TECHNISCHE UNIVERSITÄT WIEN

Diplomarbeit

Vergleich von 3D Rekonstruktionen mittels Neural Radiance Fields und Dense Image Matching

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Geodäsie und Geoinformation

eingereicht von

Markus Brezovsky BSc BA
Matrikelnummer 01429180

ausgeführt am Department für Geodäsie und Geoinformation
der Fakultät für Mathematik und Geoinformation
der Technischen Universität Wien

Betreuung
Betreuer: Univ.Prof. Dipl.-Ing. Dr.techn. Gottfried Mandlburger
Mitwirkung: -

Wien, am 18. November 2024

(Unterschrift VerfasserIn) (Unterschrift BetreuerIn)

Inhaltsverzeichnis

Tabellenverzeichnis

Abbildungsverzeichnis

1 Einleitung

Aufgrund der rasanten Entwicklungen in der Computer Vision und 3D-Modellierung eröffnen sich neue Möglichkeiten für die Erfassung der realen Welt. Diese Fortschritte wurden insbesondere durch die Weiterentwicklung von Hardwarekomponenten beschleunigt. Der Trend zum Einsatz von Machine Learning und Deep Learning Modellen brachte einen starken Anstieg des Grafikkartenmarktes mit sich. Die Verwendung von Grafikkarten zur Lösung von Problemstellungen ist heute gefragter denn je. Die Möglichkeit des Parallel Computing eröffnet Möglichkeiten, die vor einem Jahrzehnt undenkbar waren. Daten sind wichtiger denn je, und ihre Prozessierung und Aufbereitung erfordern neue Denkweisen und Lösungsansätze. Dadurch konnten auch Computer Vision Algorithmen profitieren und neuwertige Ansätze für die Photogrammetrie hervorgebracht werden.
Insbesondere NeRFs (Neural Radiance Fields) haben sich in den letzten Jahren als vielversprechende Alternative zu DIM (Dense Image Matching) herauskristallisiert. NeRFs sind in der Lage, aus einer begrenzten Anzahl an 2D-Bildern fotorealistische 3D-Modelle des Aufnahmeobjekts zu generieren. Im Bereich der Geodäsie und Geoinformation sind die Einsatzmöglichkeiten der mitgelieferten Punktwolke von hoher Relevanz und daher stellt sich die Frage, wie NeRF-generierte Modelle hinsichtlich Genauigkeit, Vollständigkeit, Wirtschaftlichkeit und Präzision im Gegensatz zu etablierten photogrammetrischen Verfahren (DIM) abschneiden.
Auf Basis der Bilddaten einer Zenmuse P1 Kamera, die mit einer DJI M350 Drohne erfasst wurden, wurden qualitative sowie quantitative Untersuchungen zweier Aufnahmeobjekte angestellt. Einerseits wird ein durchlässiges Objekt (Baum/gruppe) untersucht, andererseits liegt der Fokus auf einem nicht durchlässigen Objekt (Kapelle).
Typisch für DIM Punktwolken ist, dass scharfe Kanten tendenziell ausgerundet werden. Die Frage, ob NeRFs im Sinne der Rekonstruktionsqualität bessere Ergebnisse erzielen, soll in dieser Arbeit beantwortet werden.
Darüber hinaus wird analysiert, wie sich unterschiedliche Aufnahmepositionen und Aufnahmerichtungen der Messkamera auf die resultierende Punktwolke auswirken. Hierfür werden unterschiedliche Flugroutinen eingesetzt. Als Referenz für die NeRF und DIM Lösung wird der Datensatz eines terrestrischen Laserscanners herangezogen.

1.1 Problemstellung

Im Zuge der Arbeit hat sich herausgestellt, dass NeRF-basierte Methoden mit gewissen Aufnahmekonfigurationen, die speziell im UAV-Mapping genutzt werden, vor Herausforderungen gestellt werden.
Zur Erfassung einer Topographie verfolgt man in der Praxis meistens einen Ansatz, bei dem das Interessengebiet in einem Grid-Muster beflogen wird. Die Kamera zeigt hier in Richtung Nadir und Fotos werden in vordefinierten Abständen mit einer gewissen Überlappung aufgenommen. Die Überlappung in Längs- und Querrichtung ermöglicht in der nachgehenden Auswertung eine Beziehung zwischen den Bildern herzustellen. Diese Aufgabe wird durch einen

Bündelblockausgleich optimal gelöst. Der Erfolg des Bündelblockausgleichs ist jedoch beschränkt durch die Wahl der Überlappung. Eine zu geringe Überlappung birgt Probleme bei der Prozessierung. Eine zu hohe Überlappung benötigt längere Durchlaufzeiten und es stellt sich die Frage der Wirtschaftlichkeit. Die erfassten Daten gelangen schnell in den zweistelligen bis dreistelligen GB [1]-Bereich.

NeRFs hingegen profitieren von der Vielseitigkeit der Aufnahmeposition und die Erfassung eines 3D-Modells kann bereits durch eine überschaubare Anzahl an Fotos erreicht werden. Probleme entstehen jedoch bei homogenen bzw. koplanaren Kamerapositionen. NeRFs bevorzugen ein gewisses Maß an Diversität innerhalb der Aufnahmekonfiguration, was durch unterschiedliche Blickpunkte zum Objekt gewährleistet wäre. Je unterschiedlicher die Positionen sind, desto besser kann das resultierende Modell verfeinert bzw. trainiert werden.

Bei der Analyse der Punktwolken aus NeRF, DIM und TLS ist es wichtig, die unterschiedlichen Punktdichten und die Aufnahmearten zu berücksichtigen, da diese Auswirkungen auf die Genauigkeit und Vergleichbarkeit der Ergebnisse haben. Die NeRF- und DIM-Punktwolken wurden aus einer luftgestützten Befliegung generiert, während die TLS-Punktwolke terrestrisch aufgenommen wurde. Diese unterschiedlichen Perspektiven und Aufnahmebedingungen führen dazu, dass in der TLS-Punktwolke, bestimmte Bereiche aufgrund von Sichtbeschränkungen oder ungünstigen Aufnahmewinkeln fehlen. Diese Lücken bzw. Scanschatten können die Analyse verzerren, da sie zu einer unvollständigen oder inkonsistenten Repräsentation der Struktur führen. Zusätzlich weisen die Punktwolken unterschiedliche Punktdichten auf. Bei der TLS-Punktwolke wird die Punktdichte direkt durch die Messung bestimmt und hängt vor allem von der Standpunktdichte sowie den Scanparametern ab. Im Gegensatz dazu entstehen die Punktwolken bei NeRF und DIM erst durch die Auswertung der aufgenommenen Daten, wobei die Punktdichte je nach Parametrisierung und Bildkonfiguration variieren kann. Hier spielen also sowohl die Rekonstruktionsmethode als auch die Bildaufnahmebedingungen eine entscheidende Rolle Speziell NeRF-generierte Punktwolken weisen aufgrund der neuronalen Netzarchitektur einen hohen Grad an Komplexität auf. Ein zu hohes Punktvolumen führt zu Artefakten, die später manuell bereinigt werden müssen, wobei eine zu geringe Punktdichte wichtige Details verschwinden lässt. Bei der Wahl des Punktvolumens muss somit eine Balance zwischen Rechenzeit, Detailgrad und Ausreißermenge gefunden werden. Diese Unterschiede in den Punktdichten stellen eine Herausforderung für nachfolgenden Benchmarks dar. Ungleichheiten in der Punktmenge bzw. Dichte kann zu Verzerrungen innerhalb der statistischen Auswertungen führen. Besonders Punkt-zu-Punkt-Vergleiche leiden unter unterschiedlichen Auflösungen.

Im Zuge der Auswertungen wurden einige Vor- und Nachteile von NeRFs entdeckt. Aufgrund der Homogenität der Aufnahmekonfiguration im Falle der linearen Baumbefliegung, konnte keine Punktwolke aus der Auswertesoftware (Nerfstudio) exportiert werden. Hier zeigen sich bereits erste Einschränkungen für Mapping Flüge, die anschließend mit NeRFs ausgewertet werden sollen.

Das Hauptaugenmerk dieser Arbeit liegt daher in dem eingangs erwähnten Kapellendatensatz.

[1] Orthophoto, Punktwolke, DEM, Fotos, Projektmetadaten

Da die Kapelle im Grundriss primär eine zylindrische Geometrie aufweist und das Dach halbkugelförmig ist, stellt sich die Frage geeigneter Parameter für die Wahl der Auswerthemethoden. Zum Beispiel werden bei der Segmentierung einer Punktwolke häufig Ebenenextraktionsmethoden eingesetzt, die iterativ Punkte zu einem Segment hinzufügen. Die meisten Algorithmen sind für topographische TLS-Punktwolken ausgelegt, wobei hier die Geometrien der Objekte[2]vermehrt Ebenen gleichen. Um die Eigenschaften des Kapellendatensatzes zu berücksichtigen, wurde nach geeigneten Parametern zur Ebenenextraktion gesucht. Durch Einschränkung des Suchradius und Anpassung der maximalen Standardabweichung konnten lokale Ebenen, trotz zylindrischer Geometrie, extrahiert werden.
Nähere Betrachtungen über die Präzision, Vollständigkeit und den optimalen Einsatz von NeRFs werden in der Arbeit erläutert und diskutiert.

1.2 Hintergrund und Motivation

Mit der Weiterentwicklung von UAVs und bildgebenden Sensoren ist es möglich, hochauflösende Luftbildern zu erfassen. Dadurch entstehen Möglichkeiten, die historisch betrachtet undenkbar waren. Als Beispiel sei hier die Rekonstruktion und Modellbildung von Topographien, oder die Erzeugung hochauflösender Orthophotos zu erwähnen.
Ein augenscheinliches Problem der DIM-Verfahren ist jedoch die Darstellung feiner dynamischer Strukturen. Baumkronen erscheinen oft als zusammenhängendes Objekt und strukturelle Unterschiede sind aus der Punktwolke nicht erkennbar.
Die Motivation für diese Arbeit entspringt aus der eingangs erwähnten Beobachtung, dass photogrammetrische Punktwolken bei der Abbildung von fein strukturierten Details an ihre Grenzen stoßen. Die Verwendung hochauflösender Kameras sichert die Erfassung von Details auf Pixelebene. Es wäre daher sinnvoll, den vollen Informationsgehalt dieser Systeme auszuschöpfen.
Der Fokus dieser Arbeit liegt in der Erfassung und Rekonstruktion von natürlichen und künstlichen Objekten aus Luftbildaufnahmen. Durch variierende Bedingungen sollen wertvolle Einblicke über das Thema NeRFs gegeben werden. Darüber hinaus sollen Stärken und Schwächen von NeRFs ausgearbeitet und deren Integration in den praktischen Workflow kritisch hinterfragt werden.

1.3 Zielsetzung und Forschungsfragen

Die Zielsetzung dieser Arbeit besteht darin, die Leistungsfähigkeit und Anwendbarkeit von Neural Radiance Fields (NeRFs) im Vergleich zu konventionellen Dense Image Matching (DIM)-Methoden für die 3D-Rekonstruktion in der Photogrammetrie zu evaluieren. Durch den Einsatz von hochauflösenden Luftbildern mittels einer DJI M350 Drohne und einer Zenmuse P1 Kamera wird die Genauigkeit, Detailtreue, Vollständigkeit und Effizienz beider Ansätze bei der Modellierung von natürlichen und künstlichen Strukturen anhand praktischer Beispiele untersucht.
Besonderes Augenmerk liegt auf der Analyse der Rekonstruktionsqualitat und der Fähigkeit, feine Strukturen zu erfassen, die mit herkömmlichen Methoden (DIM) schwer darstellbar sind.

[2]Dächer, Hauswände, Straßen

Durch den Vergleich unterschiedlicher Aufnahmepositionen soll zudem der Einfluss der Datenerfassungsmethodik auf die Ergebnisqualität ausgearbeitet werden.

Aus der genannten Zielsetzung resultieren folgende Forschungsfragen:

Genauigkeit und Präzision: Wie vergleichen sich NeRFs mit DIM in Bezug auf Genauigkeit und Präzision bei der 3D-Rekonstruktion?

Rekonstruktionsqualität: Können NeRFs Kanten und feine Strukturen besser rekonstruieren als DIM? Wo liegen die Unterschiede in der Rekonstruktion zwischen DIM und NeRFs?

Einfluss der Aufnahmeposition: Wie beeinflussen unterschiedliche Flugroutinen (linear vs. orbital) die Rekonstruktionsqualität und Detailgenauigkeit der erzeugten Modelle?

Darstellung feiner Strukturen: Wie performant sind NeRFs im Vergleich zu DIM bei der Darstellung und Differenzierung feiner Strukturen?

Integration in praktische Workflows: Welche Vorteile bieten NeRFs gegenüber DIM-basierten Verfahren im praktischen Einsatz? Wie kompatibel ist eine NeRF Punktwolke mit geodätischen Genauigkeitsansprüchen?

Um die Forschungsfragen zu beantworten wurde eine Mischung aus qualitativen und vermehrt quantitativen Methoden eingesetzt. Es wurden Punkt-zu-Punkt-Vergleiche durchgeführt und die Ergebnisse mit unterschiedlichen Metriken untermauert. Weiters wurden statistische Abstandsanalysen (M3C2) bzw. Plane-Matching Methoden eingeführt, um Unterschiede der Ansätze visuell hervorzuheben.
Die vorliegende Arbeit gliedert sich in insgesamt sieben Kapitel, welche die verschiedenen Aspekte von Neural Radiance Fields (NeRFs) im Vergleich zu konventionellen Dense Image Matching (DIM)-Methoden beleuchten. Die Arbeit untersucht dabei insbesondere die 3D-Rekonstruktion einer Kapelle in Mannersdorf an der March, wobei eine breite Palette an quantitativen Analysen eingesetzt wurde, um die gestellten Forschungsfragen zu beantworten.

Kapitel 1: Einleitung

Dieses Kapitel führt in das Thema der Neural Radiance Fields (NeRFs) ein, einer neuartigen Technologie, die in den letzten Jahren signifikante Fortschritte im Bereich der 3D-Rekonstruktion erzielt hat. Insbesondere der Aufstieg von Machine Learning und Deep Learning hat dazu geführt, dass NeRFs als vielversprechende Alternative zu herkömmlichen Methoden wie Dense Image Matching (DIM) betrachtet werden. Im Gegensatz zu DIM, das auf klassischen photogrammetrischen Algorithmen basiert, nutzt NeRF neuronale Netze, um aus einer begrenzten Anzahl von 2D-Bildern ein fotorealistisches 3D-Modell zu generieren.

Kapitel 2: Stand der Technik

Der Stand der Technik beschreibt einleitend die Grundzüge der Photogrammetrie. Darüber hinaus wird das Thema Feature-Matching, insbesondere SIFT, behandelt und mit anderen Feature-Extrahierungsverfahren verglichen. Um einen grundlegenden theoretischen Rahmen für NeRFs bereitzustellen, wird das Stammpaper von NeRFs als Grundlage herangezogen. Anschließend werden Weiterentwicklungen von NeRFs vorgestellt und abschließend ein Vergleich zwischen DIM und NeRF herausgearbeitet.

Kapitel 3: Untersuchungsgebiet und Datensätze

Das Kapitel beschreibt die Wetterbedingungen am Messtag, den Arbeitsablauf des Experiments und die Probleme aufgrund der Witterung. Das Kapitel gibt einen Überblick über den Aufbau des Experiments und die Unterschiede zwischen den einzelnen Flugkonfigurationen sollen vermittelt werden.

Kapitel 4: Methodik

Die Methodik beschreibt den Ablauf der Datenverarbeitung und unterteilt diesen in seine einzelnen Schritte. Jeder Schritt der Auswertung wurde detailliert erläutert, um den Lesenden ein klares Verständnis der angewandten Verfahren zu vermitteln. Ziel ist es, die Vorgehensweise transparent darzustellen und die Arbeit als Leitfaden für eigenständige Versuche nutzbar zu machen.

Kapitel 5: Ergebnisse

Das Kapitel präsentiert die Auswertungen in tabellarischer Form und ergänzt diese durch Plots, um die Ergebnisse von NeRFs und DIM gegenüberzustellen. Die Resultate umfassen Punkt-zu-Punkt-Vergleiche, M3C2-Deformationsanalysen, Ebenenextraktionen und sektorale Statistiken. Zusätzlich werden visuelle Vergleiche zwischen NeRF bzw. DIM erstellt und interpretiert. Hierfür wurden Rastervisualisierungen mittels Hillshading erzeugt und gegenübergestellt. Dies soll dazu beitragen, die Unterschiede in der Rekonstruktionsqualität deutlicher zu präsentieren.

Kapitel 6: Beantwortung der Forschungsfragen

In diesem Kapitel werden die Einflussfaktoren auf die Genauigkeit von UAV-Mapping-Modellen, die Georeferenzierung sowie die Rekonstruktionsqualität von Modellen diskutiert. Es wird der Einfluss der Aufnahmemethode (orbital vs. linear) auf die Präzision der Modelle beleuchtet und die Herausforderungen bei der Integration von NeRFs in praktische Workflows thematisiert. Die Erläuterungen beantworten die gestellten Forschungsfragen der Arbeit.

Kapitel 7: Schlussfolgerungen und Ausblick

Das Kapitel beschreibt die Analyse der Ergebnisse hinsichtlich der Anwendung von Neural Radiance Fields (NeRFs) und Dense Image Matching (DIM) im geodätischen Bereich. Es untersucht die Stärken und Schwächen beider Methoden, insbesondere in Bezug auf die Genauigkeit und Qualität der 3D-Rekonstruktionen. Außerdem werden Einflussfaktoren wie die Rechenintensität, die Flugmuster (orbital vs. linear) und die Herausforderungen bei der Georeferenzierung thematisiert. Zusätzlich wird das Potenzial von NeRFs für zukünftige Entwicklungen, besonders im Zusammenspiel mit anderen Technologien wie LiDAR, sowie die Notwendigkeit weiterer Forschungsarbeiten angesprochen.

2 Stand der Technik

2.1 Photogrammetrie und 3D-Rekonstruktion

> "Photogrammetrie ist die Wissenschaft und Kunst aus Abbildungen, speziell Photographien, die Lage und die Form von Objekten zu rekonstruieren. Die Bildern werden in der Regel photoelektrisch aufgezeichnet (digitale Photographie)"[1].

Die Photogrammetrie ist eine Technik, die verwendet wird, um die Form, die Dimensionen und die Position eines Objekts im Raum präzise zu bestimmen, wobei hauptsächlich Messungen über eine oder mehrere Fotografien des Objekts vorgenommen werden. Heute ist die Photogrammetrie aufgrund ihrer einfachen Anwendung, geringen Kosten und guten Ergebnisse eine beliebte Aufnahme- und Rekonstruktionsmethode. Aus diesen Gründen wird sie zu einer guten Alternative zum Laserscanning. Dies hat zu ihrer Anwendung in verschiedenen Bereichen wie der Archäologie, Architektur und Topografie geführt, insbesondere für Rekonstruktionen von Gebäuden, Landschaftsformationen und archäologischen Artefakten.
Innerhalb der photogrammetrischen Forschung ist man bemüht, mathematische als auch physikalische Modelle zu entwickeln, die die reale Welt modellieren bzw. abbilden. Weiters wird erstrebt, Auswerteprozesse zu entwickeln, die innerhalb der Praxis umsetzbar sind. Besonderes Augenmerk wird hierbei auf automatisierte Auswertepraktiken gelegt, um menschliche Fehlereinflüsse zu minimieren. Die Photogrammetrie erlaubt die Abtastung und Erfassung von Objekten ohne direkte Berührung der Gegenstände. Diese Art der Datenerfassung ist in die Kategorie der passiven Aufnahmesysteme einzugliedern und auch innerhalb der Fernerkundung von hoher Bedeutung.
Die Fernerkundung beschäftigt sich mit der Gewinnung von Informationen der Erdoberfläche durch Messung der reflektierten oder emittierten elektromagnetischen Strahlung. Um Photogrammetrie von Fernerkundung begrifflich abzugrenzen, stehen in der Photogrammetrie die geometrischen Merkmale von Objekten im Vordergrund, während in der Fernerkundung das Interesse eher den radiometrischen Merkmalen der zurückgestreuten elektromagnetischen Strahlung gilt[1]. Abbildung 1 zeigt die Möglichkeiten zur Erfassung von Geometrien.

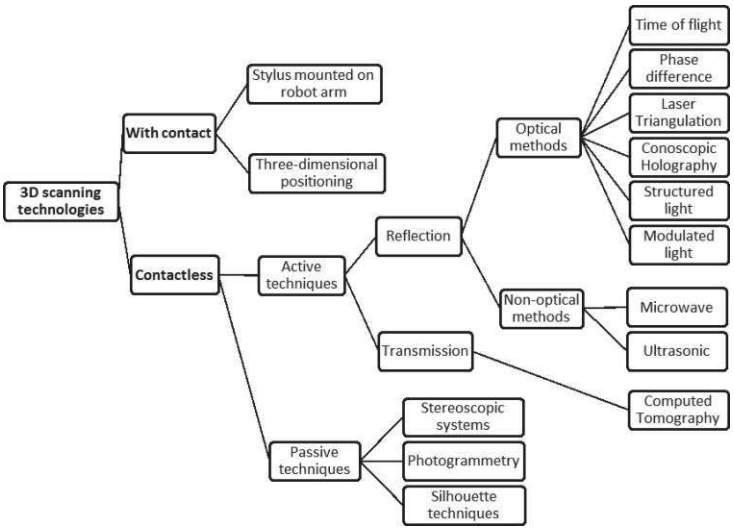


Abbildung 1: Klassifizierung Scanning Technologien [2]

Die digitale Photogrammetrie wurde in den 1980er Jahren geboren. Der wesentliche Aufschwung war der Entwicklung von Digitalkameras geschuldet. Grundlegend lässt sich Photogrammetrie in mehrere Phasen unterteilen. Die Planungsphase[1], Kamerakalibrierung, Bildprozessierung und abschließend der Export der Resultate in ein CAD oder 3D-Modell. Die Qualität des resultierenden Datensatzes hängt von der Kameraauflösung, der Textur des zu erfassenden Objektes sowie der Qualität des Abbildungssystems (Optik), einschließlich der Bildschärfe, ab.

2.1.1 Photogrammetrischer Normalfall

Der photogrammetrische Normalfall stellt eine grundlegende Konfiguration der Photogrammetrie dar und ist in Abbildung 2 schematisch dargestellt. Die Rekonstruktion eines Objektpunktes ist dahingehend einfacher, da der Abstand der beiden Bilder bekannt ist und die Ausrichtung der Bilder parallel zueinander ist. Der Abstand der beiden Bilder ist die Basis B, wobei O_1 und O_2 die Projektionszentren der Kameras darstellen. Im Falle der Luftbildvermessung können die genannten Bedingungen nur schwer eingehalten werden. Terrestrisch hingegen kann bei entsprechender Konstruktion eine fixe Basis realisiert werden. In diesem Fall spricht man von einer Stereomesskamera[1]. Die Abbildungsgleichungen, d.h. die Beziehung zwischen Kamerakoordinaten und Objektkoordinaten, sind folgendermaßen zu berechnen.

$$-Z = \frac{cB}{x_1 - x_2} = \frac{cB}{p_x}$$

$$Y = -Z\frac{y_1}{c} = -Z\frac{y_2}{c}$$

$$X = -Z\frac{x_1}{c}$$

Die Größe c entspricht der Brennweite der Kamera, und px die Parallaxe [3]. Die Objektoordinaten

[3]Der Unterschied der Koordinaten eines Punktes in verschiedenen Bildern entlang einer Achse

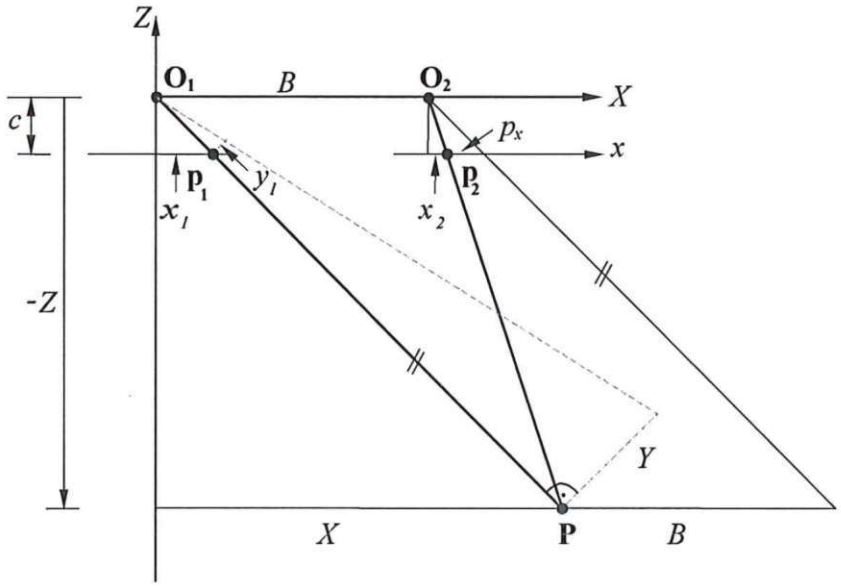


Abbildung 2: Photogrammetrischer Normalfall

X, Y, Z sind als Großbuchstaben ausgewiesen. Die Herleitung der Formelapparate findet sich in [1].

2.1.2 Structure from Motion

Das Structure from Motion Theorem (SfM) basiert auf der Überlegung, dass 3D Objektoordinaten aus mehreren gemessenen 2D-Bildkoordinaten erlangt werden, selbst bei Unkenntnis von 3D-Informationen über das Objekt. Voraussetzung hierfür ist die Annahme, dass das zu betrachtende Objekt starr ist und keine Bewegung erfährt. Im Wesentlichen beschreibt es, wie aus der Bewegung der Kamera relativ zu einem Objekt und den resultierenden Bildprojektionen die 3D-Geometrie des Objekts und dessen Bewegung berechnet werden kann [3].
Im Gegensatz zum photogrammetrischen Normalfall, dessen Unbekannte aufgrund der Kamerakonfiguration weitgehend gelöst sind, basiert SfM auf der relativen Bewegung zwischen der Kamera und dem Objekt. SfM-Algorithmen finden korrespondierende Punktpaare zwischen einem überlappenden Satz von Bilder und ermitteln dadurch die Position und Verdrehung der Kameras zueinander. Die Anzahl der Punktpaare hängt wiederum von der Textur und Belichtung des Objektes ab. Bei schwacher Textur werden aufgrund geringer Grauwertunterschiede nur wenig korrespondiere Punktpaare zwischen überlappenden Bildern erfasst. Diese sind jedoch für die Genauigkeit des auszurichtenden Bildblockes von Bedeutung. Neben der Qualität korrespondierender Punktpaare, ist die Verteilung von Merkmalen (sog. Features) innerhalb eines Bildes, ausschlaggebend. Bei ausreichender Verteilung von Features wird der Bildblock robuster ausgerichtet und ist dadurch besser gestützt.

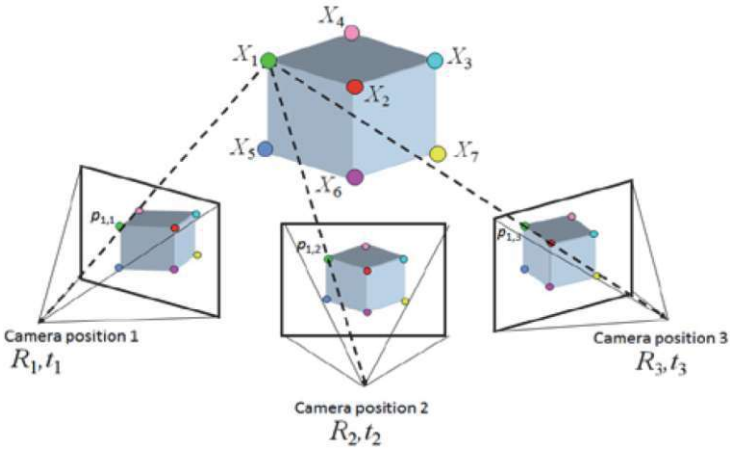


Abbildung 3: Structure from Motion - Merkmalspunkte: Eckpunkte des Würfels [2]

Feature Extrahierung Typischerweise lässt sich ein SfM-Workflow in mehrere Schritte unterteilen. Zu Beginn der Auswertungen wird eine Merkmalsextraktion (Feature extraction) anhand der Bilder vorgenommen. Da jedes Bild individuelle Features besitzt, gibt es zu diesem Zeitpunkt noch keinen Konnex zu den anderen Bildern innerhalb des Blocks. Die Feature Extrahierung erfolgt mit erprobten Algorithmen, die weitreichend in der Literatur gefunden werden können [4][5][6][7][8][9].

SIFT (Scale-Invariant Feature Transform), SURF (Speeded-Up Robust Features), KAZE werden vorwiegend für die 3D-Rekonstruktion eingesetzt, wobei AKAZE (Accelerated KAZE), ORB (Oriented FAST and Rotated BRIEF) und BRISK (Binary Robust Invariant Scalable Keypoints) für mobile Anwendungen von Vorteil sind. Der Grund liegt in der Geschwindigkeit der Merkmalsextraktion. Besonders für Echtzeitanwendungen ist die Geschwindigkeit der Merkmalsextraktion wichtig. SLAM-Anwendungen (Simultaneous Localization and Mapping) sind hoch dynamisch und die Prozessierung von Merkmalen muss in Echtzeit geschehen. Der Zuwachs der Geschwindigkeit geht jedoch mit einem Verlust der Genauigkeit einher.

SIFT ist der bekannteste Feature-Extraktor [10] und wurde von Lowe[11] entwickelt. Der SIFT-Detektor basiert auf dem Prinzip von *Difference of Gaussian* (DoG), welcher als Approximation von *Laplacian of Gaussian* (LoG) angesehen wird. Feature-Punkte werden durch Suche eines lokalen Maximums mithilfe des DoG mit variabler Skalierung gesucht. Die Beschreibungsmethode erfolgt durch eine 16x16 Nachbarschaft ausgehend von dem gefunden Feature-Punkt. Diese Nachbarschaft wird weiter unterteilt in Sub-Blöcke, wodurch insgesamt 128 Bin-Werte erzeugt werden. SIFT ist robust gegenüber Bildrotationen, Skalierungen und begrenzten Affin-Transformationen. Der größte Nachteil des SIFT-Algorithmus ist der Rechenleistungsbedarf. Die Faltung des DoG kann wie folgt dargestellt werden:

$$\mathrm{DoG}(x, y, \sigma) = (G(x, y, k\sigma) - G(x, y, \sigma)) * I(x, y)$$

Hierbei ist $G(x, y, \sigma)$ die Gaußsche Funktion, die das Bild $I(x, y)$ mit der Standardabweichung σ glättet. Der Parameter k ist ein konstanter Skalierungsfaktor zwischen den beiden Gaußschen

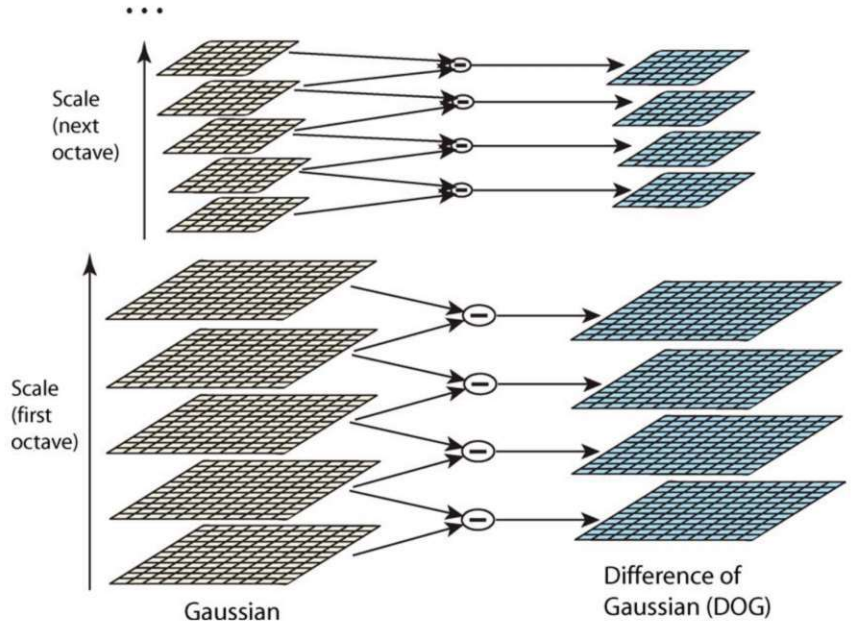


Abbildung 4: Oktaven - SIFT

Filtern. Das Symbol $*$ stellt die Faltung dar, und $I(x, y)$ bezeichnet das zu glättende Bild.

Feature Matching Aufgrund der hohen Anzahl an Features, die innerhalb eines Feature-Extraktionsalgorithmus erkannt werden, muss eine Auswahl von qualitativen Merkmalspunkten stattfinden. Hierfür werden Feature-Matching Techniken eingesetzt. Beispielhaft sei an dieser Stelle ein Distanzverhältnis-Test (NNDR), der in [4] erwähnt wird, vorgestellt. Für jeden Merkmalspunkt im zu analysierenden Bild wird der nächstgelegene Nachbar in einer Datenbank von Merkmalspunkten gesucht, basierend auf dem Vergleich der SIFT-Deskriptoren. Um jedoch die Zuverlässigkeit der Zuordnung zu erhöhen und falsche Matches zu vermeiden, wird nicht nur die Distanz zum nächstgelegenen Nachbarn betrachtet, sondern auch die Distanz zum zweitnächsten Nachbarn. Wenn das Verhältnis $d1/d2$ einen Schwellenwert von 0.8 überschreitet, wird die Korrespondenz verworfen. Ein kleines Verhältnis bedeutet, dass der nächstgelegene Nachbar deutlich näher ist als der zweitnächste. Dies deutet darauf hin, dass die Merkmalspunktzuordnung eindeutig und verlässlich ist. Durch die Verwendung des Schwellenwerts von 0,8 werden also nur diejenigen Merkmalspunkte akzeptiert, bei denen der nächstgelegene Nachbar signifikant besser übereinstimmt als alle anderen.
In [10] wird eine Gegenüberstellung unterschiedlicher Feature-Extraktoren untersucht. Dabei wurde die Leistungsfähigkeit hinsichtlich mehrerer Kriterien bewertet. Die Kriterien wurden unterteilt in Anzahl von Merkmalen, Recheneffizienz, Feature-Matching Effizienz und die Geschwindigkeit des vollständigen Abgleichs (Merkmalsdetektion bis Bildabgleich). ORB zeigt sich in allen Rankings an erster Stelle. Der Grund liegt hier in der Balance zwischen allen geforderten Kriterien. Nichtsdestotrotz ist SIFT der akkurateste Feature-Extraktor und nur geringfügig

langsamer als ORB. Topographische Einsätze benötigen oftmals keiner Real-Time Zuordnung von Merkmalen. In diesem Fall steht die Genauigkeit der Merkmalszuordnung im Vordergrund.

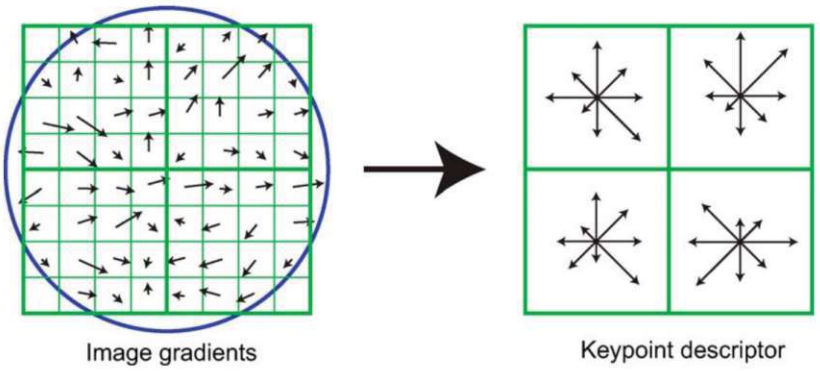


Abbildung 5: Keypoint-Deskriptor[4]

Algorithmus	**Vorteile**	**Nachteile**	**Einsatzgebiete**
SIFT	Hohe Robustheit gegenüber Skalierung, Rotation und Beleuchtung	Hohe Rechenkosten	Bildregistrierung, Objekterkennung, 3D-Rekonstruktion
SURF	Schneller als SIFT, robust gegenüber Skalierung und Beleuchtung	Weniger robust gegenüber Affin-Transformationen, etwas langsamer als ORB	Bildregistrierung, Objekterkennung, 3D-Rekonstruktion
KAZE	Robust gegenüber Skalierung, Rotation und Affin-Transformationen	Hohe Berechnungskosten, besonders bei größeren Datensätzen	Bildverarbeitung, Objekterkennung, 3D-Rekonstruktion
AKAZE	Effizienter als KAZE, robuste Invarianz gegenüber Skalierung und Rotation	Weniger robust als KAZE bei Rotationsänderungen	Echtzeitanwendungen, mobile Robotik, SLAM
ORB	Sehr effizient, gut geeignet für Echtzeitanwendungen	Weniger robust gegenüber Skalierungsvariationen	Echtzeitanwendungen, Augmented Reality, SLAM
BRISK	Gute Balance zwischen Geschwindigkeit und Genauigkeit, robust gegenüber Skalierung und Rotation	Weniger robust bei starkem Rauschen	Echtzeitanwendungen, mobile Anwendungen, SLAM

Tabelle 1: Vergleich von Feature-Extraction-Algorithmen und ihre Einsatzgebiete

RANSAC - Random Sample Consensus Nachdem eine hohe Anzahl an korrespondierenden Feature Punkte mithilfe eines Feature-Extrahierungs Algorithmus gefunden wurde, müssen die Zuordnungen auf deren Robustheit überprüft werden. Aufgrund der Automatisierung der Merkmalsextraktion gibt es Ausreißer, die die Berechnung der Zuordnung eines Bildpaares beeinträchtigen. Aus diesem Grund werden robuste Verfahren wie z.B. RANSAC eingesetzt. Die relative Orientierung zwischen eines Bildpaares hat fünf Parameter. Davon sind 12 Unbekannte der äußeren Orientierung zuzuordnen abzüglich sieben Unbekannte für das geodätische Datum. Die äußere Orientierung jeder Kamera hat drei Rotationsparameter und drei Translationsparameter. Für zwei Kameras ergibt das 12 Unbekannte. Somit sind projektiv betrachtet (Essentielle Matrix) fünf korrespondierende Punkte notwendig um die Homographie zwischen zwei Bilder zu lösen. Da am Beispiel SIFT 40 000[4] Merkmalspunkte extrahiert werden, kann mit einer hoch überbestimmten Zuordnung gearbeitet werden. RANSAC wird in Kombination mit der Essentiellen Matrix eingesetzt, um die Epipolargeometrie zwischen den Bildern robust zu schäzen. Insbesondere in der Gegenwart von Ausreißern innerhalb der Merkmalszuordnungen, erweist sich RANSAC als hoch praktikabel. RANSAC wählt fünf zufällige Punktkorrespondenzen, löst somit die Epipolargeometrie eindeutig, und durchläuft mit der gewählten Iterationszahl unterschiedliche Korrespondenzen. [10] setzt die Iterationszahl auf 2000 mit einer Konfidenz von 99.5%. Die Homographie-Matrix, welche mithilfe RANSAC gelöst wird, ist eine 3x3 Matrix welche die Transformation vom ersten Bild ins zweite Bild darstellt. Neben Homographiematrix und der Essentiellen Matrix gibt es innerhalb der Epipolargeometrie auch die Fundamentalmatrix. Die Homographiematrix ist für planare Szenen geeignet, wobei Tiefeninformationen keine Rolle spielen. Die Essentielle Matrix ist für kalibrierte Kameras einsetzbar, da die innere Orientierung bekannt ist. Mithilfe dessen können auch 3D-Rekonstruktionen umgesetzt werden. Die Fundamentalmatrix wird benötigt, wenn keine planare Szene gegeben und keine kalibrierte Kamera im Einsatz ist. An dieser Stelle ist zu erwähnen, dass nach initialer Ausrichtung der Kameras im Raum, immer ein Bündelblockausgleich folgt. Dieser optimiert die Positionen der Kameras hinsichtlich deren Lage im Raum sowie die Rotationswinkel ω, ϕ und κ. Zusätzlich wird die innere Orientierung der Kamera optimiert. Da das Thema des Bündelblocksausgleichs sehr umfangreich ist, wird an dieser Stelle auf weiterführende Literatur verwiesen [12].

2.2 Dense Image Matching (DIM)

Um aus einer dünnbesetzten Punktwolke (engl. sparse point cloud), welche das Ergebnis von z.B. SIFT darstellt, eine dichte Rekonstruktion der Szene zu generieren, wird dichte Bildzuordnung (engl. Dense Image Matching, DIM) eingesetzt. Laut Remondino et. al[13] werden an eine dichte Rekonstruktion der Szene mehrere Forderungen gestellt. Erstens muss die Punktauflösung adaptiv angepasst werden, um sowohl die Erhaltung von Details zu gewährleisten, als auch die Effizienz der Auswertung zu garantieren. Das bedeutet, dass der Algorithmus die Punktdichte dynamisch anpasst. In Bereichen mit komplexen Strukturen ist eine höhere Punktdichte erforderlich, um feine Details zu erfassen und Kanten zu erhalten. Hingegen sollte in flachen oder

[4] Ein von Lowe [4] vorgeschlagener Grenzwert

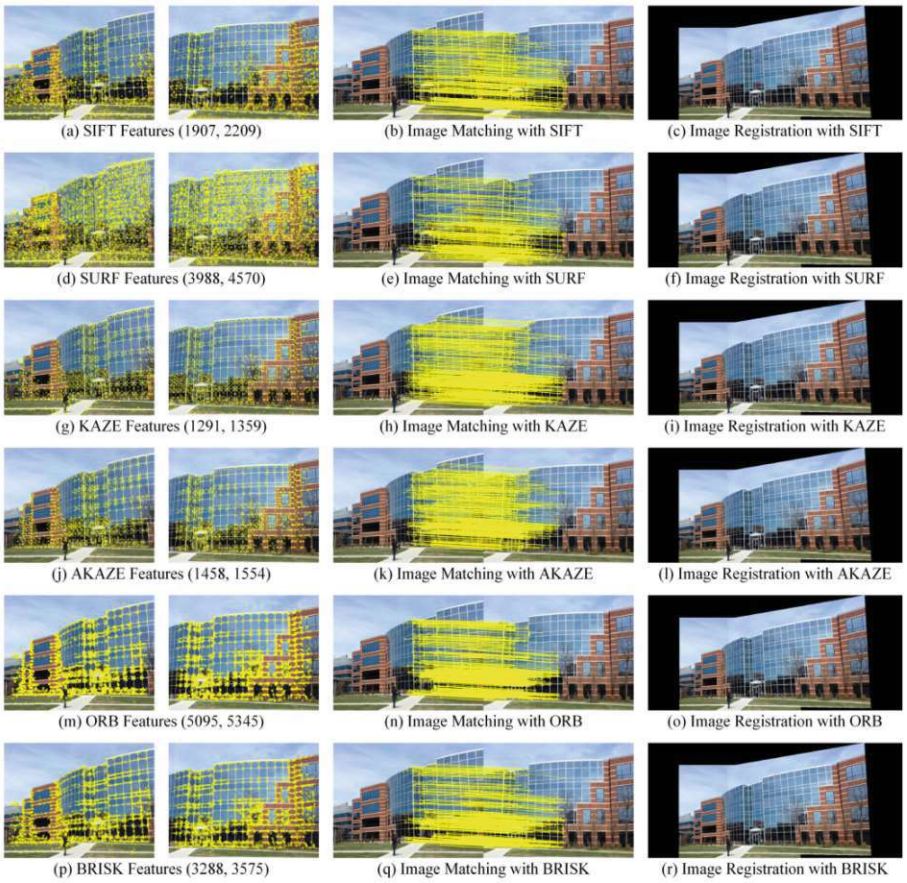


Abbildung 6: Feature Detektion, Matching und Mosaikierung [10]

homomgenen Bereichen die Punktdichte reduziert werden. Weiters soll die Rekonstruktion auch in Regionen mit schwierigen Bedingungen zuverlässig funktionieren. Damit ist gemeint, dass texturarme als auch unterbelichtete Bereiche rekonstruiert werden. Zudem soll der Algorithmus robust gegenüber Maßstabsänderungen sein, welche durch unterschiedliche Zoomstufen von Aufnahmen produziert werden. Insgesamt unterstreichen Remondino et al. die Notwendigkeit eines intelligenten und adaptiven Ansatzes für die dichte 3D-Punktwolkenextraktion.
Ähnlich wie bei der Problematik der Feature-Extrahierung, gibt es für die dichte Rekonstruktion einer Geometrie unterschiedliche Ansätze bzw. Algorithmen. Am Beispiel SURE, welches einen MVS (Multi View Stereo) Ansatz verfolgt, wird ein Referenzbild auf ein Set von adjazenten Bilder mithilfe eines SGM (Semi Global Matching) Algorithmus gematched[14]. Für jedes Paar wird eine Disparitätenkarte berechnet. Eine Disparitätenkarte zeigt, mit welcher Intensität sich Objekte zwischen zwei Bildern durch die Bewegung der Kamera verändern. Nähere Objekte werden aufgrund der Nähe zum Projektionszentrum einer stärkeren Bewegung unterliegen als Objekte die weiter entfernt von dem Aufnahmeort liegen. Alle Disparitätskarten, die das gleiche Referenzbild teilen, werden zu einer Punktwolke zusammengeführt, wobei die Redundanz der Stereo-Paare genutzt wird. Disparitätskarten stellen dabei die Verschiebung der Bildpunkte zwischen den Stereo-Bildern dar und bilden eine wichtige Grundlage für die Berechnung von

Tiefenkarten. Diese Tiefenkarten, welche die tatsächliche Entfernung eines jeden Bildpunkts zur Kamera angeben, werden aus den Disparitätswerten und den Kameraparametern berechnet.
Im Rahmen eines Preprocessing-Moduls wird eine Netzwerkanalyse durchgeführt, um geeignete Bildpaare für den Rekonstruktionsprozess auszuwählen. Anschließend werden Epipolarbilder erzeugt, und der SGM-Algorithmus (Semi-Global Matching) kommt zum Einsatz, um die Tiefenkarten aus den Disparitätskarten zu erstellen. Diese Tiefenkarten werden dann in 3D-Koordinaten umgewandelt. Dieser Prozess reduziert die Anzahl von Ausreißern und erhöht die Präzision.
Im Vergleich zum klassischen SGM-Ansatz durchsucht SURE Pixelkorrespondenzen mit dynamischen Disparitätssuchbereichen, was die Flexibilität und Genauigkeit steigert. Darüber hinaus implementiert SURE eine Ausreißerentfernung, wodurch die Rechenzeit erheblich reduziert wird.

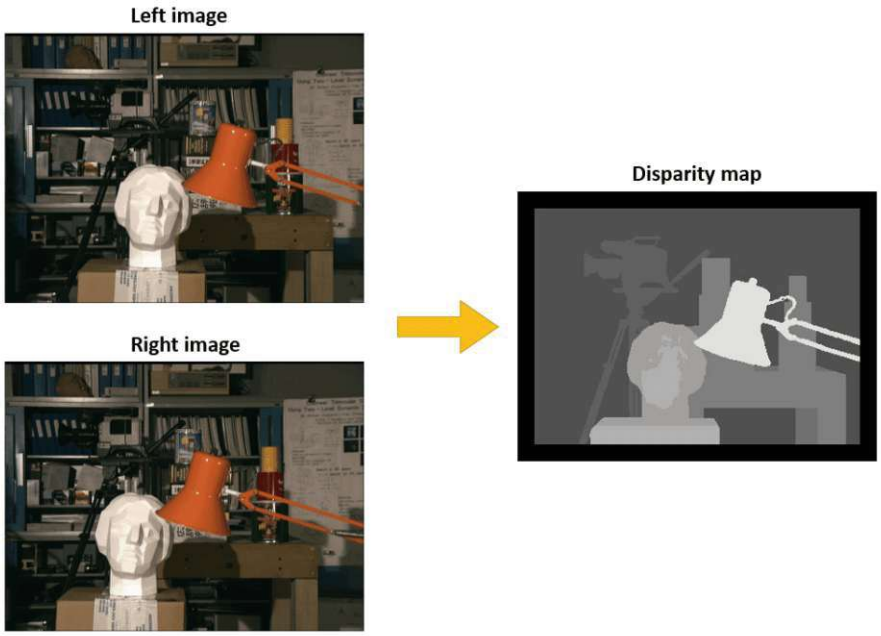


Abbildung 7: Disparity Map [15]

Um eine optimale Disparitätskarte D zu berechnen, die die beste Übereinstimmung zwischen korrespondierenden Bildpunkten findet, wird eine Energiefunktion $E(D)$ definiert. Diese Funktion bewertet die Kosten einer bestimmten Disparitätszuordnung, indem sie die Kosten für die Korrespondenz einzelner Bildpunkte und die Konsistenz der Disparität innerhalb einer Nachbarschaft berücksichtigt. Die Energiefunktion lautet:

$$E(D) = \sum_{p} C(p, D_p) + \sum_{q \in N_p} \left(P_1 \cdot T\left(|D_p - D_q| = 1\right) + P_2 \cdot T\left(|D_p - D_q| > 1\right) \right)$$

Dabei gilt:

p: Ein Pixel im Bild (i, j)

D_p: Die Disparität für das Pixel p

$C(p, D_p)$: Kostenfunktion für eine Korrespondenz $c(i, j, d)$

N_p: Die Nachbarschaft von p

$T\,()$: Indikatorfunktion

P_1: Strafterm für kleine Disparitätsunterschiede

P_2: Strafterm für große Disparitätsunterschiede

Der erste Term $C(p, D_p)$ beschreibt die Kostenfunktion, welche die Übereinstimmung zwischen dem Pixel p im linken und dem korrespondieren Pixel im rechten Bild auf Grundlage der Disparität D_p bewertet. Die Kosten hängen vom Unterschied in den Grauwerten ab. Der zweite Term $\sum_{q \in N_p} P_1 \cdot T\,(|D_p - D_q| = 1)$ führt eine Strafgebühr $P1$ für benachbarte Pixel q innerhalb der Nachbarschaft N_p von p ein, wenn der Disparitätsunterschied genau 1 beträgt. Bei z.B. geneigten oder gekrümmten Oberflächen wird somit eine Anpassung der Disparität begünstigt. Der dritte Term ist ident mit dem zweiten Term. Jedoch werden Disparitätsunterschiede größer 1 bestraft. $P2$ ist der Strafwert für größere Sprünge in der Disparität, die meistens an Kanten oder abrupten Tiefenänderungen auftreten. Dadurch werden plötzlich Sprünge in flachen Bereichen vermieden, während Kanten erhalten bleiben. Daraus wäre abzuleiten, dass Kanten aufgrund der Strafterme glatter erscheinen als bei TLS. Die Disparitätenkarte muss nun in eine Tiefenkarte mit Tiefeninformation übergeleitet werden. Hierfür muss die Disparitätenkarte in den Normalfall übergeleitet und der Bildinhalt entlang der Epipolarlinien zugeordnet werden. Abbildung 8 veranschaulicht die Prozedur. Die Zuordnung des Bildinhalts entlang der Epipolarlinien geschieht mit einer 1D-Suche. Meistens wird vor Überleitung der Disparitätenkarte in den Normalfall eine Interpolation durchgeführt. Das Ergebnis nach Überleitung ist die Tiefenkarte welche anschließend in ein übergeordnetes Koordinatensystem transformiert wird. Die Tiefenkarte wird anschließend für die Generierung einer dichten Punktwolke eingesetzt. Weitere Informationen zu SGM finden sich in Hirschmüller et. al [16].
Neben SGM sei an dieser Stelle ein Area-based Matching Ansatz erwähnt. Area-based Matching basiert auf der Annahme, dass ein Template ausgewählt wird (linkes Bild), welches dann als Suchfenster für das rechte Bild fungiert. Der passende Partner im rechten Bild wird mittels Korrelation gefunden. Das Fenster wird anschließend am Punkt des linken Bildes zentriert, als Referenz genutzt und über das rechte Bild geschoben um die best passende Position zu finden. Neben der Verschiebung wird auch eine affine Verzerrung bestimmt, wobei ggf. eine Grauwert-Anpassung zusätzlich für die Angleichung des Bildpaares durchgeführt wird. Das sogenannte Least Squares Matching ist ein Vertreter des Area-based Matching. LSM ist jedoch weniger robust, langsam und benötigt gute Näherungswerte. Die Genauigkeit liegt im Subpixelbereich (1/10 Pixel).

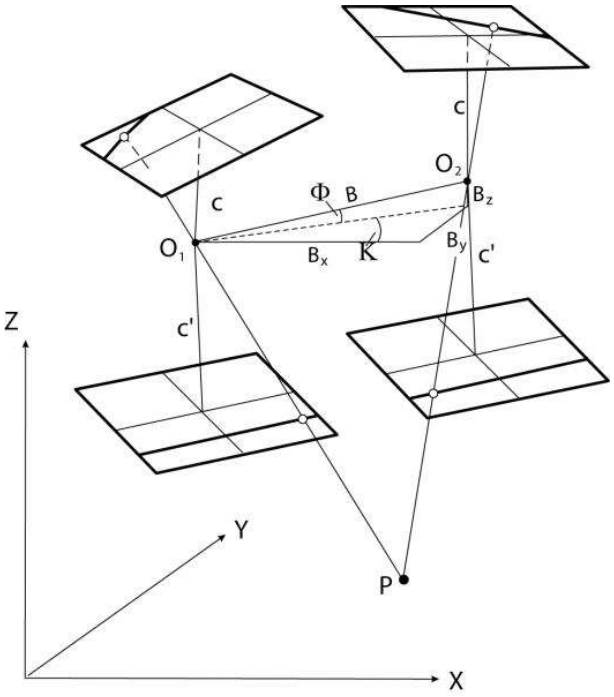


Abbildung 8: Umrechnung auf Normalfall

2.3 Neural Radiance Fields (NeRF)

Bis dato wurden in wissenschaftlichen Arbeiten einige Ansätze zur Szenenrepräsentation vorgestellt. Bei dichter Bildabdeckung wurden durch Zwischenberechnung (Interpolation) dieser Bilder Szenen generiert, die physisch gar nicht existierten[17][18][19].
Für die Rekonstruktion mit wenigen Aufnahmen sind spezielle und anspruchsvollere Methoden erforderlich. Bei geringer Bildanzahl kommen Verfahren zum Einsatz, die das Erscheinungsbild und die Struktur einer Szene vorhersagen und dabei Mesh-Modelle nutzen, um die Formen und Oberflächen darzustellen. Solche Ansätze basieren häufig auf einer Modellierung mit diffusem oder blickrichtungsabhängigem Erscheinungsbild[20][21][22] und können mit Hilfe differenzierbarer Rasterizer[23][24][25][26] oder Pathtracer[27][28]optimiert werden, die die Bildinhalte mithilfe von Gradienten-basierten Methoden anpassen. Ein wiederkehrendes Problem bei dieser Art der Optimierung ist das Feststecken in lokalen Minima, wodurch das Modell nicht immer die bestmögliche Anpassung erreicht[27]. Zudem wird bei solchen Ansätzen ein initiales Mesh-Modell mit festgelegter Topologie benötigt, das in realen, offenen Szenen häufig nicht verfügbar ist[27].
Ein alternativer Ansatz verwendet volumetrische Repräsentationen, bei denen der Raum in kleine Volumeneinheiten (Voxel) unterteilt wird, um Farbinformationen und Dichte zu speichern[29][30][31]. Volumetrische Methoden bieten eine realistische Darstellung komplexer Formen und Materialien und erzeugen weniger störende Artefakte als Mesh-basierte Methoden. Allerdings haben voxelbasierte Verfahren oft einen hohen Speicherbedarf, insbesondere bei hoher Auflösung[32][33].
In einer aktuellen Methode zur Synthese neuer Ansichten komplexer Szenen wird ein kontinu-

ierliches volumetrisches Szenenmodell verwendet, das durch ein neuronales Netzwerk optimiert wird (NeRFs). Hierbei werden 5D-Koordinaten, bestehend aus räumlicher Position und Blickrichtung, als Eingabe verwendet, um Volumendichte und strahlungsabhängige Emission zu berechnen. Durch Volumen-Rendering und die Optimierung der Netzwerkparameter anhand eines spärlichen Satzes von Bildern mit bekannten Kameraposen können fotorealistische Ansichten generiert werden[34].

Abbildung 9: Grundprinzip NeRFs[34]

2.3.1 Einleitung

Im Allgemeinen repräsentieren NeRFs eine statische Szene als eine kontinuierliche 5D Funktion, welche die Strahlen (radiance) in jede Richtung (Φ,ϕ) an jedem Punkt (x, y, z) im Raum mit einer Dichte repräsentiert. Die Dichte wirkt dabei als Opazitätsregler wie viel Strahlen akkumuliert durch den Punkt (x, y, z) laufen. Mildenhall et. al [34] optimiert ein "deep fully-connected neural network" ohne Faltungsschichten (convolutional layers). Das bedeutet, dass das Netzwerk tief[5], vollständig verbunden[6] und keine Faltungsschichten besitzt und somit die Daten global berücksichtigt werden. Dabei werden speziell lokale Beziehungen weniger berücksichtigt, was normalerweise für Bilddaten von Vorteil wäre. Da NeRFs als Ziel die Repräsentation einer Szene haben, wird das Augenmerk auf das Erlernen einer Funktion gelegt, die für jeden Punkt im Raum und jede Blickrichtung die entsprechende Dichte und Farbe vorhersagt.

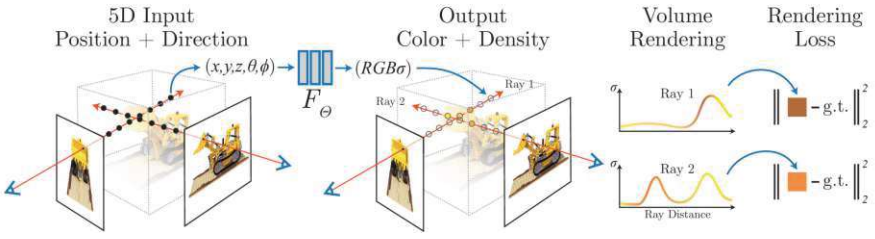


Abbildung 10: Funktionsweise von NeRFs[34]

Um eine Szene von einem bestimmten Blickpunkt zu rendern, werden mehrere Ansätze verfolgt. Zuerst werden von einer Kamera Strahlen durch die Szene geschickt. Diese Strahlen schneiden verschiedene Punkte im 3D-Raum. Durch dieses "Durchschreiten"der Szene werden eine Reihe von Probenpunkten entlang jedes Strahls generiert. Die so gewonnenen 3D-Punkte und die

[5]Das Netzwerk besteht aus vielen Schichten
[6]jedes Neuron einer Schicht ist mit jedem Neuron der nächsten Schicht verbunden

zugehörigen 2D-Blickrichtungen werden als Eingaben in das neuronale Netzwerk eingespeist. Das Netzwerk berechnet für jeden Punkt die entsprechenden Farben und Dichten, die davon abhängen, wo sich der Punkt befindet und aus welcher Richtung er betrachtet wird. Diese Vorhersagen liefern also Informationen darüber, wie der Punkt aussieht und wie dicht das Objekt an dieser Position ist. Die berechneten Farben und Dichten der 3D-Punkte werden anschließend mit klassischen Volumenrendering-Techniken kombiniert, um ein 2D-Bild zu erzeugen. Dabei werden die Informationen entlang der Strahlen gesammelt, um zu bestimmen, wie viel Licht von jedem Punkt entlang des Strahls zur Kamera gelangt, und diese in einem finalen Bild zusammengefasst.

2.3.2 Aufbau des Netzwerks

Die Eingabe des Netzwerks besteht aus einer 3D-Position (x, y, z) und einer Blickrichtung ϕ, θ, welche in einem 3D-kartesischen Einheitsvektor dargestellt wird. Die Eingabe ergibt eine 5D-Koordinate, die vom Netzwerk verarbeit wird und daraus eine Farbe und Dichte ableitet. Das Hauptnetzwerk wird durch ein tiefes neuronales Netzwerk mit acht vollständig verbunden Schichten realisiert. Jede dieser Schichten hat 256 Kanäle wobei jede Schicht die ReLU-Aktivierungsfunktion verwendet. Negative Eingaben werden hierbei auf 0 gesetzt und positive Werte bleiben unverändert. Hiermit wird die Nichtlinearität und Lernfähigkeit des Netzwerks gefördert. Nach Verarbeitung der Eingaben werden vom Netzwerk die Dichte σ (die nur von der Position abhängt) und ein 256-dimensionaler Feature-Vektor zurückgegeben. Die Dichte beschreibt, wie dicht das Material an einem bestimmten Punkt ist bzw. ob ein Objekt im Raum überhaupt vertreten ist. Der 256-dimensionale-Feature Vektor gibt eine Merkmalsbeschreibung der Szene an dieser Position zurück. Um die Farbe $c = (R, G, B)$ zu bestimmen, wird der 256-dimensionale Vektor mit der Blickrichtung d kombiniert. Die Blickrichtung fungiert hier als zusätzliche Information, um das blickrichtungsabhängige Erscheinungsbild der Szene zu berechnen. Besonders wichtig ist dies im Falle von spiegelenden Oberflächen, die sich mit Änderung der Blickrichtung ebenfalls ändern. Diese kombinierte Information wird anschließend an eine weitere vollständig verbunden Schicht mit 128 Kanälen weitergegeben, wobei ebenfalls eine ReLU-Aktivierungsfunktion verwendet wird. Das Ergebnis ist die RGB-Farbe an der Position x, y, z. Allgemein lässt sich sagen, dass der Einsatz einer ReLU-Aktivierung dabei hilft, komplexe, nichtlineare Muster in den Daten zu erlenen. Der 256-dimensionale Vektor trägt dazu bei, dass mehr Lernpotential für komplexere Szeneninformationen gegeben ist. Die Separation von Dichte und Farbe ist notwendig, um realistische Szenen zu generieren. Dabei wird die Dichte nur aus der Position bestimmt, während die Farbe zusätzlich von der Blickrichtung abhängt. Weiterführende Erläuterungen zum Aufbau des Netzwerks finden sich in Mildenhall et. al S.18 [34].

2.3.3 Volumen Rendering mit NeRFs

Im Falle von NeRFs werden grundlegende klassische Volumen Rendering Methoden verwendet[35]. Volumen Rendering hat die Aufgabe, dreidimensionale Objekte darzustellen, indem

man Farben und Dichten entlang eines Lichtstrahls kombiniert. Der Lichtstrahl durchläuft die Szene, wobei an bestimmten Punkten der Strahl auf ein Objekt trifft. Dieses Volumenpartikel wird als Voxel bezeichnet und beschreibt, wie viel Licht emittiert oder blockiert wird.

$$C(r) = \int_{t_n}^{t_f} T(t)\sigma(r(t))c(r(t), d)\, dt$$

$$T(t) = \exp\left(-\int_{t_n}^{t} \sigma(r(s))\, ds\right)$$

$$r(t) = o + td$$

$C(r)$ ist die berechnete Farbe entlang des Strahls r, wobei t_n und t_f die nahen und fernen Begrenzungen des Strahls darstellen. Die akkumulierte Transmittanz $T(t)$ entlang des Strahls beschreibt die Wahrscheinlichkeit, dass der Strahl von t_n nach t reist, ohne gestreut oder absorbiert zu werden. Die Volumendichte $\sigma(r(t))$ am Punkt $r(t)$ beschreibt, wie dicht das Medium ist, und damit die Wahrscheinlichkeit eines Treffers an dieser Stelle. Schließlich beschreibt $c(r(t), d)$ die emittierte Farbe am Punkt $r(t)$ in Richtung d.
Das Integral ist analytisch nicht lösbar. Daher werden Quadraturmethoden angelehnt an Max [36] eingesetzt, um das Integral numerisch zu approximieren. Hierfür wird der Bereich entlang des Strahls in mehrere Abschnitte unterteilt und ausgewertet. Um eine genauere Schätzung zu erhalten, wird stratified sampling verwendet. Hierbei werden Samples zufällig aus den Bins gezogen, was zu einer genaueren und gleichmäßigeren Abtastung führt. Diese Proben werden dann summiert, um die Farbe des Strahls zu bestimmen.
Die Zufallsvariable t_i, die innerhalb eines Intervalls gezogen wird, lautet:

$$t_i \sim U\left(t_n + \frac{i-1}{N}(t_f - t_n), t_n + \frac{i}{N}(t_f - t_n)\right)$$

wobei t_n der Startpunkt (Nahpunkt) des Strahls, t_f der Endpunkt (Fernpunkt) des Strahls und N die Anzahl der gleichmäßigen Intervalle ist, in die der Raum unterteilt wird. $\hat{C}(r)$ stellt die Approximation des Integrals $C(r)$ dar.

$$\hat{C}(r) = \sum_{i=1}^{N} T_i \left(1 - \exp(-\sigma_i \delta_i)\right) c_i$$

Hierbei ist T_i die akkumulierte Transmission bis zum i-ten Punkt:

$$T_i = \exp\left(-\sum_{j=1}^{i-1} \sigma_j \delta_j\right),$$

σ_i ist die Dichte am i-ten Punkt, δ_i ist der Abstand zwischen zwei benachbarten Abtastpunkten, c_i ist die Farbe am i-ten Punkt.

2.3.4 Optimierung eines NeRFs

Bisher wurden die Kernkomponenten für die Modellierung einer NeRF-Szene erläutert. Um jedoch state-of-the art Repräsentationen zu garantieren, benötigt es weiterer Techniken, die im folgenden vorgestellt werden.

Positional Encoding Obwohl neuronale Netze theoretisch universelle Funktionsapproximatoren sind, wurde festgestellt, dass das Arbeiten mit 3D-Koordinaten x, y, z und Richtungswinkeln ϕ, Φ zu schlechten Ergebnisse führt. Das Problem liegt hier in der Verarbeitung hochfrequenter Variationen in Farbe und Geometrie. Um dieses Problem zu umgehen, wird das sog. Positional Encoding eingesetzt. Diese Methode basiert auf den Arbeiten von Rahaman et. al [37], die zeigen, dass die Übertragung des Inputs in einen höherdimensionalen Raum das Netzwerk dazu veranlässt, bessere Ergebnisse zu erzielen. Die Methode besteht darin, die Eingangsdaten durch eine Funktion γ zu transformieren. Hierbei werden Sinus- und Kosinusfunktionen mit verschiedenen Frequenzen angewendet. Abbildung 11 veranschaulicht eine Szenenrepräsentation mit und ohne Positional Encoding.

$$\gamma(p) = \big(\sin(2^0\pi p), \cos(2^0\pi p), \sin(2^1\pi p), \cos(2^1\pi p), \ldots, \sin(2^{L-1}\pi p), \cos(2^{L-1}\pi p)\big)$$

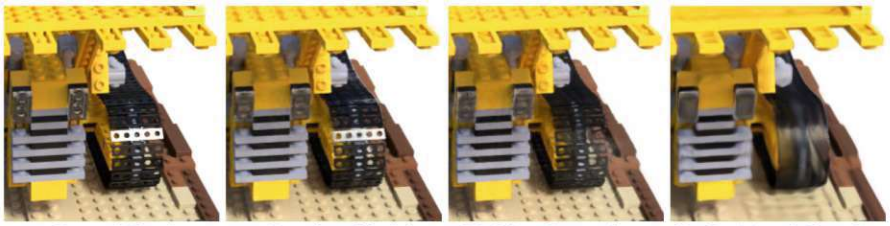


Abbildung 11: Verbesserung der Szene durch Positional Encoding

Hierarchical volume sampling Das Problem bei einer simplen Volumenabtastung ist, dass man eine große Anzahl von Proben nehmen muss, auch von Regionen, die möglicherweise leer sind oder keine relevanten Details enthalten. Um effizient Proben abzugreifen, wird HVS eingesetzt. HVS verfolgt das Prinzip eines coarse-to-fine Ansatzes. Zuerst wird eine grobe Schätzung der Szene berechnet, um Farb- als auch Dichteinformationen zu bestimmen. Basierend darauf wird eine Wahrscheinlichkeitsverteilung erstellt, die anzeigt, in welchen Bereichen des Strahls hohe Dichtewerte liegen. Nachdem eine grobe Abschätzung durchgeführt wurde, wird eine zweite Sampling-Schicht mit N_f Proben durchgeführt. Die Probenpunkte werden dabei basierend auf der vorher bestimmten Wahrscheinlichkeitversteilung platziert, sodass mehr Ressourcen auf detailliertere Bereiche der Szene gesetzt werden.
Die relevante Formel, die im Zusammenhang mit der groben Schätzung verwendet wird, ist:

$$C_c(r) = \sum_{i=1}^{N_c} w_i c_i$$

Hierbei ist $C_c(r)$ die geschätzte Farbe des Strahls nach der groben Sampling-Schicht, wobei N_c die Anzahl der Probenpunkte entlang des Strahls in der groben Sampling-Schicht darstellt. Das Gewicht der Probe i, w_i, wird durch die folgende Gleichung definiert:

$$w_i = T_i \left(1 - \exp(-\sigma_i \delta_i)\right)$$

Die Größe T_i bezeichnet die *akkumulierte Transmissionswahrscheinlichkeit* bis zum i-ten Punkt, die durch die Formel

$$T_i = \exp\left(-\sum_{j=1}^{i-1} \sigma_j \delta_j\right)$$

gegeben ist. Hierbei ist σ_i die Volumendichte am Punkt i, die angibt, wie wahrscheinlich eine Streuung oder Absorption an diesem Punkt ist, und δ_i bezeichnet den Abstand zwischen den Probenpunkten t_i und t_{i+1}. Schließlich steht c_i für die Farbe am Punkt i, die vom Netzwerk berechnet wurde.
Die Gewichte w_i bestimmen die Wahrscheinlichkeitsverteilung entlang des Strahls, basierend auf den Dichtewerten σ_i und der Transmissionswahrscheinlichkeit T_i. Diese Verteilung wird anschließend verwendet, um in der feinen Sampling-Schicht zusätzliche Proben in den relevanten Bereichen zu platzieren.

2.3.5 Weiterentwicklungen von NeRFs

Mip-NeRF Mip-NeRF[38] stellt eine Erweiterung des klassischen Vanilla-NeRF [34] dar, bei der mehrere Probleme aufgegriffen und verbessert werden. Das Hauptproblem, das Mip-NeRF löst, ist das Aliasing sowie die Unschärfe, die bei Vanilla-NeRF in Szenen auftreten. Dies passiert, weil NeRF nur einen Strahl pro Pixel verwendet, was zu verzerrten und unscharfen Darstellungen führen kann. Abgesehen davon ist Mip-NeRF um 7% schneller und halb so speicherintensiv. Dabei sinkt die Fehlerrate um 17%, wobei bei anspruchsvollen multiscale Varianten 60% weniger Abweichungen erreicht wurden. Umgesetzt wurde dies durch konische Frustums welche in Abb.12 zu sehen sind. Die punktförmige Strahldarstellung wird somit abgelegt und es ist möglich, die Szene auf mehreren Skalen darzustellen. Dies verringert wiederum das Aliasing.
Eine weitere Verbesserung stellt das “Integrated Positional Encoding“ dar. Diese berücksichtigt den räumlichen Bereich, den ein Pixel abdeckt, was zu einer verbesserten Darstellung auf mehreren Skalen führt. Vanilla-NerF verwendet zwei separate MLPs für die “coarse“ und “fine“ Samplingstrategie. Mip-NeRF kombiniert diese beiden MLPs in ein einziges Modell, was die Trainingsgeschwindigkeit verbessert und den Speicherbedarf optimiert.
Mip-NeRF-360[39] ist eine weitere Überarbeitung von Mip-NeRF und konzentriert sich auf die Darstellung unbeschränkter Szenen. Das bedeutet, dass bei Aufnahme des Objekts mit einem 360-Grad Ansatz, Details im Hintergrund im Falle Mip-NeRF nicht korrekt aufgelöst wurden. Abgesehen davon, konnte eine zusätzliche Reduzierung des Fehleranteils von 57% erreicht werden. Mip-NeRF 360 baut auf den Techniken von Mip-NeRF auf, erweitert sie aber durch die

Einführung von conical frustums, nicht-linearem Sampling, Mip Mapping und anderen Speicheroptimierungen. Diese Verbesserungen ermöglichen es, große, unendliche Szenen mit hoher Effizienz zu verarbeiten, während gleichzeitig Probleme wie Aliasing und Unschärfe bei weit entfernten Objekten optimiert werden. Nähere Informationen über die Umsetzung finden sich in der zugehörigen Literatur[39].

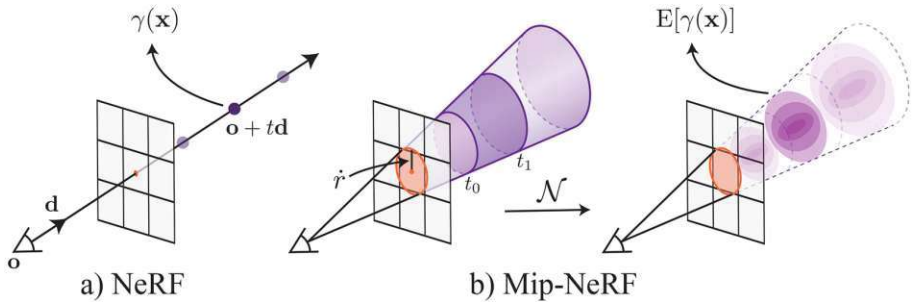


Abbildung 12: Mip-NeRF

NeRF– Eine weitere Entwicklung in der NeRF-Community wurde von Wang et al. (2021) [40] vorgestellt. Das Ziel von NeRF– ist es, Kameraparameter als zusätzlichen Input in das Netzwerk zu integrieren, um deren Optimierung direkt im Modell zu ermöglichen. Dadurch soll die Notwendigkeit vorangehender SfM-Methoden vollständig entfallen. Die Schätzung der Kamerapositionen wird als lernbarer Parameter betrachtet. Die Ergebnisse zeigten, dass NeRF-Modelle, deren Bilder Rotationen und Translationen relativ zum Objekt aufwiesen, bessere Resultate als herkömmliche SfM-Ansätze erzielten. Allerdings führte das Modell beim Objekttracking zu keiner Verbesserung. Zudem ist NeRF– auf Frontalaufnahmen beschränkt und unterstützt keine 360-Grad-Szenarien.

NeRF-W NeRF in the Wild (NeRF-W) [41] befasst sich mit der Erstellung von NeRFs aus Bildern, die aus unterschiedlichen Quellen stammen. Das bedeutet, dass beliebige Bilder aus verschiedenen Quellen genutzt werden, um ein NeRF zu generieren. Bei der Verwendung von Internetfotografien eines Objekts kann jedoch nicht von einer konsistenten Intensität der überlappenden Bilder ausgegangen werden. Dies liegt einerseits an photometrischen Variationen und andererseits am Vorhandensein bewegter Objekte. Erstere resultieren aus unterschiedlichen atmosphärischen Bedingungen zum Zeitpunkt der Aufnahme, die das Licht unterschiedlich reflektieren lassen, abhängig von den Witterungsverhältnissen. Zusätzlich enthalten Internetfotos häufig bewegte Objekte wie Menschen, Autos oder Tiere, was für 3D-Rekonstruktionen problematisch ist. NeRF-Modelle sind für statische Szenen konzipiert, und Bewegungen führen zu Artefakten in der Darstellung. Um dieses Problem zu lösen, wird ein zusätzlicher Vektor eingeführt. Das Appearance Embedding ist ein Vektor, der zusätzliche Informationen über die visuellen Eigenschaften eines Bildes speichert, insbesondere solche, die mit der Erscheinung der Szene zusammenhängen. Im Kontext von NeRF-W bezieht sich dies auf die Art und Weise, wie das Modell Unterschiede in den Bildaufnahmen behandelt, die durch variierende Beleuchtung,

Belichtungen oder Sichtwinkel verursacht werden.

Abbildung 13: NeRF-W: Input Bilder entstammen aus unterschiedlichen Quellen[41]

2.4 Zusammenfassung: DIM vs. NeRF

Die vorgestellten Methoden (DIM und NeRF) sind zum Zeitpunkt dieser Arbeit zwei der etabliertesten Ansätze, um 3D-Modelle aus Fotos zu entwickeln. NeRFs basieren auf dem Einsatz von Deep Learning, während DIM klassische Methoden der Photogrammetrie und der Computer Vision einsetzt. NeRFs verwenden ein FCNN (fully convolutional neural network) mit fest definierten Eingangsparametern, wobei das Netzwerk aus der diskreten Anzahl an Bildern die Szene erlernt und stark überangepasst ist. Dabei wird die Szene gesamtheitlich betrachtet, und mit Techniken wie Volumen-Rendering werden 3D-Objektpunkte ermittelt. DIM hingegen basiert auf rein geometrischen Ansätzen und der Korrespondenz von Pixeln in überlappenden Bildpaaren. Es sei erwähnt, dass der klassische NeRF-Workflow jedoch nicht vollständig von photogrammetrischen Methoden losgelöst ist. NeRF-Modelle erfordern die Kamerapositionen und Rotationen der Eingangsbilder. Diese sind mithilfe von Feature-Extraktion- und Feature-Matching-Techniken unverzichtbar. Erste Versuche, sich davon loszulösen, wurden mit NeRF– erprobt, erwiesen sich jedoch nicht für jede Aufnahmesituation als ausreichend. Daher kann zu diesem Zeitpunkt noch nicht von einer vollständig autarken Szenenrekonstruktion im Falle von NeRF gesprochen werden. Die eigentlichen Unterschiede betreffen die Art und Weise, wie die Szene rekonstruiert wird.

Klassische DIM-Verfahren haben daher einige Vor- und Nachteile gegenüber NeRFs. Im Falle texturarmer Objekte oder spiegelnder Oberflächen stoßen klassische photogrammetrische Ansätze an ihre Grenzen. Aufgrund der Veränderungen der Grauwertunterschiede oder des Fehlens von Grauwertsprüngen können keine Feature-Extraktionsverfahren angewandt werden. Die sparse Point Cloud enthält somit nur wenige Punkte in Bereichen mit geringer Textur oder sich ändernden Lichtverhältnissen [42]. Dadurch ist es nicht möglich, dichte Repräsentationen der

gewünschten Bereiche zu rekonstruieren. NeRF hingegen kann gut mit Reflexionen und texturarmen Oberflächen umgehen, da das neuronale Netzwerk die Lichtverhältnisse der Szene erlernen und korrigieren kann. Weiters sei erwähnt, dass DIM im Vergleich zu NeRFs eine geringere Abhängigkeit von der Anzahl der erfassten Bilder hat. Da NeRFs auf einem neuronalen Netzwerk basieren, ist die Anzahl der Bilder ausschlaggebend. Je mehr das Netzwerk mit Trainingsdaten versorgt wird, desto besser können Abhängigkeiten zwischen den Bildern erlernt werden. DIM hingegen ist aufgrund geometrischer Modelle weniger von der Quantität der Bilder abhängig. Durch Triangulation und lokale Untersuchungen von Merkmalen können photogrammetrische Auswertungen mit nur wenigen Bildern realisiert werden. Im Falle von NeRFs wäre dies für die Szenenrepräsentation nicht realisierbar.

Kriterium	Vorteile DIM	Nachteile DIM	Vorteile NeRF	Nachteile NeRF
Texturierte Oberflächen	Sehr präzise bei gut texturierten Objekten	Probleme bei texturarmen oder spiegelnden Oberflächen	Unabhängig von der Textur, kann ohne explizite Features arbeiten	Weniger präzise als DIM bei gut texturierten Oberflächen
Texturlose Oberflächen	Funktioniert schlecht, da keine Merkmale vorhanden	Hohe Fehleranfälligkeit bei nichtkooperativen Oberflächen	Überlegene Leistung bei texturlosen, reflektierenden oder transparenten Objekten	Erfordert viele Bilder für gute Ergebnisse
Reflektierende Oberflächen	Schwierige Verarbeitung, erzeugt oft Rauschen	Probleme durch spiegelnde Reflexionen	Besser im Umgang mit spiegelnden Oberflächen	Höhere Rechenanforderungen
Anforderungen Daten	Kann mit weniger Bildern arbeiten	Begrenzte Genauigkeit bei geringer Datenmenge	Erzeugt hochwertige 3D-Rekonstruktionen bei komplexen Szenen	Erfordert viele Bilder und ist rechnerisch aufwendig
Effizienz	Geringer Rechenaufwand	Begrenzte Fähigkeit zur Rekonstruktion feiner Strukturen	Kann komplexe Geometrien und Lichtverhältnisse gut modellieren	Sehr hoher Rechenaufwand und Speichernutzung
Feinstrukturen	Schwierigkeiten, feine Details exakt wiederzugeben	Glättungseffekte bei Oberflächen möglich	Bessere Rekonstruktion feiner Strukturen	Potenziell verrauscht bei hochfrequenten Details

Tabelle 2: Vergleich der Vor- und Nachteile von DIM und NeRF

3 Untersuchungsgebiete und Datensätze

3.1 Aufnahmegebiet

Die Messkampagne fand am 17. April 2024 in Mannersdorf an der March statt. Mannersdorf ist eine Ortschaft in Niederösterreich innerhalb der Katastralgemeinde Angern an der March. Ziel der Messkampagne war, die Rochuskapelle und eine angrenzende Baumgruppe mit einem

TLS (Terrestrischen Laserscanner) zu erfassen. Als passives Aufnahmesystem wurde ein UAV (Unmanned Aerial Vehicle) eingesetzt, welches im folgenden Kapitel näher beschrieben wird. Der resultierende TLS Datensatz dient als Referenz für die UAV-gestützen Bilddaten welche im Anschluss zu NeRF- bzw. DIM Daten weiterverarbeitet werden.

Abbildung 14: Überblickskarte. Bezogen von OpenStreetMap [43]

Abbildung 15: Rochuskapelle mit Schachbrettmuster im Vordergrund

3.1.1 Metereologische Daten

Die Wetterbedingungen wurden durch wechselhafte Wolkenbedeckung und moderaten Temperaturen beeinflusst. Die Wolkenbedeckung belief sich durchschnittlich auf 52% (s. Abbildung 16). Wechselhafte Lichtverhältnisse bzw. resultierender Schattenwurf der zu erfassenden Objekte wirken sich auf die Rekonstruktionsqualität von 3D Modellen aus.
Die Temperatur (s. Abbildung 17) hat im Falle der vorliegenden Arbeit keine Auswirkungen auf das resultierende Modell. Es wurde in geringen Flughöhen operiert und daher kann die Refraktion außer Acht gelassen werden.
Der Sonnenstand laut Abbildung 18 war zum Zeitpunkt der Messung zwischen 40 und 50 Grad. Der Elevationswinkel der Sonne wirkt sich direkt auf den Schattennwurf von Objekten aus. Dies ist wie eingangs erwähnt für nachfolgende photogrammetrische Auswertungen als Nachteil einzustufen.
Die Windgeschwindigkeit betrug am 17.04 rund 19 km/h. Da die Versuchsobjekte freistehend auf einem Hügel positioniert sind, ist der Durchschnittswert laut [44] als Minimalgeschwindigkeit einzustufen. Die Messungen laut [45] zeigten Windgeschwindigkeiten bis zu 30km/h. Dieser Umstand wirkte sich auf die Erfassung der Baumgruppe aus.
Tabelle 3 fasst die meteorologischen Verhältnisse zum Zeitpunkt der Messung zusammen. Insgesamt lässt sich sagen, dass die Wetterbedingungen für das Experiment als mäßig einzustufen sind. Wechselhafte Bedingungen, wie zum Beispiel wechselnde Licht- und Schattenverhältnisse,

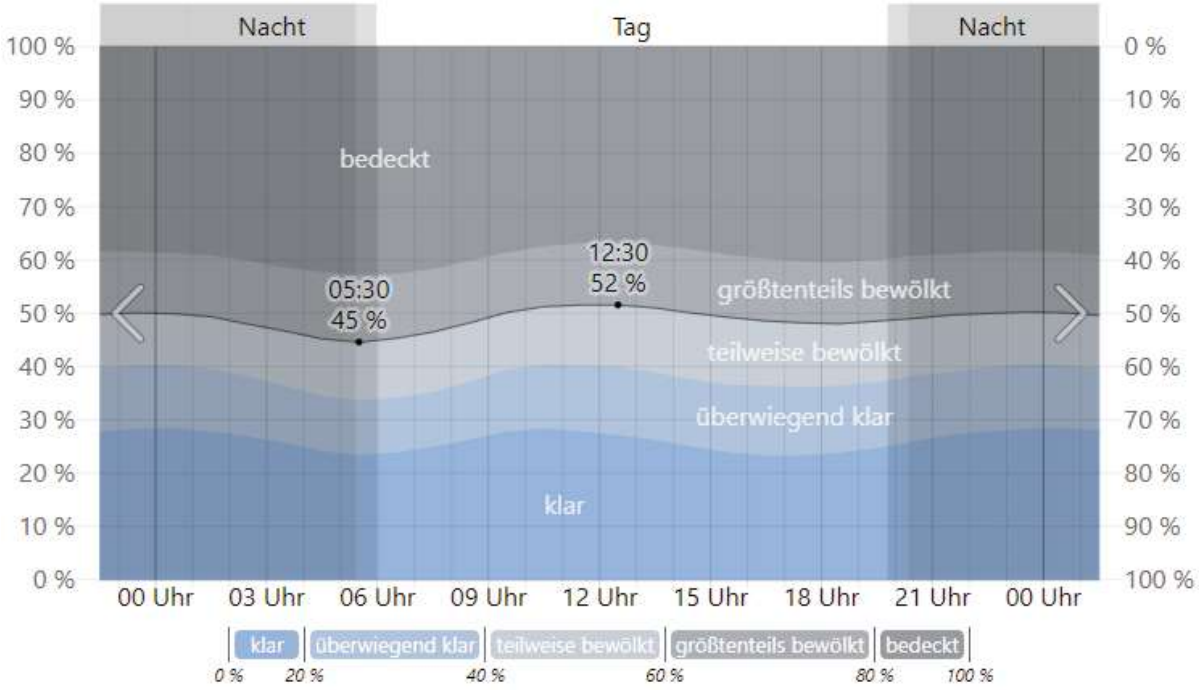


Abbildung 16: Wolkenbedeckung am 17.04.2024

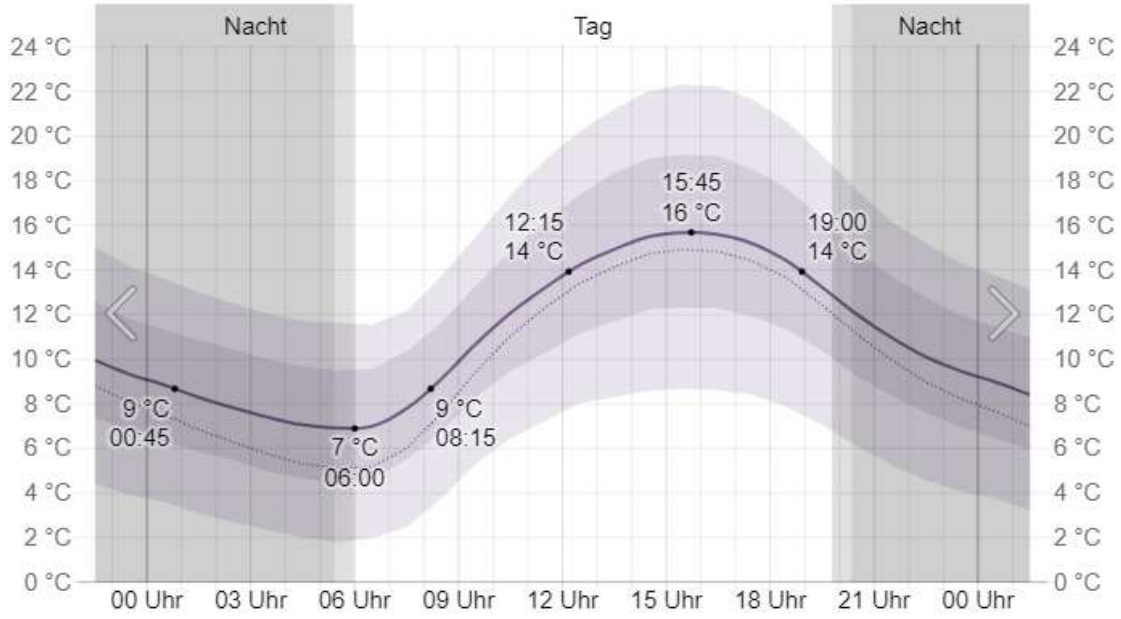


Abbildung 17: Durchschnittstemperatur am 17.04.2024

sind für photogrammetrische Messkampagnen kein Vorteil. Die Versuchsobjekte waren räumlich begrenzt, und daher wurden die Flugrouten innerhalb weniger Minuten abgeschlossen. Aufgrund dieser kurzen Dauer können die meteorologischen Bedingungen während der Messung als weitgehend konstant betrachtet werden.
Die angeführten wetterbezogenen Daten wurden aus dem Archiv von [44] etnommen.

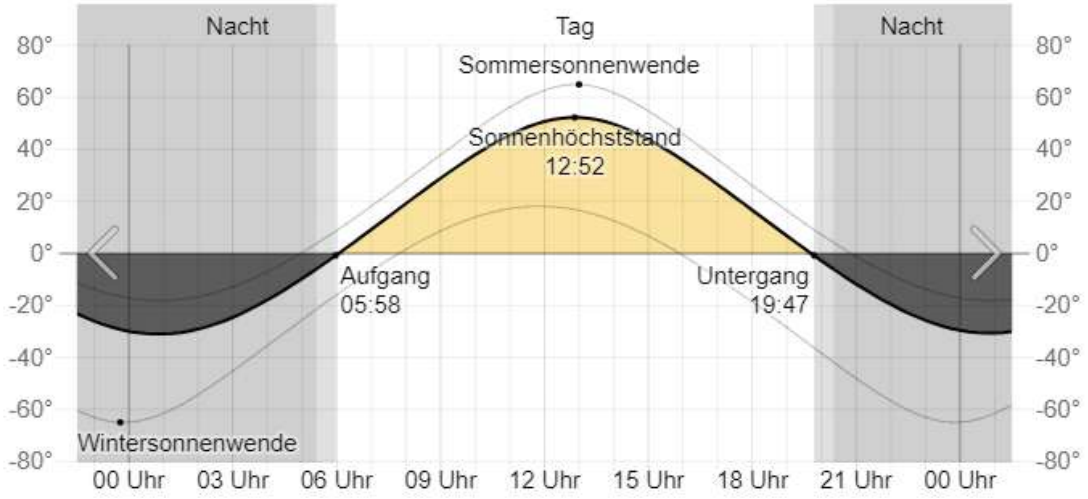


Abbildung 18: Sonnenstand am 17.04.2024

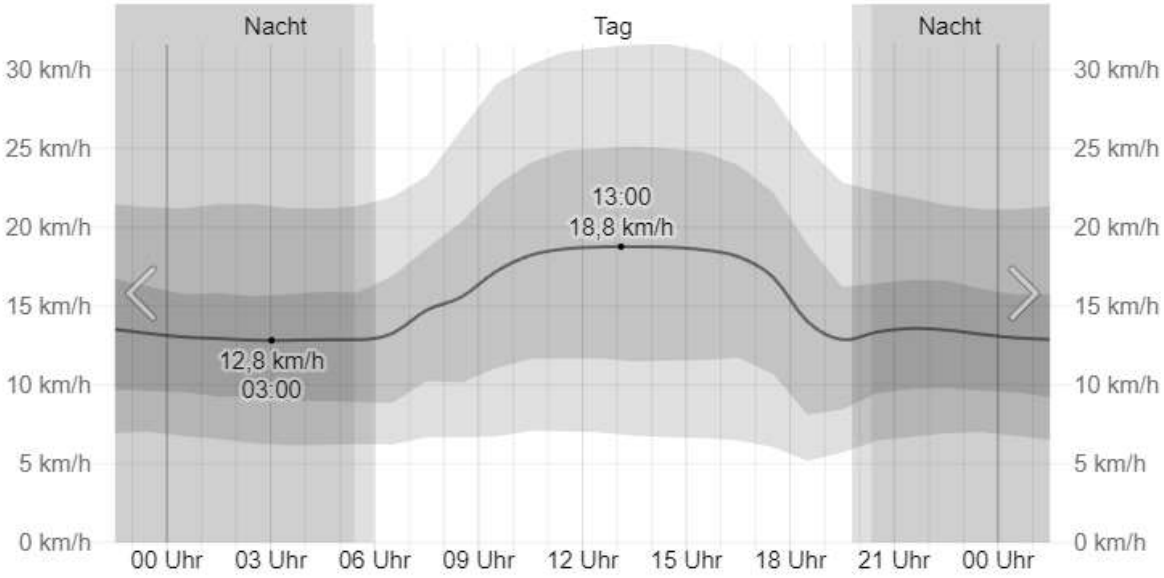


Abbildung 19: Windgeschwindigkeit am 17.04.2024

Datum:	17.04.2024
Ort:	Mannersdorf an der March
Dauer der Kampagne:	09:00 - 13:00 Uhr
Temperatur:	10 - 15 °C
Wolkenbedeckung:	wechselhaft 50%
Sonnenstand:	40 - 50 °
Windgeschwindigkeit:	bis 30 km/h
Aufnahmeobjekte:	Rochuskapelle, Baumgruppe

Tabelle 3: Überblick Messkampagne

3.2 Datenerhebung

3.2.1 Vermessungsgeräte

Für die Datenerhebung der Messkampagne wurde eine umfassende Ausstattung verschiedenster Aufnahmesysteme eingesetzt. In die Kategorie der aktiven Aufnahmesysteme sei hier die To-

talstation von Leica (Modell TS13) und RIEGL VZ-600i einzugliedern. Die Totalstation wurde in Kombination mit einem GNSS zur Gewinnung von Kontrollpunkten eingesetzt. Die Kontrollpunkte, meistens in Form von Schachbrettmustern, sind von Bedeutung für nachfolgende Qualitätsanalysen. Zuätzlich wurden Reflexionsfolien von der Firma Riegl eingesetzt, welche für die Registrierung und Georeferenzierung der TLS Punktwolke verwendet wurden. Um RTK-Korrekturdaten für das GNSS als auch TLS-System zu erhalten, wurde APOS (Austrian Positioning System) und EPOSA (Echtzeit Positionierung Austria) eingesetzt. Informationen zu dem gewählten Koordinatenrahmen sind im Kapitel 3.3.1 verfügbar.

Gerätetyp	**Hersteller**	**Modell**	Einsatz
Totalstation	Leica	TS13	Messung Passmarken
TLS	Riegl	VZ-600i	Referenzpunktwolke
UAV	DJI	M350	Fluginstrument/Flugplan
UAV-Kamera	DJI	Zenmuse P1	Bildaufnahme 45MP
GNSS	ESurvey	eSA2	RTK-Messung Bodenpunkte

Tabelle 4: Equipment zur Datenerhebung

3.3 Experiment

3.3.1 Koordinatenrahmen

Geodätische Messungen erfordern einen Ortsbezug. Um eine Verortung der Versuchsobjekte zu erreichen, wurde das Koordinatenreferenzsystem EPSG:25833, was dem ETRS89 UTM Zone 33N entspricht, gewählt. Der Grund für die Wahl des UTM-Systems ist in Abbildung 20 zu sehen. Die Abbildung zeigt auf der linken Seite das System MGI, wohingegen auf der rechten Seite das System ETRS89 mit dem Ellipsoid GRS80 ersichtlich ist. APOS Messungen beziehen sich auf das Europäische Referenzsystem ETRS89, wobei EPOSA zum Zeitpunkt dieser Arbeit den Bezugsrahmen ITRF2020 Epoche 2015.0 einsetzt.
Um in das Landessystem von Österreich zu transformieren, wird vom System ETRS89 in das System MGI transformiert. Da im Falle dieser Arbeit kein Zwang zur Verortung im Landessystem existiert, wurde die Umrechnung in das UTM System umgesetzt. UTM - Koordinaten sind weltweit vergleichbar und daher für technische Realisierungen geeignet. Die Schritte sind Abbildung 20 zu entnehmen. Nähere Informationen und Formelapparate sind in [46] zu finden.

3.3.2 Die Versuchsobjekte

Abbildungen 21 und 23 zeigen die resultierenden Orthophotos der Messkampagne. Die Geometrie der Rochuskapelle ist vorwiegend zylindrisch. Die Fassade ist hauptsächlich weiß und weist nur wenig Textur auf. Die Kapelle besitzt im unteren Drittel feinere Strukturen (Stuck), welche hinischtlich der Rekonstruktionsqualität von NeRFs und DIM von Interesse sind. Am Kopf der Kapelle befindet sich ein Kreuz. Die Frage der Qualität bzw. Existenz im resultierenden Modell ist daher von Interesse. Die Baumgruppe befindet sich in unmittelbarer Nähe zur Kapelle. Da

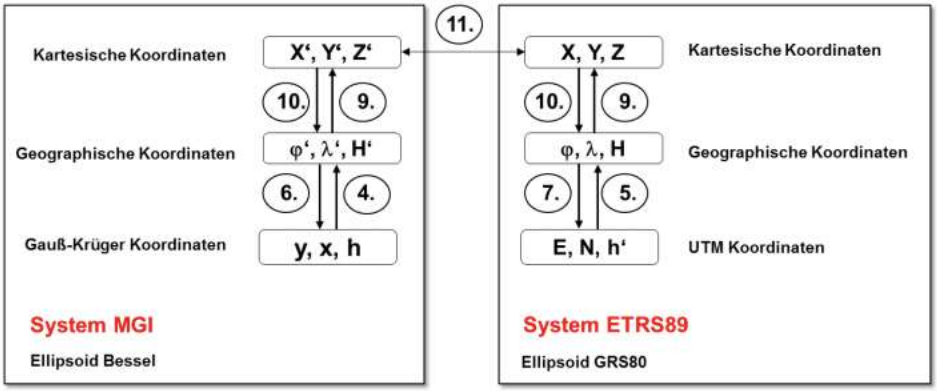


Abbildung 20: Transformation zwischen ETRS & MGI [46]

Bäume, insbesondere Äste und Blätter, durch den Einfluss des Windes ständig in Bewegung sind, konnten keine optimalen Bedingungen für die Aufnahme gewährleistet werden. Diese dynamischen Gegebenheiten stellen eine Herausforderung für SfM (Structure from Motion) Methoden dar. Die Auswirkungen auf die Qualität der Rekonstruktionen sind daher fraglich.

Abbildung 21: Rochuskapelle Orthophoto

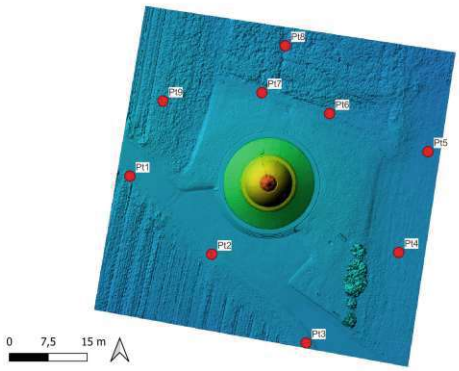


Abbildung 22: Verteilung Passpunkte Kapelle

Abbildung 23: Baum Orthophoto

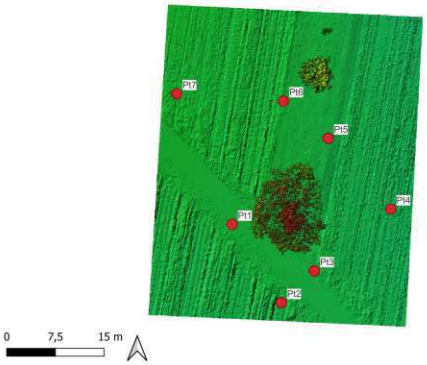


Abbildung 24: Verteilung Passpunkte Baum

Abbildung 25: Detailansicht Kapelle

3.3.3 Versuchsaufbau

Um Kontrollpunkte für die nachgehenden Auswertungen zu besitzen, wurden Schachbrettmuster (GCPs) im Interessensgebiet des Objektes verteilt. Diese dienen im photogrammetrischen Auswerteprozess zur Kontrolle des resultierenden Modells[7]. Je nach Wahl der Auswertesoftware (im Falle dieser Arbeit Agisoft Metashape), können diese Bezeichnungen leicht abweichen. Abbildung 26 zeigt ein Schachbrettmuster wobei Abbildung 27 ein Retrotarget, für die Registrierung der TLS Punktwolke, visualisiert.

Abbildung 26: Schachbrettmuster Riegl

Abbildung 27: Retrotarget Riegl

[7]Checkpoints dienen zur Kontrolle des prozessierten Modells. Im Gegensatz dazu werden Controlpoints zur Georeferenzierung eingesetzt.

Der Mittelpunkt der Vermarkung wurde mit einem GNSS, RTK-unterstützt für eine Minute beobachtet. Die resultierenden Koordinaten sind im ETRS89 System als Geographische Koordinaten gelagert. Bei näherer Betrachtung von Abbildung 22 kann man die Verteilung der Schachbrettmuster erkennen. Analog wurde der gleiche Prozess für die Baumgruppe durchgeführt.
Als weiteren Schritt wurden retroreflektive Targets (s. Abbildung 27) an stabil befestigte Objekte geklebt. Retroreflektive Targets werden innerhalb RiScanPro[8] automatisch erkannt und dienen zur Georeferenzierung der Laserscanning Punktwolke. Um die Totalstation in das benötigte Koordinatensystem zu transformieren, wurde eine freie Stationierung über die Schachbrettmuster durchgeführt[9]. Anschließend wurden die retroreflektiven Targets reflektorlos eingemessen und gespeichert. Da die Kontrollpunkte nun versichert wurden, konnte als nächster Schritt die vorab definierte Flugplanung umgesetzt werden.

3.3.4 Linear vs. Orbitflug

Die Flugplanung wurde zwecks späterer Analysen in einen "orbitalen" als auch "linearen" Flug unterteilt. Ein linearer Flugplan wird im Sinne dieser Arbeit folgendermaßen definiert: Die Kamera zeigt in das Nadir (90 Grad) und dem UAV wird erlaubt einen double-grid Flug zu absolvieren. Durch diese Fluganordnung ist die Kamera beschränkt auf eine Winkelstellung und die Sicht zum Objekt weniger gegeben als bei einem "orbitalen" Flug. Anzumerken ist, dass zwecks Höhenstabilisierung des Blockes, quer über das Interessensgebiet Schrägaufnahmen gemacht werden (s. Abbildung 35).
Dem "orbitalen" Flug ist erlaubt, unterschiedliche Winkelstellungen der Kamera einzunehmen. Das Versuchsobjekt wird somit von unterschiedlichen Positionen abgelichtet. Es sei zu erwähnen, dass die Wahl des Flugplanes vom gewünschten Ergebnis abhängt. Für die Erstellung eines Orthophotos würde ein double-grid Muster Redundanz für die Auswertung schaffen, jedoch trägt hier meistens der zusätzliche Aufwand nicht die Kosten. Um vollständige 3D Modelle eines Objektes zu generieren[10], steht dem Mehraufwand nichts im Wege. Die Vollständigkeit des Objektes steht hier im Vordergrund. Abbildung 28 zeigt den Unterschied zwischen einem single-grid und double-grid Muster. Abbildung 29 und 30 zeigen den Unterschied zwischen den "orbitalen" und "linearen" Flug. Diese Begriffe wurden für diese Arbeit eigenständig definiert und können von der zugehörigen Literatur abweichen.

3.3.5 Resultierende Datensätze

Die resultierenden Datensätze setzen sich zusammen aus Orthophotos (s. Abbildung 21 und 23), Punktwolken aus Dense Image Matching, Punktwolken aus Neural Radiance Fields und einer Referenzpunktwolke des terrestrischen Laserscanners. Für jedes Versuchsobjekt stehen somit 4 verschiedene Datensätze zur Verfügung.

[8] RiScan Pro ist eine Software von Riegl zur Registrierung und Verarbeitung von Riegl Punktwolken
[9] Mithilfe einer Spinne und einem Prisma
[10] Die räumlich weniger ausgedehnt sind

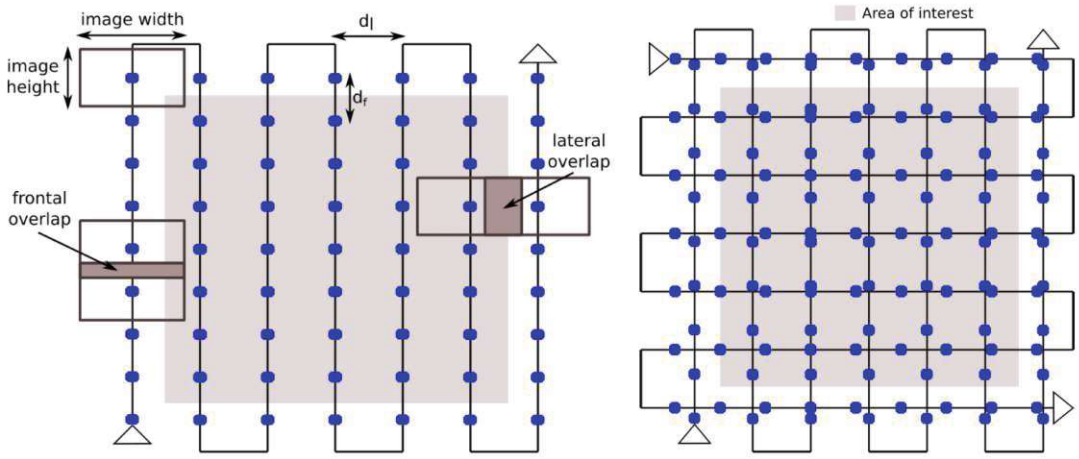


Abbildung 28: single-grid vs. double-grid [47]

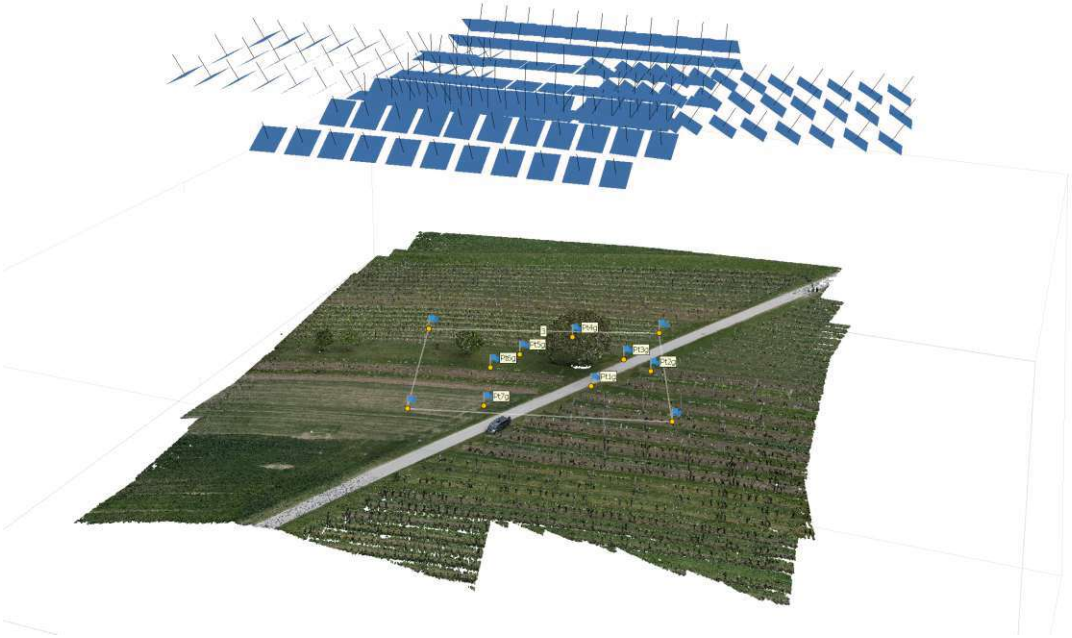


Abbildung 29: Orbit Flug Baum

Datensatz	**Methode**	**Punktanzahl**	**Punktdichte (pts/m²)**	**Min (Z)**	**Max (Z)**
Kapelle	TLS	7,682,443	6,841.62	232.665	260.899
linear Kapelle	DIM	4,554,278	4,824.50	232.682	260.510
orbit Kapelle	DIM	5,017,765	4,097.40	232.684	260.595
linear Kapelle	NeRF	2,121,324	2,214.44	232.493	260.665
orbit Kapelle	NeRF	1,337,722	1,427.49	232.479	260.511

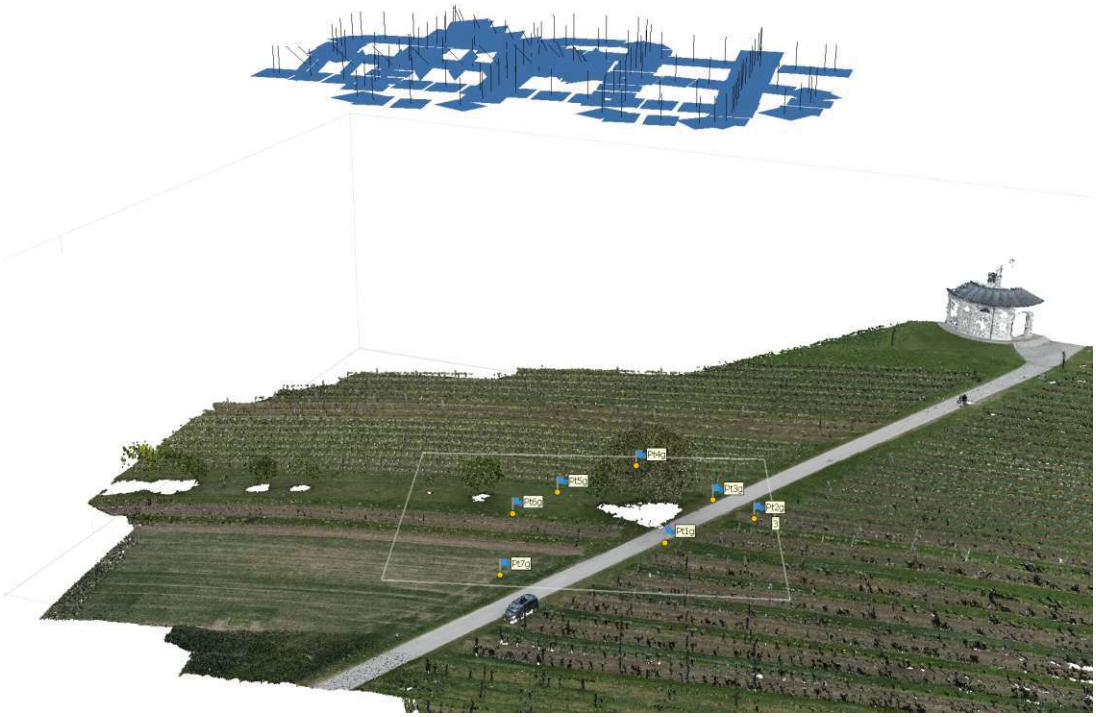


Abbildung 30: Linear Flug Baum

	Linear Kapelle	Orbit Kapelle	Linear Baum	Orbit Baum
Flugdauer [min]	9	8	7	5
Anzahl Fotos	220	213	130	181
Flughöhe [m]	60	60	60	60
Startzeit - Endzeit	13:30 - 13:39	13:55 - 14:03	13:43 - 13:50	14:12 - 14:17
Brennweite [mm]	35	35	35	35
Verschlusszeit	1/1000	1/1000	1/1000	1/1000
GSD [cm]	0.8	0.8	0.8	0.8
Tie points	690k	692k	410k	604k

Tabelle 5: Übersicht über den Flugplan und die Aufnahmedaten

Tabelle 7: Datensatz Kapelle

Tabelle 7 zeigt die wichtigsten Parameter der Kapellenpunktwolke. Die Punktwolken wurden auf die gleiche Ausdehnung (X,Y) zugeschnitten, somit können Vergleiche zwischen den Datensätzen durchgeführt werden.

Die Gegenüberstellung der Punktdichten zeigt, dass DIM-basierte Verfahren eine höhere Punktdichte aufweisen als NeRF-basierte Verfahren. Im Falle der Min-Max Werte zeigt sich, dass alle Datensätze die maximale Höhe der TLS-Punktwolke nicht erreichen. Somit ist sicher, dass das Kreuz an der Kapelle nicht vollständig abgebildet wurde. Die Höhenunterschiede betragen bei Max-Z bis zu 40 cm. Andererseits sieht man, dass die TLS Punktwolke eine minimale Höhe von 232.67 Meter aufweist. In diesem Fall liegen die DIM-basierten Auswertungen nahe an der

Referenz. Betrachtet man die NeRF Auswertungen, erkennt man, dass tendeziell zu tief gemessen wurde. Dies liegt daran, dass der Boden[11] der Punktwolke eine hohe Anzahl an Ausreißer besitzt. Die Präzision des rekonstruierten Bodens ist nicht von hoher Qualität. Dies hat mehrere Gründe, welche im Laufe der Arbeit noch diskutiert werden. Da es sich bei den oben erwähnten Vergleichen nur um einen konkreten Punkt[12] handelt, kann keine statistische Einschätzung zu diesem Zeitpunkt gemacht werden. Nähere Betrachtungen finden sich im Kapitel 5.

3.4 Probleme Baumdatensatz

Die Datensätze des Baumes sind anders zu behandeln als die der Kapelle. Aufgrund des vorherrschenden Windes während der Aufnahmen können keine direkten Vergleiche, wie sie bei der Kapelle durchgeführt wurden, gezogen werden. Insbesondere in den NeRF-Datensätzen zeigen sich erhebliche Qualitätsprobleme. Die resultierenden NeRF-Punktwolken des Baumes weisen Verzerrungen auf, und die Registrierung auf die TLS-Baumpunktwolke konnte nicht erfolgreich durchgeführt werden. Ein spezifisches Problem trat beim Export der Punktwolke aus Nerfstudio auf, das die lineare Route nicht exportieren konnte.

3.4.1 Probleme beim Export der Punktwolke

Während der Arbeit mit Nerfstudio konnte der Punktwolkenexport für die Baumgruppe, die mit einem linearen Flugmuster aufgenommen wurde, nicht erfolgreich durchgeführt werden. Dies steht im Gegensatz zu den orbital Flügen, bei denen der Export ohne Probleme möglich war. Es gibt mehrere Gründe, warum das lineare Flugmuster zu diesem Problem führte:

Unzureichende Ansichten und Abdeckungen: Ein lineares Flugmuster, bei dem die Kamera in einem konstanten Nadir-Winkel (90 Grad) nach unten gerichtet ist, bietet im Vergleich zu einem orbitalen Muster eine begrenzte Sicht auf die Baumstruktur. Bäume, insbesondere deren Äste und Blätter, erfordern eine Vielzahl von Blickwinkeln, um die Geometrien und Details korrekt zu erfassen. Da beim linearen Flug weniger Blickwinkel vorhanden sind, fehlen möglicherweise Informationen, um eine vollständige und dichte Punktwolke zu erzeugen. Dieser Mangel an Daten führt dazu, dass beim Export der Punktwolke Lücken oder unvollständige Geometrien auftreten.

Bewegungen durch Wind: Der Wind beeinflusste die Baumgruppe während der gesamten Aufnahmezeit. Diese Bewegungen führten dazu, dass die Positionen der Äste und Blätter nicht konstant blieben. NeRFs, die auf der Annahme statischer Szenen basieren, können möglicherweise die Bewegungen nicht korrekt modellieren. Während das Modell bei der orbitalen Flugroute noch ausreichend Daten für eine teilweise Korrektur hatte (durch verschiedene Blickwinkel), bot der lineare Flug zu wenige Informationen, um diese Bewegungen auszugleichen. Dies

[11] Asphalt, Gras, Beton
[12] den minimalen und maximalen Punkt

führte zu Inkonsistenzen im rekonstruierten 3D-Modell und machte einen erfolgreichen Export der Punktwolke unmöglich.

Hohe Anfälligkeit für Rauschen und Bewegungsunschärfe: Beim linearen Flug mit gleichbleibendem Nadir-Blickwinkel entstehen oft Rauschprobleme durch Bewegungsunschärfe, die durch den Wind verursacht wurde. Während ein orbitaler Flugplan die Möglichkeit bietet, diese Effekte durch die Erfassung von mehr Blickwinkeln zu glätten und zu kompensieren, verschärft der lineare Flug die Auswirkungen der Bewegungsunschärfe. Das NeRF-Modell kann diese nicht ausreichend korrigieren, was den Exportprozess behindert. Nerfstudio könnte Schwierigkeiten haben, das Rauschen zu eliminieren, was zu einem fehlerhaften Export führt.

Unvollständige Rekonstruktion des unteren Bereichs des Baumes: Beim linearen Flugmuster wird vorwiegend die Baumkrone abgebildet, während die unteren Bereiche des Baumes, wie der Stamm und die unteren Äste, nicht vollständig erfasst werden. In einer orbitalen Flugroute würde die Kamera aus verschiedenen Winkeln auch die unteren Teile des Baumes aufnehmen, was eine vollständige Rekonstruktion ermöglicht. Das Fehlen dieser wichtigen Daten in der linearen Route könnte ein weiterer Grund dafür sein, dass der Export der Punktwolke fehlschlägt, da eine vollständige, zusammenhängende Geometrie erwartet wird.

4 Methodik

Dieses Kapitel beschreibt die angewandten Methoden, die dazu dienen, die Unterschiede zwischen DIM und NeRFs herauszuarbeiten. Dabei kamen verschiedene Werkzeuge zum Einsatz: Zum einen wurden qualitative Analysen in Form von Visualisierungen durchgeführt, zum anderen wurden statistische Kennzahlen herangezogen, um die Ergebnisse quantitativ zu untermauern.

4.1 Datenverarbeitung und Analyse

Das folgende Kapitel gibt eine detaillierte Beschreibung des Datenverarbeitungsablaufs. Es wird auf die verwendeten Softwareprodukte eingegangen und abschließend die resultierenden Analysen skizziert.
Abbildung 31 zeigt den Ablauf der Datenverarbeitung. Im Wesentlichen ist der Workflow in fünf Schritte zu unterteilen.

4.1.1 Agisoft Metashape

Agisoft Metashape ist eine Software, die photogrammetrische Verarbeitung digitaler Bilder durchführt und 3D-Daten erzeugt, welche in GIS-Anwendungen, der Dokumentation von Kulturerbe und der Produktion von visuellen Effekten sowie für indirekte Messungen von Objekten unterschiedlichen Maßstabs verwendet werden können[48].

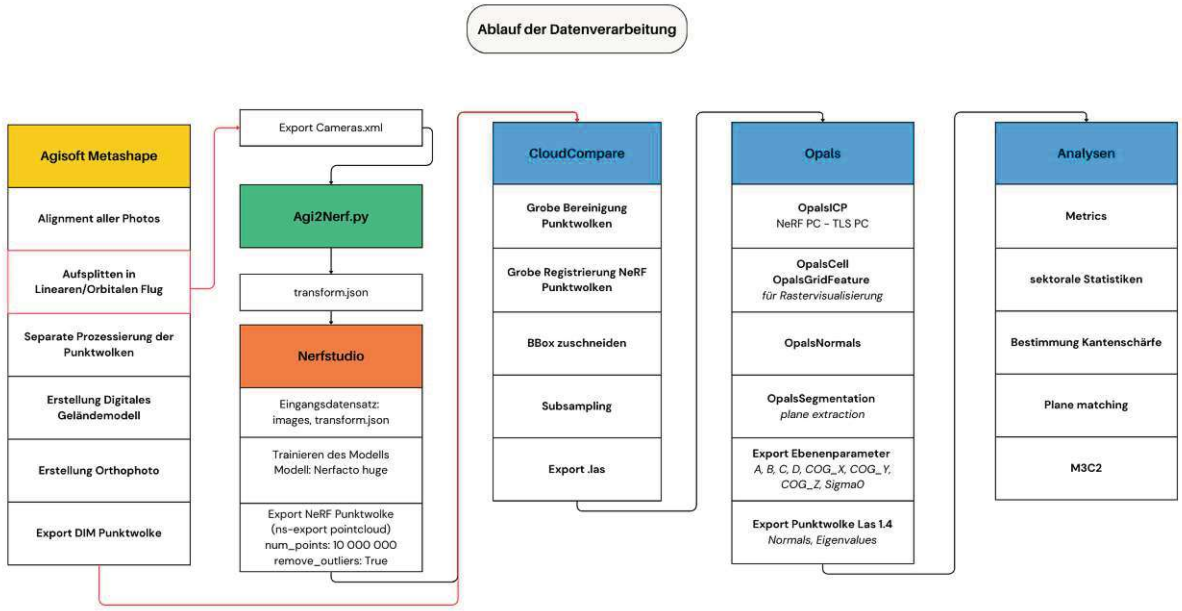


Abbildung 31: Ablauf der Datenverarbeitung

Für diese Arbeit wurde Agisoft Metashape eingesetzt, um DIM-Punktwolken zu erzeugen. Der Workflow unterteilt sich in vier wesentliche Schritte. Zuerst werden die erfassten Fotos mithilfe eines Bündelblockausgleichs ausgerichtet[13]. Damit der lineare als auch orbitale Datensatz das gleiche geodätische Datum besitzen, wurde das Alignment gesamtheitlich durchgeführt. Das bedeutet, dass die Fotos beider Flüge gemeinsam optimiert wurden. Dieser Ansatz sichert die Identität des geodätischen Datums beider Flugroutinen, und beide Modelle basieren auf den gleichen GCPs. Nach dem initialen Alignment wurde der „Chunk“[14] in einen linearen und einen orbitalen Workspace aufgeteilt. Danach wurde der übliche Workflow fortgesetzt.

Da die Exif-Information jedes Bildes RTK-unterstützte ETRS89-Koordinaten enthält, ist bereits nach der initialen Ausrichtung der Fotos ein Sparse-Pointcloud-Modell das Ergebnis. Die Sparse-Pointcloud ist noch nicht als DIM-Punktwolke zu betrachten. Da im Prozess des Feature Matching Ausreißer auftreten, sollte die Sparse-Pointcloud näher untersucht werden. Agisoft Metashape besitzt ein Gradual Selection Tool, welches Verknüpfungspunkte, die einen gewissen Schwellenwert überschreiten, eliminiert. Das Gradual Selection Tool umfasst die Auswahl und Filterung nach Reprojektionsfehlern, Rekonstruktionsunsicherheiten und Projektionsgenauigkeit.

Für die Wahl geeigneter Parameter kann keine generelle Aussage getroffen werden. Die Wahl der Parameter hängt von dem erfassten Objekt/Gebiet ab. Dennoch sei an dieser Stelle zu erwähnen, dass der Einsatz des Gradual Selection Tools die Genauigkeit des Modells positiv beeinflusst, da die relative Orientierung durch Filterung von grob falschen Merkmalspunkten verbessert wird.

[13] Die Kamerapositionen und Winkelstellungen werden gesamtheitlich optimiert
[14] Bestehend aus allen Fotos (linear, orbital)

Um die Georeferenzierung des Modells zu stützen, ist der Einsatz von GCPs (Ground Control Points) von Relevanz. Diese sollten vor der ersten Optimierung in den Workspace importiert werden. Nach der Aufteilung in Checkpoints und Kontrollpunkte kann ein erneuter Bündelblockausgleich gerechnet werden, der anhand der GCPs das bestehende Modell verbessert.
Um ein NeRF zu prozessieren, ist es notwendig, die Kamerapositionen und Winkelstellungen zu kennen. Eine kostenfreie Variante findet sich mit Colmap[49]. Innerhalb von Agisoft Metashape werden die Kamerapositionen, Rotationen sowie die Parameter der inneren Orientierung in eine separate Datei exportiert. Die XML-Datei dient dabei als Input für Agi2NeRF.

4.1.2 Agi2NeRF.py

Bevor die Funktionsweise von Agi2Nerf erläutert wird, sei an dieser Stelle angemerkt, dass es mehrere Möglichkeiten gibt, das folgende Ergebnis zu erreichen. Nerfstudio (s. Kapitel 4.1.3) besitzt eine eigene Funktionalität, um Agisoft Metashape XML-Dateien in ein geeignetes Nerfstudio-Format zu konvertieren. Im Falle dieser Arbeit wurde jedoch Agi2Nerf.py eingesetzt.
Agi2Nerf (vgl.[50]) ist ein Skript, um eine Agisoft Metashape XML-Datei[15] in ein JSON-Format zu konvertieren. Zusätzlich werden die Bildschärfe und der zentrale Blickpunkt der Kameras bestimmt.
Das Kamerakoordinatensystem für NeRFs folgt der OpenGL/Blender-Konvention (vgl.**DataNerfstudio**). Hierbei zeigt die X+-Achse nach rechts, Y+ zeigt aufwärts, und Z+ zeigt in die gegenläufige Richtung der Kamera. Sprich, die -Z-Achse ist die „look-at“-Richtung.
Der Output des Skripts ist eine transforms.json-Datei.

Parameter	**Wert**
`camera_angle_x`	0.93
`camera_angle_y`	0.64
`fl_x`	8196.29
`fl_y`	8196.29
`k1`	-0.05
`k2`	0.02
`p1`	-0.00
`p2`	0.00
`cx`	4049.59
`cy`	2778.33
`w`	8192.00
`h`	5460.00
`aabb_scale`	16
`file_path`	`./images/DJI_20240417135519_0001.jpg`
`sharpness`	1800.51

Tabelle 8: Auszug aus transform.json

[15] Mit Kamerakalibrierungs- und Bildtransformationsdaten

transform_matrix :

$$\begin{bmatrix} -0.39 & -0.50 & 0.77 & 5.86 \\ 0.88 & 0.04 & 0.47 & 4.36 \\ -0.26 & 0.87 & 0.42 & 0.77 \\ 0.00 & 0.00 & 0.00 & 1.00 \end{bmatrix} \qquad \begin{bmatrix} X0 & Y0 & Z0 & X \\ X1 & Y1 & Z1 & Y \\ X2 & Y2 & Z2 & Z \\ 0.0 & 0.0 & 0.0 & 1 \end{bmatrix}$$

Tabelle 8 zeigt einen Ausschnitt der transforms.json-Datei[16]. Diese beinhaltet mehrere Parameter, die im folgenden erklärt werden.
Die horizontalen und vertikalen Blickwinkel der Kamera werden durch die Werte für camera_angle_x und camera_angle_y definiert. Die Brennweiten in Pixeln in der x- und y-Richtung sind als fl_x und fl_y angegeben. Die radialen Verzeichnungskoeffizienten werden durch k1 und k2 beschrieben, während p1 und p2 die tangentialen Verzeichnungskoeffizienten repräsentieren. Das Projektionszentrum, auch als Principal Point bekannt, ist durch die x- und y-Koordinaten cx und cy festgelegt. Die Bildbreite und -höhe in Pixeln sind mit w und h angegeben. Der Parameter aabb_scale beschreibt den Skalierungsfaktor für die achsenorientierte, begrenzende Box (AABB). Schließlich gibt der Wert sharpness die Schärfe des Bildes an.
Die resultierende Datei ist der Startpunkt für das Trainieren eines NeRFs.

4.1.3 Nerfstudio

Nerfstudio ist ein Python Framework für die Entwicklung und Erstellung von NeRFs. Nerfstudio unterstützt verschiedene Eingangsdatensätze und besitzt einen Echtzeit Web-Viewer, um den Fortschritt des Trainings zu überwachen. Das Ziel von Nerfstudio ist, die Entwicklung von NeRF Methoden und Rekonstruktionen für praktische als auch wissenschaftliche Einsatzgebiete zu vereinfachen[51].
Viele bestehende NeRF Implementierungen konzentrieren sich hauptsächlich darauf, Metriken wie PSNR (Peak Signal-to-Noise Ratio), SSIM (Structural Similarity Index) und LPIPS (Learned Perceptual Image Patch Similarity) zu optimieren. Diese Metriken bewerten die Qualität der rekonstruierten Bilder im Vergleich zu den Originalen. Hier sei jedoch zu erwähnen, dass diese Bewertungen meistens auf Bildern basieren, die sehr ähnlich zu den Trainingsdatensätzen sind. In realen Anwendungen ist dies nicht zutreffend, insbesondere wenn unstrukturierte Umgebungen[17] erfasst werden. Auch zu erwähnen sind die hohen Anforderungen an die eingesetzte Hardware. Die Renderzeiten pro Bild betragen oft mehrere Sekunden, was für praktische Einsätze unwirtschaftlich ist. Die neuesten Entwicklungen zwecks Optimierung der Renderzeiten wurden mit Instant-NGP eingeführt. Für die Nutzung von Instant-NGP benötigt man jedoch eine leistungsstarke GPU und spezielle CUDA-Kerne. Dies erschwert die Entwicklung, weil nicht jeder Zugang zu dieser Hardware und den notwendigen Programmierkenntnissen hat.
Nerfstudio implementiert die sogenannte Nerfacto Methode. Diese Methode wurde inspiriert von MipNeRF-360 [39], wobei Anpassungen in [51] Nerfstudio durchgeführt wurden, um die

[16] Für jedes Bild gibt es eine eigene Transformationsmatrix
[17] große Unterschiede in den Aufnahmepositionen

Performance zu optimieren.

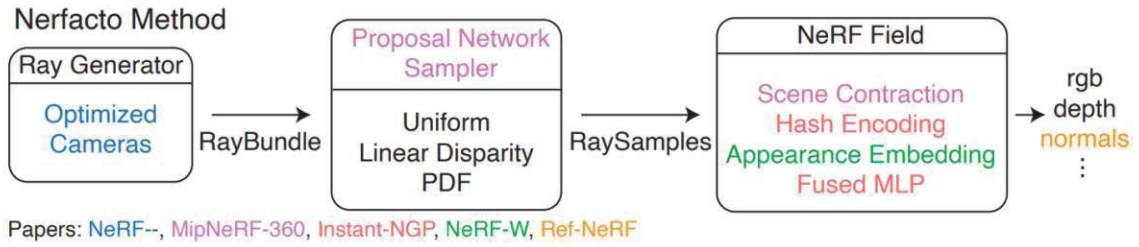


Abbildung 32: Nerfacto-Methode [51]

Der Prozess beginnt mit einem Ray Generator, der Strahlen aus optimierten Kamerablicken erzeugt, was für das Abtasten der Szene entscheidend ist. Ein Piece-wise Sampler wird verwendet, um die Strahlen gleichmäßig bis zu einer bestimmten Entfernung von der Kamera zu sampeln, wodurch sowohl nahe als auch entfernte Objekte effizient abgetastet werden können. Der Proposal Network Sampler verbessert den Abtastprozess, indem er sich auf Regionen konzentriert, die maßgeblich zum finalen Render beitragen, typischerweise die erste Oberflächenintersektion. Dieser Schritt nutzt mehrere Density Fields, um die Anzahl der Samples iterativ zu reduzieren. Ein kleines, verschmolzenes MLP mit Hash Encoding wird für die Dichte Funktion der Szene verwendet, was eine Balance zwischen Genauigkeit und Recheneffizienz bietet. Die NeRF Field-Komponente integriert dabei eine Scene Contraction und Appearance Embeddings. Die Scene Contraction komprimiert den unendlichen Raum in eine begrenzte Box mittels L_∞-Norm. In der Nerfacto-Methode wird diese Norm verwendet, um die unendliche Ausdehnung realer Szenen in einen festen Bereich zu komprimieren. Dies erleichtert die Verarbeitung und Darstellung dieser Szenen in der 3D-Rekonstruktion.
Nerfstudio unterstützt den Export von Punktwolken, TSDF to mesh (Truncated signed distance function) und Poisson Surface reconstruction. Für die weiterführenden Analysen wurde der Export in eine Punktwolke gewählt. Die Ergebnisse und Visualisierungen finden sich in Kapitel 5.

4.1.4 CloudCompare

CloudCompare [52] ist eine kostenfreie Software zur Visualisierung und Verarbeitung von 3D-Punktwolken und triangular meshes. Ursprünglich wurde CloudCompare entwickelt um Vergleiche zwischen zwei Punktwolken, oder zwischen einer Punktwolke und einem triangular Mesh, durchzuführen. Aufgrund der Datenmenge von 3D-Datensätze, wird eine sog. Octree-Struktur eingesetzt um Aufgaben effizient zu lösen. Im Laufe der Jahre entwickelte sich CloudCompare zu einer allgemeinen Software mit Plugin Funktionalitäten, die verschiedene Einsatzgebiete abdeckt. Dazu gehören unter anderem die Registrierung, Subsampling, Normalenbestimmung und Segmentierung von Punktwolkendatensätzen.
Zum Zeitpunkt des Schreibens der Arbeit besteht das Problem, dass NeRF prozessierte Punkt-

wolken den Bezug zum Ursprungskoordinatensystem nicht besitzen. Somit wurde CloudCompare eingesetzt um die NeRF-Punktwolke auf die TLS-Punktwolke grob zu registrieren. Der grobe Registrierungsschritt ist notwendig für die später durchgeführte feine Registrierung innerhalb OPALS [53]. Nach der groben Registrierung der NeRF-Punktwolke wurde ein Subsampling mit einer Distanz von 10cm durchgeführt. Dies soll einerseits dazu beitragen, dass nachfolgende Prozessierungen in einem akzeptablen Zeitrahmen durchgeführt werden können. Andererseits ist eine Homogenisierung notwendig, um anschließende KNN-Abfragen effizient zu gestalten und die Vergleichbarkeit sowie die Genauigkeit der statistischen Auswertungen zu verbessern.

4.1.5 OPALS - Orientation and Processing of Airborne Laser Scanning data

OPALS [53] steht für Orientation and Processing of Airborne Laser Scanning data. Das Ziel von OPALS ist, eine vollständige Verarbeitungskette für die Bearbeitung von Airborne Laser Scanning Daten bereitzustellen. Die Funktionen umfassen z.B. Qualitätskontrollen, Georeferenzierung von Punktwolkendatensätzen, Strukturlinienextraktion, Punktwolkenklassifizierung und die Generierung von DTM's (Digital Terrain Model). OPALS findet Anwendung in verschiedenen Bereichen wie z.B. in der Forstwirtschaft, Hydrographie und Stadtmodellierung.

OPALS-ICP Das Modul OPALS-ICP [54] wurde eingesetzt um die grob ausgerichtete NeRF-Punktwolke fein auf die TLS-Punktwolke zu registrieren. OPALS-ICP verfolgt einen "surface-matching" Anatz wobei ICP (Iterative Closest Point) folgendermaßen aufgeschlüsselt werden kann. Zugehörige Punkte werden iterativ (I) bestimmt, wobei der nächstgelegene Punkt als Korrespondenz (C) betrachtet wird. Die Korrespondenzen werden punktbasiert (P) durchgeführt. Abbildung 33 zeigt die sieben wesentlichen Schritte der Registrierung.

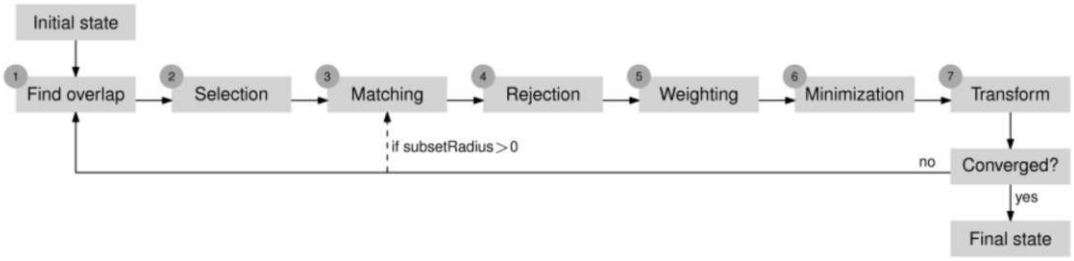


Abbildung 33: Ablauf OPALS-ICP [54]

Wenn das Konvergenzkriterium erfüllt ist, bricht die Iteration ab und das globale Minimum wurde bestimmt. Weiterführende Informationen finden sich in der Dokumentation von OPALS-ICP [55].

OPALS-Cell/Gridfeature OPALS-Cell bzw. Gridfeature [56][57] sind Module um aus Punktwolkendatensätzen Rasterdaten zu generieren (z.B. min,max,mean,normals). Durch Wahl einer Pixelgröße werden statistische Entscheidungen über den Wert der Zelle getroffen. Am Beispiel

der Max-Methode wird für jede Zelle der maximale z-Wert gewählt. Für die Visualisierung der resultierenden Punktwolken wurde das Feature Quantil eingesetzt. Der Vorteil liegt in der Handhabbarkeit von Ausreißer innerhalb einer Zelle, da isolierte Punkte die Abbildung verfälschen.

OPALS-Normals OPALS-Normals wurde für die Bestimmung der Normalenvektoren verwendet. Das Modul unterstützt vier verschiedene Algorithmen[18] und erlaubt Meta-Informationen zu speichern [19]. Weitere Informationen zu den Metavariablen finden sich in der Dokumentation von OPALS [58].
Normalenvektoren sind Grundlage für nachgehende Prozessierungen. Beispielhaft sei hier die Bestimmung von lokalen Neigungs- und Rauigkeitswerten erwähnt. Im Falle dieser Arbeit wurden folgende Parameter zur Normalenbestimmung eingesetzt:

Parameter	**Wert**
neighbours	16
searchRadius	0.25
selMode	quadrant
storeMetaInfo	medium
normalsAlg	robustPlane

Tabelle 9: Parameter OPALS-Normals

OPALS-Segmentation OPALS-Segmentation [58] wurde eingesetzt um Ebenen der Punktwolke zu extrahieren. Voraussetzung für die Ebenenbestimmung ist die Prozessierung der Normalenvektoren. Das Modul unterstützt zwei verschiedene Methoden zur Segmentierung von Punktwolken. Die Generische bedingte Clusterbildung und die Ebenenextraktion. Für die Arbeit wurde die Ebenenextraktion eingesetzt. Die Ebenenextraktion zielt darauf ab, ebene Flächen in der Punktwolke zu identifizieren. Sie nutzt einen Seeded-Region-Growing-Ansatz, bei dem Punkte, die zu einer gemeinsamen Ebene gehören, gruppiert werden. Es müssen zwei Bedingungen erfüllt werden, damit Punkte der Ebene hinzugefügt werden. Einerseits darf der Normalabstand eines Punktes zur definierten Ebene nicht den maximalen Wert überschreiten. Andererseits muss der Winkel zwischen dem Normalenvektor des Punktes und dem der Ebene innerhalb von $max(2 * \arctan(NormalSigma0, 10)$ liegen. Die Ebenenparameter werden dann aktualisiert, wenn das Kriterium $maxSigma$ erfüllt ist. Die Ebene wird sozusagen überwacht, sodass nur passende Punkte dem Segment hinzugefügt werden.
Für die Kapelledatensätze wurden folgende Parameter gewählt:
Der Grund für die Wahl der Parameter liegt in der Geometrie der Kapelle. Da die Kapelle primär einer zylindrischen Gestalt gleicht, wurde der maximale Suchradius auf einen halben Meter eingeschränkt. Somit wird verhindert, dass Punkte die bereits der Krümmung der Kapelle unterliegen, in die Ebene aufgenommen werden. Damit genügend Ebenen für weitere Vergleichs-

[18] simplePlane, robustPlane, fmcd, Count
[19] NormalX, NormalY, NormalZ, NormalSigma0, NormalEigenvalue1-3

Parameter	Wert
maxDist	0.2
maxSigma	0.2
minSegSize	200
searchRadius	0.5
searchMode	3D

Tabelle 10: Ebenenextrahierung Parameter

methoden zur Verfügung stehen, wurde die minimale Anzahl an Punkte pro Segment auf 200 festgelegt.
Das Ergebnis der Segmentierung sind die Ebenenparameter COG_X,COG_Y,COG_Z, A, B, C, D, $Sigma0$, $NumPoints$, $SegmentID$. Für den Export der Ebenenparameter wurde der in OPALS integrierte Segment Manager benutzt. Dieser unterstützt den Export der prozessierten Ebenenparameter innerhalb eines Python oder C++ Skripts. Das zugehörige Skript findet sich im Anhang (Kapitel 8.1.1).

4.2 Validierung der Punktwolken

4.2.1 KD-Tree Abfragen

Um effiziente Punktwolkenabfragen durchzuführen, wurde ein KD-Tree eingesetzt. Der folgende Codeausschnitt zeigt den Einsatz des KD-Trees im Falle der Berechnung der Standardabweichung. Es wird eine Baumstruktur für die Punktwolke 1 erstellt und anschließend der nächste Nachbar zu jedem Punkt der Punktwolke 2 bestimmt. Mithilfe des Parameters k kann die Anzahl der nächsten Nachbarn angepasst werden. Im Falle dieser Arbeit wurde für jeden Punkt $k = 1$ gewählt, also der Punkt mit der minimalen euklidischen Distanz zur Referenz. Für alle weiteren Metriken wurde der gleiche Ansatz gewählt. Im folgenden Kapitel werden die zugrundeliegende Theorie und die Formelapparate erläutert. Für lokale Nachbarschaftsanalysen wurde ein anderer Ansatz verfolgt. Am Beispiel der Oberflächenrauigkeit wird ein Suchradius definiert, der alle Punkte innerhalb des Radius in die Berechnung einbezieht. Das Gleiche gilt für die Bestimmung von Normalenvektoren.

```
1 import numpy as np
2 from scipy.spatial import cKDTree
3
4 def compute_std_dev(point_cloud1, point_cloud2):
5     distances, _ = cKDTree(point_cloud2).query(point_cloud1, k=1)
6     return np.std(distances), distances
```

4.2.2 Metriken

Root Mean Square Error (RMSE) Der Root Mean Square Error (RMSE) ist eine häufig verwendete Metrik zur Bewertung der Differenz zwischen zwei Punktwolken. Er wird wie folgt

berechnet:

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(d_i)^2} \tag{1}$$

Hierbei ist d_i die Differenz zwischen den entsprechenden Punkten der beiden Punktwolken. Der RMSE ist geeignet, um die durchschnittliche Größe der Fehler zu quantifizieren. Ein niedriger RMSE zeigt an, dass die Punktwolken ähnlich bzw. nahe beisammen sind.

Mean Absolute Error (MAE) Der Mean Absolute Error (MAE) misst die durchschnittliche Größe der Fehler unabhängig von deren Richtung. Er wird wie folgt berechnet:

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|d_i| \tag{2}$$

Die MAE ist robuster gegenüber Ausreißern als der RMSE und gibt einen klaren Überblick über die durchschnittliche Abweichung. Abgesehen davon ist MAE leichter zu interpretieren, da die Ergebnisse in der gleichen Einheit wie die ursprünglichen Daten sind [59].

Standardabweichung (Std Dev) Die Standardabweichung misst die Streuung der Fehler um den Mittelwert:

$$\text{Std Dev} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(d_i - \mu)^2} \tag{3}$$

Hierbei ist μ der Mittelwert der Fehler. Diese Metrik ist nützlich, um die Variabilität der Fehler zu bewerten. Eine hohe Standardabweichung weist auf eine große Variabilität der Abstände hin.

Hausdorff-Distanz Die Hausdorff-Distanz misst die größte Abweichung zwischen zwei Punktmengen:

$$d_H(A,B) = \max\left\{\sup_{a\in A}\inf_{b\in B}\|a-b\|, \sup_{b\in B}\inf_{a\in A}\|b-a\|\right\} \tag{4}$$

Diese Metrik ist nützlich, um die größte minimale Abweichung zwischen zwei Punktwolken zu bewerten. Nachteilhaft ist hierbei, dass die Hausdorff-Distanz empfindlich gegen Ausreißer ist. Aus diesem Grund wurde die modifizierte Hausdorff-Distanz eingesetzt, welche den Durchschnitt der minimalen Abstände betrachtet.

$$d_{MH}(A,B) = \frac{1}{|A|}\sum_{a\in A}\inf_{b\in B} d(a,b)$$

Diese Anpassung ist empfindlicher gegenüber dem Gesamterscheinungsbild, statt sich auf den maximalen Wert zu konzentrieren.

Mittlerer Abstand Der mittlere Abstand ist der Durchschnitt der minimalen Abstände zwischen den Punkten der beiden Punktwolken:

$$\text{Mean Distance} = \frac{1}{n}\sum_{i=1}^{n} d_i \tag{5}$$

Diese Metrik bietet eine einfache Möglichkeit, die durchschnittliche Abweichung zwischen den Punktwolken zu quantifizieren. Sie ist nützlich für allgemeine Vergleiche.

Prozentile der Abstände Prozentile bieten Einblicke in die Verteilung der Abstände. Zum Beispiel kann das 95. Perzentil den maximalen Fehler der besten 95% der Punkte angeben:

$$P_{95} = \inf\{d \in R : F(d) \geq 0.95\} \tag{6}$$

wobei $F(d)$ die Verteilungsfunktion der Abstände ist. Diese Metrik hilft, die Verteilung der Fehler besser zu verstehen.

Oberflächenrauigkeit Die Oberflächenrauigkeit quantifiziert die Varianz der Abstände der benachbarten Punkte um einen Referenzpunkt:

$$\text{Rauigkeit} = \text{Var}(\|\mathbf{p}_i - \mathbf{p}_j\|) \tag{7}$$

Diese Metrik ist wichtig für die Bewertung der lokalen Oberflächenstruktur einer Punktwolke.

Normalenvektoren Die Normalenvektoren einer Punktwolke werden durch die Hauptkomponentenanalyse (PCA) der Nachbarpunkte berechnet. Diese Vektoren sind wichtig für die Bewertung der Oberflächenausrichtung und -struktur.

Für jeden Punkt $\mathbf{p}_i$ in der Punktwolke wird eine Menge von Nachbarpunkten $\mathcal{N}_i$ bestimmt. Dies kann entweder durch eine Radiusabfrage oder eine k-nearest-neighbor-Methode geschehen:

$$\mathcal{N}_i = \{\mathbf{p}_j \mid \|\mathbf{p}_j - \mathbf{p}_i\| < \epsilon\}$$

oder

$$\mathcal{N}_i = \{\mathbf{p}_j \mid j \in \text{knn}(\mathbf{p}_i, k)\}$$

Nachdem die Nachbarn bestimmt wurden, wird die Kovarianzmatrix $\mathbf{C}$ für die Punktmenge $\mathcal{N}_i$ bestimmt:

$$\mathbf{C} = \frac{1}{|\mathcal{N}_i|}\sum_{\mathbf{p}_j \in \mathcal{N}_i} (\mathbf{p}_j - \bar{\mathbf{p}}_i)(\mathbf{p}_j - \bar{\mathbf{p}}_i)^T$$

wobei $\bar{\mathbf{p}}_i$ der Schwerpunkt der Nachbarpunkte ist, definiert als:

$$\bar{\mathbf{p}}_i = \frac{1}{|\mathcal{N}_i|} \sum_{\mathbf{p}_j \in \mathcal{N}_i} \mathbf{p}_j$$

Die Kovarianzmatrix $\mathbf{C}$ beschreibt die Verteilung der Punkte in der lokalen Nachbarschaft um $\mathbf{p}_i$.
Um die Hauptkomponenten der Punktverteilung zu bestimmen, wird eine Eigenwertzerlegung der Kovarianzmatrix durchgeführt:

$$\mathbf{C}\mathbf{v} = \lambda \mathbf{v}$$

Hierbei ist λ ein Eigenwert und $\mathbf{v}$ der zugehörige Eigenvektor.
Der Normalenvektor $\mathbf{n}_i$ des Punktes $\mathbf{p}_i$ entspricht dem Eigenvektor $\mathbf{v}_{\min}$, der dem kleinsten Eigenwert $\lambda_{\min}$ entspricht. Dieser Vektor repräsentiert die Richtung minimaler Varianz und zeigt die Normalenrichtung der lokalen Oberfläche an:

$$\mathbf{n}_i = \mathbf{v}_{\min}$$

Krümmung Die Krümmung einer Punktwolke wird durch die Analyse der Eigenwerte der Kovarianzmatrix der Nachbarpunkte berechnet:

$$\text{Krümmung} = \frac{\lambda_{\min}}{\sum \lambda_i} \tag{8}$$

Hierbei sind $\lambda_{\min}$ und λ_i die Eigenwerte der Kovarianzmatrix. Diese Metrik hilft bei der Bewertung der lokalen Geometrie.

Chamfer-Distanz Die Chamfer-Distanz zwischen zwei Punktwolken wird wie folgt berechnet:

$$d_C(A, B) = \sum_{a \in A} \min_{b \in B} \|a - b\| + \sum_{b \in B} \min_{a \in A} \|b - a\| \tag{9}$$

Diese Metrik misst die Summe der minimalen Abstände in beide Richtungen und ist nützlich für den Vergleich der Punktwolkenähnlichkeit.

Dichteunterschied Der Dichteunterschied zwischen zwei Punktwolken wird durch Vergleich der Punktdichten innerhalb eines bestimmten Radius berechnet:

$$\text{Density Difference} = |\text{mean}(\rho_1) - \text{mean}(\rho_2)| \tag{10}$$

wobei ρ_1 und ρ_2 die Punktdichten in den Punktwolken 1 und 2 sind. Diese Metrik ist nützlich, um Unterschiede in der Punktdichte zu quantifizieren.

Dichteüberlappung Die Dichteüberlappung zwischen zwei Punktwolken wird durch die Anzahl der gemeinsamen Punkte innerhalb eines bestimmten Radius berechnet:

$$\text{Density Overlap} = \frac{|\rho_1 \cap \rho_2|}{\min(|\rho_1|, |\rho_2|)} \times 100 \tag{11}$$

Diese Metrik gibt den Prozentsatz der überlappenden Punktdichten an und ist nützlich, um die Ähnlichkeit der Punktverteilungen zu bewerten.

Nearest Neighbor Distance Ratio (NNDR) NNDR wird wie folgt berechnet:

$$\text{NNDR} = \frac{1}{n} \sum_{i=1}^{n} \frac{d_{i1}}{d_{i2}} \tag{12}$$

wobei d_{i1} und d_{i2} die Abstände des nächsten und zweitnächsten Nachbarn für den i-ten Punkt sind. Diese Metrik ist nützlich, um das Verhältnis zwischen den nächstgelegenen Nachbarn zu bewerten. NNDR wird häufig in der Feature Matching Phase von Algorithmen wie SIFT verwendet, um falsche Übereinstimmungen herauszufiltern. Hier wird für jedes Merkmal die Distanz zum nächsten und zweitnächsten Merkmal verglichen, um sicherzustellen, dass die beste Übereinstimmung signifikant besser ist als die zweitbeste.

Median Absolute Deviation (MAD) MAD wird wie folgt berechnet:

$$\text{MAD} = \text{median}(|d_i - \text{median}(d)|) \tag{13}$$

Diese Metrik ist robust gegenüber Ausreißern und gibt die mittlere Abweichung um den Median an.

4.2.3 Ebenen Matching - Version Toleranzen

Eine weitere Möglichkeit, die Identität der Punktwolken zu überprüfen, ist der Vergleich von Ebenenparametern. Das erstellte Skript hat das Ziel, die Normalenvektoren von Ebenen zu berechnen, zu vergleichen sowie Übereinstimmungen zwischen Ebenen basierend auf Distanz- und Winkeltoleranzen zu identifizieren und eine statistische Analyse der Übereinstimmungen durchzuführen.
Zunächst wird die Textdatei `all_planes_parameters.csv` eingelesen, welche durch den Segmentierungsschritt gewonnen wurde. In der Datei befinden sich alle notwendigen Parameter einer Ebene. Dabei werden speziell die Daten der Referenzdatei `tls_kapelle.odm` isoliert, um als Basis für den Vergleich zu dienen. Diese Daten umfassen die Ebenenparameter und die zugehörigen Center-of-Gravity-Koordinaten (COG).
Um eine effiziente Suche nach Nachbarebenen zu ermöglichen, wird ein KD-Tree basierend auf den COG-Koordinaten der Referenzebenen aufgebaut. Dieser Baum erleichtert die schnelle Iden-

tifizierung von Ebenen, die nahe beieinander liegen.
Anschließend werden die Normalenvektoren der Referenzebenen berechnet. Dies erfolgt durch Normierung der Ebenenparameter (A, B, C), wodurch die Richtungsvektoren der Ebenen erhalten werden. Diese Normalenvektoren sind essenziell für den Vergleich der Ebenenorientierungen. Um den Winkel zwischen zwei Vektoren zu berechnen, wird eine Funktion `angle_between_vectors` implementiert. Diese Funktion ermittelt den Winkel zwischen zwei Normalenvektoren, was für die Bewertung der Orientierung von Ebenen von Bedeutung ist.

$$\cos(\theta) = \frac{\vec{u} \cdot \vec{v}}{\|\vec{u}\|\|\vec{v}\|} \tag{14}$$

$$\theta = \arccos(\cos(\theta)) \tag{15}$$

Hierbei ist:

- $\vec{u} \cdot \vec{v}$ das Skalarprodukt der Vektoren $\vec{u}$ und $\vec{v}$,
- $\|\vec{u}\|$ und $\|\vec{v}\|$ sind die Normen der Vektoren $\vec{u}$ und $\vec{v}$,
- θ ist der Winkel zwischen den Vektoren.

Die Funktion `find_matches` sucht nach Übereinstimmungen zwischen den Ebenen der Referenzdatei und den Ebenen der anderen Punktwolken. Dabei werden eine Distanz- und eine Winkeltoleranz verwendet, um Ebenen als übereinstimmend zu identifizieren. Zusätzlich wird eine gewichtete Summe aus Distanz, Winkelunterschied und Sigma0-Unterschied berechnet, um die Qualität der Übereinstimmung zu bewerten.

$$\text{Gewicht} = 0.4 \times \text{norm\_distance} + 0.4 \times \text{norm\_angle\_diff} + 0.2 \times \text{norm\_sigma0\_diff} \tag{16}$$

Hierbei sind:

- $\text{norm\_distance} = \frac{\text{distance}}{\text{max\_distance}}$,
- $\text{norm\_angle\_diff} = \frac{\text{angle\_diff}}{180}$ (Maximaler Winkelunterschied ist 180 Grad),
- $\text{norm\_sigma0\_diff} = \frac{\text{sigma0\_diff}}{\text{max\_sigma0\_diff}}$.

Die Ergebnisse der Übereinstimmungen werden in einer Datei `matched_planes_data` gespeichert, während die nicht übereinstimmenden Ebenen in einer Datei `unmatched_planes_data` gesichert werden.

4.2.4 Ebenen Matching - Version KNN

Neben dem zuvor beschriebenen Ansatz wurde ein alternativer Ansatz verfolgt, bei dem als Ausgangspunkt der Suche die COG-Koordinate (Center of Gravity) des TLS-Datensatzes verwendet wurde. Dieser Ansatz eliminiert sofort die Abhängigkeit von Winkeltoleranzen und Distanztoleranzen, da keine direkten Vergleichswerte mehr erforderlich sind. Stattdessen wird die

COG-Koordinate des TLS-Datensatzes als Referenz genutzt, um in einem Radius von 50 cm im Vergleichsdatensatz nach benachbarten Punkten zu suchen.
Alle Punkte, die innerhalb dieses Radius liegen, werden anschließend ermittelt und der neue Schwerpunkt dieser Punktmenge berechnet. Es ist dabei zu beobachten, dass der neue Schwerpunkt leicht von der ursprünglichen COG-Koordinate des TLS-Datensatzes abweicht. Um innerhalb dieser Nachbarschaft die bestmögliche Ebene zu bestimmen, wurde die Methode der Singular Value Decomposition (SVD) angewendet.
Die Singulärwertzerlegung (SVD) ist eine nützliche Methode, um aus einer Menge von 3D-Punkten eine Ebene zu extrahieren. Betrachtet man eine Nachbarschaft von Punkten in einer Punktwolke, die durch ihre Koordinaten $p_i = (x_i, y_i, z_i)$ beschrieben sind. Diese Punkte werden in einer Matrix P mit der Größe $n \times 3$ zusammengefasst:

$$P = \begin{pmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ \vdots & \vdots & \vdots \\ x_n & y_n & z_n \end{pmatrix}$$

Um die Analyse zu vereinfachen, berechnet man den Schwerpunkt (den Mittelwert der Punkte):

$$\bar{p} = \frac{1}{n}\sum_{i=1}^{n} p_i = \left(\frac{1}{n}\sum_{i=1}^{n} x_i, \frac{1}{n}\sum_{i=1}^{n} y_i, \frac{1}{n}\sum_{i=1}^{n} z_i \right)$$

Anschließend subtrahiert man diesen Mittelwert von allen Punkten, um die Punktwolke zu zentrieren:

$$P' = P - \bar{p}$$

Nun wendet man die Singulärwertzerlegung (SVD) auf die zentrierte Matrix P' an:

$$P' = U\Sigma V^T$$

Die Matrix V enthält die Richtungen der Hauptachsen der Punktverteilung. Der Normalenvektor der gesuchten Ebene ist der Vektor, der der kleinsten Singulärwertkomponente in V^T entspricht. Dieser Vektor beschreibt die Richtung, in der die Punktverteilung am wenigsten variiert, also die senkrechte Richtung zur Ebene.
Die Gleichung der Ebene kann dann unter Verwendung des Normalenvektors und des Schwerpunkts wie folgt formuliert werden:

$$\mathbf{n} \cdot (\mathbf{p} - \bar{p}) = 0$$

wobei $\mathbf{n}$ der Normalenvektor der Ebene ist.

4.2.5 M3C2 - Multiscale Model to Model Cloud Comparison

Im Vergleich zu einem Punkt-zu-Punkt-Vergleich (Cloud2Cloud) berücksichtigt der M3C2-Algorithmus sowohl Unsicherheiten in der Registrierung als auch in der Messmethodik. Dies ermöglicht es dem M3C2-Algorithmus, zwischen tatsächlichen Veränderungen und Messrauschen zu unterscheiden, indem er signifikante Änderungen identifiziert, die über einem bestimmten Schwellenwert, dem sogenannten Level of Detection (LOD), liegen. Dieser LOD wird basierend auf den Unsicherheiten der Punktwolkenregistrierung und der Messungen berechnet. Das bedeutet, dass nur Veränderungen, die größer als der LOD sind, als signifikant betrachtet werden, während kleinere Änderungen als statistisches Rauschen interpretiert werden. Zudem beeinträchtigt das Fehlen von Punkten (z. B. bei der TLS-Kapelle) nicht die Distanzen des M3C2-Algorithmus. Allgemein kann gesagt werden, dass der M3C2-Algorithmus robuster gegenüber unterschiedlichen Punktdichten und komplexen Geometrien ist als die C2C- oder C2M-Ansätze (Cloud-to-Mesh). Die Analyse basiert auf der Gegenüberstellung der TLS-Punktwolke und der photogrammetrisch bestimmten Datensätze. Die Umsetzung erfolgte mit einem Python-Package [60], das von der 3DGeo Research Group Heidelberg entwickelt wurde. Die TLS-Punktwolke dient als Referenzepoche, während die NeRF- und DIM-Punktwolken als Vergleichsepochen herangezogen werden. In der ersten Phase der M3C2-Analyse wird der Registrierungsfehler zwischen der TLS-Punktwolke und den Vergleichsepochen geschätzt. Dies erfolgt durch eine erste M3C2-Distanzberechnung, bei der keine Registrierungskorrektur angewendet wird. Durch diesen Schritt kann das Rauschen der Punktwolke bestimmt werden, das als Unsicherheit der Registrierung zu betrachten ist. Mit diesem geschätzten Parameter werden in der zweiten Phase der Analyse erneut die Distanzen zwischen den Punktwolken berechnet, wobei nun der zuvor bestimmte Wert in die Berechnungen einfließt. Für jede Analyse werden statistische Parameter wie der mittlere Abstand, der mediane Abstand, die Standardabweichung, der minimale und maximale Abstand sowie der Prozentsatz der signifikanten Änderungen ermittelt. Letzterer wird durch den Vergleich der M3C2-Distanzen mit dem Level of Detection (LOD) bestimmt. Wenn die gemessene Distanz den LOD übersteigt, wird die Änderung als signifikant betrachtet. Im letzten Schritt wird für alle Datensätze ein gemeinsamer Unsicherheitsparameter ermittelt, um eine konsistente Vergleichsbasis für die dritte M3C2-Analyse zu schaffen. Dies ermöglicht den Vergleich der Qualität der Rekonstruktion auf Basis eines Vergleichswertes, wobei die Ergebnisse nicht durch die Unschärfe jeder individuellen Punktwolke verzerrt werden. Die Ergebnisse werden in Kapitel 5.4.6 visualisiert und quantitativ dargestellt.

5 Ergebnisse

Im Folgenden werden die Ergebnisse der Untersuchung übersichtlich dargestellt. Dabei wird sowohl auf eine visuelle als auch auf eine tabellarische Präsentation der Daten Wert gelegt, um einen umfassenden Überblick zu ermöglichen. Eine tiefergehende Interpretation der Ergebnisse erfolgt zu einem späteren Zeitpunkt.

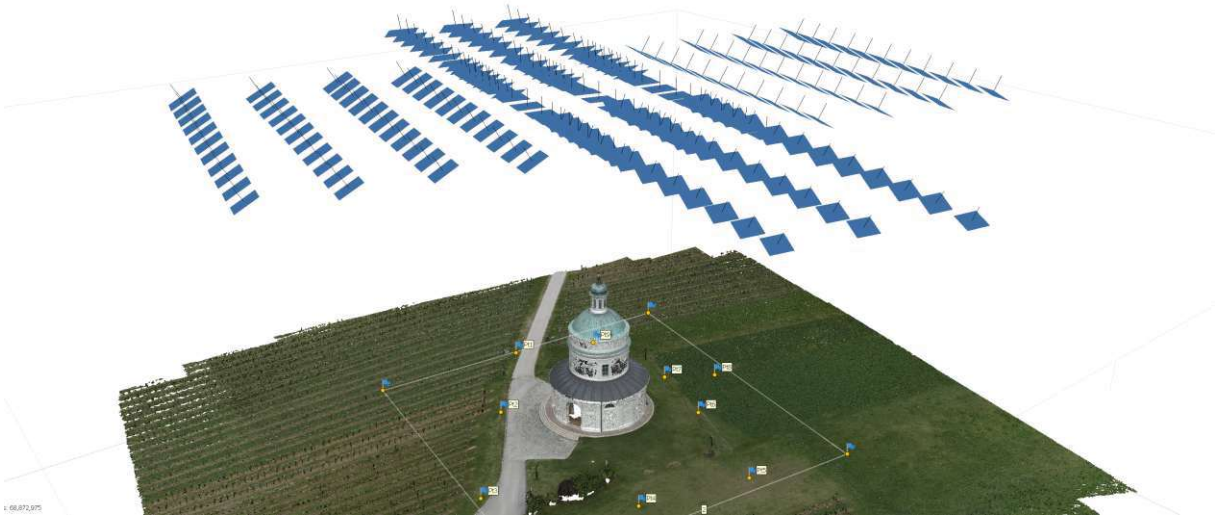


Abbildung 34: Orbit Flug Kapelle

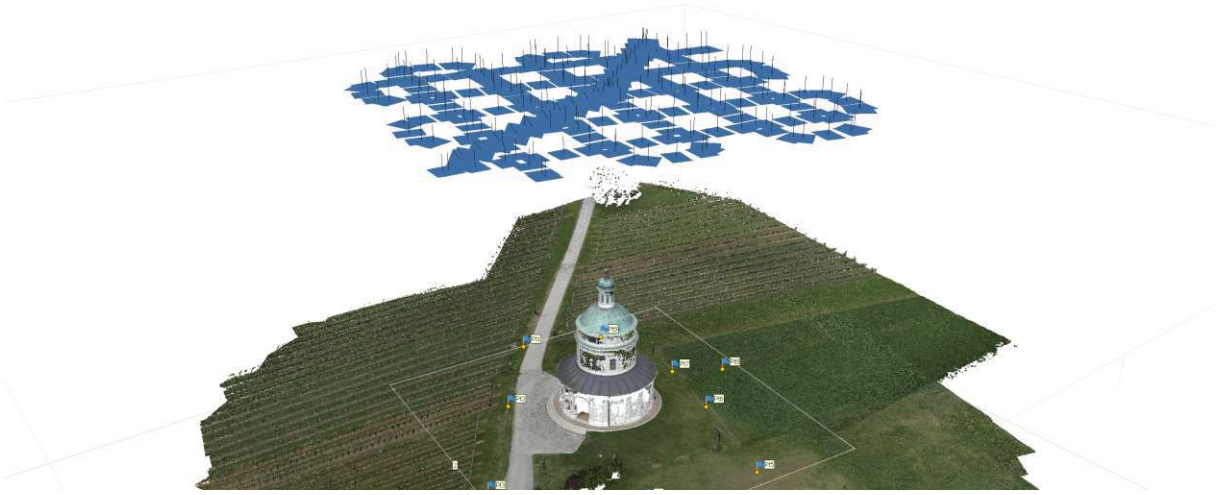


Abbildung 35: Linear Flug Kapelle

5.1 ICP Registrierung

Für den Erfolg einer ICP-Registrierung ist zu erwähnen, dass Eingangspunktwolken kein starkes Rauschen besitzen sollten. Im Falle der NeRF-Punktwolken musste eine manuelle Bereinigung des Bodens[20] durchgeführt werden. Das Rauschen der Bodenpunkte veranlasste den ICP-Algorithmus dazu, kein globales Minimum zu erreichen. Abbildung 36 zeigt einen Schnitt des Kapellen NeRF-Datensatzes. Erkennbar ist, dass Asphaltpunkte präziser erfasst wurden als Vegetation.

[20] speziell Punkte die auf der Wiese erfasst wurden weisen keine hohe Präzision auf

Abbildung 36: Rauschen NeRF-Punktwolke

Die Tabellen 11 und 12 zeigen die Ergebnisse der feinen Registrierung mithilfe OPALS-ICP. Die TLS-Punktwolke wurde fixiert, da diese bereits im UTM Zone 33N gelagert ist. Die beiden NeRF-Punktwolken wurden gesamtheitlich auf die Referenz registriert. Somit ist auch die Registrierung zwischen allen Kombinationen optimal.

Punktwolke	Typ	Fix/Loose	USD	RS	NS	MLS	Datei
1	Punktwolke [1]	fix	1.00	false	false	false	TLS Kapelle
2	Punktwolke [2]	loose	1.00	false	false	false	orbit Kapelle NERF
3	Punktwolke [3]	loose	1.00	false	false	false	linear Kapelle NERF

Tabelle 11: Eingabepunktwolken Konfiguration, USD= Uniform Sampling Distance, RS = Random Subsampling, NS = Normal Subsampling, MLS = Max Leverage Subsampling

Punktwolke	Parameter	Wert	Sigma	Signifikanz
1	omega [°]	0.0000000	-	-
	phi [°]	0.0000000	-	-
	kap [°]	0.0000000	-	-
	tx	0.0000000	-	-
	ty	0.0000000	-	-
	tz	0.0000000	-	-
	scale	1.0000000	-	-
2	omega [°]	0.0002450	0.00621	0.04
	phi [°]	-0.0454822	0.00594	7.66
	kap [°]	0.1172395	0.10155	1.15
	tx	0.0121994	0.00230	5.30
	ty	-0.0059737	0.00229	2.60

Tabelle 12: Transformationsparameter der NeRF-Punktwolken

Tabelle 12: Fortsetzung der Transformation Parameter der Punktwolken

Punktwolke	Parameter	Wert	Sigma	Signifikanz
	tz	-0.0015972	0.00115	1.39
	scale	1.0002961	0.00011	2.79
3	omega [°]	-0.1892822	0.01286	14.72
	phi [°]	0.1190493	0.01404	8.48
	kap [°]	-0.2828047	0.20149	1.40
	tx	0.0362818	0.00366	9.91
	ty	0.0036289	0.00356	1.02
	tz	0.0050863	0.00156	3.27
	scale	1.0038368	0.00019	20.15

Tabelle 12: Transformationsparameter der NeRF-Punktwolken

Tabelle 12 zeigt die Transformationsparameter ω, ϕ, κ, t_x, t_y, t_z, $scale$ nach der fünften Iteration. Dem ICP-Algorithmus wurde erlaubt, eine 7-Parameter-Transformation anzuwenden[21]. Die Skalierung ist im Falle von NeRF-Punktwolken einfacher zu behandeln als die Rücktransformation in das ursprüngliche Koordinatensystem. Da bereits eine Skalierung in Agisoft Metashape angewandt wird, kann der Skalierungsfaktor der eingangs erwähnten XML-Datei entnommen werden. Nach der Prozessierung der NeRF-Punktwolke mit Nerfstudio wird ebenfalls ein Skalierungsfaktor dokumentiert bzw. angewandt. Durch Invertierung der beiden Skalierungen kann die ursprüngliche Metrik des Datensatzes reproduziert werden. Dies spiegelt sich wiederum in den Transformationsparametern wider. Die Skalierung bewegt sich für beide NeRF-Datensätze in der Nähe von 1, und somit war die Reskalierung erfolgreich. Beim Vergleich der beiden Flugroutinen erkennt man, dass die Skalierung jedoch um eine Größenordnung weniger angepasst werden musste. Darüber hinaus ist der Signifikanzwert des Skalierungsfaktors deutlich höher als bei der orbitalen Flugroutine. Die Verschiebungen bzw. Rotationen in x, y, z sind bei beiden Datensätzen ähnlich und weisen keine Auffälligkeiten auf.

Die Signifikanz berechnet sich aus dem Quotienten zwischen dem Wert und der Genauigkeit[22]. Wenn die Signifikanz Werte $>> 3$ annimmt, dann ist von einer hohen Signifikanz zu sprechen, und der Parameter sollte im Modell behalten werden. Ist der Wert nicht signifikant, kann der Parameter aus dem Modell entfernt werden, da er nicht zur Verbesserung des Modells beiträgt. Es sollte jedoch beachtet werden, dass die Reduktion von Parametern Einfluss auf andere Parameter hat. Aufgrund der Korrelation der Parameter kann ein nicht signifikanter Parameter signifikant werden.

Die Skalierung des linearen Kapelle-Datensatzes ist mit einer Signifikanz von 20,15 hoch und

[21] Somit ist auch die Skalierung frei von Zwängen

[22] $Wert/Sigma$

daher ein wichtiger Parameter für das ICP-Modell.

Iteration	Korrespondenzen	Std(dp)	Mean(dp)	Norm(dx)
1	2076	0.03245	0.00018	0.01628
2	1436	0.02695	-0.00083	0.01566
3	1233	0.02483	-0.00061	0.00733
4	1175	0.02488	-0.00064	0.00804
5	1158	0.02447	-0.00034	0.00526

Tabelle 13: Allgemeine Statistiken der Iterationen

Iteration	Punktwolke	Korrespondenzen	Std(dp)	Mean(dp)	Datei
1	[1]	1196	0.02461	-0.00020	TLS Kapelle
	[2]	1591	0.03393	0.00022	orbit Kapelle NERF
	[3]	1365	0.03648	0.00047	linear Kapelle NERF
2	[1]	927	0.02211	-0.00060	TLS Kapelle
	[2]	1224	0.02808	-0.00062	orbit Kapelle NERF
	[3]	721	0.03044	-0.00147	linear Kapelle NERF
3	[1]	864	0.02121	-0.00064	TLS Kapelle
	[2]	1077	0.02599	-0.00035	orbit Kapelle NERF
	[3]	525	0.02777	-0.00111	linear Kapelle NERF
4	[1]	834	0.02131	-0.00044	TLS Kapelle
	[2]	1049	0.02572	-0.00040	orbit Kapelle NERF
	[3]	467	0.02860	-0.00151	linear Kapelle NERF
5	[1]	836	0.02117	-0.00050	TLS Kapelle
	[2]	1032	0.02541	-0.00008	orbit Kapelle NERF
	[3]	448	0.02781	-0.00062	linear Kapelle NERF

Tabelle 14: Punktwolkenabhängige Statistiken pro Iteration

Tabellen 13 und 14 zeigen weitere Ergebnisse der ICP-Registrierung. Der fallende Mittelwert bzw. die fallende Standardabweichung zeigen, dass das Modell konvergiert und die Registrierung Schritt für Schritt optimiert wird. Die Anzahl der Korrespondenzen nimmt mit jeder Iteration ab, da in jedem Iterationsschritt nur die relevantesten Punkte für die Registrierung beibehalten werden, während weniger relevante Punkte ausgeschlossen werden.
Beim Vergleich der Datensätze im 5. Iterationsschritt (siehe Tabelle 14) erkennt man, dass die lineare Flugroutine die höchste Standardabweichung aufweist. Dies legt nahe, dass die Punkt-

wolke aus dem linearen Flug im Vergleich zu der des orbitalen Flugs weniger gut mit dem TLS-Datensatz übereinstimmt.
Die geringere Übereinstimmung der Punktwolke aus dem linearen Flug ist durch die eingeschränkte Aufnahmegeometrie und die weniger optimalen Blickwinkel bedingt. Dies führt wiederum zu einer geringeren Rekonstruktionsqualität für vertikal ausgerichtete Geometrien. An dieser Stelle sei jedoch erwähnt, dass lineare Flugmuster für 3D-Rekonstruktionen allgemein nicht geeignet sind. Vielmehr steht bei Einsatz dieses Flugmusters die Erstellung von Orthophotos oder Dachrekonstruktionen im Vordergrund, wobei vertikal ausgerichtete Ebenen nicht gut abgebildet werden.

5.2 Visualisierung der resultierenden Datensätze

Abbildung 37: Gegenüberstellung der resultierenden Punktwolken: Top, Vorderseite, Rückseite

5.2.1 Rendering Kapelle

Abbildung 39 zeigt eine gerenderte Szene eines NeRFs und stellt die beiden Flugroutinen Orbit und Linear gegenüber. Die erste Zeile entspricht dem linearen Flugmuster, wobei die zweite Zeile das Ergebnis des orbitalen Flug präsentiert. Die gezeigte Darstellung entspricht keinem Vektor- oder Rasterdatensatz, sondern basiert auf einem trainierten neuronalen Netzwerk. Dabei dienen

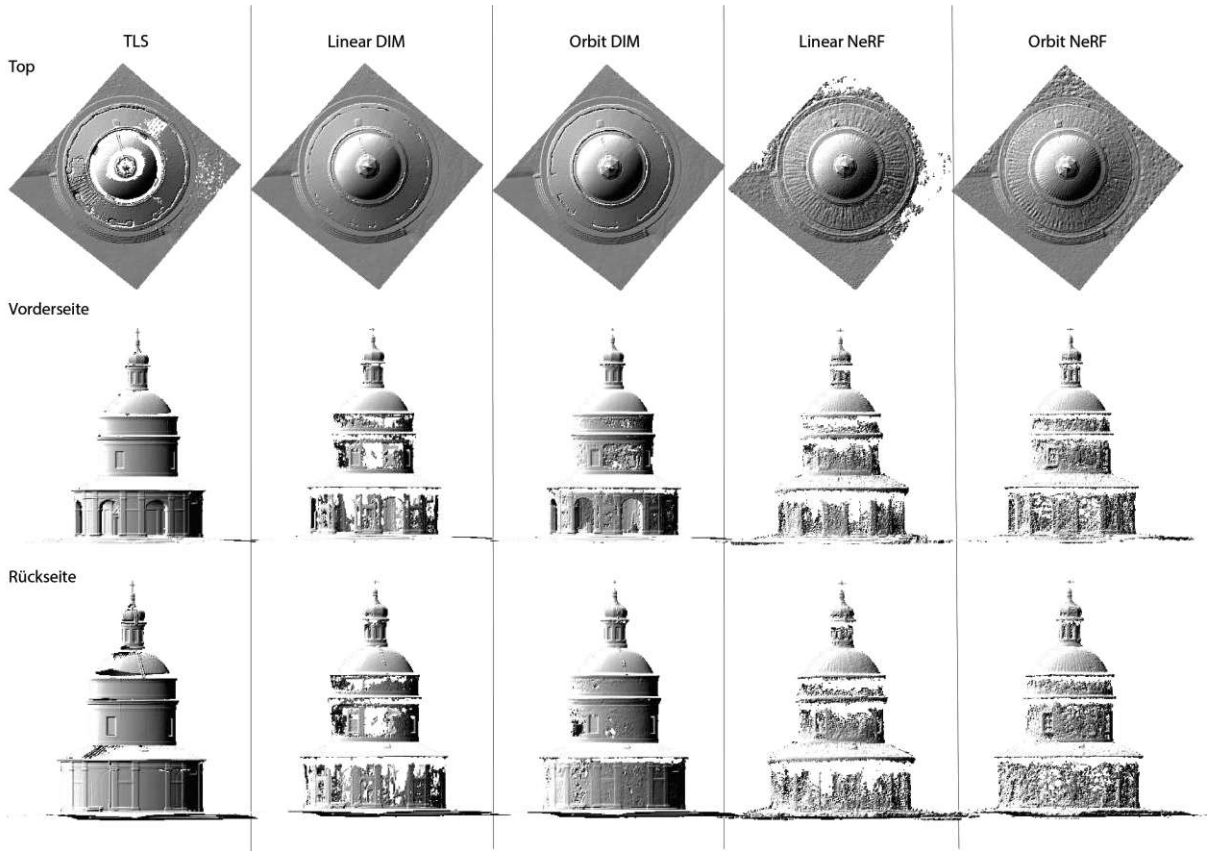


Abbildung 38: Gegenüberstellung der Rasterdatensätze: Shading Top, Vorderseite, Rückseite

Abbildung 39: Gegenüberstellung NeRF Rendering Orbit vs. Linear

der Ort (x, y, z) und die Blickrichtung der Kamera (θ) als Eingaben für das Netzwerk, das die Szene rekonstruiert und rendert.
Es fällt auf, dass perspektivische Änderungen zu Artefakten in der Visualisierung führen. Dieser Effekt ist für NeRF-Ansätze charakteristisch und hängt von der Verteilung der aufgenommenen Bilder ab. Da die meisten der erfassten Bilder primär aus einer „Vogelperspektive“ aufgenommen wurden, ähnelt die Szene in dieser Ansicht einem realistischen Foto. Mit zunehmender Änderung des Blickwinkels in Richtung der Frontalansicht treten verstärkt Artefakte auf. Dies kann darauf zurückgeführt werden, dass dem neuronalen Netz in diesen Bereichen weniger Bilddaten zur Verfügung stehen, um eine Rekonstruktion durchzuführen.
Der Vergleich zwischen den beiden Flugroutinen Orbit und Linear verdeutlicht, dass die vermehrten Schrägaufnahmen in der Orbit-Aufnahme eine bessere Rekonstruktion der Frontalansicht ermöglichen. Im Gegensatz dazu führt die lineare Aufnahme mit ihrer eingeschränkten Perspektivvielfalt zu einer geringeren Rekonstruktionsqualität, insbesondere in Bereichen, die nicht direkt im Sichtfeld der Kamera liegen.
Trotz dieser Einschränkungen zeigt sich, dass die Vollständigkeit des Modells in beiden Fällen gewährleistet ist. Insbesondere die Fassade der Kapelle, trotz ihrer geringen Textur, wird in beiden Aufnahmemodi rekonstruiert. Ebenso bleibt das Kreuz auf der Spitze der Kapelle in beiden Szenen erhalten, was darauf hinweist, dass das Modell auch in der Lage ist, kleinere und detaillierte Strukturen erfolgreich zu erfassen.
Insgesamt zeigt der Vergleich, dass die Wahl des Flugmusters einen direkten Einfluss auf die Rekonstruktionsgenauigkeit hat, wobei die Orbit-Aufnahme durch die höhere Perspektivvielfalt eine robustere Rekonstruktion ermöglicht.

5.2.2 Rendering Baum

Abbildung 40 zeigt die Ergebnisse des Baumdatensatzes, jedoch als Gaussian Splat. Die erste Zeile zeigt die gerenderte Szene des linearen Flugmusters, während die zweite Zeile den Orbitflug darstellt. Wie auch beim Kapellendatensatz (NeRF) ist zu erkennen, dass sich die Repräsentation der Szene mit zunehmender Änderung der Kameraperspektive verschlechtert. Gaussian Splatting weist jedoch im Gegensatz zu der NeRF-Visualisierung partikelförmige Artefakte auf.
Auch hier ist aufgrund mangelnder Bilddaten keine zuverlässige Visualisierung des Baumes möglich. Auffällig ist, dass in der gerenderten Szene Aststrukturen zum Vorschein kommen. Diese sind jedoch in der Punktwolke (dritte Zeile) nicht zu finden. Es ist erkennbar, dass auch hier der orbitale Flugplan bei tieferen Kameraperspektiven die Szene besser visualisiert als der lineare Flugplan.
Bei näherer Betrachtung der Punktwolke fällt auf, dass die angrenzenden Weingärten rekonstruiert wurden, jedoch mehrfach Lücken im Datensatz enthalten sind. Aufgrund der eingeschränkten Diversität der Aufnahmepositionen konnten keine Schnittpunkte der Lichtstrahlen kalkuliert werden. Auffällig ist auch, dass die Punktwolke verrauscht ist. Die Straße ist gewölbt und ähnelt nicht der Realität. Die Weingärten werden abgerundet dargestellt und sind geometrisch wenig zufriedenstellend aufgelöst. Die Abbildung zeigt nur die Orbit-NeRF-Punktwolke,

Abbildung 40: Gegenüberstellung Gaussian Splatting Rendering Orbit vs. Linear Baum

da aus Nerfstudio die lineare Repräsentation nicht exportierbar ist. Gründe hierfür sind aktuell nicht bekannt und dienen als Grundlage für weitere Untersuchungen.
Ein weiterer auffälliger Punkt ist die Repräsentation der Baumkrone. Die Baumkrone des NeRF Datensatzes (s. Abbildung 42) ist im Gegensatz zu DIM (s. Abbildung 41) dichter repräsentiert. Die Gründe hierfür liegen in der Art der Rekonstruktion: Während DIM-Verfahren auf Features basieren, verfolgt NeRF einen gesamtheitlichen Ansatz zur Repräsentation der Szene. Im Falle der DIM-Punktwolke können nur wenige Punkte zur Rekonstruktion bestimmt werden, weshalb die Baumkrone lückenhaft erscheint. Dennoch ist positiv zu bewerten, dass im DIM-Datensatz einzelne Aststrukturen erkennbar sind, die bei NeRF nicht existieren. Rechts neben der Baumkrone ist auch ein Ausschnitt des Weingartens zu erkennen. Hier wirkt die Rekonstruktion zwar nicht vollständig, jedoch präziser und feiner aufgelöst. Abbildung 43 zeigt einen Ausschnitt der Baumkrone in der Punktwolke, die mit der Nerfacto-Methode erstellt wurde. Deutlich erkennbar ist, dass durch das Volumen-Rendering Punkte innerhalb der Baumkrone identifiziert werden konnten. Wie bei der DIM-Methode wurden auch hier keine feinen Strukturen wie Äste rekonstruiert. Auffällig ist jedoch, dass NeRF im Gegensatz zu DIM die Dichte der Baumkrone erfassen konnte, wodurch eine realistischere Darstellung des Volumens ermöglicht wird.

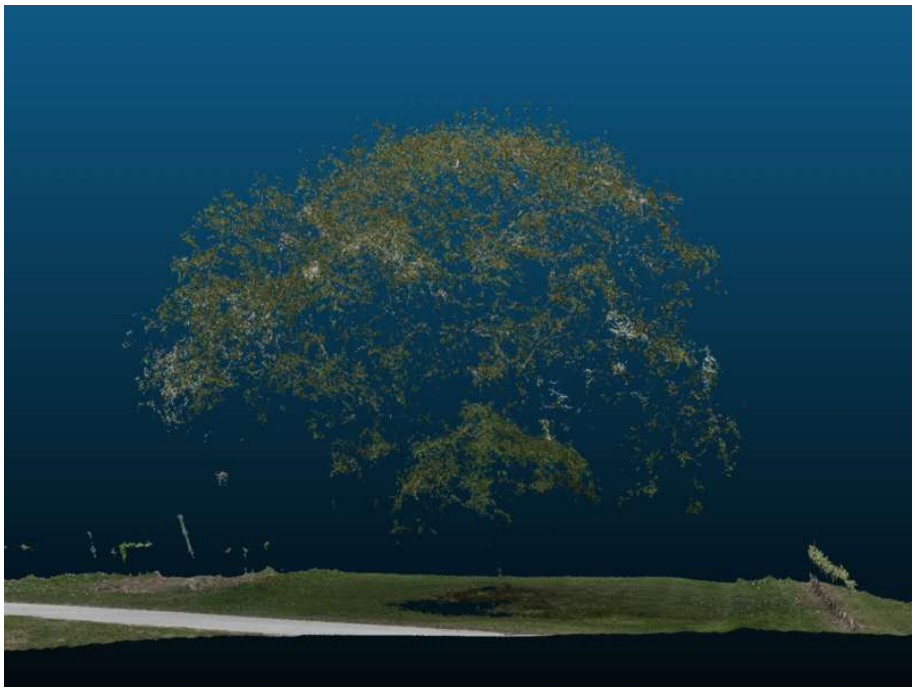


Abbildung 41: Baumkrone DIM: Aufgrund der unzureichenden Feature-Extrahierung konnte keine repräsentative Darstellung der Baumkrone rekonstruiert werden

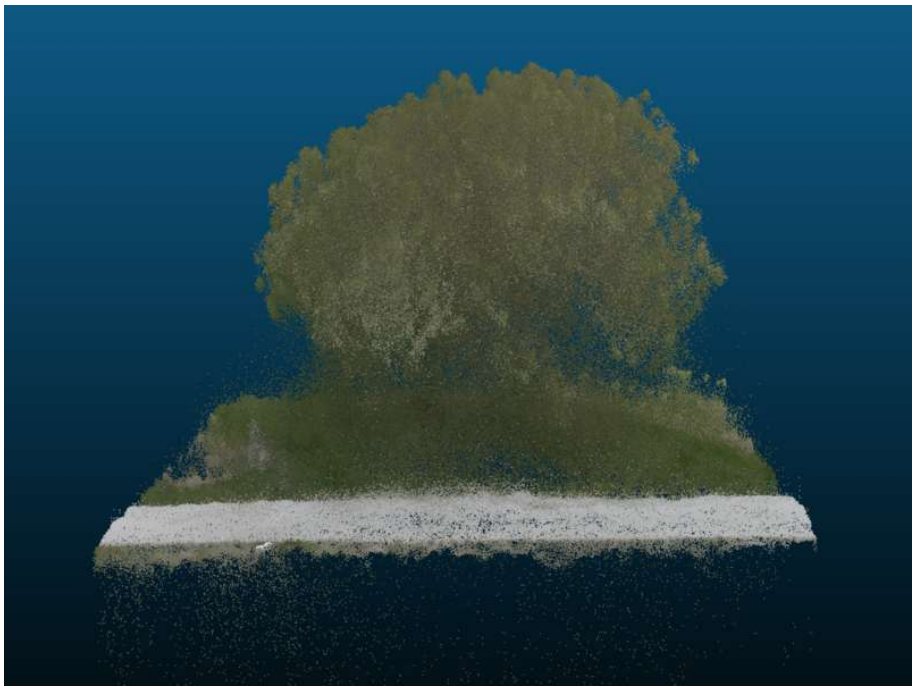


Abbildung 42: Baumkrone NeRF: Ersichtlich ist, dass die Dichte der Baumkrone im Falle NeRF deutlich höher ist als bei DIM.

5.3 Vergleich der Rekonstruktionsqualität

Die Abbildungen 37 und 38 zeigen die rekonstruierten Datensätze und stellen die Ergebnisse visuell gegenüber. Die TLS-Punktwolke dient als Referenz. Auffällig ist, dass im Dachbereich der

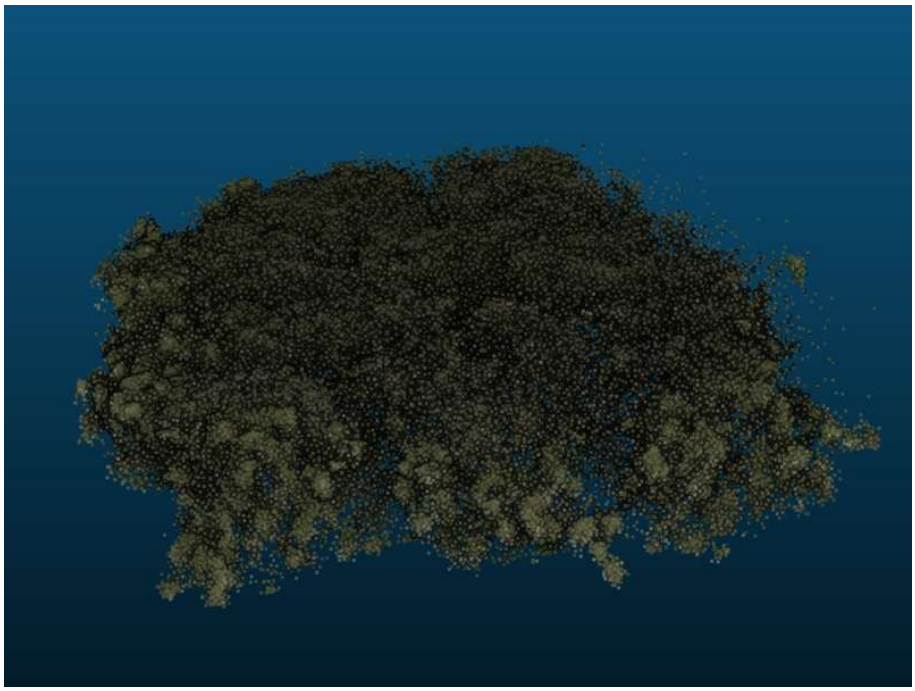


Abbildung 43: Schnitt Baumkrone NeRF: Das Volumen-Rendering von NeRF konnte Punkte innerhalb der Baumkrone identifizieren. Äste sind jedoch nicht sichtbar

Kapelle Punkte fehlen, was auf den Einsatz eines terrestrischen Laserscanners und damit verbundene Scanschatten zurückzuführen ist. Diese fehlenden Bereiche erfordern besondere Vorsicht bei der Auswertung der statistischen Kennzahlen, da sie höhere Abweichungen verursachen und die Analysen verzerren können. Dennoch zeigt die Detailtreue der TLS-Punktwolke, dass die Kapelle präzise erfasst wurde: Kanten erscheinen scharf, und das Kreuz der Kapelle ist vollständig rekonstruiert. Abgesehen von den Scanschatten ist die Punktwolke von hoher Qualität und kann als ideale Referenz betrachtet werden.
Die photogrammetrischen Punktwolken lassen sich in eine lineare und eine orbitale Aufnahme unterscheiden. Bei den DIM-Punktwolken fällt auf, dass die Wände der Kapelle lückenhaft sind. Dies ist zum einen auf die eingeschränkte Aufnahmegeometrie zurückzuführen, die nicht alle Bereiche der Fassade optimal abdeckt, vor allem aber auf die geringe Textur der Fassaden. Texturarme Oberflächen wie glatte oder homogene Wände bieten nur wenige auffällige Bildmerkmale, die von SIFT-Algorithmen erkannt und für die Bildorientierung genutzt werden können. Da Metashape Tiefenkarten nur für Bildpaare berechnet, die eine ausreichende Anzahl an übereinstimmenden Merkmalen (Tie Points) aufweisen, führt das Fehlen solcher Merkmale in texturarmen Bereichen dazu, dass bestimmte Bildpaare ausgeschlossen werden. Dies resultiert letztlich in Lücken und einer reduzierten Punktdichte in der resultierenden dichten Punktwolke. Im Fall der NeRF-Punktwolken ist die Dichte der Daten deutlich geringer als bei DIM und TLS. Die Steuerung der Punktdichte bei NeRF ist aktuell nicht so präzise wie bei DIM-Methoden. Über die Anzahl der zu rekonstruierenden Punkte pro Pixel können bei DIM gleichmäßig aufgelöste Modelle erzeugt werden. Bei NeRFs hingegen besteht noch eine Herausforderung darin,

eine Balance zwischen Auflösung und der Entstehung von Artefakten zu finden.
Trotz dieser Unterschiede zeigt der Vergleich, dass NeRF-Datensätze weniger Lücken aufweisen als DIM-Datensätze. Dies liegt an der unterschiedlichen Art der Szenenrepräsentation: NeRF modelliert die gesamte Szene als kontinuierliche, dichte Volumenrepräsentation, während DIM sich auf lokale Merkmale wie Ecken und Kanten stützt. NeRF lernt, Lichtstrahlen aus verschiedenen Blickwinkeln zu rekonstruieren, was auch die Darstellung texturarmer Bereiche ermöglicht.
Bei einem Vergleich der „Top"-Ansichten der verschiedenen Ansätze zeigen sich Unterschiede hauptsächlich in der Dichte der Punktwolken. Da die Aufnahmen der Kapelle überwiegend aus der Vogelperspektive gemacht wurden, wurden bei beiden photogrammetrischen Punktwolken gute Ergebnisse erzielt.
Speziell in Abbildung 38 (Hillshade) erkennt man, dass Unterschiede in der Rekonstruktion des Daches vorliegen. Während DIM geglättet und planer wirkt, zeigt die NeRF-Rekonstruktion insgesamt mehr Struktur. Dieser Unterschied ist sowohl auf die Punktdichte der Datensätze als auch auf die Unterschiede in der algorithmischen Verarbeitung zurückzuführen.
Der Zwiebelturm der Kapelle wurde bei beiden Ansätzen vollständig rekonstruiert. Die DIM-Punktwolke erscheint visuell weniger verrauscht und zeigt Kanten und Strukturen klarer, während die NeRF-Rekonstruktion aufgrund der volumetrischen Darstellung dichter wirkt, jedoch an feinen Strukturdetails einbüßt. Auffällig ist, dass die Leiter an der Rückseite der Kapelle nur in der DIM-Rekonstruktion sichtbar ist. Dieses Fehlen in der NeRF-Rekonstruktion deutet auf Herausforderungen bei der Erfassung feiner Höhenunterschiede und die geringere Dichte der exportierten Punktwolke hin.
Im unteren Drittel der Kapelle zeigt sich, dass die feineren Bauelemente in der orbitbasierten DIM-Aufnahme klarer erfasst werden konnten, während sie in der NeRF-Rekonstruktion nicht deutlich dargestellt sind. Dieser Unterschied lässt sich sowohl durch die algorithmischen Unterschiede bei der Rekonstruktion als auch durch die Steuerung des Detailgrads beim Export aus Nerfstudio erklären[23].
Insgesamt bietet DIM bei dieser Aufnahmegeometrie eine höhere Präzision und Genauigkeit bei der Rekonstruktion feinerer Strukturen. NeRF hingegen zeigt Vorteile in texturarmen Bereichen, da es unabhängig von klaren Bildmerkmalen arbeitet. Die Wahl des Ansatzes sollte daher auf den spezifischen Anforderungen der Anwendung basieren.

5.4 Quantitativer Vergleich

5.4.1 Metriken

Tabelle 15 zeigt die Ergebnisse der Punkt-zu-Punkt Auswertungen zwischen TLS-DIM und TLS-NeRF. Zusätzlich wurden die Ergebnisse einschließlich der Ausreißerentfernung bestimmt, die im rechten Teil der Tabelle dargestellt sind. Die Ausreißerentfernung wurde mit Open3D durchgeführt, wobei eine Anzahl an Nachbarn sowie eine Standardabweichung festgelegt wurden.

[23]Innerhalb Nerfstudio wurde für den Export eine Punktmenge von 10 Mio. gewählt, wobei Ausreißer durch Open3D entfernt wurden. Bei Erhöhung der Punktmenge wurde keine Verdichtung des Modells erzielt. Der Ausreißeranteil stieg mit Erhöhung der Punktmenge

Die Methode entfernt Punkte, die weiter von ihren Nachbarn entfernt sind als der Durchschnitt in der jeweiligen Nachbarschaft. Laut Tabelle hat die Ausreißerentfernung keinen signifikanten Einfluss auf die Ergebnisse. Stattdessen zeigen die Statistiken einen Anstieg der RMSE- und MAE-Werte. Auch die Streuungswerte sind nach der Entfernung der Ausreißer gestiegen. Der Prozentsatz der Punkte, die innerhalb der Toleranzgrenze liegen, verstärkt diese Ergebnisse. Bei den orbitalen Flugaufnahmen erreicht DIM 69 %, während NERF 35 % der Punkte innerhalb der Toleranz platziert. Im linearen Modus erreicht DIM 56 %, während NERF mit 25% weniger performant ist. Zusammengefasst zeigt die Analyse, dass DIM im orbitalen Modus in fast allen Metriken eine höhere Präzision und Konsistenz bietet.

Metrik	**Inkl. Ausreißer**				**Exkl. Ausreißer**			
	Linear DIM	**Orbit DIM**	**Orbit NERF**	**Linear NERF**	**Linear DIM**	**Orbit DIM**	**Orbit NERF**	**Linear NERF**
RMSE [m]	0.635	0.481	0.437	0.559	0.676	0.496	0.555	0.648
MAE [m]	0.279	0.178	0.215	0.286	0.310	0.185	0.268	0.352
Standardabweichung [m]	0.570	0.447	0.381	0.481	0.601	0.460	0.486	0.543
Modifizierte Hausdorff-Distanz [m]	0.279	0.178	0.215	0.286	0.310	0.185	0.268	0.352
Mittlere Distanz [m]	0.279	0.178	0.215	0.286	0.310	0.185	0.268	0.352
Prozentsatz innerhalb der Toleranz [%]	56.769	69.495	35.449	25.228	55.888	68.854	34.943	24.834
Mittlerer Fehler X-Achse [m]	0.058	0.033	0.024	0.044	0.058	0.033	0.037	0.045
Mittlerer Fehler Y-Achse [m]	0.097	0.046	0.058	0.096	0.097	0.046	0.087	0.110
Mittlerer Fehler Z-Achse [m]	0.033	0.042	0.029	0.039	0.024	0.040	0.027	0.035
Median Absolute Deviation (MAD) [m]	0.020	0.011	0.041	0.071	0.021	0.012	0.045	0.085
NNDR [dimensionslos]	0.612	0.555	0.764	0.791	0.621	0.559	0.783	0.821
Dichteunterschied [Punkte/m^3]	70.883	19.168	67.432	211.446	70.925	21.031	63.776	213.070
Chamfer-Distanz [m]	0.326	0.227	0.352	0.474	0.362	0.238	0.403	0.538
Min. Distanz [m]	0.001	0.001	0.002	0.002	0.001	0.001	0.002	0.002
Max. Distanz [m]	3.041	2.874	2.511	2.703	3.041	2.874	2.511	2.703
25% Perzentil [m]	0.021	0.021	0.041	0.050	0.021	0.021	0.041	0.050
50% Perzentil [m]	0.037	0.030	0.072	0.109	0.037	0.030	0.072	0.109
75% Perzentil [m]	0.227	0.066	0.208	0.274	0.227	0.066	0.208	0.274

Tabelle 15: Punkt-zu-Punkt Vergleiche: inkl. und exkl. Ausreißer

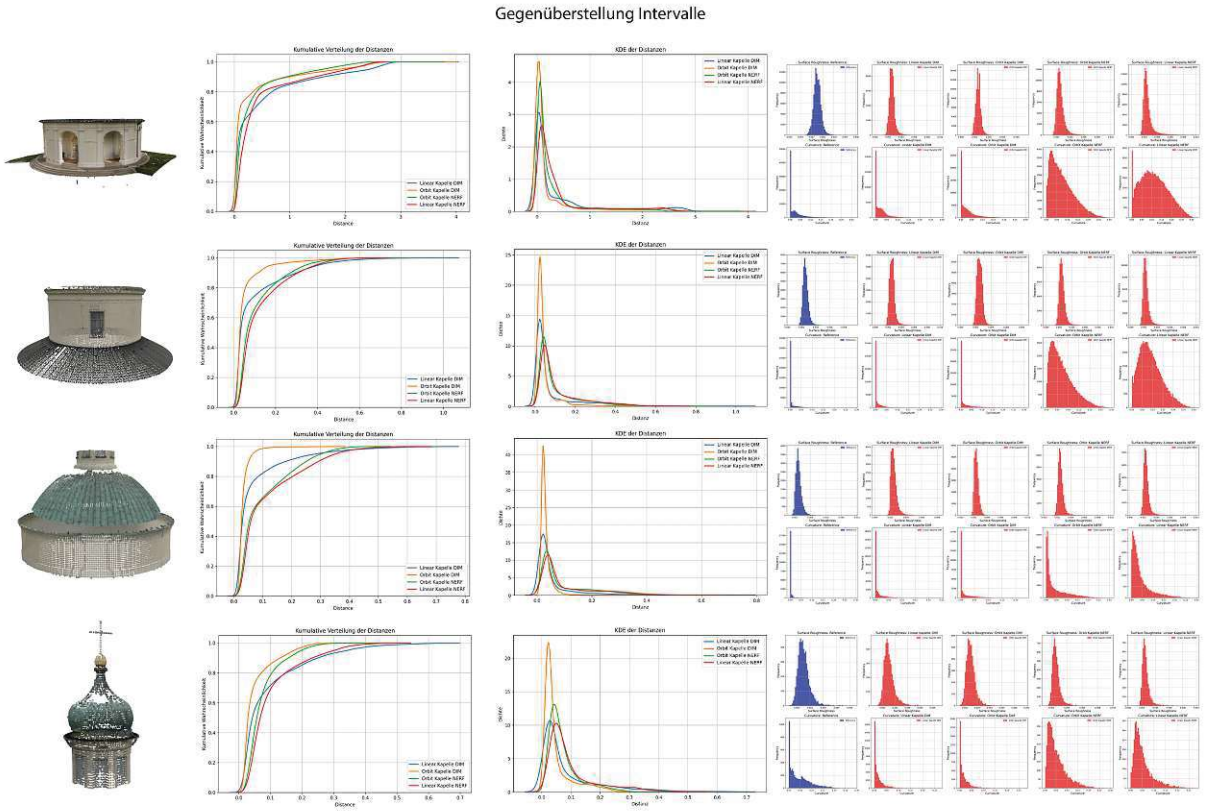


Abbildung 44: Lokale Statistiken innerhalb der Intervalle

Um sektorale Statistiken zu berechnen, wurden die Punktwolken in vier Abschnitte unterteilt. Diese sektorale Betrachtung bietet mehrere Vorteile: Zum einen reduziert sie die Verzerrung durch Ausreißer in der Gesamtauswertung, zum anderen ermöglicht sie die Analyse lokaler Unterschiede in der Rekonstruktion. Die Intervalle sind in 4-Meter-Sektionen unterteilt. Das erste Intervall reicht von einer z-Koordinate von 233 bis 240 Metern, das zweite von 240 bis 247 Metern. Weitere Intervalle wurden analog festgelegt. Abbildung 44 vergleicht die Ergebnisse. Die erste Spalte zeigt die Referenz-Punktwolke des Kapellenabschnitts, die zweite die kumulative Verteilung der Distanzen zwischen Referenz und Vergleichsdatensatz, die dritte einen Kernel Density Plot (KDE), und die letzte Spalte Histogramme von Metriken wie Oberflächenrauheit und Krümmung (Curvature). Diese Metriken sind besonders geeignet, um die lokale Geometrie innerhalb einer Nachbarschaft zu vergleichen.

Die Ergebnisse zeigen, dass der DIM Orbit Datensatz in allen Sektionen am ähnlichsten zum Referenzdatensatz ist. Die kumulative Verteilung dieses Datensatzes steigt in allen vier Fällen stark an und flacht erst bei rund 80 Prozent der Gesamtmenge an Punkten ab.

Die erste Sektion weist die größten Abweichungen vom Referenzdatensatz auf. Dies ist einerseits durch verbliebene Ausreißer in dem Datensatz, als auch auf die Existenz von Grünflächen im Datensatz zu begründen. Vegetation ist für photogrammetrische Rekonstruktionen herausfordernd. Gründe hierfür sind Unregelmäßigkeiten in der Geometrie, Bewegungsfreiheiten durch

externe Einflüsse (Wind) als auch die Texturlosigkeit. Weitere Umstände für die Erfassung von Ausreißer sind die Transparenz als auch die stärkere Lichtreflexion von Vegetation.
Die zweite Sektion zeigt die erste Dachform im Datensatz. Diese wurde frontal von den Drohnenaufnahmen abgelichtet und man kann vermuten, dass die Genauigkeit hier höher als bei vertikal ausgerichteten Ebenen ist. Diese Annahme spiegelt sich auch im KDE - Plot wieder. Der Mittelwert der Abweichungen bewegt sich für alle Rekonstruktionsmethoden unter 10 cm und die kumulative Verteilung der Distanzen steigt für jede Methode scharf an. Bei einer Distanz von 20cm schneidet sich Orbit NeRF mit Linear DIM. Dies zeigt, dass das NeRF Modell ab einem gewissen Abstandswert die Linear DIM Rekonstruktion hinsichtlich Genauigkeit überholt.
Die dritte Sektion zeigt die zweite Dachkonstruktion der Kapelle. Auffällig ist, dass Punkte im TLS-Datensatz fehlen. Bei Vergleich der beiden NeRF Ergebnisse in der kumulativen Visualisierung, kann man bis zu einem gewissen Distanzwert eine Ähnlichkeit erkennen. Ab einer Distanz von rund 0.15 cm scheint die Orbit Variante bessere Ergebnisse als die lineare Variante erzielen. Der KDE-Plot zeigt ebenfalls die Ähnlichkeit und die tendenziell bessere Rekonstruktion des orbitalen NeRF Datensatzes.
Die vierte Sektion zeigt die meisten Details des Kapellendatensatzes. Einerseits ist das Kreuz und das Zwiebeldach der Kapelle zu rekonstruieren. Betrachtet man den KDE-Plot, zeigen sich hier im Gegensatz zu den anderen Sektionen die geringsten Abweichungen zur Referenz. Alle Mittelwerte liegen innerhalb 10 cm Abweichung, wobei die maximalen Abweichungen im Gegensatz zu den anderen Sektionen geringer ausfallen. Dies deutet darauf hin, dass wenige Ausreißer in diesem Abschnitt vorhanden sind. Wie auch zuvor, zeigt sich der Orbit DIM Flug als beste Rekonstruktionsmethode und besitzt die geringsten Abweichungen zur Referenz.
Bei Betrachtung der Histogramme (dritte Spalte) sieht man die Referenz in Blau und die photogrammetrisch bestimmten Punktwolken in Rot. Die erste Zeile zeigt die Oberflächenrauheit, wobei die zweite Zeile die Curvature darstellt. Diese Metriken geben Aufschluss über die lokale Geometrie in einer Nachbarschaft.
Die Oberflächenrauheit beschreibt die kleinräumigen Abweichungen einer Oberfläche von einer idealen glatten Fläche. Sie wird bestimmt, indem die Varianz der Abstände zwischen einem Punkt und seinen benachbarten Punkten berechnet wird (Nachbarschaft 25 cm). Alle Sektionen zeigen ähnliche Ergebnisse zur Referenz. Die x-Achsen bewegen sich zwischen 0 und 0.01 und es sind keine groben Abweichungen von der Referenz zu erkennen. Gründe hierfür können einerseits in der Geometrie der Kapelle liegen. Die Kapelle ist ein glattes Objekt und weist nur wenige Variationen innerhalb einer Nachbarschaft auf. Die größten Variationen sind der Vegetation zuzuschreiben. Die Vegetation (Rasen) ist jedoch nur ein kleiner Bestandteil der Punktwolke und wirkt sich daher wenig auf die Ergebnisse aus.
Die Curvature zeigt größere Abweichungen zur Referenz. Die Curvature errechnet sich aus dem Quotienten des Minimalen Eigenwerts und der Summe aller Eigenwerte ($\frac{\lambda_{\text{min}}}{\lambda_{\text{sum}}}$). Eine Curvature von 0 bedeutet, dass die lokale Nachbarschaft flach ist, wobei eine Curvature von 1 darauf hinweist, dass die lokale Umgebung kugelförmig ist. Unter Berücksichtigung der unterschiedlichen Achsenskalierungen können parallelen im Falle NeRF - TLS beobachtet werden. Auffällig ist,

dass NeRF weniger Kurvaturen mit einem Wert von 0 besitzt, als DIM und TLS. Gründe hierfür können einerseits in der Präzision und Unschärfe der Rekonstruktion liegen. Bei erneuter Betrachtung von Abbildung 38 sieht man, dass der NeRF Rasterdatensatz mehr Strukturen bzw. Nähte am Beispiel des Daches zum Vorschein bringt, als DIM oder TLS. Abbildung 37 zeigt ebenfalls, dass die Dächer der Kapelle im Falle NeRF mehr Struktur aufweisen als DIM oder TLS. Damit lässt sich erklären, dass die Anzahl der 0-Werte geringer ist als bei den anderen Verfahren. Zusätzlich sei erwähnt, dass die Histogramme der NeRFs maximale Werte von 0.3 annehmen, welche bei DIM oder TLS nicht existieren.

Metrik	**Intervall 1**				**Intervall 2**			
	Linear DIM	**Orbit DIM**	**Orbit NERF**	**Linear NERF**	**Linear DIM**	**Orbit DIM**	**Orbit NERF**	**Linear NERF**
RMSE	0.847	0.645	0.577	0.743	0.166	0.094	0.154	0.171
MAE	0.443	0.287	0.313	0.431	0.095	0.054	0.110	0.124
Standard Deviation	0.722	0.578	0.485	0.605	0.136	0.077	0.107	0.117
Modified Hausdorff Distance	0.443	0.287	0.313	0.431	0.095	0.054	0.131	0.166
Mean Distance	0.443	0.287	0.313	0.431	0.095	0.054	0.110	0.124
Percentage within tolerance	45.486	58.250	25.314	12.626	64.972	77.324	37.634	29.947
Mean Error X-Axis	0.111	0.058	0.047	0.096	0.003	0.002	-0.001	-0.010
Mean Error Y-Axis	0.188	0.089	0.105	0.177	-0.006	-0.006	0.001	0.006
Mean Error Z-Axis	0.089	0.088	0.055	0.098	-0.006	-0.010	-0.001	-0.003
Median Absolute Deviation (MAD)	0.050	0.020	0.070	0.126	0.012	0.009	0.029	0.035
Nearest Neighbor Distance Ratio (NNDR)	0.678	0.606	0.798	0.825	0.576	0.541	0.754	0.767
Density Difference	122.296	59.136	20.714	158.468	37.303	18.965	120.602	220.955
Chamfer Distance	0.476	0.326	0.451	0.647	0.136	0.101	0.241	0.290
Metrik	**Intervall 3**				**Intervall 4**			
RMSE	0.120	0.038	0.137	0.159	0.144	0.073	0.092	0.132
MAE	0.067	0.031	0.100	0.113	0.092	0.050	0.074	0.100
Standard Deviation	0.099	0.022	0.094	0.111	0.110	0.052	0.055	0.086
Modified Hausdorff Distance	0.097	0.088	0.178	0.175	0.092	0.052	0.087	0.100
Mean Distance	0.067	0.031	0.100	0.113	0.092	0.050	0.074	0.100
Percentage within tolerance	71.030	89.422	48.745	43.865	56.460	72.916	42.074	31.574
Mean Error X-Axis	0.001	0.001	-0.005	-0.010	0.004	0.001	0.002	0.003
Mean Error Y-Axis	-0.008	-0.007	-0.001	0.002	-0.008	-0.007	-0.005	0.001
Mean Error Z-Axis	-0.008	-0.014	-0.001	-0.001	-0.005	-0.005	-0.002	-0.002
Median Absolute Deviation (MAD)	0.011	0.007	0.027	0.027	0.022	0.010	0.023	0.029
Nearest Neighbor Distance Ratio (NNDR)	0.535	0.475	0.708	0.728	0.638	0.547	0.717	0.744
Density Difference	55.472	102.477	110.291	120.449	89.636	38.575	24.501	65.196
Chamfer Distance	0.164	0.119	0.278	0.288	0.151	0.103	0.161	0.185

Tabelle 16: Vergleich der Metriken für unterschiedliche Intervalle und Methoden (DIM und NERF).

Die Abbildungen 45 bis 48 zeigen die Statistiken der Intervalle grafisch, um eine visuell ansprechendere Vergleichsbasis zu schaffen.

5.4.2 Normalenvektoren

Die Normalenvektoren sind in Nx, Ny, Nz aufzuteilen. Abbildungen 49 bis 52 zeigen eine Gegenüberstellung der Normalenvektoren zwischen dem Datensatz und der TLS Referenz. Die TLS Referenz ist dabei in rot eingefärbt.
Allgemein zeigen sich drei Peaks in den Verteilungen der TLS-Punktwolke. Normal X und Y besitzt Peaks bei $-1, 0, 1$. Normal-Z hingegen bewegt sich zwischen 0 und 1. Ein Normal-Z Wert von 0 bedeutet, dass der Vektor in der XY-Ebene liegt, wobei ein Normal-Z Wert von 1 darauf hinweist, dass der Vektor parallel zur Z-Achse ausgerichtet ist. Dementsprechend kann somit von einer horizontalen Ebene gesprochen werden. Die Referenzhistogramme deuten darauf hin, dass die meisten Punkte auf einer vertikalen als auch horizontalen Ebene sitzen. Die NeRF-Verteilungen zeigen keinen Zusammenhang zur Referenz. Die Verteilungen folgen nicht den Peaks des TLS-Datensatzen, sondern wirken eher gleichverteilt. Bei näherer Betrachtung kann man

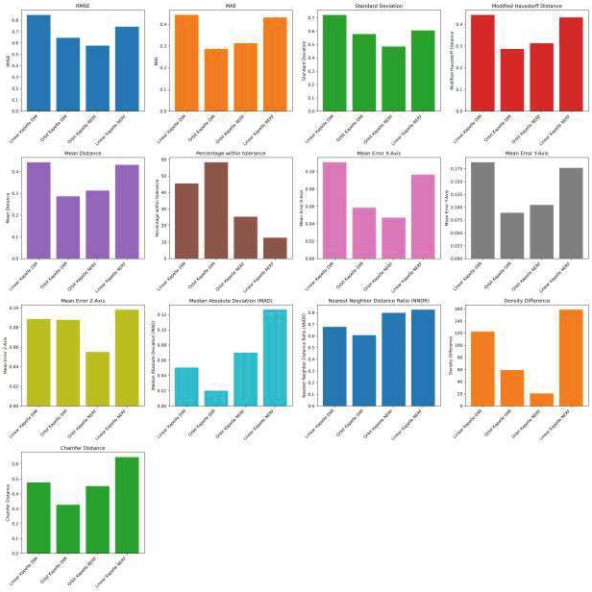


Abbildung 45: Intervall 1: Metriken

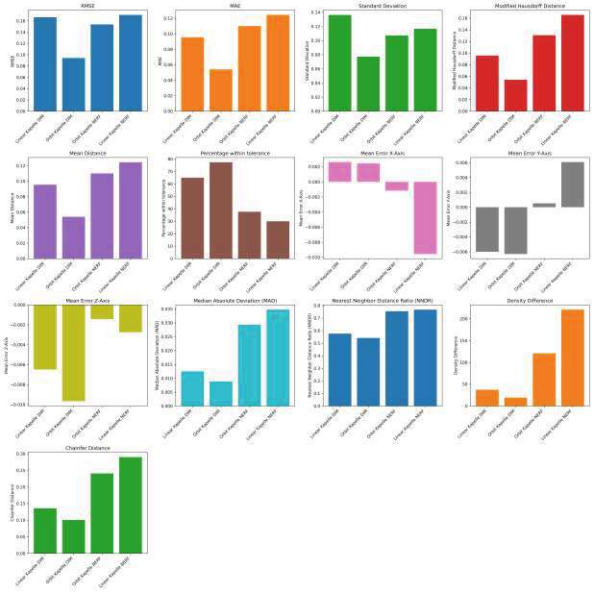


Abbildung 46: Intervall 2: Metriken

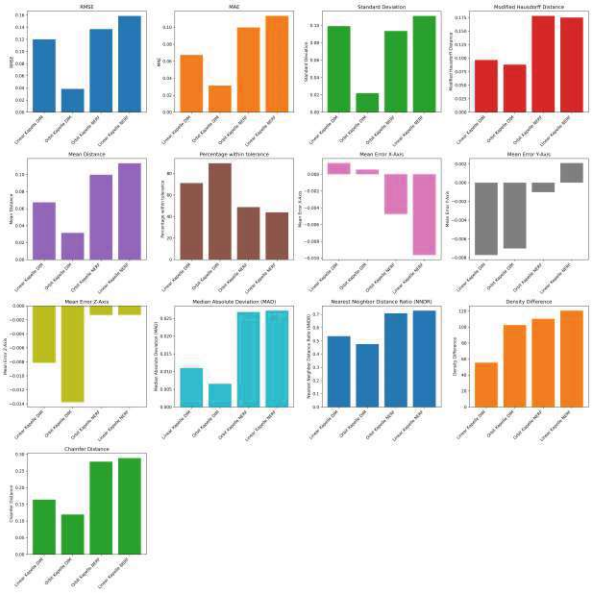


Abbildung 47: Intervall 3: Metriken

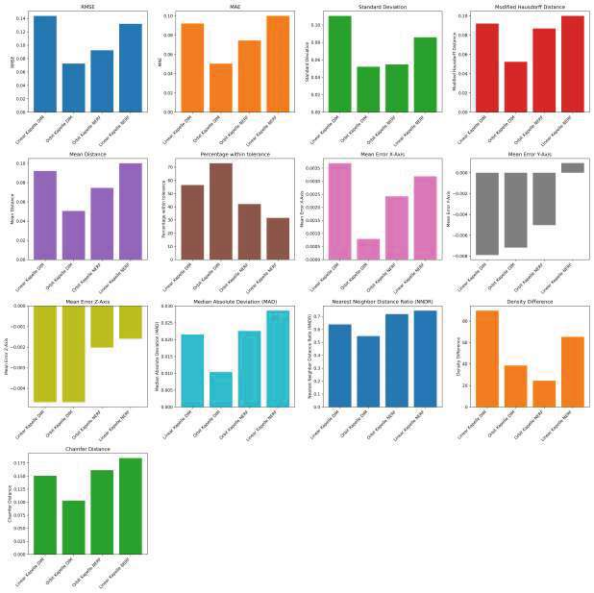


Abbildung 48: Intervall 4: Metriken

erkennen, dass auch NeRF Peaks bei 0 besitzt. Diese fallen jedoch um einiges diskreter aus als bei der Referenz. Die Referenz zeigt einen sanften Übergang auf Normal-X bzw. Normal-Y Werte von 0.

Auffällig ist, dass DIM eine ähnliche Verteilung zur Referenz (TLS) besitzt. Nur die Extremwerte von $-1, 0, 1$ zeigen stärkere Abweichungen. Gründe hierfür liegen in der bereits beschriebenen Weichzeichnung von Kanten, was für DIM-Algorithmen typisch ist.

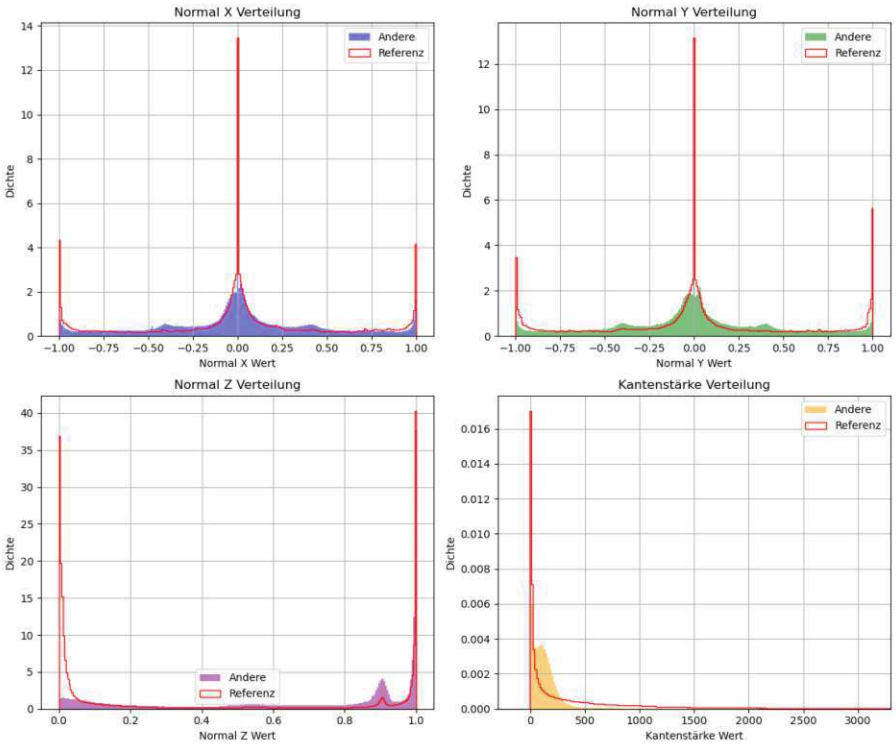


Abbildung 49: Normalenvergleich: Linear DIM

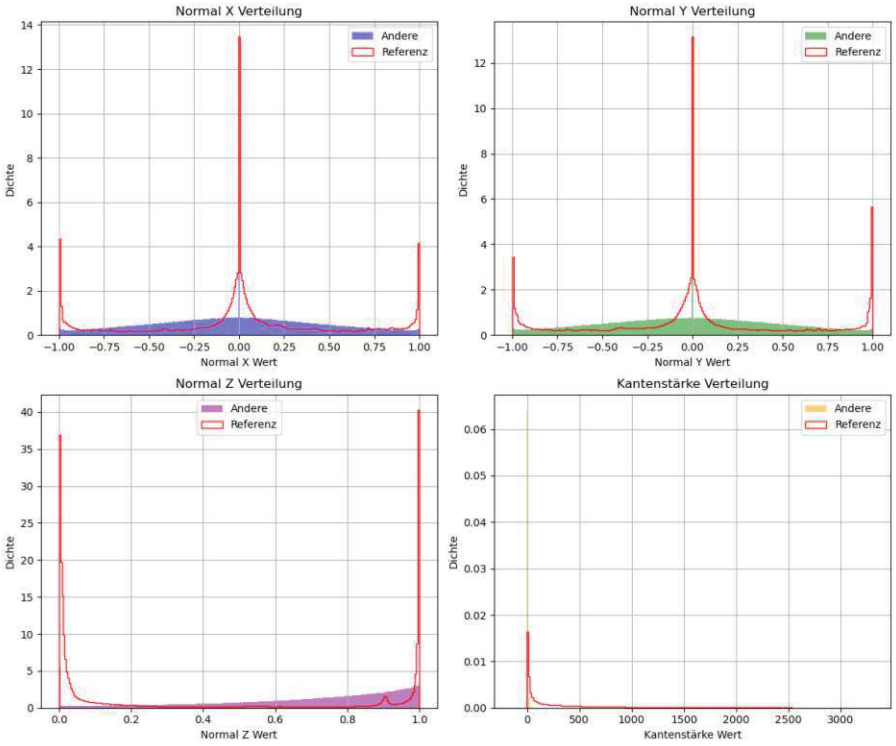


Abbildung 50: Normalenvergleich: Linear NeRF

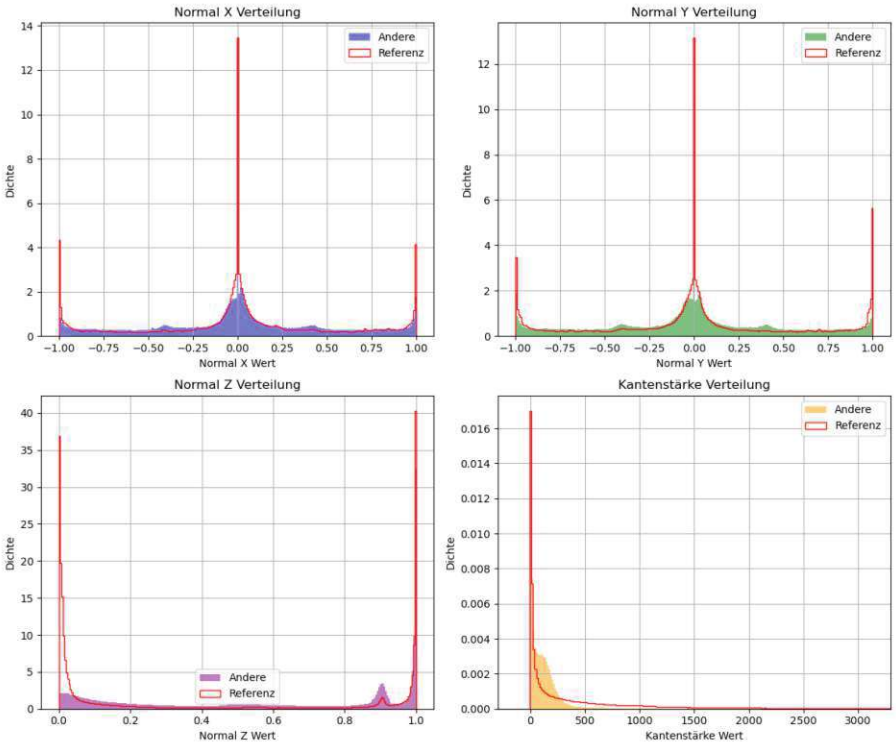


Abbildung 51: Normalenvergleich: Orbit DIM

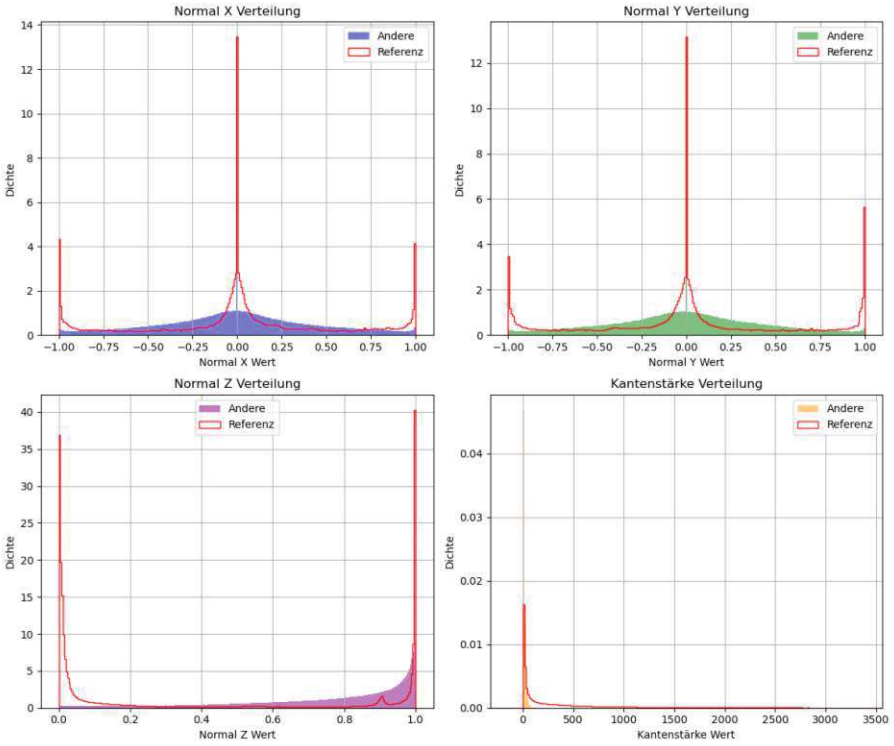


Abbildung 52: Normalenvergleich: Orbit NeRF

5.4.3 NormalSigma0

Nebenprodukt der Normalvektorbestimmung sind die NormalSigma0 Werte. Diese geben darüber Auskunft, wie präzise eine Ebene in der Nachbarschaft definiert ist. Desto weniger Rauschen eine Ebene aufweist, desto kleiner der NormalSigma0 Wert. Die Abbildungen 53 bis 56 veranschaulichen die NormalSigma0 Auswertungen. Deutlich erkennbar ist, dass die NormalSigma0-Werte für DIM tendenziell kleiner sind als für NeRF, was auf eine präzisere Ebenendefinition bei DIM hindeutet. Die linke Spalte der Abbildungen zeigt die NormalSigma0-Werte für die DIM-Datensätze, während die rechte Spalte die Werte für NeRF darstellt. Es ist zu beachten, dass die Achsenskalierungen zwischen den beiden Methoden unterschiedlich sind. Die x-Achse bei DIM reicht bis maximal 0.05, während sie bei NeRF bis 0.125 reicht.

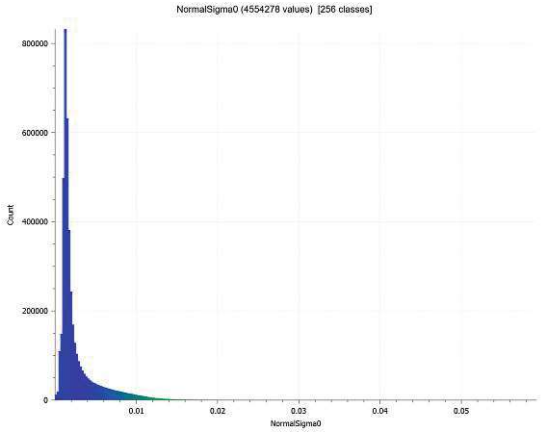


Abbildung 53: Verteilung NormalSigma0: Linear DIM

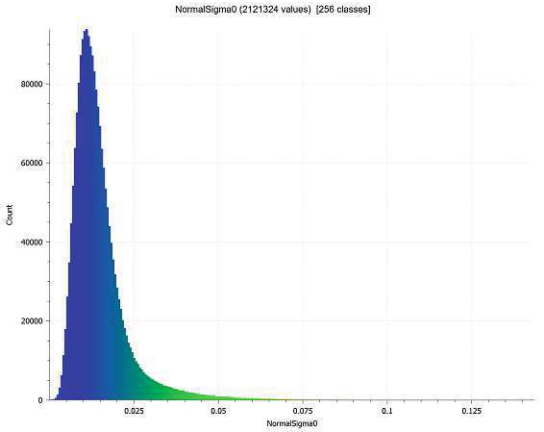


Abbildung 54: Verteilung NormalSigma0: Linear NeRF

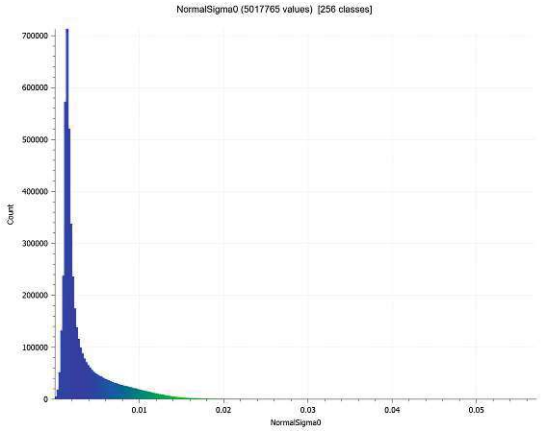


Abbildung 55: Verteilung NormalSigma0: Orbit DIM

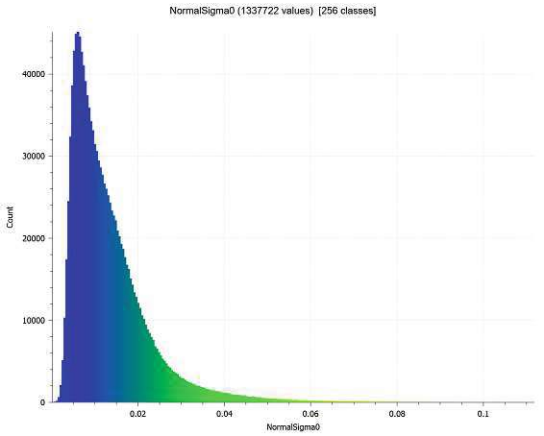


Abbildung 56: Verteilung NormalSigma0: Orbit NeRF

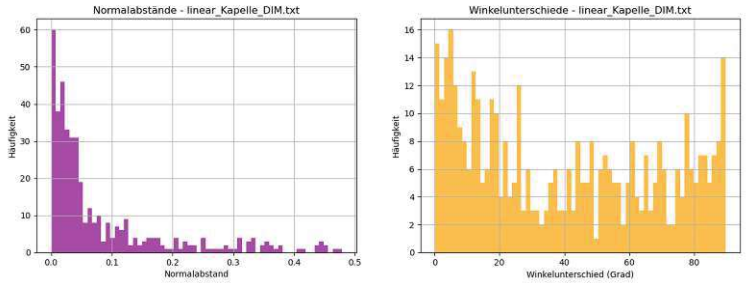
5.4.4 Plane Match Problematik - Version Toleranzen

Die Ergebnisse des Plane Matchings unter Verwendung von Toleranzen werden hier nicht berücksichtigt, da der KNN-Ansatz für die Analyse der Ebenen bevorzugt wird. Der Hauptgrund dafür ist die unterschiedliche Punktanzahl in den verschiedenen Datensätzen. Es ist nicht möglich, für jeden Punkt im TLS-Datensatz eine exakt übereinstimmende Punktkoordinate in den photogrammetrischen oder NeRF-Datensätzen zu finden. Jede Punktwolke hat aufgrund der unterschiedlichen Erfassungsmethoden und Auflösungen leicht abweichende Punktpositionen.
Da es nicht möglich ist, Punkte millimetergenau zu matchen, wird eine Punkt-zu-Ebene Korrespondenz eingesetzt. Dieser Ansatz sucht für jeden Punkt der TLS-Ebenen die k-nächsten Nachbarn in einem definierten Suchradius (zum Beispiel 50 cm) im Vergleichsdatensatz. Auf diese Weise wird trotz unterschiedlicher Punktverteilungen eine Näherung für den Vergleich der Ebenen geschaffen.
Der Vorteil des KNN-Ansatzes liegt darin, dass er auch bei variierenden Punktdichten und leicht verschobenen Punktkoordinaten sinnvolle Vergleiche ermöglicht. Somit können die Winkeldifferenzen und Normalabstände der Ebenen zwischen dem TLS-Referenzdatensatz und den photogrammetrischen oder NeRF-Datensätzen analysiert werden, ohne dass eine exakte Übereinstimmung der Punktkoordinaten notwendig ist.
Diese Methode ist daher präziser und robuster als ein Vergleich über Toleranzwerte, da sie flexibel auf die Unterschiede in der Dichte und Verteilung der Punktwolken eingeht.

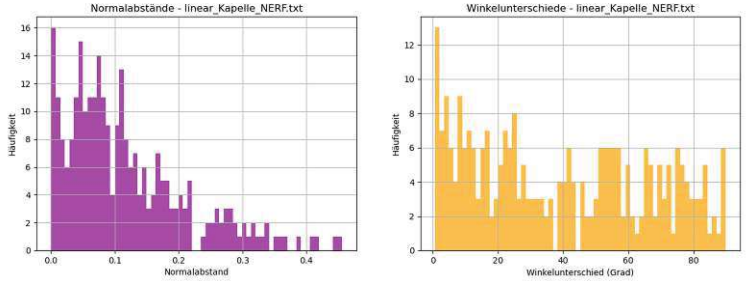
5.4.5 Plane Match - Version KNN

Um die Ergebnisse des Ebenenmatchings besser zu verstehen, sei nochmals betont, dass zunächst Ebenen aus der TLS-Kapelle extrahiert wurden, deren Schwerpunktskoordinaten und Ebenenparameter gespeichert wurden. Diese dienen als Referenz für die photogrammetrisch bestimmten Datensätze. Die Schwerpunktskoordinaten (COG) der TLS-Ebenen werden genutzt, um eine KD-Baum-Abfrage in einem Umkreis von 50 cm durchzuführen. Die minimale Punktanzahl wurde, wie bei der TLS-Kapelle, erneut auf 200 festgelegt, sodass dieselben Parameter verwendet werden und die Ebene robust gegen Ausreißer ist. Da der Vergleichsdatensatz andere Punkte als der TLS-Datensatz enthält, ändern sich die COG-Koordinaten der resultierenden Ebene. Diese Änderung wurde zugelassen und nicht fixiert.
Die Ergebnisse (s. Abbildung 57) zeigen die Winkeldifferenzen und Normalabstände der Ebenen, wobei die Normalabstände mithilfe der Hesseschen Normalform berechnet wurden. Alle Datensätze überschreiten die 50cm Normalabstand Marke nicht. Auffällig ist, dass Ebenen existieren, die 90 Grad zur Referenzebene verschoben sind. Gründe hierfür sind offensichtlich. Sobald eine leichte Verschiebung der Schwerpunktskoordinate in einem Bereich mit viel Detail auftritt, unterscheiden sich die angepassten Ebenen gröber als in Bereichen mit weniger Detail. Um die Robustheit der Ebenen zu stützen, wurde die minimale Anzahl an Punkte auf 200 Punkte festgelegt.
Die folgenden Diagramme verdeutlichen zudem die Qualität der gematchten Ebenen. Es ist

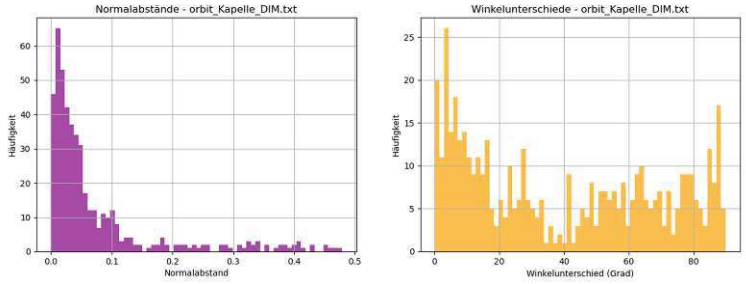
erkennbar, dass die TLS-Werte einige Ausreißer innerhalb der ersten 50 Segmente aufweisen (s. Abbildung 58). Dies ist auf die Funktionsweise der OPALS-Segmentierung zurückzuführen. OPALS fügt sukzessive Punkte zu einem Segment hinzu, was sich wiederum auf die resultierenden Ebenen auswirkt. Die Anzahl der Punkte, die zu diesem erhöhten NormalSigma0-Wert führen, müsste ebenfalls berücksichtigt werden, ist jedoch in den Diagrammen nicht ersichtlich. Betrachtet man die Segment-IDs im höheren Bereich, fällt auf, dass die Präzision der TLS-Ebenen tendenziell höher ist als bei den photogrammetrischen Datensätzen. Besonders im Vergleich zu den NeRF-Datensätzen weichen die Sigma0-Werte stärker von der TLS-Referenz ab, was darauf hindeutet, dass die Ebenen im Fall von NeRF unschärfer dargestellt werden.



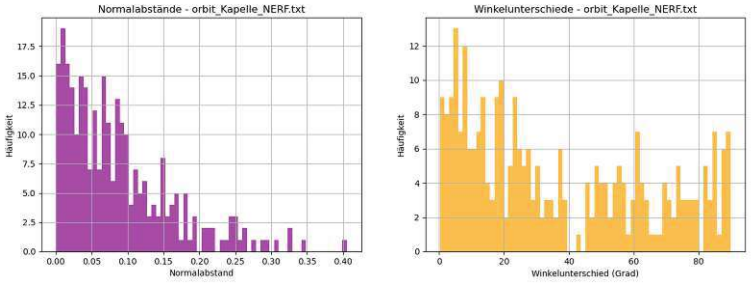
(a) linear DIM



(b) linear NeRF



(c) orbit DIM



(d) orbit NeRF

Abbildung 57: Plane Match Ergebnisse

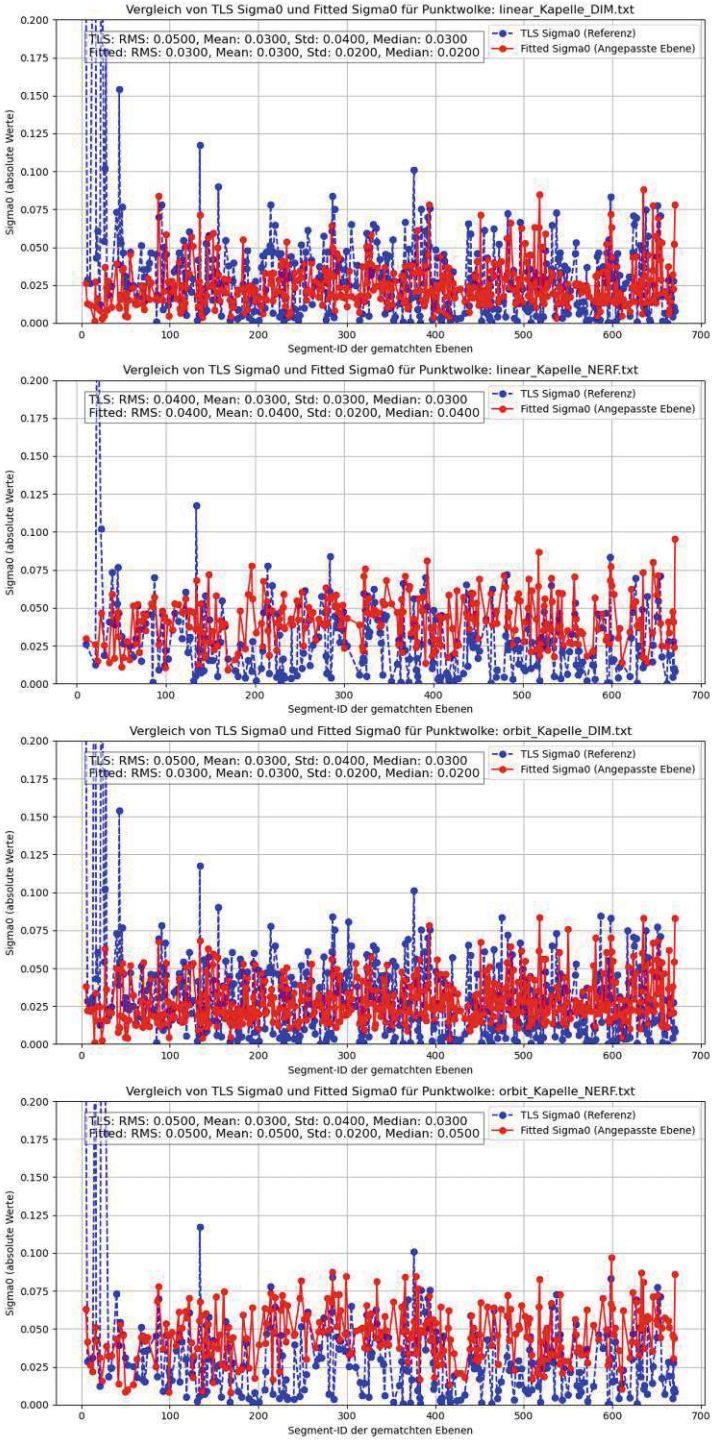


Abbildung 58: NormalSigma0 Gegenüberstellung

5.4.6 M3C2

Die Ergebnisse des M3C2-Ansatzes sind in zwei Verfahren unterteilt. Im ersten Verfahren wird für jeden Vergleichsdatensatz eine individuelle Unschärfe berechnet[24], während im zweiten Verfahren die Unsicherheit verallgemeinert und ein gemeinsamer Unschärfewert zur Vergleichbarkeit der Ergebnisse herangezogen wird.
Der erste Ansatz eignet sich besonders, um den TLS-Datensatz direkt mit dem jeweiligen Vergleichsdatensatz gegenüberzustellen. Die individuelle Präzision ermöglicht es, signifikante Abweichungen gezielt zu visualisieren. Der zweite Ansatz hingegen zielt darauf ab, alle Datensätze untereinander vergleichbar zu machen, sodass signifikante Unterschiede zwischen ihnen herausgearbeitet werden können. Je höher die Unschärfe, desto schwieriger wird es, Deformationen zwischen den Datensätzen aufzudecken, da das Messrauschen die tatsächlichen Deformationen überlagert. Aus diesem Grund wurde eine mittlere Unschärfe gewählt, da dadurch die Vergleichbarkeit der Menge an signifikanten Abweichungen innerhalb der Datensätze gegeben ist.

Metrik	**linear_DIM**	**orbit_DIM**	**linear_NERF**	**orbit_NERF**
Mean Distance (m)	0.01	0.01	-0.03	0.01
Median Distance (m)	0.01	0.01	-0.01	-0.00
Std Dev Distance (m)	0.08	0.07	0.18	0.16
Min Distance (m)	-0.67	-0.70	-0.70	-0.70
Max Distance (m)	0.66	0.73	0.69	0.67
Unschärfe/Präzision/Unsicherheit (m)	0.08	0.07	0.18	0.16
Percent Significant Change (%)	2.16	2.10	2.66	5.04

Tabelle 17: Ergebnisse mit individueller Unschärfe

Tabelle 17 zeigt die Ergebnisse des M3C2-Algorithmus, wobei für jeden Datensatz ein eigener Unsicherheitsparameter bestimmt wurde. Erkennbar ist, dass linear NeRF bezüglich Rekonstruktionsqualität die höchsten Abweichungen aufweist. Bei Vergleich der geschätzten Unsicherheiten erkennt man, dass Orbit DIM die niedrigste Unschärfe zur Referenzpunktwolke besitzt, was sich wiederum auf die Aufdeckung von signifikanten als auch nicht signifikanten Abweichungen auswirkt. NeRF zu DIM weist eine höhere Unschärfe der Rekonstruktion mit Faktor 2 auf. Der Prozentsatz der signifikaten Abweichungen ist bei NeRF dennoch um das zweifache höher als bei DIM, trotz einer höheren Signifikanzschwelle, was wiederum für die Qualität der DIM-Punktwolken spricht.

Metrik	**linear_DIM**	**orbit_DIM**	**linear_NERF**	**orbit_NERF**
Percent Significant Change (%)	1.27	1.01	14.68	10.81

Tabelle 18: Ergebnisse mit gemeinsamer Unschärfe (12cm)

Bei Unterlassung einer individuellen Unschärfe spalten sich die Ergebnisse deutlicher. Der Prozentsatz der signifikanten Änderungen steigt hier auf das 10 bis 14-fache an. Da die M3C2-

[24]Die Unschärfe wird mit der Standardabweichung der M3C2-Distanzen bestimmt.

Distanzen unberührt von der Wahl der Unschärfe bestimmt werden, ändert sich im Vergleich zu Tabelle 17 nichts an den statistischen Kennzahlen. Nur der Prozentsatz des signifikanten Anteils (s. Tabelle 18).

Wie auch bei der Tabellenansicht, unterteilen sich die Abbildungen in individuelle und einem gemeinsamen Unsicherheitsparameter. Abbildungen 59 bis 62 zeigen die Ergebnisse mit individuellem, wobei 63 bis 66 die Ergebnisse mit der durchschnittlichen Präzision zeigen[25].
Die M3C2-Distanzen werden in Form eines Histogramms dargestellt. Es zeigt sich, dass die DIM-Methoden eine höhere Genauigkeit aufweisen als der NeRF-Ansatz. Der Einfluss des Flugmusters wirkt sich im Falle DIM nur wenig auf die M3C2-Distanzen aus. Im Gegensatz dazu streut im Falle NeRF das Histogramm bei dem Orbit Flugmuster weniger als bei der linearen Variante. Der Boxplot hebt diese Feststellung hervor. Bei Betrachtung der M3C2 Distanzen ist auffällig, dass der Orbit Flug geringere Abweichungen als der lineare Flug (NeRF) aufweist (Abbildungen links oben). Speziell bei Vegetation schneidet die Orbit Variante besser ab. Die Pflastersteine der Kapelle wurden von beiden Aufnahmekonfigurationen präziser bestimmt als die Vegetation. Bereiche, welche direkt zur Kameraposition ausgerichtet sind, weisen eine höhere Rekonstruktionsqualität ab als vertikal ausgerichtete Bereiche. Die Signifikanten Änderungen sind bei DIM ähnlich verteilt. Auffällig ist jedoch, dass Kanten im linearen Fall vermehrt signifikant abweichend sind von der Referenz. Bei Vergleich der NeRF Rekonstruktionen fällt kein wesentlicher unterschied auf.
Die Unterschiede werden bei Einsatz der verallgemeinerten Unsicherheit deutlicher hervorgehoben (s. Abbildung 63 bis 66). Während die M3C2 Distanzen bei DIM relativ gering ausfallen, zeigt sich, dass der orbitale Flug (NeRF) den linearen Flug hinsichtlich Qualität deutlich übersteigt. Die in früheren Kapiteln angesprochene Unschärfe der Vegetation nimmt bei einem orbitalen Flugmuster ab, was sich positiv auf die Rekonstruktion auswirkt.

[25] Im Kontext dieses Kapitels werden die Begriffe Unsicherheit, Präzision und Unschärfe gleichgesetzt.

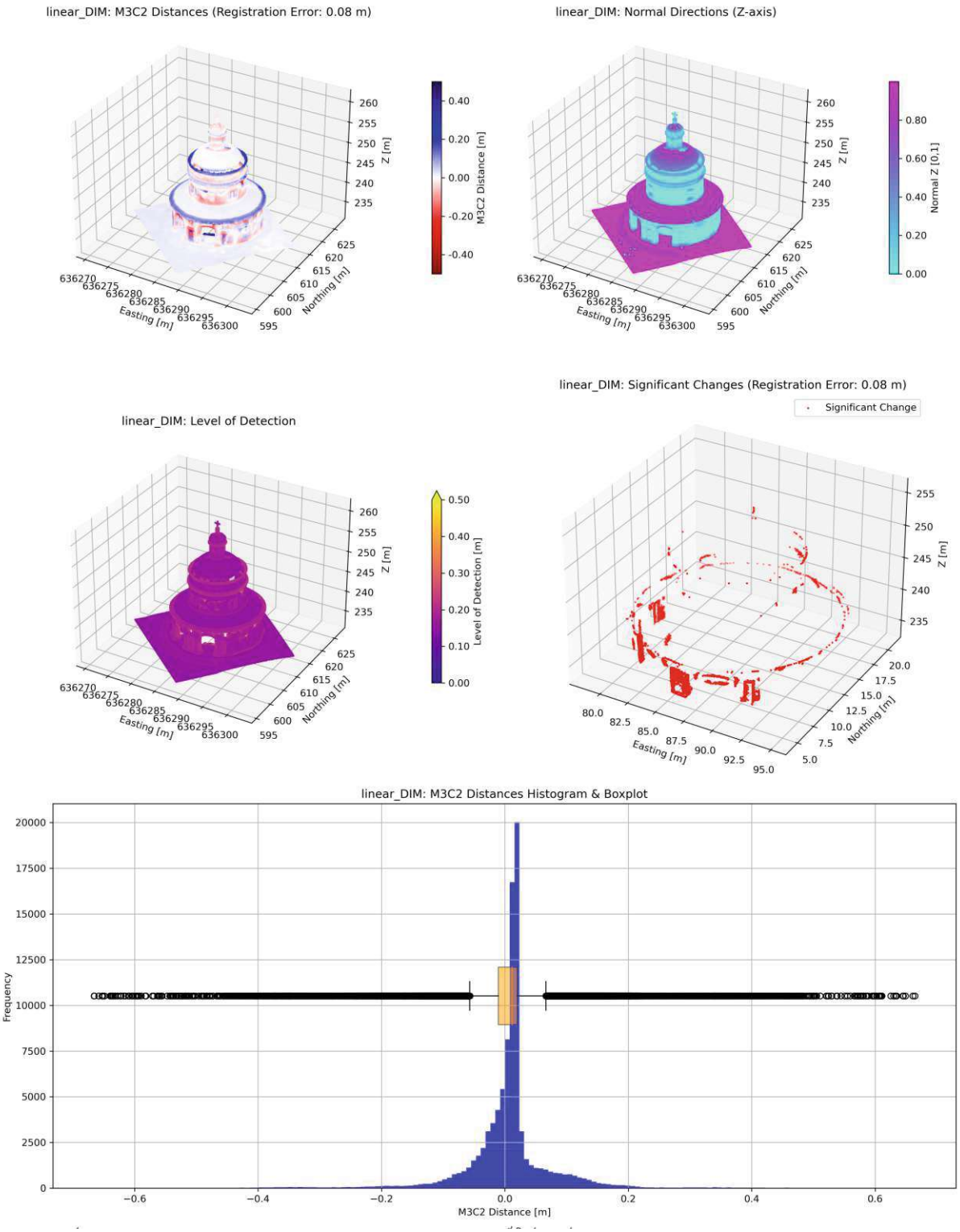


Abbildung 59: Linear-DIM: M3C2 - individuell

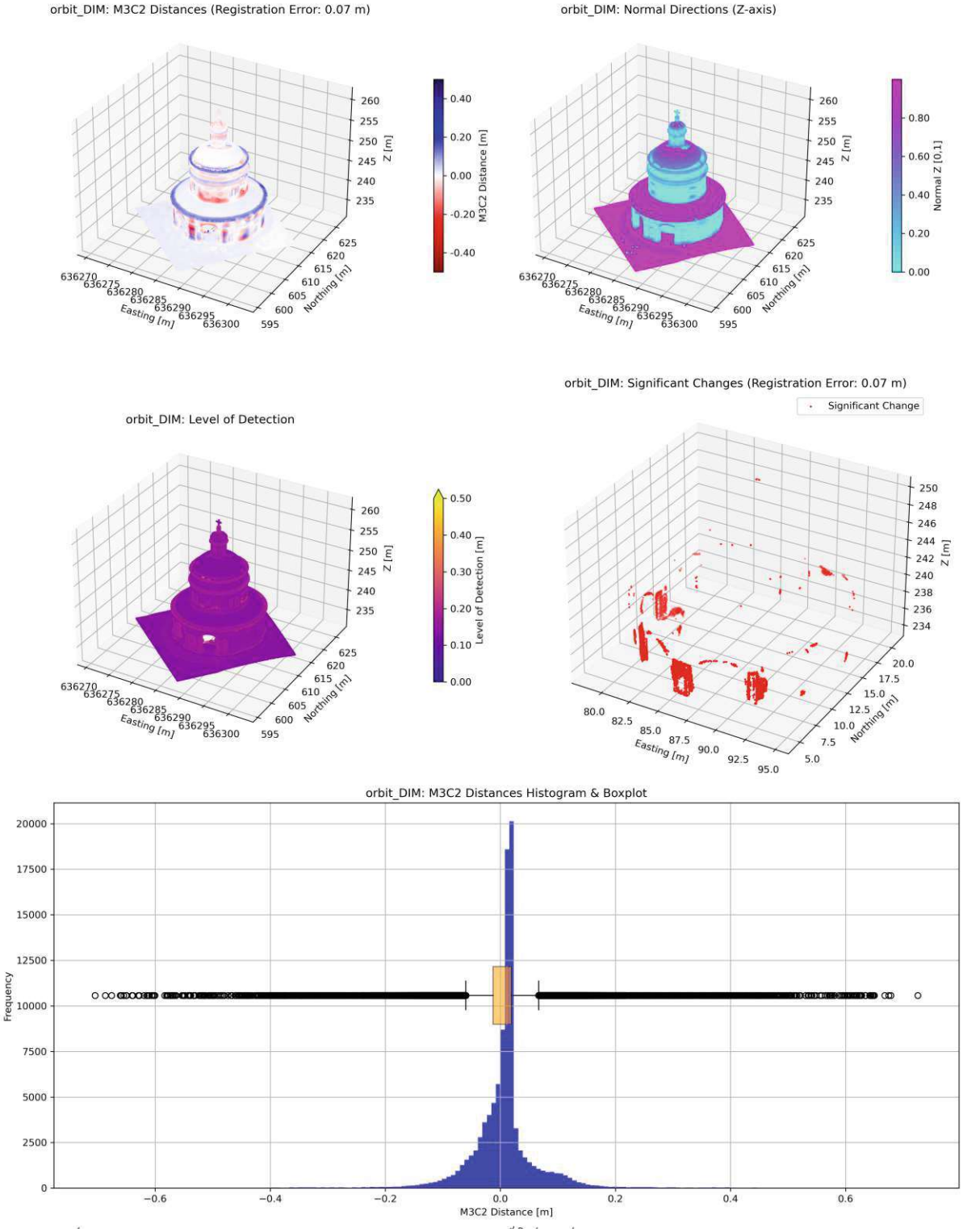


Abbildung 60: Orbit-DIM: M3C2 - individuell

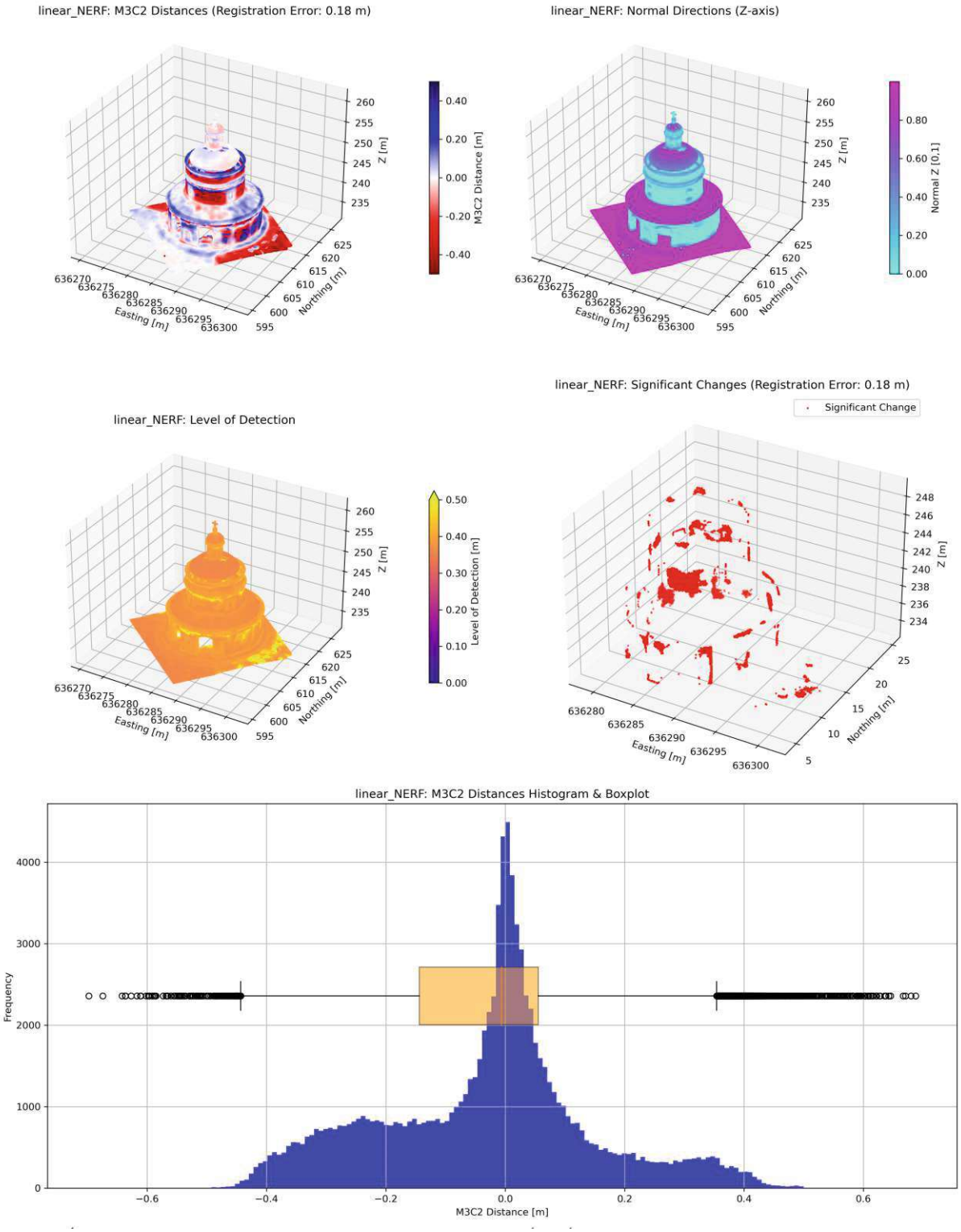


Abbildung 61: Linear-NeRF: M3C2 - individuell

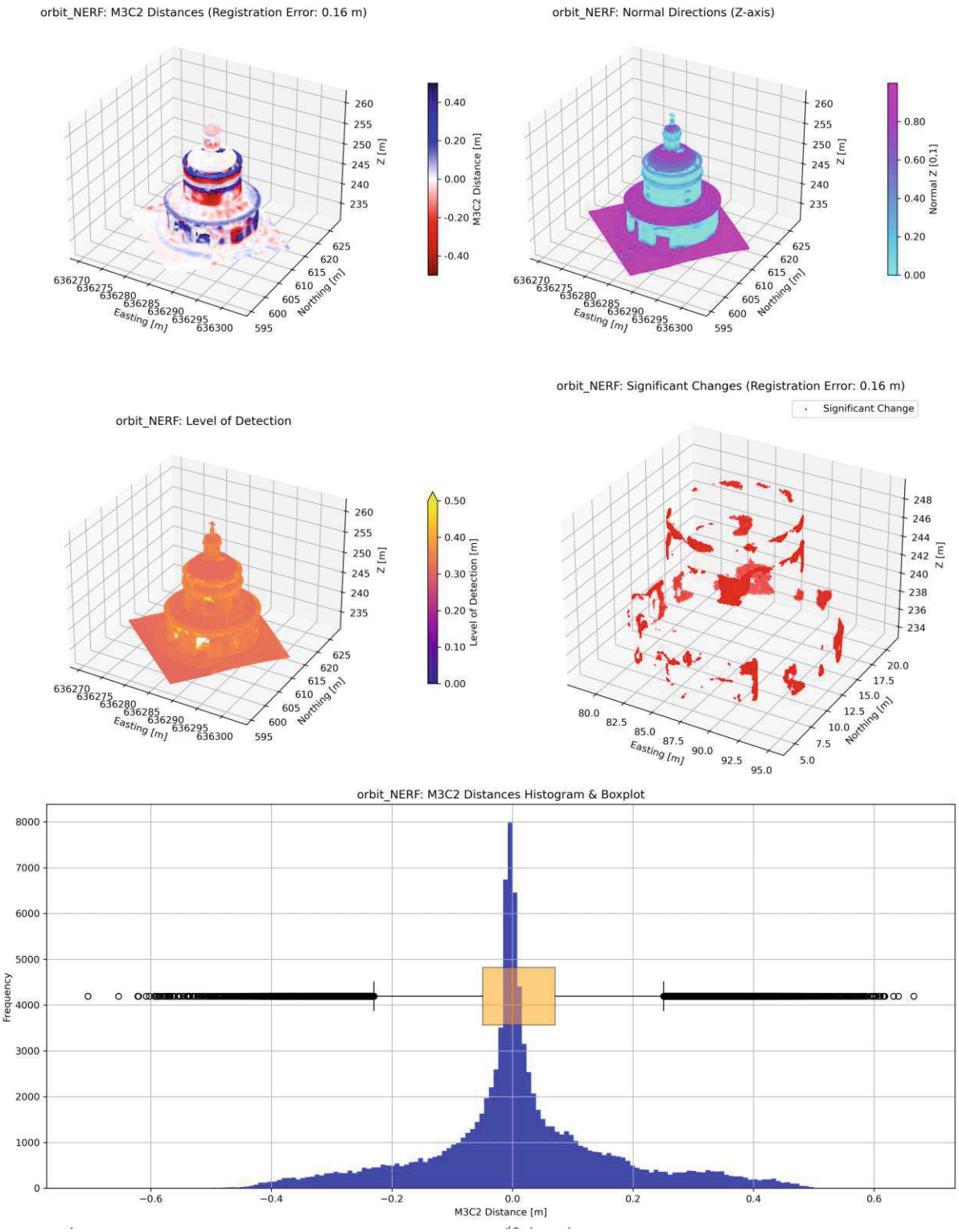


Abbildung 62: Orbit-NeRF: M3C2 - individuell

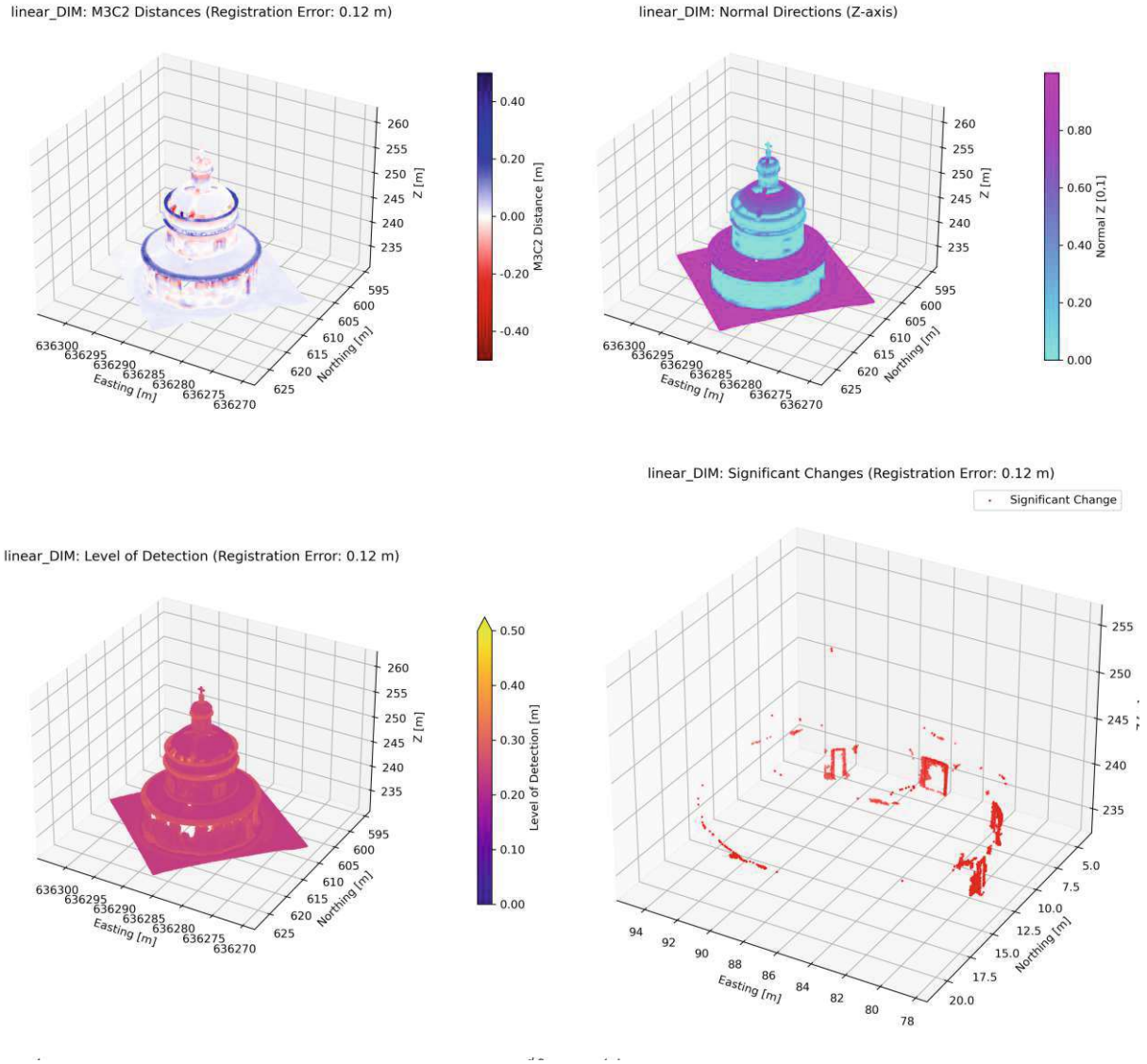


Abbildung 63: Linear-DIM: M3C2 - 12 [cm]

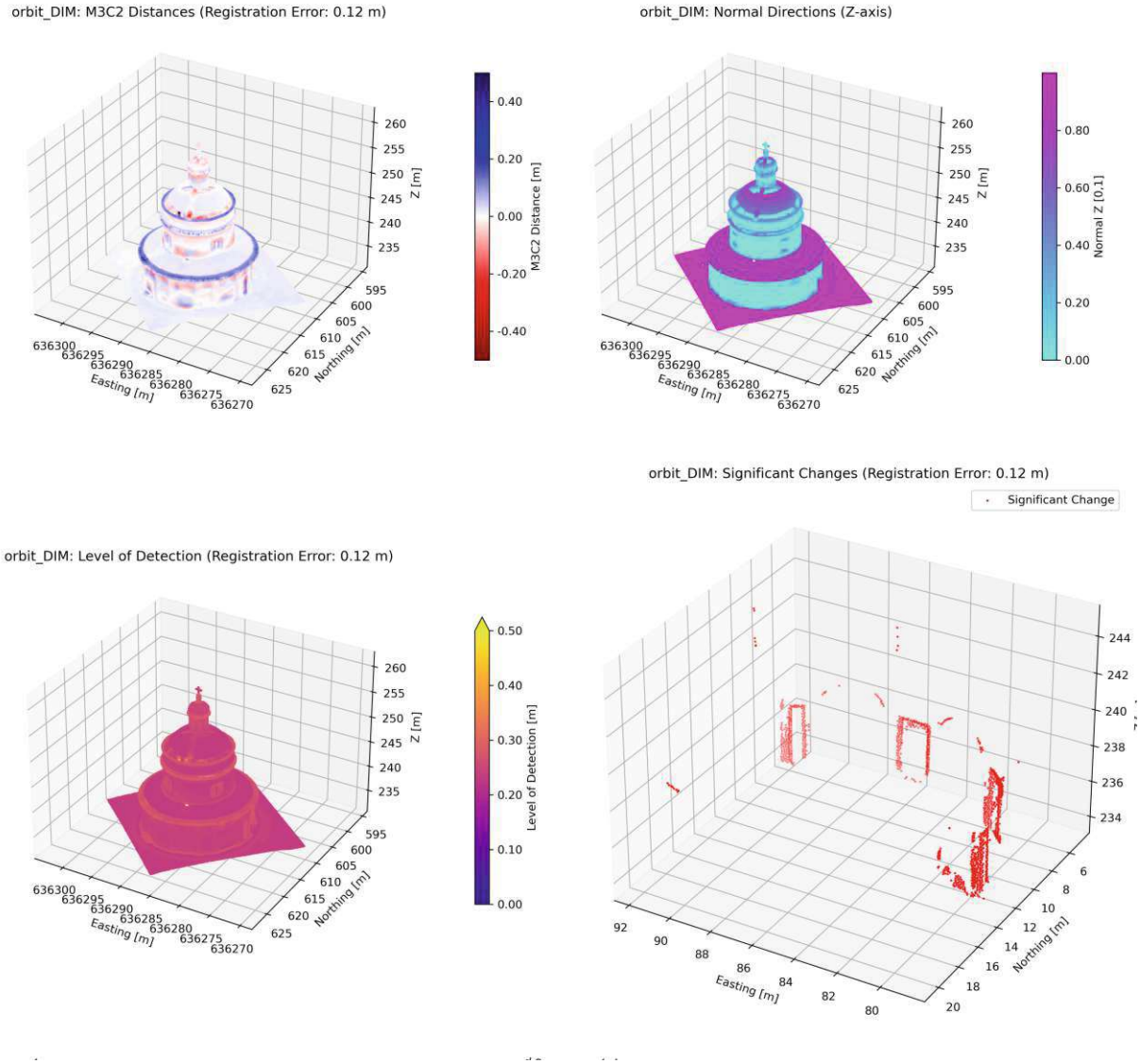


Abbildung 64: Orbit-DIM: M3C2 - 12 [cm]

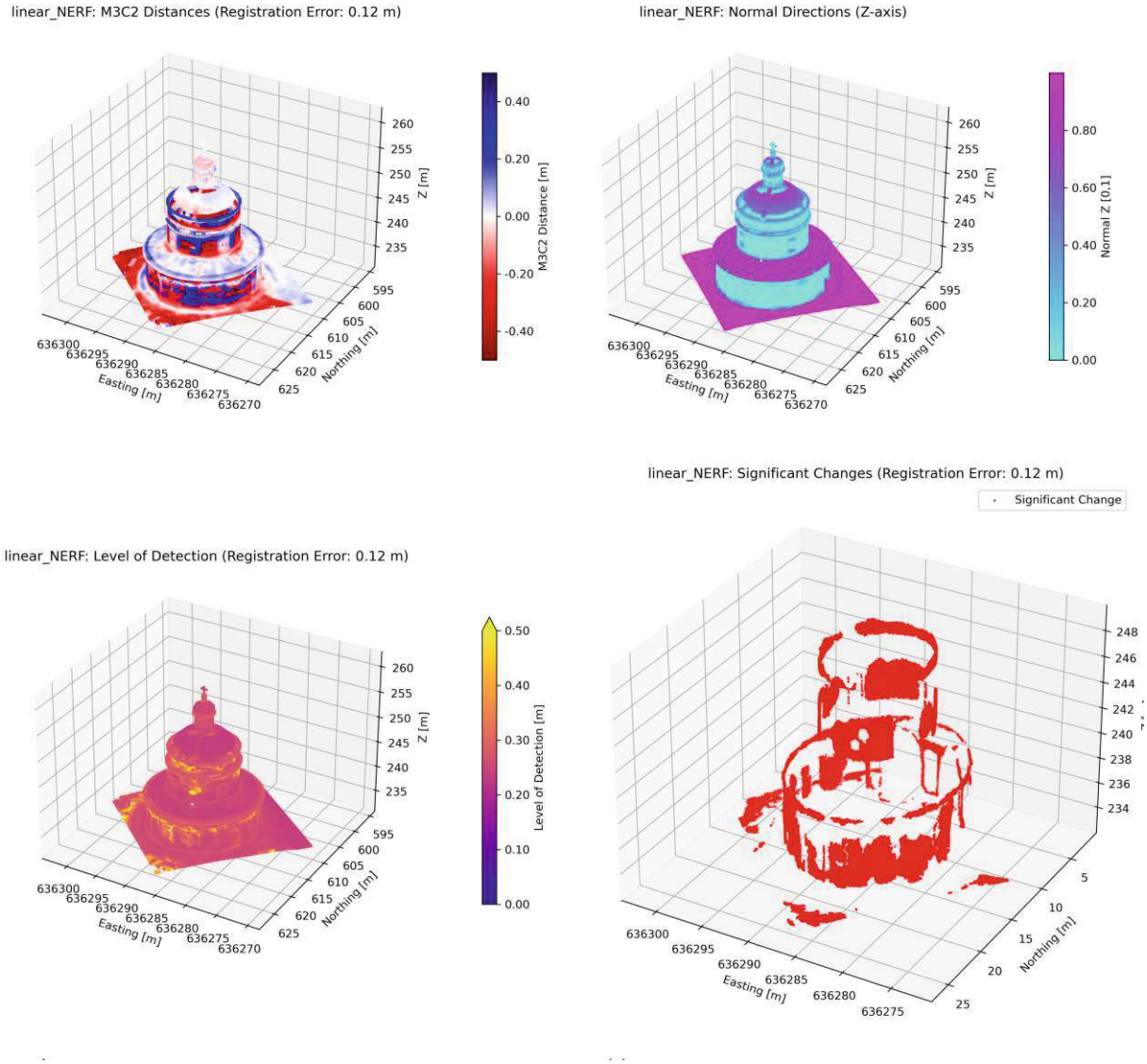


Abbildung 65: Linear-NeRF: M3C2 - 12 [cm]

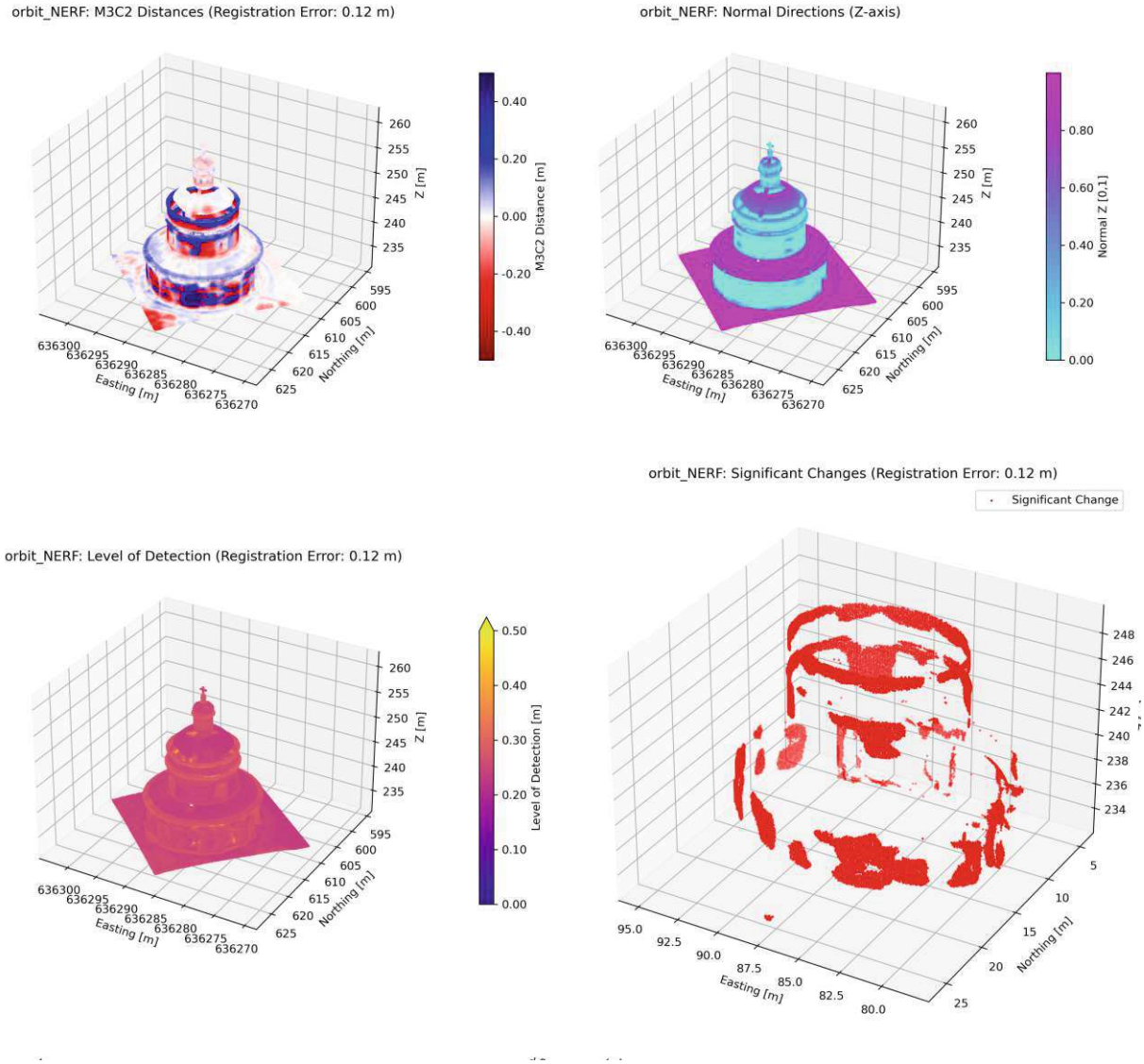


Abbildung 66: Orbit-NeRF: M3C2 - 12 [cm]

6 Beantwortung der Forschungsfragen

Das folgende Kapitel diskutiert und interpretiert die erreichten Ergebnisse. Darüber hinaus werden die Forschungsfragen basierend auf der Untersuchung beantwortet. Außerdem sollen die Grenzen der beiden Verfahren beschrieben, und ein Anreiz für zukünftige Forschungen gegeben werden.

6.1 Einflussfaktoren auf die Genauigkeit eines Modells

Im Sinne von UAV-Mapping gibt es mehrere Einflussfaktoren, die die Genauigkeit eines Datensatzes beeinträchtigen. Professionelle Drohnen, wie z.B. die DJI Matrice 350 RTK, besitzen ein GNSS-RTK Modul, welches die Position der Drohne sekündlich mit einer Genauigkeit zwischen 1-3cm bestimmen kann. Da das UAV nicht starr in der Luft verharrt und ein bewegtes Objekt darstellt, müssen weitere Sensoren die Position der Drohne, innerhalb der sekündlichen RTK Abtastung des GNSS, stützen[26]. Bei Durchführung eines Mapping-Auftrages werden gemäß der angegebenen Überlappung, Bilder automatisch von der Drohne aufgenommen. Da GNSS nur eine sekündliche Auflösung hat, muss zum Zeitpunkt des Fotos die Position der Drohne geschätzt bzw. interpoliert werden. Hierfür werden Beschleunigungssensoren eingesetzt, die hochfrequent die Lage und Rotation des UAV im Raum beobachten. Da Beschleunigungssensoren die Eigenschaft besitzen ab einer gewissen Zeit abzudriften, bedarf es gut überlegter Bewegungsgleichungen, um die Trajektorie des UAV im Raum zu modellieren. Die Kombination aus GNSS und Sensorik trifft daher z.B. oftmals auf die Welt der Kalman-Filterung. Der Kalman-Filter ist ein rekursiver Algorithmus, der Schätzungen von Zuständen eines dynamischen Systems durch die Kombination von Messdaten und einem mathematischen Modell verbessert. In Bezug auf die Trajektorienbestimmung des UAV gleicht der Kalman-Filter Unsicherheiten aus den GNSS-Daten mit den hochfrequenten Beschleunigungsmessungen ab. Die geschätzten Genauigkeiten der Bewegung eines UAV sind somit auf die Qualität der GNSS-Lösung, die Sensorik und die eingesetzte Bewegungsgleichung beschränkt. Ein weiterer Einflussfaktor für die resultierende absolute Genauigkeit eines Modells ist der Einsatz von GCP's. Aufgrund der RTK-Fähigkeit von Drohnen könnte man denken, dass keine Kontrollpunkte notwendig wären. Durch Einsatz von GCPs können aber Fehlereinflüsse auf das Modell aus der Bewegungsschätzung des UAV kontrolliert bzw. verbessert werden. Der Einsatz von GCPs ist somit die erste Kontrollstufe eines Modells und deckt Abweichungen aus der Befliegung auf. Zusätzlich wird die Lagerung in das geodätische Datum gestützt bzw. kontrolliert.
Der nächste Einflussfaktor ist die Berechnung der relativen Orientierung. Durch den in der Arbeit beschriebenen SfM Workflow können heutzutage robust die Positionen von Kameras relativ zueinander bestimmt werden. Dennoch ist die Genauigkeit der relativen Orientierung abhängig von der Auflösung der Kamera und dem gewählten Objekt. Nicht zu vergessen ist die Qualität eines Fotos. Unscharfe oder überbelichtete Fotos sind für photogrammetrische Auswertungen un-

[26] Üblicherweise wird von Korrekturdatendiensten eine sekündliche Abtastung über NTRIP zur Verfügung gestellt.

brauchbar, da keine Merkmalspunkte extrahierbar sind. Die innere Orientierung wird innerhalb eines Bündelblockausgleichs geschätzt. Zuvor muss jedoch ein entsprechendes Verzeichnungsmodell gewählt werden. Bei falscher Wahl des Verzeichnungsmodell kommt es zu tangential oder radialen Residuen, die ebenfalls einen Einfluss auf die resultierende Genauigkeit haben.
Wie man erkennen kann, ist die Qualität eines photogrammetrischen Modells, obwohl bis jetzt noch keine Rede von DIM oder NeRF war, von einigen Einflussfaktoren betroffen. Somit sind ab dem Zeitpunkt des Einsatzes von NeRF oder DIM bereits Unsicherheiten gegeben, die auf die resultierende Genauigkeit eines Modells einwirken.

6.2 Georeferenzierung

Da der DIM-Workflow seit Jahren in der Praxis eingesetzt wird, hat sich demnach auch der Softwaremarkt weiterentwickelt. Softwareprodukte wie z. B. Agisoft Metashape oder Pix4D unterstützen den User innerhalb des DIM-Workflows vom Import der Fotos bis hin zum Export des Modells. Demnach ist es auch für Laien schnell möglich, aus Fotos ein passables 3D-Modell zu erstellen. Die Verortung des Modells wird dabei direkt in den Workflow integriert und bei Einsatz von GCPs nochmals hinsichtlich der Genauigkeit verbessert.
Da NeRFs erst seit Kurzem einen Aufschwung in Bezug auf ihre Beliebtheit erfahren haben, sind bis dato nur wenige kommerzielle Softwarelösungen auf dem Markt vorhanden. Einer der Pioniere ist LumaLabs AI. Dem User wird eine Online- und App-Plattform zur Verfügung gestellt, auf der NeRFs ohne jegliches theoretisches Wissen erstellt werden können. Das Problem, das sich hinsichtlich der Georeferenzierung herausstellt, ist, dass NeRFs nicht ausschließlich für geodätische Zwecke entwickelt wurden, sondern vor allem für die Film- oder Spieleindustrie. Durch die visuell ansprechende Rekonstruktion eines Objekts kann man oft den Unterschied zwischen Realität und Fiktion nicht erkennen. Damit soll gesagt werden, dass die Verortung eines NeRFs bis dato noch nicht im zentralen Interesse der Community stand.
Die Georeferenzierung eines NeRF-Punktwolkendatensatzes muss daher aktuell noch über eine ICP-Registrierung auf eine Referenz erfolgen, wodurch eine zusätzliche Unsicherheit zu dem bereits erwähnten System hinzukommt. Dies ist ein wesentlicher Punkt, der die Einsetzbarkeit von NeRFs für geodätische Anwendungen einschränkt. Im Falle von DIM können innerhalb erprobter Software im Subpixelbereich Punkte ausgewählt werden, die die Qualität der absoluten Genauigkeit nochmals erhöhen. Diese Art von gestützter Georeferenzierung gibt es aktuell bei NeRFs noch nicht. Die Qualität der Georeferenzierung im Falle von NeRFs ist daher zu diesem Zeitpunkt mangelhaft und bedarf weiterer Entwicklungsschritte.

6.3 Rekonstruktionsqualität der Modelle

Es hat sich herausgestellt, dass die Präzision der resultierenden Modelle deutliche Unterschiede aufweist. NeRF-Modelle tendieren bei der Abbildung von Vegetation (z. B. Gras) dazu, hohe Unschärfen zu zeigen. Punkte auf grünen Oberflächen sind tendenziell stärker verrauscht als auf Asphaltflächen. Gründe hierfür könnten die hochfrequenten Änderungsraten und Variationen in

der Grashöhe sein.
Weiterhin wurde festgestellt, dass die resultierenden Punktwolken von NeRF aus der Blickrichtung der aufgenommenen Fotos sehr vielversprechend wirken. Bei einer Änderung der Blickrichtung zeigt sich jedoch, dass die Tiefenwahrnehmung von NeRFs aufgrund der koplanaren Aufnahmekonfiguration des Fluges eingeschränkt ist. Vertikale Ebenen erscheinen aus der Sicht der Aufnahme planar, bei einer erneuten Änderung der Ansicht können jedoch „Dellen“ erkannt werden. Dies lässt darauf schließen, dass NeRFs stark von einer verteilten Aufnahmekonfiguration mit ausreichender Schnittgeometrie abhängen.
Die Schnittgeometrie ist auch im Falle von DIM relevant, hat jedoch weniger Auswirkungen auf die resultierende Punktwolke. Die Triangulation erzielt hier detaillierte Ergebnisse. Bei der Abbildung feiner Strukturen, wie z. B. des Stucks der Kapelle, können die NeRF-Datensätze in Bezug auf die Qualität nicht mit DIM verglichen werden. DIM war in der Lage, feinere Strukturen der Kapelle hervorzuheben und zu rekonstruieren. Interessanterweise hat jedoch NeRF bei den Falzen des Blechdaches mehr Details erkannt, was auf eine höhere Genauigkeit bei der Modellierung dieser spezifischen Oberflächenstrukturen hindeutet.
Wie bereits in der Arbeit erwähnt, ist die Vollständigkeit der Punktwolke im Falle von DIM nicht gegeben. Die weiße Fassade der Kapelle besitzt wenig Textur, was zu fehlenden Daten innerhalb der DIM-Punktwolke führte. NeRFs hingegen konnten Punkte selbst bei wenig Textur rekonstruieren. Hier zeigt sich auch der Vorteil eines ganzheitlichen Ansatzes gegenüber einem lokalen Ansatz. Das FCNN erlernt nicht nur die Geometrie des Objekts, sondern auch die Lichtverhältnisse. Die Reflektanz von Materialien ist dabei eine weitere Unbekannte, die im Netzwerk erlernt wird. Dies ist bei DIM-Verfahren undenkbar, da diese rein auf geometrischen Überlegungen basieren.
Zukünftige Arbeiten könnten untersuchen, ob unterschiedliche DIM-Verfahren signifikant unterschiedliche Ergebnisse liefern. Ebenso wäre es lohnenswert, kommerzielle Softwarelösungen wie Pix4D oder RealityCapture in diesen Vergleich einzubeziehen. Solche Untersuchungen könnten aufzeigen, ob und wie verschiedene DIM-Technologien die Rekonstruktion von texturarmen oder geometrisch komplexen Oberflächen wie der Kapelle beeinflussen.
Der Baum-Datensatz war im Falle von DIM und NeRF mit schwierigen Ausgangsbedingungen verknüpft. Aufgrund der erhöhten Windgeschwindigkeiten konnte nicht von einem starren Objekt ausgegangen werden, was jedoch eine Voraussetzung für beide Methoden wäre. In der Arbeit wurde festgestellt, dass die lineare Punktwolke von Nerfstudio nicht exportiert werden konnte. Die Gründe hierfür sind bisher unbekannt. Da bei NeRF ein ganzheitliches Strahlenbündel durch jedes Pixel ausgesandt wird und mit konvergenten Strahlenbündeln verschnitten wird, sind NeRFs extrem sensibel gegenüber Bewegungen des Objekts. Wird die Forderung der Starrheit des Objekts missachtet, entstehen Artefakte und Verzerrungseffekte im Modell. Die Genauigkeit und Visualisierungsqualität des NeRF-Renderings leiden stark unter dieser Missachtung.
Die orbitale Flugroutine konnte erfolgreich exportiert werden. Es war erkennbar, dass im Gegensatz zu DIM auch Punkte innerhalb der Baumkrone rekonstruiert wurden. Dies spricht wiederum für den Volumen-Rendering-Ansatz. Die Qualität dieser Rekonstruktionen wurde in dieser Ar-

beit nicht weiterverfolgt. Eine Grundvoraussetzung für die Bewertung der Baumkrone wäre ein starres Objekt, was aufgrund der Witterungsverhältnisse nicht gegeben war.

6.4 Einfluss der Aufnahmemethode (orbit vs. linear)

Wie bereits in mehreren Beispielen aufgezeigt, war der Einfluss der Aufnahmemethode ausschlaggebend für die Qualität der Rekonstruktion. Die initiale Erwartungshaltung wurde somit bestätigt: Orbitale Aufnahmen weisen eine höhere Genauigkeit auf als lineare. Dennoch war nicht klar, inwiefern die Aufnahmemethode die Ergebnisse innerhalb von NeRF und DIM beeinflusst. DIM hat gezeigt, dass das Fehlen von Schrägaufnahmen, je nach Anwendungsgebiet, eine unwesentliche Verschlechterung des Modells verursacht. Die absolute Genauigkeit wurde dabei nicht beeinträchtigt, jedoch die Vollständigkeit des Modells. Demnach hängt die Wahl der Aufnahmemethode primär vom gewünschten Ergebnis ab. Wenn das Ziel ein Oberflächenmodell ist, ist die lineare Variante ausreichend, da die Genauigkeit der Dachgeometrien nicht wesentlich von der eingeschränkten Schnittgeometrie beeinflusst wurde.
Im Falle von NeRF sind die Unterschiede zwischen dem orbitalen und dem linearen Flugplan maßgeblicher. Die Ergebnisse zeigten, dass der Einsatz von Schrägaufnahmen die Qualität der Rekonstruktion positiv beeinflusst. Das Eigenrauschen der Punktwolke, das mit der M3C2-Methode berechnet wurde, konnte dabei von 18 cm auf 16 cm gesenkt werden. Die Abbildungen 61 und 62) zeigten ebenfalls, dass die Rekonstruktion des Geländes durch die orbitale Flugplanung wesentlich verbessert wurde, da die Präzision der resultierenden Punkte erhöht wurde.
Dies stützt wiederum die Aussage, dass NeRFs von einer ausreichenden Schnittgeometrie profitieren. Eine vollständige Umrundung des Objekts mit Fotoaufnahmen wäre daher anzustreben. Die Wirtschaftlichkeit dieser Aufnahmekonfiguration wird an dieser Stelle nicht hinterfragt. Durch die Vielfalt der Aufnahmen entstehen gute Schnittbedingungen, die sich positiv auf die Vollständigkeit, Präzision und Genauigkeit der resultierenden Punktwolke auswirken.

6.5 Integration in praktische Workflows

Die Integration von NeRFs in den geodätischen Alltag ist zum jetzigen Zeitpunkt noch nicht gegeben. Die Problematiken hinsichtlich der Georeferenzierung und der Notwendigkeit einer Referenz hindern eine effiziente Nutzung. Da die Punktwolken in Bezug auf die Auflösung keine ausreichende Dichte garantieren, kann eine Registrierung über eine 7-Parameter-Transformation mit Passpunkten nicht durchgeführt werden. Dieser Umstand könnte möglicherweise behoben werden, indem Nahaufnahmen von Passpunkten gemacht werden. Dadurch würde die Auflösung der Passpunkte innerhalb der Punktwolke erhöht und in der Folge auch die Genauigkeit der Georeferenzierung verbessert.
Ein weiterer wichtiger Punkt für die Integration in den praktischen Workflow wäre die Georeferenzierung mit bereits vorhandenen Geotags der Bilder. Da NeRFs in ein individuelles Koordinatensystem überführt werden, geht der Bezug zum Ursprungskoordinatensystem verloren. Im Verlauf der Arbeit konnte die Skalierung rekonstruiert werden, jedoch traten Schwierigkeiten bei

der Rückführung der Rotationen und Verschiebungen auf. Dies erfordert weitere Untersuchungen.

Da NeRFs photorealistische Szenen rekonstruieren, erweisen sie sich als ideale Hilfsmittel zur Dokumentation und Visualisierung von Objekten und sollten in den praktischen Workflow integriert werden. Heutzutage werden oft einzelne Fotos von Gebäuden zu Dokumentationszwecken angefertigt. Diese stehen jedoch in keinem räumlichen Zusammenhang, was es Dritten häufig erschwert, den Bezug zwischen einer Fotoserie herzustellen. Ein weiterer Vorteil von NeRFs ist die geringe Speicheranforderung. Neben den Aufnahmen benötigt es nur wenige Megabyte an Daten, um ein NeRF zu repräsentieren. Im Gegensatz dazu sind Punktwolken aus ALS, TLS und DIM im Gigabyte-Bereich angesiedelt.

Besonders vorteilhaft sind NeRFs für die Visualisierung und Dokumentation räumlicher Gegebenheiten. Durch die fotorealistische Darstellung eines Objekts können Dokumentationsaufgaben, Schulungen, Planungen und Simulationen kostengünstig und effizient durchgeführt werden.

7 Schlussfolgerungen und Ausblick

Zusammenfassend ist zu sagen, dass NeRFs aktuell für den geodätischen Einsatz noch nicht ausgereift sind, jedoch hohes Potenzial für zukünftige Entwicklungen und Anwendungen in diesem Bereich bieten. Vielversprechend sind die Speichereffizienz und die Fähigkeit zur Erfassung von texturarmen sowie stark reflektierenden Oberflächen. Besonders in der Industrie müssen oft stark reflektierende Oberflächen dargestellt werden, bei denen TLS an seine Grenzen stößt. Für TLS wurden daher spezielle Sprays entwickelt, um die gerichtete Reflexion des emittierten Lichtstrahls zu minimieren. Diese sind jedoch kostenintensiv und der Aufwand vor einer Messkampagne ist beträchtlich.

Problematisch ist nach wie vor der hohe Rechenzeitbedarf von NeRFs. Die Erstellung eines NeRFs erfordert leistungsstarke Computerhardware. In diesem Sinne sind DIM-Verfahren effizienter und stellen weniger hohe Anforderungen an die Hardware des Anwenders. Die Berechnungen in dieser Arbeit wurden mit einer 2x 48 GB VRAM GPU (NVIDIA L40) der Technischen Universität Wien durchgeführt. Die Rechenzeit betrug für das größte Modell von Nerfacto 2–3 Stunden. Der Zugang zu solcher Hardware ist teuer und daher hauptsächlich institutionellen Einrichtungen vorbehalten. Bei Verwendung einer preiswerteren Alternative (NVIDIA Geforce RTX 3080TI) hätte sich die Rechenzeit auf etwa 7 Tage erhöht. Daher wäre es für zukünftige Forschungsarbeiten von Interesse, die Rechenintensität zu minimieren. Das Thema NeRFs wird von vielen wissenschaftlichen Gruppen verfolgt, und es konnten bereits Verbesserungen hinsichtlich der Effizienz erzielt werden. Mit Instant-NGP [61] haben Müller et al. (2022) Hash-Tabellen eingeführt, die hinsichtlich Geschwindigkeit und Effizienz neue Maßstäbe setzen. Dadurch ist es möglich, ein NeRF nahezu in Echtzeit zu erzeugen, wobei die Trainingszeit sowie die Renderzeit erheblich reduziert wurden.

Ein vielversprechender Ansatz, dem zukünftige Forschungsarbeiten Aufmerksamkeit schenken sollten, ist die Kombination von NeRFs mit DIM-Methoden. Die Kombination von DIM und NeRFs könnte die Schwächen beider Technologien ausgleichen und zu einer deutlich verbesserten Rekonstruktionsqualität führen, insbesondere in Bereichen, in denen texturarme oder stark reflektierende Oberflächen problematisch sind.

Ein weiteres Thema für nachfolgende Forschungsarbeiten ist die hohe Abhängigkeit der Schnittgeometrie von der Qualität der Rekonstruktion. Wie in dieser Arbeit gezeigt, spielt die Aufnahmeposition eine entscheidende Rolle für die Genauigkeit der Ergebnisse. Insbesondere bei linearen Flugbahnen kommt es zu deutlichen Abweichungen und Verlusten an Genauigkeit und Detailtreue. Die Probleme mit der Baum-Punktwolke haben verdeutlicht, dass dynamische Objekte, insbesondere unter wechselnden Umweltbedingungen wie Wind, eine große Herausforderung für NeRF-Modelle darstellen. Im Vergleich zu starren Objekten wie der Kapelle verschärfen sich die Probleme bei unzureichender Abdeckung und Bewegung. Der Exportfehler bei der linearen Flugmuster-Route zeigt, dass ein ausreichender Mix an Ansichten und stabilen Bedingungen notwendig ist, um eine qualitativ hochwertige Rekonstruktion zu ermöglichen. Für zukünftige Arbeiten wird empfohlen, bei der Arbeit mit dynamischen Objekten wie Baumgruppen auf ei-

ne Flugplanung mit mehr Blickwinkelvielfalt (wie im orbitalen Flug) zu setzen, um ähnliche Probleme zu vermeiden.

Unabhängig von der Schnittgeometrie sollten neue Flugroutinen getestet werden, die die Eigenschaften von NeRFs bestmöglich nutzen. Beispielhaft sei hier der Einsatz von „Point of Interest"-Flugoperationen erwähnt. Dabei umkreist die UAV während des Fluges einen vorab definierten Punkt. Dies wäre für NeRFs eine ideale Schnittgeometrie. Für topographische Anwendungen ist dies jedoch nicht umsetzbar, da die zu erfassenden Flächen weitläufiger als ein einzelnes Objekt sind. Ein Ansatz wäre, dass die UAV vorab definierte Linien oder Polygone im „Area of Interest"-Modus verfolgt. Dies würde eine Erweiterung des klassischen „Point of Interest"-Ansatzes darstellen. Klassische Verfahren wie z. B. Grid-Muster-Flüge, die für DIM kompatibel sind, verlieren hierbei an Relevanz. Darüber hinaus sollte in mehreren Flughöhen operiert werden. Die Arbeit hat gezeigt, dass durch die Koplanarität der Aufnahmen in bestimmten Höhen möglicherweise weniger Details erfasst werden, insbesondere bei feinen Strukturen. Dies führt zu einer verringerten Auflösung in Bereichen, in denen die Überlappung der Bilder begrenzt ist oder die Blickwinkel ungünstig sind. Die Koplanarität des Flugmusters hat im Gegensatz dazu bei DIM weniger Auswirkungen auf die Qualität der Rekonstruktion.

Insgesamt zeigt die Arbeit, dass NeRFs in naher Zukunft ein wichtiger Bestandteil der 3D-Rekonstruktion im geodätischen und industriellen Kontext werden könnten. Es bedarf jedoch weiterer Experimente, um diese neue Technologie bestmöglich zu nutzen. Die fortlaufende Forschung in diesem Bereich wird zeigen, inwieweit sich diese Methoden etablieren und möglicherweise zu einem neuen Standard für geodätische Anwendungen werden können.

Literatur

[1] K. Kraus, *Photogrammetrie: Band 1, Geometrische Informationen aus Photographien und Laserscanneraufnahmen*. Berlin, Boston: De Gruyter, 2004, ISBN: 9783110908039. DOI: 10.1515/9783110908039. Adresse: https://doi.org/10.1515/9783110908039.

[2] A. Pilar Valerga Puerta, R. Aletheia Jimenez-Rodriguez, S. Fernandez-Vidal und S. Raul Fernandez-Vidal, "Photogrammetry as an Engineering Design Tool," *Product Design*, Okt. 2020. DOI: 10.5772/INTECHOPEN.92998.

[3] S. Ullman, "The Interpretation of Structure from Motion," Okt. 1976.

[4] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, Jg. 60, Nr. 2, S. 91–110, Nov. 2004, ISSN: 09205691. DOI: 10.1023/B:VISI.0000029664.99615.94/METRICS. Adresse: https://link.springer.com/article/10.1023/B:VISI.0000029664.99615.94.

[5] H. Bay, T. Tuytelaars und L. Van Gool, "SURF: Speeded Up Robust Features," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, Jg. 3951 LNCS, S. 404–417, 2006, ISSN: 1611-3349. DOI: 10.1007/11744023{_}32. Adresse: https://link.springer.com/chapter/10.1007/11744023_32.

[6] P. F. Alcantarilla, A. Bartoli und A. J. Davison, "KAZE Features," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, Jg. 7577 LNCS, Nr. PART 6, S. 214–227, 2012, ISSN: 1611-3349. DOI: 10.1007/978-3-642-33783-3{_}16. Adresse: https://link.springer.com/chapter/10.1007/978-3-642-33783-3_16.

[7] P. F. Alcantarilla, J. Nuevo und A. Bartoli, "Fast explicit diffusion for accelerated features in nonlinear scale spaces," *BMVC 2013 - Electronic Proceedings of the British Machine Vision Conference 2013*, 2013. DOI: 10.5244/C.27.13.

[8] E. Rublee, V. Rabaud, K. Konolige und G. Bradski, "ORB: An efficient alternative to SIFT or SURF," *Proceedings of the IEEE International Conference on Computer Vision*, S. 2564–2571, 2011. DOI: 10.1109/ICCV.2011.6126544.

[9] S. Leutenegger, M. Chli und R. Y. Siegwart, "BRISK: Binary Robust invariant scalable keypoints," *Proceedings of the IEEE International Conference on Computer Vision*, S. 2548–2555, 2011. DOI: 10.1109/ICCV.2011.6126542.

[10] S. A. K. Tareen und Z. Saleem, "A comparative analysis of SIFT, SURF, KAZE, AKAZE, ORB, and BRISK," *2018 International Conference on Computing, Mathematics and Engineering Technologies: Invent, Innovate and Integrate for Socioeconomic Development, iCoMET 2018 - Proceedings*, Jg. 2018-January, S. 1–10, Apr. 2018. DOI: 10.1109/ICOMET.2018.8346440.

[11] D. G. Lowe, "Object recognition from local scale-invariant features," in *Proceedings of the IEEE International Conference on Computer Vision*, 1999. DOI: 10.1109/iccv.1999.790410.

[12] B. Triggs, P. F. McLauchlan, R. I. Hartley und A. W. Fitzgibbon, "Bundle Adjustment — A Modern Synthesis," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, Jg. 1883, S. 298–372, 2000, ISSN: 16113349. DOI: 10.1007/3-540-44480-7{_}21. Adresse: https://link.springer.com/chapter/10.1007/3-540-44480-7_21.

[13] F. Remondino, M. G. Spera, E. Nocerino, F. Menna, F. Nex und S. Gonizzi-Barsanti, "Dense image matching: Comparisons and analyses," *Proceedings of the DigitalHeritage 2013 - Federating the 19th Int'l VSMM, 10th Eurographics GCH, and 2nd UNESCO Memory of the World Conferences, Plus Special Sessions fromCAA, Arqueologica 2.0 et al.*, Jg. 1, S. 47–54, 2013. DOI: 10.1109/DIGITALHERITAGE.2013.6743712.

[14] H. Hirschmüller, "Accurate and efficient stereo processing by semi-global matching and mutua information," in *Proceedings - 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR 2005*, 2005, ISBN: 0769523722. DOI: 10.1109/CVPR.2005.56.

[15] *Disparity Map in Stereo Vision — Baeldung on Computer Science*. Adresse: https://www.baeldung.com/cs/disparity-map-stereo-vision.

[16] H. Hirschmüller, M. Buder und I. Ernst, "MEMORY EFFICIENT SEMI-GLOBAL MATCHING," *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, Jg. I-3, S. 371–376, Juli 2012, ISSN: 2194-9042. DOI: 10.5194/ISPRSANNALS-I-3-371-2012.

[17] M. Levoy und P. Hanrahan, "Light field rendering," in *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1996*, 1996, ISBN: 0897917464. DOI: 10.1145/237170.237199.

[18] S. J. Gortler, R. Grzeszczuk, R. Szeliski und M. F. Cohen, "The lumigraph," in *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1996*, 1996, ISBN: 0897917464. DOI: 10.1145/237170.237200.

[19] A. Davis, M. Levoy und F. Durand, "Unstructured light fields," in *Computer Graphics Forum*, 2012. DOI: 10.1111/j.1467-8659.2012.03009.x.

[20] M. Waechter, N. Moehrle und M. Goesele, "Let there be color! Large-scale texturing of 3D reconstructions," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2014. DOI: 10.1007/978-3-319-10602-1{_}54.

[21] C. Buehler, M. Bosse, L. McMillan, S. Gortler und M. Cohen, "Unstructured lumigraph rendering," in *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 2001*, 2001, ISBN: 158113374X. DOI: 10.1145/383259.383309.

[22] P. E. Debevec, C. J. Taylor und J. Malik, "Modeling and rendering architecture from photographs: A hybrid geometry- And image-based approach," in *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1996*, 1996, ISBN: 0897917464. DOI: 10.1145/237170.237191.

[23] W. Chen, J. Gao, H. Ling u. a., "Learning to predict 3D objects with an interpolation-based differentiable renderer," in *Advances in Neural Information Processing Systems*, 2019.

[24] K. Genova, F. Cole, A. Maschinot, A. Sarna, D. Vlasic und W. T. Freeman, "Unsupervised Training for 3D Morphable Model Regression," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2018, ISBN: 9781538664209. DOI: 10.1109/CVPR.2018.00874.

[25] S. Liu, W. Chen, T. Li und H. Li, "Soft rasterizer: A differentiable renderer for image-based 3D reasoning," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, ISBN: 9781728148038. DOI: 10.1109/ICCV.2019.00780.

[26] M. M. Loper und M. J. Black, "OpenDR: An approximate differentiable renderer," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2014, ISBN: 9783319105833. DOI: 10.1007/978-3-319-10584-0{_}11.

[27] T. M. Li, M. Aittala, F. Durand und J. Lehtinen, "Differentiable Monte Carlo ray tracing through edge sampling," in *SIGGRAPH Asia 2018 Technical Papers, SIGGRAPH Asia 2018*, 2018, ISBN: 9781450360081. DOI: 10.1145/3272127.3275109.

[28] M. Nimier-David, D. Vicini, T. Zeltner und W. Jakob, "Mitsuba 2: A retargetable forward and inverse renderer," *ACM Transactions on Graphics*, 2019, ISSN: 15577368. DOI: 10.1145/3355089.3356498.

[29] K. N. Kutulakos und S. M. Seitz, "Theory of shape by space carving," *International Journal of Computer Vision*, 2000, ISSN: 09205691. DOI: 10.1023/A:1008191222954.

[30] S. M. Seitz und C. R. Dyer, "Photorealistic scene reconstruction by voxel coloring," *International Journal of Computer Vision*, 1999, ISSN: 09205691. DOI: 10.1023/A:1008176507526.

[31] R. Szeliski und P. Golland, "Stereo matching with transparency and matting," *International Journal of Computer Vision*, 1999, ISSN: 09205691. DOI: 10.1109/iccv.1998.710766.

[32] T. Porter und T. Duff, "Compositing digital images," in *ACM Siggraph Computer Graphics*, Bd. 18, 1984, S. 253–259.

[33] T. Zhou, R. Tucker, J. Flynn, G. Fyffe und N. Snavely, "Stereo magnification: Learning view synthesis using multiplane images," *ACM Transactions on Graphics*, 2018, ISSN: 15577368. DOI: 10.1145/3197517.3201323.

[34] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi und R. Ng, "NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, Jg. 12346 LNCS, S. 405–421, März 2020, ISSN: 16113349. DOI: 10.1007/978-3-030-58452-8{_}24. Adresse: https://arxiv.org/abs/2003.08934v2.

[35] J. T. Kajiya und B. P. Von Herzen, "Ray tracing volume densities," *Computer Graphics*, Jg. 18, Nr. 3, S. 165–174, Jan. 1984. DOI: 10.1145/800031.808594. Adresse: https://typeset.io/papers/ray-tracing-volume-densities-htob4avd93.

[36] N. Max, "Optical Models for Direct Volume Rendering," *IEEE Transactions on Visualization and Computer Graphics*, Jg. 1, Nr. 2, S. 99–108, 1995, ISSN: 10772626. DOI: 10.1109/2945.468400.

[37] N. Rahaman, A. Baratin, D. Arpit u. a., *On the Spectral Bias of Neural Networks*, Mai 2019. Adresse: https://proceedings.mlr.press/v97/rahaman19a.html.

[38] J. T. Barron, B. Mildenhall, D. Verbin, P. P. Srinivasan und P. Hedman, "Mip-NeRF 360: Unbounded Anti-Aliased Neural Radiance Fields," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Jg. 2022-June, S. 5460–5469, Nov. 2021, ISSN: 10636919. DOI: 10.1109/CVPR52688.2022.00539. Adresse: https://arxiv.org/abs/2111.12077v3.

[39] J. T. Barron, B. Mildenhall, M. Tancik, P. Hedman, R. Martin-Brualla und P. P. Srinivasan, "Mip-NeRF: A Multiscale Representation for Anti-Aliasing Neural Radiance Fields," *Proceedings of the IEEE International Conference on Computer Vision*, S. 5835–5844, März 2021, ISSN: 15505499. DOI: 10.1109/ICCV48922.2021.00580. Adresse: https://arxiv.org/abs/2103.13415v3.

[40] Z. Wang, S. Wu, W. Xie, M. Chen und V. A. Prisacariu, "NeRF–: Neural Radiance Fields Without Known Camera Parameters," Feb. 2021. Adresse: https://arxiv.org/abs/2102.07064v4.

[41] R. Martin-Brualla, N. Radwan, M. S. Sajjadi, J. T. Barron, A. Dosovitskiy und D. Duckworth, "NeRF in the Wild: Neural Radiance Fields for Unconstrained Photo Collections," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, S. 7206–7215, Aug. 2021, ISSN: 10636919. DOI: 10.1109/CVPR46437.2021.00713. Adresse: https://arxiv.org/abs/2008.02268v3.

[42] F. Remondino, A. Karami, Z. Yan, G. Mazzacca, S. Rigon und R. Qin, "A Critical Analysis of NeRF-Based 3D Reconstruction," *Remote Sensing 2023, Vol. 15, Page 3585*, Jg. 15, Nr. 14, S. 3585, Juli 2023, ISSN: 2072-4292. DOI: 10.3390/RS15143585. Adresse: https://www.mdpi.com/2072-4292/15/14/3585/htm%20https://www.mdpi.com/2072-4292/15/14/3585.

[43] OpenStreetMap contributors, *OpenStreetMap*, 2024. Adresse: https://www.openstreetmap.org.

[44] WeatherSpark, *Durchschnittswetter am 17. April in Angern an der March, Österreich*, 2024. Adresse: https://de.weatherspark.com/d/81296/4/17/Durchschnittswetter-am-17.-April-in-Angern-an-der-March-%C3%96sterreich.

[45] Windfinder, *Windfinder - Wind & Wetter weltweit*, 2024. Adresse: https://www.windfinder.com/.

[46] Bundesamt für Eich- und Vermessungswesen (BEV), *Transformation zwischen MGI/GK und ETRS /UTM*, 2024. Adresse: https://www.bev.gv.at/dam/jcr:3344b22c-ed51-4564-a23c-4ec0518f49d2/Transformation_GK_MGI_UTM_ETRS89.pdf.

[47] F. Pessacg, F. Gómez-Fernández, M. Nitsche u. a., "Simplifying UAV-Based Photogrammetry in Forestry: How to Generate Accurate Digital Terrain Model and Assess Flight Mission Settings," *Forests 2022, Vol. 13, Page 173*, Jg. 13, Nr. 2, S. 173, Jan. 2022, ISSN: 1999-4907. DOI: 10.3390/F13020173. Adresse: https://www.mdpi.com/1999-4907/13/2/173/htm%20https://www.mdpi.com/1999-4907/13/2/173.

[48] Agisoft Metashape, *Agisoft Metashape: Agisoft Metashape*, 2024. Adresse: https://www.agisoft.com/.

[49] J. L. Schönberger und J.-M. Frahm, "Structure-from-Motion Revisited," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[50] Ahlers Enrico, *https://github.com/EnricoAhlers/agi2nerf*, 2022. Adresse: https://github.com/EnricoAhlers/agi2nerf.

[51] M. Tancik, E. Weber, E. Ng u. a., "Nerfstudio: A Modular Framework for Neural Radiance Field Development," *Proceedings - SIGGRAPH 2023 Conference Papers*, Feb. 2023. DOI: 10.1145/3588432.3591516. Adresse: http://arxiv.org/abs/2302.04264%20http://dx.doi.org/10.1145/3588432.3591516.

[52] *CloudCompare (version 2) [GPL software]*, 2024.

[53] N. Pfeifer, G. Mandlburger, J. Otepka und W. Karel, "OPALS – A framework for Airborne Laser Scanning data analysis," *Computers, Environment and Urban Systems*, Jg. 45, S. 125–136, 2014, ISSN: 0198-9715. DOI: https://doi.org/10.1016/j.compenvurbsys.2013.11.002. Adresse: https://www.sciencedirect.com/science/article/pii/S0198971513001051.

[54] *OPALS - Orientation and Processing of Airborne Laser Scanning data*. Adresse: https://opals.geo.tuwien.ac.at/html/stable/ModuleICP.html.

[55] P. Glira, N. Pfeifer, C. Briese und C. Ressl, "A correspondence framework for ALS strip adjustments based on variants of the ICP algorithm," *Photogrammetrie, Fernerkundung, Geoinformation*, 2015, ISSN: 14328364. DOI: 10.1127/pfg/2015/0270.

[56] *OPALS - Orientation and Processing of Airborne Laser Scanning data*. Adresse: https://opals.geo.tuwien.ac.at/html/stable/ModuleCell.html.

[57] *OPALS - Orientation and Processing of Airborne Laser Scanning data*. Adresse: https://opals.geo.tuwien.ac.at/html/stable/ModuleGridFeature.html.

[58] *OPALS - Orientation and Processing of Airborne Laser Scanning data*. Adresse: https://opals.geo.tuwien.ac.at/html/stable/ModuleNormals.html.

[59] C. J. Willmott und K. Matsuura, "Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance," *Climate Research*, Jg. 30, Nr. 1, S. 79–82, Dez. 2005, ISSN: 0936-577X. DOI: 10.3354/CR030079. Adresse: https://www.int-res.com/abstracts/cr/v30/n1/p79-82/.

[60] py4dgeo Development Core Team, "py4dgeo: library for change analysis in 4D point clouds," 2022. Adresse: https://github.com/3dgeo-heidelberg/py4dgeo.

[61] T. Müller, A. Evans, C. Schied und A. Keller, "Instant Neural Graphics Primitives with a Multiresolution Hash Encoding," *ACM Transactions on Graphics*, Jg. 41, Nr. 4, S. 102, Jan. 2022. DOI: 10.1145/3528223.3530127. Adresse: http://arxiv.org/abs/2201.05989%20http://dx.doi.org/10.1145/3528223.3530127.

[62] BEV, *APOS - Austrian Positioning Service*. Adresse: https://www.bev.gv.at/Services/Produkte/Grundlagenvermessung/APOS.html.

[63] Nerfstudio, *Data conventions - nerfstudio*. Adresse: https://docs.nerf.studio/quickstart/data_conventions.html.

[64] H. Huang, G. Tian und C. Chen, "Evaluating the Point Cloud of Individual Trees Generated from Images Based on Neural Radiance Fields (NeRF) Method," *Remote Sensing*, Jg. 16, Nr. 6, S. 967, März 2024, ISSN: 20724292. DOI: 10.3390/RS16060967/S1. Adresse: https://www.mdpi.com/2072-4292/16/6/967/htm%20https://www.mdpi.com/2072-4292/16/6/967.

[65] LumaLabs AI, *Linear_Kapelle - Created by @brz_1409 with Luma*, 2024. Adresse: https://lumalabs.ai/capture/eb7f65c4-5f48-468d-b902-ef2ed2b522f9.

[66] G. Mazzacca, A. Karami, S. Rigon, E. M. Farella, P. Trybala und F. Remondino, "Nerf for heritage 3d reconstruction," *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences - ISPRS Archives*, Jg. 48, Nr. M-2-2023, S. 1051–1058, Juni 2023, ISSN: 16821750. DOI: 10.5194/ISPRS-ARCHIVES-XLVIII-M-2-2023-1051-2023.

[67] Nerfstudio, *Nerfacto - nerfstudio*. Adresse: https://docs.nerf.studio/nerfology/methods/nerfacto.html.

[68] *OPALS - Orientation and Processing of Airborne Laser Scanning data*. Adresse: https://opals.geo.tuwien.ac.at/html/stable/ModuleSegmentation.html.

[69] S. Pahari, *Processing Drone Image Using AgiSoft Metashape and Comparative Analysis of different Digital Elevation Model (DEM)*, Juli 2023. DOI: 10.13140/RG.2.2.16060.86405.

[70] M. Rothermel, K. Wenzel, D. Fritsch und N. Haala, "SURE: Photogrammetric Surface Reconstruction From Imagery," Sep. 2012.

[71] D. N. Wood, D. I. Azuma, K. Aldinger u. a., "Surface Light Fields for 3D Photography," in *SIGGRAPH 2000 - Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*, 2000, ISBN: 1581132085. DOI: 10.1145/344779.344925.

[72] J. Shan, Z. Hu, P. Tao, L. Wang, Z. Shenman und S. Ji, "Toward a unified theoretical framework for photogrammetry," *Geo-spatial Information Science*, Juli 2020.

8 Anhang

8.1 Python Code

8.1.1 Segmentmanager - Extrahierung Ebenen

```
from __future__ import print_function
import matplotlib
import numpy as np
import os
import opals
from opals import pyDM
from opals import Segmentation, Import, Normals
import platform
import subprocess
import csv


# Define the preprocessing function
def preprocessing_steps(name):
    imp = Import.Import(inFile=name+".las")
    imp.run()

    normals = Normals.Normals(inFile=name+".odm", normalsAlg=opals.Types.
        NormalsAlgorithm.robustPlane, neighbours=16 ,searchMode=opals.Types.
        SearchMode.d3, searchRadius=0.5, storeMetaInfo=opals.Types.
        NormalsMetaInfo.medium, selMode=opals.Types.SelectionMode.quadrant)
    normals.run()

# Get all .las files in the current directory
directory = '.'  # Change this to the directory where your .las files are
    located
las_files = [f for f in os.listdir(directory) if f.endswith('.las')]

# Process each .las file
for las_file in las_files:
    name = os.path.splitext(las_file)[0]
    preprocessing_steps(name)

imp = Import.Import(inFile='tls_kapelle.las')
imp.run()
normals = Normals.Normals(inFile='tls_kapelle' + ".odm", normalsAlg=opals.Types.
    NormalsAlgorithm.robustPlane, neighbours=16,
                          searchMode=opals.Types.SearchMode.d3, searchRadius
                              =0.5,
                          storeMetaInfo=opals.Types.NormalsMetaInfo.medium,
                              selMode=opals.Types.SelectionMode.quadrant)
normals.run()

# Get all .odm files in the current directory
```

```
odm_files = [f for f in os.listdir('.') if f.endswith('.odm')]

# Check if the reference file exists
reference_filename = 'tls_kapelle.odm'
if reference_filename not in odm_files:
    raise FileNotFoundError("Reference file not found in the current directory.")
odm_files.remove(reference_filename)

# Select SegmentationMethod here:
mode = opals.Types.SegmentationMethod.planeExtraction
# mode = opals.Types.SegmentationMethod.condClustering

# Function to extract plane parameters
def extract_plane_parameters(filename):
    plane_params = []

    # Perform segmentation using the selected method
    mod = Segmentation.Segmentation()

    if mode == opals.Types.SegmentationMethod.planeExtraction:
        mod.method = opals.Types.SegmentationMethod.planeExtraction
        mod.planeExtraction.maxDist = 0.2
        mod.planeExtraction.maxSigma = 0.2
        mod.minSegSize = 200
        mod.searchRadius = 0.5
        mod.sort = opals.Types.SegmentSort.descendingPSize
        mod.inFile = filename

    elif mode == opals.Types.SegmentationMethod.condClustering:
        mod.method = opals.Types.SegmentationMethod.condClustering
        mod.minSegSize = 200
        mod.searchRadius = 0.5
        mod.filter = "Generic[z>250]"

    mod.run()

    # Get segmentation manager (gives access to the segment objects)
    segManager = mod.segments
    odm = pyDM.Datamanager.load(filename, True, False)
    pi = odm.getPointIndex()  # get point index object of odm

    print(f"The segmentation process has found {segManager.sizeSegments()} segment(s) in {filename}")

    # Iterate over all segments and output relevant segments properties
    for seg in segManager.getSegments():
        seg_id = seg.getID()
        num_points = seg.sizePoints()
```

```
85          cog_x, cog_y, cog_z = seg.getCoG().x, seg.getCoG().y, seg.getCoG().z
86          try:
87              planeParams = seg.getPlaneParams()
88              assert len(planeParams) == 5
89              a, b, c, d, sigma0 = planeParams
90              plane_params.append([filename, seg_id, num_points, cog_x, cog_y,
                    cog_z, a, b, c, d, sigma0])
91              print(f"Segment-Nr {seg_id}: {num_points} points; COG ({cog_x:.3f} /
                    {cog_y:.3f} / {cog_z:.3f})")
92              print(f"\tplane parameters: a={a:.4f} b={b:.4f} c={c:.4f} d={d:.4f}
                    (sigma0={sigma0:.3f})")
93              print("---------------")
94          except:
95              print(f"Segment-Nr {seg_id}: {num_points} points; COG ({cog_x:.3f} /
                    {cog_y:.3f} / {cog_z:.3f})")
96              print("No plane params")
97              print("---------------")
98
99      return plane_params
100
101 # Extract plane parameters from the reference file
102 reference_plane_params = extract_plane_parameters(reference_filename)
103 reference_csv_filename = reference_filename.replace('.odm', '_plane_parameters.
        csv')
104 with open(reference_csv_filename, mode='w', newline='') as file:
105     writer = csv.writer(file)
106     writer.writerow(['Filename', 'Segment_ID', 'Num_Points', 'COG_X', 'COG_Y', '
            COG_Z', 'A', 'B', 'C', 'D', 'Sigma0'])
107     writer.writerows(reference_plane_params)
108 print(f"Reference plane parameters have been saved to '{reference_csv_filename}'
        ")
109
110 # Extract plane parameters from all other files
111 all_planes = [reference_plane_params]
112 for filename in odm_files:
113     print(filename)
114     plane_params = extract_plane_parameters(filename)
115     csv_filename = filename.replace('.odm', '_plane_parameters.csv')
116     with open(csv_filename, mode='w', newline='') as file:
117         writer = csv.writer(file)
118         writer.writerow(['Filename', 'Segment_ID', 'Num_Points', 'COG_X', 'COG_Y
                ', 'COG_Z', 'A', 'B', 'C', 'D', 'Sigma0'])
119         writer.writerows(plane_params)
120     print(f"Plane parameters for {filename} have been saved to '{csv_filename}'"
            )
121     all_planes.append(plane_params)
122
123
124 # Save all p arameters from all files into a single CSV file
```

```
with open('all_planes_parameters.csv', mode='w', newline='') as file:
    writer = csv.writer(file)
    writer.writerow(['Filename', 'Segment_ID', 'Num_Points', 'COG_X', 'COG_Y', '
        COG_Z', 'A', 'B', 'C', 'D', 'Sigma0'])
    for planes in all_planes:
        writer.writerows(planes)
print("All plane parameters from all files have been saved to '
    all_planes_parameters.csv'")
```

8.1.2 Metrics-Datei

```
def filter_correspondences(point_cloud1, point_cloud2, threshold=np.inf):
    tree = cKDTree(point_cloud2)
    distances, indices = tree.query(point_cloud1, k=1, workers=1)
    valid_mask = distances <= threshold
    filtered_point_cloud1 = point_cloud1[valid_mask]
    filtered_neighbors = point_cloud2[indices[valid_mask]]
    return filtered_point_cloud1, filtered_neighbors, distances[valid_mask]

def compute_rmse(point_cloud1, point_cloud2, threshold=np.inf):
    point_cloud1, point_cloud2, distances = filter_correspondences(point_cloud1,
        point_cloud2, threshold)
    if distances.size == 0:
        return float('nan'), distances
    rmse = np.sqrt(np.mean(distances ** 2))
    print("rmse", np.round(rmse, 2))
    return rmse, distances

def compute_mae(point_cloud1, point_cloud2, threshold=np.inf):
    point_cloud1, point_cloud2, distances = filter_correspondences(point_cloud1,
        point_cloud2, threshold)
    if distances.size == 0:
        return float('nan')
    mae = np.mean(distances)
    print("mae", np.round(mae, 2))
    return mae

def compute_std_dev(point_cloud1, point_cloud2, threshold=np.inf):
    point_cloud1, point_cloud2, distances = filter_correspondences(point_cloud1,
        point_cloud2, threshold)
    if distances.size == 0:
        return float('nan')
    std_dev = np.std(distances)
    print("std_dev", np.round(std_dev, 2))
    return std_dev

def compute_modified_hausdorff_distance(point_cloud1, point_cloud2):
    tree1 = cKDTree(point_cloud1)
    tree2 = cKDTree(point_cloud2)
```

```
    d1, _ = tree1.query(point_cloud2, k=1, workers=1)
    print(len(d1))
    d2, _ = tree2.query(point_cloud1, k=1, workers=1)
    print(len(d2))
    if d1.size == 0 or d2.size == 0:
        return float('nan')
    mhd_1_to_2 = np.mean(d1)
    mhd_2_to_1 = np.mean(d2)
    modified_hausdorff_distance = max(mhd_1_to_2, mhd_2_to_1)
    print("modified_hausdorff_distance", np.round(modified_hausdorff_distance,
        2))
    return modified_hausdorff_distance

def compute_mean_distance(point_cloud1, point_cloud2, threshold=np.inf):
    point_cloud1, point_cloud2, distances = filter_correspondences(point_cloud1,
         point_cloud2, threshold)
    if distances.size == 0:
        return float('nan')
    mean_distance = np.mean(distances)
    print("mean_distance", np.round(mean_distance, 2))
    return mean_distance

def percentage_within_tolerance(point_cloud1, point_cloud2, tolerance, threshold
    =np.inf):
    point_cloud1, point_cloud2, distances = filter_correspondences(point_cloud1,
         point_cloud2, threshold)
    if distances.size == 0:
        return float('nan')
    within_tolerance = np.sum(distances <= tolerance) / len(distances) * 100
    print("within_tolerance", np.round(within_tolerance, 2))
    return within_tolerance

def compute_mean_error(point_cloud1, point_cloud2, threshold=np.inf):
    point_cloud1, point_cloud2, _ = filter_correspondences(point_cloud1,
        point_cloud2, threshold)
    if point_cloud1.size == 0 or point_cloud2.size == 0:
        return (float('nan'), float('nan'), float('nan'))
    mean_error = np.mean(point_cloud1 - point_cloud2, axis=0)
    print("mean_error", np.round(mean_error, 2))
    return tuple(mean_error)

def compute_mad(point_cloud1, point_cloud2, threshold=np.inf):
    point_cloud1, point_cloud2, distances = filter_correspondences(point_cloud1,
         point_cloud2, threshold)
    if distances.size == 0:
        return float('nan')
    mad = np.median(np.abs(distances - np.median(distances)))
    print("mad", np.round(mad, 2))
    return mad
```

```python

def compute_nndr(point_cloud1, point_cloud2, threshold=np.inf):
    tree = cKDTree(point_cloud2)
    distances, _ = tree.query(point_cloud1, k=2, workers=1)
    if distances.size == 0:
        return float('nan')
    valid_mask = distances[:, 0] <= threshold
    if np.sum(valid_mask) == 0:
        return float('nan')
    nndr = np.mean(distances[valid_mask, 0] / distances[valid_mask, 1])
    print("nndr", np.round(nndr, 2))
    return nndr

def compute_density_difference(point_cloud1, point_cloud2, radius=1.0, threshold=np.inf):
    tree1 = cKDTree(point_cloud1)
    tree2 = cKDTree(point_cloud2)
    density1 = [len(d) for d in tree1.query_ball_point(point_cloud1, r=radius, workers=1)]
    density2 = [len(d) for d in tree2.query_ball_point(point_cloud2, r=radius, workers=1)]
    if len(density1) == 0 or len(density2) == 0:
        return float('nan')
    density_diff = np.abs(np.mean(density1) - np.mean(density2))
    print("density_diff", np.round(density_diff, 2))
    return density_diff

def compute_chamfer_distance(point_cloud1, point_cloud2, threshold=np.inf):
    tree1 = cKDTree(point_cloud1)
    tree2 = cKDTree(point_cloud2)
    distances1, _ = tree1.query(point_cloud2, k=1, workers=1)
    distances2, _ = tree2.query(point_cloud1, k=1, workers=1)
    valid_distances1 = distances1[distances1 <= threshold]
    valid_distances2 = distances2[distances2 <= threshold]
    if valid_distances1.size == 0 or valid_distances2.size == 0:
        return float('nan')
    chamfer_distance = np.mean(valid_distances1) + np.mean(valid_distances2)
    print("chamfer_distance", np.round(chamfer_distance, 2))
    return chamfer_distance

def compute_density_overlap(point_cloud1, point_cloud2, radius=1.0, threshold=np.inf):
    tree1 = cKDTree(point_cloud1)
    tree2 = cKDTree(point_cloud2)
    density1 = tree1.query_ball_point(point_cloud1, r=radius, workers=1)
    density2 = tree2.query_ball_point(point_cloud2, r=radius, workers=1)
    overlap_set1 = set([tuple(point_cloud1[i]) for sublist in density1 for i in sublist])
```

```
    overlap_set2 = set([tuple(point_cloud2[i]) for sublist in density2 for i in
        sublist])
    if len(overlap_set1) == 0 or len(overlap_set2) == 0:
        return float('nan')
    overlap = len(overlap_set1 & overlap_set2)
    overlap_percentage = (overlap / min(len(point_cloud1), len(point_cloud2))) *
         100
    print("overlap_percentage", np.round(overlap_percentage, 2))
    return overlap_percentage

def print_stats(point_cloud1, point_cloud2, name, threshold=np.inf):
    point_cloud1, point_cloud2, valid_distances = filter_correspondences(
        point_cloud1, point_cloud2, threshold)
    if valid_distances.size == 0:
        print(f"No valid correspondences found for {name}")
        return
    stats = {
        'min': round(np.min(valid_distances), 2),
        'max': round(np.max(valid_distances), 2),
        '25_percentile': round(np.percentile(valid_distances, 25), 2),
        '50_percentile': round(np.percentile(valid_distances, 50), 2),
        '75_percentile': round(np.percentile(valid_distances, 75), 2),
        'median': round(np.median(valid_distances), 2),
    }
    print(f'Statistics of valid distances: {name}')
    for key, value in stats.items():
        print(f"{key}: {value}")
    print("---------------------------------------")

def compute_surface_roughness(point_cloud, search_radius):
    tree = cKDTree(point_cloud)
    roughness_values = []
    for point in point_cloud:
        indices = tree.query_ball_point(point, r=search_radius, workers=1)
        neighbors = point_cloud[indices]
        if len(neighbors) > 1:
            distances = np.linalg.norm(neighbors - point, axis=1)
            roughness = np.var(distances)
            roughness_values.append(roughness)
    return roughness_values

def compute_curvature(point_cloud, search_radius):
    tree = cKDTree(point_cloud)
    curvature_values = []
    for point in point_cloud:
        indices = tree.query_ball_point(point, r=search_radius, workers=1)
        neighbors = point_cloud[indices]
        if len(neighbors) > 1:
```

```
            covariance_matrix = np.cov(neighbors - neighbors.mean(axis=0),
                rowvar=False)
            eigenvalues = np.linalg.eigvalsh(covariance_matrix)
            curvature = eigenvalues.min() / eigenvalues.sum()
            curvature_values.append(curvature)
    return curvature_values

def compute_normal_vectors(point_cloud, search_radius):
    tree = cKDTree(point_cloud)
    normals = []
    for point in point_cloud:
        indices = tree.query_ball_point(point, r=search_radius, workers=1)
        neighbors = point_cloud[indices]
        if len(neighbors) > 1:
            centroid = np.mean(neighbors, axis=0)
            cov_matrix = np.cov((neighbors - centroid).T)
            eigvals, eigvecs = np.linalg.eigh(cov_matrix)
            normal = eigvecs[:, np.argmin(eigvals)]
            if np.dot(normal, point - centroid) < 0:
                normal = -normal
            normal /= np.linalg.norm(normal)
            normals.append(normal)
        else:
            normals.append([0.0, 0.0, 0.0])
    return np.array(normals)
```

8.1.3 M3C2-Datei

```

import py4dgeo
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import matplotlib.colors as mcolors

tls_path = "las/TLS.las"
orbit_NERF_path = "las/orbit_Kapelle_NERF - Cloud.las"
linear_NERF_path = "las/linear_Kapelle_NERF - Cloud.las"
orbit_DIM_path = "las/orbit_Kapelle_DIM - Cloud.las"
linear_DIM_path = "las/linear_Kapelle_DIM - Cloud.las"

tls_cloud = py4dgeo.read_from_las(tls_path).cloud
orbit_NERF_cloud = py4dgeo.read_from_las(orbit_NERF_path).cloud
linear_NERF_cloud = py4dgeo.read_from_las(linear_NERF_path).cloud
orbit_DIM_cloud = py4dgeo.read_from_las(orbit_DIM_path).cloud
linear_DIM_cloud = py4dgeo.read_from_las(linear_DIM_path).cloud

epoch_tls = py4dgeo.Epoch(tls_cloud)

```

```
epochs_to_compare = {
    "orbit_NERF": py4dgeo.Epoch(orbit_NERF_cloud),
    "linear_NERF": py4dgeo.Epoch(linear_NERF_cloud),
    "orbit_DIM": py4dgeo.Epoch(orbit_DIM_cloud),
    "linear_DIM": py4dgeo.Epoch(linear_DIM_cloud),
}

epoch_tls.build_kdtree()

statistics_list = []
registration_errors = []

for name, epoch in epochs_to_compare.items():

    m3c2 = py4dgeo.M3C2(
        epochs=(epoch_tls, epoch),
        corepoints=epoch_tls.cloud[::],
        normal_radii=(0.5,),
        cyl_radii=(0.5,),
        max_distance=(0.5),
        registration_error=(0.0),
    )

    m3c2_distances_first, uncertainties_first = m3c2.run()

    registration_error = np.nanstd(m3c2_distances_first)
    registration_errors.append(registration_error)
    print(f"{name} - Estimated Registration Error: {registration_error:.3f} m")

    m3c2 = py4dgeo.M3C2(
        epochs=(epoch_tls, epoch),
        corepoints=epoch_tls.cloud[::],
        normal_radii=(0.5,),
        cyl_radii=(0.5,),
        max_distance=(0.5),
        registration_error=(registration_error),
    )

    m3c2_distances, uncertainties = m3c2.run()

    mean_distance = np.nanmean(m3c2_distances)
    median_distance = np.nanmedian(m3c2_distances)
    std_dev_distance = np.nanstd(m3c2_distances)
    min_distance = np.nanmin(m3c2_distances)
    max_distance = np.nanmax(m3c2_distances)

    significant_changes = np.where(abs(m3c2_distances) > uncertainties["
        lodetection"], True, False)
```

```
    percent_significant_change = np.sum(significant_changes) / len(
        m3c2_distances) * 100

    statistics_list.append({
        "Dataset": name,
        "Mean Distance (m)": mean_distance,
        "Median Distance (m)": median_distance,
        "Std Dev Distance (m)": std_dev_distance,
        "Min Distance (m)": min_distance,
        "Max Distance (m)": max_distance,
        "Registration Error (m)": registration_error,
        "Percent Significant Change (%)": percent_significant_change
    })

    valid_distances = m3c2_distances[~np.isnan(m3c2_distances)]

    fig = plt.figure(figsize=(14, 18))

    ax1 = fig.add_subplot(3, 2, 1, projection="3d")
    ax2 = fig.add_subplot(3, 2, 2, projection="3d")
    ax3 = fig.add_subplot(3, 2, 3, projection="3d")
    ax4 = fig.add_subplot(3, 2, 4, projection="3d")

    ax5 = fig.add_subplot(3, 2, (5, 6))

    d = ax1.scatter(
        tls_cloud[:, 0],
        tls_cloud[:, 1],
        tls_cloud[:, 2],
        c=m3c2_distances,
        cmap="seismic_r",
        vmin=-0.5,
        vmax=0.5,
        s=1,
    )
    ax1.set_title(f"{name}: M3C2 Distances (Registration Error: {
        registration_error:.2f} m)")
    ax1.set_xlabel("Easting [m]")
    ax1.set_ylabel("Northing [m]")
    ax1.set_zlabel("Z [m]")
    plt.colorbar(d, format=("%.2f"), label="M3C2 Distance [m]", ax=ax1, shrink
        =0.5, pad=0.15)

    directions = m3c2.directions()
    dz = ax2.scatter(
        tls_cloud[:, 0],
        tls_cloud[:, 1],
        tls_cloud[:, 2],
        c=directions[:, 2],
```

```
        cmap="cool",
        s=1,
    )
    ax2.set_title(f"{name}: Normal Directions (Z-axis)")
    ax2.set_xlabel("Easting [m]")
    ax2.set_ylabel("Northing [m]")
    ax2.set_zlabel("Z [m]")
    plt.colorbar(dz, format=("%.2f"), label="Normal Z [0,1]", ax=ax2, shrink
        =0.5, pad=0.15)

    l = ax3.scatter(
        tls_cloud[:, 0],
        tls_cloud[:, 1],
        tls_cloud[:, 2],
        c=uncertainties["lodetection"],
        cmap="plasma",
        s=1,
        vmin=0.0,
        vmax=0.5,
    )
    ax3.set_title(f"{name}: Level of Detection")
    ax3.set_xlabel("Easting [m]")
    ax3.set_ylabel("Northing [m]")
    ax3.set_zlabel("Z [m]")
    plt.colorbar(
        l, format=("%.2f"), label="Level of Detection [m]", ax=ax3, extend="max"
            , shrink=0.5, pad=0.15
    )

    ax4.scatter(
        tls_cloud[significant_changes][:, 0],
        tls_cloud[significant_changes][:, 1],
        tls_cloud[significant_changes][:, 2],
        label="Significant Change",
        c="red",
        s=1,
    )
    ax4.set_title(f"{name}: Significant Changes (Registration Error: {
        registration_error:.2f} m)")
    ax4.set_xlabel("Easting [m]")
    ax4.set_ylabel("Northing [m]")
    ax4.set_zlabel("Z [m]")
    ax4.legend()

    ax5.hist(m3c2_distances, bins=128, range=(-0.5, 0.5), color="blue", alpha
        =0.7)
    ax5.set_title(f"{name}: M3C2 Distances Histogram & Boxplot")
    ax5.set_xlabel("M3C2 Distance [m]")
    ax5.set_ylabel("Frequency")
```

```
    ax5.grid(True)

    ax5b = ax5.twinx()
    if len(valid_distances) > 0:
        ax5b.boxplot(valid_distances, vert=False, patch_artist=True, boxprops=
            dict(facecolor="orange", alpha=0.5))
        ax5b.set_yticks([])
    else:
        print(f"No valid M3C2 distances to plot for {name}.")

    plt.tight_layout()
    plt.savefig(f"{name}_m3c2_combined.png", dpi=300)
    plt.show()

common_registration_error = np.median(registration_errors)
print(f"Common Registration Error for all datasets: {common_registration_error
    :.3f} m")

third_statistics_list = []

for name, epoch in epochs_to_compare.items():
    m3c2 = py4dgeo.M3C2(
        epochs=(epoch_tls, epoch),
        corepoints=epoch_tls.cloud[::],
        normal_radii=(0.5,),
        cyl_radii=(0.5,),
        max_distance=(0.5),
        registration_error=(common_registration_error),
    )

    m3c2_distances, uncertainties = m3c2.run()

    mean_distance = np.nanmean(m3c2_distances)
    median_distance = np.nanmedian(m3c2_distances)
    std_dev_distance = np.nanstd(m3c2_distances)
    min_distance = np.nanmin(m3c2_distances)
    max_distance = np.nanmax(m3c2_distances)

    significant_changes = np.where(abs(m3c2_distances) > uncertainties["
        lodetection"], True, False)
    percent_significant_change = np.sum(significant_changes) / len(
        m3c2_distances) * 100

    third_statistics_list.append({
        "Dataset": name,
        "Mean Distance (m)": mean_distance,
        "Median Distance (m)": median_distance,
        "Std Dev Distance (m)": std_dev_distance,
        "Min Distance (m)": min_distance,
```

```
        "Max Distance (m)": max_distance,
        "Percent Significant Change (%)": percent_significant_change
    })

    print(f"{name} - Third M3C2 run completed using common registration error.")

    fig, axs = plt.subplots(2, 2, figsize=(14, 14), subplot_kw={"projection": "3d"})
    (ax1, ax2), (ax3, ax4) = axs

    d = ax1.scatter(
        tls_cloud[:, 0],
        tls_cloud[:, 1],
        tls_cloud[:, 2],
        c=m3c2_distances,
        cmap="seismic_r",
        vmin=-.5,
        vmax=.5,
        s=1,
    )
    ax1.set_title(f"{name}: M3C2 Distances (Registration Error: {common_registration_error:.2f} m)")
    plt.colorbar(d, format=("%.2f"), label="M3C2 Distance [m]", ax=ax1, shrink=0.5, pad=0.15)

    directions = m3c2.directions()
    dz = ax2.scatter(
        tls_cloud[:, 0],
        tls_cloud[:, 1],
        tls_cloud[:, 2],
        c=directions[:, 2],
        cmap="cool",
        s=1,
    )
    ax2.set_title(f"{name}: Normal Directions (Z-axis)")
    plt.colorbar(dz, format=("%.2f"), label="Normal Z [0,1]", ax=ax2, shrink=0.5, pad=0.15)

    l = ax3.scatter(
        tls_cloud[:, 0],
        tls_cloud[:, 1],
        tls_cloud[:, 2],
        c=uncertainties["lodetection"],
        cmap="plasma",
        s=1,
        vmin=0.0,
        vmax=0.5,
    )
```

```
    ax3.set_title(f"{name}: Level of Detection (Registration Error: {
        common_registration_error:.2f} m)")
    plt.colorbar(
        l, format=("%.2f"), label="Level of Detection [m]", ax=ax3, extend="max"
            , shrink=0.5, pad=0.15
    )

    ax4.scatter(
        tls_cloud[significant_changes][:, 0],
        tls_cloud[significant_changes][:, 1],
        tls_cloud[significant_changes][:, 2],
        label="Significant Change",
        c="red",
        s=1,
    )
    ax4.legend()
    ax4.set_title(f"{name}: Significant Changes (Registration Error: {
        common_registration_error:.2f} m)")

    for ax_set in axs:
        for ax in ax_set:
            ax.set_xlabel("Easting [m]")
            ax.set_ylabel("Northing [m]")
            ax.set_zlabel("Z [m]")
            ax.set_aspect("auto")
            ax.view_init(elev=30.0, azim=120.0)

    plt.tight_layout()
    plt.savefig(f"{name}_m3c2_results_common.png", dpi=300)
    plt.show()

stats_df = pd.DataFrame(statistics_list)
third_stats_df = pd.DataFrame(third_statistics_list)

stats_df.to_csv("m3c2_statistics_results.csv", index=False)
third_stats_df.to_csv("m3c2_third_run_statistics.csv", index=False)

print("Original M3C2 Statistics:")
print(stats_df)
print("\nThird M3C2 Run Statistics (Using Common Registration Error):")
print(third_stats_df)
```

8.1.4 Ebenen Matching Funktion - Toleranzen

```

tls_kapelle_normals = compute_normal_vectors(tls_kapelle_data)


def angle_between_vectors(vec1, vec2):
```

```
    dot_product = np.dot(vec1, vec2)
    cos_angle = np.clip(dot_product / (np.linalg.norm(vec1) * np.linalg.norm(
        vec2)), -1.0, 1.0)
    angle_rad = np.arccos(cos_angle)
    angle_deg = np.degrees(angle_rad)
    return angle_deg


def find_matches(distance_tolerance, angle_tolerance_degrees):
    matches = []
    max_distance = data[['COG_X', 'COG_Y', 'COG_Z']].apply(lambda row: norm(row)
        , axis=1).max()
    max_sigma0_diff = data['Sigma0'].max() - data['Sigma0'].min()

    for filename in other_filenames:
        matched_indices_other_files = set()
        file_data = data[data['Filename'] == filename]
        file_coords = file_data[['COG_X', 'COG_Y', 'COG_Z']].values

        distances, indices_coords = kdtree.query(file_coords,
            distance_upper_bound=distance_tolerance)

        for idx in range(len(file_data)):
            if distances[idx] < float('inf') and idx not in
                matched_indices_other_files:
                matched_row = tls_kapelle_data.iloc[indices_coords[idx]]
                row = file_data.iloc[idx]

                normal_ref = np.array([matched_row['A'], matched_row['B'],
                    matched_row['C']])
                normal_current = np.array([row['A'], row['B'], row['C']])
                angle_diff = angle_between_vectors(normal_ref, normal_current)

                if angle_diff <= angle_tolerance_degrees:
                    matched_indices_other_files.add(idx)
                    sigma0_diff = np.round(abs(row['Sigma0'] - matched_row['
                        Sigma0']), 2)


                    norm_distance = distances[idx] / max_distance
                    norm_angle_diff = angle_diff / 180  # Maximaler
                        Winkelunterschied ist 180 Grad
                    norm_sigma0_diff = sigma0_diff / max_sigma0_diff


                    weight = np.round(0.4 * norm_distance + 0.4 *
                        norm_angle_diff + 0.2 * norm_sigma0_diff, 10)

                    matches.append({
```

```
                        'Filename': filename,
                        'Segment_ID': row['Segment_ID'],
                        'Matched_Segment_ID': matched_row['Segment_ID'],
                        'Distance': distances[idx],
                        'Angle_Difference': angle_diff,
                        'Sigma0_Difference': sigma0_diff,
                        'Weight': weight,
                    })
    return matches
```

8.1.5 Normalabstand, Winkeldifferenz und Plane Fit in Python

```

def fit_plane_to_points(points):

    centroid = np.mean(points, axis=0)
    centered_points = points - centroid

    U, S, Vt = np.linalg.svd(centered_points)
    normal_vector = Vt[2, :]

    A, B, C = normal_vector
    D = -np.dot(normal_vector, centroid)

    distances_to_plane = np.abs(np.dot(centered_points, normal_vector))
    NormalSigma0 = np.std(distances_to_plane)
    return {'A': A, 'B': B, 'C': C, 'D': D, 'Centroid': centroid, 'NormalSigma0'
        : NormalSigma0}


def compute_normal_distance(plane1, plane2):

    n1 = np.array([plane1['A'], plane1['B'], plane1['C']])
    n2 = np.array([plane2['A'], plane2['B'], plane2['C']])
    n1_normalized = n1 / norm(n1)
    n2_normalized = n2 / norm(n2)

    cos_theta = np.dot(n1_normalized, n2_normalized)
    cos_theta = np.clip(cos_theta, -1.0, 1.0)
    theta = np.arccos(abs(cos_theta))  # Absolutwert verwenden
    angle_difference = np.degrees(theta)

    if angle_difference > 90:
        angle_difference = 180 - angle_difference

    point = plane1['Centroid']
    D2 = plane2['D']
    numerator = abs(plane2['A'] * point[0] + plane2['B'] * point[1] + plane2['C'
        ] * point[2] + D2)
```

```
    denominator = norm([plane2['A'], plane2['B'], plane2['C']])
    normal_distance = numerator / denominator
    return normal_distance, angle_difference
```

8.2 Gerätespezifikationen

8.2.1 DJI Zenmuse P1

Parameter	**Wert**
Sensor	Vollformat, 35,9 × 24 mm
Auflösung	45 MP (8192 × 5460)
Pixelgröße	4,4 µm
Objektivhalterung	DJI DL-Mount
Kompatible Objektive	24/35/50 mm
Verschluss	Mechanischer Verschluss
Verschlusszeit	1/2000 - 1/8 Sekunde
Belichtungssteuerung	Programmautomatik, Zeitautomatik, Blendenautomatik, Manuell
ISO-Bereich	100 - 25600
Bildformate	JPEG, RAW
Gimbal	3-Achsen, Stabilisierung auf ±0,01°
Gewicht	789 g
Betriebstemperatur	-20°C bis 50°C
Speicher	microSD Karte
Weitere Funktionen	TimeSync 2.0, Unterstützung für RTK, High Precision GNSS

Tabelle 19: Technische Daten der DJI Zenmuse P1 Kamera

8.2.2 DJI Matrice 350 RTK

Parameter	Wert
Abmessungen	810 × 670 × 430 mm (ohne Propeller)
Gewicht	Ca. 6,3 kg (ohne Nutzlast)
Maximales Abfluggewicht	9 kg
Maximale Flugzeit	55 Minuten
Maximale Fluggeschwindigkeit	23 m/s
Maximale Windgeschwindigkeit	12 m/s
Maximale Flughöhe	5000 m
Betriebstemperatur	-20°C bis 50°C
Satellitensysteme	GPS, GLONASS, BeiDou, Galileo
RTK Genauigkeit	Horizontal: 1 cm + 1 ppm, Vertikal: 1,5 cm + 1 ppm
Positionierungssystem	RTK, TimeSync 2.0
Nutzlast	Max. 2,7 kg
Gimbal Unterstützung	3-Achsen Stabilisierung
Fernsteuerung	DJI RC Plus
Videoübertragung	O3 Enterprise, bis zu 15 km
Intelligente Funktionen	Waypoint 2.0, AI Spot-Check, PinPoint, Smart Oblique Capture
Sicherheitsmerkmale	Hinderniserkennung in 6 Richtungen, ADS-B Empfänger

Tabelle 20: Technische Daten der DJI Matrice 350 RTK

Nomenclature

AKAZE Accelerated KAZE

ALS Airborne Laser Scanning

APOS Austrian Positioning System

BRISK Binary Robust Invariant Scalable Keypoints

C2C Cloud to Cloud

C2M Cloud to Mesh

COG Center of Gravity

DIM Dense Image Matching

DTM Digital Terrain Model

EPOSA Echtzeit Positionierung Austria

FCNN fully convolutional neural network

GCP Ground Control Point

GNSS globale Navigationssatellitensysteme

GPU Graphics Processing Unit

GSD Ground Sampling Distance

KAZE Nichtlineare Merkmalsextraktion

KDE Kernel Density Estimation

KNN K-Nearest-Neighbour

LPIPS Learned Perceptual Image Patch Similarity

M3C2 Multiscale Model-to-Model Cloud Comparison

MAD Median Absolute Deviation

MAE Mean Absolute Error

MH Modifizierte Hausdorff-Distanz

MVS Multi View Stereo

NeRF Neural Radiance Fields

NNDR Nearest Neighbor Distance Ratio

ORB Oriented FAST and Rotated BRIEF

PCA Principal Component Analysis

PSNR Peak Signal-to-Noise Ratio

RMSE Root Mean Square Error

RTK Real Time Kinematic

SfM Structure from Motion

SGM Semi-Global-Matching

SIFT Scale-Invariant Feature Transform

SSIM Structural Similarity Index

Std Dev Standardabweichung

SURF Speeded-Up Robust Features

Tie points Verknüpfungspunkte

TLS Terrestrischen Laserscanner

TSDF Truncated signed distance function

UAV Unmanned Aerial Vehicle