ausgegraut. Beim erneuten Öffnen des Popovers ist das Feld wieder leer — es gibt keine Vorbelegung mit dem aktiven Preset-Namen.

I3 Cancel-Button (Save-Dialog)

WO

Save-Popover → Cancel-Button (links).

TECHNISCH

Schließt den Popover ohne Speichern. Verwirft den Textfeld-Inhalt — beim nächsten Öffnen wird er wieder durch die Save…-Button-Logik (I1) auf leer zurückgesetzt. Standard-Button-Style, keine Bestätigungs-Dialoge, keine Hotkeys. Die aktuelle TrainingConfig bleibt unverändert, da der Save-Pfad gar nicht ausgeführt wurde.

EINFACH GESAGT

Schließt den Save-Popover, ohne irgendetwas zu speichern. Falls du dir's anders überlegt hast, dich vertippt oder den Dialog versehentlich geöffnet hast — einfach Cancel klicken. Deine aktuelle Trainingskonfiguration bleibt unverändert, weil noch gar nichts geschrieben wurde. Beim nächsten Öffnen des Popovers startet das Namensfeld wieder leer. Keine Sicherheitsabfrage, kein Hotkey — einfach Klick und weg.

I4 Save-Button (Save-Dialog)

WO

Save-Popover → Save-Button (rechts, prominenter Stil).

TECHNISCH

Triggert die eigentliche Persistierung. Validiert nochmal nicht-leerer Name (defensiver Check) und schreibt dann die aktuelle TrainingConfig als JSON in den App-Speicher. Schließt anschließend den Popover. Blau hervorgehoben, ausgegraut solange das Textfeld leer ist. Wenn das Speichern fehlschlägt (z.B. weil der App-Speicher voll ist — sehr unwahrscheinlich), gibt es momentan keinen sichtbaren Fehlerdialog; das Preset würde dann beim nächsten App-Start einfach nicht erscheinen.

EINFACH GESAGT

Mit Klick auf Save übernimmst du den Namen und schreibst dein aktuelles Setup als neues Preset weg. Der Popover schließt sich, das Preset taucht sofort in der Custom-Kategorie der Preset-Liste auf und kann ab jetzt per Klick aktiviert werden. Der Button ist blau hervorgehoben (`borderedProminent`) und bleibt ausgegraut, solange das Namensfeld leer ist. Falls das Speichern fehlschlägt (z.B. UserDefaults voll), gibt's keinen sichtbaren Fehlerdialog — das Preset würde dann beim nächsten App-Start einfach fehlen.

I5 Export…-Button

WO

Inspector → Presets-Sektion → Action-Leiste → Export…-Button.

TECHNISCH

Exportiert das aktuell ausgewählte Preset als `.radiancepreset`-Datei (intern JSON). Disabled, wenn kein Preset selektiert ist. Beim Klick öffnet die App einen Save-Dialog mit vorgegebenem Dateinamen (Preset-Name + `.radiancepreset`-Extension). Das gespeicherte Format enthält die komplette TrainingConfig plus Metadaten (Name, Kategorie, ID, Built-in-Flag). Doppelklick im Finder öffnet die App — aber **nicht** automatisch den Import; der Anwender muss den Import-Button (I6) verwenden.

EINFACH GESAGT

Wähle ein Preset in der Liste an und klick Export — dann kannst du es als `.radiancepreset`-Datei speichern und z.B. einem Kollegen schicken oder auf einen zweiten Mac übertragen. Der Empfänger lädt's drüben mit dem Import…-Button (I6) wieder ein. Funktioniert für Built-ins und für deine eigenen Custom-Presets gleich gut. Der Button ist ausgegraut, solange in der Liste nichts angeklickt ist. Tipp: Über das Kontextmenü (I8) geht's noch schneller — da musst du das Preset nicht erst selektieren.

I6 Import…-Button

WO

Inspector → Presets-Sektion → Action-Leiste → Import…-Button.

TECHNISCH

Öffnet einen Datei-Dialog, der nur `.radiancepreset`-Dateien zulässt (Mehrfachauswahl deaktiviert). Beim Auswählen wird die JSON-Datei geladen, validiert und in die Custom-Kategorie eingefügt — mit neuer interner ID, damit keine Kollisionen mit Built-ins entstehen. Der Import setzt automatisch die Kategorie auf Custom, selbst wenn das exportierte Preset ursprünglich z.B. ein Built-in war. Beschädigte oder mit einer älteren Schema-Version inkompatible Dateien werden stillschweigend abgewiesen, ohne Fehlerdialog (Console-Log gibt aber Auskunft).

EINFACH GESAGT

Eine `.radiancepreset`-Datei von Festplatte einlesen. Nützlich, wenn dir jemand ein erprobtes Setup schickt oder du selbst deine Lieblings-Presets über mehrere Macs synchron halten willst. Importierte Presets landen immer in der Custom-Kategorie — auch wenn sie ursprünglich aus den Built-ins exportiert wurden. Beschädigte oder veraltete Dateien werden stillschweigend ignoriert; im Konsolen-Log steht dann der Grund. Mehrfachauswahl im Dialog ist deaktiviert, also pro Klick nur eine Datei.

I7 Preset-Zeile (Klick-Aktivierung)

WO

Inspector → Presets-Sektion → jede Preset-Zeile in jeder Kategorie.

TECHNISCH

Klick auf eine Preset-Zeile ersetzt alle Felder der TrainingConfig durch die Werte aus dem Preset, merkt sich die ID des aktiven Presets und setzt den Modified-Status zurück. Das Aktiv-Häkchen vor der Zeile erscheint nur, wenn das Preset ausgewählt UND unmodifiziert ist. Sobald ein Wert in der TrainingConfig geändert wird (Slider, Stepper, Toggle in den anderen Inspector-Sektionen), erscheint ein orangenes „Modified„-Badge hinter dem Namen. Eingebaute Presets können nicht überschrieben werden — bei Modifikation muss via Save-Button (I1) eine eigene Kopie angelegt werden.

EINFACH GESAGT

Klick auf eine Zeile aktiviert das Preset und übernimmt alle dort gespeicherten Werte in die aktuellen Trainings-Einstellungen. Das Häkchen vor dem Namen zeigt, welches Preset gerade aktiv ist. Sobald du danach irgendeinen Slider, Stepper oder Toggle in den anderen Sektionen verstellst, erscheint hinter dem Namen ein orangenes „Modified„-Badge — weil dein Setup jetzt vom Preset abweicht. Built-in-Presets lassen sich nicht überschreiben; wenn du Änderungen behalten willst, leg via Save…-Button (I1) eine eigene Kopie an oder dupliziere das Preset (I9).

I8 Context-Menü „Export…„

WO

Rechtsklick auf jede Preset-Zeile → erster Eintrag „Export…„.

TECHNISCH

Identische Funktionalität wie I5 (Export…-Button), aber bequemer erreichbar — ohne dass das Preset vorher selektiert sein muss. Exportiert direkt das in der Zeile angeklickte Preset. Funktioniert für alle Preset-Kategorien gleich (Built-in oder Custom), keine Einschränkung. Der Export enthält das Built-in-Flag und die Original- Kategorie, aber beim Re-Import wird die Kategorie wie unter I6 beschrieben auf Custom gemappt.

EINFACH GESAGT

Schneller Weg zum Exportieren — Rechtsklick auf das gewünschte Preset und „Export… „ auswählen. Spart den Umweg über Vorher-Anklicken und dann den Export…-Button drücken. Funktioniert für alle Kategorien gleich, auch für Built-ins. Die erzeugte `.radiancepreset`-Datei ist identisch zu der von I5; beim späteren Re-Import landet sie automatisch in der Custom-Kategorie.

I9 Context-Menü „Duplicate„

WO

Rechtsklick auf jede Preset-Zeile → zweiter Eintrag „Duplicate„.

TECHNISCH

Klont das Preset in die Custom-Kategorie. Erzeugt eine neue interne ID, hängt „ Copy„ an den Namen an und speichert die Kopie. Funktioniert auch für Built-in-Presets — der Klon ist dann editierbar. Das Original bleibt unangetastet. Die TrainingConfig wird Wert-für-Wert kopiert (JSON-Roundtrip), sodass keine Referenz-Bindungen zwischen Original und Kopie bestehen.

EINFACH GESAGT

Erzeugt eine bearbeitbare Kopie eines Presets in der Custom-Kategorie. Praktisch, wenn du z.B. das Built-in „Quality„-Preset als Ausgangspunkt willst und dann nur den SSIM-Slider ein wenig verschieben magst. Workflow: Duplizieren, neu benennen (Kontextmenü oder neuer Save…-Lauf), anpassen, fertig. Das Original bleibt unangetastet — du kannst jederzeit darauf zurück. Funktioniert auch für Built-ins, was der einzige Weg ist, deren Werte als Basis zu übernehmen und gleichzeitig editierbar zu machen.

I10 Context-Menü „Delete„

WO

Rechtsklick auf eigene Preset-Zeilen → letzter Eintrag „Delete„ (rot, destructive).

TECHNISCH

Nur sichtbar für Custom-Presets. Built-ins lassen sich nicht löschen. Der Eintrag ist als destruktiv markiert, erscheint im Kontextmenü rot und wird hinter einem Divider abgesetzt, damit man ihn nicht versehentlich klickt. Es gibt **keinen** Confirmation-Dialog — ein Klick löscht das Preset sofort. Das gelöschte Preset ist nicht wiederherstellbar (Cmd-Z funktioniert hier nicht — Undo gibt es im aktuellen Build nur für die Bild-Liste, nicht für Preset- Operationen). War das gelöschte Preset gerade aktiv, bleibt die aktuelle TrainingConfig unverändert, nur die aktive Preset-Auswahl wird genullt.

EINFACH GESAGT

Eigene Presets löschen. Bei den Built-ins (Quick, Preview, Balanced, Quality, Ultra Detail, Drone / Aerial, 360° Walkaround, Photo / Object usw.) ist „Delete„ gar nicht sichtbar — die kannst du nicht versehentlich killen. Achtung: Es gibt keine Sicherheitsabfrage und kein Undo, ein Klick und das Preset ist weg. Wenn du dir nicht sicher bist, vorher per Export… (I5/I8) eine Sicherheitskopie auf Platte ziehen — die kannst du jederzeit wieder importieren. War das Preset gerade aktiv, bleibt deine TrainingConfig unverändert, nur das Häkchen verschwindet.

I11 Kategorie-Header (Aufklappen/Einklappen)

WO

Inspector → Presets-Sektion → jeder Kategorie-Header (Capture Class, Classic, MCMC, Hybrid, Custom).

TECHNISCH

Klapp-Status pro Kategorie mit unterschiedlichem Default: die kuratierte Gruppe Capture Class startet **aufgeklappt**, Classic, MCMC, Hybrid und Custom starten **zugeklappt**. Der Status wird nicht persistiert — bei App-Neustart sind alle Kategorien wieder im Default-Zustand. Der Chevron-Pfeil rotiert animiert. Die Zahl rechts im Header zeigt die Anzahl der Presets in dieser Kategorie. Die Klick-Hit-Area umfasst den ganzen Header-Bereich.

EINFACH GESAGT

Kategorien ein- und ausklappen, um die Preset-Liste übersichtlich zu halten. Beim App-Start ist die Capture-Class-Gruppe offen, Classic, MCMC, Hybrid und Custom sind zu. Klick auf den Header (kompletter Bereich ist klickbar) und die Liste fährt mit kurzer Chevron-Animation auf oder zu. Die kleine Zahl rechts zeigt, wie viele Presets in der Kategorie liegen. Nach Neustart der App ist wieder der Default-Zustand da — die App speichert diese Klapp-Einstellung bewusst nicht.

Trainingskonfiguration-Sektion (I12–I22)

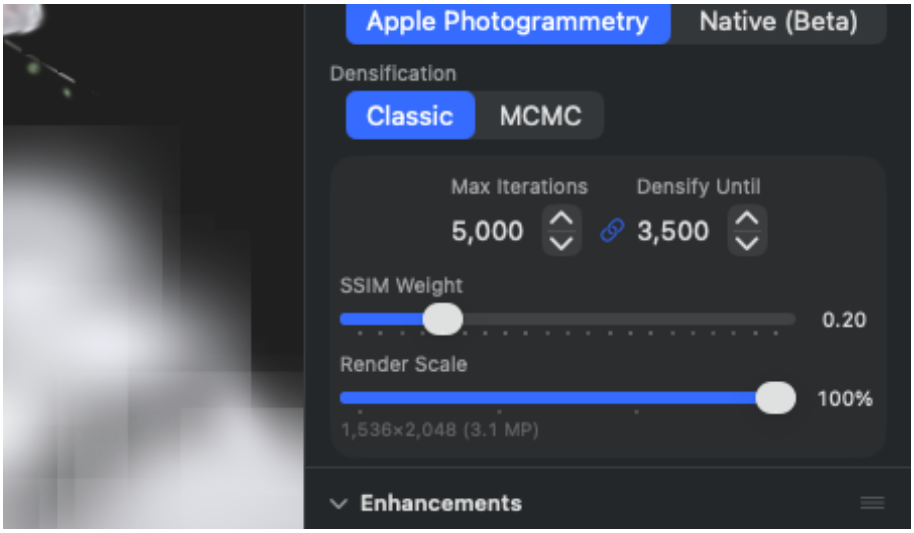

Abbildung 11: Crop nur Trainingskonfiguration-Sektion — Camera Alignment (Apple Photogrammetry aktiv, Native (Beta) inaktiv), Densification (Classic aktiv), Max Iterations 5,000 / Densify Until 3,500 mit Link-Symbol, SSIM Weight Slider 0.20, Render Scale Slider auf 100 % (1,536×2,048 = 3.1 MP)

Hier landen die zentralen Hebel: welches SfM-Backend rechnen soll, wie die Densification arbeitet, wieviele Iterationen, wie groß die SSIM-Gewichtung. Bei MCMC-Strategie tauchen zwei zusätzliche Toggles auf („MCMC Quality„ und „Auto-scale by scene“), die im Classic-Modus ausgeblendet werden. Bei Native-SfM-Backend kommt das FOV-Override-Feld dazu, das nur für Video-Frames ohne EXIF-Brennweite gebraucht wird.

I12 Camera Alignment-Picker

WO

Inspector → Trainingskonfiguration → Camera Alignment (segmentierter Picker oben).

TECHNISCH

Segmentierter Picker mit zwei Optionen: Apple Photogrammetry und Native (Beta). Die Auswahl bestimmt das verwendete SfM-Backend bei der nächsten Kamera-Rekonstruktion. Sie beeinflusst zugleich, welche weiteren Inspector-Elemente sichtbar sind: Native zeigt zusätzlich das FOV-Override (I13), das nur bei EXIF-losen Video- Frames gebraucht wird. Hinweis: für sehr große Outdoor-Aufnahmen kannst du das Ergebnis eines externen Tools (Metashape oder COLMAP) per Workspace-Import einspielen — siehe Kapitel 1 (M5) und Kapitel 9 (Q3, Q6).

EINFACH GESAGT

Hier wählst du, wie die Kamera-Positionen rekonstruiert werden — der wichtigste Schalter für die Endqualität. Apple Photogrammetry ist der schnelle Standard und reicht für die meisten Objekt-Scans völlig aus. Native (Beta) ist die App-Store-konforme Eigenentwicklung, gut für Orbits und Turntable-Szenen, und braucht bei EXIF-losen Video-Frames das FOV-Override (I13). Bei sehr großen Outdoor- Sets kannst du die Kameras alternativ in Metashape oder COLMAP rechnen und das Ergebnis über den Workspace-Import laden. Details und Empfehlungen pro Szenentyp findest du in Kapitel 9.

I13 FOV Override-Feld (Native SfM)

WO

Inspector → Trainingskonfiguration → FOV Override (nur sichtbar bei Camera Alignment = Native).

TECHNISCH

Numerisches Textfeld (Range 0-170°), Default 0 = automatische Bestimmung aus EXIF oder Heuristik. Die manuelle Eingabe ist nötig, wenn die Eingabe-Bilder aus einem Video extrahiert wurden, das keine Brennweiten-Metadaten enthält. Typische Werte: iPhone Wide ≈ 73°, DJI Mavic Wide-Crop ≈ 70°, Drohne mit Vollformatsensor ≈ 84°. Der Wert wird auf [0, 170] geclampt — Werte außerhalb werden direkt zurückgestaucht. Wirkt sich nur auf die native SfM-Pipeline aus (Q4/Q5); Apple Photogrammetry ignoriert diesen Wert komplett.

EINFACH GESAGT

Wenn deine Bilder kein EXIF haben (typisch bei extrahierten Video-Frames), trägst du hier die horizontale Sichtweite der Kamera in Grad ein. Faustwerte: iPhone Wide ≈ 73°, DJI Mavic Wide-Crop ≈ 70°, Drohne mit Vollformatsensor ≈ 84°. Eine 0 lässt die App selbst raten — das geht oft gut, kann aber bei seltenen Objektiven schiefgehen. Werte über 170° werden automatisch zurückgestaucht. Das Feld ist nur sichtbar und nur wirksam, wenn du Native als Camera Alignment (I12) gewählt hast — Apple Photogrammetry ignoriert es komplett.

I15 Densification-Picker

WO

Inspector → Trainingskonfiguration → Densification (segmentierter Picker, immer sichtbar).

TECHNISCH

Schaltet zwischen den beiden Densification-Strategien: Classic (Original-3DGS-Verfahren mit Clone/Split/Prune und Gradienten-Threshold) und MCMC (Stochastic Gradient Langevin Dynamics mit Relocation, NeurIPS 2024). Beim Wechsel von Classic auf MCMC setzt die App die MCMC-spezifischen Felder automatisch auf erprobte Default-Werte (Reg-Weights = 0, MCMC-Cap- Multiplier 3.0, Sample-/Noise-Schedule). Ohne diese automatische Initialisierung litten Sessions mit alten Presets unter dem 1.4.4-MCMC-Collapse-Bug (460K→5 Gaussians, Watchdog-Kill). Die Picker- Auswahl bestimmt zusätzlich, welche Inspector-Elemente sichtbar sind — bei MCMC tauchen I16/I17 auf. Detaillierte Feld-Wirkung in Kapitel 6, T11–T16 (Classic) und T61–T73 (MCMC).

EINFACH GESAGT

Die zentrale Strategiewahl für das Wachsen der Gaussian-Anzahl. Classic ist gut getunet aus 459 Experimenten, erzeugt schnelle und hochwertige Ergebnisse und braucht keine MCMC-Felder zu kennen. MCMC ist der neuere Ansatz (NeurIPS 2024), reproduzierbarer und verzichtet auf manuelle Threshold-Justage — dafür rechnet er etwa 6× länger bei vergleichbarer Qualität. Beim Umschalten auf MCMC setzt die App automatisch sichere Defaults, damit das Training nicht in den 1.4.4-Collapse läuft. Details zu den Strategiefeldern stehen in Kapitel 6 (T11–T16 Classic, T61–T73 MCMC).

I16 MCMC Quality-Toggle

WO

Inspector → Trainingskonfiguration → MCMC Quality (nur bei Densification = MCMC).

TECHNISCH

Schaltet die Gradient-Accumulation auf 2 Schritte (aktiv) bzw. 1 Schritt (inaktiv). Akkumuliert die Gradienten aus zwei aufeinanderfolgenden Kamera-Views, bevor der Optimizer-Step ausgeführt wird. Empirisch (Session 33, V544a) reduziert das den finalen L1-Fehler um ca. 6% (0.0246 mit Quality vs 0.0261 ohne, bei 3-Trial-Average auf Horse-Full-MCMC). Der Preis: verdoppelte Trainingszeit. Bei sehr langen Trainings (200K Iterationen) führt das zu zusätzlichen 10+ Minuten Wartezeit — also nur dann lohnt, wenn die letzten paar Prozent Qualität wirklich gebraucht werden. Wirkt sich nur auf das Training aus, nicht auf das Export-Format oder die Viewport-Darstellung.

EINFACH GESAGT

Quality-Mode für MCMC mit Gradient-Accumulation über zwei Views. Macht das Endergebnis empirisch etwa 6% besser (L1 0.0246 statt 0.0261 im Horse-Test), kostet dafür doppelt so lange. Wenn du sowieso schon ein 200K-MCMC-Training fährst (gerne mal 2 Stunden), kommt nochmal eine knappe Stunde drauf. Lohnt sich bei finalen Showcase-Renderings oder zum Ende einer Quality-Sweep-Session, im Daily-Workflow eher nicht. Nur sichtbar, wenn Densification auf MCMC steht (I15).

I17 Auto-scale by scene-Toggle

WO

Inspector → Trainingskonfiguration → Auto-scale by scene (nur bei MCMC).

TECHNISCH

Wenn aktiv, skaliert die effektive Max-Gaussians-Obergrenze mit der SfM-Init-Point-Count × MCMC-Cap-Multiplier (Default 3.0). Beispiel: SfM liefert 250K Initpunkte, Basis-Cap = 150K, Multiplier 3.0 → effektive Obergrenze = max(150K, 750K) = 750K. Wenn deaktiviert, gilt strikt nur die Basis. War für v1.4.5 eingeführt, weil große Outdoor-Aufnahmen mit über 1000 Frames und entsprechend hoher SfM-Punkt-Dichte mit der starren 150K-Cap-Default die Densification ausgehungert haben — überflüssige Punkte blieben, neue durften nicht entstehen. Default OFF in Custom-Presets, ON in MCMC-Built-ins. Wirkt sich nur zur Trainingszeit aus, nicht im Export.

EINFACH GESAGT

Lässt die Maximalzahl der Gaussians mit der Szenengröße mitwachsen (genauer: mit der Anzahl der SfM-Initpunkte). Bei kleinen Szenen merkst du kaum einen Unterschied, bei großen Outdoor-Szenen ist es oft entscheidend für die Qualität — sonst „erstickt„ das Training, weil die Default-Obergrenze von 150K für die Szene viel zu niedrig ist. War extra für v1.4.5 eingeführt, nachdem sehr große Outdoor-Sets (über 1000 Frames) sichtbar an der Cap hingen.
Bei den MCMC-Built-in-Presets schon vorab eingeschaltet; in eigenen Presets defaultmäßig aus.

I18 Max Iterations-Stepper

WO

Inspector → Trainingskonfiguration → GroupBox → Max Iterations.

TECHNISCH

Stepper mit Range 1 000–100 000, Schrittweite 1 000. Bestimmt die Gesamtzahl der Optimizer-Iterationen. Linear korreliert mit der Trainingszeit (Halbierung = ca. 50% Zeit). Empirische Sweet-Spots: 20K (Classic Balanced, L1≈0.028), 40K (Classic Quality, L1≈0.023), 200K (MCMC Full, L1≈0.0246). Über 40K bei Classic bringt im Mittel kaum Verbesserung — Diminishing Returns. Beim Verändern wird, falls die Link-Funktion (I19) aktiv ist, Densify Until proportional mitgezogen (Default-Ratio: 0.5, d.h. Densify-Until = Max/2).

EINFACH GESAGT

Wie viele Trainingsschritte gefahren werden — mehr ist besser, kostet aber auch linear mehr Zeit. Faustregel: 20 000 für gute Qualität, 40 000 für das Optimum bei Classic-Strategie (darüber bringt's im Mittel kaum noch was). MCMC braucht deutlich mehr, 200 000 ist hier Standard. Verdoppeln der Iterationen verdoppelt grob die Trainingszeit. Bei aktivem Link-Button (I19) wird Densify Until proportional mitgezogen — praktisch immer das, was du willst.

I19 Link/Unlink-Button (Densify ↔ Iterations)

WO

Inspector → Trainingskonfiguration → GroupBox → kleiner Link-Button zwischen Max Iterations und Densify Until.

TECHNISCH

Toggle-Button, der das Verhältnis von Densify Until zu Max Iterations einfriert. Bei Aktiv (Link-Icon hervorgehoben) wird bei jeder Änderung von Max Iterations das Densify Until proportional nachgezogen. Beim Unlink (Link-Plus-Icon) bleiben die Werte unabhängig. Default ist linked, weil das die typische Korrelation widerspiegelt — wenn du das Training auf doppelte Iterationen ziehst, willst du meistens auch die Densification proportional länger laufen lassen. Das Verhältnis wird beim Setzen des Link-Buttons aus dem aktuellen Wert berechnet; ein typisches Verhältnis ist 0.5 (Densify-Until = halbe Iterations-Zahl).

EINFACH GESAGT

Kleine Klammer-Schaltfläche zwischen Max Iterations und Densify Until. Wenn aktiv (Link-Icon hervorgehoben), wandern die beiden Werte gemeinsam — verdoppelst du die Iterations, verdoppelt sich auch Densify Until im selben Verhältnis. Wenn nicht (`link.badge.plus` -Icon), kannst du sie unabhängig setzen. Standard ist verlinkt, weil das die typische Korrelation widerspiegelt — längeres Training will meist auch längere Densification-Phase. Für 99% der Fälle lass es eingerastet.

I20 Densify Until-Stepper

WO

Inspector → Trainingskonfiguration → GroupBox → Densify Until.

TECHNISCH

Stepper mit Range 500–50 000, Schrittweite 500. Bestimmt den Iterations-Index, ab dem keine neuen Gaussians mehr durch Clone/Split (Classic) oder Relocation (MCMC) hinzukommen. Nach Erreichen werden nur noch Position und Farbe verfeinert. Höhere Werte = mehr Gaussians = größere Datei, längere Pro-Iteration-Zeit (+30-60% GPU-Time pro Schritt). Typische Werte: 15K (für 30K Max-Iter), 20K (für 40K), 100K (für 200K MCMC). Bei aktivem Link (I19) automatisch mitskaliert. Wirkt sich anders bei Classic vs MCMC aus: Classic stoppt komplett das Wachstum, MCMC stoppt die Relocation-Logik, aber Sample-/Noise-Adaption läuft weiter.

EINFACH GESAGT

Bis zu welcher Iteration neue Gaussians hinzugefügt werden dürfen — bei Classic durch Clone/Split, bei MCMC durch Relocation. Danach geht es nur noch um die Farb- und Form- Verfeinerung der bestehenden Punkte. Höher = mehr Detail, aber auch größere Datei und +30-60% GPU-Zeit pro Schritt. Typische Werte: 15K (für 30K Max-Iter), 20K (für 40K), 100K (für 200K MCMC). Hängt normalerweise via Link (I19) an Max Iterations — selten sinnvoll, das manuell zu entkoppeln.

I21 SSIM Weight-Slider

WO

Inspector → Trainingskonfiguration → GroupBox → SSIM Weight.

TECHNISCH

Slider 0.0–1.0 in 0.05-Schritten, Anzeige als „0.20". Mischt L1-Loss (0.0) und SSIM-Loss (1.0). L1 strafft die Helligkeit pro Pixel, SSIM die strukturelle Ähnlichkeit (Kanten, lokale Statistiken). Default 0.2 ist der Wert aus dem Original-3DGS-Paper (Kerbl 2023) und reverse-engineered als robuster Kompromiss in zahlreichen Sessions. Höhere Werte (0.5+) bevorzugen Detailerhalt, können aber lokale Helligkeitsfehler ignorieren. Niedrigere Werte (< 0.1) führen zu Detail-Verlust an scharfen Kanten. Die SSIM- Berechnung läuft im Shader mit einem 11×11-Gaussian-Window. Performance: Bei 0.0 (nur L1) ist das Training ca. 8-12% schneller, weil die SSIM-Berechnung im Shader übersprungen wird.

EINFACH GESAGT

Wie stark die strukturelle Bildähnlichkeit (Kanten, lokale Muster) gegenüber dem reinen Helligkeitsvergleich gewichtet wird. 0.2 ist der Standard aus dem Original-3DGS-Paper und reicht für fast alle Szenen aus. Höher (0.5+) bei feinen Strukturen wie Haaren, Fell oder Vegetation — da hilft mehr Strukturgewicht. Niedriger (0.0) macht das Training etwa 8-12% schneller, weil die SSIM- Berechnung im Shader übersprungen wird, kostet aber Detail an scharfen Kanten. Wer keinen guten Grund für eine Änderung hat, lässt 0.2 stehen.

I22 Render Scale-Slider

WO

Inspector → Trainingskonfiguration → GroupBox → Render Scale.

TECHNISCH

Slider 0.25–1.0 in 0.25-Schritten, Anzeige als „100%„. Skaliert die Trainings-Rendering-Auflösung relativ zur Quellbild-Größe. Größter Hebel auf die Performance: 50% reduziert GPU-Zeit um ca. 75% (weil 4× weniger Pixel), 25% um ca. 94%. Der Gradient-Threshold wird automatisch mitskaliert. Unter dem Slider erscheint eine Live-Resolutions-Anzeige in MP (z.B. „2304×1296 (3.0 MP)"). Falls der aktuelle Wert vom empfohlenen abweicht, wird in oranger Schrift „— recommended: 50%„ eingeblendet. Die Empfehlung zielt auf ~3 MP effektive Auflösung — der von Apple Silicon GPUs am effizientesten verarbeitbare Bereich. 4K-Quellbilder bekommen z.B. automatisch 25% empfohlen, FullHD-Bilder 100%. Eine Änderung triggert zusätzlich die Buffer-Reallokation.

EINFACH GESAGT

Mit welcher Auflösung das Training rendert — einer der größten Performance-Hebel. Voll (100%) gibt die beste Qualität, kostet aber bei großen Bildern viel GPU-Zeit. Halb (50%) spart ungefähr 75% GPU-Zeit, weil viermal weniger Pixel berechnet werden — perfekt für 4K-Quellen. Unter dem Slider siehst du die effektive Auflösung in Megapixeln; die App zielt auf rund 3 MP, weil das auf Apple Silicon am effizientesten läuft. Wenn dein Wert davon abweicht, blendet die App einen orangenen „recommended„-Hinweis ein — meistens lohnt es, dem zu folgen.

Enhancements-Sektion (I26–I29, I42–I44)

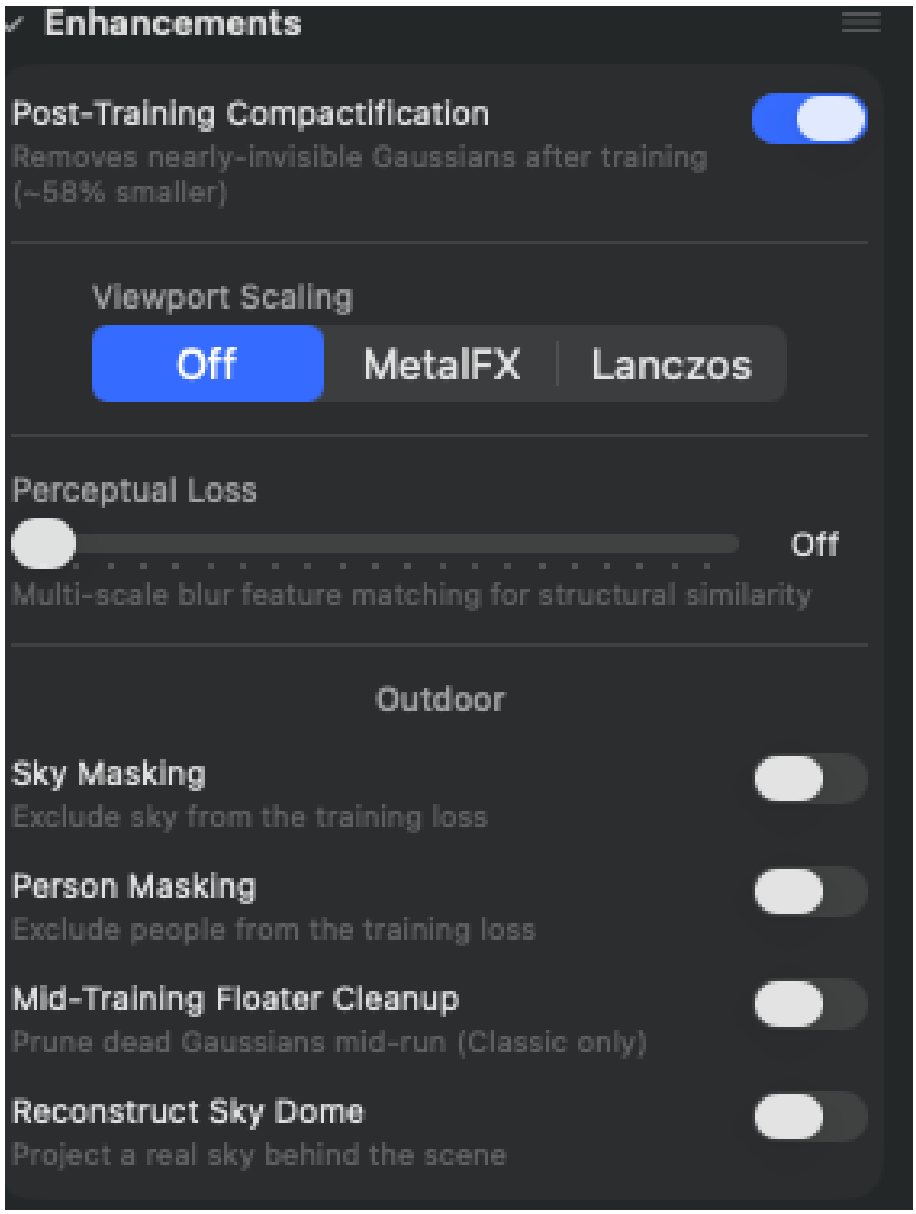

Abbildung 12: Crop nur Enhancements-Sektion — drei Reihen: Post-Training Compactification (Toggle an), Viewport Scaling (segmentierter Picker Off/MetalFX/Lanczos), Perceptual Loss (Slider auf „Off„). Jede Reihe mit Subtitle erklärt Funktion

Die Enhancements-Sektion gruppiert drei Features, die Bildqualität verbessern, ohne den Kern-Trainings-Loop selbst zu verändern. Die ersten beiden (I26-I27) sind **Post-Training**- bzw. **Viewport-Stufen**: Compactification räumt nach Trainingsende auf, das Viewport-Scaling ist ein reiner Viewport-Renderer, der das laufende Training nicht beeinflusst. Die Perceptual Loss (I29) ist trotz der Sektions-Zugehörigkeit ein Trainings-Bestandteil — sie wird während des Trainings als zusätzlicher Loss-Term aktiviert, deshalb die Trennung von den Viewport-Reglern über einen Divider. Seit v1.6 enthält die Sektion zusätzlich eine **Outdoor**-Gruppe (I42–I44: Sky Masking, Mid-Training Floater Cleanup, Reconstruct Sky Dome) — Trainings-Optionen gegen Himmels-Floater, die früher im Einstellungs-Fenster lagen und jetzt pro Projekt hier sitzen.

I26 Post-Training Compactification-Toggle

WO

Inspector → Enhancements → Post-Training Compactification.

TECHNISCH

Aktiviert das V443-Post-Processing: Nach Abschluss der Trainings-Iterationen werden Gaussians mit Opacity unter 0.01 (1% Sichtbarkeit) gelöscht. Empirisch reduziert das die Datei-Größe um ~55-58% bei null sichtbarem Qualitätsverlust — weil diese Gaussians visuell ohnehin nicht beitragen. Die Compactification läuft als GPU-Compact-Pass und dauert je nach Gaussian-Count Sekundenbruchteile bis wenige Sekunden. Beeinflusst die Trainings-Performance nicht. Wenn dieser Toggle aus ist, werden auch invisible Gaussians exportiert — relevant nur, wenn du das Format für ein weiteres Training-Stage gebrauchen willst (Continue Training), sonst Verschwendung von Speicher.

EINFACH GESAGT

Räumt nach dem Training Gaussians auf, die du sowieso nicht sehen kannst (Opacity unter 1%). Macht die Export-Dateien etwa zur Hälfte kleiner (55-58% Größenreduktion) ohne sichtbaren Qualitätsverlust. Läuft als kurzer GPU-Pass nach der letzten Iteration, dauert nur Sekundenbruchteile bis wenige Sekunden. Sollte praktisch immer an sein — der einzige Grund, das auszuschalten, ist wenn du das Training später per Continue Training fortsetzen willst und auch unsichtbare Gaussians behalten musst. Bei normalen Export-Workflows einfach anlassen.

I27 Viewport Scaling-Picker

WO

Inspector → Enhancements → Viewport Scaling (segmentierter Picker mit drei Optionen: Off, MetalFX, Lanczos).

TECHNISCH

Ein einziger segmentierter Picker, der den Viewport-Upscaler wählt — die drei Optionen sind **gegenseitig ausschließend**. Wenn die Trainings-Auflösung (durch I22 Render Scale) niedriger ist als die Viewport-Größe, skaliert der gewählte Modus das gerenderte Frame auf die Anzeigegröße hoch. **Off** = schlichtes bilineares Strecken. **MetalFX** = Apples ML-basierter Spatial Upscaler, die schärfste Option (das ML-Modell ist auf scharfe Kanten optimiert), Overhead ca. 1-2ms pro Frame auf M3-GPUs. **Lanczos** = Apples Metal Performance Shaders mit 8-Tap-Sinc-Resampling, klassisch ohne ML, Minimal-Overhead (< 0.5ms), Qualität unter MetalFX, aber ohne ML-typisches „Weichbügeln„ feiner Linien-Strukturen. Die Renderer- Pipeline wird beim Umschalten live umkonfiguriert — sichtbar sofort, ohne Neustart. **Hintergrund:** Früher waren das zwei separate Toggles (MetalFX + Lanczos), die gleichzeitig an sein konnten — ein widersprüchlicher Zustand, in dem MetalFX still über Lanczos hinwegging. Der Picker entfernt diesen Zustand; ein eventuell aus älteren Sessions geerbter „beide-an"-Zustand heilt sich beim nächsten Wechsel selbst zu MetalFX. Wirkt **nur** auf das Live-Viewport, nicht auf gerenderte Exporte (Orbit-Video, Screenshots) — die werden in voller Quell-Auflösung gerendert.

EINFACH GESAGT

Schärft das Live-Bild im Viewport hoch — besonders nützlich, wenn du mit reduzierter Trainings-Auflösung (Render Scale 50%, siehe I22) arbeitest. Drei Stufen, von denen immer nur eine aktiv ist: „Off„ streckt einfach die Pixel, „MetalFX" nutzt Apples Machine-Learning für die schärfsten Kanten (praktisch immer die beste Wahl), „Lanczos„ ist der klassische Filter ohne ML — nimm den als Fallback, falls MetalFX dir in einer Szene Linien glättet oder Artefakte zeigt. Greift live, ohne Neustart. Wirkt nur im Live-Viewport, nicht auf exportierte Orbit-Videos oder Screenshots — die werden in voller Quell-Auflösung gerendert. Anders als früher kannst du nicht mehr aus Versehen zwei Modi gleichzeitig wählen.

I29 Perceptual Loss-Slider

WO

Inspector → Enhancements → Perceptual Loss.

TECHNISCH

Slider 0.0–0.2 in 0.01-Schritten, Anzeige bei 0.0 als „Off„, sonst als „0.05“ usw. Aktiviert einen zusätzlichen Loss-Term, der multi-skalierten Gaussian-Blur des Renderings mit dem Ground-Truth-Bild vergleicht (3 Blur-Skalen). Fängt strukturelle Unterschiede ein, die L1+SSIM allein nicht erkennen. V460-Implementierung. Empirisch verbessert ein Wert von 0.05-0.1 den L1-Score in Sessions um ein paar Prozent, kostet aber ~5% Trainingszeit (zusätzlicher Forward-Pass durch die Blur-Kernel). Über 0.15 wird das Training instabil und L1 verschlechtert sich wieder (Loss-Term dominiert die Optimierung). Wirkt **während** des Trainings, nicht im Post-Processing — trotz Position in der „Enhancements„-Sektion ist das also keine reine Aufwertung im Nachgang.

EINFACH GESAGT

Ein zusätzlicher Loss-Anteil, der strukturelle Bildähnlichkeit über drei verschiedene Unschärfe-Stufen prüft. Hilft besonders bei Szenen mit feinen Strukturen wie Haaren, Stoff oder Vegetation, weil er Muster einfängt, die L1+SSIM allein nicht sehen. Kleinere Werte sind sicherer — 0.05 bis 0.1 ist der sweet spot, über 0.15 wird das Training instabil und der Loss verschlechtert sich wieder. Auf 0 (Off) ist die Funktion komplett aus und kostet nichts; aktiv schluckt sie etwa 5% Trainingszeit für den extra Forward-Pass durch die Blur-Kernel. Wirkt trotz „Enhancements„-Sektion direkt während des Trainings, nicht erst im Post-Processing.

I42 Sky Masking

WO

Inspector → Enhancements (Outdoor-Gruppe) → Toggle „Sky Masking„. Bound: `AppState.trainingConfig.skyMaskingEnabled` (pro Projekt, `@DefaultFalse`). Default: `false`.

TECHNISCH

Aktiviert pre-training Apple-Vision-basierte Sky-Pixel-Segmentation. Vor Trainings-Start wird für jede Eingabe-Kamera die Sky-Region per Apple-Vision-Foreground-Mask extrahiert (Sky = Background) und als Pro-Pixel-Maske der jeweiligen Kamera zugeordnet. Während des Trainings wird der Loss-Beitrag pro Pixel mit dem Komplement der Sky-Maske multipliziert — Sky-Pixel tragen 0 zum Gradient bei, sodass Gaussians, die in den Himmel projizieren, keine Optimierungs-Signale erhalten und damit nicht „dichter„ oder „heller“ werden. Reduziert Floater (dunkle Klümpchen im Himmel) bei Outdoor-/Drohnen-Szenen signifikant. Kostet ~3% L1-Regression bei klassischem 40K-Training (siehe `memory/dev_outdoor-floater-reduction.md`). Nur sinnvoll bei Outdoor-Szenen mit klar erkennbarem Himmel; bei Innenraum-Szenen oder weißem Hintergrund identifiziert die Sky-Segmentierung falsche Bereiche und blockt valide Loss-Signale. Der Wert wird pro Projekt gespeichert (nicht mehr app-global) und folgt dem Preset bzw. der Szenendatei.

EINFACH GESAGT

Bei Outdoor-Aufnahmen mit Himmel im Bild entstehen oft schwarze oder bunte Klümpchen im Himmel — sogenannte „Floater„. Diese Option erkennt automatisch, wo der Himmel ist, und sagt dem Training: „Lass den Himmel in Ruhe.“ Funktioniert sehr gut bei Drohnenflügen und Landschafts-Szenen. Bei Innenräumen oder dunklen Hintergründen kann es das Bild verschlechtern — also nur einschalten, wenn echter Himmel zu sehen ist. Details: `memory/dev_outdoor-floater-reduction.md`.

I43 Mid-Training Floater Cleanup

WO

Inspector → Enhancements (Outdoor-Gruppe) → Toggle „Mid-Training Floater Cleanup„. Bound: `AppState.trainingConfig.floaterCleanupEnabled` (pro Projekt, `@DefaultFalse`). Default: `false`.

TECHNISCH

Schaltet bei Classic-40K-Training (Preset „P4 Quality„) zwei zusätzliche Density-Control-Passes ein: bei Iteration 20,000 und bei Iteration 30,000. Beide Passes durchsuchen alle Gaussians nach drei Kriterien: (a) sehr niedrige Opacity (Standard 0.005), (b) winzige Screen-Space-Größe, (c) keine Loss-Beiträge in den letzten 1000 Iterationen. Gaussians, die alle drei Bedingungen erfüllen, werden gepurged. Effekt: ~5–15% weniger Gaussians am Ende des Trainings, sichtbar weniger dunkle Klümpchen im Himmel bei Drohnen-/Outdoor-Szenen. Kostet ~1–3% L1-Regression bei Close-Up-Indoor-Szenen, daher nicht als Default aktiviert. Der Wert wird über Neustarts gemerkt (im Gegensatz zu S7). Die zwei Cleanup-Iterationen (20K, 30K) sind hart definiert und können aktuell nicht per UI verändert werden; bei kürzeren Trainings (z.B. P2 Preview 5K) hat der Toggle keinen Effekt, weil er die Iterationsmarken nie erreicht. **Neu:** Der Toggle ist nur noch dann bedienbar, wenn das aktive Preset den **Classic**-Densifier verwendet (`densificationStrategy == .classic`). Bei MCMC oder Hybrid wird er **disabled** und ein Inline-Hinweis erscheint, weil diese Strategien tote Gaussians ohnehin selbst behandeln (MCMC via Relocation, Hybrid via kombinierter Reloc-/Noise-Logik) — die manuellen Cleanup- Passes wären dort wirkungslos bzw. kontraproduktiv. Code-Referenz: `RadianceKitApp.swift`, General-Tab. Details: `memory/dev_outdoor-floater-reduction.md`.

EINFACH GESAGT

Während des Trainings entstehen manchmal „tote„ Gauß-Punkte, die nichts mehr zur Bildqualität beitragen, aber Speicher belegen. Diese Option räumt zweimal während eines langen Trainings (bei 20K und 30K Iterationen) auf und entfernt diese Leichen. Bei Outdoor-Szenen mit Himmel ist das besonders sinnvoll, weil sich dort die meisten Floater sammeln. Bei kleinen Trainings oder nahaufnahmen von Möbeln eher nicht nötig. Der Schalter lässt sich nur einschalten, wenn dein Preset den Classic-Densifier nutzt — bei MCMC- oder Hybrid-Presets ist er ausgegraut (samt kurzer Erklärung), weil die ihre toten Punkte selbst aufräumen.

I44 Reconstruct Sky Dome

WO

Inspector → Enhancements (Outdoor-Gruppe) → Toggle „Reconstruct Sky Dome„. Bound: `AppState.trainingConfig.skyDomeEnabled` (pro Projekt, `@DefaultFalse`). Default: `false`.

TECHNISCH

Aktiviert die Pre-Training-Sky-Dome-Projektion (V549e MVP). Nach SfM und vor Trainings-Start wird für jede Eingabe-Kamera die in S7 gemeinsam genutzte Apple-Vision-Sky-Maske aus dem Bild extrahiert, die Sky-Pixel werden mit den Kamera-Intrinsics ent-projiziert auf eine virtuelle Kugel-Oberfläche (Standard-Radius 8× Szenen-Radius). Auf dieser Kugel werden ~5000 neue Gaussians initialisiert mit Farb-Mittelwerten aus den projizierten Sky-Pixeln, sehr großer Skalierung (1.0 in Szenen-Einheiten) und Anfangs-Opacity 0.95. Diese 5000 Gaussians sind kein Sky-Mask im klassischen Sinn — sie werden trainiert wie alle anderen, aber durch die hohe Anfangs-Opacity in einer dünnen Schale belassen. Ergebnis: bei 360°-Novel-Views in Outdoor-/Drohnen-Szenen erscheinen statt dunkler Confetti-Klümpchen tatsächliche Himmels-Farbe und Wolken-Strukturen. Der Wert wird über Neustarts gemerkt. Sinnvoll nur bei Outdoor-Szenen mit zumindest 360°-Kamera-Abdeckung; bei reinen Object-Captures ohne Himmel-Sicht hat es keinen Effekt. Status: experimentell, breitere A/B-Validierung über weitere Outdoor-Sets steht noch aus.

EINFACH GESAGT

Statt dass das Training versucht, den Himmel aus den paar sichtbaren Pixeln zu „erraten„ (was zu Floatern führt), projiziert RadianceKit die Himmel-Pixel direkt auf eine virtuelle Kugel um die Szene, bevor das Training startet. Wenn du dann die fertige Szene in 360° drehst, siehst du echten Himmel statt schwarzer Klümpchen. Funktioniert nur bei Outdoor-Aufnahmen, bei denen tatsächlich Himmel im Bild ist. Bei Wohnzimmer-Scans oder Studio-Aufnahmen bringt es nichts.

Metriken-Sektion (I30–I38)

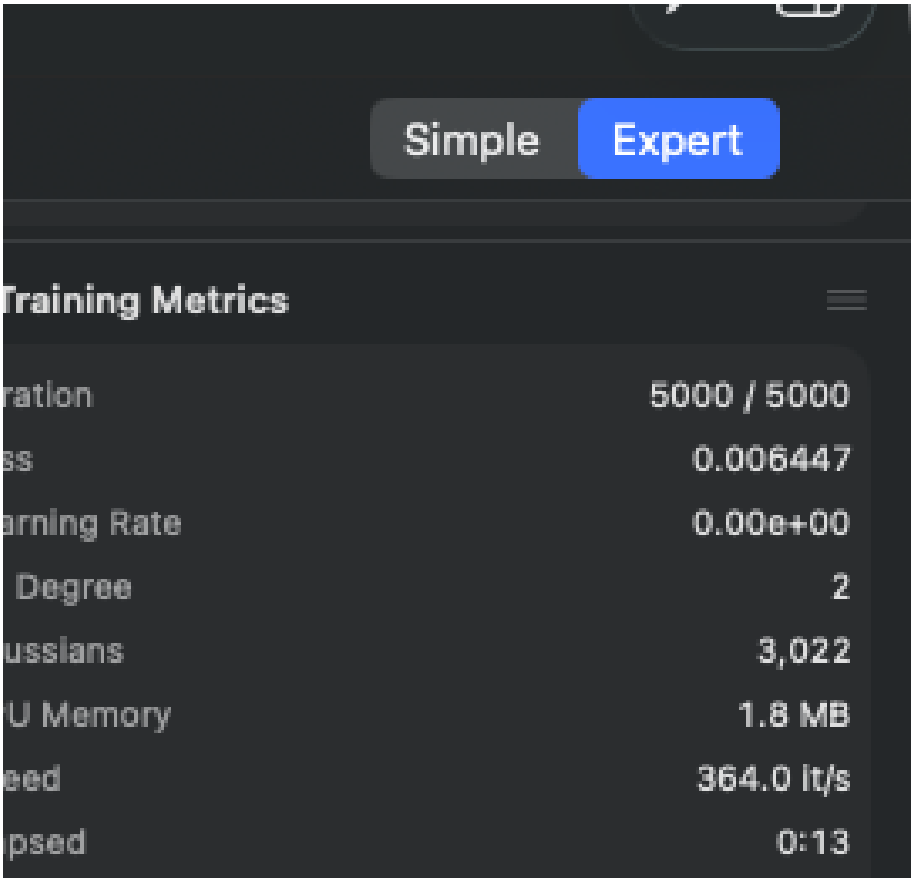

Abbildung 13: Crop nur Training Metrics-Sektion nach abgeschlossenem Training auf Bouquet (5K Iterationen, 2 991 Gaussians final) — Tabelle mit Trainings-Metriken (Iteration, Loss, SSIM Loss, Combined Loss, Gaussian Count, Learning Rate, Elapsed, ETA)

Während ein Training läuft, zeigt die Metriken-Sektion neun Live-Werte aus dem Trainings-Loop. Vor Start eines Trainings ist die Sektion leer („Start training to see live metrics„). Alle Werte werden alle ~30 Iterationen aktualisiert (Update-Frequenz des -Streams). Die Sektion ist read-only — kein Element ist anklickbar oder veränderbar. Für tiefere Analyse die JSONL-Trainings-Logs unter `~/Documents/RadianceKit/Logs/` heranziehen (Skript `python3 scripts/analyze_logs.py best 5`).

I30 Iteration

WO

Inspector → Metriken → Iteration. Read-only.

TECHNISCH

Anzeige im Format „4523 / 40000" — aktuelle Iteration über totale geplante Iterationen. Zählt synchron mit dem Trainings-Loop, der die Werte alle ~30 Iter pushed. Die zweite Zahl entspricht dem Max-Iterations-Wert zum Start-Zeitpunkt; sie ändert sich nicht mehr, auch wenn der Anwender den Stepper danach verstellt — der laufende Lauf nutzt seine eigene Snapshot-Kopie. Wenn die App über das Training-Menü Iterationen draufschiebt (Continue Training +5K/+10K/+20K), erhöht sich der Nenner.

EINFACH GESAGT

Wo das Training gerade steht. „4523 / 40000" heißt: 4523 von 40 000 Schritten sind durch, also etwa 11% fertig. Die linke Zahl zählt im Sekundentakt hoch; wenn sie minutenlang stehen bleibt, hängt das Training fest — meistens ein Hinweis auf GPU- Throttling oder eine konkurrierende App. Die rechte Zahl entspricht dem Max-Iterations-Wert (I18) beim Trainings-Start und ändert sich nicht mehr, auch wenn du den Stepper später verstellst. Bei Continue Training (+5K/+10K/+20K) wächst sie um die zusätzlichen Schritte mit.

I31 Loss

WO

Inspector → Metriken → Loss. Read-only.

TECHNISCH

Float-Wert mit sechs Nachkommastellen (z.B. „0.024385"). Misst den kombinierten L1+SSIM-Loss (Mix kontrolliert über I21 SSIM Weight) plus optional Perceptual Loss (I29) und andere Regularizer. Skala ist nicht absolut, sondern szenenabhängig — verlangt für die meisten Vergleiche denselben Datensatz. Typische Endwerte bei guten Konfigurationen: - Classic Quality 40K iters: 0.022–0.025 (Horse, Truck, Garden) - MCMC Full 200K iters: 0.024–0.028 - Outdoor Drohne 30K: 0.030–0.060 (geometrie-bedingt schlechter) - Indoor Apartments: 0.018–0.025

Werte über 0.10 nach 5K Iterationen deuten auf SfM-Probleme (schlechte Kamera-Posen) hin — abbrechen und SfM neu rechnen.

EINFACH GESAGT

Wie weit das gerenderte Bild noch vom Original abweicht — kombiniert aus L1, SSIM und ggf. Perceptual Loss. Kleiner ist besser. Unter 0.03 ist meistens richtig gut, unter 0.05 noch okay, Outdoor-Szenen liegen geometrie-bedingt eher bei 0.03-0.06. Über 0.10 nach mehreren Tausend Iterationen ist ein Warnsignal — meist liegt's an der Kamera-Rekonstruktion (SfM hat nicht sauber geklappt). Die Skala ist nicht absolut, sondern szenenabhängig; Vergleiche nur innerhalb desselben Datensatzes machen. Wenn die Zahl plötzlich nach oben springt, ist meistens ein Gradient-Explosion-Event passiert.

I32 Learning Rate

WO

Inspector → Metriken → Learning Rate. Read-only.

TECHNISCH

Scientific-Notation-Anzeige (z.B. „1.60e-04"). Aktuelle Lernrate für die Position-Parameter (3DGS hat sechs unabhängige LRs für Position, SH-DC, SH-Rest, Opacity, Scale, Rotation — angezeigt wird hier die Position-LR als repräsentative Größe). Default-Anfangswert 1.6e-4, der über einen Exponential-Decay bis ~1.6e-6 am Trainingsende absinkt. Der Verfall ist über das LR-Schedule-Feld in der Trainings-Konfiguration (T-Feld in Kap. 6) anpassbar. Wenn die LR ungewöhnlich hoch bleibt (z.B. 1e-3 oder mehr nach 10K Iterationen), könnte das auf eine fehlerhaft geladene Konfiguration hinweisen.

EINFACH GESAGT

Wie groß die Optimierungs-Schritte gerade sind — konkret die Lernrate für die Gaussian-Positionen. Startet bei 1.60e-04 und sinkt exponentiell auf etwa 1.60e-06 zum Trainingsende („1.60e-06" = 0.0000016). Der Verlauf läuft automatisch, du musst hier nichts justieren. Wenn der Wert nach 10 000+ Iterationen noch immer größer als 1e-3 ist, wurde wahrscheinlich eine fehlerhafte Config geladen — Training abbrechen und Preset neu wählen. Intern hat 3DGS sechs unabhängige Lernraten (Position, SH-DC, SH-Rest, Opacity, Scale, Rotation); hier siehst du nur die Position-LR als Stellvertreter.

I33 SH Degree

WO

Inspector → Metriken → SH Degree. Read-only.

TECHNISCH

Ganzzahl 0-3. Spherical-Harmonics-Grad für die Farb-Repräsentation. Beginnt bei 0 (nur die DC-Komponente, d.h. richtungs-unabhängige Farbe pro Gaussian — also nur eine RGB-Konstante) und steigt im Verlauf des Trainings progressiv auf 3 an. Der Standard-Schedule hebt den Grad bei 1000/2000/3000 Iterationen um je 1. SH-3 entspricht 48 Farb-Koeffizienten pro Gaussian (3 RGB-Channels × 16 SH-Basisfunktionen). Höherer SH-Grad = mehr richtungs-abhängige Reflexion (glänzende Oberflächen sehen unter verschiedenen Blickwinkeln korrekt unterschiedlich aus), aber auch mehr Speicher und langsameres Training.

EINFACH GESAGT

Wie komplex die Farbdarstellung pro Gaussian gerade ist. Startet bei 0 (nur eine richtungs-unabhängige Farbe pro Punkt) und wird stufenweise auf 3 hochgezogen — typisch bei Iteration 1000, 2000 und 3000. Stufe 3 bedeutet 48 Farb-Koeffizienten pro Gaussian und erlaubt richtungs-abhängige Reflexionen, also dass glänzende Oberflächen aus verschiedenen Blickwinkeln korrekt unterschiedlich aussehen. Brauchst du nicht aktiv anzufassen, der Schedule läuft automatisch. Höherer Grad kostet mehr Speicher und verlangsamt das Training leicht — aber das ist der Preis für realistische Glanzlichter.

I34 Gaussians

WO

Inspector → Metriken → Gaussians. Read-only.

TECHNISCH

Aktuelle Anzahl der Gaussians im Modell, formatiert mit Locale-Separator (z.B. „524.318"). Wachstum: - Classic: startet bei den SfM-Init-Punkten (typisch 50K-300K), wächst durch Clone/Split bis kurz vor Densify Until, dann statisch bis Trainingsende (modulo Pruning) - MCMC: Sample-Punkte werden hinzugefügt bis zum MCMC-Cap, dann nur noch Relocation

Healthy Endwerte: - Classic Quality: 400K-700K (Horse 524K, Garden 800K) - MCMC Full: exakt auf dem Cap (Default 150K, mit Auto-Scale Multiplier × SfM-Count je nach Szene 500K-1.5M)

Bei MCMC fällt die Zahl auf < 60% des Caps → Anomalie (Collapse-Indikator, deutet auf zu aggressive Regularizer hin).

EINFACH GESAGT

Wie viele Gaussian-Punkte das 3D-Modell gerade hat. Wächst während des Trainings, bis Densify Until (I20) erreicht ist; danach bleibt die Zahl praktisch konstant. Mehr Punkte = mehr Detail, aber auch größere Datei und langsameres Rendering im Viewport. 500.000 Gaussians ist ein typischer Mittelwert für Classic-Quality auf einer mittleren Szene; MCMC Full landet je nach Auto-Scale (I17) bei 500K bis 1.5M. Wenn die Zahl bei MCMC plötzlich unter 60% des Caps fällt, ist das ein Collapse-Indikator — meist zu aggressive Regularizer.

I35 GPU Memory

WO

Inspector → Metriken → GPU Memory. Read-only.

TECHNISCH

Schätzung des Gaussian-Buffer-Speicherverbrauchs als Gaussian-Count × 616 Bytes (formatiert im Memory-Style). 616 Bytes ist die empirische Größe eines voll ausgestatteten Gaussians (Position, Skalierung, Rotation, Opacity, SH-Koeffizienten Grad 3, Gradient- Akkumulator). Die Anzeige erfasst **nicht** den Renderer-Overhead (Tile-Buffer, Sort-Buffer, Backward-Buffer) — der reale GPU- Speicherbedarf liegt typisch 2-3× über diesem Wert. Bei 500K Gaussians: angezeigt ~290 MB, real ~700 MB. Bei 1.5M Gaussians: angezeigt ~880 MB, real ~2.5 GB. Auf M3 Max mit 64+ GB Unified Memory unkritisch, auf M3 Pro mit 18 GB schon ein Limit.

EINFACH GESAGT

Eine Schätzung, wie viel GPU-Speicher die Gaussians selbst belegen — rund 616 Bytes pro Punkt. Der tatsächliche GPU-Verbrauch ist 2-3× höher als angezeigt, weil der Renderer noch eigene Tile-, Sort- und Backward-Buffer dazulegt. Bei einem MacBook mit 16-18 GB Unified Memory solltest du unter 500K Gaussians bleiben; mit M3 Max oder Studio (64+ GB) kannst du locker 1.5M und mehr fahren. Wenn das Training plötzlich crasht oder das System swappt, ist meistens hier die Grenze erreicht — Render Scale (I22) runter oder Densify Until (I20) reduzieren.

I36 Speed

WO

Inspector → Metriken → Speed. Read-only.

TECHNISCH

Iterations-pro-Sekunde mit einer Nachkommastelle („24.3 it/s„). Berechnet vom Trainer als gleitender Durchschnitt über die letzten ~100 Iterationen. Typische Werte: - Quick Preset (1K iters): 80-120 it/s (kurz, kein steady-state) - Classic 20K @ 1.0 Render Scale (Truck-Szene, M3 Max): 25-35 it/s - Classic 20K @ 0.5 Render Scale: 80-120 it/s - MCMC 200K @ 0.5 Render Scale: 25-50 it/s (langsamer wegen Relocation) - Bei 1M+ Gaussians und voller Auflösung: < 10 it/s

Sinkende Speed im Verlauf des Trainings ist normal — mehr Gaussians = mehr Compute pro Iteration. Plötzliche Einbrüche (z.B. von 30 → 5 it/s) deuten auf GPU-Thermal-Throttling oder konkurrierende Apps hin.

EINFACH GESAGT

Wie schnell das Training läuft, in Iterationen pro Sekunde. Steht typischerweise bei 20-50 it/s, bei reduzierter Render Scale (50%) und kleinen Szenen auch mal 80-120 it/s. Sinkt im Verlauf des Trainings ganz natürlich, weil mehr Gaussians = mehr Arbeit pro Iteration. Plötzliche Einbrüche (z.B. 30 → 5 it/s) deuten auf GPU- Thermal-Throttling oder konkurrierende Apps hin — Browser-Tabs mit Video, Time-Machine-Backup, Photos-Indexing. App im Vordergrund halten und Hintergrund-Programme schließen hilft oft. Bei 1M+ Gaussians und voller Auflösung sind unter 10 it/s normal.

I37 Elapsed

WO

Inspector → Metriken → Elapsed. Read-only.

TECHNISCH

Bereits vergangene Zeit als „4:23″ (m:ss) oder „1:23:45″ (h:mm:ss). Format-Switch ab 1 Stunde. Misst nur die reine Trainings-Zeit, nicht die vorgelagerten Phasen (SfM-Berechnung, Bild- Import). Bei Pause/Resume läuft die Uhr weiter — ist also wall-clock, nicht CPU-Zeit.

EINFACH GESAGT

Wie lange das Training schon läuft, als reine Stoppuhr (Wall-Clock-Zeit). Format ist „m:ss„ bis eine Stunde, danach „h:mm:ss“. Nicht „CPU-Zeit„, sondern „wie lange warten wir schon“ — also auch Pause-Zeiten zählen mit. Misst nur die reine Trainings-Phase, nicht die vorgelagerte SfM-Berechnung oder den Bild-Import. Hilfreich zum Vergleich mit der ETA (I38) — wenn Elapsed deutlich über die ursprüngliche ETA hinaus schießt, ist das Training irgendwo langsamer geworden als geplant.

I38 ETA

WO

Inspector → Metriken → ETA. Read-only.

TECHNISCH

Geschätzte Restzeit als „17:42″ oder „1:12:35″. Berechnung: (Max Iterations – aktuelle Iteration) / Iterationen-pro- Sekunde. Zeigt „–„ wenn die Speed gerade null ist (ganz am Anfang oder bei Pause). Die Schätzung wird **nicht** an die typische Verlangsamung gegen Trainingsende angepasst — gerade bei MCMC und Classic mit großen Densify-Until-Werten neigt das Training dazu, langsamer zu werden, weil immer mehr Gaussians ins Bild kommen. Real bleibt es typischerweise 10-20% über der Anfangs-ETA.

EINFACH GESAGT

Wie lange voraussichtlich noch zu warten ist — berechnet aus den verbleibenden Iterationen und der aktuellen Speed (I36). Eine grobe Schätzung: Wenn der Mac plötzlich langsamer wird (mehr Gaussians ab Densify-Phase, Thermal-Throttling, andere Apps), kann's länger dauern als angezeigt. Die App rechnet die typische Verlangsamung gegen Trainingsende nicht ein, deshalb landet das echte Ende meist 10-20% über der Anfangs-ETA. Plus 15% einkalkulieren, dann passt es meistens. Zeigt „–„ wenn die Speed gerade 0 ist (Trainings-Anfang oder Pause).

Verlust-Diagramm-Sektion (I39–I41)

Abbildung 14: Crop nur Loss History-Sektion nach abgeschlossenem Training — Current 0.0064, Min 0.0035 (grün), blauer Verlauf von 0.027 (Iteration 1) zu 0.0035 (Iteration 5K) mit charakteristischem Knick um Iter 200, darunter Gaussian Count-Chart orange

Die Verlust-Diagramm-Sektion visualisiert den Trainings-Verlauf über die Zeit. Sie besteht aus zwei Charts: einem Loss-Curve-Chart (groß, oben, blau) und einem Gaussian-Count-Chart (kleiner, unten, orange). Beide werden live während des Trainings aufgebaut und persistieren bis zum nächsten Trainings-Start. Vor dem ersten Training ist der Bereich leer („Loss curve will appear during training„). Die Charts sind reine SwiftUI-Path-Drawings (kein Swift-Charts-Framework), damit sie auch bei 100K+ Punkten flüssig rendern.

I39 Current Loss (Anzeige)

WO

Inspector → Verlust-Diagramm → linker Label-Bereich „Current: 0.0287". Read-only.

TECHNISCH

Float-Wert des letzten Loss-Sample-Punkts, formatiert mit vier Nachkommastellen. Identisch mit I31 (Loss in der Metrics-Sektion), nur hier kompakter formatiert. Quelle ist die Loss- History — eine Liste, die pro ~30 Iterationen einen Eintrag bekommt. Nur endliche Werte werden in die Liste aufgenommen — NaN/Infinity (sehr selten, im Falle eines Gradient-Explosion-Bugs) werden gefiltert.

EINFACH GESAGT

Der aktuelle Loss-Wert in kürzerer Schreibweise als in der Metrics-Sektion (vier Nachkommastellen). Inhaltlich identisch mit I31, aber hier sitzt die Anzeige direkt am Loss-Chart und gibt dir beim Beobachten der Kurve den exakten Zahlenwert. Wird wie alle Live-Metriken alle 30 Iterationen aktualisiert. NaN- oder Infinity- Werte (extrem selten bei Gradient-Explosion-Bugs) filtert die App automatisch heraus. Nützlich, um beim Schauen aufs Diagramm nicht in die andere Sektion springen zu müssen.

I40 Min Loss (Anzeige)

WO

Inspector → Verlust-Diagramm → rechter Label-Bereich „Min: 0.0245" (grün). Read-only.

TECHNISCH

Minimum aller jemals gesehenen Loss-Werte des aktuellen Trainings-Laufs. Wird live aus der Loss-History rekomputiert — keine separate Persistenz. Wird mit grüner Schrift dargestellt, weil „Min„ = „Best so far". Die gestrichelte grüne Linie am unteren Chart-Rand markiert diese Y-Position visuell. Bei Continue-Training- Sessions startet die Mindest-Verfolgung neu — die alte History wird in der UI durch die neue ersetzt (nicht angehängt). Wenn das aktuelle Training schlechter als das vorhergehende läuft, kann die Min-Anzeige also größer sein als das vorherige Endergebnis.

EINFACH GESAGT

Der niedrigste Loss-Wert, den dieses Training bisher gesehen hat — grün dargestellt, weil „best so far„. Die gestrichelte grüne Linie am unteren Chart-Rand markiert diese Position auch visuell. Wenn die aktuelle Kurve gerade deutlich darüber liegt, gibt's mit Glück noch eine Verbesserung; meistens ist Min aber das Endergebnis-Indiz, das dich später interessiert. Bei Continue-Training- Sessions startet die Min-Verfolgung neu, weil die alte History in der UI durch die neue ersetzt wird — der Min-Wert kann dadurch schlechter aussehen als das vorherige Endergebnis.

I41 Gaussian Count Chart

WO

Inspector → Verlust-Diagramm → zweites Chart darunter (orange). Read-only.

TECHNISCH

Linien-Diagramm der Gaussian-Anzahl über die Trainings-Iterationen. Quelle: die Gaussian-Count-History (Liste von (Iter, Count)-Paaren, gefüllt vom Trainer alle ~30 Iter). Y-Skala dynamisch zwischen Minimum und Maximum der History. Bei Classic-Strategie sieht die Kurve typischerweise so aus: stetig steigend bis Densify Until, dann flach (mit kleinen Pruning-Schwankungen). Bei MCMC: steiler Anstieg bis Cap, dann horizontale Linie (Relocation hält die Zahl konstant). Wenn die Kurve **fällt** trotz aktivem Training, prunt die Densification zu aggressiv — Indiz für falsche Defaults oder ein bekannter MCMC-Collapse-Bug (v1.4.4-Hotfix-Thema).

EINFACH GESAGT

Wie sich die Anzahl der Gaussians über die Trainings-Zeit entwickelt — das kleinere orange Chart unter der Loss-Kurve. Bei Classic-Strategie steigt die Linie stetig an, bis Densify Until (I20) erreicht ist, danach bleibt sie flach mit kleinen Pruning-Schwankungen. Bei MCMC schnellt sie steil auf den Cap hoch und bleibt dann horizontal, weil Relocation die Zahl konstant hält. Wenn die Kurve trotz aktivem Training plötzlich nach unten knickt, ist die Densification zu aggressiv beim Prunen — klassisches Anzeichen für den MCMC-Collapse-Bug aus v1.4.4. Dann hilft App-Update oder ein Wechsel zurück auf Classic.

Wie liest man die Loss-Kurve?

Das Loss-Chart ist das wichtigste Diagnose-Tool im Inspector — kein anderer Indikator zeigt so direkt, ob das Training nützlich vorangeht oder festhängt. Die typische gesunde Form ist ein schneller Abfall in den ersten 1000-3000 Iterationen (von ~0.15 auf ~0.05), gefolgt von einem langsamen, gleichmäßigen Abfall bis zum Trainingsende (auf 0.020-0.030). Logarithmisch wirkt die Kurve dabei wie eine glatte Diagonale.

Was bedeutet ein Plateau bei der Loss? Wenn die Kurve über mehrere Tausend Iterationen flach bleibt, gibt's zwei mögliche Lesarten: (a) Training ist „konvergiert„ — der Loss kann nicht mehr signifikant sinken, weil das Modell so gut ist, wie es mit den gegebenen Daten und Settings sein kann. Das ist erwünscht; das ist „fertig". (b) Training „hängt„ — der Loss kann eigentlich noch sinken, aber Optimierung stagniert (lokales Minimum, Lernrate zu klein, Densification aus). Unterscheiden: Wenn der Loss-Wert in einem typisch guten Bereich liegt (0.020-0.030 bei Indoor/Object, 0.040-0.060 bei Outdoor) und die Kurve seit 5K Iterationen flach ist, ist's konvergiert. Wenn der Wert deutlich höher ist als bei vergleichbaren Szenen (z.B. 0.08), hängt es.

Achtung Gaussian-Plateau ≠ Loss-Plateau. Ein Plateau in der Gaussian-Anzahl bedeutet **nicht** „Training ist fertig„. Es bedeutet nur, dass die Densification aufgehört hat, neue Punkte hinzuzufügen — entweder weil erreicht ist (Classic) oder weil das MCMC-Cap voll ist. Das Training läuft danach weiter und verfeinert nur noch die bestehenden Punkte. Das eigentliche „fertig"-Signal liest du an der Loss-Kurve und an der Iteration-Anzeige (I30), nicht hier.

Faustregel zum Abbrechen: Wenn die Loss-Kurve nach 5000+ Iterationen über 0.08 liegt und kaum noch sinkt, ist mit hoher Wahrscheinlichkeit die SfM-Rekonstruktion schief. Training abbrechen, in Kapitel 9 nachschauen, ob das gewählte SfM-Backend zur Szene passt, ggf. auf COLMAP/Native umstellen, dann neu starten. Lieber 10 Minuten in besseres SfM investieren als 2 Stunden Training mit schlechter Kamera-Ausrichtung.

Wann nach Inspector greifen?

Schnell-Referenz: Welche Sektion + welche Controls für welchen typischen Use-Case?

Common-Task	Sektion	Control-IDs
Farben des fertigen Splats entsättigen	Look	L1 (Saturation)
Nadel-/Konfetti-Splats runden	Look	L2 (Splat length)
Löchrige Wolke füllen / Splats vergrößern	Look	L3 (Splat size)
Ferne „Far-Konfetti„ bei Orbits ausblenden	Look	L4 (Fade far region)
Look-Anpassungen verwerfen	Look	L5 (Reset finishing)
Vorgefertigtes Setup laden	Presets	I7 (Zeile anklicken)
Eigenes Setup speichern	Presets	I1 → I2 → I4
Setup mit Kollegen teilen	Presets	I5 (Export) bzw. I6 (Import)
SfM-Backend wechseln (z.B. weil Apple-PG zu instabil)	Trainingskonfiguration	I12 (siehe Kap. 9)
Video-Frames ohne EXIF-Brennweite verarbeiten	Trainingskonfiguration	I13 (FOV Override)
COLMAP-Performance: GLOMAP statt Klassisch	Trainingskonfiguration	I14
Von Classic auf MCMC umschalten	Trainingskonfiguration	I15
Training länger laufen lassen	Trainingskonfiguration	I18 (Max Iter) + I20 (Densify Until) — gekoppelt via I19
GPU-Zeit halbieren	Trainingskonfiguration	I22 (Render Scale auf 50%)
Trainings-Qualität +6% (MCMC)	Trainingskonfiguration	I16 (MCMC Quality)
Outdoor-Szene mit vielen SfM-Punkten	Trainingskonfiguration	I17 (Auto-scale by scene)
COLMAP-Pfad einrichten / wechseln	Trainingskonfiguration	I23 / I24 / I25
Export-Dateien kleiner machen	Enhancements	I26 (immer an lassen)
Viewport schärfer ohne mehr Trainingszeit	Enhancements	I27 (Viewport Scaling → MetalFX)
MetalFX glättet zu sehr → Alternative	Enhancements	I27 (Viewport Scaling → Lanczos)
Letztes Quäntchen Detail bei feinen Strukturen	Enhancements	I29 (Perceptual Loss 0.05-0.1)
Training überwachen	Metriken	I30 (Fortschritt), I36 (Tempo), I38 (Restzeit)
[illegible]	[illegible]	[illegible]

KAPITEL

Kapitel 3 — Einstellungen

Das Einstellungs-Fenster öffnet sich über `RadianceKit → Einstellungen…` oder die Standard-Tastenkombination `⌘,`. Es enthält zwei Tabs: **General** und **AI Helpers**. Anders als die Inspector-Werte aus Kapitel 2 wirken die Einstellungen aus diesem Fenster **app-global** (über alle Projekte hinweg) — sie werden in persistiert und überleben App-Neustarts. Der General-Tab gruppiert drei inhaltliche Abschnitte: Interface, Viewport und Training. (Die früher hier angesiedelten Outdoor-Floater-Toggles — Sky Masking, Mid-Training Floater Cleanup und Reconstruct Sky Dome — sind seit v1.6 in die **Enhancements**-Sektion des Expert-Inspectors gewandert, wo sie pro Projekt gespeichert werden; siehe Kapitel 2, I42–I44.) Der AI-Helpers-Tab schaltet die On-Device-Machine-Learning-Helfer (Vision, CoreML) für SfM- und Trainings-Vorverarbeitung an.

Frühere Bedienelemente zum gesammelten Aktivieren oder Deaktivieren aller AI-Helpers existieren in der aktuellen Version nicht mehr — entsprechend werden sie hier nicht dokumentiert. Auch der frühere „Coming Soon„-Bereich für noch nicht ausgelieferte Helfer wurde entfernt und ist hier nicht referenziert.

General-Tab

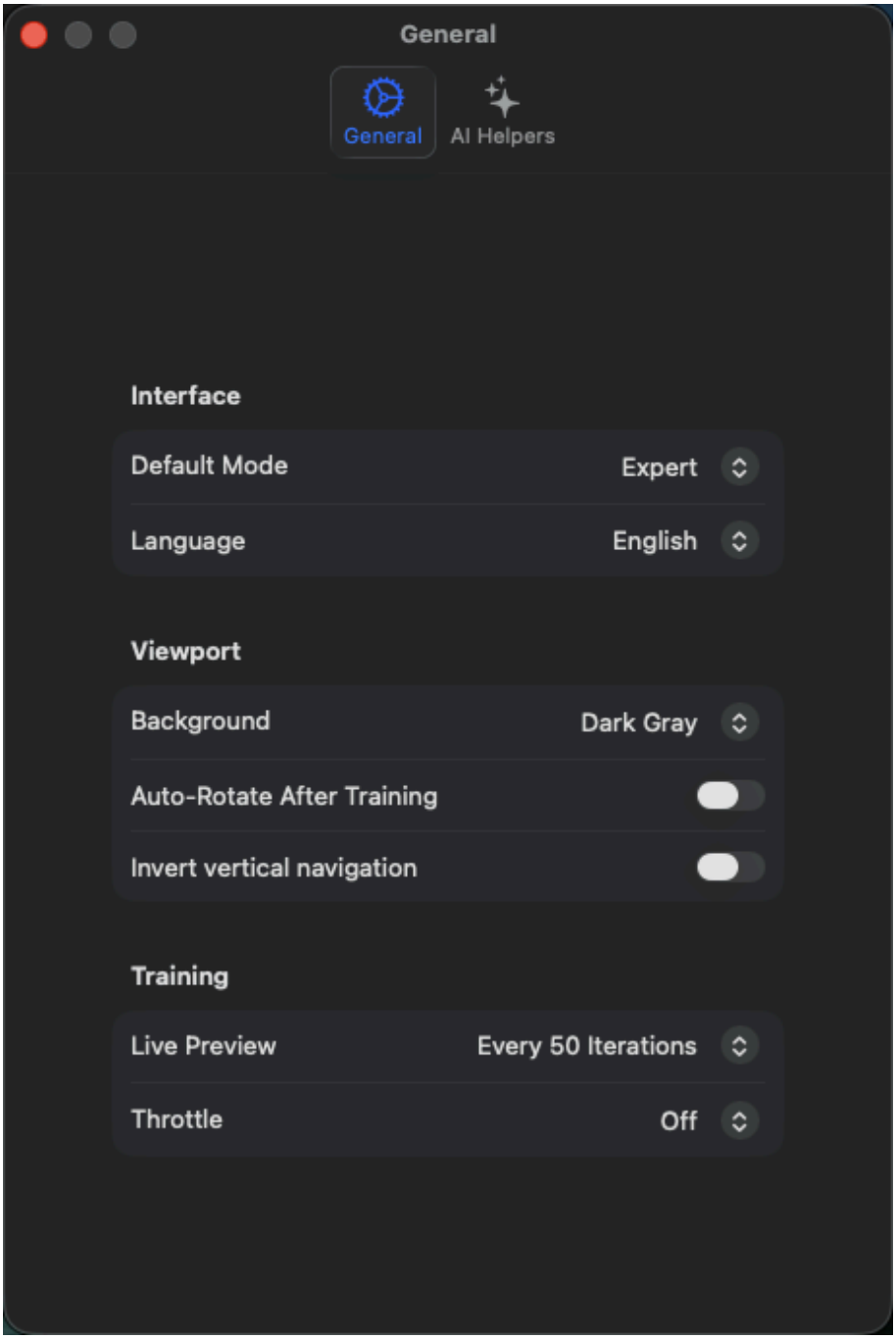

Abbildung 15: Einstellungen → General-Tab mit Benutzeroberfläche, Ansichtsfenster, Training und Experimental-Sektion

S1 Default Mode

WO

Settings → General → Interface → Default Mode-Picker. Bound:. Default: `.simple`.

TECHNISCH

Steuert, in welchem von zwei UI-Modi die App nach dem nächsten Start öffnet. „Simple Mode„ ist der geführte Wizard-Workflow in 4 Schritten (Import → Processing → Preview → Export, dokumentiert in Kapitel 10 unter Z1–Z4), „Expert Mode“ das klassische Drei-Panel-Layout mit Navigator, 3D-Viewport und Expert-Inspector aus Kapitel 2. Der Wert wird über Neustarts gemerkt. Identische Wirkung wie das Menü `Mode → Simple Mode` (⌘1) / `Mode → Expert Mode` (⌘2), nur dass das Menü die laufende Sitzung umschaltet, während dieser Picker den Default für künftige Sitzungen setzt. Beide Modi greifen auf denselben Projekt-State zu — Projekte, Kameras und Trainingskonfiguration bleiben beim Modus-Wechsel erhalten. Mode-spezifische Toolbar-Buttons werden sofort neu gerendert.

EINFACH GESAGT

Hier wählst du, mit welcher Oberfläche RadianceKit beim nächsten Start hochfährt. „Simple Mode„ ist der Anfänger-Modus: vier klare Schritte, vorgegebene Voreinstellungen, kaum Optionen. „Expert Mode“ ist das volle Werkzeugkasten-Layout mit allen Reglern, die du in Kapitel 2 siehst. Du kannst jederzeit über das Menü „Mode„ hin- und herwechseln, ohne dass Bilder oder Trainings-Fortschritt verloren gehen.

S2 Language

WO

Settings → General → Interface → Language-Picker. Bound:. Default: `.system` (folgt der macOS-Sprache).

TECHNISCH

Wählt die Anzeigesprache der gesamten App-UI, unabhängig von der macOS-System-Sprache. RadianceKit ist in 17 Sprachen lokalisiert (`de` , `en` , `pl` , `en-AU` , `ar-SA` , plus 12 weitere). Bei „System„ folgt die App der macOS-Sprache. Bei einer expliziten Wahl wird die Spracheinstellung über Neustarts gemerkt; vollständige Wirkung braucht in der Regel einen App-Neustart, weil Lokalisierungs-Bundles nur beim Start geladen werden. Die 298 dokumentierten Lokalisierungs-Schlüssel im Projekt werden alle berücksichtigt, einschließlich aller Texte in Sub-Views und Hilfe-Tooltips.

EINFACH GESAGT

Falls dein Mac auf Englisch läuft, du aber lieber die deutsche RadianceKit-Oberfläche möchtest (oder umgekehrt), stellst du das hier ein. Die meisten Texte tauschen sofort. Manche Dialoge erscheinen erst nach einem App-Neustart in der neuen Sprache.

S3 Viewport Background

WO

Settings → General → Viewport → Background-Picker. Bound:. Default: `.darkGray` (RGB 0.1, 0.1, 0.1).

TECHNISCH

Setzt die Standard-Hintergrundfarbe für den 3D-Viewport. Drei Optionen: „Dark Gray„ (RGB 0.1, 0.1, 0.1 — Default), „Black“ (0, 0, 0) und „White„ (1, 1, 1). Das Setting persistiert den Default für neue Projekte und Sitzungen über Neustarts und aktualisiert gleichzeitig den laufenden Metal-Renderer sofort. Identische Optionen finden sich im Menü `Viewport → Background` (M21, M22, M23), aber der Settings-Picker setzt den Default, während das Menü die laufende Anzeige umschaltet. Wichtig für Screenshots und Demo-Videos: weiße Hintergründe heben grüne/blaue Floater stärker hervor, dunkle Hintergründe sind besser für saubere Render-Aufnahmen.

EINFACH GESAGT

Die Farbe hinter deinen 3D-Modellen im Vorschaufenster. Dunkelgrau ist Standard und passt für die meisten Szenen. Weiß ist gut für Screenshots, schwarz wirkt edler bei Render-Aufnahmen. Du kannst die Farbe jederzeit über das Menü „Viewport → Background„ für die laufende Szene umschalten — diese Einstellung legt nur fest, welche Farbe beim nächsten Öffnen wieder aktiv sein soll.

S4 Auto-Rotate After Training

WO

Settings → General → Viewport → Toggle „Auto-Rotate After Training„. Bound:. Default: `false`.

TECHNISCH

Startet unmittelbar nach Trainings-Ende eine kontinuierliche Turntable-Rotation der Viewport-Kamera um den Szenen-Schwerpunkt (Standard-Drehrate ~0.3 rad/s). Praktisch nützlich für Demo-Sessions, A/B-Vergleiche und um aus 360°-Sicht direkt zu beurteilen, ob „Floater„ am Szenen-Rand entstanden sind. Effekt ist visuell identisch zum Menü `Viewport → Toggle Auto-Rotation` (M16, ⌘⌥T), nur dass der Toggle hier das Verhalten automatisch nach Trainings-Ende auslöst statt manuell. Lässt sich später jederzeit über das Menü oder durch Klick in den Viewport (was die Rotation pausiert) unterbrechen. Hat keinen Einfluss auf Trainings-Performance — die Rotation läuft erst, wenn das Training fertig ist.

EINFACH GESAGT

Wenn aktiviert, dreht sich die 3D-Szene automatisch, sobald das Training fertig ist — wie ein Karussell. Schön, wenn du beim nächtlichen Training morgens das Ergebnis schon in Bewegung siehst, ohne selbst klicken zu müssen. Bei langen Sitzungen, in denen du das Training nur überwachst, lass es lieber aus.

S5 Live Preview Interval

WO

Settings → General → Training → Live Preview-Picker. Bound: `AppState.trainingConfig.livePreviewInterval`. Default: `0` (Off).

TECHNISCH

Bestimmt, in welchem Iterations-Abstand der laufende Trainings-Snapshot in den 3D-Viewport gerendert wird. Vier diskrete Werte: 0 („Off„), 50, 250, 1000 Iterationen. Bei aktivem Live Preview kopiert der Trainer den Gaussian-Buffer aus der GPU in einen separaten Render-Buffer und triggert einen Viewport-Redraw. Bei „Off“ wird der Viewport erst nach Trainings-Abschluss aktualisiert. Performance-Kosten: alle 50 Iterationen ~5–10% langsamer auf M3 Ultra, alle 250 Iterationen ~1–2% langsamer, alle 1000 Iterationen unmessbar. Memory-Overhead konstant ~2 GB für den Snapshot-Buffer, unabhängig vom Intervall. Der Wert dient als Default für neue Trainings; nach Trainings-Start zeigt der Trainings-Inspector den realen Live-Wert dieses Trainings an. Bei Intervall 50 ist der Visual-Eindruck ein flüssiges „Aufwachsen„ der Punktwolke, bei 1000 wirkt es stockend.

EINFACH GESAGT

Während das Training läuft, kannst du wählen, wie oft die 3D-Ansicht aktualisiert wird. „Off„ heißt: keine Aktualisierung während des Trainings (am schnellsten). „Every 50 Iterations“ zeigt fast in Echtzeit, wie deine Szene entsteht (etwas langsamer). Für gemütliches Zuschauen bei kleinen Trainings ist „Every 250“ ein guter Kompromiss.

S6 Throttle Delay

WO

Settings → General → Training → Throttle-Picker. Bound: `AppState.trainingConfig.throttleDelayMs`. Default: `0` (Off).

TECHNISCH

Fügt zwischen Trainings-Iterationen eine artifizielle Verzögerung in Millisekunden ein. Vier diskrete Werte: 0 („Off„), 2 („Light“), 5 („Moderate„), 10 („Eco“). Sinn: bei längeren Trainings (mehrere Stunden) wird die GPU sonst zu 100% ausgelastet, was zu spürbar langsamer System-UI führt (Mauszeiger ruckelt, andere Apps werden träge). Die Throttle-Verzögerung gibt der GPU Pausen, in denen andere Tasks ausgeführt werden können. Performance-Kosten sind erheblich: bei 5 ms Throttle dauert ein typisches 40K-Training etwa 50–80% länger als ohne Throttle. Im Performance-Modus „Eco„ (10 ms) ist die Verzögerung pro Iteration länger als die Iteration selbst — Faktor 2–3× langsamer. Bei aktivem Throttle erscheint unter dem Picker ein Hinweis: „Throttle is on. Training will be slower than usual.“ Die App selbst reagiert nicht spürbar besser — nur andere Apps profitieren.

EINFACH GESAGT

Wenn dein Mac während eines langen Trainings zu heiß läuft oder andere Programme zu zäh werden, schalte hier eine Bremse ein. „Off„ gibt der GPU Vollgas (am schnellsten). „Light“ macht eine kleine Pause zwischen jedem Schritt (etwas langsamer, aber System reagiert besser). „Eco„ ist die stärkste Bremse — gut für Nacht-Trainings auf einem MacBook, die nicht zu heiß werden sollen.

AI-Helpers-Tab

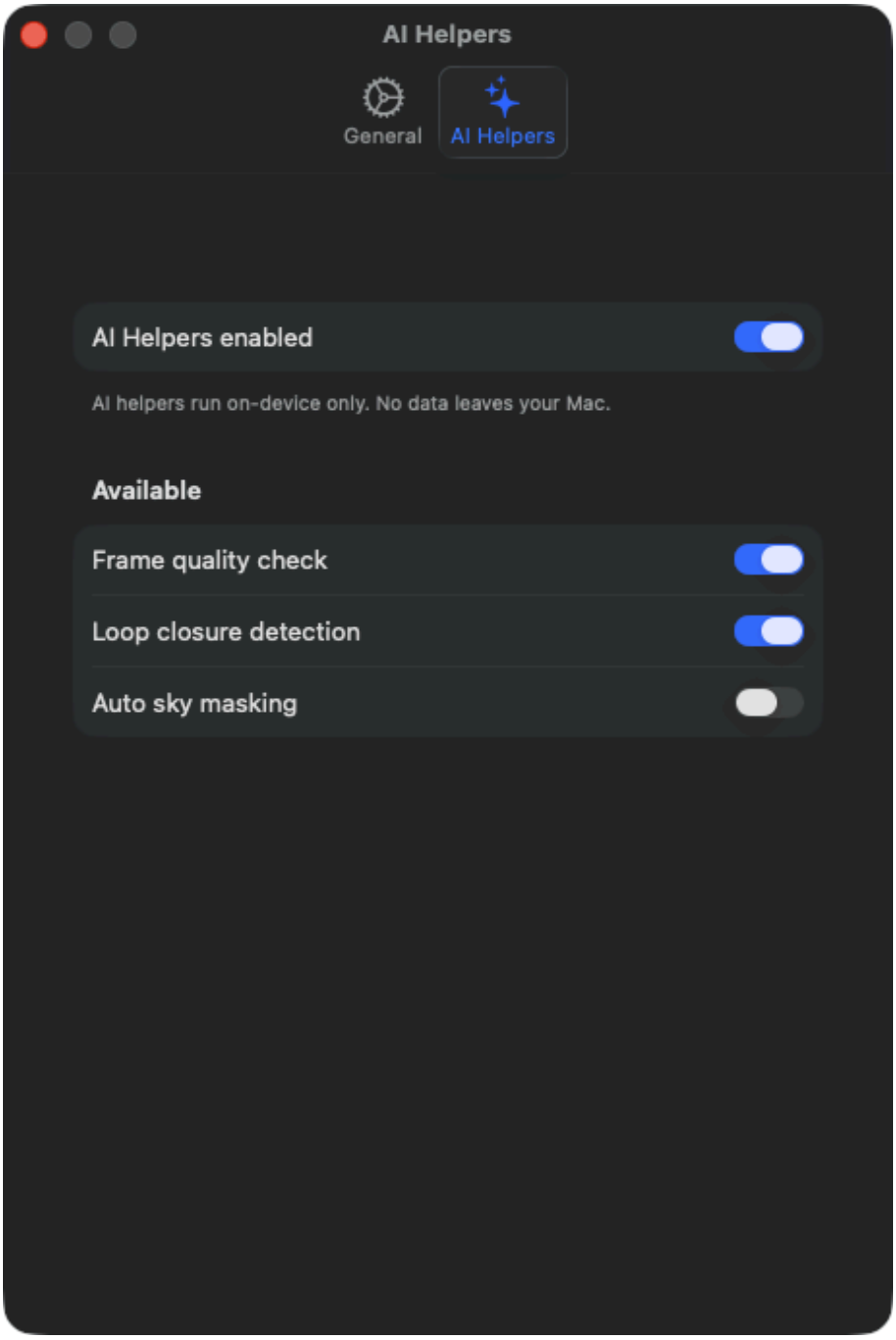

Abbildung 16: Einstellungen → KI-Helfer-Tab mit Master-Schalter und Sub-Toggles

S11 AI Helpers enabled (Master)

WO

Settings → AI Helpers → erste Section → Toggle „AI Helpers enabled„. Bound:. Default: `true`.

TECHNISCH

Master-Schalter über sämtliche AI-Helpers-Features in der Pipeline. Wenn aus, überspringt die Import- und SfM-Pipeline alle ML-basierten Vorverarbeitungs-Stages komplett — kein Apple-Vision-Aufruf, kein CoreML-Model-Load, keine NPU-Aufweckung. Wenn an, werden die individuellen Sub-Toggles (S12–S13) konsultiert. Der Wert wird über Neustarts gemerkt. Wirkt sich auf folgende Stages aus: (a) Frame-Quality-Pre-Check vor SfM (S12), (b) Loop-Closure-Erkennung (S13). Wichtig: bei aus sind die zwei Sub-Toggles deaktiviert und visuell ausgegraut. Footer-Hinweis betont, dass alle AI-Helpers strikt on-device laufen — kein Bild-Upload, keine Cloud-Verarbeitung. Datenschutz-Garantie kommt durch die Verwendung von ausschließlich Apple-Vision-Framework (lokal auf der Neural Engine) und CoreML-Models, die direkt im App-Bundle liegen.

EINFACH GESAGT

Der Hauptschalter für alle Funktionen, die KI/Machine-Learning intern benutzen. Standard ist „an„, weil die Helper viel Zeit sparen, ohne dass deine Bilder den Mac verlassen. Wenn du sie komplett aus haben willst (z.B. um Strom zu sparen oder weil dein Mac kein NPU hat), schalte sie hier aus — dann werden die zwei Unter-Optionen unten automatisch grau und tun nichts mehr.

S12 Frame quality check

WO

Settings → AI Helpers → Available-Section → Toggle „Frame quality check„. Bound:. Default: `true`.

TECHNISCH

Aktiviert den Frame-Quality-Screener (Phase 3.11), der vor dem SfM-Aufruf jeden importierten Frame analysiert. Pipeline-Schritte pro Frame: (a) Laplacian-Variance-Filter aus Apple Vision (Blur-Erkennung — Schwellwert ~150), (b) Histogramm-basiertes Over/Under-Exposure-Check (Schwellwert: >5% Pixel bei 0 oder 255), (c) Blank-Frame-Detect (Standardabweichung < 5 über alle Pixel). Frames, die alle drei Checks bestehen, gehen direkt durch. Frames, die mindestens einen Check failen, lösen einen modalen Confirmation-Dialog aus, der jeden problematischen Frame mit Thumbnail und Begründung listet und fragt, ob er entfernt werden soll. Wichtig: keine automatische Löschung — der Dialog ist immer erforderlich, der Nutzer behält die letzte Entscheidung. Performance: ~50 ms pro Frame auf M3 Ultra, läuft parallel. Bei aus werden alle Frames ungeprüft an SfM weitergereicht. Bei deaktiviertem Master (S11) ist dieser Toggle visuell ausgegraut und ohne Wirkung. Geshippter Status laut Memory: SHIPPED 2026-05-23.

EINFACH GESAGT

Vor dem eigentlichen Training schaut die App jedes Foto an: ist es verwackelt? komplett dunkel oder weiß? leer? Wenn ja, fragt sie dich nach, ob du das Bild rausschmeißen willst — sie entfernt nie automatisch etwas. Das spart später viele Stunden, weil ein einziges total verwackeltes Bild manchmal das ganze Training kaputt machen kann. Standard ist „an„, weil der Aufwand fast null ist und der Nutzen groß.

S13 Loop closure detection

WO

Settings → AI Helpers → Available-Section → Toggle „Loop closure detection„. Bound:. Default: `true`.

TECHNISCH

Aktiviert die Apple-Vision-Feature-Print-basierte Loop-Closure-Erkennung. Für jeden importierten Frame wird ein ~768-dimensionaler Feature-Vektor berechnet, der ein neuronales Embedding des Bildinhalts darstellt. Anschließend werden alle Feature-Prints paarweise per Cosine-Similarity verglichen. Paare mit Similarity > 0.85 und Distanz im Frame-Index > 50 (also nicht-benachbarte Frames) werden als „Loop-Closure-Kandidaten„ identifiziert und in eine Sidecar-JSONL-Datei im Projekt-Ordner geschrieben. Informational only — die importierte Bilder-Sequenz wird nicht modifiziert. Sinn: gibt dem SfM-Solver (insbesondere COLMAP) einen Hinweis, dass diese Frames im 3D-Raum zusammen-cluster gehören. Für native SfM ist die Sidecar-Information aktuell nur dokumentierend; COLMAP nutzt die Hints intern via custom matches-file (manuelle Integration möglich, nicht automatisch verknüpft). Performance: ~200 ms pro Frame auf M3 Ultra, läuft parallel. Bei aus werden keine Feature-Prints generiert. Bei deaktiviertem Master (S11) visuell ausgegraut.

EINFACH GESAGT

Wenn du beim Fotografieren um ein Objekt herumgehst und am Ende wieder am Anfangspunkt landest, hilft es dem Computer enorm, das zu wissen. Diese Option erkennt automatisch, welche deiner Fotos „nahezu vom selben Standort„ aufgenommen wurden, und schreibt das in eine kleine Hilfsdatei. SfM-Tools (vor allem COLMAP) können diese Information nutzen, um eine sauberere 3D-Rekonstruktion zu liefern. Standard ist „an", weil es ohne dein Zutun läuft und nichts an deinen Bildern ändert.

Inspector-Spiegel-Settings

Die übrigen Settings-Einträge (S17–S33) aus der Inventar-Tabelle sind Spiegelungen aus dem Expert-Inspector und in Kapitel 2 (Inspector-Controls I12–I29) dokumentiert. Sie erscheinen nicht physisch im Settings-Fenster, sondern wurden im Inventar nur deshalb gelistet, weil sie über `TrainingConfig`-Properties laufen, die per persistiert werden und insofern formal Settings-Charakter haben. Für inhaltliche Erläuterungen siehe dort.

Wann was?

Setting	Geltungsbereich	Persistenz
S1 Default Mode	App-Global	App-Neustart
S2 Language	App-Global	App-Neustart
S3 Viewport Background	App-Global (Default) + Runtime	App-Neustart
S4 Auto-Rotate After Training	App-Global	App-Neustart
S5 Live Preview Interval	Default für neue Trainings	App-Neustart
S6 Throttle Delay	Default für neue Trainings	App-Neustart
S11 AI Helpers Master	App-Global	App-Neustart
S12 Frame quality check	App-Global	App-Neustart
S13 Loop closure detection	App-Global	App-Neustart

App-Global = wirkt auf alle Projekte. Default für neue Trainings = wirkt nur auf das nächste angelegte Training, laufende Sitzungen bleiben unverändert. Aktuelles Training = wirkt sofort auf die laufende Trainings-Konfiguration, persistiert aber nicht ohne expliziten Reimport.

KAPITEL

Kapitel 4 — Aux-Fenster

Neben dem Hauptfenster (3D-Viewport plus Inspector) verwaltet RadianceKit sieben weitere Fenster, die alle über das Help-Menü geöffnet werden. Die Liste ist von oben nach unten: User Guide (⌘?), Keyboard Shortcuts (⌘/), Open Training Logs… (öffnet keinen App-Fenster, sondern den Finder; deshalb hier nicht weiter behandelt), Manage Storage…, Pareto Dashboard… (⇧⌘D), Holdout Analysis… (⇧⌘H), BayesOpt Console… (⇧⌘B). Drei davon — Dashboard, Holdout, BayesOpt — sind eigenständige Analyse-Werkzeuge. Sie haben jeweils einen eigenen View-Model-Stack, lesen oder schreiben JSON-Dateien auf der Platte, und es gibt für jedes ein CLI-Argument, mit dem du das Fenster gleich beim App-Start auf eine bestimmte Datei zeigen lassen kannst (`--dashboard-dir`, `--holdout-file`, `--bayesopt-autorun`).

Die vier einfachen Fenster (User Guide, Keyboard Shortcuts, Manage Storage, plus die Untermenüpunkte Open Training Logs / Open Exports Folder) bekommen pro Steuerelement einen kurzen Eintrag. Die drei Analyse-Fenster sind ausführlicher dokumentiert — jeweils mit einer Einleitung, die erklärt was du im Fenster siehst, wann du es öffnen solltest und wie du das angezeigte Bild interpretierst.

Am Ende des Kapitels gibt es einen Querverweis-Abschnitt zum Inspector des Hauptfensters: was du im Live-Loss-Chart und in der Gaussian-Count-Anzeige während eines laufenden Trainings sinnvoll ablesen kannst.

User Guide (W1–W4)

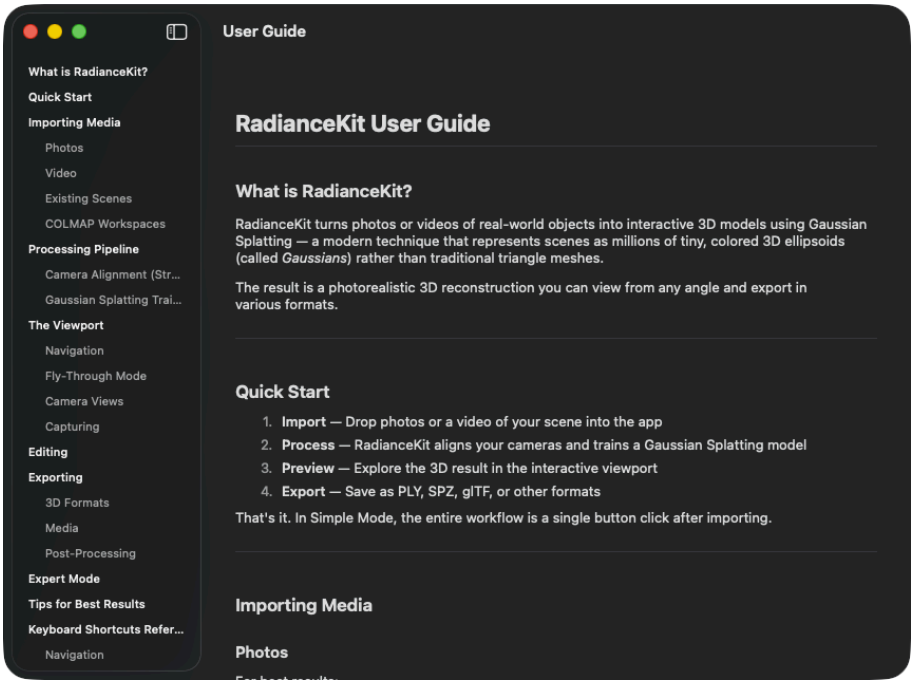

Abbildung 17: User-Guide-Fenster mit Sidebar links und gerendertem Markdown-Inhalt rechts

Was es ist: Ein eingebautes Hilfefenster, das die bei der App mitgelieferte `guide_<sprache>.md` rendert. Die Sprache wird aus den Settings (Tab General → Language) oder, wenn dort „System„ steht, aus den macOS-Sprachpräferenzen abgeleitet. Layout ist klassisch: links Sidebar mit allen Überschriften, rechts der Fließtext.

WANN ÖFFNEN Wenn du eine schnelle Erinnerung an einen einzelnen Punkt brauchst — also als Stichwort-Ersatz. Die ausführliche Referenz ist dieses Manual; das eingebaute Hilfefenster ist eher das, was eine `--help` auf der Kommandozeile wäre. Es wird bei jedem App-Release mit aktualisiert, aber inhaltlich oberflächlicher gehalten.

W1 NavigationSplitView (Sidebar + Detail)

WO

Help → User Guide (⌘?)..

TECHNISCH

Zweispalt-Layout mit schmaler Sidebar (mindestens 180 pt breit) für den Inhaltsbaum und einem scrollbaren Detail-Bereich für den eigentlichen Markdown-Inhalt. Das Fenster hat eine Mindestgröße von 700 × 500 pt. Beim ersten Öffnen lädt das Fenster die passende `guide_<lang>.md` aus dem App-Bundle (Fallback `guide_en.md`), parst sie in Block-Records (Überschriften H1–H4, Absätze, Listen, Tabellen, Trennlinien) und extrahiert separat die Überschriften-Struktur für die Sidebar. Inline-Formatierung (Bold, Italic, Code-Span) wird über die eingebaute Markdown-Engine gerendert. Die Sprache wird aus den App-Einstellungen gelesen, mit dem Spezialfall Chinesisch (`zh-Hans`) und Brasilianisches Portugiesisch (`pt-BR`), die als volle Locale-Tags behalten werden, weil sich diese Varianten von zh bzw. pt unterscheiden.

EINFACH GESAGT

Der eingebaute Hilfetext, links die Themenliste, rechts der Inhalt. Sprache stellt sich automatisch nach deinen System-Settings ein. Funktioniert offline, ist aber bewusst nur eine Kurzversion — die vollständige Referenz ist dieses Manual.

W2 List (Heading-Sidebar)

WO

Linke Spalte im User-Guide-Fenster..

TECHNISCH

Liste über alle H2- und H3-Überschriften des aktuellen Markdown-Dokuments. H2-Einträge erscheinen ohne Einrückung mit Medium-Schriftgewicht, H3-Einträge mit 16 pt Einrückung links und reduziertem Foreground-Style. H4 und tiefer werden ignoriert, weil die Tiefe sonst die Sidebar unübersichtlich macht. Anker-IDs werden aus dem Heading-Text per Slugifizierung erzeugt (lowercase + Spaces zu Dashes + Filterung auf Letters/Numbers/Dashes — derselbe Algorithmus, den GitHub für seine Markdown-Anker verwendet, sodass auch externe URLs zur Doku potenziell auf denselben Anker landen würden). Die Liste verwendet den nativen macOS-Stil.

EINFACH GESAGT

Die Navigationsleiste auf der linken Seite. Tippe einen Eintrag und du springst zum Abschnitt.

W3 Button (Heading → Anchor-Sprung)

WO

Pro Sidebar-Zeile ein Button..

TECHNISCH

Jeder Sidebar-Eintrag ist ein Button, der den aktuellen Anker setzt, optisch aber wie ein Listeneintrag aussieht. Eine Beobachter-Variable triggert dann den Scroll-Sprung zum entsprechenden Anker mit einer weichen Animation über 0,3 s. Nach dem Sprung wird der Anker-Wert zurückgesetzt, damit der nächste Klick auf denselben Anker wieder feuert (sonst würde der Beobachter nicht erneut auslösen, weil sich der Wert nicht geändert hat).

EINFACH GESAGT

Klick führt dich an die passende Stelle im Text rechts.

W4 ScrollView (Detail-Inhalt)

WO

Rechte Spalte..

TECHNISCH

Scrollbarer, vertikal-stapelnder Inhaltsbereich mit Lazy-Rendering, weil längere Guides leicht über 200 Markdown-Blöcke haben können — eine nicht-lazy Variante würde alle gleichzeitig instanziieren. Jeder Block bekommt eine eigene ID, entweder den Heading-Anker (für sprungbare H1–H3) oder einen Index-Platzhalter. Maximalbreite ist 720 pt, Padding 32 horizontal / 24 vertikal, sodass lange Zeilen ein gut lesbares Layout behalten. Tabellen werden zellenweise mit horizontalen Stacks und Trennlinien gerendert; Inline- Code durch die eingebaute Markdown-Engine. Echte Codeblöcke werden aktuell als Paragraph behandelt — eine bekannte Einschränkung des Help-Fensters.

EINFACH GESAGT

Der eigentliche Hilfetext. Scrollbar, gut lesbare Breite, klare Typo.

Keyboard Shortcuts (W5–W6)

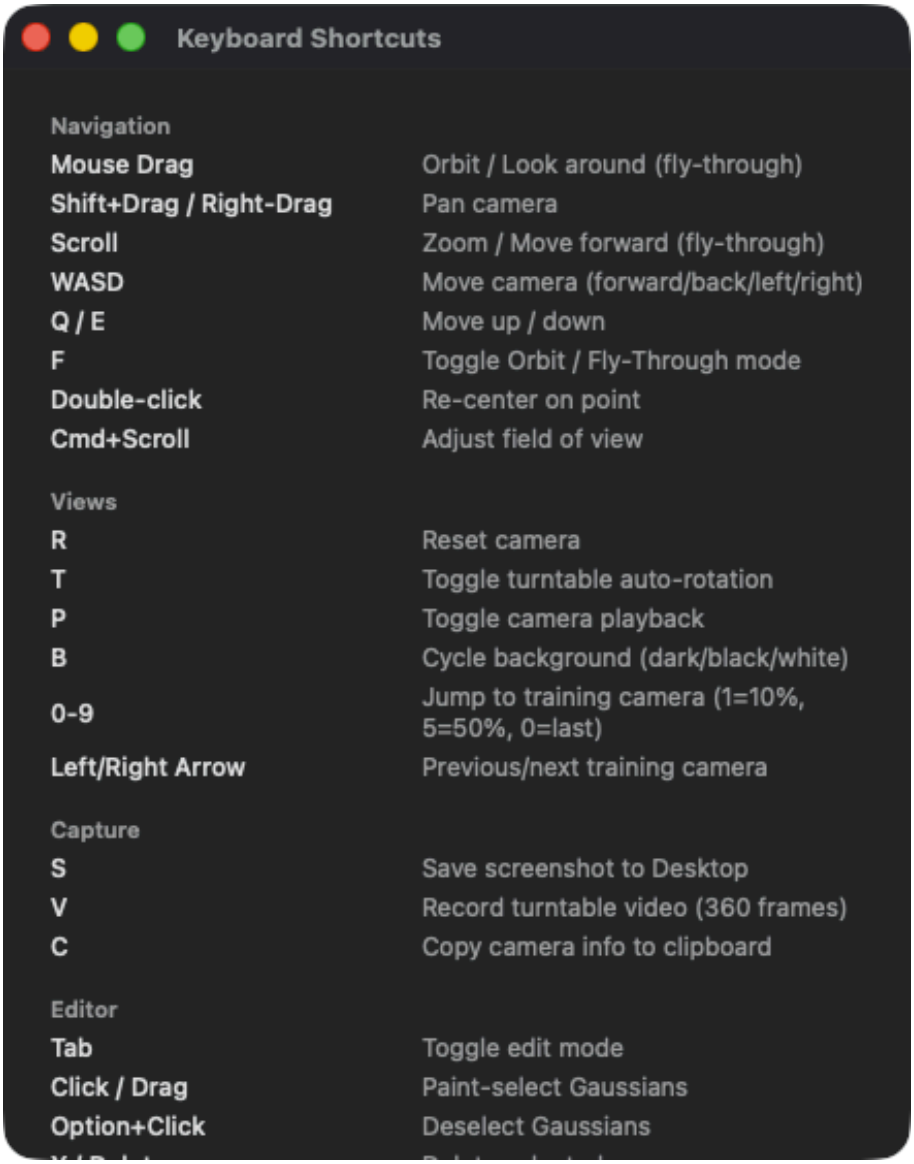

Abbildung 18: Keyboard Shortcuts Fenster — fünf Gruppen Navigation/Views/Capture/ Editor/Training mit Hotkey-Spalte links und Beschreibung rechts

WAS IM BILD ZU SEHEN IST Statische Referenz-Liste in fünf Sektionen. **Navigation:** Mouse Drag (Orbit/Fly), Shift+Drag/Right-Drag (Pan), Scroll (Zoom), WASD (Fly-Through-Bewegung), Q/E (Up/Down), F (Toggle Orbit/Fly), Double-click (Re-center), Cmd+Scroll (FoV-Adjust). **Views:** R (Reset Camera), T (Auto-Rotation), P (Camera Playback), B (Background-Cycle), 0–9 (Springe zu Training-Cam 1=10%/5=50%/0=last), Left/Right Arrow (Prev/Next Cam). **Capture:** S (Screenshot to Desktop), V (Turntable-Video), C (Copy Camera Info). **Editor:** Tab (Edit-Modus), Click/Drag (Paint-Select), Option+Click (Deselect), X / Delete (Selektion löschen), Cmd-Z (Letzte Löschung rückgängig), [/] (Pinselgröße kleiner/größer), Esc (Selektion aufheben). **Training:** Start, Pause/Resume, Cancel, Continue +5K/+10K/+20K über die Menü-Shortcuts in M9–M14.

Was es ist: Eine simple statische Übersicht aller Tastenkürzel — Navigation, Views, Capture, Editor, Training. Inhalt ist hartkodiert in, kein Markdown-Loading.

WANN ÖFFNEN Wenn du nach dem schnellsten Weg suchst, etwas im Viewport zu tun. WASD-Fly-Through, R für Camera-Reset, B fürs Background-Cycling — alle stehen hier.

W5 ScrollView (Inhaltsbereich)

WO

Help → Keyboard Shortcuts (⌘/)..

TECHNISCH

Ein einfacher Scroll-Bereich mit einer vertikalen Liste drin. Padding 20 rundherum, kein Sidebar-Navigations-Tree (die Liste ist kurz genug). Inhalte sind in fünf Sektionen gruppiert (Navigation, Views, Capture, Editor, Training). Pro Tastenkombination eine Zeile mit übersetzbarem Text in beiden Spalten. Linke Spalte (Tastencode) fixiert auf 180 pt Breite, sodass die Beschreibungen rechts vertikal aligned bleiben. Keine Interaktion außer Scrollen — Klicken auf eine Zeile löst nichts aus, die Tastenkürzel sind echte Tastatur-Modifier im Menü und auf dem Viewport.

EINFACH GESAGT

Tabelle aller Shortcut-Tasten. Statisches Cheat-Sheet zum kurzen Nachschauen.

W6 VStack (Shortcut-Sektionen)

WO

Innerhalb des ScrollView..

TECHNISCH

Linksbündig gestapelte Sektionen mit 16 pt Abstand. Innerhalb der fünf Sektionen jeweils Heading + Zeilen-Folge. Headings nutzen einen sekundären Subheadline-Stil — bewusst kein Title-Format, weil die Sektionen nicht navigierbar sein müssen. Inhalt ist bewusst flach (kein Disclosure, kein Search, kein Filter), damit die Komponente auf jeder macOS-Version unverändert läuft und die Datei lesbar bleibt.

EINFACH GESAGT

Die Gruppierung der Tasten nach Funktion (Navigation, Views, Editor und so weiter).

Manage Storage (W7–W12)

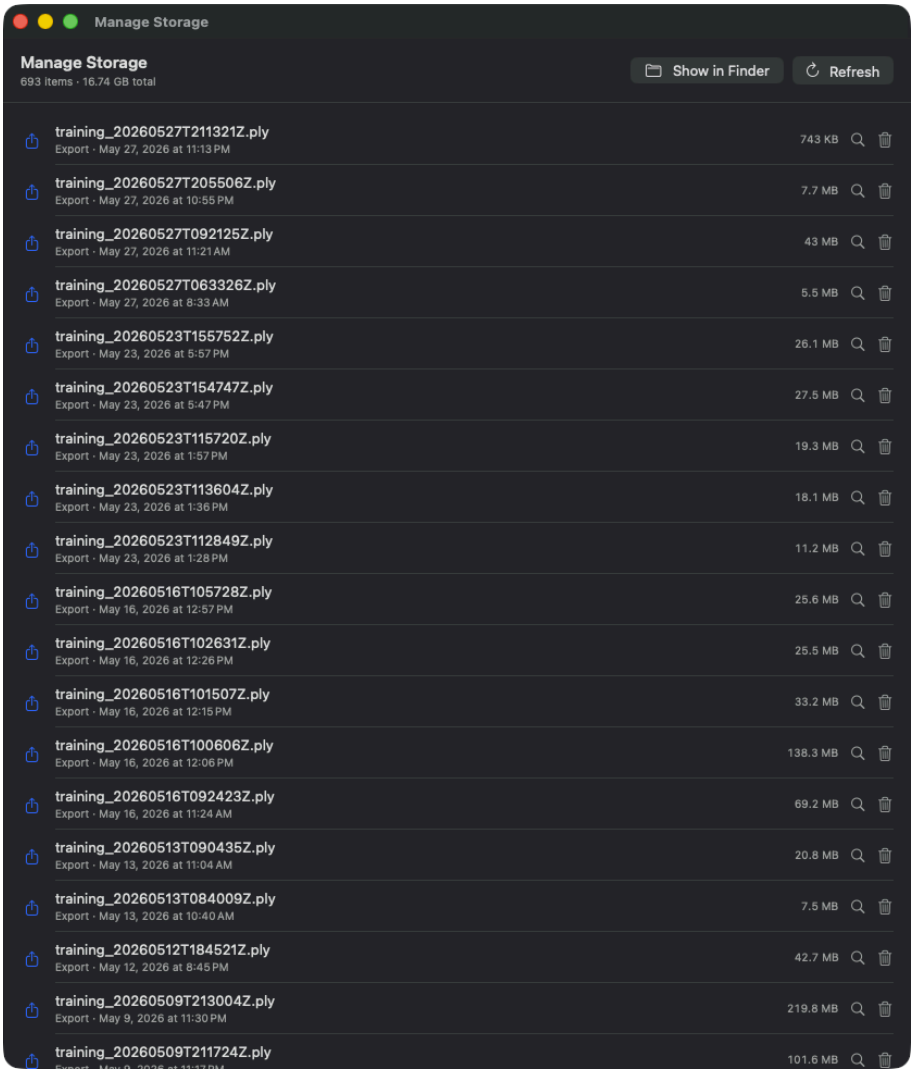

Abbildung 19: Manage Storage Fenster — Header zeigt „693 items · 16.74 GB total„, Tabelle mit Export-PLY-Dateien sortiert nach Datum, jeweils Format-Pill + Dateiname + Größe + Datum

WAS IM BILD ZU SEHEN IST Tabellen-Ansicht aller von RadianceKit verwalteten Files. Header zählt 693 Items, 16.74 GB Gesamtgröße. Toolbar oben: „Show in Finder„ + „Refresh“. Jede Zeile: PLY-Icon, Dateiname (z.B. `training_20260527T211321Z.ply`), Export-Datum, Größe (variiert 7 KB bis 218 MB), Lupen-Icon (Reveal) und Papierkorb-Icon (Move to Trash). Files sind nach Datum sortiert, neueste oben. In dieser Demo-Aufnahme dominieren PLY-Exports, weil viel mit `--benchmark` gearbeitet wurde.

Was es ist: Eine Disk-Usage-Übersicht für alles, was RadianceKit unter `~/Documents/RadianceKit/` ablegt — Logs, Exports, Scenes, Capture-Bundles (vom iOS-Begleiter), Imports (Staging-Kopien der Eingabe-Bilder). Pro Eintrag eine Größe in Bytes und zwei Buttons: „im Finder anzeigen„ und „in den Papierkorb verschieben“. Ist KEIN automatisches Aufräumen — die App löscht selbst nichts; du entscheidest pro Eintrag.

WANN ÖFFNEN Wenn die Platte voll wird. Vor allem die Logs sammeln sich (eine JSONL pro Trainingsversuch, plus die `_qualityMetrics.json`); die Exports natürlich auch (PLY

100% rohe Daten, einer pro Export). Auch nützlich nach einem Crash, wenn das Imports-Staging-Verzeichnis noch alte Kopien der Eingangsbilder herumliegen hat (siehe „Disk-pressure incident„ in `dev_v549f-needle-reduction.md`).

W7 Button „Show in Finder„

WO

Header oben rechts im Storage-Browser-Fenster..

TECHNISCH

Öffnet das gesamte RadianceKit-Verzeichnis (`~/Documents/RadianceKit/`) im Finder, sodass du die Ordnerstruktur direkt sehen und auch mit dem Finder selbst manipulieren kannst. Die Aktion öffnet ein neues Finder-Fenster und wechselt nicht in den App-Sandbox-Container — `~/Documents/RadianceKit/` ist die regulär für Apps zugängliche Documents-Domain, kein Sandboxed-Container-Pfad.

EINFACH GESAGT

Öffnet das Verzeichnis im Finder, damit du selbst mit den Dateien hantieren kannst.

W8 Button „Refresh„

WO

Header, neben dem Finder-Button..

TECHNISCH

Triggert einen Hintergrund-Scan, der auf einem nutzer-initiierten asynchronen Task läuft, damit das Scannen großer Verzeichnisbäume die UI nicht blockiert. Das eigentliche Walken geht jeden bekannten Unterordner (Logs, Exports, Scenes, Captures, Imports) durch und erzeugt pro direktem Kind einen Storage-Eintrag. Pro Eintrag wird die rekursive Größe ermittelt — bevorzugt der tatsächliche Plattenverbrauch (inklusive APFS-Hardlinks-Sharing) mit Fallback auf die logische Dateigröße.

EINFACH GESAGT

Liest die Liste neu, falls du zwischendurch im Finder etwas gelöscht oder hinzugefügt hast.

W9 List (Storage-Einträge)

WO

Hauptinhalt unter dem Header..

TECHNISCH

Liste mit pro Zeile diesem Layout: kategoriespezifisches SF-Symbol-Icon (Dokument für Logs, Upload-Pfeil für Exports, Würfel für Scenes, Tray für Imports), Name + Untertitel (Kind-Label + formatiertes Modifications-Datum), Bytes-Counter rechts (rechtsbündig, monospaced), Reveal-Button (Lupe-Symbol), Trash-Button (Papierkorb). Sortierung: primär nach Kind (Scenes zuerst, dann Exports, Logs, Captures, Imports, Other), sekundär nach Modifikationsdatum absteigend (neueste oben). Wenn der Scan noch läuft, zeigt die Stelle stattdessen einen „Scanning…„-Fortschritt. Wenn nichts gefunden wurde, eine Empty-State-Anzeige mit Tray-Icon.

EINFACH GESAGT

Liste aller deiner RadianceKit-Daten, sortiert nach Typ und Aktualität. Pro Eintrag siehst du Größe und kannst direkt löschen.

W10 Row-Button „Reveal in Finder„

WO

Pro Zeile, Lupe-Symbol rechts..

TECHNISCH

Öffnet den Finder und selektiert das spezifische Item (Datei oder Ordner). Unterschied zu W7: W7 öffnet das Root-Verzeichnis; W10 markiert genau diesen einen Eintrag. Praktischer Workflow: identifiziere einen großen Eintrag, klick auf die Lupe, kopiere ihn dann zum Beispiel auf ein externes Volume.

EINFACH GESAGT

Springt im Finder direkt auf diesen Eintrag, damit du ihn schnell findest.

W11 Row-Button „Move to Trash„

WO

Pro Zeile, Papierkorb-Symbol rechts neben der Lupe..

TECHNISCH

Löst die Bestätigungs-Dialog-Box (W12) aus. Erst nach Bestätigung läuft die macOS-Standard-Operation „in den Papierkorb verschieben„ (also reversibel, kein direktes Löschen). Nach erfolgreichem Trash wird der Eintrag aus der Liste entfernt und der Gesamtbyte-Counter aktualisiert. Bei Fehlern wird ein modaler Fehler-Dialog eingeblendet.

EINFACH GESAGT

Verschiebt den Eintrag in den Papierkorb. Dialog fragt vorher nach.

W12 ConfirmationDialog (Lösch-Bestätigung)

WO

Wird durch W11 ausgelöst, dargestellt als macOS-Sheet..

TECHNISCH

Standard-Bestätigungsdialog mit dynamischem Titel „Delete <name>?„ und einer Message-Zeile, die explizit darauf hinweist, dass der Eintrag im Papierkorb landet und von dort wiederherstellbar ist (bis der Papierkorb geleert wird). Zwei Buttons: „Move to Trash“ als destruktive Aktion (rot dargestellt) und „Cancel„ mit automatischer Esc-Bindung. Der Dialog ist non-modal in dem Sinne, dass er nur dieses Fenster blockiert, nicht die ganze App — das ist macOS-Standard für reversible Löschungen.

EINFACH GESAGT

Sicherheitsfrage vor dem Löschen. „Move to Trash„ ist reversibel — solang der Papierkorb nicht geleert ist.

Pareto Dashboard (W13–W22)

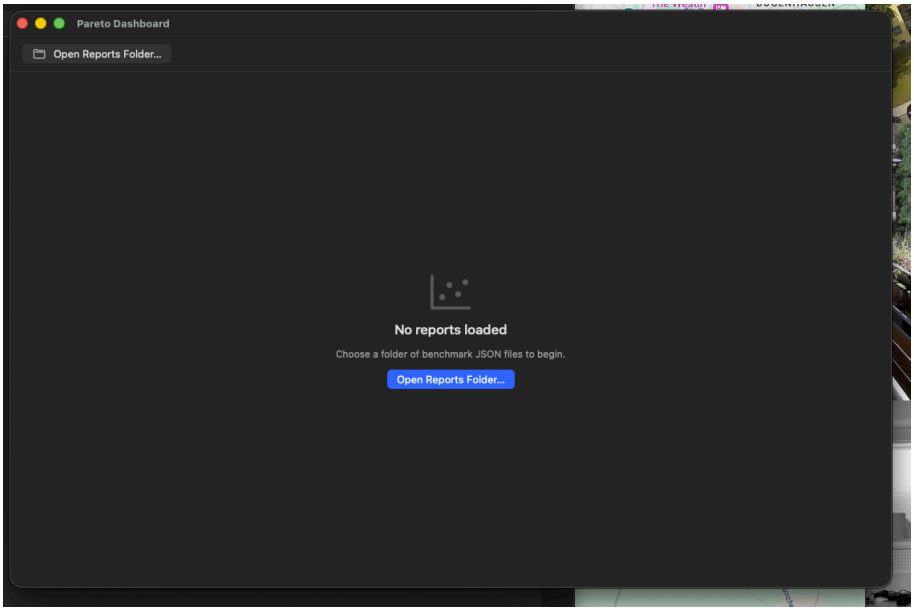

Abbildung 20: Pareto Dashboard — leerer Zustand vor Report-Import

Leerer Zustand (nach erstem Öffnen) — Empty-State mit Call-to-Action „Open Reports Folder…„. Die Datenpunkte erscheinen, sobald Trainings-Reports geladen sind, siehe nächster Shot.

Abbildung 21: Pareto Dashboard mit 129 geladenen Benchmark-Reports — Gaussians vs PSNR mit Pareto-Front, Scene/Strategy/Mip-Filter

WAS IM BILD ZU SEHEN IST Header-Toolbar zeigt „129 reports of 129" (alle Reports im gewählten Ordner erfolgreich geparst — 21 zusätzliche Files konnten wegen älterem Format nicht geparst werden, siehe Hinweis-Liste rechts). Achsen: X-Axis-Picker auf `Gaussians`, Y-Axis-Picker auf `PSNR (dB)`. Scatter-Plot: grüne Punkte = Classic-Strategy, blaue Punkte = MCMC. Die gestrichelte Pareto-Front-Linie verläuft entlang der

best-erreichten PSNR-Werte und plateaut um PSNR≈30 dB ab ca. 500K Gaussians. Filter-Chips rechts: 7 Scenes (bicycle, bonsai, family, flowers, kitchen, stump, truck), 2 Strategies (classic, mcmc), 3 Mip-Splatting-Optionen (All, On, Off). Aktuell sind alle Filter offen, deshalb der dichte Punkte-Cluster.

Was es ist: Ein Multi-Run-Vergleichs-Werkzeug. Trainiert hast du in der Vergangenheit mehrere Szenen oder dieselbe Szene mit unterschiedlichen Presets — jeder dieser Trainingsläufe produziert (wenn du `--benchmark` mitgegeben oder über die Benchmark-Funktion aufgerufen hast) eine JSON-Report-Datei, die unter anderem Final-PSNR, SSIM, LPIPS, Gaussian-Count und Wallclock-Zeit enthält. Das Dashboard liest einen ganzen Ordner solcher Reports gleichzeitig ein und plottet sie als 2D-Scatter mit selektierbaren Achsen. Zusätzlich wird die Pareto-Front (die Menge der nicht-dominierten Punkte) als gestrichelte Linie eingezeichnet.

WANN ÖFFNEN Nachdem du mindestens drei oder vier Trainings-Reports angelegt hast. Mit weniger Punkten ist die Frontier-Linie nicht aussagekräftig. Typischer Use-Case: du hast versucht, eine Outdoor-Szene zu rekonstruieren, und hast nacheinander P3 Balanced (Classic), P4 Quality (Classic), P7 MCMC Quality und P9 Outdoor (tuned) durchgespielt — jetzt willst du wissen, welche Konfiguration die beste PSNR pro Sekunde Trainingszeit liefert oder welche die wenigsten Gaussians braucht für gegebene PSNR.

WIE DEUTEN Beide Achsen sind frei wählbar (X-Achse:,, `psnr`, `ssim`, `lpips`, …; Y-Achse genauso). Die Pareto-Front-Logik in `ParetoFront2D.indices` weiß für jede Metrik, ob „kleiner = besser„ (z.B. LPIPS, Loss, Time) oder „größer = besser“ (PSNR, SSIM) — die Linie verläuft also je nach Achsen-Wahl von links unten nach rechts oben oder von links oben nach rechts unten, immer entlang der besten erreichten Kombination. Ein Punkt ist Pareto-optimal, wenn KEIN anderer Punkt in BEIDEN Dimensionen mindestens gleich gut ist (also kein anderer dominiert ihn). Pareto-optimale Punkte liegen auf der Linie, andere Punkte rechts/oberhalb (je nach Achsenorientierung) davon. Punkte AUF der Linie sind die echten Kandidaten für „bestes Preset„; Punkte WEIT von der Linie sind verschwendete Trainingszeit.

FILTER-CHIPS Du kannst die Auswahl auf eine bestimmte Szene einschränken (wenn du z.B. nur Outdoor-Runs vergleichen willst), auf eine bestimmte Strategie (Classic oder MCMC), oder auf Mip-Splatting an/aus (relevant nach Phase Q1.5, wo Mip als opt-in Advanced Flag bleibt).

Du hast drei Reports für „truck„-Szene unter `~/Documents/RadianceKit/Reports/`: Run A (P4 Quality, 40K iter, 524K Gs, 105 s, PSNR 23.4), Run B (P7 MCMC, 200K iter, 150K Gs, 693 s, PSNR 24.6), Run C (P9 Outdoor, 100K iter, 1.25M Gs, 312 s, PSNR 25.8). Setze X-Achse auf trainingTime, Y-Achse auf PSNR. Run B liegt rechts oben, Run C noch weiter rechts oben, Run A links unten. Die Pareto-Front verbindet A und C — beide nicht-dominiert. Run B ist „lost“ (C ist besser in Time UND PSNR). Erkenntnis: für „truck„ lohnt sich der MCMC-Default nicht; entweder schnell+ok (A) oder lang+sehr gut (C). Konfiguration aus C als eigenes Preset speichern (Inspector → I1 Save Preset).

Nächste Aktion: Beste Konfiguration als Preset abspeichern. Konkret: schau dir die Pareto-Punkte an (Hover zeigt PSNR/SSIM/LPIPS/Gs/Time im Tooltip), entscheide welcher dir vom Time-vs-Quality-Trade-off am besten passt, öffne den zugehörigen Report (Dateiname enthält Run-Timestamp), kopiere dessen Trainingskonfiguration in einem

neuen Run oder speichere sie nach der nächsten Trainings-Session als Preset über den Inspector.

W13 Button „Open Reports Folder…„

WO

Toolbar oben links..

TECHNISCH

Öffnet einen Ordner-Auswahldialog mit der Aufforderung „Select a folder containing benchmark .json reports„. Nach Bestätigung läuft ein Hintergrund-Task, der alle `.json`-Dateien im Ordner sequentiell parst. Fehlerhafte Reports (kaputtes JSON, falsches Schema) werden gesammelt und unten in der Sidebar als „N file failed to parse“ angezeigt — kein Crash. Wenn ein zweiter Klick erfolgt, während ein erster Load noch läuft, wird der vorherige Task abgebrochen, sodass nicht zwei Ergebnisse gleichzeitig in den State schreiben.

Auch über CLI: `--dashboard-dir /pfad/zu/reports` lädt den Ordner gleich beim App-Start.

EINFACH GESAGT

Wählt den Ordner, in dem deine Benchmark-Reports liegen. Standardpfad ist `~/Documents/RadianceKit/Reports/`. Lädt dann alle JSONs auf einmal.

W14 Picker „X-Axis„

WO

Über dem Chart, links..

TECHNISCH

Menü-Picker mit allen verfügbaren Metrik-Achsen des Dashboard-Moduls (PSNR, SSIM, LPIPS, Gaussian-Count, Trainingszeit und so weiter). Default ist Gaussian-Count. Beim Wechsel wird der gehoverte Punkt zurückgesetzt, weil eine bisher gehighlightete Position im alten Achsen-Koordinatensystem nach Achsenwechsel keinen Sinn mehr macht. Picker ist auf Inhaltsbreite begrenzt, damit er nicht über die ganze Breite zieht.

EINFACH GESAGT

Welche Metrik auf der waagerechten Achse stehen soll. Üblicherweise „Trainingszeit„ oder „Gaussian-Anzahl“, weil das die „Kosten„ sind, die du vergleichen willst.

W15 Picker „Y-Axis„

WO

Über dem Chart, neben X-Axis..

TECHNISCH

Identisch zu W14, nur dass das Default PSNR ist. Die Achsenwahl wird unabhängig gespeichert, also kann der Nutzer auch Unsinns-Kombinationen wählen (X=PSNR, Y=PSNR — würde alle Punkte auf eine Diagonale werfen). Solche Kombinationen werden aber nicht abgefangen; bewusste Entscheidung, weil ein Vergleich „SSIM vs PSNR„ durchaus interessant ist, um zu sehen wie konsistent die Metriken sich verhalten.

EINFACH GESAGT

Was auf der senkrechten Achse steht. Normalerweise „PSNR„ oder „SSIM“ als Qualitätsmaß.

W16 Toggle „Show Pareto Front„

WO

Rechts neben den Achsen-Pickern..

TECHNISCH

Standard-macOS-Toggle. Wenn aktiv, wird im Pareto-Chart zusätzlich zur Punktwolke eine Linie mit der errechneten 2D-Pareto-Front gezeichnet. Stil: gestrichelt (Strichmuster 4–4), grau halbtransparent, Linienstärke 1,5 pt. Die Pareto-Berechnung läuft auf dem Hauptthread — bei der typischen Anzahl Reports (≤ ~50) ist das problemlos schnell. Wenn der Toggle aus ist, wird die Linie weggelassen, sodass nur die nackten Punkte stehen.

EINFACH GESAGT

Zeigt die Linie an, die durch die „bisher besten„ Punkte führt. Wenn dir die Linie im Weg ist (z.B. weil du nur die einzelnen Trades vergleichen willst), schalte sie aus.

W17 Chips „Scene„-Filter

WO

Rechte Sidebar im Dashboard-Fenster..

TECHNISCH

Filter-Chips für jede in den geladenen Reports vorkommende Szene. Eigenes Flow-Layout, das Chips in mehrere Zeilen automatisch umpackt, sobald die Breite ausgereizt ist. Aktive Chips bekommen den Akzent-Hintergrund, inaktive einen neutralen Standard-Material-Hintergrund. Mehrfach-Auswahl ist möglich (Set-Semantik); wenn keine Chip selektiert ist, gelten alle Szenen als „durchgelassen„ — d.h. die Set-Logik ist „leere Selektion = alles“, nicht „leere Selektion = nichts„.

EINFACH GESAGT

Klick auf einen Szenen-Namen filtert die Punkte auf nur diese Szene. Mehrfachauswahl möglich. Leer = alle Szenen.

W18 Chips „Strategy„-Filter

WO

Unter Scene-Filter in der Sidebar..

TECHNISCH

Genau wie W17, aber für Trainings-Strategien — typischerweise die zwei Werte „classic„ und „mcmc“, abgeleitet aus dem Strategy-Feld der Benchmark-Report-JSONs. Hilfreich, wenn du Reports beider Strategien gemischt hast und nur eine Sorte sehen willst (z.B. „nur MCMC-Runs zeigen, weil ich Classic schon ausgeschlossen habe„).

EINFACH GESAGT

Filter nach Klassisch oder MCMC. Standardmäßig sind beide aktiv.

W19 Chips „Mip-Splatting„-Filter

WO

Unter Strategy-Filter in der Sidebar..

TECHNISCH

Dreiwertiger Filter (statt Set wie W17/W18): „All„ / „On“ / „Off„. Hintergrund: Mip-Splatting wurde in Phase Q1.5 als experimentelle Multi-Skalen-Verbesserung evaluiert und der finale Verdict war „kein netter Win durchgehend; behalten als opt-in Flag“. Wenn du Mip-on/off-Vergleiche machst, willst du oft sehr scharf trennen können. Daher der dedizierte ternäre Filter mit den Zuständen „alles durchlassen„, „nur Mip an“, „nur Mip aus„. Die Sidebar-Section wird nur eingeblendet, wenn mindestens ein Mip-Report UND mindestens ein Nicht-Mip-Report in der Daten-Menge ist (sonst macht die Filterung keinen Sinn).

EINFACH GESAGT

Wenn du Mip-Splatting an/aus vergleichen willst, hier dreigeteilter Filter. Sonst ignorieren.

W20 ChipButton (Filter-Toggle, all/on/off)

WO

Helper-Component, wird in W17/W18/W19 verwendet..

TECHNISCH

Minimalistischer Button-Wrapper. Inhalt: Label- Text mit Caption-Schriftgrad und Padding 10 horizontal / 5 vertikal. Hintergrund konditional: wenn aktiv → App-Akzentfarbe mit weißem Text; sonst neutraler Standard-Material-Hintergrund mit schwarzem Text. Form ist eine Capsule (pillenartig). Plain-Buttonstyle, damit das Capsule- Material nicht von einem System-Border überlagert wird.

EINFACH GESAGT

Die runden Filter-Knöpfe selbst. Optisch wie ein iOS-Tag.

W21 Chart (Pareto-Scatter)

WO

Mittelfläche des Dashboards..

TECHNISCH

Swift-Charts-Diagramm mit zwei Layern: 1. ein Punkt pro Report — Position aus den gewählten X- und Y-Metriken, Farbe nach Strategy, Symbol nach Mip-Status. Symbol-Größe normal 80, gehighlighted 200 (wenn die ID dem aktuell gehoverten Report entspricht).

2. eine Linie für die Pareto-Front, nur wenn der Toggle an ist.

Chart-Overlay: ein transparentes Rechteck registriert Maus-Bewegung; pro Frame wird die euklidisch nächste Punktposition im Plot-Frame ermittelt und der gehoverte Report aktualisiert, falls die Distanz unter 24 px liegt (sonst zurückgesetzt). So bekommst du den Tooltip ohne Klicken — Hovern reicht.

EINFACH GESAGT

Das eigentliche Streudiagramm. Jeder Punkt ist ein Trainingslauf. Hover für Detail-Tooltip.

W22 Tooltip (Hover-Detail)

WO

Unter dem Chart, eingeblendet bei Hover..

TECHNISCH

Horizontaler Stack: Szene-Name (Headline), Strategy-Tag (Caption), Trennlinie, dann PSNR/SSIM/LPIPS/Gs/Time-Metriken in jeweils einer kleinen vertikalen Gruppe (Label + monospaced Wert). Falls Mip aktiviert war, zusätzlich ein „Mip„-Capsule-Tag in Akzentfarbe. Hintergrund halbtransparenter Blur, abgerundetes Rechteck mit 8 pt Radius. Wird nur eingeblendet, wenn die Maus tatsächlich über einem Punkt ist. Verschwindet automatisch beim Verlassen.

EINFACH GESAGT

Die Detail-Karte unten, wenn du mit der Maus über einen Punkt gehst. Zeigt alle Qualitätsmetriken und die Run-Konfiguration auf einmal.

Holdout Analysis (W23–W29)

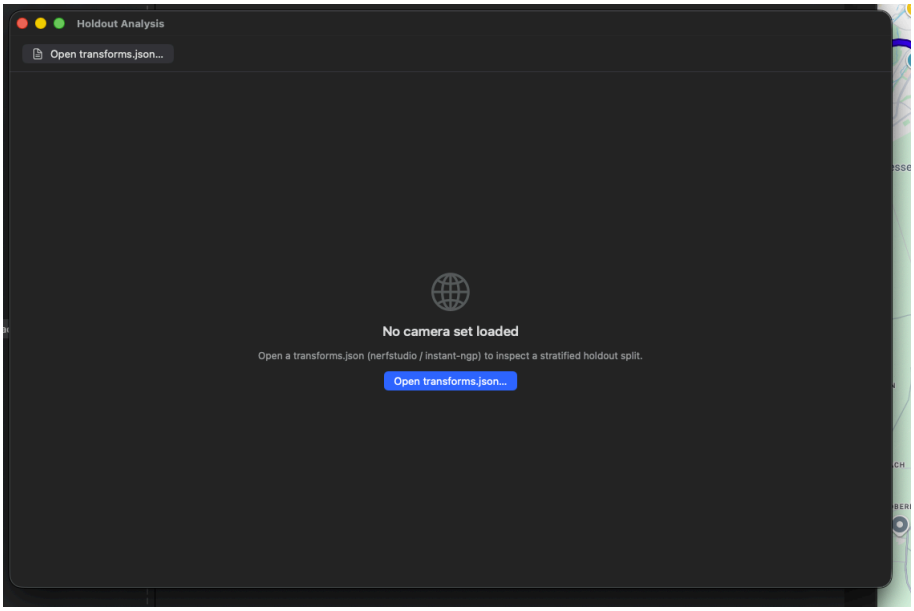

Abbildung 22: Holdout Analysis — leerer Zustand vor dem Laden einer transforms.json

Leerer Zustand mit Empty-State und Call-to-Action „Open transforms.json…„. Akzeptiert NeRF-Studio- und Instant-NGP-Format.

Leerer Zustand (nach erstem Öffnen) — die Kamera-Marker erscheinen, sobald eine `transforms.json` geladen ist, siehe nächster Shot.

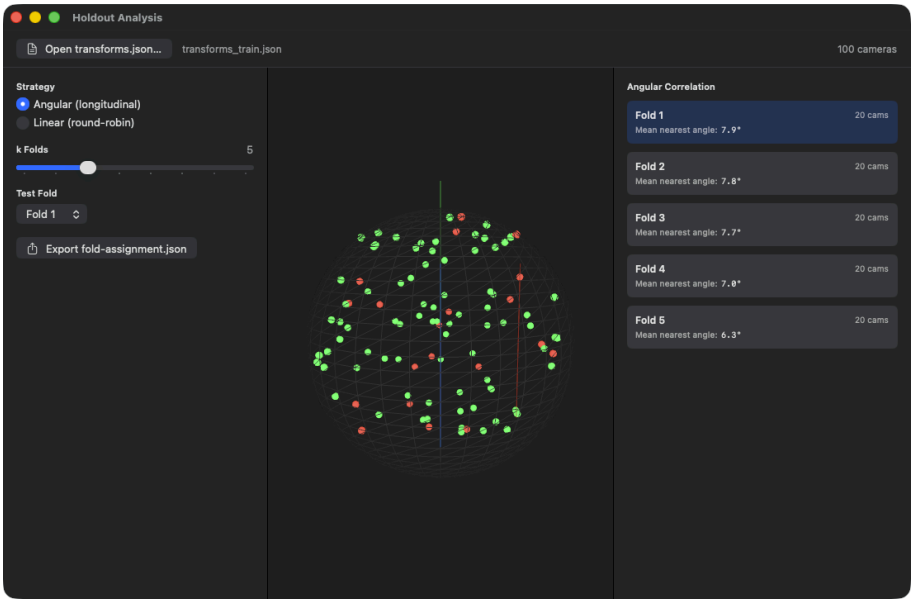

Abbildung 23: Holdout-Globe mit 100 NeRF-Blender-Mic-Kameras, 5 Folds à 20 Kameras, Angular-Strategy aktiv

WAS IM BILD ZU SEHEN IST Header zeigt geladene Datei (`transforms_train.json`) und Cam-Count („100 cameras„). Linke Sidebar: Strategy-Picker mit zwei Optionen — `Angular (longitudinal)` aktiv (richtet Folds nach Längen-/Breiten-Sektoren auf der Sphäre aus, sodass jeder Test-Fold geometrisch dicht ist) vs `Linear (round-robin)`

(Reihenfolge-basiert, alle k-ten Frames als Test-Set). k-Folds-Slider steht auf 5, Test-Fold-Picker auf Fold 1. Export-Button erzeugt eine `fold-assignment.json` für Nerfstudio/Instant-NGP. Mittel-Panel: 3D-Globe-Projektion aller 100 Kameras — grüne Punkte = Train, rote Punkte = aktueller Test-Fold (Fold 1 mit 20 Kameras). Rechte Sidebar (Angular Correlation): pro Fold 20 Cams + Mean Nearest Angle (Fold 1: 7.9°, Fold 2: 7.8°, Fold 3: 7.7°, Fold 4: 7.0°, Fold 5: 6.3°) — kleinerer Wert bedeutet, dass die Kameras innerhalb dieses Folds eng beieinander liegen, also der Holdout-Split räumlich kohärent ist.

Was es ist: Ein 3D-Visualisierer für deine Kamera-Anordnung mit Cross-Validation-Logik. Du lädst eine `transforms.json` (das Standardformat von Nerfstudio / Instant-NGP für Kamera-Posen), die App liest alle Kameras, projiziert ihre Blickrichtungen auf eine Einheitskugel und zeigt sie als kleine Kugel-Marker auf einem virtuellen Globus. Dann teilt sie die Kameras in `k` Folds auf (per gewählter Strategie: angular oder linear), markiert grün den Trainings-Anteil und rot den Test-Anteil (Holdout), und berechnet pro Fold einen Angular-Correlation-Score, der dir sagt wie weit der Test-Fold im Blickwinkel-Raum vom Trainings-Fold entfernt liegt.

WANN ÖFFNEN Wenn du Holdout-Evaluation machen willst — also: wie gut generalisiert dein Modell auf ungesehene Blickwinkel? Standard im Training ist „every-8th view als Holdout„ (Mip-NeRF360-Konvention), aber das ist eine sehr lineare Aufteilung. Wenn deine Bilder zum Beispiel zeitlich geclustert sind (erst eine Seite des Objekts, dann die andere), dann ist „every-8th" nicht repräsentativ — eine zufällige Sequenz-Position landet im Test, aber alle ihre Nachbarn sind im Training, das ist zu einfach. Mit „angular„ stratifiziert man stattdessen über den Blickwinkel-Raum: jeder Fold enthält Kameras aus allen Bereichen des Orbits, sodass der Test wirklich Generalisierungs-Lücken testet.

WIE DEUTEN Angular vs Linear: - Angular (Standard): teilt die Kameras nach Longitudinalwinkel (ϕ-Koordinate um die Y-Achse) in `k` gleiche Sektoren. Fold 0 sind Kameras mit ϕ ∈ [0°, 360/k°), Fold 1 die nächsten, und so weiter. Vorteil: jeder Fold deckt einen Teilausschnitt des Orbits ab; der Test-Fold ist räumlich kompakt aber breit über den Welt-Datensatz verteilt. Gut für klassische Orbit-Aufnahmen. - Linear (Round-Robin): Fold-Index = (image_index modulo k). Das ist die simple „every-k-th„-Aufteilung. Funktioniert, wenn die Bildreihenfolge KEINEN spatialen Bias hat (z.B. zufällig sortierte Drohnenaufnahmen). Funktioniert schlecht, wenn die Bilder zeitlich clustern.

Im 3D-Globus siehst du sofort: grüne Punkte (Training) und rote Punkte (Test). Wenn die roten Punkte alle in einer Ecke clustern, ist der Holdout schlecht (kein guter Generalisierungs-Test). Wenn sie gleichmäßig zwischen den grünen liegen, ist er gut. Der Angular-Correlation-Score pro Fold (rechte Sidebar, in Grad) sagt zusätzlich: kleinerer Wert = der Test ist nah am Training (jede Test-Kamera hat eine nahe Trainings-Kamera, leichter Test); größerer Wert = der Test ist weit weg vom Training (härtere Generalisierung).

Du hast deine Truck-Szene mit 251 Bildern aufgenommen, exportierst per Menü-Item M33 (Export SfM transforms.json) ein nerfstudio-File. Öffne das Holdout-Window (⇧⌘H), lade die JSON via „Open transforms.json…„, schau dir den Globus an. k=5 (Default) gibt dir 5 Folds. Klick auf „Fold 3" — siehe ob die roten Marker einigermaßen gleichmäßig sind. Falls ja: „Export fold-assignment.json„, lege die exportierte Datei in den Reports-Ordner, und beim nächsten Training-Run mit `--benchmark` (oder den

entsprechenden Inspector-Einstellungen) wird genau diese Fold-Aufteilung als Test-Holdout verwendet — anstelle des Standards „every-8th".

W23 Button „Open transforms.json…„

WO

Toolbar oben links..

TECHNISCH

Öffnet einen Datei-Auswahldialog, der auf JSON-Dateien beschränkt ist. Nach Bestätigung lädt das Holdout-Modul die Datei. Der Loader parst sowohl das nerfstudio-Format (Kamera-Intrinsics plus Liste von Frames mit Bildpfad und Transform-Matrix) als auch das instant-ngp-Format (gleicher Aufbau). Pro Frame wird die Blickrichtung aus der Transform-Matrix extrahiert (z-Achse der Kamera-Lokalbasis) und gespeichert. Wenn das Parsen fehlschlägt, wird eine Fehlermeldung im Status-Bereich angezeigt.

Auch via CLI: `--holdout-file /pfad/zu/transforms.json` startet das Fenster direkt mit geladener Datei.

EINFACH GESAGT

Lädt deine Kamera-Posen-JSON. Standard sind Nerfstudio- und Instant-NGP-Exports. Radiance-Kit selbst kann transforms.json via Menü → Export → SfM exportieren.

W24 Picker „Strategy„ (angular/linear)

WO

Linke Sidebar, oben..

TECHNISCH

Radio-Picker mit zwei Optionen: Angular und Linear. Strategy-Wechsel triggert automatisch eine Neuberechnung der Folds. Die Blickrichtungen sind eine Liste von 3D-Einheitsvektoren auf der Sphäre; die Angular-Strategie projiziert sie auf den Longitudinalwinkel ϕ und sortiert, die Linear-Strategie macht einfach eine Modulo-Aufteilung über den Frame-Index.

EINFACH GESAGT

Angular für gleichmäßige Orbit-Aufnahmen (Standard, sicher), Linear nur wenn deine Bilder nicht räumlich clustern.

W25 Slider „k Folds„

WO

Linke Sidebar, mittig..

TECHNISCH

Slider von 3 bis 10, Schrittweite 1. Bei Änderung wird die Fold-Berechnung automatisch neu angestoßen, sodass die Folds-Liste, die Trainings/Test-Indices und der pro-Fold-Score sofort neu berechnet werden. Der gewählte Wert wird als monospaced-Digit-Text rechts neben dem Label angezeigt.

Faustregel: k=5 ist Standard (gibt dir 20% Test pro Fold, das ist üblich für Cross-Validation). k=10 wenn du sehr viel Daten hast und mehr Folds für statistische Aussagekraft brauchst. k=3 wenn du wenig Daten hast.

EINFACH GESAGT

Wie viele Folds in die Aufteilung. 5 ist Standard und passt fast immer.

W26 Picker „Test Fold„

WO

Linke Sidebar, unter dem k-Slider..

TECHNISCH

Menü-Picker. Optionen sind dynamisch 0..<k, Label „Fold 1″ bis „Fold N„ (also 1-indexed in der UI, 0-indexed intern). Wenn der vorher gewählte Index ≥ k ist (z.B. weil du k von 10 auf 5 reduziert hast), wird er automatisch auf 0 zurückgesetzt. Der gewählte Test-Fold wird im Globus rot dargestellt, alle anderen grün.

EINFACH GESAGT

Welcher Fold gerade der Test-Fold ist. Du kannst durchklicken und siehst, wie jeder einzelne Fold im Globus aussieht.

W27 Button „Export fold-assignment.json„

WO

Linke Sidebar, unten..

TECHNISCH

Öffnet einen Speichern-Dialog mit Default- Dateiname `fold_assignment.json`. Nach Bestätigung kodiert das Holdout- Modul die aktuelle Aufteilung in ein JSON-Schema (per-Frame Fold- Zuordnung plus Strategy-Meta-Block). Diese Datei kann dann beim nächsten Training mit `--benchmark` mit übergeben werden, sodass derselbe Holdout für die finale Metrik-Auswertung verwendet wird. Schreib-Fehler werden als Fehlertext angezeigt; Erfolg in grünem Text als „Saved to (filename)„.

EINFACH GESAGT

Speichert die aktuelle Train/Test-Aufteilung als JSON. Diese Datei kannst du dann beim Training direkt mitgeben, sodass dasselbe Test-Set wieder verwendet wird.

W28 SCNView (3D Camera Globe)

WO

Mittelpanel im Holdout-Fenster..

TECHNISCH

SceneKit-Globus-View. Die Szene besteht aus: einer Wireframe-Kugel (Radius 1.0, 36 Segmente, dunkelgrau), drei farbigen Achsenstummeln (rot/grün/blau für X/Y/Z, je 1.2 lang), und pro Kamera einer kleinen Markerkugel (Radius 0.03) an der entsprechenden Blickrichtungs-Position auf der Einheitskugel (leicht außerhalb, damit sie nicht IN der Wireframe-Kugel verschwindet). Die Marker werden bei jeder Fold-Änderung NICHT neu gebaut — Rebuild ist nur dann nötig, wenn sich die Frame-Liste ändert (also eine neue JSON geladen wird). Stattdessen läuft pro Update eine in-place-Aktualisierung der Material- Farben: rot für Test-Indices, grün für Training, hellgrau wenn weder noch. So bleiben Slider-Ticks performant auch bei N > 1000 Kameras.

Die Kamera-Kontrolle ist aktiviert — du kannst mit der Maus den Globus drehen, zoomen, pannen. Beleuchtung sorgt dafür, dass die Marker nicht flach aussehen. Hintergrund ist dunkelgrau.

EINFACH GESAGT

Der 3D-Globus mit den Kamera-Positionen. Grün = Training, Rot = Test, Hellgrau = nicht zugeordnet (kommt nicht vor, alle Kameras gehören irgendwohin). Mit der Maus kannst du den Globus rotieren und zoomen.

W29 FoldCard (Tap to Select Fold)

WO

Rechte Sidebar, „Angular Correlation„-Sektion..

TECHNISCH

Pro Fold eine Karten-View — abgerundetes Rechteck mit 6 pt Radius, Padding 10, vertikales Layout mit zwei Zeilen (oben „Fold N„ + Kamera-Anzahl, unten „Mean nearest angle:“ + Wert in Grad). Hintergrundfarbe konditional: aktiver Fold = Akzentfarbe halbtransparent, inaktive = neutrales Standard-Material. Tippen wählt den Fold, und der Globus färbt sich live um.

Der „Mean nearest angle„-Score ist der mittlere kleinste Winkel pro Test-Kamera zur nächsten Trainings-Kamera (in Radiant intern berechnet, in Grad in der UI angezeigt).

EINFACH GESAGT

Pro Fold eine kleine Karte rechts mit Anzahl der Kameras und dem Durchschnitts-Abstand zur nächsten Trainings-Kamera. Klick darauf wählt diesen Fold als Test.

BayesOpt Console (W30–W39)

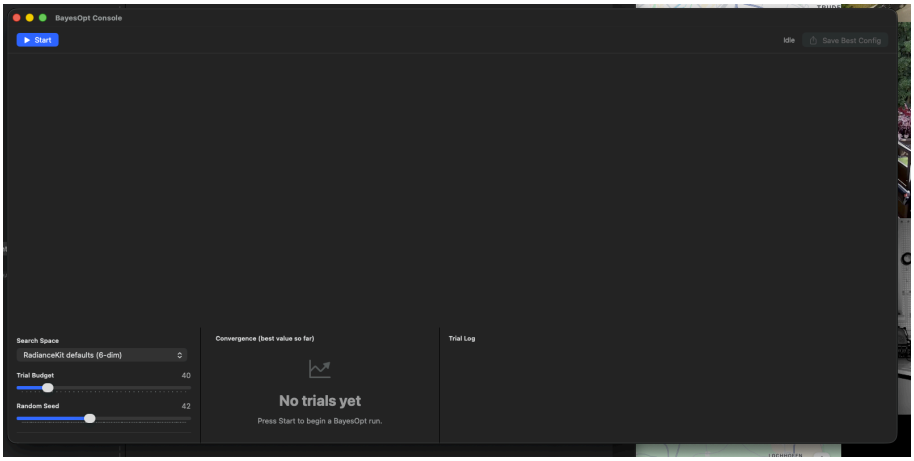

Abbildung 24: BayesOpt-Konsole — leerer Zustand vor Trial-Start

Leerer Zustand mit Search-Space-Picker (`RadianceKit defaults (6-dim)`), Trial-Budget-Slider (default 40), Random-Seed (42) und drei Empty-Panels für Convergence-Chart, Trial Log und Search-Space-Parameter-Liste.

Leerer Zustand (nach erstem Öffnen) — Convergence-Chart und Trial-Tabelle füllen sich, sobald ein Run gestartet wurde, siehe nächster Shot.

Abbildung 25: BayesOpt-Konsole nach 40 Trials — Convergence-Chart steigt steil bis Trial 15, Best Value 0.9943, Trial Log mit init/bo/restart-Tags

WAS IM BILD ZU SEHEN IST Status oben rechts „Finished — best 0.9943 after 40 trials„. Linke Sidebar: Search-Space-Picker auf `RadianceKit defaults (6-dim)`, Trial-Budget 40, Random Seed 42. Parameter-Liste zeigt die sechs zu tunenden Hyperparameter mit ihren Wertebereichen: `mipSmoothing3DScale [0.05, 0.5]`, `mipFilter2DVariance [0.1, 0.6]`, `densifyGradThreshold [5e-07, 5e-06]`, `ssimWeight [0.05, 0.5]`, `mcmcNoiseScale [1e-05, 0.0001]`, `mcmcRelocationInterval [50, 200]`. Mitte: Convergence-Chart (X = Trial-Index 1–40, Y = Objective Value 0–1) — graue Punkte = Initial-Samples (LHS), blaue Punkte = BayesOpt-Acquisition, orange Punkte = Restart-Trials (#22 und #31). Beste-Wert-Linie steigt steil bis Trial ~7, dann nur noch marginale Verbesserung bis Trial 15, ab dort flacher Plateau bei 0.99+. Rechte Sidebar: Trial-Log #1–#34 mit Score + Tag (init/bo/restart). Save-Best-Config-Button oben rechts schreibt `bayesopt-best.json`.

Was es ist: Eine Bayes-Optimierungs-Konsole für Hyperparameter-Suche. Bayes-Opt ist ein automatisches Verfahren, das versucht, mit möglichst wenig Experimenten den optimalen Punkt einer unbekannten Funktion zu finden — typischerweise: „welche Kombination aus mcmcMaxGaussians, capMultiplier, ssimWeight und gradThreshold liefert die beste PSNR für meine Szenenklasse?„ Statt eines Grids von 6^4 = 1296 Trials probiert Bayes-Opt etwa 40–100 informierte Trials und kommt damit nahe ans Optimum.

Wichtig: Die aktuelle in der App ausgelieferte Version führt die Optimierung nicht gegen echte Trainings-Runs aus (das würde Tage dauern), sondern gegen eine synthetische Demo-Objektive — eine multi-modale Landschaft mit Hill-Climbing-Charakter plus leichtem Noise. Das ist mit Absicht so: das Fenster soll dir das Verhalten des Optimierers zeigen (Konvergenz-Verlauf, Sample-Punkte, Best-So-Far) und dich die Search-Space-Definitionen verstehen lassen. Für echte Trainings-getriebene BayesOpt-Läufe (wie in Phase Q7 für die Scene-Class-Presets durchgeführt) wird ein separater offline CLI-Workflow benutzt; das Fenster ist die Live-UI-Variante.

WANN ÖFFNEN Drei Anwendungsfälle: 1. Du willst verstehen, wie BayesOpt arbeitet — dann starte einen Demo-Run und beobachte den Convergence-Chart. 2. Du planst eine neue Szenenklasse (etwa „Aquarien„ oder „Antike Möbel“), für die die eingebauten 10 Presets nicht perfekt passen. Definiere mental einen Suchraum, prüfe ihn hier mit dem „Bowl demo„ oder „Densify“-Preset, exportiere dann die Best-Config als JSON und nutze sie als Startpunkt für einen echten Trainings-Run. 3. Du willst die im RKBayesOpt-

Package definierten Default-Search-Spaces (Mip-Subset, RadianceKit Defaults) inspizieren — die werden im Parameter-Panel der linken Sidebar aufgelistet.

WIE DEUTEN - **Convergence-Chart** (mittlere Spalte): Y = beste bisher erreichte Objective-Funktion-Wert. X = Trial-Index. Anfangs steil ansteigend (BayesOpt probiert die Initial-Samples zufällig, einige davon sind glücklich), dann zunehmend flach, weil die nahen-Optimum-Region ausgeschöpft ist. Wenn die Linie für 20+ Trials flach bleibt, kannst du den Run stoppen — weitere Trials bringen nichts mehr. Die einzelnen Punkte im Chart sind die individuellen Trial-Werte (also nicht „best so far„), gefärbt nach Phase: grau = initial sample, blau = bayesopt acquisition, orange = restart. - **Trial-Tabelle** (rechte Spalte): #1, #2, #3, … mit jeweils Wert und Phase-Tag. Der bisher beste Trial ist mit einem gelben Stern markiert. Aus der Tabelle kannst du den Best-Trial identifizieren und seine Parameter-Werte später beim Export anschauen. - **Search-Space-Inspektor** (linke Sidebar): zeigt für das gewählte Preset alle Parameter-Namen und ihre Suchbereiche `[lo, hi]`. Wenn du beim Preset „RadianceKit defaults (6-dim)“ stehst, siehst du z.B. „densifyGradThreshold [5e-7, 5e-6]„ — also log-uniform zwischen diesen beiden Werten.

Wähle Preset „RadianceKit defaults (6-dim)„, Trial-Budget 40, Seed 42. Klick „Start“. Beobachte: die ersten 8 Trials sind grau (initial samples, LHS-Latin-Hypercube), die folgenden blau (BayesOpt-akquiriert). Der Convergence-Chart wird steil bis Trial ~15, danach flacht er ab. Bei Trial ~30–40 stabilisiert sich der beste Wert. Klick „Save Best Config„ — eine `bayesopt-best.json` wird gespeichert mit dem Preset-Namen, Trial-Index, Wert und den dekodierten Parameter-Werten. Diese JSON kannst du dann manuell in deine Preset-Definition übernehmen.

W30 Button „Start„

WO

Toolbar links, im Idle/Finished-State..

TECHNISCH

Setzt die Trial-Liste zurück, wechselt in den Running-State, generiert eine neue Run-ID (für Stale-Detection bei mehrfachen Start-Clicks) und erzeugt ein frisches Pause-Gate. Dann startet ein Hintergrund-Task, der den Optimierer als asynchronen Stream ausführt. Initial-Samples-Größe ergibt sich aus `min(8, budget / 4 + 1)` — also typisch 8 Latin-Hypercube-Samples bei Budget ≥ 28, weniger bei kleinem Budget. Trial-Updates werden inkrementell empfangen und in die Liste angehängt. Stale-Run-Protection: wenn währenddessen ein zweiter Start-Klick die Run-ID neu setzt, werden Updates aus dem alten Run verworfen.

Primary-Action-Style für den prominenten Knopf-Look.

EINFACH GESAGT

Startet einen frischen Optimierungs-Lauf mit dem aktuellen Suchraum, Budget und Seed.

W31 Button „Pause„

WO

Toolbar links, im Running-State..

TECHNISCH

Setzt das Pause-Gate aktiv und wechselt in den Paused-State. Der eigentliche Effekt: der Runner wartet in einem 50-ms-Polling-Loop bevor er die nächste Objective-Funktion auswertet. Das bedeutet, ein gerade laufendes Trial wird zu Ende geführt (es ist ja synthetisch und dauert nur Mikrosekunden), aber kein weiteres Trial wird angestoßen. Sobald Resume läuft, geht es weiter wo aufgehört wurde.

EINFACH GESAGT

Hält den Lauf an. Aktuelle Berechnung läuft noch zu Ende, dann pausiert es.

W32 Button „Stop„

WO

Toolbar links, im Running- und Paused-State..

TECHNISCH

Bricht den Runner-Task ab, nullt die Referenz, löst das Pause-Gate (falls noch paused war), und wechselt in den Finished-State (wenn Trials existieren) oder Idle-State (wenn keine). Die bereits berechneten Trials bleiben in der Liste sichtbar — Stop löscht sie nicht. Destruktive Button-Rolle zeigt den Knopf in Rot, weil er den Run abbricht.

EINFACH GESAGT

Bricht den Lauf endgültig ab. Trials bleiben sichtbar, du kannst die Best-Config trotzdem exportieren.

W33 Button „Resume„

WO

Toolbar links, im Paused-State..

TECHNISCH

Löst das Pause-Gate und wechselt in den Running-State zurück. Der Runner-Task läuft schon (er wartet ja im Polling-Loop); sobald der Loop merkt, dass die Pause aufgehoben ist, läuft er weiter und startet das nächste Trial.

EINFACH GESAGT

Setzt einen pausierten Lauf fort.

W34 Button „Save Best Config„

WO

Toolbar rechts, immer sichtbar (aber disabled wenn kein bestTrial vorhanden)..

TECHNISCH

Öffnet einen Speichern-Dialog mit Default- Dateiname `bayesopt-best.json`, auf JSON beschränkt. Nach Bestätigung wird ein Payload-Dictionary gebaut: Preset-Name, Trial-Index, Wert (Objective-Score), Parameter (Dictionary der dekodierten Parameter- Namen → Werte). Die Decodierung projiziert die normalisierten Suchraum-Koordinaten in [0,1]^d zurück in den Original-Wertebereich (mit log-uniform/linear/integer-Skalen entsprechend). JSON-Output ist pretty-printed und mit sortierten Keys. Bei Schreib-Fehler wird (in der aktuellen Demo-Version) still ignoriert — keine Error-UI, weil das ein Demo-Path ist.

Der Button bleibt grau, solange kein Trial gelaufen ist.

EINFACH GESAGT

Speichert die Parameter-Werte des bisher besten Trials als JSON. Du kannst diese Werte dann manuell in deine Preset-Konfiguration übernehmen.

W35 Picker „Search Space„-Preset

WO

Linke Sidebar, oben..

TECHNISCH

Menü-Picker mit vier Preset-Optionen: - „RadianceKit defaults (6-dim)„ — der vollständige Standard-Suchraum mit allen Q7-Hyperparametern. - „Mip subset (2-dim)“ — nur `mipSmoothing3DScale` [0.05, 0.5] log-uniform und `mipFilter2DVariance` [0.1, 0.6] linear. Nützlich wenn du Mip-Splatting für eine Szenenklasse tunen willst. - „densify-until + ssim-weight + grad-thresh„ — drei Densify-relevante Parameter (`densifyGradThreshold` log-uniform, `ssimWeight` linear, `densifyUntilIter` integer). - „Bowl demo (1-dim)“ — pädagogischer Single-Parameter-Suchraum für „so funktioniert BayesOpt„-Demos.

Während ein Lauf aktiv ist, kann der Suchraum nicht gewechselt werden (würde den Optimizer verwirren).

EINFACH GESAGT

Welchen Hyperparameter-Suchraum BayesOpt durchsucht. Standard ist „RadianceKit defaults„. Für gezielte Mip-Tuning-Versuche „Mip subset“. Zum Verstehen wie BayesOpt arbeitet „Bowl demo„.

W36 Slider „Trial Budget„

WO

Linke Sidebar, unter dem Search-Space-Picker..

TECHNISCH

Slider von 10 bis 200, Schrittweite 5. Default 40. Das bedeutet: BayesOpt darf maximal N Trials machen. Davon sind die ersten paar initiale Samples (Latin-Hypercube), der Rest sind echte BayesOpt-Trials. Faustregeln für die Praxis: ein Suchraum mit d Dimensionen braucht etwa 10*d bis 20*d Trials für ein gutes Optimum. Bei 6-dim Defaults also 60–120, bei 2-dim Mip-Subset 20–40, bei 1-dim Bowl-Demo 10–20.

Während des Laufs ist der Slider deaktiviert.

EINFACH GESAGT

Wie viele Optimierungs-Versuche maximal. Mehr Versuche = bessere Lösung, aber kostet mehr Zeit. 40 ist guter Default für die Demo-Objective.

W37 Slider „Random Seed„

WO

Linke Sidebar, unter dem Budget-Slider..

TECHNISCH

Slider von 1 bis 100, Schrittweite 1. Default 42. Der Seed wird sowohl an die initialen Latin-Hypercube-Samples als auch an die Noise-Komponente der Demo-Objective weitergereicht. Reproduzierbarkeit: gleicher Seed + gleicher Suchraum + gleiches Budget ergibt exakt identische Trial-Sequenz. Nützlich für „bekommen alle deine Kollegen denselben Lauf wenn sie das Demo nachbauen?„. Während des Laufs deaktiviert.

EINFACH GESAGT

Steuert den Zufallsgenerator. Selber Seed = selber Lauf — zum Reproduzieren.

W38 Chart (Convergence)

WO

Mittlere Spalte des Fensters..

TECHNISCH

Swift-Charts-Diagramm mit zwei Layern: 1. eine Linie für „best-value-so-far„ pro Trial — eine monoton steigende oder gleichbleibende Kurve in Akzentfarbe. 2. ein Punkt pro Trial mit dem individuellen Objective-Wert, gefärbt nach Phase. Symbol-Größe 40. Drei Phase-Labels: „init“ (grau), „bo„ (blau), „restart“ (orange).

Eine kleine Legende zeigt die Phase-Farben oben links. Wenn die Trial- Liste leer ist (vor dem ersten Start), wird stattdessen eine Empty- State-Anzeige mit Chart-Icon und Hinweis „Press Start to begin a BayesOpt run.„ eingeblendet.

EINFACH GESAGT

Der Verlaufs-Chart. Die durchgezogene Linie ist „beste bisher gefundene Lösung„; die Punkte sind die einzelnen Versuche. Wenn die Linie für lange Zeit flach bleibt, hat BayesOpt das Optimum gefunden.

W39 Table (Trial Log)

WO

Rechte Spalte des Fensters..

TECHNISCH

Scroll-Bereich mit lazy gestapelten Trial- Zeilen. Pro Zeile ein horizontaler Stack: Trial-Nummer (3-stellig monospaced, links), Wert (monospaced, rechtsbündig, 70 pt breit), Phase- Tag (Capsule, gefüllt mit Phase-Farbe bei 25% Opacity), optional ein gelber Stern wenn dieser Trial der aktuell beste ist. Ein Auto-Scroll-Mechanismus springt automatisch ans Ende, sobald ein neuer Trial dazukommt — sodass du den Live-Verlauf am Bildschirm-Boden mitlesen kannst, ohne selbst zu scrollen.

EINFACH GESAGT

Die Tabelle aller Versuche. Wert, Phase, Stern für den besten. Scrollt automatisch mit, neue Trials erscheinen unten.

Hauptfenster: Verlustverlauf und Gaussian-Count (I39–I41, Querverweis)

Drei der Inspector-Anzeigen im Hauptfenster verdienen eine eigene Erklärung, weil sie während eines laufenden Trainings ständig zu sehen sind und es wichtige Faustregeln gibt, wann der Verlauf gesund aussieht. Die Anzeigen sind im Inspector unter der „Loss Chart„-Sektion (siehe Kapitel 2 — Inspector) und ergänzen die Holdout-Analyse aus dem Aux-Fenster oben.

Wann ist die Loss-Kurve gesund? Eine gesunde Loss-Kurve zeigt drei Phasen: (1) **Warmup** — die ersten 200–500 Iterationen fällt der Loss steil von hoch (typisch 0.15–0.25 für L1+SSIM-kombiniert je nach Szene) auf etwa die Hälfte. Wenn der Loss in dieser Phase NICHT fällt, ist meist die Eingabe falsch (Bilder kaputt, SfM-Posen schlecht, Anzahl Initial-Gaussians zu klein). (2) **Densification** — zwischen ~500 und densifyUntilIteration (klassisch 15K, MCMC bis 20K oder 25K) fällt der Loss weiter, oft mit kleinen Sprüngen nach unten wenn Densify-Operationen neue Gaussians einfügen und der Optimizer sie ausnutzt. Der Gaussian-Count steigt in dieser Phase. (3) **Refinement** — danach läuft der Loss in einen flacher werdenden Tail. Typische End-Werte: Tanks-&-Temples Truck mit P4 Quality landet bei L1 ≈ 0.023, Horse mit Full Classic V546 bei L1 ≈ 0.0230, Outdoor-Mip-NeRF360-Szenen oft schlechter (0.04–0.07).

Was bedeutet ein Plateau? Ein Plateau (Loss-Kurve verläuft horizontal über mehrere Tausend Iterationen) hat zwei Interpretationen: (a) das Modell hat konvergiert, weiteres Training bringt nichts mehr — das ist der gute Fall. (b) das Modell ist stuck (lokales Minimum, schlechte Gradient-Information, ein Cap am Buffer-Limit) — der schlechte Fall. Beide sehen im Chart identisch aus. Unterscheidung: schau dir den Gaussian-Count an. Wenn er auch flach ist UND dem MCMC-Cap nahe (z.B. 150K von 150K bei `.fullMCMC`), bist du am Limit — entweder Cap erhöhen oder Plateau akzeptieren. Wenn der Gaussian-Count noch wächst, aber der Loss nicht fällt, ist das stuck.

Wann abbrechen vs weitertrainieren? Faustregel: 10K Iterationen lang keine Verbesserung des Min-Loss → abbrechen, weitere Iterationen sind verschwendet. Davor: kannst du via Cmd+T (Training-Menü → Continue Training → +5K iterations) noch eine Verlängerung anhängen, falls du grenzwertige Verbesserung siehst. Achtung: bei MCMC ist das Plateau oft echt — das Cap ist die natürliche Grenze.

Gaussian-Count-Plateau ist KEIN „fertig„-Signal. Es bedeutet nur, dass MCMC das Cap erreicht hat oder dass Classic Densification ausgereizt ist. Die echte „fertig"-Frage stellt erst die Holdout-Analyse — PSNR/SSIM/LPIPS auf einem unabhängigen Test-Set, ausgewertet im Holdout-Window (W23–W29) oder via `--benchmark`-Flag.

PSNR/Holdout ist die Wahrheit, Loss nur Proxy. Der Loss ist eine relative Metrik: er fällt während dein Modell sich an die Trainings-Views anpasst. Ein niedriger Loss heißt aber nicht automatisch gutes Modell — wenn das Modell die Trainings-Bilder auswendig gelernt hat (Overfitting), wäre der Loss klein, aber die PSNR auf ungesehenen Views (Holdout) wäre schlecht. Deshalb: für die finale Qualitätsbewertung immer auf Holdout-Metriken schauen, nicht auf End-Loss allein.

Faustregel-Box

- User Guide und Keyboard Shortcuts sind statische Hilfe — bei Stichwort-Fragen schnell, für Tiefe das hier vorliegende Manual nutzen.
- Manage Storage öffnen, sobald die Platte unter 10% freien Speicher fällt. Logs und Imports-Staging sind die üblichen Sünder.
- Pareto Dashboard erst nach mindestens drei oder vier Trainings-Reports sinnvoll. X-Achse = Kosten (Time / Gs), Y-Achse = Qualität (PSNR / SSIM). Pareto-Front zeigt die effizienten Kombinationen.
- Holdout Analysis nutzen, bevor du PSNR-Benchmarks mit anderen veröffentlichst — sichert dir, dass dein Test-Set wirklich repräsentativ ist.
- BayesOpt Console ist primär ein Lern- und Inspektions-Werkzeug für Suchraum-Definitionen. Für echte Trainings-getriebene Hyperparameter-Tuning den offline CLI-Workflow nutzen.
- Loss-Plateau und Gaussian-Count-Plateau sind getrennt zu interpretieren. Cap-Limit ist kein „fertig„-Signal. Echte Qualität misst nur Holdout-PSNR.
- 10K Iterationen ohne Min-Loss-Verbesserung → Training stoppen.

KAPITEL

Kapitel 6 — Trainings-Konfiguration

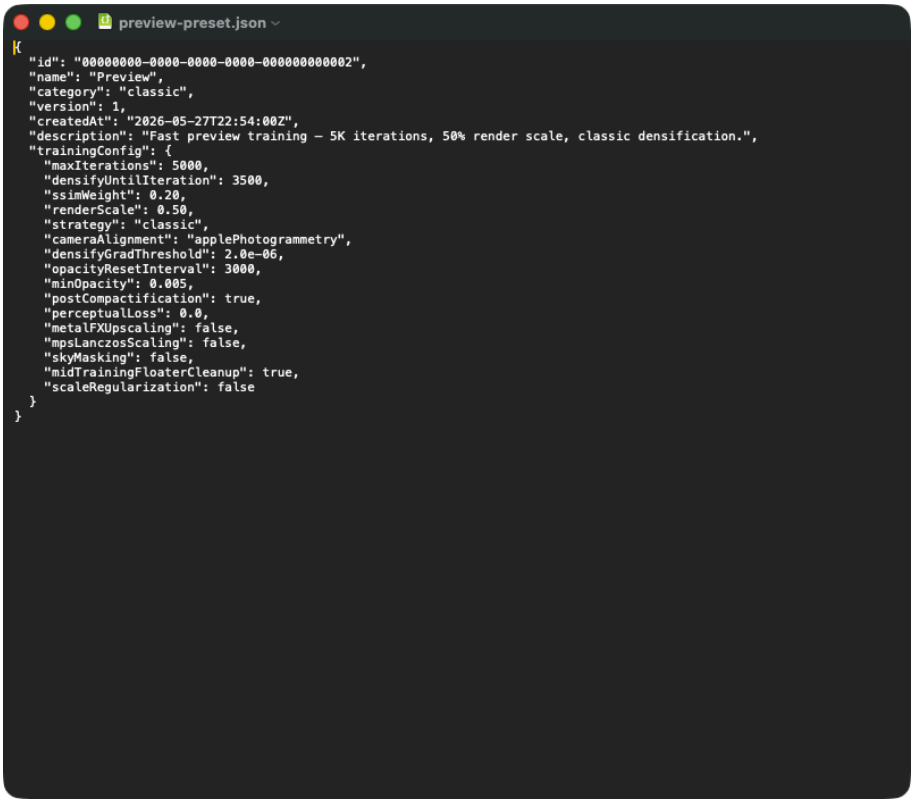

Abbildung 26: Preview-Preset als JSON exportiert + in TextEdit angezeigt — Felder id/name/category/version/createdAt/description, trainingConfig mit allen relevanten Parametern (maxIterations 5000, densifyUntillIteration 3500, ssimWeight 0.20, renderScale 0.50, strategy classic, cameraAlignment applePhotogrammetry, densifyGradThreshold 2.0e-06, opacityResetInterval 3000, minOpacity 0.005, sechs Bool-Toggles)

WAS IM BILD ZU SEHEN IST Ein typischer Preset-JSON-Export. Top-Level-Felder: `id` (UUID), `name`, (classic | mcmc | sceneClass | custom), (Schema-Version), (Timestamp), (Free-Text). Verschachteltes -Objekt enthält die für Reproduzierbarkeit kritischen Parameter — beim Import wird der gesamte Block in die `TrainingConfig` -Struktur deserialisiert, und Defaults aus der App-Version füllen die Felder, die in der JSON fehlen (z. B. nach App-Update). Wer ein Preset an einen anderen Mac übergibt, schickt einfach diese JSON-Datei rüber.

Die `TrainingConfig` -Struktur ist das Herzstück jedes Trainingslaufs in RadianceKit. Sie versammelt jeden Parameter, der das Training beeinflusst — von der maximalen Iterationszahl über die acht Lernraten bis hin zu den Spezialfeldern für MCMC, Mip-Splatting, das Curriculum und die scene-aware Cap-Logik. Du bearbeitest sie in der Sidebar im Bereich Trainings-Konfigurations-Sektion (Expert View), speicherst sie als Preset oder

reichst sie als JSON-Export an einen anderen Mac weiter. Beim Training wird genau dieses Objekt eingefroren und ans GPU-Backend gegeben.

Dieses Kapitel ist Referenz-Material für Power-User und Skript-Autoren. Es listet alle 81 öffentlichen Felder, die 9 statischen Presets und die eine öffentliche Methode. Quelldatei ist `TrainingConfig.swift` — bei Zweifeln gilt der dort hinterlegte Doc-Comment und der Initializer-Default als Source-of-Truth.

> **HINWEIS · UI VS. PRESET/CLI**
>
> Nur 12 der 81 Felder haben einen direkten Schieberegler, Toggle oder Picker im Inspector (sandboxed App-Store-Build): **T1, T2, T17, T20, T22, T38, T56–T58, T60, T61, T73**. Die restlichen 69 Felder werden über das gewählte **Preset** (Kapitel 7) gesetzt und lassen sich nur per **CLI-Flag** (siehe Kapitel 5) direkt überschreiben. Diese Trennung ist Absicht: Defaults bleiben stabil und produktionserprobt, Power-User haben dennoch eine Escape-Luke. Wenn ein Feld dich besonders interessiert: schau zuerst in Kapitel 2 (Inspector) und Kapitel 5 (CLI) nach, ob du es ohne JSON-Bastelei erreichst.

Inhaltsverzeichnis:

1. Iteration (T1–T2)
2. Learning Rates (T3–T10)
3. Densification — Classic (T11–T16)
4. Loss (T17–T20)
5. SH-Degree-Progression (T21)
6. Performance (T22–T25)
7. Diagnose und Punktwolken-Vorbereitung (T26–T30)
8. Regularisierung (T31–T37)
9. Refinement (T38–T44)
10. Sky-Dome (T45–T48)
11. Adam + LR-Schedule (T49–T55)
12. Post-Processing + Apple AI (T56–T60)
13. MCMC-Densification (T61–T73)
14. Mip-Splatting (Q1.5) (T74–T76)
15. Adaptive Densification (Q5) (T77–T79)
16. Curriculum (Q6) (T80–T81)
17. Statische Presets (TP1–TP9)
18. Methode:
19. Welches Feld wofür? (Cheat-Sheet)
20. Gefährliche Felder

Iteration (T1–T2)

T1 maxIterations

DETAILS

Default: 30 000 (Initializer), 35 000 (`.full`), 200 000 (`.fullMCMC`) **Range:** 1 000 – 500 000 (UI-Slider), keine harte obere Grenze in der Logik **Defined in:**

TECHNISCH

Gesamtzahl der Trainings-Iterationen, die das Backend durchläuft. Eine Iteration bezeichnet ein Forward-Render einer einzelnen Trainingskamera, einen Backward-Pass über alle Loss-Komponenten (L1 + SSIM + optionale Regularisierungen + Sky-Mask) und einen Adam-Optimizer-Schritt. Diese Zahl wirkt direkt auf die anderen Schedules: Position-Lernrate folgt einer Cosine-Annealing-Kurve von 0 bis entweder `T1` selbst oder bis `T49 positionLRScheduleEndIteration`; Densification stoppt bei `T2 densifyUntilIteration`; MCMC-Noise-Decay endet bei `T69 mcmcNoiseDecayEnd`; SH-Degree-Upgrades passieren an den drei in `T21` definierten Marken. Bei klassischer Densification liegt der empirisch ermittelte sweet spot bei 20 000–35 000 Iterationen (Sessions 1–32, V546-Tests), bei MCMC bei 60 000–200 000 (V534). Eine drastische Erhöhung über die in Preset hinterlegten Werte hinaus bringt selten zusätzliche Qualität — Adam-Momentum sättigt, und ohne LR-Decay-Ende stagniert der Loss. Umgekehrt führt Unterschreitung von ~5 000 zu unvollständig konvergierten Geometrien (Density-Control hat zu wenig Zeit zum Klonen/Splitten).

EINFACH GESAGT

Wie lange die App rechnet. Mehr Iterationen = besseres Ergebnis, aber irgendwann nicht mehr spürbar besser, dafür viel länger. Die Presets sind so gewählt, dass du ohne nachzudenken einen guten Wert hast: Quick 1 000, Preview 5 000, Balanced 20 000, Quality 35 000, MCMC Quality 200 000. Wenn du selbst dran drehst, gilt: bei MCMC darfst du gerne hoch (100 000–200 000), bei Classic nicht über 40 000 — bringt dann nichts mehr.

T2 densifyUntilIteration

DETAILS

Default: 15 000 (Initializer), 5 000 (`.full`), 160 000 (`.fullMCMC`) **Range:** 0 – **Defined in:**

TECHNISCH

Iteration, ab der die Densification aufhört. Bis hierhin werden Gaussians über die in `T11–T16` (Classic) oder `T67–T70` (MCMC) parametrisierten Regeln geklont, gesplittet und geprunt; danach bleibt die Gaussian-Anzahl konstant und nur noch Positionen, Rotationen, Skalen, Opazitäten und SH-Koeffizienten werden optimiert (Refinement-Phase). Im 3DGS-Originalpaper liegt der Wert bei 50 % von `T1` , in RadianceKits `.full` -Preset bei nur ~14 % (5 000 von 35 000) — Folge der V310/V338-Experimente, die zeigten, dass nach 5 000 Iterationen weitere Densifizierung das Resultat eher verschlechtert (mehr Floaters, mehr Speicherbedarf, kein Qualitätsgewinn). MCMC dagegen läuft die Relocation bis 80 % von `T1` (V504b), weil MCMC keine schädlichen Floaters produziert. Wird `T2` zu klein gewählt (< 1 000), entstehen zu wenige Gaussians; zu groß bei Classic (> 50 % von `T1`) führt zu Overgrowth und RGB-Saturation-Outliers (siehe Outdoor-Overtraining-Findings).

EINFACH GESAGT

Bis wann die App neue Gaussians erzeugen darf. Danach wird nur noch verfeinert, was schon da ist. Bei klassischem Training mit 35 000 Iterationen ist hier 5 000 der richtige Wert — alles darüber macht die Szene matschiger. Bei MCMC ist es 80 % der Gesamtiterationen (also 160 000 bei 200 000-Lauf). Wenn du Quality-Preset änderst, lass dieses Feld lieber in Ruhe.

Learning Rates (T3–T10)

T3 positionLearningRate

DETAILS

Default: 0.00016 **Range:** 1e-7 – 1e-3 (empfohlen)
Defined in:

TECHNISCH

Adam-Lernrate für die XYZ-Position jeder Gaussian zu Beginn des Trainings (Iteration 0). Folgt einer Cosine-Annealing-Kurve und sinkt im Laufe des Trainings auf `T4 positionLearningRateFinal`. Der Default 0.00016 stammt aus dem 3DGS-Originalpaper (Kerbl et al. 2023) und ist in RadianceKit auch bei Erhöhung der Bildauflösung nicht zu skalieren — die Position bewegt sich im Welt-Koordinatensystem, nicht im Pixelraum. Eine deutliche Erhöhung (> 0.0005) bewirkt, dass Gaussians über lange Distanzen springen und der Loss instabil wird; Werte deutlich darunter (< 0.00005) führen dazu, dass falsch initialisierte Punktwolken nie ihren Platz finden. V414 testete eine Verdopplung des Init-Werts → 16.8 % schlechterer L1-Loss; die V544a-Tunings bestätigten den Paper-Default als optimal. Beachte: bei `.fullMCMC` lassen wir diesen Wert bewusst beim Default — MCMC braucht konstante Lernraten für seine Relocation-Logik, daher bringt das Tunen hier nichts.

EINFACH GESAGT

Wie schnell sich die Splatt-Punkte im Raum bewegen dürfen. Der Standardwert ist sehr gut justiert und braucht eigentlich keine Änderung. Nur wenn dir Splats im Bild „herumeiern„ oder eine ganze Ecke fehlt, weil sich nichts hinbewegt, wäre die Lernrate ein Punkt zum Drehen — aber dann passt typischerweise vorher schon etwas anderes nicht (Kameraposen, Initialpunktwolke).

T4 positionLearningRateFinal

DETAILS

Default: 0.0000016 (Initializer + Paper), 0.000016 (`.full` , `.fullMCMC` — 10× höher) **Range:** 0 – **Defined in:**

TECHNISCH

Endwert der Position-LR-Cosine-Annealing-Kurve. Erreicht wird er entweder bei `T1 maxIterations` oder, falls gesetzt, bei `T49 positionLRScheduleEndIteration`. Der RadianceKit-`.full`-Preset verwendet 0.000016 — also 10× höher als der Paper-Default 0.0000016. V420-Experimente zeigten, dass 0.5× des Final-Werts (0.000008) den Loss um 6.4 % verschlechtert; V414 zeigte, dass 2× Init-Wert ihn um 16.8 % verschlechtert. Der hohe Final-Wert ist nicht Trade-off, sondern bewusste Wahl: bei zu starkem Decay verlieren die Gaussians während der Refinement-Phase ihre Fähigkeit, sich auf neu hinzugekommene Densification-Kandidaten einzustellen. Über die V431/V433-Erweiterung kann die Schedule-Phase verkürzt werden (`T49 < T1`), sodass `T4` bereits vor Trainingsende erreicht wird und der Rest des Trainings bei konstanter Mini-LR läuft — typische Konfiguration: `T49 = 20 000` , `T1 = 35 000` , Refinement also bei 0.000016 für 15 000 Iterationen.

EINFACH GESAGT

Wie langsam die Position-Lernrate am Ende des Trainings wird. Wir haben das bewusst weniger aggressiv eingestellt als das Originalpaper — Splats können bis zum Schluss noch ein bisschen wackeln, das macht sie schärfer. Wenn du daran drehst: höher = unruhigere Splats am Ende, niedriger = Splats können sich nicht mehr anpassen, wenn neue auftauchen.

T5 shDCLearningRate

DETAILS

Default: 0.0025 (Initializer + Paper), 0.005 (`.full` und alle MCMC-Presets — 2×) **Range:** 0.0001 – 0.05 **Defined in:**

TECHNISCH

Adam-Lernrate für den DC-Anteil (degree 0, also konstantes Albedo) der spherical-harmonic-Farbe. SH-DC entspricht dem direktionsunabhängigen Grundton einer Gaussian, gewissermaßen die „Basisfarbe„. Die V176- und V188-Experimente fanden 2× höher als der Paper-Default optimal — schnellere Farb-Konvergenz, gerade weil bei kurzem Training (, 5 000 Iterationen) die SH-DC sonst nicht in Form kommt. Anders als die geometrischen LRs hat SH-DC keinen Decay; die Lernrate bleibt über alle Iterationen konstant (oder folgt nur dem optionalen extended-phase-Decay aus `T51`). V416 testete eine Vervierfachung auf 0.01 → 6.4 % schlechterer Loss bei beta2=0.99-Adam.

EINFACH GESAGT

Wie schnell sich die Grundfarbe jedes Splats anpasst. Den Wert ändert man fast nie selbst — die Presets haben den richtigen Wert. Höher ginge schneller, kann aber zu instabilen Farben führen.

T6 shRestLearningRate

DETAILS

Default: 0.000125 (Initializer + Paper), 0.00025 (`.full` und MCMC — 2×) **Range:** 0.000001 – 0.005 **Defined in:**

TECHNISCH

Adam-Lernrate für die SH-Koeffizienten höherer Ordnung (Degree 1, 2, 3 — also die view-direction-abhängigen Farbanteile, die für Glanzlichter, Spiegelungen und sanfte Schattierung sorgen). 20× kleiner als `T5` per Paper-Konvention, weil diese Koeffizienten quadratisch in Anzahl wachsen (3 für Degree 1, 5 für Degree 2, 7 für Degree 3 → insgesamt 15 Floats pro Gaussian) und ohne kleinere Lernrate das Bild übersättigen würden. Wird in zwei Schritten freigeschaltet — bis zur ersten Marke in `T21 shDegreeUpgradeIterations` ist nur Degree 0 aktiv (also nur `T5`), danach 1, später 2, schließlich 3. Niedrige Werte hier sind besonders wichtig auf Szenen mit viel diffuser Beleuchtung; bei sehr glänzenden Oberflächen (Auto-Lack, Wasser) lohnt sich kein Drehen — die SH-Repräsentation an sich ist begrenzt.

EINFACH GESAGT

Wie schnell die richtungsabhängigen Farb-Effekte (Spiegelungen, Glanz) lernen. Standardmäßig sehr klein, weil sonst alles glänzt. Den Wert lässt man besser stehen — wer Glanzlichter besser hinbekommen will, ist eher bei MCMC und längerer Trainingszeit besser bedient als bei dieser LR.

T7 opacityLearningRate

DETAILS

Default: 0.05 (Initializer + Paper), 0.1 (`.full` , MCMC — 2×) **Range:** 0.001 – 1.0 **Defined in:**

TECHNISCH

Adam-Lernrate für die logit-Opazität jeder Gaussian. Die App speichert Opazität als unbeschränkten Float-Wert und transformiert ihn mit Sigmoid in [0, 1]; die LR wirkt im Logit-Space. Der Paper-Default 0.05 ist nach V50-Tests (Best Single-Run L1 0.1664) wiederhergestellt, V71 revertete V67s 0.025. Die V188-Verdopplung auf 0.1 macht das Pruning effizienter — tote Gaussians fallen schneller unter den `T14 pruneOpacityThreshold` ab. V418 zeigte: 0.05 mit beta2=0.99-Adam ist 7.1 % schlechter als 0.1 — die Wechselwirkung mit der Adam-Konfiguration ist nicht trivial. Niedrige Werte (< 0.01) führen dazu, dass „dead„ Gaussians ewig herumliegen und Speicher verbrauchen; zu hohe Werte (> 0.5) können zu Opacity-Explosion führen, daher wird der Logit-Wert im Optimizer auf [−15, 3] geclampt (siehe Notiz „Opacity Explosion Prevention“ in CLAUDE.md).

EINFACH GESAGT

Wie schnell Splats durchsichtig oder undurchsichtig werden. Wichtig fürs Aufräumen — Splats, die nichts beitragen, müssen schnell verschwinden, damit kein Schleier entsteht. Standardwert passt, nur Profis ändern ihn.

T8 opacityLearningRateFinal

DETAILS

Default: 0.0 (= „kein Decay„) **Range:** 0 oder 0.001 – **Defined in:**

TECHNISCH

Optionaler Cosine-Decay-Endwert für die Opacity-LR (V427). Wenn 0.0, ist Decay deaktiviert und die Opacity-LR bleibt über das gesamte Training konstant bei `T7`. V427 testete einen Decay 0.1 → 0.01 — Ergebnis 11.5 % schlechterer Loss; reverted, daher der Default „aus„. Die Hypothese hinter dem Feld: in der Refinement-Phase könnte konstante Opacity-LR zu Oszillation führen, sodass Splats, die schon das richtige Maß an Transparenz erreicht haben, durch zufällige Gradient-Schwankungen wieder verschoben werden. Empirisch bestätigt sich das nicht — die Logit-Clamping-Logik fängt das ohnehin ab. Das Feld bleibt verfügbar für zukünftige Experimente; auch sehr lange MCMC-Läufe (> 500K Iterationen) könnten davon profitieren.

EINFACH GESAGT

Ob die Opacity-Lernrate gegen Ende kleiner werden soll. Standard: nein. Wir haben es probiert, war schlechter, lassen es deaktiviert. Bleib bei 0.

T9 scaleLearningRate

DETAILS

Default: 0.005 (Initializer + Paper), 0.01 (`.full`, MCMC — 2×) **Range:** 0.0001 – 0.1 **Defined in:**

TECHNISCH

Adam-Lernrate für die drei Skalen-Komponenten jeder Gaussian im log-Space (RadianceKit speichert log(scale), damit Skalen positiv bleiben). Der Paper-Default 0.005, in RadianceKit verdoppelt auf 0.01 für besseren Scale-Konvergenz bei den optimierten Lernraten-Konfigurationen. V423-Experiment: 0.005 mit beta2=0.99-Adam → 18.7 % schlechterer Loss und sichtbar zu wenige Gaussians (Density-Control konnte nicht klonen, weil die Skalen-Updates zu lahm waren). Skala kontrolliert die Ausdehnung jeder Gaussian — zu schnelles Lernen führt zu „needle„-Gaussians (extrem lange dünne Splats, siehe `T34 scaleRatioPruneThreshold`), zu langsames Lernen lässt Splats zu kompakt bleiben und Density-Control muss zu oft splitten.

EINFACH GESAGT

Wie schnell die Form der Splats sich anpasst. Standard ist gut. Wenn du das hochdrehst, bekommst du gerne „Nadel„-Splats — extreme lange dünne Tropfen, die das Bild floaten lassen.

T10 rotationLearningRate

DETAILS

Default: 0.001 (Initializer + Paper), 0.002 (`.full` , MCMC — 2×) **Range:** 0.0001 – 0.05 **Defined in:**

TECHNISCH

Adam-Lernrate für die vier Quaternion-Komponenten jeder Gaussian. Die Quaternion wird in jedem Optimizer-Schritt nach der Adam-Update wieder normalisiert (L2-Norm = 1) — andernfalls würde die Kovarianzmatrix entartet. RadianceKit verdoppelt den Paper-Default in den Quality-Presets, weil Rotation gegenüber Skala / Position kleinere absolute Gradient-Magnituden hat (auf der Einheitssphäre bleibt jeder Schritt kurz) und ohne 2× wäre die Rotation im 35 000-Iterations-Window deutlich unterkonvergiert. V188 dokumentiert. Auf NeRF-Blender-Szenen (Lego, Chair) wirkt sich Rotation besonders aus — die Kanten der Objekte richten sich erst nach 5 000–10 000 Iterationen richtig aus.

EINFACH GESAGT

Wie schnell sich die Splats drehen lernen — also auf der Oberfläche eines Objekts in die richtige Ausrichtung kommen. Standard passt. Anders ausgedrückt: wenn Splats wie schief liegende Klötze aussehen statt sich an die Oberfläche anzuschmiegen, ist eher die Trainingszeit zu kurz, nicht diese Lernrate zu niedrig.

Densification — Classic (T11–T16)

T11 densifyGradThreshold

DETAILS

Default: 0.000002 (Initializer, kalibriert für 0.5× Auflösung), 0.0000011 (`.full` , kalibriert für 1.0×), 0.000004 (`.quickTest` , kalibriert für 0.25×), 2e-7 (`.fullClassicPaper`) **Range:** 1e-8 – 1e-3 (auflösungsabhängig) **Defined in:**

TECHNISCH

Schwellwert für die L2-Norm des bildschirmraum-projizierten Gradienten `dMean2D` , oberhalb dessen eine Gaussian für Klonen oder Splitten markiert wird. Der absolute Wert hängt direkt von der Trainingsauflösung ab — `dMean2D` skaliert ungefähr wie 1/Auflösung² (mehr Pixel = kleinere Per-Pixel-Gradienten). Daher braucht jede `T22 trainingRenderScale`-Stufe einen kalibrierten Schwellwert: 0.25× → 4e-6, 0.5× → 2e-6, 1.0× → 5e-8 … 1.1e-6 (`.full`). Der Paper-Default 0.0002 ist NDC-normalisiert und in RadianceKits Welt-Raum-Pipeline nicht direkt vergleichbar. Mit dem in V440 zugeschalteten `T52 adaptiveDensifyThreshold`-Flag lässt sich der Wert in Laufzeit aus dem p98 der aktuellen Gradient-Verteilung berechnen — aber V440 testete das auf realen Szenen und produzierte 63 K Gaussians (katastrophaler Pruning-Verlust); das Flag bleibt aus. Q5 (T77–T79) liefert eine alternative Adaptive-Logik via rolling median. **Gefahrlos ist dieses Feld nicht** — Halbierung erzeugt 2–4× mehr Gaussians (Speicher-Druck, OOM-Risiko); Verdopplung kann die Szene unter-densifizieren.

EINFACH GESAGT

Wie empfindlich die App ist, wenn sie entscheiden soll, ob ein Splat zu wenig dargestellt wird und vervielfältigt werden muss. Niedriger Wert = empfindlicher = mehr Splats. Höher = weniger Splats. Das ist einer der gefährlichsten Werte überhaupt: zu niedrig und der Mac läuft mit Millionen Splats voll Speicher und stürzt eventuell ab. Lass das Feld in Ruhe, oder ändere es nur in Schritten von 10 %.

T12 densifyFromIteration

DETAILS

Default: 500 **Range:** 100 – 5 000 **Defined in:**

TECHNISCH

Erste Iteration, ab der Densification aktiv wird. Vorher passiert nur „nacktes„ Lernen auf der initialen SfM-Punktwolke, ohne dass neue Gaussians erzeugt werden. Der Default 500 stammt aus dem 3DGS-Paper und gibt der Initialisierung Zeit, sich zu stabilisieren — wenn schon ab Iteration 0 densifiziert wird, klonen sich falsch positionierte SfM-Punkte vielfach, bevor sie überhaupt ihren richtigen Platz finden. V349 testete 1000 → leicht schlechter Loss; der Default ist optimal.

EINFACH GESAGT

Wann die App das erste Mal anfängt, Splats zu klonen. Vorher lernt sie nur die schon vorhandenen Punkte. 500 ist der Standardwert — gibt der App genug Zeit, sich erst mal zu orientieren, bevor sie vervielfacht.

T13 densifyInterval

DETAILS

Default: 100 (Initializer, MCMC), 200 (`.full`) **Range:** 50 – 1 000 **Defined in:**

TECHNISCH

Wie viele Iterationen zwischen zwei Densification-Schritten liegen. Im Paper-Default 100 — alle 100 Iterationen wird die Liste der densify-Kandidaten ausgewertet, geklont/gesplittet und gleichzeitig die Liste der prune-Kandidaten (sigmoid(opacity) < `T14 pruneOpacityThreshold`) entfernt. V112-Tests fanden 200 als optimal für `.full` — das entlastet die GPU, weil weniger Reorganisations-Passes laufen, und gibt jeder Gaussian mehr Zeit, sich nach einer Klone-Aktion einzupendeln. V417 testete 100 mit beta2=0.99 → 5.8 % schlechter (957 K Gaussians, Überdensifizierung). Bei MCMC wird dasselbe Feld als Relocation-Interval interpretiert; siehe `T67 mcmcRelocationInterval` für die MCMC-spezifische Logik.

EINFACH GESAGT

Wie oft die App nach neuen Splats schaut. 100 = oft, 200 = mittel. Höher heißt: jeder Splat hat länger Zeit sich einzurichten, bevor wieder vervielfältigt wird. Das ist gut. Auf 50 zu senken kann den GPU dauerhaft beschäftigen, ohne dass es nennenswert besser wird.

T14 pruneOpacityThreshold

DETAILS

Default: 0.005 (Initializer, Paper, MCMC), 0.001 (`.full`) **Range:** 0.0001 – 0.1 **Defined in:**

TECHNISCH

Sigmoid-Opazitäts-Schwelle, unter der eine Gaussian beim nächsten Densification-Step gelöscht wird. Wirkt zusammen mit `T7 opacityLearningRate` und der Logit-Clamp-Logik im Optimizer. V393 senkte den Default von 0.005 auf 0.001 in `.full` — Folge: Splats die nur unter exotischen Blickwinkeln eine Rolle spielen, bleiben länger erhalten und tragen zur SH-Detail bei. V394 testete 0.0001 → leicht schlechter (zu wenig gepruntet, Speicher verschwendet). Wichtig: Density-Control muss IMMER prunen, auch wenn die Buffer-Kapazität durch andere Maßnahmen schon voll ist (siehe „Density Control Must Always Prune„ in CLAUDE.md) — sonst akkumulieren tote Gaussians und der Count friert ein.

EINFACH GESAGT

Wann ein Splat als „durchsichtig genug„ gilt, um gelöscht zu werden. 0.005 ist der Paper-Standard, wir haben in Quality 0.001 — also wir geben Splats länger eine Chance. Das macht weiches Licht und schwache Schatten besser darstellbar. Höher zu setzen (über 0.01) lässt die Splat-Anzahl schnell sinken — kann sinnvoll sein bei Speicherknappheit, kostet aber Detail.

T15 opacityResetInterval

DETAILS

Default: 3 000 (Initializer + Paper), 100 000 (`.full` = effektiv deaktiviert), 200 000 (`.fullMCMC` = deaktiviert) **Range:** 1 000 – 100 000+ **Defined in:**

TECHNISCH

Alle wie viele Iterationen wird die Opazität aller Gaussians auf einen niedrigen Wert (~0.01) zurückgesetzt — eine Maßnahme aus dem 3DGS-Paper, um „eingefrorene„ Splats neu zu beurteilen. V194 zeigte, dass mit RadianceKits Warmup + Stochastic-Trainings-Setup + 2× Lernraten der Opacity-Reset 5.5 % Qualität kostet und der Logit-Clamp die Reset-Funktion bereits abdeckt. Daher in `.full` praktisch deaktiviert (100 000 > 35 000 = nie ausgelöst). V421 testete Reset alle 3 000 mit beta2=0.99 → 4.9 % schlechter; reverted. Bei `.fullClassicPaper` (Q1.5-A, Paper-treuer Test) ist es bewusst wieder auf 3 000 gesetzt — das war eines der Lever, mit denen die Paper-Magnitude-Gaussian-Budgets erreicht werden sollten.

EINFACH GESAGT

Alle wie viele Iterationen die App die Sichtbarkeit aller Splats auf „fast unsichtbar„ zurücksetzt — eine Art Reset-Knopf für die Opacity. Bei uns deaktiviert (Wert so hoch, dass es nie passiert), weil andere Mechanismen das überflüssig machen. Nur bei Paper-getreuen Experimenten anschalten.

T16 maxScreenSize

DETAILS

Default: 0.0 (= deaktiviert) **Range:** 0 (off) oder > 0
Defined in:

TECHNISCH

Maximale Bildschirmraum-Größe (in projizierten Pixeln), die eine Gaussian erreichen darf, bevor sie zwangsweise gesplittet wird. Der Wert ist auf 0 gesetzt (V48 testete und reverted) — RadianceKits Density-Control verwendet stattdessen den Welt-Raum-Skala-Schwellwert aus der `dMean2D`-Logik. Bleibt im Feldkatalog enthalten, weil künftige Experimente mit Mip-Splatting (T74–T76) oder szenenspezifischen Splatting-Strategien davon profitieren könnten. Aktivierung (Wert > 0, z.B. 20) würde sehr groß gewordene Splats im Bildschirm zwingen, sich aufzuteilen — relevant bei großen, glatten Wandflächen, wo ein einzelnes Riesensplat zu wenig Detail bietet.

EINFACH GESAGT

Begrenzung, wie groß ein einzelner Splat auf dem Bildschirm werden darf. Bei uns aus. Eingeschaltet würde es bewirken, dass riesige flache Splats (z.B. auf einer Wand) zwangsweise in mehrere kleine zerlegt werden. Lass es aus, falls nicht ausdrücklich Experimente damit.

Loss (T17–T20)

T17 ssimWeight

DETAILS

Default: 0.2 (Initializer + Paper + `.full`), 0.05 (alle MCMC-Presets) **Range:** 0.0 – 1.0 **Defined in:**

TECHNISCH

Gewicht des D-SSIM-Anteils in der kombinierten Loss-Funktion `loss = (1 − λ) ∗ L1 + λ ∗ D-SSIM`, wobei λ = `T17`. Der 3DGS-Paper-Default 0.2 ist für Classic-Densification optimal — V383 testete 0.3 → 28.9 % schlechter, V373b bestätigte 0.2 als sweet spot. Für MCMC wurde in V521b/V534 unabhängig festgestellt: 0.05 ist optimal, weil MCMC durch seine stochastische Exploration einen stärkeren L1-Signal-Anteil braucht — höhere SSIM-Gewichte würden die Relocation-Entscheidungen verwässern. SSIM ist deutlich teurer zu rechnen als L1 (lokale 11×11-Fenster über das ganze Bild); RadianceKit nutzt eine MPS-beschleunigte Implementierung, die unter 1 ms pro 1080p-Bild bleibt. Q7-BayesOpt-Sweeps fanden szenenspezifische Optima zwischen 0.05 (`.outdoorPreset` : 0.082) und 0.171 (`.indoorPreset`).

EINFACH GESAGT

Wie wichtig die App neben „jeder Pixel stimmt„ auch „Strukturen sind ähnlich“ findet. 0.2 ist der Standard und liefert ein gutes Bild. Niedriger = pixel-genauer, aber kann weichere Übergänge bekommen. Höher = strukturähnlicher, aber Details werden weicher. Lass die Presets entscheiden.

T18 ssimWeightRefinement

DETAILS

Default: 0.0 (= „kein Wechsel, behalte ssimWeight„) **Range:** 0 oder 0 – 1.0 **Defined in:**

TECHNISCH

Optionaler SSIM-Wert für die Refinement-Phase nach `T2 densifyUntilIteration`. V428 testete 0.2 → 0.3 im Refinement → 16 % schlechterer Loss (sowohl L1 als auch SSIM verschlechterten sich); revertet, daher Default 0.0. Die Hypothese hinter dem Feld war, dass nach der Densification — wenn keine neuen Gaussians mehr entstehen — ein stärkerer SSIM-Anteil die strukturelle Schärfe maximieren würde. Empirisch falsch: SSIM-Gewicht zu erhöhen heißt indirekt L1-Gewicht zu senken, und L1 ist das deutlich aussagekräftigere Signal in der Final-Refinement-Phase. Das Feld bleibt verfügbar für künftige Experimente mit perceptueller Loss (T60) oder Edge-Loss (T19), wo eine Refinement-spezifische Loss-Komposition sinnvoll sein könnte.

EINFACH GESAGT

Spezial-Einstellung für die zweite Trainingsphase (Verfeinerung nach Splat-Vervielfältigung). Bei 0.0: dieselbe SSIM-Gewichtung wie davor. Drehen bringt empirisch nichts, daher aus.

T19 edgeLossWeight

DETAILS

Default: 0.0 (= deaktiviert) **Range:** 0 oder 0.001 – 1.0 **Defined in:**

TECHNISCH

V437-Experimental-Loss: Gewicht eines Sobel-Gradient-Domain-L1-Loss, der die Bildkanten direkt vergleicht (Ground-Truth-Sobel vs Render-Sobel) zusätzlich zu L1+SSIM. Hypothese: Kanten-Information ist ein perceptueller Eckpfeiler von Bildqualität und ein expliziter Term sollte Gaussians ermutigen, Kanten besser zu treffen. Test-Ergebnisse: Gewicht 0.1 → 11 % schlechterer Loss, 0.01 → quality-neutral aber 10 % langsamer. Der Sobel-Pass kostet einen zusätzlichen MPS-Forward auf Ground-Truth und Render. Daher dauerhaft deaktiviert. Künftiger Use-Case: Szenen mit harten künstlichen Kanten (Architektur, Möbel, Renderings) könnten profitieren — Q7-Scene-Class-Presets haben das aber nicht gepickt, sondern stattdessen das SSIM-Gewicht skaliert.

EINFACH GESAGT

Experimenteller Zusatz, der Kanten extra wichtig nimmt. Bringt empirisch nichts. Bleibt aus.

T20 skyMaskingEnabled

DETAILS

Default: false (Initializer und alle Presets) **Range:** boolean **Defined in:**

TECHNISCH

Schaltet Sky Masking ein. Dabei wird in jedem Bild via Apple Vision Framework (`VNGenerateForegroundInstanceMaskRequest`) die Sky-Region ausmaskiert, und der Loss in diesem Bereich auf null gesetzt. Sinn: Outdoor-Szenen leiden oft daran, dass blaue/graue/weiße Sky-Pixel die App dazu bringen, Gaussians genau dort zu plazieren — was als „floater„ wahrgenommen wird. Ohne Sky-Mask wäre der Loss in diesem Bereich nie null, weil der Himmel im Bild leicht variiert und die App ewig versucht, das mit Splats nachzubauen. Die Vision-Maske wird einmal pro Kamera vor dem Training berechnet und im RAM gehalten. Wird typischerweise zusammen mit `T45 skyDomeEnabled` aktiviert (UI-Logik in der Settings-View). Bei Innen-Szenen oder synthetischen Renderings deaktiviert lassen — die Maske würde dort fälschlicherweise Decken oder Wände als „Sky“ erkennen.

EINFACH GESAGT

Schaltet einen Spezialmodus für Außen-Aufnahmen an: der Himmel wird beim Training ignoriert, damit nicht versucht wird, ihn mit Splats nachzubauen. Empfohlen für jede Außen-Szene. Bei Innen oder bei 3D-Renderings aus Blender aus lassen.

SH-Degree-Progression (T21)

T21 shDegreeUpgradeIterations

DETAILS

Default: `[1_000, 2_000, 3_000]` (Initializer), `[2_000, 5_000, 8_000]` (`.full` , MCMC), `[1_000, 2_000]` (`.preview` — Degree 3 übersprungen) **Range:** `[Int]` , jeder Wert in `[0, maxIterations]` , monoton steigend **Defined in:**

TECHNISCH

Iterationen, an denen der aktive SH-Degree von 0→1, 1→2, 2→3 hochgeschaltet wird. Vor der ersten Marke sind nur die DC-Komponenten aktiv (also `T5 shDCLearningRate`), nach der ersten Marke die DC + 3 Degree-1-Koeffizienten, nach der zweiten Marke + 5 Degree-2-Koeffizienten, nach der dritten Marke alle 15 Koeffizienten. Der Speicherbedarf pro Gaussian wächst dabei in Stufen — 4 Floats → 16 Floats → 36 Floats → 64 Floats. Die Quality-Presets verzögern die Aufstufungen gegenüber Initializer Defaults (V228), weil die Geometrie zuerst stabilisieren soll, bevor die Farbdetails mit ihrer höheren Frequenz draufkommen. V384 testete `[1K, 2K, 3K]` für `.full` → 9.3 % schlechter — bestätigt das Delay. `.preview` kappt bei Degree 2, weil Degree 3 in 5 000 Iterationen nicht konvergiert und nur Optimizer-Kapazität verbraucht. Q6 (T80–T81) bietet eine alternative Curriculum-Logik, die diese Liste dynamisch überschreibt.

EINFACH GESAGT

An welchen Punkten im Training die App lernt, dass Farben aus verschiedenen Blickwinkeln unterschiedlich aussehen können (Glanzlichter, Spiegelungen). Erst spät — damit erst die Form stimmt, dann die Farbe. Die Werte in den Presets sind so eingestellt, dass das gut funktioniert. Nichts dran ändern, außer du weißt genau warum.

Performance (T22–T25)

T22 trainingRenderScale

DETAILS

Default: 1.0 (Initializer, `.full`, MCMC, Scene-Class), 0.5 (`.preview`), 0.25 (`.quickTest`) **Range:** 0.05 – 2.0 (typisch 0.25, 0.5, 1.0) **Defined in:**

TECHNISCH

Render-Auflösung beim Training relativ zur Originalauflösung der Trainingsbilder. Bei 0.5 wird jedes Bild auf 50 % Breite × 50 % Höhe heruntergerechnet (also 25 % der Pixel) und das Gaussian-Rendering erfolgt in dieser kleineren Auflösung. Reduziert sowohl Speicher- als auch Rechenaufwand quadratisch. Wichtig: `T11 densifyGradThreshold` muss zur gewählten Auflösung passen — die Gradient-Magnituden skalieren mit 1/Auflösung², daher hat `.quickTest` (0.25×) einen viel höheren Threshold (4e-6) als `.full` (1.0×, 1.1e-6). RadianceKit warnt bei sehr großen Bildern und passt automatisch an — 3-MP-Ziel-Auflösung. Bei extremen 4K-Eingangsbildern wäre 0.5 oder sogar 0.25 sinnvoll, sonst läuft jeder Mac auch nur in CPU-Compaction.

EINFACH GESAGT

Wie groß die Bilder beim Training sind. 1.0 = original, 0.5 = halb so groß. Halbe Größe = vier mal schneller, aber feinste Details fehlen. Die Presets wählen den richtigen Wert; bei extrem großen Eingangsbildern (über 12 Megapixel) schaltet die App automatisch runter.

T23 resolutionWarmupScale

DETAILS

Default: 0.0 (= deaktiviert) **Range:** 0 oder 0.1 – **Defined in:**

TECHNISCH

V133-Optimierung: Trainiere die Densification-Phase (Iter 0 bis `T2`) in einer niedrigeren Auflösung als die Refinement-Phase. V308 hat sie für `.full` wieder ausgeschaltet, weil bei `T22 = 1.0` und Cosine-Annealing der Time-Win marginal war und Qualität minimal litt. Bleibt im Feldkatalog, weil sie bei 4K-Eingaben und langen Trainingsläufen wieder sinnvoll werden könnte — Q6 Curriculum (T80) hat eine ähnliche Logik aufgegriffen, dort ist sie aber an die LR-Schedule gekoppelt. Wenn aktiviert und `T80 curriculumResolutionRamp` ebenfalls true, gewinnt Q6 und überschreibt diesen Wert.

EINFACH GESAGT

Spezial-Feature: in der ersten Trainingshälfte mit kleineren Bildern lernen, in der zweiten mit großen. Spart Zeit. Aus, weil die neuere Q6-Variante das besser löst.

T24 tileSize

DETAILS

Default: 16 **Range:** 8, 16, 32 **Defined in:**

TECHNISCH

Größe der Rasterisierungs-Tiles in Pixeln. Das Gaussian-Splatting-Rendering ist tile-basiert: das Bild wird in 16×16-Pixel-Kacheln zerlegt, jede Kachel sammelt die für sie relevanten Gaussians, sortiert sie nach Tiefe und blendet sie ein. 16 ist der von praktisch allen 3DGS-Implementierungen verwendete Standard und in den RadianceKit-Metal-Kernels hartkodiert; eine Änderung dieses Werts würde Re-Compilation der Shader bedingen und ist im aktuellen Stand nicht effektiv. Bleibt als Feld, falls eine künftige Engine-Version Tile-Size dynamisch unterstützt.

EINFACH GESAGT

Interner Render-Parameter. Standard 16, nicht ändern.

T25 throttleDelayMs

DETAILS

Default: 0 (Initializer, `.full`, MCMC, Scene-Class), 0 (`.preview`) **Range:** 0 – 100 **Defined in:**

TECHNISCH

Künstliche Verzögerung zwischen Trainings-Iterationen in Millisekunden. 0 = volle Geschwindigkeit (Standard). Höhere Werte machen den Mac während des Trainings „benutzbarer„, indem GPU/CPU regelmäßig Atempausen bekommen — die Bedienfreundlichkeit anderer Apps steigt, die Trainingszeit aber linear mit dem Delay. Typische Werte: 1–2 ms („leichtes“ Throttling, +5 % Trainingszeit, Mac fühlt sich responsiver an), 5 ms („Mittelschwer„, +15 % Trainingszeit), 10+ ms („Eco“, potentiell doppelte Trainingszeit). Wird im Inspector unter „Performance„ geboten, ist aber nicht in der Standardansicht — siehe Backlog `dev_ux-backlog.md`, der vorschlägt, ihn aus dem Expert View zu entfernen, weil falsch verstanden er die Trainingszeit dramatisch verlängert.

EINFACH GESAGT

Wie viele Millisekunden Pause die App zwischen Trainingsschritten macht. 0 = keine Pause, schnellstmöglich. Höhere Werte machen den Mac während des Trainings besser benutzbar — aber das Training dauert dann auch länger. Auf einem M3 Ultra oder Mac Studio kannst du das auf 0 lassen; auf einem MacBook Air wäre 2 oder 5 ein guter Wert.

Diagnose und Punktwolken-Vorbereitung (T26–T30)

T26 depthDistortionWeight

DETAILS

Default: 0.0 (= deaktiviert) **Range:** 0 oder 0.0001 – 0.05 **Defined in:**

TECHNISCH

V366-Experimental: Gewicht eines Depth-Distortion-Regularisierungs-Loss. Bestraft Gaussians, die entlang eines Render-Strahls zwar tief gestaffelt sind, aber konzeptuell zur selben Oberfläche gehören — das encouraged konzentrierte Tiefenverteilungen und reduziert Floaters. Tests: 0.01 → 4.5 % schlechter, 0.001 → 8.1 % schlechter. Der theoretische Vorteil — Multi-View-Konsistenz verbessern — schlägt sich nicht im L1-Loss nieder, weil die Hypothese implizit annimmt, dass die SfM-Geometrie korrekt ist und die Gaussians nur „gestapelt„ werden müssen. In der Praxis ist die SfM-Punktwolke meist die schwächste Komponente, nicht die Stapelung. Bleibt verfügbar für Multi-View-Datensätze mit besonders sauberen Posen (Synthetic, Mip-NeRF 360 mit Ground Truth).

EINFACH GESAGT

Experimentelles Feature zur Vermeidung mehrerer Splats hintereinander an derselben Stelle. Nicht aktiviert, weil die Tests nichts gebracht haben.

T27 singleViewOverfit

DETAILS

Default: false **Range:** boolean **Defined in:**

TECHNISCH

Diagnose-Flag: wenn true, wird in jeder Trainingsiteration zwingend Kamera-Index 0 verwendet statt einer zufällig aus dem Camera-Pool. Sinn: Wenn das Modell nicht mal eine einzige View überfitten kann (sprich, der Loss auf View 0 auch nach 10 000 Iterationen nicht gegen Null geht), ist im Forward/Backward-Pass ein fundamentaler Bug. Dieser Schalter wurde während der Entwicklung der Metal-Shader und der Differentiable-Rasterizer-Kernels intensiv genutzt — V42–V47 Phase. Heute nur noch als Sanity-Check verfügbar, wenn jemand Backend-Code modifiziert hat und ein Regression Test machen will. Per CLI mit `--single-view`.

EINFACH GESAGT

Test-Modus für Entwickler. Sie können damit prüfen, ob die App überhaupt aus EINEM Bild lernen kann. Für normale Benutzer irrelevant, immer aus lassen.

T28 maxCameras

DETAILS

Default: 0 (= „alle Kameras verwenden„) **Range:** 0 oder 1 – N **Defined in:**

TECHNISCH

Diagnose-Limit aus V43: trainiere nur mit den ersten N Kameras, ignoriere alle weiteren. Sinn ursprünglich: Hypothese testen, dass zu viele Kameras Gradient-Konflikte erzeugen (zu viele widersprüchliche Loss-Signale für dieselbe Gaussian). Test-Ergebnis: kein systematischer Vorteil bei künstlicher Begrenzung — Mehr-Frames bringen praktisch immer mehr Qualität. Bleibt als CLI-Flag (`--max-cameras N`) für gezielte Experimente, z.B. „funktioniert das Training auf den ersten 100 Bildern eines 1 500-Bild-Drohnenflugs?„ Im UI nicht exponiert.

EINFACH GESAGT

Diagnose-Feld für Entwickler — nur die ersten N Bilder verwenden, Rest ignorieren. Normaler Benutzer braucht nicht, Wert auf 0 = alle Bilder. Mehr Bilder = besseres Ergebnis (siehe `feedback_more-frames-better.md`).

T29 maxInitialPoints

DETAILS

Default: 0 (= „alle SfM-Punkte verwenden„) **Range:** 0 oder 1 000 – 200 000+ **Defined in:**

TECHNISCH

V54-Sicherung: limitiert die Anzahl der initialen SfM-Punkte, mit denen das Training startet. Dichte COLMAP-Rekonstruktionen können > 60 000 Punkte produzieren, was bei großen Initial-Skalen zu 200–300 Gaussians pro Pixel-Überlapp führt — das macht ein „Nebelfeld„ aus, in dem das Training nicht konvergiert. Subsampling auf ~16 000 Punkte (Hard-cap-Logik in der Trainings-Engine) bringt die Initial-Dichte auf das Niveau, das das Referenz-3DGS verwendet, und reduziert Overlap dramatisch. Wird bei sehr dichten SfMs automatisch gesetzt; per CLI mit `--max-points N`.

EINFACH GESAGT

Wie viele Anfangs-Punkte aus der Kamera-Rekonstruktion verwendet werden. Bei sehr dichten Rekonstruktionen (mehr als 60 000) limitiert die App automatisch auf 16 000 — sonst gibt es zu viel Nebel am Anfang. Du brauchst das nicht zu setzen; die App regelt es.

T30 cameraClusterOutlierMultiplier

DETAILS

Default: 10.0 (alle Presets — niemals überschrieben) **Range:** 1.0 – 100.0 **Defined in:**

TECHNISCH

Multiplikator für den Camera-Cluster-Outlier-Filter, eingeführt in Phase 3.10 A.1. Vor dem Training berechnet die Trainings-Engine das Centroid aller Kamera-Positionen und die maximale Distanz einer Kamera vom Centroid. SfM-Punkte, deren Distanz vom Centroid `multiplier × maxCameraDistance` überschreitet, werden als Outlier verworfen. Default 10× bewahrt das Verhalten vor Phase 3.10. Ein subtiler Bug: Tighter SfM (Kameras enger zusammen) → kleinerer → kleinerer Schwellwert → mehr Punkte werden als Outlier verworfen. Looser SfM → größerer Schwellwert → weniger Punkte verworfen. Dies ist eine der Ursachen für die Phase-3.9-Funnel-vs-Training-Anti-Korrelation: bessere SfM kann downstream zu schlechterem Training führen, weil zu viele Initialpunkte gekillt werden. Das Feld liegt als CLI-Override (`--camera-cluster-outlier-multiplier`) für die A.3-Sweeps; im UI nicht exponiert. Werte unter 5 sind in der Regel zu restriktiv, über 20 wirkungslos.

EINFACH GESAGT

Spezial-Filter, der Punkte aus der Rekonstruktion verwirft, die weit weg von der Kamera-Wolke liegen. 10 = die App ist großzügig, behält fast alles. Erhöhen kann sinnvoll sein, wenn weit entfernte Punkte (Berge in der Ferne) im Bild aussehen wie schwebende Klümpchen. Niedriger setzen nur in Notfall — verlierst dabei Detail in der Ferne.

Regularisierung (T31–T37)

T31 coarseToFineBlurRadius

DETAILS

Default: 0 (= deaktiviert) **Range:** 0 oder 1 – 10 **Defined in:**

TECHNISCH

V369-Experimental: Box-Blur-Radius, der zu Beginn der Densification-Phase auf das Ground-Truth-Bild angewendet und linear bis zum Ende der Densification (`T2`) auf 0 reduziert wird. Hypothese: Coarse-to-Fine-Training — erst grobe Strukturen lernen, dann Details — sollte stabilere Geometrie liefern. Tests: r=3 → 9.6 % schlechter, r=1 → 5.1 % schlechter. Der Grund für das Fehlschlagen: die Densification entscheidet basierend auf bilddomänen-Gradienten, und Bluren reduziert genau die Signale, die für „hier muss klone werden„ wichtig sind. Bleibt im Feld-Katalog für künftige Tests mit anderem Density-Control-Schema.

EINFACH GESAGT

Experimenteller „Erst-grob-dann-detailliert„-Modus. Hat nichts gebracht, bleibt aus.

T32 scaleRegWeight

DETAILS

Default: 0.0 (= deaktiviert) **Range:** 0 oder 0.0001 – 0.05 **Defined in:**

TECHNISCH

V370-Experimental: L1-Regularisierung auf welträumliche Skala. Bestraft Gaussians, die zu groß werden — verhindert „Mega-Splatts„, die ganze Wandflächen mit einer Gaussian abdecken. Tests: 0.01 → 200 % schlechterer Loss (2 M Gaussians, totale Explosion), 0.001 → 214 % schlechter. Der Grund: Skala-Regularisierung kommt mit Density-Control in Konflikt — kleinere Skalen heißen, mehr Gaussians werden gebraucht, also splittet Density-Control häufiger, was wiederum mehr Gradient-Aufwand bedeutet. Disabled, aber dokumentiert für Mip-Splatting-Experimente (T74): in diesem Kontext könnte eine Skalen-Untergrenze sinnvoll sein.

EINFACH GESAGT

Regularisierung, die Splats zwingt, klein zu bleiben. Hat in Tests Splat-Explosionen ausgelöst (Millionen Splats). Nicht aktivieren.

T33 anisotropyRegWeight

DETAILS

Default: 0.0 (= deaktiviert) **Range:** 0 oder 0.0001 – 0.05 **Defined in:**

TECHNISCH

V445-Experimental: Penalty auf das max(scale)/min(scale)-Verhältnis, soll extrem langgezogene „Needle„-Gaussians verhindern, die als floater wahrgenommen werden. Tests: 0.01 → 69 % schlechter, 0.001 → 15 % schlechter. Der Grund: Regularisierung zwingt Splats in Richtung „runde“ Form, was auf einer flachen Oberfläche (Wand, Tisch, Boden) genau falsch ist — dort ist eine flache, breite Gaussian effizienter als eine kugelförmige. Disabled. V549f bot mit `T34 scaleRatioPruneThreshold` einen alternativen, gezielteren Ansatz, der ebenfalls revertet wurde.

EINFACH GESAGT

Regularisierung, die zu lange dünne Splats bestraft. Klingt sinnvoll, war in Tests aber schlechter. Aus.

T34 scaleRatioPruneThreshold

DETAILS

Default: 0.0 (= deaktiviert) **Range:** 0 oder 5.0 – 100.0 (typisch 10.0 – 30.0) **Defined in:**

TECHNISCH

Experimentelles post-Training-Pruning, das jede Gaussian löscht, deren max(scale)/min(scale)-Verhältnis den hier gesetzten linearen Schwellwert überschreitet. Zielt auf extrem langgezogene „Needle/Disc„-Floaters ab, die durch Regularisierung allein nicht eliminiert werden können. Im Test entfernte das Pruning Floaters wie erhofft, aber gleichzeitig auch sinnvolle flache Splats auf Wänden und Böden — das Bild wurde löchriger. Daher per Default aus, das CLI-Flag (`--scale-ratio-prune N`) bleibt für gezielte Experimente verfügbar. Empfohlene Werte falls man trotzdem testen will: 30 (sehr konservativ, entfernt nur extreme Outlier), 10 (aggressiv, kostet Detail).

EINFACH GESAGT

Versuch, ganz langgezogene Splatts nach dem Training rauszufiltern. War in Tests netto-negativ — Floater weg, aber auch Detail weg. Aus.

T35 opacityRegWeight

DETAILS

Default: 0.0 (= deaktiviert) **Range:** 0 oder 0.0001 – 0.05 **Defined in:**

TECHNISCH

V446-Experimental: Binary-Cross-Entropy-Penalty, der Opazität gegen 0 oder 1 zieht (also weg von „halb-transparent„). Hypothese: scharfere Opazitätsverteilung würde Bildklarheit verbessern. Test mit T33 kombiniert → Regularisierung kostet Qualität, beide deaktiviert. Disabled. Achtung: in 1.4.3-Beta tauchte ein Bug auf, der genau dieses Feld in einer Default-Wert-Veränderung (Initializer = 0.01) hatte, was zu Mass-Extinction des Gaussian-Counts (460 K → 5 in einer Iteration) führte. Seit 1.4.4 fest auf 0.0 als Default verankert.

EINFACH GESAGT

Regularisierung, die Splats entweder ganz transparent oder ganz solid macht. Bringt nichts, kann sogar gefährlich werden (1.4.3-Bug Mass-Extinction). Lass auf 0.

T36 opacityDecayFactor

DETAILS

Default: 0.0 (Initializer = deaktiviert), 0.9995 (`.full`, `.classicBalanced` — HTGS-Standard) **Range:** 0 (off) oder 0.95 – 1.0 **Defined in:**

TECHNISCH

V546-Implementation des HTGS-Schemas (Hierarchical Time-Gating, Eurographics 2025): alle `T37 opacityDecayInterval` Iterationen wird die sigmoid-Opazität jeder Gaussian mit diesem Faktor multipliziert. 0.9995 × 100 Anwendungen ergibt ~95 %-Verbleib pro Densification-Phase — ein leichter aber stetiger Abwärts-Druck auf alle Opacities, der schwach beigetragende Gaussians verlässlich gegen den `T14 pruneOpacityThreshold` sinken lässt. Das Resultat: 14 % besserer L1-Loss auf Horse Full (3-Trial-Avg V546) gegenüber V438 ohne Decay. Nur während Densification-Phase aktiv (bis `T2`), danach läuft das Training ohne Decay weiter, damit die im Refinement etablierten Opacities stabil bleiben. Bei MCMC nicht verwendet (MCMC hat eigene Mechanismen via `T67 mcmcRelocationInterval` + `T68 mcmcDeadOpacityThreshold`).

EINFACH GESAGT

„Sanftes Verblassen„ aller Splats über die Trainingszeit. Macht inaktiv gewordene Splats schneller transparent, sodass sie beim Aufräumen weggehen. War der wichtigste Quality-Hebel des V546-Updates: 14 % besser. In Quality-Preset eingebaut. Selbst zum Drehen nicht empfohlen, weil exakt austariert.

T37 opacityDecayInterval

DETAILS

Default: 50 **Range:** 10 – 500 **Defined in:**

TECHNISCH

Iterations-Intervall, in dem `T36 opacityDecayFactor` angewendet wird. HTGS-Paper-Default 50, in `.full` belassen. Lange Intervalle (>200) heben den Effekt teilweise auf, weil zwischen zwei Anwendungen genug Gradient-Updates passieren, dass Opacity wieder steigt. Kürzere Intervalle (<20) machen Decay zu aggressiv. Nur in Densification-Phase aktiv.

EINFACH GESAGT

Wie oft das „Verblassen„ angewendet wird. 50 = alle 50 Iterationen ein bisschen verblasse-Schritt. Passt.

Refinement (T38–T44)

T38 gradientAccumulationSteps

DETAILS

Default: 1 (= „eine View pro Adam-Schritt„) **Range:** 1 – 8 **Defined in:**

TECHNISCH

V424-Feature: Anzahl der Views, deren Gradienten akkumuliert werden, bevor ein Adam-Update ausgeführt wird. Bei `> 1` läuft die App auf einem separaten, „unfused„ Backward-Project-Pfad, der die Gradienten in einem separaten Buffer summiert; die finale Anwendung skaliert mit 1/N, um die Magnitude konstant zu halten. V424 testete 2-View → quality-neutral, aber 10 % langsamer (weil unfused-Pfad teurer ist als fused-Pfad). Reverted für `.full`, aber für MCMC bewusst verwendet — `.fullMCMC` läuft mit, aber V544a-Tests zeigten, dass mit der Quality-Gap zu Classic auf 5 % schrumpft (statt 11 %). Im Initializer-Default 1, im aktuellen Preset 1, bleibt CLI-Flag (`--accum-steps N`).

EINFACH GESAGT

Wie viele Bilder die App betrachtet, bevor sie die Splats anpasst. 1 = jedes Bild einzeln. Höher = mehrere Bilder gleichzeitig anschauen und dann einen Mittelwert anwenden. Bringt im Standardfall nichts; bei MCMC kann 2 ein bisschen helfen.

T39 testViewIndices

DETAILS

Default: `[]` (= leer, alle Views werden zum Training verwendet) **Range:** `Set<Int>`, beliebige Untermenge der Camera-Indices **Defined in:**

TECHNISCH

V546-Feature: Set von Camera-Indices, die NICHT zum Training verwendet, sondern als Holdout für PSNR/SSIM/LPIPS-Auswertung gespart werden. Wird automatisch gesetzt, wenn der `--benchmark`-CLI-Flag aktiv ist: dann jede achte View, beginnend bei Index 0 (LLFF-Standard, identisch mit den Mip-NeRF-360- und 3DGS-Paper-Konventionen). Ohne benchmark leer — das Training nutzt alle Views. **Vorsicht:** das manuelle Setzen dieses Feldes ohne Verständnis der Indizes kann den Benchmark unbrauchbar machen (z.B. wenn alle Indices über N gesetzt werden, während es nur N-50 Views gibt → keine Holdouts → keine Auswertung). Beim eigenen Preset-Export wird testViewIndices nicht persistiert, weil es szenenabhängig ist und sonst zwischen verschiedenen Datensätzen sinnlose Werte hinterlassen würde.

EINFACH GESAGT

Welche Bilder beim Training „ausgespart„ werden, um sie später für Qualitätsmessung zu nutzen. Du setzt das nicht selbst; das `--benchmark`-Flag macht das automatisch (jedes achte Bild ist Test). Wenn du eigene Indices setzt: gefährlich, kann den Benchmark verfälschen.

T40 refinementPruneInterval

DETAILS

Default: 0 (= deaktiviert) **Range:** 0 oder 100 – 5 000 **Defined in:**

TECHNISCH

V425-Feature: alle N Iterationen während der Refinement-Phase (nach `T2`) wird ein zusätzlicher Prune-Pass laufen gelassen, der Gaussians mit sigmoid(opacity) < `T41 refinementPruneOpacityThreshold` entfernt. Sinn: während Densification gibt es regelmäßige Density-Control-Calls, danach nicht mehr — Gaussians, deren Opacity weiter sinkt, bleiben aber im Buffer. V425 testete und reverted: das zusätzliche Pruning korrelierte mit V426 (Two-Phase Densification, ebenfalls in 0 Gaussians cascade failure abgebrochen). Disabled. CLI-Flag verfügbar für Experimente; falls aktiviert, sind 1 000 oder 2 000 sinnvolle Werte.

EINFACH GESAGT

Zusätzliches Aufräumen während der Verfeinerungsphase. Bringt nichts, bleibt aus.

T41 refinementPruneOpacityThreshold

DETAILS

Default: 0.0 (= „verwende `T14`“) **Range:** 0 oder 0.001 – 0.1 **Defined in:**

TECHNISCH

V425b: separater Opacity-Schwellwert für Refinement-Pruning. Nach Densification haben die meisten Gaussians eine deutlich höhere Opacity erreicht (> 0.001), sodass der Standard-`T14 pruneOpacityThreshold` zu lasch wäre. Wenn `T40` aktiv, bestimmt dieses Feld den eigenen Schwellwert. Bei 0.0 wird `T14` weiterverwendet. Nur relevant wenn T40 > 0.

EINFACH GESAGT

Schwellwert für das zusätzliche Refinement-Aufräumen (siehe T40). Beide Felder nicht aktiv, also irrelevant.

T42 midTrainingCompactificationIterations

DETAILS

Default: `[]` (= deaktiviert) **Range:** `[Int]`, Werte in `(densifyUntilIteration, maxIterations)` **Defined in:**

TECHNISCH

V549-Feature: explizite Iteration-Punkte während der Refinement-Phase, an denen ein Compactification-Pass läuft (entfernt sigmoid(opacity) < 0.01 + Outlier-Scale-Gaussians, dieselbe Logik wie `T56 postTrainingCompactification`). Sinn: lange Refinement-Phasen können Confetti-/Floater-Akkumulation zeigen, deren SH dann auf view-spezifische Artefakte überfitten. Typische Konfiguration falls aktiviert: `[10000, 20000, 30000]` für 40K Classic. **ABER:** V549-A/B-Tests auf Family-Dataset zeigten in allen Konfigurationen schlechteres L1: `[10K,20K,30K]@0.01` → −48 % Count aber +36 % L1; `[20K,30K]@0.005` → −44 % Count aber +45 % L1; `[20K,30K]@0.001` → −17 % Count aber +87 % L1. Daher disabled. CLI-Flag `--mid-compact "10000,20000"` verfügbar, falls man den visuellen Floater-Tradeoff (weniger Confetti im Viewport) gegenüber dem Loss-Regression bevorzugt.

EINFACH GESAGT

Zwischendurch-Aufräumaktionen während des Trainings. In Tests hat das Aufräumen das Endergebnis schlechter gemacht (zwar weniger Floater, aber auch weniger Detail). Aus, kann via CLI eingeschaltet werden, falls Floater dich mehr stören als ein etwas matschigeres Bild.

T43 frustumCullEnabled

DETAILS

Default: false **Range:** boolean **Defined in:**

TECHNISCH

V549b-Feature: nach Training werden alle Gaussians entfernt, die außerhalb der Vereinigung aller Trainings-Kamera-Frusta liegen. Solche Gaussians wurden nie vom Loss-Signal eingeschränkt und sind immer Floater. Besonders effektiv für Szenen, in denen die Novel-View hinter oder neben dem Kamerapfad liegt (z.B. Rückseite eines linearen Drohnenflugs) — die Floater dort werden in der Trainingsphase nie sichtbar, beim späteren Bewegen im 3D-Viewer aber sehr wohl. V549b A/B auf Drohnenflügen positive Ergebnisse, daher als Opt-In verfügbar. Default false, weil bei Object-Captures mit voller Orbit-Coverage die Frustum-Union die ganze Szene umfasst und das Feature nichts entfernt — wird in Settings unter „Floater Reduction„ angeboten und auch in Q9 Outdoor-Preset implizit über `T44 frustumCullExpansion` getestet (Q7-BayesOpt hat es aber nicht aktiviert, weil Outdoor-Sky-Dome dasselbe Problem besser löst).

EINFACH GESAGT

Spezial-Filter für Drohnenflüge oder lineare Aufnahmen: nach dem Training werden Splats gelöscht, die in keiner Kamera „gesehen„ wurden. Optional einschaltbar in den Settings. Bei einfachen Objekt-Aufnahmen unnötig.

T44 frustumCullExpansion

DETAILS

Default: 1.1 **Range:** 1.0 – 2.0 **Defined in:**

TECHNISCH

NDC-Margin für `T43 frustumCullEnabled`. 1.0 würde exakt am Bildrand schneiden, was wackelige Splatts am Bildrand zu sehr kürzen würde. 1.1 = 10 % Padding über die exakte Kamera-Framing hinaus — gibt etwas Toleranz für Randpixel, die in einer leicht versetzten Novel-View doch sichtbar werden könnten. Werte > 1.2 machen den Cull praktisch unwirksam, weil das erweiterte Frustum sehr viel mehr Raum umfasst.

EINFACH GESAGT

Wie streng der oben beschriebene Filter zuschneidet. 1.1 = ein bisschen Sicherheitsabstand zum Bildrand. Lass den Wert.

Sky-Dome (T45–T48)

T45 skyDomeEnabled

DETAILS

Default: false (Initializer + alle Presets außer P9 Outdoor) **Range:** boolean **Defined in:**

TECHNISCH

V549e-Feature: vor Training-Start wird eine kugelförmige Punktwolke generiert (Fibonacci-sphere mit `T46` Sample-Points), in einem Radius von `T47 skyDomeRadiusMultiplier × scene_extent` um den Szenen-Mittelpunkt platziert und mit den Farben aus den sky-maskierten Pixeln aller Trainings-Kameras (siehe `T20 skyMaskingEnabled`) initialisiert. Diese Sky-Dome-Gaussians werden am Anfang des Gaussian-Buffers eingefügt und während Training „eingefroren„ (Position/Skala/Rotation-Gradienten = 0, nur SH und Opacity bleiben optimierbar). Effekt: statt schwarzer „Confetti"-Bereiche in der Ferne sieht der User in Novel-Views einen echten Himmel. V549e-MVP funktioniert auf Drohnen- und Landschaftsszenen sehr gut; in P9 Outdoor-Preset Default-on. Bei Innen-Szenen aus lassen — die Sphere würde sinnlos außerhalb des Raums hängen.

EINFACH GESAGT

Schaltet eine künstliche „Himmelskuppel„ um die Szene ein. Macht Außenaufnahmen viel schöner: statt schwarzer Klümpchen am Bildrand zeigt die App den echten Himmel. Pflicht für Drohnenflüge und Landschaften, sinnlos für Innenräume.

T46 skyDomeSampleCount

DETAILS

Default: 5 000 **Range:** 1 000 – 50 000 (typisch 2 000 – 10 000) **Defined in:**

TECHNISCH

Anzahl der Fibonacci-Sphere-Sample-Punkte auf der Sky-Dome-Sphere. Höhere Werte → dichteres Sky-Dome (besser bei großen Auflösungen und viel sichtbarem Himmel), aber mehr Speicherbedarf. 5 000 ist sweet spot für 4K-Renderings; bei niedrigeren Auflösungen reicht 2 000–3 000. Die Punkte werden nach Cosine-Distance zu jedem Trainings-Kamera-View-Vektor mit den entsprechenden sky-maskierten Pixeln initialisiert — Sample-Points, deren View-Cone keine Kamera sieht, bleiben mit niedrigem Opazitäts-Initialwert hinten, werden aber im Training nicht verändert (eingefroren).

EINFACH GESAGT

Wie dicht der künstliche Himmel ist. 5 000 Punkte reichen normalerweise. Mehr = besserer Übergang aus der Ferne, aber kostet etwas Speicher.

T47 skyDomeRadiusMultiplier

DETAILS

Default: 30.0 (Initializer + meisten Presets), 59.0 (P9 Outdoor, Q7-BayesOpt-Optimum) **Range:** 5.0 – 200.0 **Defined in:**

TECHNISCH

Radius der Sky-Dome-Sphere relativ zur Szenen-Ausdehnung (= mittlere Distanz zwischen Kamera-Positionen). 30 = die Kugel hat den 30-fachen Durchmesser der Kamera-Wolke. Zu klein (< 5) → Sky-Dome interferiert mit der Szene selbst (z.B. ein Sky-Dome-Splatt landet im Vordergrund); zu groß (> 100) → float32-Präzisionsverlust an den Sky-Dome-Positionen, was Render-Glitches in der Ferne auslöst. Q7-BayesOpt auf Bicycle (Mip-NeRF 360) fand 59.0 als szenenspezifisches Optimum für outdoor — das deutet darauf hin, dass die Standard-30.0 für tiefe Landschaften zu klein ist und die Sky-Dome-Pixel in den Bildrand-Bereichen sichtbar als „Wand„ rendern.

EINFACH GESAGT

Wie weit weg die künstliche Himmelskuppel sein soll. 30 = ziemlich weit. Bei großen Landschaften ist 50–60 besser (Outdoor-Preset macht das automatisch). Zu klein wäre, als hätte man Klümpchen direkt vor der Linse.

T48 frozenGaussianCount

DETAILS

Default: 0 (= keine eingefrorenen Gaussians) **Range:** 0 oder 1 – `T46` **Defined in:**

TECHNISCH

Anzahl der Gaussians am Anfang des Buffers, deren Position/Skala/Rotation-Gradienten im Optimizer auf null gesetzt werden — sie bleiben über das gesamte Training räumlich starr. Density-Control darf sie nicht klonen, splitten oder prunen. Genutzt für Sky-Dome-Injection (siehe `T45`): wenn Sky-Dome an ist, wird dieses Feld automatisch auf `T46 skyDomeSampleCount` gesetzt. Manuelles Setzen ist möglich (z.B. um eine vor-platzierte Punktwolke aus einem LiDAR-Scan einzufrieren), aber im UI nicht direkt zugänglich. Wichtig: die ersten N Gaussians im Buffer sind immer die frozen — die Reihenfolge im Buffer entscheidet, nicht ein expliziter Index.

EINFACH GESAGT

Wie viele Splats am Anfang fest sind und sich nicht bewegen dürfen. Wird automatisch auf die Sky-Dome-Anzahl gesetzt, wenn Sky-Dome an ist. Selbst dran drehen brauchst du nicht.

Adam + LR-Schedule (T49–T55)

T49 adamResetIteration

DETAILS

Default: 0 (= deaktiviert) **Range:** 0 oder 100 – **Defined in:**

TECHNISCH

V430-Feature: Iteration, an der die Adam-Optimizer-Momentum-Akkumulatoren (m1, m2) auf null zurückgesetzt werden. Bias-Korrektur danach läuft mit `(iter − adamResetIteration)` statt mit `iter`. V430 testete Reset bei 5 000 (nach Densification-Ende) → 12.8 % schlechter Loss. Grund: das Adam-Momentum, das sich während Densification aufgebaut hat, trägt Information über die typischen Gradient-Magnituden bei und beschleunigt die Refinement-Phase. Es wegzuwerfen kostet die ersten ~500 Iterationen Refinement an Konvergenz. Disabled. Bleibt CLI-Flag für Forschungsexperimente.

EINFACH GESAGT

Reset-Knopf für das interne Adam-Optimierer-„Gedächtnis„. Hat in Tests geschadet, bleibt aus.

T50 positionLRScheduleEndIteration

DETAILS

Default: 0 (Initializer = „verwende maxIterations„), 20 000 (`.full` — Cosine endet bei 20K trotz maxIter=35K), 30 000 (`.fullClassicPaper`) **Range:** 0 oder 1 000 – **Defined in:**

TECHNISCH

V431-Feature: Iteration, an der die Cosine-Annealing-Kurve für Position-LR ihr Minimum erreicht. Wenn 0, ist das identisch mit `T1 maxIterations`. Wenn > 0, läuft die Schedule bis zu diesem Wert und bleibt danach bei `T4 positionLearningRateFinal` konstant. Das erlaubt eine „extended refinement phase„ mit minimaler aber konstanter Lernrate — verfeinert Positionen langsam ohne erneuten Decay. `.full` macht das (Schedule-Ende bei 20K, Training läuft bis 35K), V434c/V434d bestätigten: 15K und 25K beide etwa gleich, 20K minimal optimal. Wird in Verbindung mit `T51` weiterverwendet, um auch die Nicht-Position-LRs in der extended phase zu modifizieren.

EINFACH GESAGT

Wann die App aufhört, die Position-Lernrate weiter zu senken. Wenn niedriger als die maximale Iteration, läuft danach mit konstanter Mini-Rate — das verfeinert sehr langsam aber sehr stabil. In Quality-Preset eingebaut, brauchst nicht zu drehen.

T51 extendedPhaseLRDecay

DETAILS

Default: 0.0 (= deaktiviert, konstante LRs) **Range:** 0 oder 0.01 – 1.0 **Defined in:**

TECHNISCH

V433-Feature: minimaler Multiplikator für die Nicht-Position-LRs (Skala, Rotation, Opacity, SH) in der „extended phase„ — sprich: nachdem `T50` erreicht ist und Position-LR bereits bei `T4` ist. Wenn 0.1, werden Skala/Rotation/Opacity/SH ihrerseits cosine-decayed von 1.0 (= ihr Standard-LR) auf 0.1× ihres Standards. Wenn 0.0 (Default), bleiben sie konstant. V457 testete vollen Decay (0.0 = decay-bis-null) gegen kein-Decay und fand: avg 0.0400 (2 Runs) = gleicher Loss wie V438 ohne Decay. Behavior cleaner mit Decay, aber nicht messbar besser. Daher disabled. Bleibt im CLI als `--nonpos-lr-scale F`.

EINFACH GESAGT

In der späten Verfeinerungsphase auch Farb- und Form-Lernraten kleiner machen. Macht das Training „stabiler„, aber empirisch nicht besser. Aus.

T52 adaptiveDensifyThreshold

DETAILS

Default: false **Range:** boolean **Defined in:**

TECHNISCH

V440-Experimental: wenn true, berechnet die App in jedem Densification-Schritt das p98 der aktuellen Gradient-Verteilung und nutzt es als dynamischen Schwellwert (geclamped auf mindestens 0.5× des konfigurierten Werts aus `T11`, damit es nicht zu sehr ausreißt). Hypothese: Automatische Anpassung an aktuelle Szenen-Phase würde Density-Control robuster machen — z.B. zu Anfang strenger pruning, später lasser, oder umgekehrt. V440 testete und revertete: katastrophaler Drop auf 63 K Gaussians (Mass-Pruning, weil das p98 in den ersten Iterationen extrem hoch ist und dann fast nichts den Threshold überschreitet). Der fixe Threshold ist bereits gut kalibriert, dynamische Anpassung schadet mehr als sie nutzt. Q5 (T77) bietet eine alternative Adaptive-Logik via rolling median, die das Problem umgeht.

EINFACH GESAGT

Adaptive Versions des Densify-Schwellwerts. In Tests katastrophal (Splat-Anzahl auf 63K abgestürzt). Aus. Q5 hat eine bessere Variante davon.

T53 mergeAfterDensification

DETAILS

Default: false (Initializer), true (`.full` , `.classicBalanced` , `.fullClassicPaper`) **Range:** boolean **Defined in:**

TECHNISCH

V438-Feature: am Ende der Densification-Phase (Iter `T2`) wird ein einmaliger Merge-Pass durchgeführt, der nahe beieinanderliegende Gaussians mit ähnlicher Skala und Farbe zusammenfasst. Reduziert die Gaussian-Anzahl um typisch 5–15 % ohne sichtbaren Qualitätsverlust. Sinn: nach intensivem Klonen entstehen Cluster von quasi-identischen Gaussians, die nichts Neues beitragen — das Merging gibt Optimizer-Kapazität für andere Bereiche frei. Standard in Classic-Quality-Presets. Bei MCMC nicht verwendet, weil MCMC durch seine Relocation-Logik solche Cluster gar nicht erst entstehen lässt.

EINFACH GESAGT

Am Ende der Splat-Vervielfältigungs-Phase Klone, die fast identisch sind, zusammenfassen. Reduziert Datenmenge ohne sichtbaren Effekt. Standardmäßig in Quality-Preset an.

T54 densifyPhase2FromIteration

DETAILS

Default: 0 (= deaktiviert) **Range:** 0 oder `T2` – `T1`
Defined in:

TECHNISCH

V426-Experimental: ermöglicht eine zweite Densification-Phase, die nach der Refinement-Pause bei dieser Iteration startet und bis `T55` läuft. Hypothese: nach einer Refinement-Phase haben die Gradient-Akkumulatoren stabilere Magnitudes und können präziser sagen, welche Bereiche noch zusätzliche Gaussians brauchen. V426 testete und revertete: Two-Phase Densification fiel in 0-Gaussians-Cascade-Failure (mit V425 Refinement-Pruning kombiniert zerstörte es den Buffer). Disabled. CLI-Flag verfügbar für Experimente.

EINFACH GESAGT

Zweite Vervielfältigungsrunde nach Pause. Hat in Tests den Splat-Bestand vernichtet. Aus.

T55 densifyPhase2UntilIteration

DETAILS

Default: 0 **Range:** 0 oder `T54` – `T1` **Defined in:**

TECHNISCH

Ende der V426-Two-Phase-Densification. Nur relevant wenn `T54 > 0`. Beide Felder zusammen disabled.

EINFACH GESAGT

Ende der zweiten Vervielfältigungsrunde (siehe T54). Beide aus.

Post-Processing + Apple AI (T56–T60)

T56 postTrainingCompactification

DETAILS

Default: true (in allen Production-Presets), false (`.quickTest` , `.preview`) **Range:** boolean **Defined in:**

TECHNISCH

V443-Feature: nach Trainingsende werden Gaussians mit sigmoid(opacity) < 0.01 hart entfernt (sie tragen praktisch nicht mehr zum Bild bei). Reduziert Gaussian-Count um typisch 58 % und Export-Dateigröße um 55 % ohne sichtbaren Qualitätsverlust. Standardmäßig in Production-Presets aktiv — das Endresultat soll möglichst kompakt ausgeliefert werden können. In `.quickTest` aus, weil ein Diagnose-Lauf sowieso nicht exportiert wird. Anders als `T42 midTrainingCompactificationIterations` (V549) findet die Compactification erst am Ende statt — Refinement kann bis dahin alle Gaussians benutzen.

EINFACH GESAGT

Aufräumen nach dem Training: nahezu unsichtbare Splats werden entfernt. Macht die Export-Datei rund halb so groß ohne Qualitätsverlust. Pflicht-Feature, aus lassen nur in Diagnose-Läufen.

T57 metalFXUpscaling

DETAILS

Default: false **Range:** boolean **Defined in:**

TECHNISCH

V444-Feature: aktiviert Apples MetalFX Spatial Upscaler statt bilineare Interpolation im 3D-Viewer-Output. Wenn Trainingsauflösung < Viewport-Größe (z.B. Training auf 0.5×, Viewport-Anzeige in voller Auflösung), kann MetalFX ein deutlich schärferes Bild liefern. Ändert sich live im Viewport, kein Re-Training nötig. Schließt sich mit `T58 mpsLanczosScaling` aus — MetalFX hat Vorrang. Empfehlung: einschalten, wenn das Bild im Viewer „verwaschen„ wirkt im Vergleich zum erwarteten Detail.

EINFACH GESAGT

Apple-ML-basierte Bild-Schärfung im 3D-Viewer. Hilft, wenn du in einem niedrigerer Auflösung trainiert hast und das Ergebnis im Vollbild zeigst. Live-Toggle, probier aus.

T58 mpsLanczosScaling

DETAILS

Default: false **Range:** boolean **Defined in:**

TECHNISCH

V444-Feature: MPSImageLanczosScale für Viewport-Skalierung statt bilineare Interpolation. Lanczos ist ein klassisches Sinc-basiertes Resampling-Verfahren, das deutlich schärfere Ergebnisse als Bilinear liefert mit minimalem Overhead. Live-Toggle. Wird von `T57` überschrieben, wenn beide an.

EINFACH GESAGT

Klassisches Schärfungsverfahren für den 3D-Viewer (Lanczos). MetalFX (T57) ist ML-basiert und meist besser; Lanczos ist eine weniger aggressive Alternative.

T59 livePreviewInterval

DETAILS

Default: 50 (Initializer und meisten Presets) **Range:** 0 (off) oder 10 – 5 000 **Defined in:**

TECHNISCH

Wie oft während des Trainings der 3D-Viewer mit den aktuellen Gaussians aktualisiert wird. 50 = alle 50 Iterationen ein neues Render im Viewer — gut genug, um den Fortschritt zu beobachten, ohne das Training zu verlangsamen. 0 = Viewer wird gar nicht aktualisiert (Hintergrund-Training, max Geschwindigkeit). Typische Anpassung: bei `.quickTest` runter auf 10 (man will jeden Schritt sehen), bei langen MCMC-Läufen hoch auf 500–2000 (Update-Overhead in Summe spürbar).

EINFACH GESAGT

Wie oft die 3D-Vorschau während des Trainings aktualisiert wird. 50 = alle 50 Iterationen. Höher = weniger oft = etwas schneller, aber du siehst seltener Fortschritt. 0 = keine Vorschau (für maximales Tempo).

T60 perceptualLossWeight

DETAILS

Default: 0.0 (= deaktiviert) **Range:** 0 oder 0.001 – 0.5 **Defined in:**

TECHNISCH

V444-Future-Feature: Gewicht eines perceptuellen Loss-Terms via MPSGraph (VGG-ähnliches kleines Netzwerk). Würde strukturelle und texturelle Ähnlichkeit auf einer höheren semantischen Ebene als L1+SSIM erfassen — typisch in Forschungs-Pipelines, wo „pixel-perfect„ weniger wichtig ist als „sieht realistisch aus“. Implementation noch ausstehend (Code-Stub vorhanden, aber Forward-Pass nicht implementiert). Default 0.0. Bleibt im Feldkatalog für künftige Aktivierung; CLI-Flag `--percep-weight F` reserviert.

EINFACH GESAGT

Geplantes Feature, das mit KI-Hilfe „natürlich aussehend„ anstrebt statt „pixelgenau“. Noch nicht fertig implementiert.

MCMC-Densification (T61–T73)

T61 densificationStrategy

DETAILS

Default: `.classic` (Initializer + Classic-Presets), `.mcmc` (alle MCMC-Presets + Scene-Class) **Range:** `.classic` oder `.mcmc` **Defined in:**

TECHNISCH

Wählt zwischen Classic-Densification (Klon/Split/Prune, Kerbl et al. 2023) und MCMC-Densification (Stochastic Gradient Langevin Dynamics mit Relocation, Kheradmand et al. NeurIPS 2024). Bei `.classic` werden `T11–T16` ausgewertet, bei `.mcmc` die `T62–T73`. Achtung beim Wechsel: Classic Defaults und MCMC Defaults sind völlig anders kalibriert — wer den Picker im Expert View flippt, ohne ein passendes Preset zu laden, riskiert 1.4.3-Bug-Style Mass-Extinction (460 K → 5 in einer Iteration, weil MCMC-OpacityReg auf 0.01 die Classic-Opacities killt). Daher die MCMC-Init Defaults absichtlich „weichgespült„ (alle Reg-Werte 0.0).

EINFACH GESAGT

Welcher Algorithmus zum Vermehren der Splats verwendet wird. Classic = ursprüngliche Methode (schnell, viele Splats). MCMC = neuere Methode (langsamer, viel weniger Splats, dafür kompakter). Die Presets wählen das richtige. Selbst nur umschalten, wenn du auch das passende Preset (P5–P7 oder P8–P10) lädst.

T62 mcmcMaxGaussians

DETAILS

Default: 150 000 (Initializer + `.fullMCMC` + `.mcmcBalanced`), 100 000 (`.mcmcPreview`), 1 500 000 (`.fullMCMCMip` — Mip-Splatting-Variante mit 10× Budget), 1.19 M (`.renderPreset`), 1.25 M (`.outdoorPreset`), 670 K (`.indoorPreset`) **Range:** 0 (= „verwende Buffer-Kapazität„) oder 10 000 – 5 000 000 **Defined in:**

TECHNISCH

Harte Obergrenze für die Anzahl der Gaussians bei MCMC-Strategie. Die Anzahl wächst graduell um `T70 mcmcGrowthRate` (typisch 5 %) pro Relocation-Step bis zu diesem Cap. V473/V531 fanden 150 K als sweet spot — über 200 K dilutet die Splat-Qualität (zu viele kleine, redundante Gaussians), unter 100 K bleibt die Szene unter-densifiziert. Bei sehr großen Szenen (z.B. 1 545-Foto-Drohnenflug mit 158 K SfM-init) ist 150 K zu niedrig — daher die 1.4.5-Erweiterung `T72 mcmcCapMultiplier` + `T73 mcmcAutoScaleByScene`. Q7-BayesOpt fand szenenspezifische Optima zwischen 670 K (Indoor) und 1.25 M (Outdoor). Bei Wert 0 verwendet die Engine die volle Buffer-Kapazität als Cap.

EINFACH GESAGT

Maximale Anzahl Splats bei MCMC. 150 000 ist der Standard und reicht für die meisten Szenen. Outdoor- und Render-Presets (P8, P9) gehen auf 1+ Million für Detail-reichere Szenen. Hochsetzen kann Detail bringen, kostet Speicher; runter ist eher eine Notbremse.

T63 mcmcNoiseScale

DETAILS

Default: 0.00005 (5e-5 = Paper-Default) **Range:** 1e-6 – 1e-3 **Defined in:**

TECHNISCH

Multiplikator für das Gauss'sche Rauschen, das in jeder MCMC-Iteration zur Position jeder Gaussian addiert wird (SGLD-Logik). Höher = mehr Exploration (Gaussians wandern mehr, finden potenziell bessere Plätze), niedriger = mehr Exploitation (Gaussians bleiben da, wo sie schon gut sind). V467 und V536 bestätigten 5e-5 als optimal — 1e-5/2e-5 zu wenig Exploration, 1e-4 zu viel (Splats zerlaufen). Wird über die Trainingszeit cosine-decayed bis `T69 mcmcNoiseDecayEnd` — am Ende des Decay-Bereichs ist Noise effektiv 0 und die Gaussians konvergieren.

EINFACH GESAGT

Wie viel zufälliges „Wackeln„ die App den Splats erlaubt, damit sie selber den besten Platz finden. Standardwert ist optimal getestet. Wenn du das hochdrehst, werden die Splats unruhig.

T64 mcmcOpacityRegWeight

DETAILS

Default: 0.0 (= deaktiviert in den RadianceKit Defaults, Paper: 0.01) **Range:** 0 oder 0.001 – 0.05 **Defined in:**

TECHNISCH

MCMC-spezifische L1-Penalty auf Opacity. Paper-Default 0.01 (drückt unbenutzte Gaussians gegen Null, macht sie für Relocation verfügbar). V464b zeigte aber: ohne Reg ist es in RadianceKit messbar besser (Session 28 bestätigt). Grund: Das mit `T68 mcmcDeadOpacityThreshold` definierte Pruning-Kriterium reicht alleine — eine zusätzliche L1-Penalty zwingt auch wertvolle, niedrig-Opacity-Gaussians zum Sterben. Daher Default 0. **Achtung:** in 1.4.3-Beta-Build war der Initializer-Default fälschlich 0.01, was im Mass-Extinction-Bug resultierte (siehe T61-Erläuterung); seit 1.4.4 auf 0.0 fixiert.

EINFACH GESAGT

MCMC-Spezialregularisierung. Aus, weil der andere MCMC-Mechanismus (Schwellwert in T68) das schon abdeckt. Auf 0 lassen.

T65 mcmcScaleRegWeight

DETAILS

Default: 0.0 (= deaktiviert, Paper: 0.01) **Range:** 0 oder 0.001 – 0.05 **Defined in:**

TECHNISCH

MCMC-spezifische L1-Penalty auf die Skala-Eigenwerte. Paper-Default 0.01. V464b: ohne Reg besser, gleiche Begründung wie T64. Disabled in allen RadianceKit-MCMC-Presets. Achtung wie bei T64: 1.4.3-Bug.

EINFACH GESAGT

Wie T64, aber für Splat-Größe. Aus.

T66 mcmcRelocationInterval

DETAILS

Default: 100 (Initializer + alle MCMC-Presets, Paper-Standard), 155 (P9 Outdoor — Q7-BayesOpt-Optimum) **Range:** 50 – 500 **Defined in:**

TECHNISCH

Iterations-Intervall, in dem MCMC tote Gaussians (sigmoid(opacity) < `T68 mcmcDeadOpacityThreshold`) zu neuen Positionen relociert. V537 testete 50 (zu disruptiv, Loss schwankt) und 200 (marginal schlechter, MCMC verliert Reaktionsfähigkeit). 100 ist optimal. Q7-BayesOpt auf Bicycle fand 155 als szenenspezifisches Optimum für outdoor — die etwas längeren Intervalle geben Adam mehr Zeit, neu plazierte Gaussians zu integrieren, bevor das nächste Reloc-Event sie unter Druck setzt.

EINFACH GESAGT

Alle wie viele Iterationen MCMC die toten Splats irgendwo anders hin verschiebt. 100 ist Standard. Selbst dran drehen brauchst du nicht — Outdoor-Preset hat schon den optimalen Wert.

T67 mcmcWarmupIterations

DETAILS

Default: 500 **Range:** 100 – 5 000 **Defined in:**

TECHNISCH

Anzahl der Initial-Iterationen, in denen noch keine MCMC-Relocation passiert. Erst nach diesem Warmup beginnt die Reloc-Logik. Sinn: in den ersten Iterationen sind die Opacity-Werte noch nicht eingeschwungen — würde direkt mit Reloc gestartet, würden Gaussians an den falschen Stellen plaziert und müssten gleich wieder bewegt werden, was Adam-Momentum zerstört. Paper-Default 500. RadianceKit übernimmt diesen Wert, weil V464b zeigte dass es robust ist.

EINFACH GESAGT

Wie viele Iterationen MCMC erst „ankommen„ darf, bevor es anfängt, Splats umzustellen. 500 ist Standard und passt.

T68 mcmcDeadOpacityThreshold

DETAILS

Default: 0.005 (Initializer, Paper-Standard), 0.01 (`.fullMCMC` und alle MCMC-Presets — V535-Optimum) **Range:** 0.001 – 0.05 **Defined in:**

TECHNISCH

sigmoid(Opacity)-Schwellwert, unter dem eine Gaussian als „tot„ gilt und für Relocation in Frage kommt. V535 fand 0.01 als optimal (0.005 marginal, 0.02 schlechter). Höher = aggressiveres Reloc (mehr Gaussians werden bewegt), niedriger = vorsichtiger. 0.01 entspricht ungefähr „0.5 % visuelle Sichtbarkeit“. P10 Indoor verwendet via Q7-Bayes-Opt 0.0142 als Optimum.

EINFACH GESAGT

Ab welcher Durchsichtigkeit ein Splat als „tot„ gilt, sodass MCMC ihn woanders hinschiebt. 0.01 ist optimal in unseren Tests. Selbst dran drehen brauchst du nicht.

T69 mcmcNoiseDecayEnd

DETAILS

Default: 0 (Initializer = „kein Decay„), 160 000 (`.fullMCMC` = 80 % von 200K), 96 000 (`.mcmcBalanced` = 80 % von 120K), 40 000 (`.mcmcPreview`) **Range:** 0 oder 1 000 – **Defined in:**

TECHNISCH

Iteration, an der das `T63 mcmcNoiseScale` -Rauschen vollständig auf null gedämpft wird (Cosine-Decay von Iter 0 bis hier). V497c/V502 fanden 80 % der maxIterations optimal — gibt MCMC genug Exploration-Zeit, lässt aber die letzten 20 % zur Konvergenz ohne Rauschen. 0 = konstantes Rauschen über alle Iterationen (selten sinnvoll, MCMC kann dann nicht konvergieren).

EINFACH GESAGT

Wann das zufällige „Wackeln„ der Splats aufhört. In den MCMC-Presets bei 80 % der Gesamtiterationen — erst Exploration, dann Konvergenz. Lass den Wert.

T70 mcmcGrowthRate

DETAILS

Default: 0.05 (Paper-Standard = 5 %) **Range:** 0.01 – 0.2 **Defined in:**

TECHNISCH

Wachstumsrate des MCMC-Populations-Targets pro Relocation-Step. Die Logik: bei jedem Reloc-Event wird das Ziel-Populations-Größe um `(1 + growthRate)` erhöht, bis `T62 mcmcMaxGaussians` (oder die per `T72/T73` skalierte Variante) erreicht ist. V512/V522 fanden 0.05 als optimal — höhere Werte führen zu zu schnellem Wachstum (Gaussians werden eingefügt, bevor das Adam-Momentum sie integrieren kann), niedrigere zu unter-densifizierten Szenen am Ende.

EINFACH GESAGT

Wie schnell die Splat-Anzahl bei MCMC wächst. 5 % pro Schritt ist optimal. Lass den Wert.

T71 mcmcSigmoidK

DETAILS

Default: 100.0 **Range:** 10.0 – 500.0 **Defined in:**

TECHNISCH

Sigmoid-Sharpness-Parameter für die MCMC-Noise-Attenuation. Im SGLD-Schritt wird das pro-Gaussian-Rauschen durch gedämpft — hoch-opake Gaussians (deren Logit positiv ist) bekommen exponentiell weniger Rauschen als niedrig-opake. K = 100 ist scharf, sprich der Übergang von „voll-Noise„ zu „kein-Noise“ passiert sehr schnell um Opacity 0.5. V484–V487 fanden K = 100 optimal — kleinere Werte (10–50) lassen auch hoch-opake Gaussians mit-Wackeln (zerstört konvergierte Gaussians), größere (> 500) machen den Übergang künstlich hart und tote Gaussians werden gar nicht mehr bewegt.

EINFACH GESAGT

Spezialparameter, der bestimmt, wie scharf MCMC zwischen „durchsichtig genug zum Verschieben„ und „solid, nicht anrühren“ unterscheidet. Standardwert ist optimal. Nicht dran drehen.

T72 mcmcCapMultiplier

DETAILS

Default: 3.0 (Initializer + `.fullMCMC`), 2.0 (`.mcmcPreview`), 2.5 (`.mcmcBalanced`), 2.98 (P8 Render), 5.32 (P9 Outdoor), 1.76 (P10 Indoor) **Range:** 0 (= deaktiviert) oder 1.0 – 10.0 **Defined in:**

TECHNISCH

1.4.5-Feature: szenen-adaptive Cap-Skalierung. Wenn `T73 mcmcAutoScaleByScene` true ist, wird der effektive Cap als berechnet (geclamped an Buffer-Kapazität). Hintergrund: bei großen Szenen (z.B. 1 545-Foto-Drohnenflug → 158 K SfM-init) ist `T62 = 150 000` zu niedrig — Density-Control würde gar nicht wachsen können. Mit Multiplier 3.0 wird der Cap bei diesem Beispiel auf 474 K skaliert (158 K × 3.0). Q7-BayesOpt fand szenenspezifische Optima: Outdoor profitiert von hohem Multiplier (5.32 → ~830 K Cap bei 156 K bicycle-init), Indoor begnügt sich mit 1.76 (Wände sättigen schneller). Komplette Auflösung des Caps siehe -Methode.

EINFACH GESAGT

Multiplikator, der den Splat-Cap automatisch an die Szenen-Größe anpasst. Große Szene = mehr Anfangspunkte = höherer Cap. Standard 3× passt für die meisten Szenen; Outdoor-Preset geht auf 5× (große Tiefenbereiche), Indoor auf 1.76× (Wände begrenzen ohnehin).

T73 mcmcAutoScaleByScene

DETAILS

Default: true (Initializer + alle MCMC-Presets) **Range:** boolean **Defined in:**

TECHNISCH

1.4.5-Feature: Master-Switch für die scene-aware Cap-Logik (siehe T72 +). Wenn false, wird ausschließlich `T62 mcmcMaxGaussians` als Cap verwendet (zurück zu 1.4.4-Verhalten). Standardmäßig an, weil die Mass-Extinction-Probleme bei großen Szenen aus 1.4.3 sonst wiederkehren. Manuell deaktivieren nur, wenn du explizit ein hartes Cap setzen willst — z.B. um eine 150 K-Variante zu trainieren, deren Endgröße planbar ist.

EINFACH GESAGT

Schaltet die automatische Anpassung des Splat-Caps an die Szenen-Größe an. Standardmäßig an. Aus lassen nur, wenn du selbst genau eine bestimmte Splat-Anzahl haben willst.

Mip-Splatting (Q1.5) (T74–T76)

Status: Q1.5 wurde am 2026-05-25 nach 14 autonomen Iterationen + Overnight-1.5M-Confidence-Check als „closed no-win„ verworfen (max Δ@2× = +0.27 dB, Original-Gate verlangte ≥ +1.5 dB Mittelwert über 0.5×/2×, FAILT auf 0/11 Pair-Scenes). Die Felder bleiben **opt-in** für Forschungs-Experimente; alle Production-Presets haben. Siehe Verdict: `docs/plans/2026-05-25-phase-q1.5-final-verdict.md`.

T74 useMipSplatting

DETAILS

Default: false (alle Production-Presets), true (`.fullMCMCMip` — Forschungs-Sibling) **Range:** boolean **Defined in:**

TECHNISCH

Aktiviert Mip-Splatting (Yu et al. CVPR 2024): 3D-Smoothing-Filter + 2D-Filter + α-Kompensation, der die per-Gaussian-Frequenz auf die Nyquist-Grenze der dichtesten Trainingskamera-Sampling-Rate begrenzt. Theoretisches Ziel: Eliminierung von Aliasing bei Rendering in off-training-Skalen (0.5× oder 2× der Trainingsauflösung). In den Preprocess- und Backward-Projection-Shadern aktiviert, funktional korrekt verifiziert in Q1.5-D-Test. Aber: der Original-Akzeptanz-Gate (Δ@1× ≥ +0.3 dB UND avg(Δ@0.5×, Δ@2×) ≥ +1.5 dB) wurde auf keinem von 11 Pair-Scenes erreicht. Maximal beobachtet: family 750K classic Δ@2× = +0.270 dB. Outdoor-Szenen (Truck, Flowers) zeigten sogar Verschlechterung 1× und 0.5×. Hypothese: 3D-Smoothing konkurriert mit MCMC-Relocation bei high-Gs. Feld bleibt für künftige Multi-Scale-Re-Eval mit korrekter Mip-NeRF-360-Methodology (siehe O3-Backlog im Benchmark-Pfad).

EINFACH GESAGT

Aliasing-Filter aus einem 2024er-Paper. Theoretisch toll, praktisch hat er in unseren Tests nichts gebracht und manchmal sogar geschadet. Bleibt verfügbar für Experimentierer, aber wir empfehlen ihn nicht. Aus lassen.

T75 mipSmoothing3DScale

DETAILS

Default: 0.2 (Paper-Default) **Range:** 0.05 – 1.0 **Defined in:**

TECHNISCH

3D-Smoothing-Skala-Parameter (Yu et al. §3.3, Paper-Default 0.2). Größer = mehr Welt-Raum-Glättung pro Gaussian (= mehr Anti-Aliasing, aber auch mehr blur in der Default-Skala), kleiner = schärfer aber anfälliger für Aliasing. Wird nur konsultiert, wenn `T74 useMipSplatting = true`. In Q1.5-Tests nicht weiter optimiert — die A/B-Gate hat schon mit Paper-Default 0.2 verloren, weitere Sweeps wären zwecklos.

EINFACH GESAGT

. Wenn du Mip nicht eingeschaltet hast, irrelevant.

T76 mipFilter2DVariance

DETAILS

Default: 0.3 (= exakt das V242-Legacy-Verhalten) **Range:** 0.1 – 1.0 **Defined in:**

TECHNISCH

2D-Mip-Filter-Varianz, die zur Σ_2D-Diagonale addiert wird (Varianz direkt, nicht quadriert). 0.3 ist exakt der V242-Legacy-Wert, der vor Mip-Splatting hardcoded im Kernel war. Wenn `T74 useMipSplatting = false`, ignoriert der Kernel diesen Wert komplett und schreibt das hartkodierte 0.3 — sodass die Baseline nicht regredieren kann (Codex-Round-1-S3-1-Garantie). Wenn, wird der hier gesetzte Wert verwendet. Bleibt im Feld-Katalog für Mip-Sweeps.

EINFACH GESAGT

Weiterer Mip-Splatting-Parameter. Bei Mip-aus: irrelevant.

Adaptive Densification (Q5) (T77–T79)

T77 adaptiveDdensification

DETAILS

Default: false **Range:** boolean **Defined in:**

TECHNISCH

Q5-Feature: rolling-median-Tracker als Alternative zum fixen `T11 densifyGradThreshold`. Wenn true, wird in jedem Densify-Step der aktuelle Schwellwert mit `median(letzte N avgGrad-Samples) × T79 adaptiveDensifyMultiplier` überschrieben. N = `T78 adaptiveWindow`. Strikter als V440 p98 (die katastrophale 63 K-Pruning-Falle), median + 2× sitzt etwa beim p70–p80 der Gradient-Verteilung in steady state. Q5-Tests: alleinstehend FAIL 0/3 Szenen, aber gemeinsam mit Q6 (siehe T80/T81) PASS 1/3 Szenen — das Bundle Q5+Q6 wurde 2026-05-25 als opt-in gepasst und ist via CLI `--adaptive-densify` aktivierbar. Q6 ist dabei der „Carrier„ des Quality-Gewinns, Q5 trägt eher zur Stabilität bei.

EINFACH GESAGT

Selbstlernender Densify-Schwellwert. Statt einer fest eingestellten Empfindlichkeit passt die App sich der Szene an. Alleine getestet nicht besser, mit dem Curriculum aus Q6 zusammen aber ja. Beide gemeinsam einschalten oder beide aus.

T78 adaptiveWindow

DETAILS

Default: 1 000 **Range:** 100 – 10 000 **Defined in:**

TECHNISCH

Rolling-Median-Window in Densification-Events (NICHT Iterationen — jeder `T13 densifyInterval`-Step liefert ein Sample). Default 1 000 — bei heißt das die letzten 100 000 Trainings-Iterationen tragen zum Median bei, also typisch die gesamte Trainingshistorie bis hierhin. Frühe Phase (vor `T78` Samples): Tracker returns nil → Fallback auf fixen Threshold `T11`. Nur relevant wenn.

EINFACH GESAGT

Wie viele alte Densify-Schritte in den Median für T77 einfließen. Standard 1000 ist gut. Nur relevant wenn Q5-Adaptive an.

T79 adaptiveDensifyMultiplier

DETAILS

Default: 2.0 **Range:** 1.0 – 4.0 **Defined in:**

TECHNISCH

Multiplikator auf den Rolling-Median für den adaptiven Schwellwert. Default 2.0 entspricht ungefähr p70–p80 der typischen Gradient-Verteilung. Niedriger = aggressiveres Wachstum (mehr Klone), höher = strenger (weniger Klone). Q5-Tests in Range 1.5–3.0 — 2.0 bestes Default. Nur relevant wenn.

EINFACH GESAGT

Faktor für T77/T78. Standard 2.0 = strenger als der typische Median. Nicht dran drehen.

Curriculum (Q6) (T80–T81)

T80 curriculumResolutionRamp

DETAILS

Default: false **Range:** boolean **Defined in:**

TECHNISCH

Q6-Feature: Trainings-Auflösung startet bei 0.5× und wechselt bei `T50 positionLRScheduleEndIteration / 2` (oder `T1 maxIterations / 2`, falls T50 nicht gesetzt) auf `T22 trainingRenderScale`. Verwendet die in Q1.5.1 entwickelte `resize/restoreImageBuffers`-Infrastruktur. Überschreibt `T23 resolutionWarmupScale`, wenn aktiviert. Q6 ist als „Carrier des Quality-Gewinns„ im Q5+Q6-Bundle gepasst (siehe T77) — die schrittweise Auflösungserhöhung gibt der App Zeit, grobe Geometrie auf der niedrigeren Auflösung zu finden, bevor sie zur feinen Detailarbeit übergeht. Über CLI: `--curriculum-resolution`.

EINFACH GESAGT

„Erst grob, dann fein„ für die Trainingsauflösung. Halbe Auflösung in der ersten Hälfte, dann volle Auflösung. Hilft in bestimmten Szenen, in anderen nicht — am besten mit T81 zusammen einschalten.

T81 curriculumSHProgression

DETAILS

Default: false **Range:** boolean **Defined in:**

TECHNISCH

Q6-Feature: überschreibt `T21 shDegreeUpgradeIterations` mit `[maxIter/4, maxIter/2, maxIter*3/4]`, verteilt also die SH-Aufstufungen gleichmäßig über die Trainingszeit statt sie front-zu-loaden. Hypothese: stabile Geometrie wird vor Color-Detail-Explosion etabliert, was die View-Direction-abhängigen Glanz-Effekte präziser positioniert. Q5+Q6 zusammen PASS 1/3 Szenen, Q6 als Carrier des Gewinns (Q5 alone FAIL). Via CLI: `--curriculum-sh`.

EINFACH GESAGT

„Erst Form, dann Farbe„ — die Glanz-Effekte werden erst spät im Training freigeschaltet, sodass die Splats zuerst ihre Position und Größe finden. Mit T80 zusammen einschaltbar; alleine bringt es nicht ganz so viel.

Statische Presets (TP1–TP9)

Hier nur die strukturellen Unterschiede zum Initializer-Default. Die volle Marketing-Beschreibung der elf UI-Presets P1–P11 findest du in Kapitel 7.

TP1 `.preview`

DETAILS

Diagnose-/Vorschau-Preset für Systeme ≥ 10 GB RAM. Overrides gegenüber Initializer: - 30 000 → 5 000 - 15 000 → 3 500 (70 % von maxIter) - 1.6e-6 → 1.6e-5 (10× höher, weniger aggressiver Decay) -,,,, jeweils 2× (V176) - 3 000 → 100 000 (effektiv aus, V172: Reset zerstört kurze Trainings) - `[1K, 2K, 3K]` → `[1K, 2K]` (V182: Degree 3 konvergiert in 2K Iter nicht) - 1.0 → 0.5

EINFACH GESAGT

jede beliebige Initial-Begutachtung einer neu importierten Bildreihe — 2–3 min Wartezeit, anschließend reicht das Ergebnis für eine binäre Frage „lohnt sich Quality-Lauf?„.

TP2 `.full`

DETAILS

Production-Quality Classic. Overrides: - 30 000 → 35 000 (V550: 40K-Tests Truck-Overtraining +10.7 % Gs bei –1.3 % L1) - 15 000 → 5 000 (V310 sweet spot, V338 7K worse) - Alle LRs 2× (V188) - 1.6e-6 → 1.6e-5 (V45 10×) - 2e-6 → 1.1e-6 (V335) - 100 → 200 (V112) - 0.005 → 0.001 (V393) - 3 000 → 100 000 (V194 disabled, V421 confirmed) - `[1K, 2K, 3K]` → `[2K, 5K, 8K]` (V228 delayed) - 0.0 → 0.9995 (V546 HTGS, 14 % Verbesserung) - 50 (unverändert, V546) - false → true (V438) - 0 → 20 000 (V431) - true (V443, schon Initializer-Default für `.full`)

EINFACH GESAGT

jede Standard-Foto-Aufnahme (Objekt, kleiner Raum, Skulptur) mit < 500 Bildern. Die in V546 angekündigte 14 % Loss-Verbesserung gegen V438 wurde 3-Trial-gemittelt auf Horse Full bestätigt.

TP3 `.fullClassicPaper`

DETAILS

Q1.5-A-Test-Sibling von TP2, paper-treuer Classic. Overrides gegenüber TP2: - 35 000 → 30 000 (Paper-Standard) - 5 000 → 15 000 (Paper: 50 % von maxIter) - 1.6e-5 → 1.6e-6 (Paper-Default) -,, zurück auf Paper Defaults (0.05, 0.005, 0.001) - 1.1e-6 → 2e-7 (kalibriert für ~1-2M Gs auf Bicycle) - 200 → 100 (Paper) - 0.001 → 0.005 (Paper-Default) - 100 000 → 3 000 (Paper §5.2, riskant — kann V194-Regression auslösen) - 0.9995 → 0.0 (Paper hat keinen Decay) - 20 000 → 30 000 (cosine läuft auf 100 % von maxIter)

EINFACH GESAGT

Q1.5-Forschungs-Experimente, die Paper-Magnitude-Gaussian-Budgets (1–2 M) für Mip-Splatting-Tests brauchen. Nach Q1.5-„closed no-win„-Verdict bleibt das Preset für advanced users zugänglich, ist aber nicht Production-empfohlen.

TP4 `.fullMCMC`

DETAILS

Production-Quality MCMC. Overrides gegenüber Initializer: - 30 000 → 200 000 (V534, MCMC braucht 5× mehr Iter als Classic) - 15 000 → 160 000 (V504b 80 % von maxIter) - 1.6e-6 → 1.6e-5 - LR-Schedule wie TP2 (alle 2×) - 0.2 → 0.05 (V521b/V534: MCMC braucht stärkeres L1-Signal) - `[1K, 2K, 3K]` → `[2K, 5K, 8K]` - `.classic` → `.mcmc` - 150 000 (im Initializer schon, im Preset bestätigt) - 5e-5 (V467/V536 optimal) - 0.005 → 0.01 (V535 optimal) - 0 → 160 000 (80 % von maxIter, V497c/V502) - 3.0 (im Initializer schon) - true (im Initializer schon) - 3 000 → 200 000 (effektiv aus, MCMC nutzt Reloc statt Reset)

EINFACH GESAGT

Web-Auslieferung, Object-Captures mit Detail-Anspruch, Drohnenflüge (auch wenn dann P9 Outdoor noch besser ist). 71 % weniger Gaussians als Classic bei vergleichbarer L1.

TP5 `.fullMCMCMip`

DETAILS

Q1.5-D-Test-Sibling von TP4, mit Mip-Splatting + Paper-Magnitude-MCMC-Budget. Overrides gegenüber TP4:

- mcmcMaxGaussians 150 000 → 1 500 000 (10×, Paper-Magnitude)
- useMipSplatting false → true (Mip-on)

EINFACH GESAGT

Alle anderen Felder identisch zu TP4. Q1.5 D-PASS auf Bicycle 2026-05-24 (bricht 12-Iter-Multi-Scale-FAIL-Streak). Q1.5-Endurteil 2026-05-25 trotzdem closed-no-win — Mip-Splatting-Gewinn nicht reproduzierbar über 11 Pair-Scenes. Preset bleibt opt-in.

TP6 `.classicBalanced`

DETAILS

Mid-Tier Classic. Overrides gegenüber TP2: - 35 000 → 20 000 (V149: 20K = 30K bei 33 % weniger Zeit) - 20 000 → 0 (Cosine läuft auf maxIter = 20K, kein extended phase)

EINFACH GESAGT

Standardfälle mit kürzerer Wartezeit. V149 als sweet spot identifiziert.

TP7 `.mcmcPreview`

DETAILS

MCMC-Diagnose. Overrides gegenüber TP4: - 200 000 → 60 000 (V494b) - 160 000 → 48 000 (80 %) - 150 000 → 100 000 (V473b) - 160 000 → 40 000 (V494b) - 3.0 → 2.0 (1.4.5: Preview = lighter scaling)

EINFACH GESAGT

schnell ein MCMC-Resultat sehen, um zu beurteilen, ob TP4 oder ein Scene-Class-Preset lohnt.

TP8 `.mcmcBalanced`

DETAILS

Mid-Tier MCMC. Overrides gegenüber TP4: - 200 000 → 120 000 (V518) - 160 000 → 96 000 (80 %) - 160 000 → 96 000 (80 %) - 3.0 → 2.5 (zwischen Preview 2.0 und Full 3.0)

EINFACH GESAGT

MCMC ohne den vollen 200K-Lauf. ~120 K Iterationen sind ein guter Kompromiss aus Qualität und Wartezeit.

TP9 `.quickTest`

DETAILS

Reiner Funktionstest. Overrides gegenüber Initializer: - 30 000 → 1 000 - 15 000 → 500 - 2e-6 → 4e-6 (kalibriert für 0.25× Auflösung) - 100 → 50 - 3 000 → 100 000 (aus, da viel zu kurz) - 1.0 → 0.25

EINFACH GESAGT

Sanity-Check „startet das Training überhaupt sinnvoll?„. Dauer < 30 s auf M3 Ultra. Sieht garantiert matschig aus.

Methode:

Signature: `public func resolveMcmcMaxGaussians(initialPointCount: Int, bufferCapacity: Int) -> Int` **Defined in:**

Einzige Source-of-Truth für die Frage „wie viele Gaussians darf MCMC maximal wachsen lassen?„. Berechnet sich aus drei Eingaben: dem konfigurierten `T62 mcmcMaxGaussians` (mit Mass-Extinction-Floor 150 000, falls 0), dem (Anzahl der SfM-Init-Punkte) und der (vorallozierte Gaussian-Buffer-Größe). Logik:

1. `base = T62 > 0 ? T62: 150_000` (der Mass-Extinction-Floor schützt gegen Initializer-Default-Bugs wie den 1.4.3-Mass-Extinction-Vorfall)
2. Falls `T73 mcmcAutoScaleByScene && initialPointCount > 0 && T72 mcmcCapMultiplier > 0`:
   • `scaled = max(base, ceil(initialPointCount × T72))` sonst
3. Falls `bufferCapacity > 0` : `return min(scaled, bufferCapacity)`

4. Sonst `return scaled`

Beispiel: Bicycle (Mip-NeRF 360, 194 Foto-Frames) → SfM-init ~156 K Punkte, `T62 = 150 000`, `T72 = 5.32`,, Buffer-Kapazität 8 M. Resolved Cap = `min(8M, max(150K, ceil(156K × 5.32))) = min(8M, 830K) = 830 K`. Das ist der effektive Wachstums-Cap, an den die MCMC-Relocation-Logik sich hält.

💬 Berechnet die wirkliche maximale Splat-Anzahl bei MCMC. Nimmt deine Einstellung, schaut sich an, wie viele Punkte deine Szene anfangs hat, und skaliert mit dem `Multiplier`, falls die automatische Anpassung an ist. So passt sich der Cap an die Szene an, statt für eine kleine und eine riesige Szene den gleichen Wert zu erzwingen. Du musst die Methode nicht selbst aufrufen — das Training nutzt sie intern.

Welches Feld wofür? (Cheat-Sheet)

Ziel	Felder zum Drehen
Mehr Detail in der Ferne	`T62 mcmcMaxGaussians` hoch, `T72 mcmcCapMultiplier` 5+
Mehr Detail allgemein (Classic)	`T1 maxIterations` hoch (≤ 40K), `T2 densifyUntilIteration` ≤ 14 % von T1
Floater in Drohnenflügen reduzieren	`T43 frustumCullEnabled` an, `T20 skyMaskingEnabled` an, `T45 skyDomeEnabled` an
Schöner Himmel in Außen-Szenen	`T45 skyDomeEnabled` an, `T47 skyDomeRadiusMultiplier` 30–60
Kleinere Export-Datei	Strategie `.mcmc` (T61), `T56 postTrainingCompactification` an, `T62 mcmcMaxGaussians` ≤ 200K
Schnelleres Training	`T22 trainingRenderScale` 0.5, `T1 maxIterations` halbieren — aber nicht beides!
Bessere Glanzlichter	`T21 shDegreeUpgradeIterations` mit `[2K, 5K, 8K]` (kein early-front-load), MCMC + 200K iter
Mac responsiv halten	`T25 throttleDelayMs` 5–10 (kostet ~15 % Trainingszeit)
Live-Vorschau häufiger	`T59 livePreviewInterval` runter auf 10–20
Sanftere Übergänge an Schatten	`T17 ssimWeight` etwas hoch (0.15–0.25), aber nicht über 0.3
Innenräume kompakt halten	P10 Indoor-Preset (, `T72 = 1.76`)

Gefährliche Felder

Diese Felder können bei Falsch-Konfiguration zu OOM, App-Crash, Mass-Extinction der Gaussians oder unbrauchbaren Benchmark-Daten führen. Mit Vorsicht zu behandeln:

- `T11 densifyGradThreshold` — eine Halbierung kann 2–4× so viele Gaussians erzeugen, was schnell den GPU-Speicher sprengt. Auch zu beachten: muss zur `T22 trainingRenderScale` passen (1.0× → 1e-6, 0.5× → 2e-6, 0.25× → 4e-6).
- `T72 mcmcCapMultiplier` — bei großen Szenen mit > 200 K SfM-init-Punkten und Multiplier > 5 entsteht ein Resolved-Cap von Millionen Gaussians. Auf 36-GB-RAM-Macs OOM möglich. Outdoor-Preset 5.32 funktioniert nur, weil Mip-NeRF-360-Bicycle 156 K init-Punkte hat → 830 K Cap.
- `T39 testViewIndices` — manuelles Setzen kann den Benchmark unbrauchbar machen (alle Indices > N → keine Holdouts). Lass das `--benchmark`-Flag das setzen.
- `T64 mcmcOpacityRegWeight` **und** `T65 mcmcScaleRegWeight` — In 1.4.3-Beta auf 0.01 gesetzt, was zu Mass-Extinction führte (460 K → 5 Gaussians in einer Iteration). Seit 1.4.4 auf 0.0 fixiert, aber manuelles Erhöhen kann das Problem reproduzieren.
- `T15 opacityResetInterval` — wenn nicht 100 000+ (effektiv aus) und das Training kürzer als 10 000 Iterationen ist, zerstört der Reset die Konvergenz. `.preview` hat es deshalb auf 100 000 trotz `maxIterations = 5 000`.
- `T54/T55 densifyPhase2*` — Two-Phase Densification ist in Tests in 0-Gaussians-Cascade abgebrochen. Lass beide auf 0.
- `T74 useMipSplatting` — Q1.5 closed-no-win 2026-05-25, kann auf manchen Outdoor-Szenen sogar PSNR verschlechtern. Default off, opt-in nur für Forschung.

Wenn ein Feld auf dieser Liste steht und du es ändern willst, mache vorher eine Sicherung deines aktuellen Presets (Export als JSON) und überlege, ob du das Resultat reproduzierbar messen kannst — sonst weißt du hinterher nicht, ob du eine Verbesserung oder Verschlechterung herbeigeführt hast.

KAPITEL

Kapitel 7 — Eingebaute Qualitäts-Presets

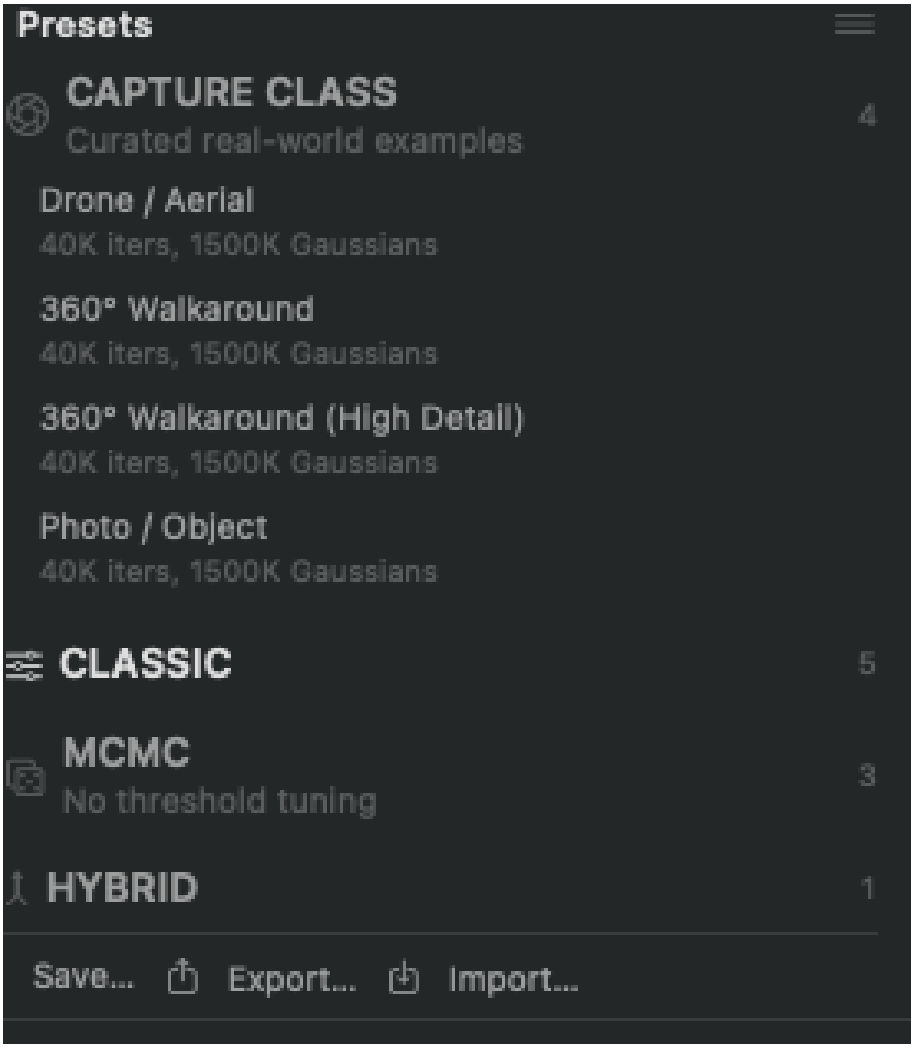

Abbildung 27: Vorlagen-Sektion mit allen vier Gruppen ausgeklappt — CAPTURE CLASS (4 Presets: Drone/Aerial, 360° Walkaround, 360° Walkaround (High Detail), Photo/Object), CLASSIC (5 Presets: Quick/Preview/Balanced/Quality/Ultra Detail), MCMC (3 Presets, Hinweis „No threshold tuning„), HYBRID (1 Preset: Balanced (Hybrid))

WAS IM BILD ZU SEHEN IST Presets-Sektion im Inspector, alle vier Gruppen ausgeklappt. CAPTURE CLASS mit den vier kuratierten Real-World-Presets (Drone / Aerial, 360° Walkaround, 360° Walkaround (High Detail), Photo / Object) — das ist die primäre Gruppe und im Simple-Mode die einzige sichtbare. CLASSIC mit Quick (1K iters), Preview (5K

iters, aktive Auswahl mit blauem Häkchen), Balanced (20K iters), Quality (35K iters) und Ultra Detail (35K iters). MCMC mit Untertitel „No threshold tuning„ — MCMC braucht keinen Densify-Until-Threshold: Preview (60K iters, 100K Gaussians), Balanced (120K, 150K), Quality (200K, 150K). HYBRID mit dem Balanced (Hybrid) (20K iters, 150K Gaussians). Footer-Action-Row: Save…, Export…, Import…

Ein Preset ist eine vorbereitete Konfiguration für das Training. RadianceKit liefert dreizehn eingebaute Presets in vier Gruppen aus: vier **Capture-Class**-Presets (P9–P12) — kuratierte, an echtem Community-Material per Auge validierte Rezepte für reale Aufnahme-Arten (Drohne, 360°-Rundgang, Foto-Objekt) und die primäre Achse seit v1.6 —, fünf Classic-Presets (P1–P5: Quick/Preview/Balanced/Quality/Ultra Detail), drei MCMC-Presets (P6–P8) und ein Hybrid-Preset (P13), das die Classic- und MCMC-Strategien kombiniert. Die früheren „Scene-Class„-Presets (Render/3D, Outdoor, Indoor, in Phase Q7 akademisch gegen Mip-NeRF-360- und NeRF-Blender-Szenen getuned) wurden in v1.6 als sichtbare Gruppe zurückgezogen — die per Auge an echtem Footage validierte Capture-Class ist jetzt die primäre Achse; die Q7-getunten Konfigurationen bleiben nur noch intern erhalten. Du wählst die Presets in der Sidebar im Bereich **Presets** oder im Simple-Mode beim Import. Die `+`-Buttons öffnen Dialoge, um eigene Presets daneben anzulegen — die dreizehn eingebauten lassen sich nicht löschen, aber duplizieren.

Im Expert-View erscheinen die Presets gruppiert nach Aufnahme-Art und Strategie (Capture Class / Classic / MCMC / Hybrid). Ein Klick auf einen Eintrag schreibt die hinterlegte Trainingskonfiguration in den aktuellen Zustand. Das ist kein Snapshot — wenn du danach an Schiebereglern drehst, ändert sich der Zustand, aber das Preset selbst bleibt unverändert; ein farbiger Hinweis zeigt dann „modified„.

Welches Preset wann das richtige ist, hängt vor allem von Szenen-Typ und Hardware ab. Die drei tabellarischen Übersichten am Kapitel-Ende fassen das zusammen.

P1 — Quick

WO

Inspector → Presets-Sektion → Gruppe „Classic„ → Eintrag „Quick“. UUID-Suffix `…001`.

TECHNISCH

Diagnose-Preset mit 1 000 Iterationen, klassischer (adaptiver) Densification-Strategie und einer Trainings-Auflösungs-Skalierung von 0.25× (Eingabebild wird vor dem Training auf 25 % verkleinert). Soll nicht zur Auslieferung einer Szene dienen, sondern schnell feststellen, ob das Setup (Kameraposen, Punktwolke, Bildreihe) überhaupt sinnvolle Bewegung in den Loss-Werten zeigt. Auf einem M3 Ultra typischerweise unter 30 Sekunden auf 50–200 Bildern. Die kleine Auflösung verschleiert echte Bildqualität, hält aber Speicherbedarf und Render-Aufwand sehr gering. Wird auch automatisch als Default beim ersten Start gewählt, wenn das System weniger als 10 GB RAM hat.

EINFACH GESAGT

Schneller Funktionstest. Bilder rein, eine knappe halbe Minute warten, gucken ob der Rohumriss der Szene erscheint. Wenn das Bild im Viewer wie ein matschiger Klecks aussieht — passt, das soll so sein. Wenn du dagegen nur dunkle Punkte oder eine total verzerrte Form siehst, sind wahrscheinlich die Kameraposen falsch (siehe Kapitel 9). Für ein zeigbares Ergebnis brauchst du danach mindestens P2 oder P3.

P2 — Preview (Classic)

WO

Inspector → Presets-Sektion → Gruppe „Classic„ → Eintrag „Preview“. UUID-Suffix `…002`.

TECHNISCH

5 000 Iterationen Classic-Densification, 0.5× Auflösungs-Skalierung, doppelte Lernraten gegenüber Standard. Densification (Klonen + Splitten) ist über die ersten 2 500 Iterationen aktiv, danach nur noch Pruning. Default-Preset für Systeme mit ≥ 10 GB RAM. Auf einem M3 Ultra typischerweise 90 Sekunden bis 3 Minuten für eine 200-Bild-Szene. Liefert einen brauchbaren Eindruck der Geometrie und der Kamera-Pose, aber Texturen sind sichtbar weichgezeichnet — die 0.5× Render-Auflösung lässt sich danach durch erneutes Trainieren mit P3 oder P4 nicht direkt umgehen, weil die Lernraten passend zur halben Auflösung kalibriert sind.

EINFACH GESAGT

Der Standard für „einmal kurz anschauen„. Wenn du gerade neue Bilder importiert hast und sehen willst, ob die Szene überhaupt rekonstruierbar ist, ist das die richtige Stufe. Etwa 2–3 Minuten Wartezeit, danach kannst du im 3D-Viewer drehen und beurteilen, ob die Anschaffung weiterer Trainingsdurchgänge sinnvoll ist. Erst wenn das Vorschau-Ergebnis bereits gut aussieht, lohnt sich Balanced oder Quality.

P3 — Balanced (Classic)

WO

Inspector → Presets-Sektion → Gruppe „Classic„ → Eintrag „Balanced“. UUID-Suffix `…005`.

TECHNISCH

20 000 Iterationen Classic-Densification bei voller Bildauflösung. Densification läuft über die ersten 15 000 Iterationen, ab Iter 3 000 mit einem Densify-Intervall von 100. Empirisch der „sweet spot„ aus den dokumentierten Trainings-Sessions: bei klassischer Densification auf Horse Full und Truck stabilisiert sich der L1-Loss zwischen Iter 18 000 und 22 000, ein längeres Training bringt unterhalb von Quality (P4) keine signifikante Verbesserung mehr. Auf einem M3 Ultra typischerweise 30–60 Sekunden auf 200 Bildern, 5–8 Minuten auf 1 000+ Bildern.

EINFACH GESAGT

Der „gute Kompromiss„. Die meisten Szenen sehen hiermit schon gut aus, ohne dass du eine Stunde warten musst. Wenn du das Endergebnis irgendwo zeigen willst (Social Media, Website, eine Kunden-Demo), reicht das oft. Erst wenn du in das Splat-Modell hineinzoomen willst oder du Details der Oberflächentextur brauchst, lohnt sich der Sprung zu P4 Quality oder P7 MCMC.

P4 — Quality (Classic)

WO

Inspector → Presets-Sektion → Gruppe „Classic„ → Eintrag „Quality“. UUID-Suffix `…003`.

TECHNISCH

35 000 Iterationen Classic-Densification mit V546-„Opacity Decay„ (HTGS, Eurographics 2025): nach jedem Densify-Cycle multipliziert sich die Opacity aller existierenden Gaussians mit einem Faktor < 1.0, was inaktiv gewordene Gaussians beim Pruning verlässlich entfernt und damit auf identischer Iter-Zahl 14 % besseren L1-Loss als der klassische 35 000-Lauf erreicht. SSIM-Loss ist aktiviert (`ssimWeight=0.05`). Auf einem M3 Ultra typischerweise 2–4 Minuten auf 200 Bildern. Liefert auf NeRF-Blender (Lego, Chair, Drums) finale L1 ≈ 0.023 — beste Classic-Variante in den 560+ dokumentierten Experimenten. Beachte: braucht ~3–5 GB GPU-Speicher; auf 8-GB-Systemen ist P3 die sichere Wahl.

EINFACH GESAGT

Die beste klassische Variante. Liefert scharfe Textur und feine Geometrie, gerade auf Objekt-Aufnahmen (eine Skulptur, ein Stuhl, eine Vase). Bei großen Outdoor-Szenen oder Räumen merkst du dagegen kaum noch einen Unterschied zu Balanced — dort lohnt sich der Wechsel zu einem MCMC-Preset (P6–P8) oder einem Capture-Class-Preset (P9–P12) mehr als der Sprung von P3 auf P4. Wer das absolute Maximum der Classic-Familie will, nimmt P5 Ultra Detail.

P5 — Ultra Detail (Classic)

WO

Inspector → Presets-Sektion → Gruppe „Classic„ → Eintrag „Ultra Detail“. UUID-Suffix `…008`.

TECHNISCH

Rund 35 000 Iterationen Classic-Densification — der Sieger des Held-out-Durchgangs der Quality-Matrix (2026-06-10). Auf allen drei getesteten Mip-NeRF-360-Szenen schlägt Ultra Detail das eingebaute MCMC-„Quality„-Preset (P8) bei vergleichbarer Wall-Clock-Zeit um rund +0.94 dB PSNR. Damit ist es das stärkste Quality-Preset der Classic-Gruppe und die schärfste Classic-Variante, die RadianceKit ausliefert. Auf einem M3 Ultra typischerweise im selben Zeitrahmen wie P4 Quality (2–5 Minuten auf 200 Bildern), braucht aber etwas mehr GPU-Speicher; auf 8-GB-Systemen bleibt P3 die sichere Wahl.

EINFACH GESAGT

Die schärfste Classic-Stufe und der Held-out-Sieger unserer Qualitäts-Tests: auf realen Szenen rund ein Dezibel besser als die MCMC-„Quality„-Variante — bei ähnlicher Wartezeit. Wenn du maximale Detailtreue mit der bewährten klassischen Densification willst und genug GPU-Speicher hast, ist das die erste Wahl. Reicht der Speicher nicht oder brauchst du eine möglichst kleine Export-Datei, bleib bei P4 Quality oder einem MCMC-Preset.

P6 — Preview (MCMC)

WO

Inspector → Presets-Sektion → Gruppe „MCMC„ → Eintrag „Preview“. UUID-Suffix `…006`.

TECHNISCH

60 000 Iterationen MCMC-Densification (3DGS-MCMC, NeurIPS 2024) bei einem Cap von 100 000 Gaussians. MCMC ersetzt die heuristische Klon/Split-Logik durch Markov-Chain-Monte-Carlo-Relocation: tote Gaussians werden über Sigmoid-gewichtete Sampling-Tiefen neu plaziert, was eine kontrollierte und reproduzierbare Anzahl Gaussians ergibt. Der Cap deckelt die maximale Anzahl hart bei 100K — das spart Speicher und Render-Zeit, kostet aber Detail. Auf einem M3 Ultra typischerweise 5–8 Minuten auf 200 Bildern. Eignet sich als „MCMC-Funktionstest„ — Hilft zu beurteilen, ob ein Wechsel von Classic auf MCMC sinnvoll wäre, bevor du in P7 oder P8 mehr Zeit investierst.

EINFACH GESAGT

Wie P2 Preview, aber mit dem neueren MCMC-Verfahren. Liefert oft etwas kompaktere, gleichmäßiger verteilte Splat-Wolken als die Classic-Variante. Für eine erste Begutachtung einer Szene reichen die 5–8 Minuten meist. Wenn dir das Vorschau-Ergebnis gefällt, ist der nächste Schritt P7 (Balanced) oder direkt P8 (Quality MCMC).

P7 — Balanced (MCMC)

WO

Inspector → Presets-Sektion → Gruppe „MCMC„ → Eintrag „Balanced“. UUID-Suffix `…007`.

TECHNISCH

120 000 Iterationen MCMC bei einem Cap von 150 000 Gaussians. Die mittlere MCMC-Stufe — fast die finale Gaussian-Anzahl von P8 Quality, aber nur 60 % der Iterationen. Empirisch liegt der L1-Loss in den dokumentierten Trainings-Sessions bei 0.026–0.028 auf Horse Full, gegenüber P8 mit 0.0246 — also rund 7 % höher, dafür halbe Wartezeit. Auf einem M3 Ultra typischerweise 8–15 Minuten auf 200 Bildern. Verwendet ein Verfahren, das den effektiven Gaussian-Cap an die Punktdichte der Eingangs-SfM-Punktwolke skaliert (siehe T75 in Kapitel 6).

EINFACH GESAGT

MCMC mit ordentlicher Detailtiefe, aber ohne den langen Voll-Lauf von P8. Für die meisten Szenen reicht das aus, gerade wenn du einen MCMC-Lauf in den Mittagspausen-Zeitrahmen pressen willst. Wenn Speicher knapp ist (etwa auf M-Prozessoren mit nur 16 GB), bleib hier — P8 braucht mehr GPU-Speicher.

P8 — Quality (MCMC)

WO

Inspector → Presets-Sektion → Gruppe „MCMC„ → Eintrag „Quality“. UUID-Suffix `…004`.

TECHNISCH

200 000 Iterationen MCMC bei einem Cap von 150 000 Gaussians, SSIM-Loss 0.05, MCMC-Noise-Decay über 80 % der Iterationen. Best-Single-Run-L1 in den 560+-Experimenten: 0.0238 auf Horse Full, gemittelt über 3 Trials 0.0246 (gegenüber P4 0.0230 auf gleicher Szene). MCMC liefert dabei 71 % weniger Gaussians (150K vs ~524K) — entscheidend, wenn du das Ergebnis auf dem Web ausliefern willst, weil die kleinere Wolke deutlich kleinere Export-Dateien produziert. Trainingszeit auf einem M3 Ultra typischerweise 20–35 Minuten auf 200 Bildern; auf 1 000+-Bild-Sets eher 1–2 Stunden. Beste Wahl wenn maximale Bildqualität bei minimaler Endgröße gewünscht ist.

EINFACH GESAGT

Die beste MCMC-Variante. Liefert sehr saubere, kompakte Splat-Wolken — ideal wenn du das Ergebnis später als Web-3D-Viewer einbetten oder als Datei verschicken willst (die Datei ist kleiner als bei P4 Quality bei vergleichbarer optischer Qualität). Brauchst dafür aber Geduld — auf großen Aufnahmen über eine Stunde Wartezeit. Plane das eher als „Schnitt-über-Nacht„-Lauf.

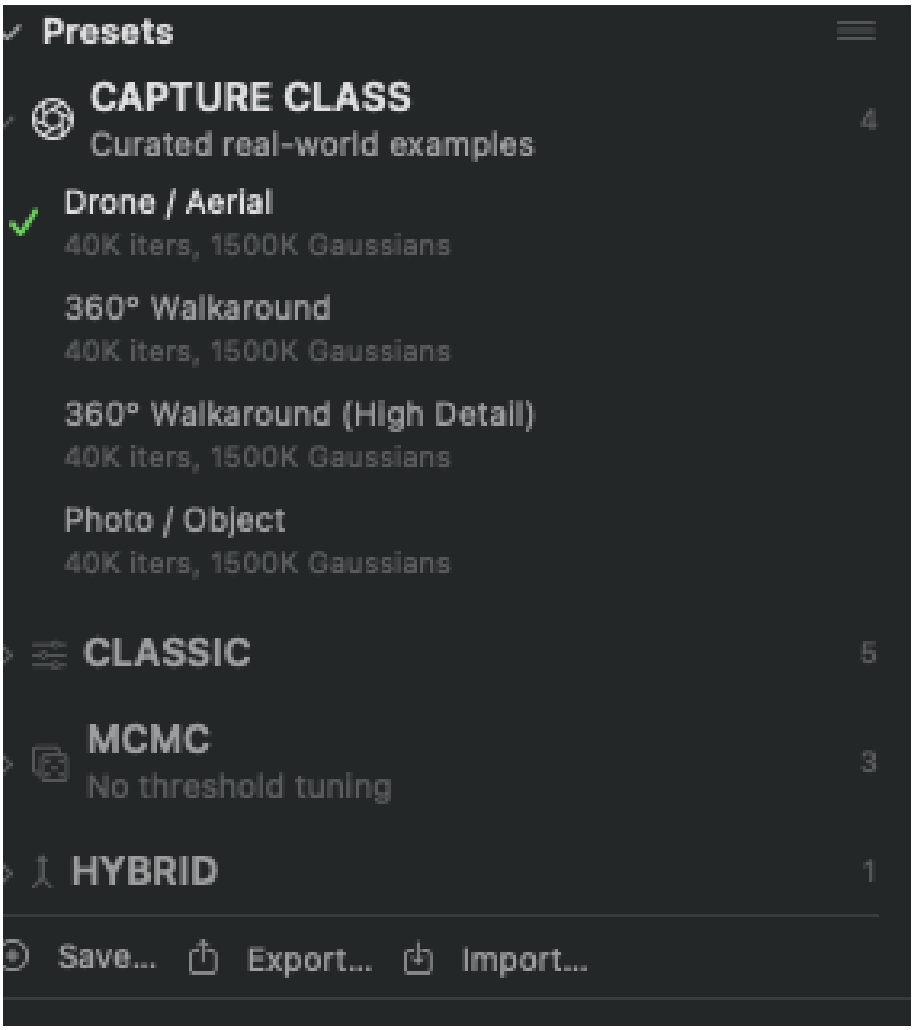

Abbildung 28: CAPTURE CLASS Gruppe ausgeklappt mit allen vier kuratierten Real-World-Presets — Drone / Aerial (MCMC, 40K iters, Cap 1,5 M), 360° Walkaround (MCMC, 40K, Cap 1,5 M), 360° Walkaround (High Detail) (Hybrid, 40K, Cap 1,5 M, opt-in) und Photo / Object (Hybrid, 40K, Cap 1,5 M). Diese Gruppe steht ganz oben und ist im Simple-Mode die einzige sichtbare.

WAS IM BILD ZU SEHEN IST Inspector mit der CAPTURE CLASS-Gruppe ausgeklappt — die primäre Preset-Gruppe seit v1.6, im Simple-Mode die einzige angezeigte. Jeder Eintrag ist ein an echtem Community-Material per Auge validiertes Rezept für eine konkrete Aufnahme-Art (Drohne, 360°-Rundgang, Foto-Objekt), nicht ein gegen ein akademisches Test-Set optimierter Wert. Auswahl per Klick schreibt die hinterlegte Trainingskonfiguration in den aktuellen Zustand.

P9 — Drone / Aerial

WO

Inspector → Presets-Sektion → Gruppe „Capture Class„ → Eintrag „Drone / Aerial“. UUID-Suffix `…010`.

TECHNISCH

Capture-Class-Preset für Luft- und Drohnen-Orbits von Gebäuden und Landschaften. MCMC-Densifier, 40 000 Iterationen, Cap 1,5 Mio. Gaussians, SSIM-Loss 0.5 plus Edge-Aware-Term 0.1. Entscheidend ist die Anisotropie-Strafe mit Gewicht 0.003 bei einer Ratio-Schwelle von 6 — der „Spaghetti-Killer„ gegen die typisch nadelförmigen Artefakte, die Drohnen-Footage erzeugt. Validiert an einem echten DJI-4K-Drohnenflug über das Pensford-Viadukt (per Auge geprüft, nicht nur metrisch).

EINFACH GESAGT

Für Aufnahmen aus der Luft — Drohnenflüge um ein Gebäude, über eine Landschaft, entlang einer Fassade. Die kräftige Anisotropie-Strafe räumt die nadel- bzw. spaghettiförmigen Artefakte weg, die Drohnen-Material gern produziert. Wenn dein Material vom Boden aus aufgenommen ist, passt eher Photo / Object oder ein Classic-Preset.

P10 — 360° Walkaround

WO

Inspector → Presets-Sektion → Gruppe „Capture Class„ → Eintrag „360° Walkaround“. UUID-Suffix `…011`.

TECHNISCH

Capture-Class-Preset für 360°-Walkaround-Videos. MCMC-Densifier, 40 000 Iterationen, Cap 1,5 Mio. Gaussians, SSIM-Loss 0.5 plus Edge-Aware-Term 0.1, sanfte Anisotropie-Strafe (Gewicht 0.001 bei Ratio-Schwelle 15). Personen- und Himmel-Maske sind aktiv. Das Preset erwartet ein 360°-Equirect-Video, das intern auf rund 90° breite Perspektiv-Crops reprojiziert wird, bevor das Training startet. Validiert an 8K-360°-Rundgängen mit Selfie-Stick (Monument-Szene, per Auge geprüft).

EINFACH GESAGT

Für 360°-Rundgang-Videos — du gehst mit einer 360°-Kamera oder Selfie-Stick durch einen Raum oder um ein Objekt herum. RadianceKit zerlegt das Kugel-Panorama selbst in normale Blickwinkel und maskiert Passanten und Himmel weg. Für maximale Schärfe auf demselben Material probiere zusätzlich die High-Detail-Variante (P11).

P11 — 360° Walkaround (High Detail)

WO

Inspector → Presets-Sektion → Gruppe „Capture Class„ → Eintrag „360° Walkaround (High Detail)“. UUID-Suffix `…013` (Opt-in).

TECHNISCH

Opt-in-Capture-Class-Preset für 360°-Walkaround-Videos mit maximalem Detail. Hybrid-Densifier (klassisches Abs-Gradienten-Klonen/Splitten

1. MCMC-Noise + Relocation), 40 000 Iterationen, Cap 1,5 Mio. Gaussians,

Anisotropie-Strafe 0.0015 bei Ratio-Schwelle 15, SSIM-Loss 0.2 und Edge-Aware-Term 0 — das gelockte „r50”-Screen-Split-Rezept. Auf 360°-Material schlägt es das Standard-MCMC-Preset „360° Walkaround„ (P10) bei PSNR, LPIPS und sichtbarem Konfetti, und das mit rund einem Drittel der Splat-Anzahl. Steht bewusst opt-in *neben* dem Standard-360-Preset, bis es an mehr Szenen validiert ist.

EINFACH GESAGT

Die schärfere Alternative zum Standard-360-Preset (P10): mehr Detail, weniger Konfetti, deutlich kleinere Datei. Steht bewusst daneben statt es zu ersetzen — bisher an einer Hand voll Szenen bestätigt. Wenn dein 360°-Rundgang sauber aufgenommen ist, probiere dieses Preset zuerst und vergleiche das Ergebnis mit P10.

P12 — Photo / Object

WO

Inspector → Presets-Sektion → Gruppe „Capture Class„ → Eintrag „Photo / Object“. UUID-Suffix `…012`.

TECHNISCH

Capture-Class-Preset für Objekt-Orbits aus scharfen Einzelfotos (kein Video). Hybrid-t1-Densifier (mit Relocation), 40 000 Iterationen, Cap 1,5 Mio. Gaussians, SSIM-Loss 0.5 plus Edge-Aware-Term 0.1, sanfte Anisotropie-Strafe (Gewicht 0.001 bei Ratio-Schwelle 15), Opacity Decay 0.9995 alle 50 Iterationen, **kein** Masking. Validiert an 163 hochauflösenden 41-MP-Fotos eines Skeletts (per Auge geprüft). Wenige Views (bis etwa 600) bleiben dabei unter der Hybrid-Collapse-Schwelle.

EINFACH GESAGT

Für Objekt-Aufnahmen aus scharfen Einzelfotos — du umrundest eine Skulptur, ein Modell, ein Produkt mit der Kamera und machst Fotos statt Video. Kein Masking, weil scharfe Fotos meist sauberen Hintergrund haben. Für Video-Quellen nimm stattdessen ein 360°- oder Drone-Preset.

P13 — Ausgewogen (Hybrid)

WO

Inspector → Presets-Sektion → Gruppe „Hybrid„ → Eintrag „Ausgewogen (Hybrid)“. UUID-Suffix `…009`.

TECHNISCH

20 000 Iterationen mit der Hybrid-Densification-Strategie: klassisches gradientengetriebenes Klonen/Splitten platziert Kapazität dort, wo der Loss sie braucht, MCMC-SGLD-Noise exploriert weiter, und tote Gaussians werden reloziert, statt beim Pruning verloren zu gehen. Opacity Decay (V546) ersetzt Opacity-Resets; eine Anisotropie-Strafe (Gewicht 0.001, Ratio-Schwelle 15) hält nadelförmige Splats in Schach. Der Gaussian-Cap skaliert mit der Szene (150K Basis, scene-aware ×3.0). Auf fünf Szenen gegen reines MCMC bei gleichem Budget validiert: im Schnitt +0.45 dB PSNR bei 20–30 % weniger Gaussians (stonehenge +1.23, family +0.82, garden +0.47 dB). Auf einem M3 Ultra typisch 5–10 Minuten auf 200 Bildern.

EINFACH GESAGT

Eine starke erste Wahl für ein finales Ergebnis: schärfere Details als die MCMC-Presets bei ähnlich kompakter Datei, in einem Bruchteil der P8-Trainingszeit. Wenn du nur Zeit für einen Quality-Lauf hast und keine der Capture-Klassen klar passt, starte hier. Die Classic-Presets bleiben für schnelle Tests besser, und die Capture-Class-Presets (P9—P12) sind erste Wahl, wenn deine Szene klar zu einer dieser Aufnahme-Arten passt.

Wann welches Preset?

Szenario	Erst-Test	Hauptlauf
Funktionstest neuer Bilder, < 30s	**P1 Quick**	—
Objekt-Orbit aus scharfen Einzelfotos	P2 Preview	**P12 Photo / Object**
Einzeln-Objekt-Scan (Video), < 500 Fotos	P2 Preview	**P4 Quality** oder **P8 Quality MCMC**
360°-Walkaround-Video	P6 Preview MCMC	**P10 360° Walkaround** (scharf: **P11 High Detail**)
Luft-/Drohnen-Orbit, Landschaft	P6 Preview MCMC	**P9 Drone / Aerial**
Web-Auslieferung (klein, kompakt)	P2	**P8 Quality MCMC** (kleinste Datei bei voller Qualität)
Scharfe Details in wenig Zeit, kompakter Export	P2 oder P6	**P13 Ausgewogen (Hybrid)**
Maximale Detailtreue, Classic-Strategie	P3 oder P6	**P5 Ultra Detail**
Print, Marketing, voller Detail	P3 oder P6	**P4 Quality (Classic)** oder **P5 Ultra Detail**

Schnell-Vergleich

Preset	Strategie	Iters	Max-Gs	Render-Skala	Typische Zeit (200 Bilder, M3 Ultra)	Q-Sweep
P1 Quick	Classic	1 000	∞	0.25×	~30 s	—
P2 Preview	Classic	5 000	∞	0.5×	2–3 min	—
P3 Balanced	Classic	20 000	∞	1.0×	30–60 s	—
P4 Quality	Classic	35 000	∞	1.0×	2–4 min	V546 HTGS
P5 Ultra Detail	Classic	~35 000	∞	1.0×	2–5 min	Matrix Δ+0.94 dB
P6 Preview MCMC	MCMC	60 000	100 K	1.0×	5–8 min	—
P7 Balanced MCMC	MCMC	120 000	150 K	1.0×	8–15 min	—
P8 Quality MCMC	MCMC	200 000	150 K	1.0×	20–35 min	V544a
P9 Drone / Aerial	MCMC	40 000	1.5 M	1.0×	10–25 min	Auge / Viadukt
P10 360° Walkaround	MCMC	40 000	1.5 M	1.0×	10–25 min	Auge / Monument
P11 360° Walkaround (High Detail)	Hybrid	40 000	1.5 M	1.0×	10–25 min	Auge (opt-in)
P12 Photo / Object	Hybrid	40 000	1.5 M	1.0×	10–25 min	Auge / Skelett
P13 Ausgewogen (Hybrid)	Hybrid	20 000	150 K	1.0×	5–10 min	Matrix Δ+0.45 dB

Eigene Presets

Über den Button **Save…** in der Presets-Sektion (I1 in Kapitel 2) speicherst du die aktuelle Trainingskonfiguration als eigenes Preset. Eigene Presets sind nicht „Built-in„ und lassen sich umbenennen, exportieren (als JSON), per Drag-and-Drop teilen, duplizieren und löschen. Die dreizehn eingebauten Presets P1–P13 bleiben vom Lösch-Button unberührt.

Faustregel: Wenn du an einem Preset etwas änderst, das du noch öfter brauchen willst — Sky-Dome an, höheres SSIM-Gewicht für eine bestimmte Szenen-Klasse, abweichende Iter-Zahlen — dann speichere die Variante als eigenes Preset. So weißt du beim nächsten Lauf gleich, dass es eine vom Standard abweichende Konfiguration ist.

KAPITEL

Kapitel 8 — Export-Formate

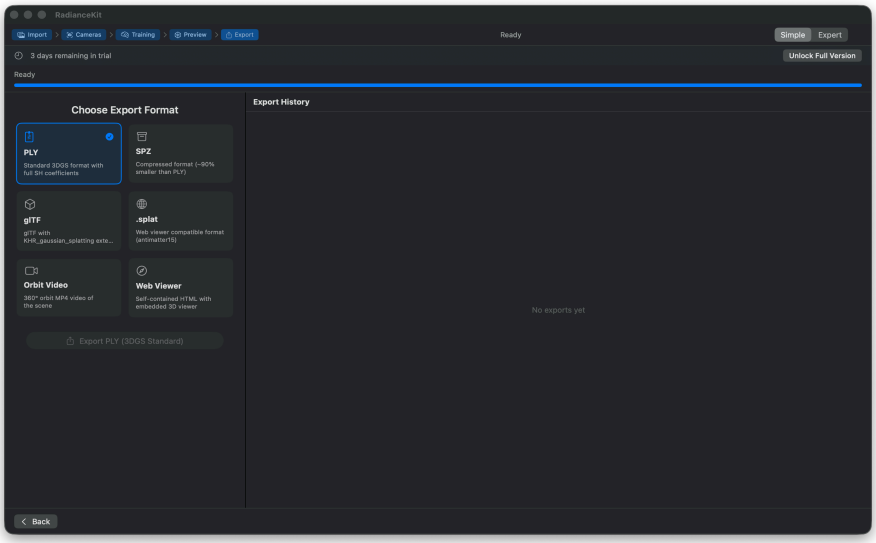

Abbildung 29: Export-Format-Auswahl im Simple-Modus — sechs Format-Karten

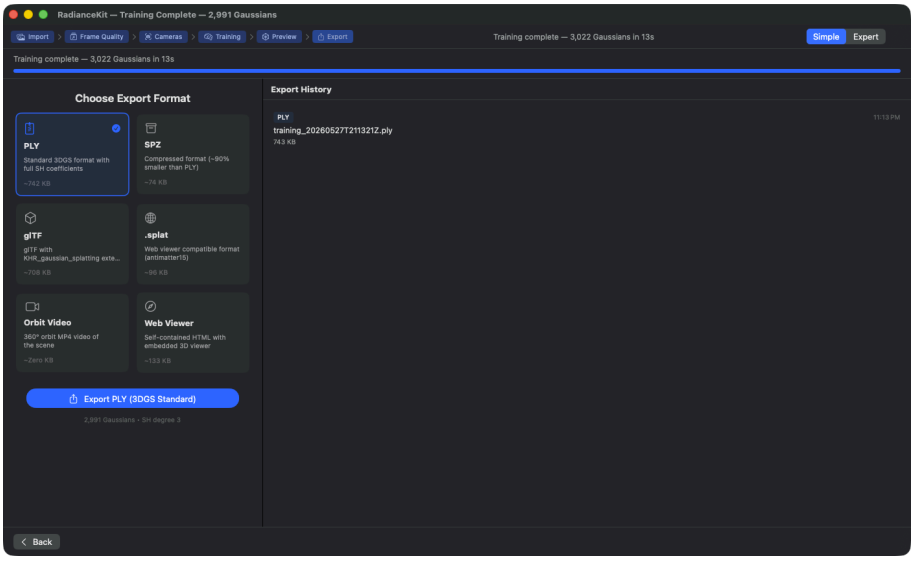

Abbildung 30: Export-Format-Grid live nach 5K-Iter-Training auf flowers-Bouquet — alle sechs Karten mit dynamischer Größenberechnung (PLY 742 KB ausgewählt, SPZ 74 KB, glTF 708 KB, .splat 96 KB, Orbit Video ~Zero KB, Web Viewer 133 KB), Export History rechts mit bereits gespeichertem PLY

Was im Bild zu sehen ist (2 991 Gaussians, SH degree 3, Bjoerns synthetisches Blender-Bouquet als IP-clean Test-Set): Die Größenangaben unter jeder Format-

Karte werden live aus aktuellem Gaussian-Count und Format-Overhead berechnet — nicht hartcodiert. Aus 2 991 Gaussians (SH degree 3) entstehen so 742 KB PLY, 74 KB SPZ (Faktor ~10× kleiner durch Quantisierung), 708 KB glTF (mit `KHR_gaussian_splatting`-Extension, daher fast PLY-äquivalent), 96 KB .splat (komprimiertes 24-Byte-pro-Gaussian-Format). Orbit Video zeigt „~Zero KB„, weil die Größe erst nach dem MP4-Encoding bekannt ist. Web Viewer (133 KB) bündelt eine eigenständige HTML-Datei mit eingebettetem WebGL-Viewer und komprimierten Splat-Daten — größer als reines .splat wegen Viewer-Overhead. Export-History rechts listet bereits abgeschlossenen PLY-Export („training_20260527T211321Z.ply, 743 KB, 23:13“) mit Format-Pill und Reveal-im-Finder-Action.

Ein abgeschlossenes Training liefert eine Gaussian-Cloud — eine Sammlung von ein paar Hunderttausend bis Millionen 3D-Gaussverteilungen, die zusammen die Szene rekonstruieren. RadianceKit kennt zehn Wege, diese Cloud auf die Festplatte zu schreiben. Sechs davon sind reine 3D-Daten-Formate (PLY, Compressed PLY, SPZ, SOG, glTF, .splat), eines bündelt die Cloud zusammen mit einem fertigen HTML-Viewer (Web Viewer), eines rendert eine MP4-Datei aus einer Orbit-Kamerafahrt (Orbit Video), und zwei exportieren keinen Gaussian-Inhalt sondern lediglich das SfM-Ergebnis (Kamera-Posen und grobe Punkt-Wolke) zur Wiederverwendung in anderen Trainings-Pipelines (transforms.json + COLMAP-Workspace).

Welches Format wann der richtige ist, hängt vom Ziel ab. Für die Archivierung der vollen Daten ohne Qualitätsverlust nimmt man PLY. Für Web-Viewer auf der eigenen Seite reicht meist .splat oder der eingebaute Web-Viewer. Wenn die Datei minimal sein muss, lohnt sich SPZ oder SOG. Für die Wiederverwendung des SfM-Resultats in Nerfstudio, Postshot oder Brush sind transforms.json und der COLMAP-Workspace die richtigen Wege.

Alle Export-Funktionen liegen im Menü „Export„ sowie im Simple-Mode auf der letzten Wizard-Stufe. Die meisten Formate sind vollständig sandbox-konform und funktionieren in der App-Store-Version. Nur SOG erfordert eine externe Binary (`cwebp`), die im App-Store-Build nicht zwingend vorhanden ist — Details siehe E4.

E1 — PLY (.ply)

WO

Menüleiste → Export → 3D Formats → Export PLY… (⌘E). Simple-Mode: Wizard-Schritt Export → Format-Karte „PLY„. **Größe:** typisch 100 % (Referenzwert). **Kompatibel mit:** SuperSplat, PolyCam, alle 3DGS-Viewer.

TECHNISCH

PLY ist das kanonische Speicherformat für 3D Gaussian Splatting. RadianceKit schreibt eine binäre Little-Endian-Datei mit dem standardisierten 3DGS-Property-Layout: pro Gaussian dreikomponentige Position, drei stets auf Null gesetzte Normalen, drei DC-SH-Koeffizienten (`f_dc_0..2`) für die Basis-RGB-Farbe, anschließend bis zu 45 weitere SH-Koeffizienten (`f_rest_0..44`) in der vom Kerbl-2023-Paper definierten transponierten Channel-Major-Anordnung (erst alle R-Kanal-Koeffizienten, dann alle G, dann alle B), gefolgt von Logit-Opazität (rohe pre-Sigmoid-Werte), drei Log-Space-Skalen und einer wxyz-Quaternion-Rotation. Der maximal exportierte SH-Grad wird auf das Minimum aus User-Wunsch und tatsächlich gelerntem Grad geclampt; Default ist 3 (45 Rest-Koeffizienten). Vor dem Schreiben wird die Payload-Größe in 64-Bit-Integer berechnet, um Überlauf bei extrem großen Clouds zu fangen. Die Datei wird atomar geschrieben, was bei großen Clouds kurzzeitig den doppelten Plattenspeicher belegt.

EINFACH GESAGT

Das ist die „Originaldatei„. Größte Datei, höchste Kompatibilität, keine Verluste. Wenn du nicht weißt, welches Format du nehmen sollst, nimm PLY — das öffnet sich in fast jedem 3DGS-Tool. Für 1 Million Gaussians sind das je nach SH-Grad zwischen 200 und 800 MB. Wenn die Datei zu groß wird, schau dir E2 (komprimiertes PLY) oder E3 (SPZ) an.

E2 — Compressed PLY (.ply)

WO

Menüleiste → Export → 3D Formats → Export Compressed PLY…. Simple-Mode: Format-Karte „Compressed PLY„. **Größe:** ca. 10–20 % gegenüber PLY (5- bis 10-fache Kompression). **Kompatibel mit:** SuperSplat, PlayCanvas-Engine, web-basierte Viewer.

TECHNISCH

Die PlayCanvas-Variante des PLY-Formats mit chunked Quantisierung. Die Gaussians werden in 256er-Chunks gruppiert. Pro Chunk werden Min/Max-Bounds für Position, Scale und Color separat im Header abgelegt; die einzelnen Gaussians referenzieren ihre Werte relativ zu diesen Bounds und werden auf je 32 Bit komprimiert: Position und Skala mit 11-10-11-Bit-Packing, Rotation als 2-10-10-10-Bit „Smallest-Three„-Quaternion, Farbe als 8-8-8-8-RGBA. Höhere SH-Koeffizienten werden mit nur 8 Bit pro Komponente quantisiert (`shCoeffCount * 3` uchar pro Gaussian). Das Format selbst ist immer noch ASCII-Header-PLY und damit grundsätzlich validierbar mit PLY-Tools, aber die Vertex-Properties sind als `uint` -Felder deklariert. SH-Grad ist per Default 0 (keine Rest-Koeffizienten), um die Kompression zu maximieren — höhere SH-Grade können explizit gewählt werden.

EINFACH GESAGT

Die platzsparende PLY-Variante. Identische Engine-Kompatibilität wie das normale PLY, aber 5- bis 10-mal kleiner. SuperSplat und PlayCanvas lesen es nativ. Für Web-Deployment fast immer besser als normales PLY. Der Qualitätsverlust durch die Quantisierung ist visuell meist nicht wahrnehmbar, solange die Szene nicht extrem hochfrequente Details enthält.

E3 — SPZ (.spz)

WO

Menüleiste → Export → 3D Formats → Export SPZ…. Simple-Mode: Format-Karte „SPZ„. **Größe:** ca. 10 % gegenüber PLY (90 % kleiner). **Kompatibel mit:** Niantic Scaniverse, Niantic Spatial Fields, MetalSplatter.

TECHNISCH

Niantics SPZ-v2-Format. Positionen werden als 24-Bit-Fixed-Point gepackt (das ergibt ca. 0,25 mm Auflösung), Skalen als 8-Bit-Quantisierung im Log-Raum, Rotationen als 8-Bit-Smallest-Three (in v2 werden nur xyz gespeichert, w wird im Decoder aus der Quaternion-Norm abgeleitet), Opazitäten als sigmoidisierte 8-Bit-Werte. DC-SH wird mit einer SPZ-spezifischen Pack-Formel (`dc_raw * 0.15 * 255 + 0.5 * 255`) gespeichert, höhere SH-Bands mit 5 Bit (Band 1) bzw. 4 Bit (Band 2-3) pro Koeffizient. Der gesamte gepackte Binärblob wird anschließend mit Standard-gzip (RFC 1952) komprimiert, was ein gzipped-Container-Format mit Magic Bytes `1f 8b` ergibt. RadianceKit ruft hierfür das System-`gzip` auf, weil Apples eingebaute zlib-API proprietäres Apple-Framing erzeugt, das nicht kompatibel zu den SPZ-Readern in Spatial Fields oder MetalSplatter wäre. Das System-`gzip` bleibt auch innerhalb der macOS-Sandbox spawnbar.

EINFACH GESAGT

Die kleinste Standard-Datei. Wenn du Scaniverse von Niantic kennst — das ist das Format, das die App benutzt. Sehr klein, sehr ladefreundlich für mobile Apps. In Niantics eigenem Cloud-Viewer (Spatial Fields) direkt nutzbar. Etwa 90 % kleiner als ein PLY mit denselben Daten, dabei für die meisten Szenen optisch kaum unterscheidbar.

E4 — SOG (.sog)

WO

Menüleiste → Export → 3D Formats → Export SOG…. Simple-Mode: Format-Karte „SOG„. **Größe:** ca. 5–6 % gegenüber PLY (15- bis 20-fache Kompression — die kleinste Option). **Kompatibel mit:** PlayCanvas-Engine, SuperSplat-Editor.

TECHNISCH

„Spatially Ordered Gaussians„ — ein PlayCanvas-Format, das die Cloud GPU-ready in mehreren Lossless-WebP-Bildern speichert. Erst werden alle Gaussians per 3D-Morton-Code (30-Bit Z-Order, je 10 Bit pro Achse) räumlich sortiert, was den Bildern spätere Cache-Locality im Renderer beschert. Dann werden Positionen mit symmetrischer Log-Transformation (für besseren Dynamikumfang) auf 16-Bit-Werte quantisiert und in zwei RGBA-Bilder gesplittet (`means_l.webp` für die unteren 8 Bit, `means_u.webp` für die oberen). Rotationen werden als Smallest-Three mit 3×8-Bit plus 2-Bit-Mode in einem RGBA-Bild kodiert (Mode landet in Alpha als `252 + largest`). Skalen und DC-SH werden mit je einem 256-Eintrag-Codebook quantisiert (Perzentil-basiert über alle Werte verteilt), die Indizes landen in `scales.webp` und `sh0.webp` . Die fünf Bilder plus eine `meta.json` mit Codebooks und Bounds werden in eine ZIP-Datei gepackt (Custom-Encoder, weil die Sandbox das System- `zip` blockiert) und mit der Endung `.sog` gespeichert.

Achtung Sandbox: SOG ist die einzige Format-Option, die eine externe Binary erfordert. Die WebP-Encoder-Stufe ruft `cwebp` aus `/usr/local/bin/cwebp` oder `/opt/homebrew/bin/cwebp` auf. Falls keine cwebp-Binary gefunden wird, fällt der Code auf rohes PNG-Encoding zurück — aber: **PNG-Fallback funktioniert nicht in SuperSplat**. In der App-Store-Version evaluiere die Verfügbarkeit anhand der Build-Variante; in der Developer-Variante muss cwebp via Homebrew installiert sein (`brew install webp`).

EINFACH GESAGT

Das kleinste 3DGS-Format überhaupt, deutlich kleiner als SPZ. Aber: braucht das `cwebp` -Tool auf deinem Mac, weil RadianceKit selbst nicht alle Bildformate erzeugen kann. Installier es einmal mit Homebrew, dann läuft alles. In der App-Store-Version eventuell nicht voll funktionsfähig — wenn beim Export PNG statt WebP rauskommt, kannst du die Datei nicht direkt in SuperSplat öffnen. Wer ohne Homebrew arbeitet, nimmt stattdessen SPZ (E3).

E5 — glTF (.glb)

WO

Menüleiste → Export → 3D Formats → Export glTF.... Simple-Mode: Format-Karte „glTF„. **Größe:** vergleichbar mit PLY. **Kompatibel mit:** glTF-Viewer mit `KHR_gaussian_splatting`-Extension (Khronos-Draft-Standard).

TECHNISCH

Schreibt eine selbsterhaltende `.glb`-Binärdatei (kein separater Bin-File-Anhang) gemäß der `KHR_gaussian_splatting`-Extension-Spezifikation. Positionen werden als reguläre glTF-`POSITION`-Vertex-Daten (float3) gespeichert, alle anderen Attribute (Rotation als float4, Scale als float3, Opacity als float, SH-Koeffizienten als float3 × shCoeff-Count) liegen in zusätzlichen Vertex-Attributen und werden über die Extension referenziert. Wichtig: glTF benutzt rechtshändiges Y-up-Koordinatensystem, COLMAP/3DGS arbeitet Y-down/Z-forward. Der Exporter wendet daher eine 180-Grad-Drehung um die X-Achse an — Positionen werden mit `(x, −y, −z)` umgeschrieben, Quaternionen werden auf `(w, x, −y, −z)` angepasst. Das ergibt eine geometrisch korrekte, händige (nicht spiegelverkehrte) Darstellung in glTF-Viewern. JSON- und Binärchunks werden auf 4-Byte-Alignment gepaddet, wie vom GLB-Standard verlangt.

EINFACH GESAGT

Das offizielle Khronos-Standardformat für 3D-Daten, in der frischen Erweiterung für Gaussian Splats. Vorteil: glTF ist in allen großen 3D-Engines verbreitet (Babylon.js, Three.js, Unity, Unreal). Nachteil: Die Erweiterung ist 2026 noch im Draft-Stadium, viele Viewer können sie noch nicht. Sinnvoll vor allem, wenn du Splat-Daten in eine bestehende glTF-Pipeline integrierst oder einen Viewer schreibst, der bereits glTF-fähig ist.

E6 — Splat (.splat)

WO

Menüleiste → Export → 3D Formats → Export .splat…. Simple-Mode: Format-Karte „.splat„. **Größe:** exakt 32 Bytes pro Gaussian. **Kompatibel mit:** gsplat.js, web-basierte Viewer (antimatter15-Referenz), die meisten Browser-3DGS-Demos.

TECHNISCH

Das antimatter15-`.splat`-Format — 32 Bytes pro Gaussian, kein Header, keine Indirektion. Layout pro Eintrag: 3 × float32 Position (Welt-Koordinaten), 3 × float32 Scale (exp-transformiert aus dem Log-Space des internen Buffers), 4 × uint8 RGBA-Farbe (DC-SH-Koeffizient mit `SH_C0 = 0.282...` skaliert und auf [0,255] geclampt), 4 × uint8 Quaternion (w,x,y,z, normalisiert und als `128 + 128*q` in den Byte-Bereich kodiert). Nur DC-SH wird gespeichert — höhere SH-Bänder werden verworfen. Das macht das Format extrem kompakt, kostet aber die View-abhängigen Farbänderungen, die bei Spiegelungen oder spekularen Highlights auftreten. Die Schreibreihenfolge ist exakt die Index-Reihenfolge der Cloud (keine räumliche Sortierung), Web-Viewer wie `gsplat.js` rendern davon ausgehend.

EINFACH GESAGT

Das Format der Wahl, wenn du den Splat in einem eigenen Web-Viewer mit `gsplat.js` darstellen willst. Sehr kompakt (32 Bytes/Gaussian), aber kein höherer SH-Grad — also keine glänzenden Reflexionen oder feinen Farbveränderungen je nach Blickwinkel. Für die meisten Web-Anwendungen kein Problem, weil DC-Farbe völlig ausreicht und die fehlende View-Abhängigkeit kaum auffällt.

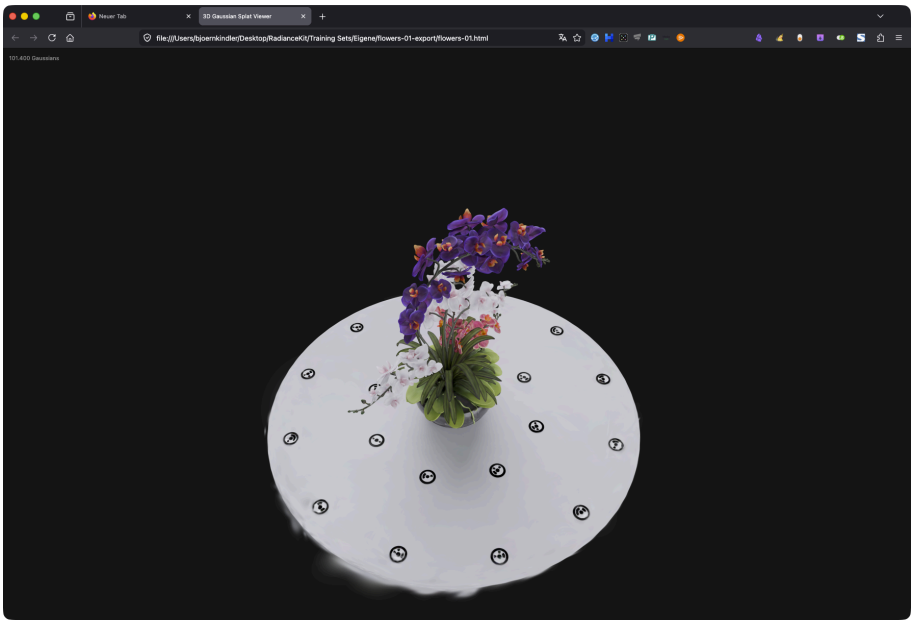

Abbildung 31: Web Viewer geöffnet im Firefox — Bjoerns Bouquet-Splat gerendert mit umgebenden Kamera-Marker-Sphären, Browser-Tab-Bar oben sichtbar, kein CDN-/Server-Setup nötig. Eigenständige `flowers-01.html` *direkt aus dem Finder per Doppelklick im Default-Browser geöffnet — das eingebettete WebGL2-Programm rendert die Gaussian-Cloud sofort, ohne Netz oder Server. Die schwarzen Marker um das Bouquet sind die Trainings-Kameras, optional einblendbar. Maus-Drag rotiert, Scroll zoomt.*

E7 — Web Viewer (.html)

WO

Menüleiste → Export → Media → Export Web Viewer…. Simple-Mode: Format-Karte „Web Viewer„. **Größe:** Splat-Daten base64-kodiert (≈ 4/3 Overhead) + ca. 5 KB HTML/JS-Shell. **Kompatibel mit:** jeder moderne Browser mit WebGL2 (alle Desktops, iOS 15+, Android 5+).

TECHNISCH

Bündelt die Gaussian-Cloud zusammen mit einem vollständig inline geschriebenen WebGL2-Renderer in eine einzelne `.html`-Datei. Es gibt keine CDN-Abhängigkeiten, kein WASM, kein zweites File. Die Cloud wird intern erst als `.splat`-Binary kodiert (gleiche 32-Byte-Logik wie E6), dann base64-eingebettet, dann mit `atob` im Browser dekodiert. Der eingebaute Renderer macht eigene WebGL2-Sortierung, Maus-Orbit-Steuerung und CPU-Sortierung pro Frame; der gesamte JS-Code (Shader, Mathematik, Loop) ist im Output-HTML zu sehen. Die Achsen-Konvention an der Speicher-zu-Renderer-Grenze ist exakt dieselbe wie in E5: Position `(x, −y, −z)`, Quaternion `(w, x, −y, −z)`. Optional kann ein Branding-Overlay eingeblendet werden (Free-Tier-Schalter). Da alles inline ist, funktioniert die Datei auch direkt aus dem `file://`-Protokoll — kein lokaler Webserver nötig zum Testen.

EINFACH GESAGT

Eine einzige HTML-Datei, die du jemandem per Mail schicken oder auf einer Webseite einbinden kannst. Doppelklick im Finder, und der Browser zeigt deine Szene mit Maus-Drehung. Kein Upload zu einer Cloud nötig, keine zweite Datei, kein Server. Ideal für Kundenpräsentationen, Portfolio, Mail-Anhänge. Nachteil: die Datei wird durch die base64-Kodierung etwa ein Drittel größer als ein reines `.splat` — für sehr große Szenen lohnt sich daher das separate Hosting der `.splat`-Datei zusammen mit einem Standard-Viewer.

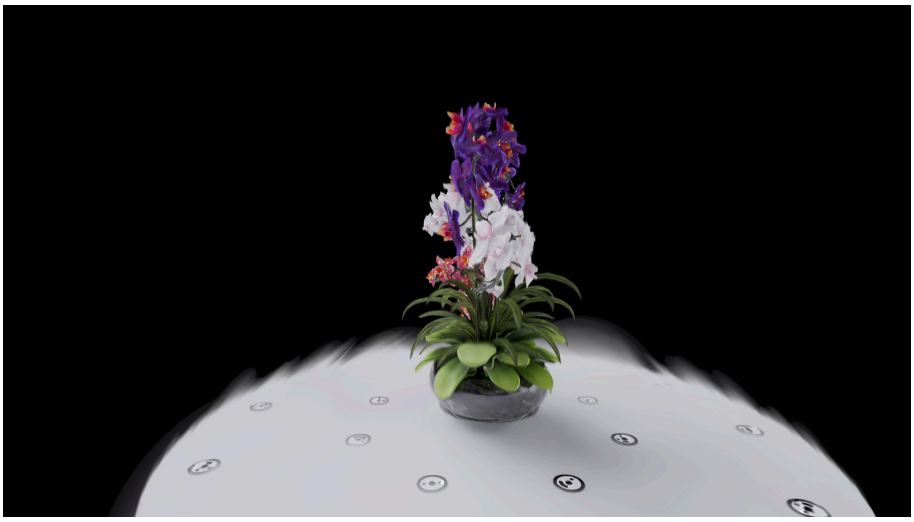

Abbildung 32: Einzelframe extrahiert aus flowers-01.mp4 — Bjoerns Bouquet im Profil-Render, weiße Plattform mit Kamera-Markern sichtbar, schwarzer Hintergrund (Default-Viewport-Background, in Settings änderbar). Die Kamera umkreist die Szene auf einer parametrischen Bahn (Elevation + Distanz fest, Yaw rotiert), Dauer typisch 6–10 Sekunden bei 30 oder 60 fps. Frame-Auflösung skalierbar von 480p bis 8K via VideoPreset.

E8 — Orbit Video (.mp4/.mov)

WO

Menüleiste → Viewport → Record Turntable Video ODER Menüleiste → Export → Media → Export Orbit Video…. Simple-Mode: Format-Karte „Orbit Video„ mit Dauer-Slider 3–30 s. **Größe:** abhängig von Dauer, Auflösung, Bitrate. **Kompatibel mit:** alle Plattformen (H.264 und HEVC sind Apple-Standard).

TECHNISCH

Rendert die Gaussian-Cloud entlang einer parametrischen Orbit-Kamerafahrt und encodiert jeden Frame über AVAssetWriter in eine MP4- oder MOV-Datei. Die Orbit-Konfiguration steuert Drehzahl (Umdrehungen), Distanz, Elevation, FOV, Dauer und Ease-In/Out-Faktor. Der Orbit-Video-Export läuft über RadianceKits EIGENEN ForwardPass mit voller SH-Auswertung — pixelgleich mit dem In-App-Viewport (WYSIWYG). Pro Frame wird die Welt-Anpassungsmatrix (vom Renderer berechnet, um die internen Koordinaten in die Y-up-Orbit-Welt zu drehen) mit der Kamera multipliziert, anschließend eine Kamera-Konvertierungs-Spiegelung (camFlip: Orbit-Y-up → COLMAP-Y-down) angewendet. Das Offscreen-Render-Target wird via IOSurface zu einem CVPixelBuffer für den Encoder gezogen. Der Encoder unterstützt H.264 und HEVC, konfigurierbare Bitrate und Auflösung von 480p bis 8K. Vor dem ersten Frame wartet der Renderer 200 ms, damit die initiale Splat-Sortierung abgeschlossen ist. Dieser Export ist GPU-bound — bei 8K und Millionen Gaussians liegt die Render-Zeit pro Frame bei mehreren Sekunden, also Gesamt-Renderzeiten von 10–30 Minuten für 6 s Video möglich.

EINFACH GESAGT

Eine fertige MP4-Datei mit einer Drehung um deine Szene. Perfekt für Social Media, Marketing, Vorstellungen. Du kannst Dauer (3–30 Sekunden), Drehrichtung und Geschwindigkeit einstellen. Die Datei lässt sich auf YouTube, Instagram, in PowerPoint und überall sonst direkt einbinden. Geht teils langsam, weil die App jeden Frame komplett rendern muss — für ein 8K-Video kannst du fünf bis dreißig Minuten einplanen, je nach Anzahl der Gaussians.

E9 — SfM Transforms (transforms.json)

WO

Menüleiste → Export → Photogrammetry → Export SfM (transforms.json)…. **Größe:** typisch 1–10 KB (nur Posen + Intrinsics, keine Bilder, keine Gaussians). **Kompatibel mit:** nerfstudio, Brush, gsplat, OpenSplat, Meshroom, alle modernen feed-forward 3DGS-Trainer.

TECHNISCH

Schreibt das nerfstudio-`transforms.json`-Format mit einer Liste von Kamera-Posen plus geteilten Intrinsics. Pro Kamera wird die View-Matrix (RadianceKit-intern: World-to-Camera in COLMAP-Konvention) invertiert, anschließend werden die kameralokalen Y- und Z-Basisvektoren gespiegelt, um in die nerfstudio-Konvention (OpenGL-Style, Kamera schaut entlang `-Z`, `+Y` ist oben) zu konvertieren. Die finale 4×4-Matrix landet als row-major nested array von Doubles im `transform_matrix`-Feld jedes Frames. Intrinsics werden auf der Top-Level gespeichert (Brennweite x/y, Hauptpunkt x/y, Bildbreite/-höhe, `camera_model = "OPENCV"`, plus die Distortion-Coefficients `k1, k2, p1, p2`) — außer wenn der Exporter mehrere unterschiedliche Intrinsics-Sets erkennt, dann werden sie per Frame geschrieben. Bildpfade werden als `images/<filename>` relativ zur JSON-Datei geschrieben; der User muss einen Sibling-`images/`-Folder mit den Trainingsfotos anlegen.

EINFACH GESAGT

Diese JSON-Datei beschreibt für jedes Foto, wo die Kamera stand und wohin sie schaute. Die Datei alleine ist klein und nutzlos — sie wird zusammen mit den Originalbildern in einem Ordner verwendet. Nerfstudio, Brush und ein paar andere Trainer lesen genau dieses Format, und du kannst damit deine RadianceKit-SfM-Ergebnisse in ein anderes Tool übergeben, ohne dass dort die Kamera-Rekonstruktion neu gerechnet werden muss. Spart bei großen Szenen Stunden.

E10 — COLMAP Workspace (sparse/0/)

WO

Menüleiste → Export → Photogrammetry → Export SfM (COLMAP Workspace)…. **Größe:** drei Binärdateien zusammen typisch 4–8 MB — `points3D.bin` dominiert (eine Zeile pro 3D-Punkt der Sparse-Cloud), `images.bin` und `cameras.bin` sind jeweils deutlich unter 100 KB. **Kompatibel mit:** COLMAP selbst, Nerfstudio, Postshot, Meshroom, alle Tools, die ein COLMAP-`sparse/`-Verzeichnis erwarten.

TECHNISCH

Schreibt das Standard-COLMAP-`sparse/0/`-Layout mit drei binären Dateien: `cameras.bin`, `images.bin`, `points3D.bin`. Format-Referenz ist die offizielle COLMAP-Dokumentation. `cameras.bin` enthält die deduplizierte Intrinsics-Liste (Kameras mit identischen Intrinsics + Bildgröße werden zu einem einzigen Eintrag zusammengefasst); das verwendete Camera-Model ist `OPENCV` (Model 4), mit fx/fy/cx/cy plus den vier Distortion-Coefficients k1/k2/p1/p2. `images.bin` listet pro Bild die Pose als `wxyz`-Quaternion plus Translation, gefolgt von der Kamera-ID und dem Dateinamen; keine 2D-3D-Korrespondenzen werden gespeichert. `points3D.bin` enthält die SfM-Punktwolke mit Position, Farbe (0-255 RGB) und Default-Werten für Reprojektion und Track-Length. Alles wird in Little-Endian geschrieben. Re-Import in RadianceKit funktioniert über das File-Menü → „Import COLMAP/Metashape Workspace…„ (siehe Q3 im SfM-Backend-Kapitel).

EINFACH GESAGT

Das offizielle COLMAP-Format. Wenn du dein Training in Postshot, Nerfstudio oder einer anderen COLMAP-fähigen Software fortsetzen willst, ist das der Weg. Drei kleine Dateien plus deine Originalbilder, und das Zielprogramm akzeptiert es als wäre COLMAP selbst das Quellprogramm gewesen. Mehr Programme verstehen das als das transforms.json-Format (E9), gleichzeitig etwas weniger handlich, weil binär statt textbasiert.

Welches Format wann?

Ziel	Format
Web-Viewer auf eigener Seite	E7 Web Viewer (.html)
Web-Viewer mit `gsplat.js`	E6 Splat (.splat)
Pipeline-Reuse in Postshot / Nerfstudio	E9 transforms.json + E10 COLMAP Workspace
SuperSplat-Edit	E1 PLY oder E2 Compressed PLY
Niantic Scaniverse / Spatial Fields	E3 SPZ
Maximale Kompression	E4 SOG (cwebp erforderlich)
Marketing-/Social-Video	E8 Orbit Video

Schnell-Vergleich

Format	Erweiterung	Sandbox	Größe (1M Gauss)	Best-Use
E1 PLY	`.ply`	ja	~250 MB	Archiv, höchste Kompatibilität
E2 Compressed PLY	`.ply`	ja	~40 MB	Web + SuperSplat
E3 SPZ	`.spz`	ja (gzip-Spawn)	~40 MB	Niantic + Mobile
E4 SOG	`.sog`	bedingt (cwebp)	~20 MB	Maximale Kompression
E5 glTF	`.glb`	ja	~250 MB	Khronos-Pipeline
E6 Splat	`.splat`	ja	~32 MB	gsplat.js Web-Viewer
E7 Web Viewer	`.html`	ja	~45 MB	Standalone Browser-Datei
E8 Orbit Video	`.mp4` / `.mov`	ja	variabel	Social/Marketing
E9 SfM Transforms	`.json`	ja	~5 KB	Posen-Übergabe
E10 COLMAP Workspace	Verzeichnis	ja	~4–8 MB	Posen-Übergabe binär

Größen-Spalte sind grobe Richtwerte für 1 Mio. Gaussians mit SH-Grad 3. Reale Werte variieren je nach Komprimierbarkeit der Szene; SH-Grad 0 reduziert PLY/glTF um den Faktor 4.

KAPITEL

Kapitel 9 — SfM-Backends

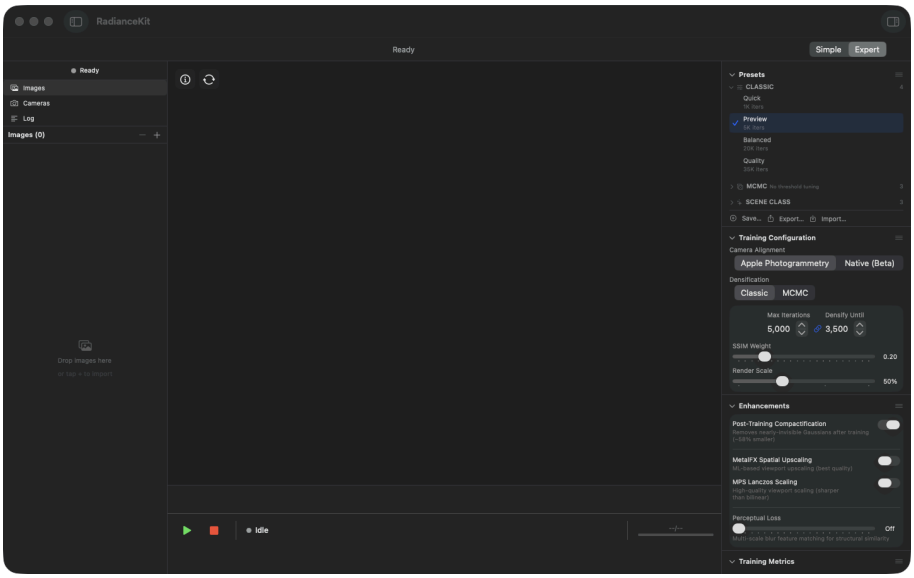

Abbildung 33: Expert-Modus mit Camera-Alignment-Picker im Inspector (Apple Photogrammetry / Native (Beta))

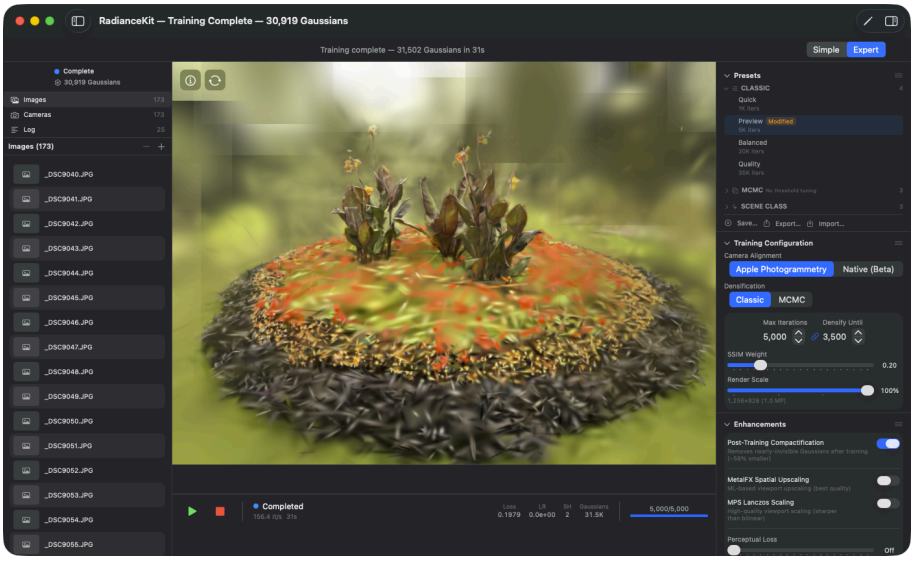

Abbildung 34: Inspector mit aktivem Native (Beta) — Camera-Alignment-Picker zweite Option ausgewählt, alle anderen Trainingskonfigurations-Parameter unverändert

WAS IM BILD ZU SEHEN IST Der Camera-Alignment-Picker im Inspector ist ein Segmented-Control mit zwei Optionen — Apple Photogrammetry (Default für App-Store-Builds, voll sandbox-konform) und Native (Beta) (RadianceKit's eigenes FAST+BRIEF+GLOMAP-

Pipeline-Backend, Phase 3.8/3.9 entwickelt, Stand 2026-05). Native (Beta) ist orbit-only validiert und schneller bei ≥1 000 Frames als Apple Photogrammetry, schließt aber das Phase-3-§5-Quality-Gate (finalLoss ≤ 0.0115) noch nicht ein — daher der Beta-Tag. Externe SfM-Ergebnisse aus Metashape, COLMAP oder einer anderen Photogrammetrie-Software können zusätzlich über das File-Menü importiert werden (Q3 COLMAP-Text-Format, Q6 Workspace-Import) — der Picker wechselt nicht, aber die importierten Posen ersetzen das SfM-Resultat.

SfM steht für **Structure from Motion**. Aus einer Menge überlappender Fotos rekonstruiert die Software für jedes Bild die Position und Blickrichtung der Kamera in einem gemeinsamen 3D-Koordinatensystem. Dazu wird eine grobe 3D-Punktwolke erzeugt, die das Training mit Gaussian Splatting initialisiert. Das SfM-Ergebnis ist die Eingabe für das eigentliche Training und entscheidet maßgeblich über die spätere Bildqualität.

RadianceKit bietet fünf SfM-Wege: zwei in der App eingebaute Backends (Q1 Apple Photogrammetry, Q4/Q5 Native), zwei Import-Pfade aus externen Tools (Q3 COLMAP-Text-Format, Q6 binärer Workspace-Import) sowie Q2 COLMAP-Binary, das nur in Developer-Builds außerhalb des App Store verfügbar ist. Welcher der richtige ist, hängt vom Szenentyp ab (Orbit um ein Objekt, Innenraum, Drohnenflug) und davon, ob eine externe Software bereits eine Rekonstruktion liefert.

Q1 — Apple Photogrammetry

WO

Expert View → Inspector → Trainingskonfiguration → Camera Alignment-Picker, Eintrag „Apple Photogrammetry„.

TECHNISCH

Wrapt Apples eingebautes Photogrammetry-Framework, das ursprünglich für Object Capture entwickelt wurde. Apple extrahiert intern Features mit einer proprietären Pipeline (Schritte sind nicht öffentlich dokumentiert), verifiziert sie über Multi-View-Matching und löst Bundle-Adjustment auf der Apple-Silicon Neural Engine + GPU. Das Backend ist vollständig App-Store-konform (keine externe Binary, Sandbox=true, on-device), liefert aber nur Kamera-Posen plus eine grobe Punktwolke — keine Diagnose-Metriken wie Track-Länge oder Reprojektionsfehler. Skaliert nach Apples Empfehlung bis ein paar Hundert Bilder. Bei mehr als ~500 Frames in linearen Drohnenflügen oder großen Outdoor-Szenen wurden reproduzierbar Crashes oder stilles Verwerfen einzelner Kameras beobachtet.

EINFACH GESAGT

Das ist der einfachste Weg. Bilder rein, App rechnet. Funktioniert sehr gut bei klassischen Objekt-Scans — wenn du um ein Möbelstück oder eine Skulptur herumläufst und 50–200 Fotos machst. Bei Drohnenflügen über Landschaften oder bei sehr vielen Bildern (über 500) wird Apples Verfahren aber gerne instabil. Für solche Szenen das Native-Backend (Q4/Q5) testen oder die Kameras in Metashape rechnen und über den Workspace-Import (Q6) einspielen.

POWER-USER

Q2 COLMAP-Binary — spawnt das externe COLMAP-Programm als Sub-Prozess und ist deshalb in der App-Store-Version (Sandbox) **nicht verfügbar**. Funktioniert nur in Developer-Builds außerhalb des App Stores. Für die Qualität, die COLMAP liefert, gibt es in der App-Store-Version den Workspace-Import (Q3 oder Q6): rechne die SfM in COLMAP oder Metashape außerhalb und lade das Ergebnis ein.

Q3 — COLMAP-Text-Format (Metashape / ETH3D)

WO

Menü „File → Import COLMAP / Metashape Workspace…„ (Cmd+⇧+I) ODER Drag-and-Drop eines Ordners mit `sparse/0/cameras.txt`.

TECHNISCH

Liest den standardisierten COLMAP-Text-Export — drei Textdateien `cameras.txt`, `images.txt`, `points3D.txt` im `sparse/0/`-Unterordner — und konvertiert in das interne SfM-Ergebnis-Modell. Selbe Format-Definition wie der COLMAP-Binär-Export, nur als ASCII statt Binär. Wird von Agisoft Metashape, RealityCapture, PolyCam und dem ETH3D-Benchmark in genau diesem Layout ausgegeben. Der Parser teilt die Camera-Model-Erkennung mit dem Binary-Parser (alle gängigen Modelle: SIMPLE_PINHOLE, PINHOLE, OPENCV, OPENCV_FISHEYE, FULL_OPENCV). Robust gegen Kommentar-Zeilen und leere Zeilen. Skaliert in Tests bis ~1 400 Kameras (ETH3D Tunnel) ohne Probleme.

EINFACH GESAGT

Wenn du schon mit Metashape, RealityCapture oder einer anderen kommerziellen Foto-3D-Software gearbeitet hast und das Ergebnis exportiert hast — diesen Export kannst du in RadianceKit direkt laden, ohne dass die App selbst neu rechnen muss. Das spart Stunden Wartezeit. Lade einfach den ganzen Ordner über das File-Menü oder zieh ihn ins Fenster.

Q4 — Native SfM (inkrementell)

WO

Expert View → Inspector → Trainingskonfiguration → Camera Alignment-Picker, Eintrag „Native (Beta)„. Inkrementell ist der Default-Modus dieses Backends — es gibt im Inspector keinen separaten Mapper-Picker. Per CLI lässt sich der Modus explizit setzen mit `--native-sfm` oder `--sfm-mapper incremental`.

TECHNISCH

Eigene GPU-beschleunigte Implementierung der gesamten SfM-Pipeline: FAST+BRIEF-Features ODER SuperPoint+LightGlue über CoreML (mit `--coreml-features`), gefolgt von Hamming-KNN-Matching, RANSAC-Fundamentalmatrix, Track-Building, Initial-Pair-Auswahl, Two-View-Bootstrap (F→E plus DLT), greedy inkrementellem Mapper mit PnP-Registrierung und Multi-View-Triangulation und finalem Bundle-Adjustment via Schur-reduziertem Levenberg-Marquardt mit Huber-Loss und analytischen Jacobians über Cholesky-Solve. Komplett App-Store-konform: keine externe Binary, Sandbox=true. Mit dem in Phase 3.10 ausgelieferten R2-Collapse-Detektor: registriert die App weniger als 60 % der Eingabe-Frames oder fällt die Points-per-Camera-Rate unter 13, wird automatisch auf den globalen Mapper (Q5) ausgewichen. Empirisch sauber auf Orbit-/Turntable-Szenen; auf allgemeineren Bewegungen (Drohnenflug, Innenräume mit komplexer Geometrie) ist die Erfolgsrate niedriger — der Detektor fängt diese Fälle aber ab. Skaliert bis ~200 Kameras zuverlässig, höher mit deutlich längerer Laufzeit.

EINFACH GESAGT

Apples Stärken (App-Store-Kompatibel, schnell für Orbits) mit zusätzlichen Diagnose-Werten. Funktioniert besonders gut, wenn du wie für einen Object Capture um ein Motiv herumgehst. Bei komplizierteren Aufnahmen (Drohnenflug oder Wohnzimmer) erkennt RadianceKit automatisch, dass das nichts wird, und springt auf das globale Verfahren um. Markiert „Beta„, weil noch in Erprobung — die Standard-Empfehlung ist nach wie vor Apple Photogrammetry für einfache Object-Scans und der Workspace-Import (Q3 oder Q6) für anspruchsvolle Outdoor-Sets.

Q5 — Native SfM (global)

WO

Wird automatisch aufgerufen, wenn der inkrementelle Mapper (Q4) den Collapse-Detektor auslöst (weniger als 60 % der Eingabe-Frames registriert oder Points-per-Camera-Rate unter 13). Manuell forcierbar nur via CLI `--sfm-mapper global`. Im Inspector ist das Globale Verfahren über keinen separaten Picker erreichbar — die App entscheidet selbst, wann sie umschaltet.

TECHNISCH

Globale Variante der nativen Pipeline. Erst Feature-Extraktion + Matching wie Q4, dann Relativ-Pose-Schätzung für alle verifizierten Paare, anschließend Rotation-Averaging (synchronisiert alle Kamera-Rotationen im Welt-Koordinatensystem) und Translation-Averaging (LSQR-basiert auf einer matrix-freien Sparse-Formulierung, um Integer-Overflow bei großen Kamera-Mengen zu vermeiden). Skaliert auf ~5 000 Kameras im Prinzip, in der Praxis Quality-degraded oberhalb einiger Hundert Kameras — die Phase-3.8-§5-Akzeptanz-Gate-Messung auf K-1351 ergab finalLoss 0.07 statt der angestrebten 0.0115. Wird als „Fallback-Tier„ gehandhabt: kommt zum Einsatz, wenn der inkrementelle Mapper degeneriert, wird selbst aber nicht erneut auf Qualität geprüft.

EINFACH GESAGT

Der Plan-B-Pfad für die native Engine. Wird automatisch aufgerufen, wenn der schnellere inkrementelle Pfad versagt. Liefert ein brauchbares Ergebnis, ist aber bei sehr großen oder schwierigen Szenen meist nicht so präzise wie das, was du aus Metashape oder einer externen COLMAP-Installation bekommst. Wenn Native dein Standard-Workflow wird, lohnt sich in solchen Fällen der Umweg über den Workspace-Import (Q3 oder Q6).

Q6 — Metashape / COLMAP-Text-Workspace Import (Phase Q7)

WO

File-Menü → „Import COLMAP / Metashape Workspace…„ (Cmd+⇧+I). Drag-and-Drop eines Ordners mit `sparse/0/cameras.{bin,txt}` und `images/`.

TECHNISCH

Erkennt automatisch, ob ein per Drag-and-Drop oder Open-Panel ausgewählter Ordner einer der drei COLMAP-Workspace-Layouts entspricht (`sparse/0/`, `sparse/`, oder Root) und ob die Reconstruction binär (`cameras.bin`) oder Text (`cameras.txt`) vorliegt. Der binäre Pfad nutzt den COLMAP-Binär-Parser, der Text-Pfad den ETH3D-Loader — beide produzieren dasselbe SfM-Ergebnis-Modell und der Rest der Pipeline (Bilder importieren, MCMC-Training starten) ist agnostisch gegenüber der Quelle. Die Bilder werden über das App-Sandbox-Bookmark-System security-scoped geöffnet, sodass der Import auch in der App-Store-Version funktioniert. Speziell für den Fall „Metashape-Export ohne Rekonstruktion neu rechnen„ gedacht. Die im File-Menü-Eintrag erwähnte Erkennung warnt im App-Log, falls der gewählte Ordner kein erkennbarer Workspace ist.

EINFACH GESAGT

Speziell die Metashape-User-Funktion. Wenn du eine Lizenz für Metashape oder RealityCapture hast und dort die Kamera-Rekonstruktion gemacht hast, kannst du den Export-Ordner einfach hier reinziehen und sofort mit dem Training starten. Spart bei großen Szenen mehrere Stunden Rechenzeit, weil RadianceKit das SfM dann nicht selbst macht.

Welches Backend wann?

Szenario	Empfohlenes Backend
Objekt-Scan, 50–200 Fotos	Q1 Apple Photogrammetry
Großer Outdoor / Drohne / >500 Bilder	Q6 Workspace-Import (in Metashape oder COLMAP rechnen, dann einladen)
Metashape/RealityCapture-Export liegt vor	Q6 Import (kein SfM nötig)
ETH3D / akademisches COLMAP-Text-Set	Q3 COLMAP-Text-Import
Strikt App-Store-konform + Orbit-Szene	Q4 Native inkrementell
Q4 schlägt fehl	Q5 Native global (automatisch)
ETH3D-Benchmark-Daten	Q3 (autotest precomputed)

Schnell-Vergleich

Backend	App-Store	Sandbox	Externe Binary	Best Use	Max ~Cams
Q1 Apple PG	✅	✅	—	Orbit-Object	~300
Q2 COLMAP Binary	❌ (nur Developer-Build)	—	colmap/glomap	Outdoor large	~5 000
Q3 COLMAP-Text-Import	✅	✅	—	Bench rigs	~1 500
Q4 Native incremental	✅	✅	—	Orbit-Object	~200
Q5 Native global	✅	✅	—	Q4-Fallback	~1 351
Q6 Workspace-Import	✅	✅	—	Metashape-Reuse	per Quelle

KAPITEL

Kapitel 10 — Einsteiger-Modus

Der Einsteiger-Modus (engl. Simple Mode, Cmd+1) ist der geführte Workflow für alle, die zum ersten Mal eine 3D-Gaussian-Splatting-Szene rekonstruieren. Statt eine Sidebar voller Inspector-Felder anzuzeigen, führt die App durch vier Schritte: zuerst Bilder oder ein Video importieren und ein Qualitäts-Preset wählen, dann läuft die Verarbeitung (SfM + Training), anschließend kann die fertige Szene in einer 3D-Vorschau begutachtet werden, und zum Schluss wird in das gewünschte Format exportiert. Eine schmale Fortschrittsleiste am oberen Fensterrand zeigt jederzeit an, in welchem Schritt du gerade bist.

Im Vergleich zum Expert-Modus (Cmd+2), der alle Bedienfelder gleichzeitig zeigt, blendet der Einsteiger-Modus ungenutzte Optionen aus, gibt Validierungs-Warnungen bei zu wenigen oder schlechten Bildern und bietet auf jedem Schritt nur die Buttons an, die im aktuellen Zustand sinnvoll sind. Du kannst jederzeit zwischen Einsteiger- und Expert-Modus wechseln (Cmd+1 / Cmd+2), der gesamte Zustand — importierte Bilder, gewähltes Preset, gerade laufendes Training, fertige Punktwolke — bleibt erhalten und ist im jeweils anderen Modus sofort verfügbar.

Z1 — Import (Bilder & Preset wählen)

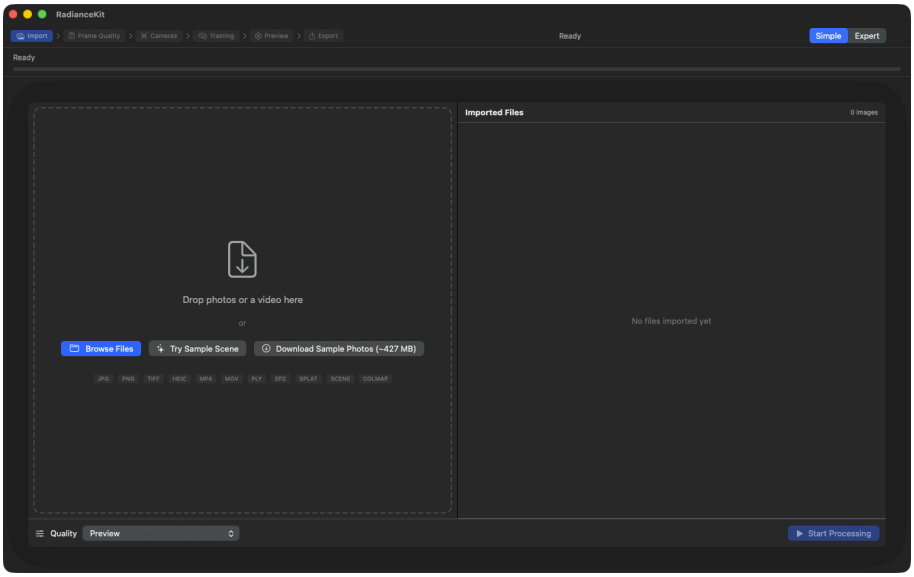

Abbildung 35: Simple-Mode Schritt 1 — leere Drop-Zone vor Bild-Import, Crumb-Trail oben (Import → Frame Quality → Cameras → Training → Preview → Export), Format-Pills JPG/PNG/TIFF/HEIC/MP4/MOV/PLY/SPZ/SPLAT/SCENE/COLMAP

WAS IM BILD ZU SEHEN IST Crumb-Trail (Import aktiv) zeigt den vierschrittigen Workflow. Linke Drop-Zone mit drei CTAs: „Browse Files„ (NSOpenPanel), „Try Sample Scene“ (bundled Demo), „Download Sample Photos (~427 MB)„ (Mip-NeRF360 flowers Subset). Format-Pills darunter listen alle akzeptierten Dateitypen. Rechts „Imported Files“ mit Counter „0 images„ und Empty-State „No files imported yet“. Unten Quality-Picker (Default: Preview) und „Start Processing„ (deaktiviert solang keine Bilder da sind).

Der erste Schritt besteht darin, der App Bildmaterial zu geben. Per Drag-and-Drop in das große, gestrichelte Feld in der Mitte, per „Browse Files„-Button oder per Klick auf die mitgelieferte Sample-Szene. Rechts erscheint eine Liste aller importierten Bilder mit Auflösung und Dateigröße; unten in der schwebenden Werkzeugleiste wählst du das Qualitäts-Preset und startest mit „Start Processing“ die Pipeline. Validierungs-Warnungen (rot bei < 3 oder < 10 Bildern, orange bei 10–19) zeigen an, ob die App eine sinnvolle Rekonstruktion erwartet oder nicht.

C-01 ProgressIndicator (Schritt-Anzeige)

WO

Oben über dem Workflow, immer sichtbar.

TECHNISCH

Zeigt eine horizontale Fortschrittsleiste über die gesamte Pipeline (Frame-Quality → SfM → Training) mit Stage-Allocation: Frame-Quality belegt 0–5 % (Phase 3.11, sehr kurz), SfM belegt 0–30 % der Bar, Training 30–100 %. Daneben Status-Text und phasen-benannte Prozent-Anzeige („SfM 41 %„, „Training 12 500/20 000“), damit Anwender nicht den scheinbaren Rückschritt „41 % SfM → 25 % Training„ als Fehler lesen — die Bar zeigt den gesamten Pipeline-Fortschritt, nicht die Sub-Stage. ETA-Berechnung beginnt, sobald genug Trainings-Tempo gemessen ist (typischerweise nach den ersten 100 Iterationen). Dieselbe Anzeige wird auch im Expert-Modus oberhalb des Inspectors verwendet.

EINFACH GESAGT

Die schmale Leiste ganz oben ist deine Karte durch den Workflow. Sie sagt dir nicht nur, was die App gerade macht (Kameras ausrichten, Training läuft, …), sondern auch, wie weit sie insgesamt schon ist. Die Aufteilung ist absichtlich so, dass die Kamera-Berechnung das erste Drittel der Leiste belegt und das eigentliche Training die hinteren zwei Drittel — sonst würde es wirken, als wäre der Fortschritt nach SfM plötzlich zurück bei null. Du kannst dich also entspannt zurücklehnen, ein Blick auf die Leiste genügt, um die grobe Etappe zu sehen. Der Text daneben sagt dir, ob du gerade in der SfM-Stufe (z. B. „SfM 41 %„) oder im Training (z. B. „Training 12 500/20 000“) steckst, damit die Zahlen nicht verwirrend wirken. Wenn du die ETA nicht angezeigt bekommst, ist das Training einfach noch zu jung — die App schätzt erst, sobald sie genug Tempo gemessen hat.

C-03 DropZoneView (Drag-and-Drop-Bereich)

WO

Linke Seite des Import-Schritts, großes gestricheltes Rechteck mit Symbol. Wird im Einsteiger-Modus mit dem Label „Drop photos or a video here„ angezeigt.

TECHNISCH

Drop-Bereich, der das Symbol kurz hüpfen lässt und den Hintergrund einfärbt, sobald Drag-Items über dem Feld schweben. Akzeptiert JPG, PNG, TIFF, HEIC, MP4, MOV, PLY, SPZ, .splat, .radiancescene-Bundles und Verzeichnisse. Drop-Routing nach Typ: Bilder werden gesammelt und sortiert übergeben, Videos triggern den Frame-Sampling-Pfad, Splat-Dateien öffnen direkt die Vorschau, Scene-Bundles werden eingelesen. Verzeichnisse werden enumeriert und alle enthaltenen Bilder importiert. Security-scoped Bookmarks für sandbox-konformen Zugriff werden korrekt aufgenommen und freigegeben. Nicht-unterstützte Endungen werden als Warnbanner für 5 Sekunden angezeigt.

EINFACH GESAGT

Das große gestrichelte Feld ist die Hauptbedienung des ersten Schritts. Zieh einfach Fotos oder ein Video hinein, oder einen ganzen Ordner — die App nimmt alles, was sie kennt, und ignoriert den Rest. Wenn das Feld blau wird und das Symbol kurz hüpft, hat die App den Drag erkannt. Lass los, und der Import startet sofort: Bilder wandern in die Liste rechts, Videos triggern automatisch den Frame-Sampling-Schritt, und bereits trainierte `.ply` / `.spz` / `.splat` Dateien öffnen direkt die Vorschau. Falls ein Format gar nicht passt (etwa PDF oder BMP), erscheint ein kurzer Hinweis am oberen Rand — die App schluckt unbekanntes Material nicht stumm.

C-05 Browse Files Button

WO

Innerhalb der Drop-Zone, prominenter Button.

TECHNISCH

Button, der den macOS-Datei-Dialog mit Mehrfach-Auswahl und den Dateitypen JPG, PNG, TIFF, MP4, MOV, Ordner sowie dem App-eigenen Scene-Format öffnet. Ergebnis-URLs sind security-scoped und werden über dieselben Import-Pfade weitergeleitet wie Drag-and-Drop. Wenn der Benutzer einen Ordner auswählt, wird er rekursiv nach Bildern enumeriert.

EINFACH GESAGT

Wenn dir Drag-and-Drop unbequem ist, klick einfach diesen Button und navigiere im macOS-Datei-Dialog zu deinen Fotos. Du kannst mehrere Dateien gleichzeitig wählen (Cmd-Klick auf die einzelnen Bilder) oder einen ganzen Ordner auswählen — die App durchsucht den Ordner dann rekursiv nach allen unterstützten Bildtypen. Das ist besonders praktisch, wenn deine Aufnahmen verschachtelt in Unterordnern liegen (z. B. „shoot-day1/„, „shoot-day2/“) — ein Klick auf den Hauptordner reicht. Funktional macht der Button exakt das, was auch Drag-and-Drop macht; wähl einfach den Weg, der dir bequemer ist.

C-06 Try Sample Scene Button

WO

Innerhalb der Drop-Zone, nur sichtbar wenn das App-Bundle die Sample-Scene enthält und noch keine Bilder/Splats importiert sind.

TECHNISCH

Erscheint nur, wenn (a) eine `sample-scene.splat`, `.spz` oder `.ply` im App-Bundle vorhanden ist UND (b) noch keine Bilder/Videos importiert und noch keine Punktwolke vorhanden ist. Bei Klick lädt die fertige Punktwolke (bevorzugt das kleinste Format — `.splat` ~3 MB, `.spz` ~1.4 MB, Fallback `.ply`) und setzt nach 400 ms hartkodierte Kamera-Werte aus dem Original-Metadaten der Blumen-Szene für eine ästhetisch sinnvolle Eingangs-Perspektive.

EINFACH GESAGT

Wenn du die App zum ersten Mal startest und einfach mal sehen willst, was am Ende rauskommt — klick hier. Das öffnet eine fertig trainierte Blumen-Szene, die du sofort drehen und exportieren kannst, ohne dass die App rechnen muss. Die Kamera ist voreingestellt auf eine ästhetisch sinnvolle Eingangsperspektive, sodass du gleich was Schönes siehst. Perfekt, um die 3D-Steuerung und den Export-Schritt risikolos auszuprobieren, bevor du auf eigene Aufnahmen losgehst. Sobald du eigene Bilder importierst, verschwindet der Button automatisch — er wird nur angezeigt, solange das Projekt komplett leer ist.

C-07 Download Sample Photos Button

WO

Innerhalb der Drop-Zone, neben „Try Sample Scene„; gleiche Sichtbarkeits-Bedingungen.

TECHNISCH

Triggert einen Download (Repo `github.com/bkindler/radiancekit-sample-photos`), der ca. 427 MB an 960 Vollauflösungs-Frames lädt und in die App speist. Während des Downloads wird der Button deaktiviert. Der Fortschritt erscheint in der oberen Progress-Bar als „Downloading X %„ in einer eigenen Stage, weil diese Stage ihre eigene 0–100 %-Skala behält und der spätere SfM-Stage nicht überlappt.

EINFACH GESAGT

Genau wie die Sample-Scene, nur mit den Ausgangs-Fotos statt mit dem fertigen Ergebnis. So kannst du die gesamte Pipeline einmal selbst durchlaufen lassen und sehen, wie lange SfM und Training auf deinem Mac wirklich dauern. Der Download ist groß (etwa eine halbe DVD = 427 MB), passiert aber nur einmal — danach sind die Fotos lokal und du kannst die Pipeline beliebig oft mit verschiedenen Presets neu starten. Während der Download läuft, zeigt die obere Fortschrittsleiste den aktuellen Download-Stand in Prozent, sodass du abschätzen kannst, wann es losgeht. Tipp: nimm dazu am besten ein schnelles WLAN oder Kabel-Netz — die 427 MB ziehen sich sonst.

C-09 Quality Presets Picker

WO

Schwebende untere Werkzeugleiste des Import-Overlays, links neben dem Start-Button.

TECHNISCH

Bedienelement mit Label „Quality„ gruppiert die verfügbaren Presets nach Kategorie (Classic / MCMC / Custom). Built-in Presets werden nach Kategorie gruppiert; die Abschnitts-Header sind hartkodiert. Custom-Presets nur sichtbar wenn welche existieren. Locked-State: Presets, die nicht in der Free-Liste (Quick + Preview) liegen, bekommen ein „🔒"-Suffix am Namen, wenn der User nicht gekauft hat; bei Auswahl springt der Picker auf Preview zurück und öffnet automatisch das Purchase-Sheet. Bei Wahl wird das Preset angewendet, was die gesamte Trainings-Konfiguration ersetzt.

EINFACH GESAGT

Hier wählst du, wie genau und wie lange die App rechnen soll. „Quick„ und „Preview" sind ohne Kauf nutzbar und liefern in wenigen Minuten ein erstes Ergebnis — ideal, um zu testen, ob deine Bilder überhaupt sinnvoll sind. „Balanced„ und „Quality" brauchen die Vollversion und liefern deutlich saubere Modelle, dauern dafür aber Stunden statt Minuten. MCMC ist eine andere Strategie, die mit weniger Gauss-Splats auskommt — gut, wenn du das Modell später kompakt exportieren oder ins Web stellen willst. Premium-Presets erkennst du am kleinen Schloss-Symbol am Namen; tippst du eines ohne Lizenz an, springt der Picker zurück auf Preview und das Kauf-Sheet öffnet sich automatisch. Faustregel: starte immer mit Preview, schau dir das Ergebnis an, und entscheide dann, ob sich ein längerer Lauf lohnt.

C-10 Start Processing Button

WO

Schwebende untere Werkzeugleiste des Import-Overlays, rechts neben dem Preset-Picker.

TECHNISCH

Button, der grau bleibt, solange weder Bilder noch ein Video importiert sind. Bei Klick startet die Pipeline und schaltet die Stage-Maschine in die Reihenfolge Frame-Quality → SfM → Training um. Der Button selbst hat keinen weiteren Status; eine laufende Verarbeitung erscheint stattdessen als separater Verarbeitungsbildschirm.

EINFACH GESAGT

Der „Loslegen„-Knopf. Solange er grau ist, fehlen noch Eingabe-Bilder oder ein Video. Sobald du Fotos hereingezogen hast, wird er aktiv und du klickst ihn an, um SfM und Training nacheinander zu starten. Ab da übernimmt die App den gesamten Ablauf und du landest automatisch auf dem Verarbeitungsbildschirm (Z2). Du musst nichts weiter klicken — erst nach Trainings-Ende wechselt die App wieder in die Vorschau (Z3). Wenn du es dir nochmal anders überlegst, kannst du auch danach noch jederzeit per Cancel abbrechen.

C-11 Video Sampling Slider

WO

Rechte Bilderliste, sichtbar nur wenn ein Video (statt Bildern) importiert wurde.

TECHNISCH

Schieberegler 0.5 fps – 30 fps in 0.5er-Schritten. Bei Änderung wird die Frame-Dichte aktualisiert und zusätzlich die Anzahl der Zielframes (mindestens 10) aus Dichte und Videolänge berechnet. Der Schieberegler liegt außerhalb der Bilderliste, weil Listen-Elemente Maus-Events von Schiebereglern blockieren würden. Unter dem Schieberegler stehen die berechneten Zielframes („247 frames„) und die Videolänge („1m23s video“). Tooltip warnt: „Doubling the density doubles the number of frames and increases SfM time by ~100%.„

EINFACH GESAGT

Wenn du statt Fotos ein Video importiert hast, entscheidet dieser Schieberegler, wie viele Einzelbilder die App aus dem Video herausziehen soll. Mehr Bilder = bessere Qualität, aber linear mehr Rechenzeit. Für ein 30-Sekunden-Orbit-Video sind 5 fps (150 Bilder) ein guter Anfang; bei 1-minütigen Aufnahmen reicht oft 3 fps völlig. Unter dem Regler zeigt die App live an, wie viele Frames bei der aktuellen Einstellung herauskommen — so siehst du sofort, ob du in den sinnvollen Bereich von etwa 100–300 Bildern triffst. Wenn das Ergebnis schlecht wird, zieh den Regler nach rechts und probier es nochmal; die Verdopplung der Frame-Rate verdoppelt aber auch grob die SfM-Dauer.

C-12 Clear All Button

WO

Rechte Bilderliste, unten rechts; sichtbar nur wenn Bilder importiert wurden.

TECHNISCH

Roter Button. Klick öffnet einen Bestätigungs-Dialog mit Titel „Clear all imported files?„ und Message „N images will be removed.“. Bestätigung leert alle importierten Bilder/Videos, Staging-Verzeichnisse, die Punktwolke, den Trainings-Status, das SfM-Ergebnis und alle Caches; die Stage springt zurück auf Import. Auf Cancel bleibt alles erhalten. Der Dialog ist als zerstörungsfreier Default-Pfad konfiguriert (zerstörender Button rot markiert).

EINFACH GESAGT

Wenn du komplett neu anfangen willst, klick hier. Die Bestätigungs-Frage erscheint, weil das Löschen alle aktuellen Importe samt ggf. bereits berechneten Kameras und Trainings-Resultaten verwirft — du kannst es nicht rückgängig machen. Sinnvoll, wenn du das gewählte Bildmaterial komplett austauschen willst oder ein altes Projekt loswerden möchtest, bevor du ein neues startest. Beachte: ein einzelnes Bild rauszunehmen geht über die Liste rechts (siehe nächster Punkt), nicht über diesen Button. Deine Dateien auf der Festplatte werden dabei nicht gelöscht — die App vergisst nur ihre Referenzen.

C-13 File List ForEach (Einzel-Image-Entfernen)

WO

Rechte Bilderliste, jeder Eintrag.

TECHNISCH

Liste über die importierten Bilder mit Swipe-zum-Löschen. Pro Bild eine Zeile mit Icon, Dateiname, Auflösung („1920 × 1080″) und Dateigröße (formatiert KB/MB). Auflösung kommt aus einem Metadaten-Cache, der asynchron aus den Bild-Headern befüllt wird, damit die Oberfläche nicht blockiert. Die Löschen-Aktion bietet macOS-typisches Swipe-Delete (Trackpad-Swipe links auf einer Zeile) sowie Tastatur-Delete bei selektierter Zeile. Hinweis: Der erweiterte Image-Delete-Pfad mit explizitem Minus-Button, Backspace und Cmd-Z zum Rückgängigmachen wurde *nur im Expert-Modus* im Project Navigator ergänzt — im Einsteiger-Modus bleibt es bei Swipe-Delete.

EINFACH GESAGT

Die Liste rechts zeigt jedes importierte Bild mit Auflösung und Dateigröße — praktisch, um auf einen Blick zu sehen, ob du gemischt hochauflösendes mit niedrigauflösendem Material zusammengewürfelt hast. Um ein einzelnes Bild rauszunehmen, wisch es mit zwei Fingern nach links auf dem Trackpad — wie in iOS Mail — oder wähle es an und drücke Delete. Die App löscht die Datei selbst nicht; sie nimmt sie nur aus dem aktuellen Projekt. Falls du eine richtige Minus-Schaltfläche oder Cmd-Z-Rückgängig brauchst, wechsle in den Expert-Modus (Cmd+2), dort gibt es das im Project Navigator. Im Einsteiger-Modus bleibt es bewusst beim einfachen Swipe-Pattern.

C-15 Validation Warnings (3-Stufen-Tier)

WO

Unter der Bilderliste, oberhalb des Clear-All-Buttons.

TECHNISCH

Drei aufeinander folgende Schwellen basierend auf der Anzahl importierter Bilder (nur aktiv wenn Bilder vorhanden und kein Video): - < 3 Bilder: rotes Banner (red octagon), Text „At least 3 images are required. Camera alignment cannot be computed from fewer images.„ - 3–9 Bilder: rotes Banner, Text „With fewer than 10 images, SfM often fails and the trained scene tends to overfit [...]. 15–20 images minimum recommended; 30+ for object captures.“ - 10–19 Bilder: orangefarbenes Banner (warning triangle), Text „Workable, but quality usually improves with 20+ images and good coverage around the scene.„

Ab 20 Bildern verschwindet das Banner. Schwellenwerte sind hartkodiert und basieren auf empirischen 560+-Trainings-Experimenten.

EINFACH GESAGT

Die App schaut sich an, wie viele Bilder du importiert hast, und gibt dir eine farbige Einschätzung. Rot heißt: das wird mit hoher Wahrscheinlichkeit nichts — entweder kann SfM keine Kameras berechnen oder das Training überfittet auf zu wenig Material. Orange heißt: könnte klappen, aber rechne nicht mit Top-Qualität, weil der Algorithmus zwischen den Bildern wenig Überlapp findet. Kein Banner heißt: gute Voraussetzungen, du hast genug Material. Wenn du wirklich saubere Modelle willst, peile mindestens 30–50 gleichmäßig verteilte Aufnahmen rund um dein Motiv an — gerne auch deutlich mehr bei Außenszenen oder großen Räumen. Du kannst trotz Warnung starten, aber sei nicht überrascht, wenn SfM kommentarlos abbricht oder das Modell löchrig aussieht.

C-16 COLMAP Workspace Detection

WO

Beim Drop eines Ordners — keine sichtbare Schaltfläche, sondern Erkennungs-Logik.

TECHNISCH

Beim Drop eines Verzeichnisses wird geprüft, ob darin eine der drei kanonischen Workspace-Layouts vorliegt: `sparse/0/cameras.bin`, `sparse/cameras.bin` oder direkt `cameras.bin` im Root. Trifft das zu, wird die Standard-Bild-Enumeration abgebrochen und stattdessen ein modaler Alert geöffnet, der den User fragt, ob die bestehende Rekonstruktion verwendet oder die Bilder neu durch Apple Photogrammetry geschickt werden sollen. Gleicher Pfad auch für Text-Format-Workspaces (cameras.txt) und ETH3D-Exporte. Siehe Kapitel 9 Backend Q6 für Details. Wirkt im Einsteiger-Modus genauso wie im Expert-Modus.

EINFACH GESAGT

Wenn du schon einmal mit Metashape, RealityCapture oder COLMAP gearbeitet hast und dort die Kamera-Berechnung schon laufen ließest, kannst du den Export-Ordner einfach hier reinziehen. RadianceKit erkennt am Inhalt automatisch, dass es sich um ein COLMAP-Workspace handelt (es prüft auf `sparse/0/`, `cameras.bin` und Co.), und fragt dich, ob es die fertige Rechnung übernehmen oder selbst neu rechnen soll. Übernehmen spart bei großen Szenen Stunden Wartezeit, weil SfM komplett übersprungen wird — das Training startet sofort. Auch Text-Format-Workspaces (cameras.txt) und ETH3D-Exporte werden erkannt. Diese Funktion ist im Einsteiger-Modus genauso wie im Expert-Modus verfügbar; mehr Details stehen in Kapitel 9 unter Backend Q6.

Wann zur nächsten Stufe?

Du kannst Start Processing klicken, sobald (a) mindestens ein Bild oder ein Video importiert ist und (b) das Validation-Banner orange oder verschwunden ist. Bei rotem Banner lässt die App dich zwar trotzdem starten, du kannst aber mit hoher Wahrscheinlichkeit die Verarbeitung gleich wieder abbrechen. Empfohlen: mindestens 20 Bilder, scharf, mit deutlichem Überlapp zwischen aufeinander folgenden Aufnahmen, alle aus etwa der gleichen Distanz zum Motiv. Wähle vor dem Start ein Preset, das zu deinem Zeitbudget passt — bei 30 Bildern und Quick-Preset bist du in wenigen Minuten durch, bei Quality dauert es eher 1–2 Stunden.

Z2 — Verarbeitung (SfM + Training)

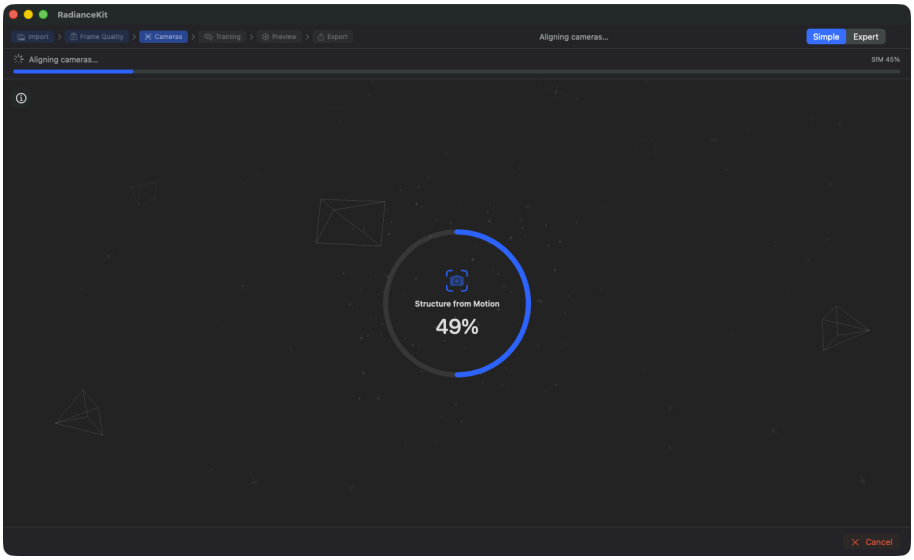

Abbildung 36: Z2 SfM-Phase — Stage-Icon „Structure from Motion„ mit 41 % im großen Kreis, oberer Statusbalken bei „SfM 25 %“, Cancel-Button unten rechts

SfM-Phase (Kameras werden ausgerichtet): Großer Fortschrittskreis zeigt Sub-Stage-Progress (hier 41 % der laufenden Apple-Photogrammetry-Session). Status-Text „Aligning cameras…„ oben links. Crumb-Trail markiert „Cameras“ als aktive Stufe. Oberer Statusbalken zeigt Pipeline-Gesamtprogress (25 %) — SfM belegt die erste Hälfte des Balkens. Schwebende Wireframe-Kameras im Hintergrund deuten an, dass Posen geschätzt werden.

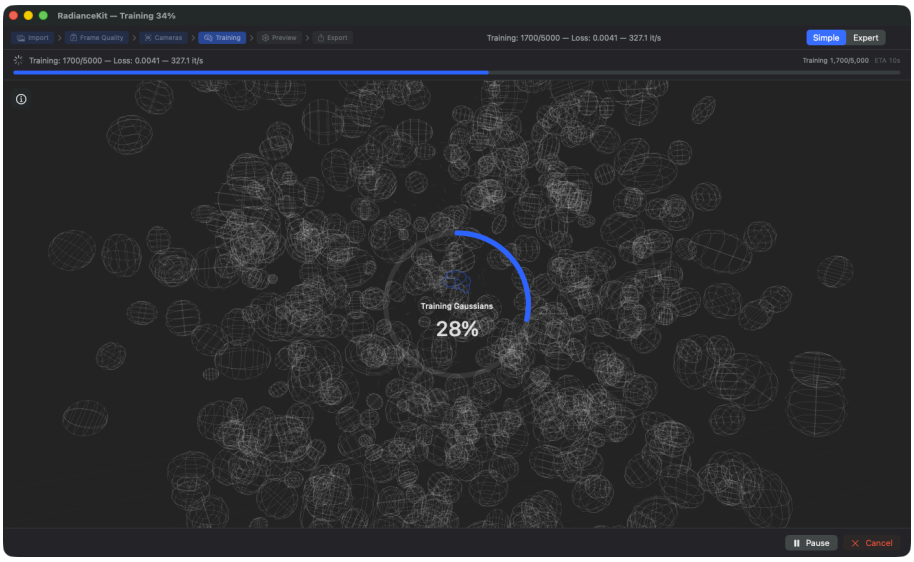

Abbildung 37: Z2 Training-Phase — Stage-Icon „Training Gaussians„ mit 6 %, Live-Metriken oben (Training: 400/5000 — Loss: 0.1642 — 138.7 it/s), ETA 33s, Pause/Cancel unten

Training-Phase (Gaussians werden optimiert): Sub-Stage-Icon wechselt zu „Training Gaussians„, Prozent zählt Iterationen vom gewählten Preset (hier 400 / 5 000 für Preview-Preset = 8 % der Stage). Live-Metrik-Zeile zeigt Loss-Wert (0.1642), Iterationen-pro-Sekunde (138.7 it/s) und ETA (33 s). Pipeline-Gesamtprogress klettert von 50 % bis

100 % während dieser Phase. Pause-Button (statt Cancel-Only in SfM-Phase) erlaubt Resume später; Cancel verwirft das Training-Ergebnis und kehrt zurück nach Z1.

Sobald die Pipeline läuft, blendet die App das Import-Overlay aus und zeigt einen vollflächigen Verarbeitungsbildschirm. Mittig läuft ein großer Fortschrittskreis (220 × 220 Pixel) mit Stage-Icon, Status-Text und Prozent-Zahl; im Hintergrund visualisiert eine dezente Splat-Animation symbolisch die laufende Berechnung. Oben links lässt sich ein Info-Panel einblenden, das Live-Metriken aus Training und SfM zeigt. Unten gibt es Pause/Resume, Cancel und im Fehlerfall einen Retry-Button.

C-18 SplatTrainingView (Hintergrund-Animation)

WO

Vollflächiger Hintergrund hinter dem Progress-Kreis, ausgeblendet bei Abbruch oder Fehler.

TECHNISCH

Dekorative Animation, die je nach Pipeline-Fortschritt (0…1) eine zunehmende Anzahl kleiner animierter Splat-Partikel rendert. Die Quelle ist ein berechneter Fortschrittswert, der SfM-Phasen auf 0–0.2 abbildet und Training auf 0.2–1.0 (Frame-Quality auf 0–0.05). Damit „bauen„ sich die Splats sichtbar auf, während das Training läuft. Ausschließlich dekorativ — die Anzeige zeigt keine echten Zwischenergebnisse des aktuellen Trainings (das wäre Live-Preview im Expert-Modus). Bei Cancel oder Failure wird sie ausgeblendet und nur der Status-Kreis bleibt sichtbar.

EINFACH GESAGT

Im Hintergrund läuft eine kleine Animation aus tanzenden Punkten, damit der Bildschirm nicht so leer wirkt während der Berechnung. Das ist nicht dein echtes 3D-Modell — das siehst du erst nach dem Training in Schritt Z3. Die Animation hat aber dieselbe Tonalität, sodass du am ungefähren Verdichtungsgrad ablesen kannst, wie weit das Training fortgeschritten ist. Anfangs sind nur wenige Punkte sichtbar, gegen Ende füllt sich der Hintergrund deutlich dichter — ein hübscher visueller Indikator zusätzlich zur Prozent-Anzeige im Kreis. Wenn dich die Animation stört (z. B. weil du im Hintergrund nebenher arbeiten willst), kannst du in den Expert-Modus wechseln, wo sie wegfällt.

C–19 Großer Progress-Kreis

WO

Mittig auf dem Verarbeitungsbildschirm, 220 × 220 Pixel.

TECHNISCH

Zwei übereinander gerenderte Ringe: außen ein gedämpfter Track-Ring, innen ein gefüllter Fortschritts-Ring mit Akzent- oder Rot-Stroke (rot bei Fehler). Innerhalb des Kreises ein Stage-Icon (Gehirn für Training, Kamera für SfM, Film für Video-Frame-Extraktion, Sparkles für Frame-Quality), Stage-Titel und die live animierte Prozent-Zahl in 32-Punkt-Rounded-Font. Das Icon pulsiert sanft, solange die Verarbeitung aktiv ist. Die Anzeige interpoliert auf einem 30-Hz-Timer sanft in Richtung des aktuellen echten Fortschritts — mit Constant-Creep (0.0003/Frame) plus Proportional-Anteil (4 % des Gaps) und einem Soft-Ceiling, das auf 80 % des nächsten erwarteten Milestone setzt (für SfM aus einer hartkodierten Milestone-Tabelle). So wirkt der Fortschritt flüssig, selbst wenn die echten SfM-Updates nur alle paar Sekunden eintreffen.

EINFACH GESAGT

Der große Kreis in der Mitte ist deine Haupt-Anzeige während die App rechnet. Er füllt sich sanft, auch wenn die echten Berechnungs-Updates nur alle paar Sekunden kommen — das gibt dir das Gefühl, dass etwas passiert, statt minutenlang auf ein eingefrorenes Prozent zu starren. Das Symbol in der Mitte wechselt, je nachdem ob gerade Frames extrahiert werden (Film-Icon), Kameras ausgerichtet werden (Kamera-Icon), oder Gaussians trainiert werden (Gehirn-Icon). Die Prozent-Zahl bezieht sich auf den aktuellen Teilschritt — die Gesamt-Pipeline siehst du in der schmalen Leiste ganz oben. Bei einem Fehler färbt sich der Ring rot statt blau, und das Icon pulsiert nicht mehr, sodass du sofort merkst, dass etwas schiefgelaufen ist.

C–22 Info Button (Metriken einblenden)

WO

Oben links auf dem Verarbeitungsbildschirm, 32 × 32 Pixel.

TECHNISCH

Schlichter Button mit Material-Hintergrund. Schaltet das Info-Panel ein oder aus. Icon wechselt zwischen Info-Kreis-Outline und Info-Kreis-Gefüllt, wenn aktiv. Sanfte Einblend-Animation. Im Tooltip „Show detailed processing metrics„.

EINFACH GESAGT

Standardmäßig ist der Bildschirm bewusst aufgeräumt — nur der große Fortschrittskreis, mehr siehst du erstmal nicht. Wenn du als technisch interessierter Nutzer genauer wissen willst, was passiert (welche Iteration, wie hoch der Loss, wie viele Gaussians), klick auf das i-Symbol oben links. Ein kleines Panel klappt unten auf und zeigt alle Live-Werte. Ein erneuter Klick blendet es wieder aus. Die Einstellung ist nicht persistent — bei jedem neuen Trainingslauf ist das Panel zunächst wieder ausgeblendet, was bewusst so gewählt ist, um Einsteiger nicht zu erschrecken.

C-23 Info Panel (Live-Metriken)

WO

Unten links auf dem Verarbeitungsbildschirm, sichtbar nur wenn `showProcessingInfo == true`.

TECHNISCH

Zwei-spaltiges Panel mit Ultra-Thin-Material-Background. Linke Spalte: stage-spezifische Info-Zeilen — für SfM Status-Text und Prozent; für Training Iteration, kombinierter Loss, L1-Loss, D-SSIM-Loss, Gaussian-Count (orange gefärbt), Speed (it/s), Elapsed-Time, berechnete ETA, SH-Degree und Learning-Rate. Rechte Spalte: Status-Text, Time-Info-String, inline Loss-Chart (siehe C-28) und ein Discoverability-Nudge (siehe C-32). Alle Werte werden aus dem Trainings-Status gelesen, der bei jedem Trainings-Tick aktualisiert wird.

EINFACH GESAGT

Das Info-Panel zeigt alle Live-Werte, die im Expert-Modus dauerhaft in der Inspector-Sidebar stehen würden: aktuelle Iteration, Loss-Wert (kleiner = besser), Anzahl der Gaussians, Geschwindigkeit, geschätzte Restzeit, SH-Degree und Learning-Rate. Auf der rechten Seite läuft zusätzlich eine winzige Loss-Kurve mit, die dir auf einen Blick verrät, ob das Training in die richtige Richtung läuft. Wenn das Training zäh wirkt, hilft ein Blick hier — ein Loss, der nicht mehr fällt, oder eine ETA, die nicht mehr sinkt, deuten auf Probleme hin. Wenn der Loss explodiert (plötzlich riesig wird) oder NaN anzeigt, ist das Training instabil geworden und ein Cancel + Retry oder Wechsel auf ein anderes Preset ist sinnvoll.

C-25 Pause/Resume Button

WO

Untere Navigationsleiste, sichtbar nur während der Trainings-Stage (NICHT während SfM) und solange die Verarbeitung läuft.

TECHNISCH

Bordered Button. Ruft je nach Status Pause oder Resume auf. Label wechselt zwischen „Pause„ (mit Pause-Icon) und „Resume“ (Play-Icon). Während des SfM-Schritts wird der Button nicht gezeigt, weil Apple Photogrammetry keine Pause-Semantik kennt. Der Pause-Zustand erhält Iteration, Gaussian-Status und Optimizer-Momentum komplett — Resume macht da weiter, wo zuvor angehalten wurde.

EINFACH GESAGT

Während das Training läuft, kannst du es jederzeit anhalten und später fortsetzen. Sinnvoll, wenn du zwischendurch etwas anderes auf dem Mac machen willst, das viel GPU braucht — z. B. Videoschnitt, Spieletest oder ein Render-Export aus einer anderen App. Klick Pause, mach dein Ding, klick Resume, das Training läuft genau dort weiter, wo es war. Iterations-Zähler, Gaussian-Anzahl und Optimizer-Momentum bleiben dabei vollständig erhalten, der Pause-State kostet dich nichts an Qualität. Während der SfM-Phase ist Pause nicht verfügbar — Apple Photogrammetry kennt keine Anhalte-Funktion, da musst du im Notfall mit Cancel arbeiten.

C–26 Cancel Button

WO

Untere Navigationsleiste, sichtbar während die Verarbeitung läuft (SfM oder Training).

TECHNISCH

Roter Bordered-Button. Öffnet einen Bestätigungs-Dialog mit Titel „Stop and discard progress?„, Buttons „Discard Progress“ (zerstörend) und „Keep Running„ (Cancel). Auf Bestätigung wird das Cancel-Flag gesetzt, der Trainings-Task beendet, der SfM-Subprozess wenn nötig beendet und eine Summary-Zeile mit Abbruch-Status in das JSONL-Log geschrieben. Im Gegensatz zu Pause werden Trainings-Buffer und Status verworfen.

EINFACH GESAGT

Der Abbruch-Knopf. Im Gegensatz zu Pause ist das endgültig — wenn du danach neu starten willst, läuft die Verarbeitung von vorne, sämtliche bereits trainierten Iterationen sind verloren. Sinnvoll, wenn du dich beim Preset vertan hast, das Training viel zu langsam läuft, oder die App offensichtlich Müll-Ergebnisse produziert und du nicht warten willst. Vor dem tatsächlichen Abbruch fragt die App über einen Bestätigungs-Dialog nochmal nach, damit du nicht aus Versehen Stunden Rechenzeit verlierst. Falls du nur kurz unterbrechen willst, nimm lieber Pause.

C–27 Retry Button

WO

Untere Navigationsleiste, sichtbar wenn die Pipeline fehlgeschlagen ist (SfM-Status startet mit „SfM failed„ oder Training ist im Fehler-Zustand).

TECHNISCH

Akzent-Button. Startet die gesamte Pipeline neu. Vor dem Start wird geprüft, ob noch importierte Bilder/Videos vorhanden sind. Vorherige Fehler-Logs bleiben im JSONL-Verzeichnis erhalten; ein neuer Run schreibt eine neue Logdatei mit aktuellem Timestamp.

EINFACH GESAGT

Falls SfM oder Training mit einer Fehlermeldung abbricht, kannst du es hier neu versuchen. Manchmal hilft das, weil viele Schritte (RANSAC, Densification) Zufallsanteile haben und ein zweiter Anlauf erfolgreich sein kann, wo der erste gescheitert ist. Die gesamte Pipeline läuft dann erneut von vorne durch — SfM und Training, in einer frischen JSONL-Log-Datei. Wenn auch der zweite Versuch fehlschlägt, sind meistens die Eingangsbilder das Problem (zu wenige, zu wenig Überlapp, Bewegungsunschärfe, schlechtes Licht); geh dann zurück mit Back und tausch dein Material aus. Tipp: schau parallel in die Training-Logs (Help → Open Training Logs), dort steht detaillierter, an welcher Stelle es konkret hakte.

C-28 Inline Loss Chart

WO

Im Info-Panel, rechte Spalte, sichtbar nur während Training mit nicht-leerer Verlaufs-Historie.

TECHNISCH

Kompakter Zeichen-Bereich (40 Pixel hoch), zeichnet die Loss-History als 1-Pixel-Linie in Akzent-Farbe. Daten werden auf finite Werte gefiltert (NaN-Schutz für unstabile Trainings). Min/Max werden über die gesamte History berechnet — der Chart auto-zoomt also auf den Wertebereich. Der letzte Loss-Wert steht rechts oben über dem Chart. Die History selbst wird im App-Zustand bei jedem Trainings-Tick aufgebaut (typisch alle 100 Iterationen).

EINFACH GESAGT

Eine winzige Loss-Kurve, die dir auf einen Blick zeigt, ob das Training „konvergiert„ (Linie fällt nach rechts) oder ob es feststeckt oder explodiert (Linie flach oder steigt). Bei einem gesunden Training fällt die Linie zu Beginn steil und flacht dann ab — das ist der erwartete Verlauf, ähnlich einer Halbierungskurve. Der Chart zoomt automatisch auf den aktuellen Wertebereich, sodass auch kleine Verbesserungen am Ende des Trainings sichtbar bleiben. Wenn die Linie plötzlich nach oben schießt oder einfriert, ist das ein gutes Signal, dass etwas schiefläuft — entweder das Material ist problematisch oder ein anderes Preset wäre besser geeignet. Den Chart findest du im Info-Panel, das du oben links mit dem i-Symbol einblendest.

C-32 Discoverability Nudge (Expert-Mode-Hinweis)

WO

Im Info-Panel, rechte Spalte unten, sichtbar nur während Training UND im Einsteiger-Modus.

TECHNISCH

Kleine Zeile mit Augen-Icon und Caption-Text „Switch to Expert Mode (⌘2) for live splat preview„, in zurückhaltender Tönung und 10-Punkt-Schrift. Kein interaktives Element, nur Hinweis. Reagiert nicht auf Klick — der User muss tatsächlich Cmd+2 drücken oder das Menü Mode → Expert Mode anklicken.

EINFACH GESAGT

Ein dezenter Hinweis, dass im Expert-Modus während des Trainings die aktuelle Zwischenversion deines 3D-Modells live im Viewport zu sehen ist. Im Einsteiger-Modus wird das absichtlich ausgeblendet, um die Oberfläche ruhig zu halten — aber viele User wissen gar nicht, dass es diese Funktion gibt, also weisen wir hier sanft darauf hin. Drücke Cmd+2 und das Training läuft im Hintergrund weiter, während du dabei zuschauen kannst, wie sich dein Modell vor deinen Augen zusammensetzt. Das ist auch ein gutes Werkzeug, um schon nach wenigen tausend Iterationen abzuschätzen, ob das Ergebnis was wird, oder ob du lieber abbrichst und neu anfängst. Cmd+1 bringt dich jederzeit wieder zurück in die Einsteiger-Ansicht.

Wann zur nächsten Stufe?

Die App wechselt automatisch in Z3 (Vorschau), sobald das Training erfolgreich abgeschlossen ist — du musst nichts klicken. Die untere Navigationsleiste wechselt dann von Pause/Cancel auf einen Back-Button (zurück zu Import) und einen Export-Button (vorwärts zu Export). Im Fehlerfall (rote Fehlermeldung, Stage-Icon ist X) erscheint stattdessen Retry, und du musst entscheiden, ob du nochmal startest oder mit Back zum Import zurückgehst, um Bildmaterial zu ändern.

Z3 — Vorschau (3D-Modell drehen)

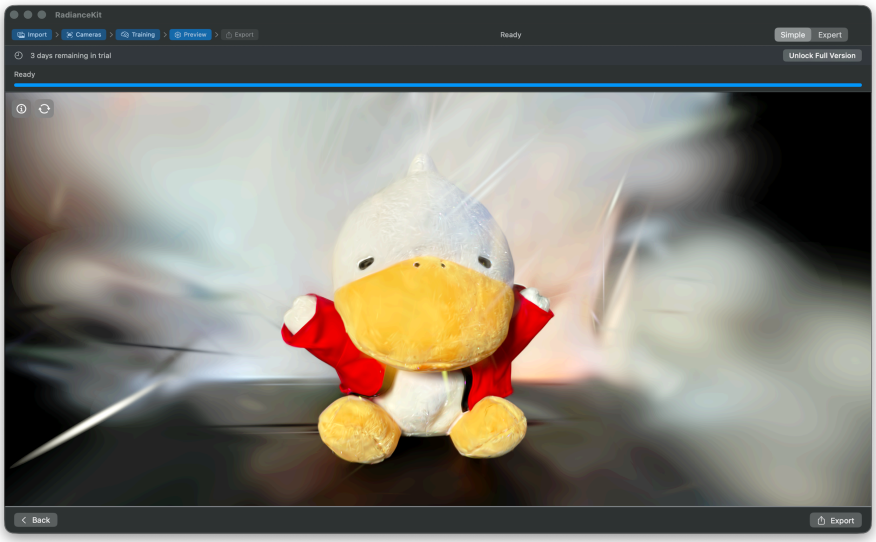

Abbildung 38: Simple-Modus Preview-Schritt mit 3D-Viewer

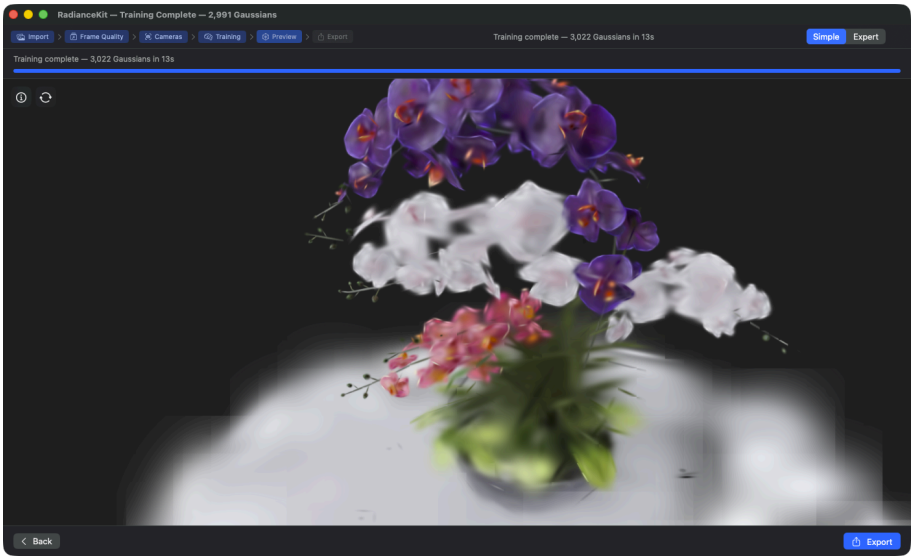

Abbildung 39: Z3 Preview nach Training-Abschluss — Bjoerns Blender-Bouquet rekonstruiert, Header zeigt „Training complete — 3,022 Gaussians in 13s„, Back- und Export-Buttons unten

WAS IM BILD ZU SEHEN IST Crumb-Trail markiert „Preview„ als aktive Stufe. Vollflächiger 3D-Viewport rendert die fertig trainierte Bouquet-Szene (synthetisches Blender-Test-Set von Bjoern, 60-Frame-Subset aus 960 hemisphärischen Cams). Header-Statusbalken: „Training complete — 3 022 Gaussians in 13 s“ — gibt finale Gaussian-Anzahl und Trainingszeit. Drag im Viewport rotiert die Kamera (Yaw/Pitch); Scroll-Wheel zoomt entlang der View-Direction. „Back„-Button (unten links) kehrt zurück zu Z2 für Resume oder Re-Run; „Export“-Button (unten rechts, primary) navigiert weiter zu Z4.

Nach Abschluss des Trainings landet die App automatisch in der Vorschau. Hier siehst du dein fertiges Gaussian-Splatting-Modell in einer Fullscreen-Metal-Ansicht und kannst es mit Maus und Trackpad drehen, zoomen und schwenken. Auf der Oberseite

des Viewports liegt ein kleines Overlay mit Kamerasteuerung und Info — Auto-Rotation, Trainings-Statistik, Reset-Knopf. Vor dem nächsten Schritt (Export) bietet es sich an, das Modell aus verschiedenen Winkeln zu prüfen, um sicherzugehen, dass die Rekonstruktion sauber ist.

C-36 SplatViewportView (3D-Hauptansicht)

WO

Fullscreen-Hintergrund des Vorschau-Schritts.

TECHNISCH

Metal-basierter 3D-Viewport, der die fertige Punktwolke rendert. Der Renderer ist RadianceKits EIGENER ForwardPass-Rasterizer — derselbe, der die Splats schon während des Trainings rendert — also echtes WYSIWYG (was trainiert wird, wird genau so angezeigt und exportiert). Tile-basierte Rendering-Pipeline mit Order-Independent Transparency. Wenn der Renderer nicht initialisiert werden kann (z. B. weil Metal auf dem System nicht verfügbar ist), erscheint stattdessen ein schwarzer Hintergrund mit „Metal not available„-Text. Die Ansicht ignoriert die Safe-Area, sodass das Modell bis an die Fensterkante reicht.

EINFACH GESAGT

Der Hauptviewport. Hier siehst du dein fertiges 3D-Modell aus deinen Fotos rekonstruiert, gerendert auf der GPU in Echtzeit. Klick und zieh mit der linken Maustaste, um zu drehen. Scrollrad oder Trackpad-Geste mit zwei Fingern, um zu zoomen. Rechte Maustaste oder Cmd+Drag zum Schwenken. Das Modell besteht aus zigtausenden semi-transparenten 3D-Ellipsoiden („Gaussians„), die deine Szene fotorealistisch rekonstruieren — jedes einzelne hat eine Position, Ausrichtung, Form und Farbe, die das Training gelernt hat. Im seltenen Fall, dass dein Mac kein Metal unterstützt, siehst du stattdessen einen schwarzen Hintergrund mit einer Hinweismeldung — RadianceKit braucht zwingend eine Metal-fähige GPU.

C–37 CameraControlsOverlay (Steuerungs-Overlay)

WO

Über dem Viewport, schwebend.

TECHNISCH

Kompaktes UI-Overlay mit Buttons für Auto-Rotation (Turntable), Reset-Camera, Hintergrund-Auswahl (Gray/Black/White), Save-Screenshot, Toggle-Info-Panel. Bindet an die Kamera-Parameter (Distanz, Azimut, Elevation, Target, FOV) und steuert das Auto-Turntable. Während des Trainings (wenn der User im Expert-Modus den Viewport mit-laufen sehen will) zeigt das Overlay zusätzlich eine kompakte Trainings-Status-Zeile.

EINFACH GESAGT

Die kleine Schwebeleiste über dem Modell. Hier startest du die Auto-Rotation (das Modell dreht sich von alleine, gut für Screenshots und kurze Demos), setzt die Kamera per Reset zurück auf die Anfangsposition (falls du dich verirrt hast), wechselst den Hintergrund (Grau für neutral, Schwarz für maximalen Kontrast, Weiß für helle Modelle), und machst direkt Screenshots, die unter /Pictures gespeichert werden. Praktisch, wenn du ein bestimmtes Detail aus einem ganz bestimmten Winkel zeigen willst, ohne extra das ganze Modell zu exportieren. Die Auto-Rotation ist auch ein guter Test, ob das Modell von allen Seiten gleich gut aussieht oder ob es eine „Schmuddelseite„ gibt, die durch fehlende Aufnahmen entstanden ist.

C–38 Export Button (Navigationsleiste)

WO

Untere Navigationsleiste in Z3.

TECHNISCH

Akzent-Button mit Label „Export„ und Share-Icon. Klick triggert den Wechsel zu Z4. Vorher prüft die übergeordnete Ansicht, ob die Vollversion freigeschaltet ist — wenn nicht, wird statt der Export-Bühne die Sperr-Ansicht gezeigt (siehe U-06).

EINFACH GESAGT

Wenn du mit dem Ergebnis zufrieden bist, klick Export und du landest im letzten Schritt, wo du das Format auswählst und speicherst. Ohne gekaufte Vollversion landest du stattdessen auf einer Bildschirmsperre mit Unlock-Hinweis und Kauf-Button — die App will dir keine Vollversion in die Schuhe schieben, aber der Export ist eines der Premium-Features. Sobald du den Kauf abgeschlossen hast, läuft die App direkt im freigeschalteten Zustand weiter und du landest in der gewohnten Export-Bühne. Wenn du es dir doch anders überlegst, kommst du über den Back-Button wieder in die Vorschau und kannst weiter am Modell drehen.

Wann zur nächsten Stufe?

Bevor du exportierst, dreh das Modell einmal komplett herum und prüfe: Sind alle Bereiche, die du in deinen Eingangsbildern abgedeckt hast, vorhanden? Gibt es schwebende „Floater„ (frei in der Luft schwebende Gauss-Splat-Wolken)? Wirkt der Hintergrund/Himmel sauber oder verschmiert? Schwerwiegende Probleme lassen sich nur durch Neu-Training fixieren — entweder mit mehr Bildern, anderem Preset, oder im Expert-Modus mit Floater-Reduction-Settings.

Z4 — Export (Format wählen & speichern)

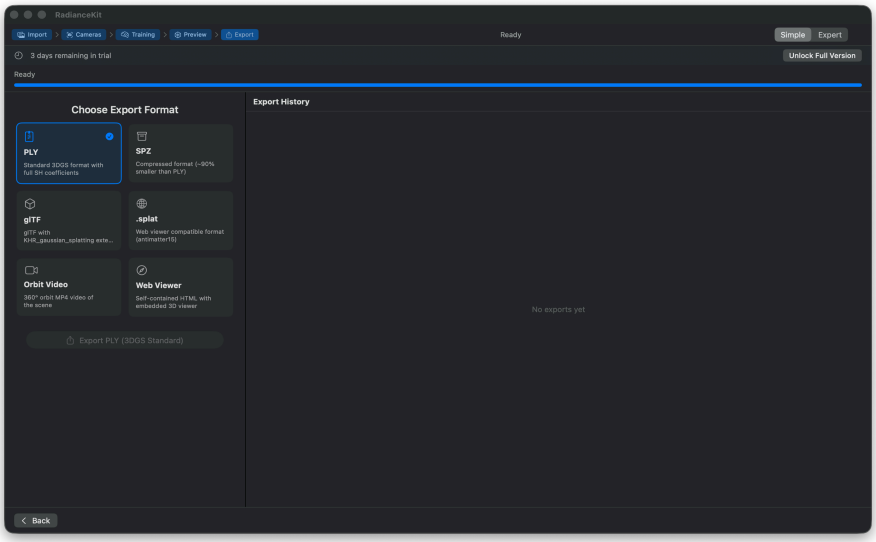

Abbildung 40: Simple-Modus Export-Schritt mit Format-Karten

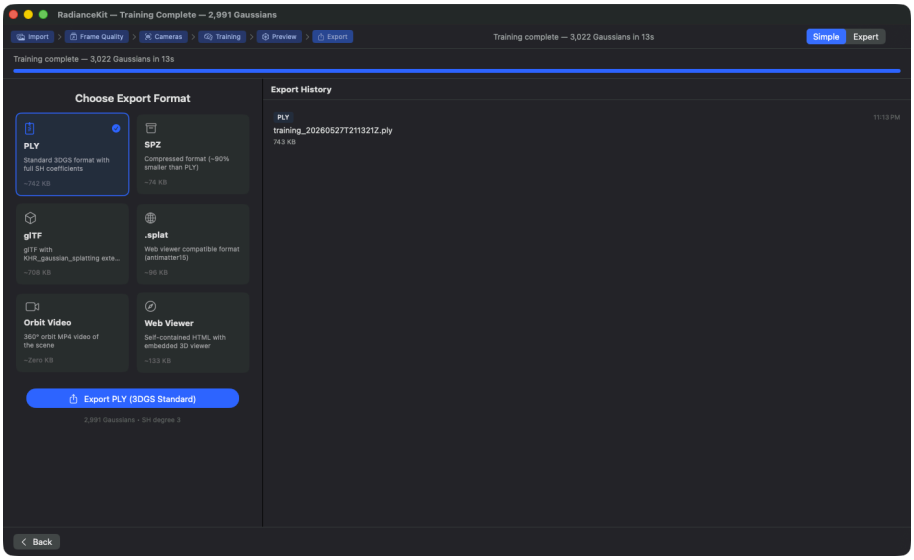

Abbildung 41: Z4 Export-Karten — 6 Formate (PLY 742 KB ausgewählt, SPZ 74 KB, glTF 708 KB, .splat 96 KB, Orbit Video, Web Viewer 133 KB), Export-History-Sidebar rechts mit bereits exportiertem PLY

WAS IM BILD ZU SEHEN IST Crumb-Trail markiert „Export„ als aktive Stufe. Linkes Karten-Grid „Choose Export Format“ mit allen sechs Optionen: PLY (Standard-3DGS, 742 KB, mit vollen SH-Koeffizienten — hier vorausgewählt mit blauem Häkchen), SPZ (komprimiertes 3DGS-Format, ~90 % kleiner als PLY, 74 KB), glTF (mit `KHR_gaussian_splatting`-Extension, 708 KB), .splat (Web-Viewer-kompatibel via antimatter15, 96 KB), Orbit Video (360°-MP4 der Szene, Live-Größenberechnung), Web Viewer (eigenständiges HTML mit eingebettetem 3D-Viewer, 133 KB). Größenangaben werden live aus dem aktuellen Gaussian-Count und Format-Overhead berechnet. Rechts „Export History„ listet bereits abgeschlossene Exporte mit Format-Pill, Dateiname und Zeitstempel

— Click reveals im Finder. Primary-CTA unten links: „Export PLY (3DGS Standard)“ mit Gaussian-Subtitle „2,991 Gaussians · SH degree 3“.

Im letzten Schritt wählst du aus 6 Export-Formaten (PLY, SPZ, glTF, .splat, Orbit-Video, Web-Viewer) per 2-spaltiges Karten-Grid, klickst Export und wählst Speicherort im macOS-Dialog. Rechts läuft eine History aller bisherigen Exports — bei der Karten-Auswahl wird unter jeder Karte sofort der geschätzte Dateigröße angezeigt, sodass du z. B. SPZ bevorzugst, wenn du ins Web willst (klein), und PLY, wenn du in eine andere Software (SuperSplat, Postshot, Blender via Plugin) importieren willst (groß und vollständig).

C-39 2-Column Format Grid

WO

Linke Hauptseite des Export-Schritts.

TECHNISCH

Karten-Raster mit zwei flexiblen Spalten und 12 Punkt Abstand. Iteriert über die im Einsteiger-Modus angebotenen Formate — eine gefilterte Teilmenge der vollen Formatliste, die nur die 6 wichtigsten Formate enthält: PLY, SPZ, glTF, .splat, Orbit-Video, Web-Viewer. Compressed-PLY und SOG werden NUR im Expert-Modus angeboten.

EINFACH GESAGT

Ein Karten-Raster mit den 6 Formaten, die im Einsteiger-Modus relevant sind: PLY (Standardformat für andere 3D-Tools), SPZ (komprimierte Variante für Web), glTF (offizieller Web3D-Standard), .splat (für antimatter15-Web-Viewer), Orbit-Video (fertiges MP4 zum Vorzeigen), und Web Viewer (eigenständige HTML-Datei mit eingebettetem 3D-Player). Damit deckst du 90 % der Anwendungsfälle ab. Wenn du eines der weniger gängigen Formate brauchst (Compressed-PLY oder SOG für extreme Komprimierung), wechsle in den Expert-Modus, dort sind alle 8 Formate verfügbar. Die kompakte Auswahl hier ist bewusst, damit Einsteiger nicht von der Vielfalt überfordert werden.

C-40 Format Card Button

WO

Jede Karte im Grid.

TECHNISCH

Schlichter Button mit Karten-Layout: Icon (z. B. Dokument-Zipper für PLY, Archivbox für SPZ, Video-Icon für Orbit-Video) oben, Format-Name als Headline, Beschreibungs-Caption (2-zeilig gekürzt), darunter die geschätzte Dateigröße (live aus Format, Gaussian-Count und SH-Degree berechnet und als KB/MB formatiert). Bei Klick wird das Format ausgewählt. Selektierte Karte bekommt Akzent-Hintergrund, Akzent-Border und ein Häkchen-Icon rechts oben. Tooltip ist die Format-Beschreibung.

EINFACH GESAGT

Eine Karte pro Format. Klick eine an, sie wird mit Akzent-Farbe und einem Häkchen hervorgehoben, und der Export-Button darunter passt seinen Text an („Export PLY„, „Export SPZ“, etc.). Jede Karte zeigt ein passendes Symbol, den Namen, eine zweizeilige Kurz-Erklärung und die geschätzte Dateigröße bei deinem aktuellen Trainings-Ergebnis. Die Größe hilft dir, sinnvoll zu wählen — wenn du das Ergebnis per Email schicken willst, nimm die kleinste Variante (meist SPZ oder .splat); wenn du in einer anderen 3D-Software weiterarbeiten willst, nimm die mit der besten Kompatibilität (typischerweise PLY). Beim Hovern über einer Karte zeigt der Tooltip eine ausführlichere Beschreibung, falls du die Unterschiede zwischen den Formaten unklar findest.

C-41 Video Duration Slider

WO

Unter dem Format-Grid, sichtbar nur wenn ein Video-Format gewählt ist (Orbit-Video oder Social-Video).

TECHNISCH

Schieberegler 3–30 Sekunden in 1-Sekunden-Schritten, bindet an die Video-Länge im App-Zustand. Maximal-Breite 300 Pixel. Wird nur eingeblendet, wenn ein Video-Format ausgewählt ist. Bei nicht-Video-Formaten wird der Schieberegler komplett aus der Ansicht entfernt — kein toter Platz.

EINFACH GESAGT

Wenn du ein Orbit-Video als Export wählst, kannst du hier die Länge bestimmen. 3 Sekunden = sehr schnelle Drehung, 30 Sekunden = langsame, ruhige Drehung um dein Modell. Für Social-Media-Reels (Instagram, TikTok) ist meistens 6–10 Sekunden ideal — lang genug, um das Modell zu zeigen, kurz genug, dass die Zuschauer nicht abspringen. Bei Präsentationen oder Portfolio-Videos darfst du gerne 15–20 Sekunden nehmen. Der Slider taucht nur auf, wenn ein Video-Format ausgewählt ist; bei Datei-Formaten wie PLY oder SPZ wäre er sinnlos und ist ausgeblendet.

C–42 Export Button

WO

Unter dem Format-Grid (und unter dem Duration-Slider, falls Video gewählt).

TECHNISCH

Großer Akzent-Button. Label: „Export {Format-Name}„, Share-Icon. Bei Klick wird der macOS-Speichern-Dialog mit Format-passender Endung und Default-Filename „scene.{ext}“ geöffnet; bei Bestätigung wird der Export an die gewählte URL geschrieben. Deaktiviert, wenn kein Trainings-Ergebnis vorhanden ist oder ein Export bereits läuft.

EINFACH GESAGT

Klick, wähle Speicherort im macOS-Dialog, fertig — die App schreibt die Datei im gewählten Format an die ausgewählte Stelle. Default-Name ist „scene.{Endung}„ (z. B. „scene.ply“ oder „scene.spz„), den kannst du im Dialog beliebig ändern, bevor du speicherst. Der Button ist grau, solange noch kein Training-Ergebnis vorhanden ist (sollte hier nie passieren, weil du sonst gar nicht im Export-Schritt wärst) oder ein anderer Export bereits läuft. Sobald der Export läuft, erscheint darunter eine Fortschrittsanzeige; die App bleibt bedienbar, du kannst also schon den nächsten Export vorbereiten.

C–43 Export Progress Bar

WO

Unter dem Export-Button, sichtbar nur während ein Export läuft.

TECHNISCH

Fortschrittsanzeige mit Max-Breite 300 Pixel, darunter Caption „Exporting… N %„. Der Wert läuft von 0 bis 1 und wird während des Schreibens aktualisiert — bei PLY in Chunks von 10 000 Gaussians, bei SPZ einmalig nach Quantisierung, bei Orbit-Video in Frame-Intervallen.

EINFACH GESAGT

Während der Export läuft, siehst du hier den Fortschritt als schmalen Balken plus Prozent-Anzeige. PLY ist meistens binnen Sekunden fertig, weil die Datei einfach binär weggeschrieben wird. SPZ braucht etwas länger, weil die Daten dabei quantisiert und komprimiert werden. Orbit-Video ist der zeitaufwendigste Export — hier wird jedes einzelne Frame neu gerendert; je nach Auflösung und Länge kann das eine Minute oder länger dauern. Während des Exports bleibt die App bedienbar, du kannst also schon das nächste Format vorbereiten oder im Viewport weiterklicken.

C-44 Export Error Display

WO

Unter der Progress-Bar, sichtbar nur wenn beim letzten Export ein Fehler aufgetreten ist.

TECHNISCH

Rote Zeile mit Warning-Icon und Fehlertext. Rote 8 %-Hintergrund-Opacity, abgerundete Ecken. Max-Breite 400 Pixel. Häufige Fehlerursachen: SOG erwartet `cwebp` im System-PATH (nicht App-Store-konform); Schreibfehler bei vollem Plattenspeicher; Sandbox-Fehler bei Speicherzielen außerhalb des erlaubten Bereichs.

EINFACH GESAGT

Wenn der Export schiefgeht, erscheint hier in Rot eine kurze Klartext-Beschreibung des Problems. Meistens ist die Ursache offensichtlich — kein Platz auf der Platte, keine Schreibrechte für den Zielordner, oder ein Zielort außerhalb der Sandbox-erlaubten Bereiche. Speziell beim SOG-Format kommt es vor, dass `cwebp` im System fehlt; in diesem Fall ist SOG nicht nutzbar und du musst auf SPZ ausweichen. Wenn die Fehlermeldung unklar ist, schau in das Log-Verzeichnis (Help → Open Training Logs), dort steht ausführlicher, was schiefgegangen ist. Im Zweifel hilft es, einfach einen anderen Speicherort zu wählen — z. B. den Schreibtisch.

C-46 Export History List

WO

Rechte Seite des Export-Schritts.

TECHNISCH

Liste über die Export-Historie (persistent als JSON in den UserDefaults gespeichert, nach jedem erfolgreichen Export gepflegt). Jede Zeile zeigt Format-Badge (klein, akzentfarbig), Timestamp (HH:mm), Dateiname (1 Zeile gekürzt) und formatierte Dateigröße. Klick auf eine Zeile öffnet Finder mit selektierter Datei. Empty-State: „No exports yet„.

EINFACH GESAGT

Eine Liste deiner bisherigen Exports — Format, Uhrzeit, Dateiname, Größe, in chronologischer Reihenfolge. Klick eine Zeile an und die Datei wird im Finder hervorgehoben angezeigt, ohne dass du selber durch Ordner navigieren musst. Praktisch, wenn du eine Stunde später nochmal den letzten Export brauchst und nicht mehr weißt, wo du ihn hingespeichert hast — die History merkt sich das. Wenn du noch nie etwas exportiert hast, steht hier ein freundlicher Hinweis „No exports yet„. Die Liste übersteht Neustarts der App, weil sie in den UserDefaults gespeichert ist.

C–48 History Context Menu (Rechtsklick)

WO

Rechtsklick auf eine History-Zeile.

TECHNISCH

Kontextmenü auf jedem Listen-Eintrag mit zwei Aktionen: „Reveal in Finder„ (öffnet Finder mit selektierter Datei, wie der Einfach-Klick) und „Copy Path“ (legt den vollen Dateipfad als Text in die Zwischenablage). Letzteres ist nützlich für Drag-and-Drop in andere Apps oder zur Übergabe an die Kommandozeile.

EINFACH GESAGT

Rechtsklick auf einen History-Eintrag öffnet ein kleines Menü mit zwei Aktionen. „Reveal in Finder„ macht dasselbe wie ein normaler Klick — öffnet den Finder mit selektierter Datei, sodass du sie sofort siehst. „Copy Path“ legt den kompletten Dateipfad in die Zwischenablage, sodass du ihn z. B. in Terminal-Befehlen, in andere Apps oder in eine Notiz einfügen kannst. Besonders praktisch, wenn du den Export an jemanden weitergeben willst oder ihn in einem anderen Programm öffnen möchtest, das per Pfad-Eingabe arbeitet. Funktional ein kleines, aber hilfreiches Detail, das auf Mac-typische Bedienungs-Patterns setzt.

Wann ist der Workflow abgeschlossen?

Nach einem erfolgreichen Export hast du dein 3D-Modell als Datei auf der Platte und die History zeigt einen neuen Eintrag. Es gibt keinen „Done„-Button — du kannst beliebig viele Exports in unterschiedlichen Formaten anhängen, ohne neu zu trainieren. Wenn du zurück zur Vorschau willst (z. B. um nochmal eine Kameraperspektive zu prüfen), nutze den Back-Button in der unteren Navigationsleiste. Wenn du eine komplett neue Szene anfangen willst, gehe via Back bis Z1 und nutze dort Clear All, oder File → New Project (Cmd+⇧+N).

Wechsel zu Expert-Modus

Drücke jederzeit Cmd+2 oder wähle Mode → Expert Mode (`M8`). Der gesamte Zustand bleibt erhalten: importierte Bilder, gewähltes Preset, laufendes oder fertiges Training, fertige Punktwolke, Export-History, sogar die aktuelle Stage. Im Expert-Modus wird statt der vier-Schritt-Bühne das volle Inspector-Sidebar gezeigt mit allen ~150 Bedienfeldern. Insbesondere: der Project Navigator (siehe Kapitel 2) bietet die erweiterten Bild-Operationen (Minus-Button, Backspace-Delete, Cmd-Z-Undo, Quick-Look-Vorschau), die Live-Preview im Viewport während des Trainings, sowie alle Loss-, MCMC-, Densification- und Mip-Splatting-Parameter. Cmd+1 schaltet zurück in den Einsteiger-Modus — auch das verliert keinen Zustand.

Häufige Fragen

Warum bleibt mein Start-Processing-Button grau?

Du hast noch keine Bilder oder kein Video importiert. Zieh mindestens eine Datei in die Drop-Zone oder benutze „Browse Files„. Sobald die Bilder-Liste rechts mindestens einen Eintrag enthält, wird der Button aktiv. (Bei nur 1–2 Bildern startet er zwar, aber SfM bricht direkt mit Fehler ab — siehe das rote Validation-Banner.)

Warum ist mein Export-Button gesperrt?

Im Einsteiger-Modus gibt es zwei Stufen: (a) Wenn die Trainings-Pipeline noch nicht fertig ist und du keine hast, ist der Button deaktiviert — du musst erst Z2 abschließen. (b) Wenn du die Vollversion noch nicht gekauft hast (`PurchaseManager.hasAccess == false`), siehst du statt der Export-Bühne eine Sperr-Ansicht mit Schloss-Icon und „Unlock Full Version„-Button, der das Purchase-Sheet öffnet. Quick- und Preview-Presets erlauben Training kostenlos, aber Export ist Premium.

Warum kann ich kein Preset auswählen?

Du kannst es auswählen — aber wenn du ein Premium-Preset (Balanced, Quality, MCMC-Varianten) ohne gekaufte Vollversion antippst, springt der Picker automatisch zurück auf Preview und das Purchase-Sheet öffnet sich. Quick und Preview sind die einzigen kostenlos benutzbaren Presets.

Warum ist meine Drop-Zone leer und gestrichelt-grau, obwohl ich Bilder reinziehe?

Wahrscheinlich ein UTI-Typ-Mismatch. Die App akzeptiert JPG, PNG, TIFF, HEIC, MP4, MOV plus die App-eigenen Splat-Formate. Andere Bildformate (BMP, GIF, WebP, RAW-Formate) werden NICHT erkannt. Wenn du sicher bist, dass dein Bildtyp dabei sein müsste, prüfe die Dateinamens-Endung — die App geht primär nach Extension, nicht nach Datei-Inhalt.

Warum dauert SfM so lange, obwohl ich nur 30 Bilder habe?

Apple Photogrammetry skaliert nicht linear — bei manchen Bild-Konstellationen (Innenräume mit komplexen Texturen, Bewegungsunschärfe, schlechtes Licht) braucht es deutlich länger als das Bild-Count vermuten lässt. Wenn SfM nach 10+ Minuten bei 30 Bildern noch hängt, breche ab und versuche es nochmal mit besserem Material, oder wechsle in den Expert-Modus und probiere COLMAP/Native-SfM (Cmd+2 → Inspector → Camera Alignment).

Wo finde ich meine Training-Logs?

Help → Open Training Logs (Cmd+⇧+L). Das öffnet `~/Documents/RadianceKit/Logs/`. Jede Trainings-Session schreibt eine eigene JSONL-Datei mit Timestamp im Dateinamen — erste Zeile ist die Konfiguration, danach folgt eine Progress-Zeile alle 100 Iterationen, letzte Zeile ist die Summary mit Final-Loss und Success-Flag.

KOLOPHON

Gesetzt in SF Pro · Code in SF Mono · Typst 0.14 ·
22. June 2026

Made with ❤ in Unterhaching