Entwicklung eines Informationssystems für Laserscannerdaten mit open source Software

Diplomarbeit

zur Erlangung des akademischen Grades
Magister der Naturwissenschaften

eingereicht von Bernhard Höfle

am Institut für Geographie der Leopold-Franzens-Universität Innsbruck

bei o.Univ.Prof. Dr. Johann Stötter

Innsbruck, Jänner 2005

„Alle Dinge sind dazu da,
damit sie uns Bilder werden in irgendeinem Sinn.“

Rainer Maria Rilke

Vorwort

Diese Diplomarbeit wäre ohne die Hilfe und Unterstützung vieler Personen nie in dieser Form zustande gekommen. Ich möchte mich an dieser Stelle ganz offiziell für die finanzielle Unterstützung bei meinem Vater, bei den Partnern des alpS-Projekts A 2.5, bei der Universität Innsbruck und beim Institut für Geographie bedanken. Diese Arbeit basiert auf den Laserscannerdaten des Landes Vorarlberg, die dem Diplomanden kostenlos zur Verfügung gestellt wurden.

Ohne Fleiß, kein Preis.
Ohne Freunde, kein Leben.

Ein Dank gebührt all jenen, dich mich während des Studiums ausgebildet und unterstützt haben. Namentlich erwähnen und ganz besonders bedanken möchte ich mich bei

Johann Stötter, für die Betreuung der Diplomarbeit und das große Vertrauen,
Thomas Geist, für den unermüdlichen Einsatz, aus uns gute Wissenschaftler zu machen,
Armin Heller, für das immer offene Ohr und die Einführung in die Welt des GIS,
Klaus Förster und **Bernd Öggl**, für die Geduld und Hilfsbereitschaft,
Martin Rutzinger und **Michael Fecht**, für die vielen Diskussionen und schönen Abende an der Uni,
der **NatGef-Gruppe**, für die interessanten Brown-Bags,
meiner **Mutter**, meinem **Vater** und meinen fünf **Schwestern** und

meiner Freundin **Moni**!

Außer Konkurrenz:

dem Erfinder der Post-Its,
und meinem armen Computer, der härter als alle anderen zu kämpfen hatte.

Kurzfassung

Die Diplomarbeit *„Entwicklung eines Informationssystems für Laserscannerdaten mit open source Software*“ widmet sich der vektorbasierten Auswertung geographischer Daten.
Das flugzeuggestützte Laserscanning liefert als Ergebnis eine große Menge von Punkten, die räumlich sowie zeitlich verortet sind. Zusätzlich zu den geometrischen Informationen besitzt jeder Laserpunkt ein Attribut, das die Stärke des reflektierten Signals (Intensität) angibt.
In der Geographie wird das Laserscanning erst in den letzten Jahren verstärkt als Verfahren für die Geländeaufnahme angewandt. Die flächendeckende Aufnahme der Erdoberfläche bietet neue Möglichkeiten für die Analyse von räumlichen Phänomenen. Die hohe räumliche Auflösung (mehrere Punkte/m^2) und die bekannte Genauigkeit der Messpunkte (Dezimeterbereich) sind ausschlaggebend für die erfolgreiche Anwendung dieser Technologie.
Gerade die große Datendichte der Laserscannerdaten stellt eine Herausforderung für vektorbasierte Verfahren, die keine Reduzierung der Punkte anstreben, dar. Die Diplomarbeit präsentiert ein **Konzept eines Systems für die Verwaltung, Verarbeitung und Visualisierung dieser Daten**, das ausschließlich mit freier Software umgesetzt wird. Die Zusammenführung eines Geographischen Informationssystems (GIS), einer räumlichen Datenbank und einer Statistiksoftware ermöglicht eine breite Palette an Analysemöglichkeiten. Im Zentrum des *Informationssystems* steht die Datenbank (PostgreSQL/PostGIS), die eine direkte Verarbeitung der gespeicherten Daten an Ort und Stelle erlaubt. Die breiten Schnittstellen zwischen den Systemkomponenten verhindern zeitraubende Datentransfers und Fehler bei der Datenübertragung. Mit der Einbindung des Geographischen Informationssystems GRASS wird die Konvertierung der Vektordaten in das Rasterdatenformat möglich. Die Statistiksoftware R bietet zahlreiche Möglichkeiten der statistischen Auswertung und der Visualisierung.
Das Informationssystem wird mit einem Datensatz im Vorarlberger Rheintal (Hohenems) getestet. Das Untersuchungsgebiet weist viele verschiedene Landnutzungsklassen auf. Die Ergebnisse eignen sich somit für die Übertragung auf andere Gebiete mit ähnlichen Nutzungen.
Mit Hilfe einer einfachen Geländemodellierung – aus drei benachbarten Laserpunkten wird eine Ebene konstruiert – wird die Scangeometrie für jeden Laserpunkt berechnet. Die Kenntnis über die Aufnahmegeometrie soll dazu beitragen, das Intensitätssignal besser zu verstehen, um in weiterer Folge eine geometrische Korrektur mit einfachen statistischen Modellen zu ermöglichen.
Die Visualisierung der Punktwolke in zwei- und dreidimensionalen Profilschnitten sowie die Einbindung von Software, die eine virtuelle dreidimensionale Ansicht der Punkte erlaubt, erleichtert das Verstehen des Laserscannings und das Entwickeln von Auswertestrategien für die Klassifizierung der Punkte.

Inhaltsverzeichnis

Vorwort I

Kurzfassung II

Inhaltsverzeichnis III

1 Einleitung **1**

1.1 alpS Forschungsprojekt A 2.5 1
1.2 Problemstellung und Ziel der Arbeit 2
1.3 Aktueller Forschungsstand 3
1.4 Aufbau der Arbeit 4

2 Grundlagen des flugzeuggestützten Laserscannings **5**

2.1 Begriffserklärung 5
2.2 Aufnahmeprinzip 5
2.3 Anwendungsbereiche 9
2.4 Zusammenfassung 9

3 Einführung in das Untersuchungsgebiet **11**

3.1 Geographische Lage 11
3.2 Situation 12
3.2.1 Topographie 12
3.2.2 Siedlungsstruktur 13
3.2.3 Vegetation 15
3.2.4 Naturgefahren 17
3.3 Laserscannerbefliegungen 18
3.4 Zusammenfassung 20

4 Datenerfassung und Datengrundlagen **21**

4.1 Einleitung 21
4.2 Erfassung und Vorverarbeitung von Laserscannerdaten 21
4.2.1 Projektplanung 21
4.2.2 Metadaten zur Befliegung 25
4.2.3 Flugzeugposition und -ausrichtung 26
4.2.4 Laserpunkte 28
4.2.5 Signalintensität 29
4.3 Vorliegende Datenstruktur 30
4.3.1 Anordnung der Datenfiles 30
4.3.2 Festlegung eines Standards 33
4.4 Zusammenfassung 33

5 Entwicklung des Informationssystems 35

5.1 Einleitung 35
5.2 Methodik 35
5.3 Systemkomponenten 39
5.4 Entwicklungsumgebung 40
5.4.1 Hardware 40
5.4.2 Betriebssystem 40
5.4.3 Datenbanksystem 41
5.4.3.1 PostgreSQL 41
5.4.3.2 PostGIS 45
5.4.4 Geographisches Informationssystem GRASS 48
5.4.5 Statistiksoftware R 49
5.4.6 Verwendete Sprachen 50
5.4.6.1 SQL 50
5.4.6.2 PL/pgSQL 50
5.4.6.3 PL/R 51
5.4.6.4 Python 52
5.4.6.5 Tkinter 53
5.5 Zusammenfassung 53

6 Datenverwaltung und -speicherung 55

6.1 Einleitung 55
6.2 Struktur der Datenbank 55
6.2.1 Entitäten 56
6.2.2 Erstellen der Tabellen 59
6.2.3 Räumliche Daten - Geometriespalten 62
6.3 Indexierung 64
6.4 Zusammenfassung 65

7 Datenimport 66

7.1 Methodik 66
7.2 Aufbau der Relationen 70
7.3 Berechnung erster Informationen 72
7.3.1 Attribute der Befliegung 72
7.3.2 Attribute des Flugstreifens 73
7.4 Performancetests 75
7.4.1 INSERT versus COPY 75
7.4.2 Import der Rohdaten 75
7.5 Zusammenfassung 76

8 Datenverarbeitung **77**

8.1 Einleitung 77
8.2 Absolute zeitliche Verortung 77
8.3 Aufnahmegeometrie 79
8.3.1 Gründe für die Rekonstruktion der Aufnahmegeometrie 79
8.3.2 Rekonstruktion der Flugzeugposition für jeden Laserpunkt 79
8.3.3 Länge des Laservektors (*range*) 84
8.3.4 Geländemodellierung 86
8.3.4.1 Methodik 86
8.3.4.2 Neigung (*slope*) 91
8.3.4.3 Exposition (*aspect*) 93
8.3.4.4 Einfallswinkel (*angle of incidence*) 96
8.3.4.5 Fläche des Lichtkegels am Boden (*footprint*) 99
8.3.4.6 Relative Flughöhe (*height above ground level*) 102
8.3.4.7 Zusammenfassung 103
8.4 Statistische Auswertung 104
8.4.1 Einleitung 104
8.4.2 Deskriptive Statistik 104
8.4.3 Zusammenhangsmaße 109
8.4.3.1 Korrelationskoeffizient 109
8.4.3.2 Regressionsgerade 111
8.5 Berechnung von Rasterdaten 113
8.5.1 Methodik 113
8.5.2 Intensitätsraster 115
8.6 Zusammenfassung 119

9 Datenexport und Visualisierung **121**

9.1 Methodik 121
9.2 Direkter Datenbankexport 122
9.2.1 PostgreSQL 122
9.2.2 PostGIS 123
9.2.3 Profile mit PL/R 123
9.3 Indirekter Datenbankexport 129
9.3.1 GRASS 129
9.3.2 QGIS 129
9.3.3 R 130
9.3.3.1 XGobi 130
9.3.3.2 GGobi 131
9.4 Graphische Benutzeroberfläche 132
9.5 Zusammenfassung 134

10 Evaluierung und Ausblick **135**
10.1 Diskussion 135
10.2 Ausblick 140
Literaturverzeichnis 142
Abbildungsverzeichnis 149
Tabellenverzeichnis 152
Appendix 154
A Befliegungskampagne Vorarlberg 154
B Quelltextbeispiele 155
B.1 Performancetests Datenimport 155
B.2 Datenbankfunktionen 156
C Logfile des Datenimports 159
D Statistische Datenplots 163
D.1 Intensität 163
D.2 Länge des Laservektors 165
D.3 Neigung 167
D.4 Exposition 169
D.5 Einfallswinkel 169
D.6 Fläche des Lichtkegels am Boden 170
D.7 Relative Flughöhe 171
E Profillinien 172
E.1 Übersicht 172
E.2 Zweidimensionale Profile 173
E.3 Dreidimensionale Profile 181
F Intensitätsraster 186
G Lizenzen 191
G.1 PostgreSQL 191
G.2 GRASS und R 191
G.3 Python 197

1 Einleitung

1.1 alpS-Forschungsprojekt A 2.5

Die Diplomarbeit *„Entwicklung eines Informationssystems für Laserscannerdaten mit open source Software“* wird im Rahmen des alpS-Forschungsprojekts A 2.5 *„Analyse von Laserscannerdaten im Hinblick auf die Bestimmung von Oberflächeneigenschaften“* durchgeführt. Das Kplus-Zentrum **alpS Zentrum für Naturgefahren Management**, das in Innsbruck angesiedelt ist, beschäftigt sich schwerpunktmäßig mit dem Management von Naturgefahren. Das Projekt A 2.5, das auf eine Laufzeit von drei Jahren ausgerichtet ist (mit Beginn Juli 2004), hat sich zum Ziel gesetzt, die Möglichkeiten des *Laserscannings* im Zusammenhang mit der Modellierung von Naturgefahrenprozessen zu untersuchen und zu bewerten.

Project Aims	*The main project goal is to determine surface characteristics from laser scanner data utilising both the digital elevation information and the signal intensity information. Emphasis will be laid on the qualitative assessment and, if possible, the quantification of surface properties (e.g. surface roughness as a key parameter in hydrological modelling) and the temporal change of these properties. The analysis will comprise a great variety of surface types with regard to alpine natural hazard management. Standard GIS and remote sensing tools will be used as well as more refined methods (e.g. object oriented analysis tools). The results will be validated and calibrated by integration of other datatypes (e.g. field data, high resolution optical data).*
Expected Results	• *An information system for laser scanner data which is flexible and easy to adapt to user requirements.* • *Methods for combined analysis of laser scanner data (DTM and signal intensity) and other data types based on GIS and remote sensing tools.* • *Classification and, if possible, quantification of surface properties and their temporal changes.* • *Strategies for utilising laser scanner data for specific purposes in modern alpine natural hazard management.*

Tab. 1.1: Auszug aus der Projektbeschreibung des alpS-Forschungsprojekts A 2.5 (www.alps-gmbh.com oder http://lisa.uibk.com)

Ein wesentlicher Bestandteil bzw. ein Ziel des Projektes stellt die Entwicklung eines Datenverwaltungs- und Datenverarbeitungssystems dar (Tab. 1.1).

1.2 Problemstellung und Ziel der Arbeit

Das flugzeuggestützte Laserscanning liefert eine nahezu flächendeckende Erfassung des gescannten Gebiets mit einer Genauigkeit, die im Meterbereich und darunter liegt. Das gängigste Produkt einer Laserscannerbefliegung ist eine sehr große Anzahl (mehrere Punkte / m²) von gemessenen Punktkoordinaten, die eine zusätzliche Information über die Intensität des reflektierten Signals enthalten (x, y, z, I). Das Grundproblem der systematischen, flächendeckenden Aufnahme ist, dass man nicht weiß, was die Koordinaten der Punkte repräsentieren.
Die traditionelle Vorgehensweise besteht darin, diese Punkte als Stützpunkte für die Berechnung von Rasterdaten heranzuziehen. Die Methoden der rasterbasierten Auswertungen sind weit fortgeschritten und erlauben die Trennung von Gelände und Objekten, die sich darauf befinden. Das Resultat sind Digitale Geländemodelle (ohne Objekte) und Digitale Oberflächenmodelle (mit Objekten). Diese Vorgehensweise liefert brauchbare Ergebnisse, weil es die Genauigkeit und die Möglichkeiten der bisherigen Methoden der großflächigen Oberflächenerfassung übertrifft und weil es die ursprüngliche Datenmenge der Messpunkte um vieles verringert. Bei der Überführung der Laserpunkte in das Rasterdatenmodell treten unweigerlich Verluste in der Genauigkeit auf. Ferner enthalten die einzelnen Laserpunkte sowie ihre Verteilung im Raum, zusätzliche Informationen, die zur Beschreibung der aufgenommenen Erdoberfläche nützlich sein können.
Am Markt gibt es bis zum jetzigen Zeitpunkt kein befriedigendes kommerzielles Softwarepaket, das die Verwaltung der Laserpunkte im Vektordatenmodell, die Visualisierung dieser Punktwolken, die statistische Auswertung und die Entwicklung und Einbindung eigener Auswertestrategien in einem einzigen System zusammenführt. Der Begriff „Informationssystem“, der im Titel der Diplomarbeit genannt wird, steht repräsentativ für ein System, das die Datenverwaltung, die Datenverarbeitung und die Visualisierung der Daten vereint.

Ziel dieser Diplomarbeit ist die Entwicklung eines flexiblen Systems, das den schnellen und einfachen Zugriff auf die Laserscannerdaten und deren deskriptive Informationen ermöglicht. Auf die Transparenz und Nachvollziehbarkeit der Vorgehensweise sowie die Evaluierung der Ergebnisse wird Wert gelegt.
Die ausschließliche Verwendung von freier, nichtkommerzieller Software steht im Vordergrund der technischen Umsetzung. Die offene Struktur (*open source*) dieser Software eröffnet neue Möglichkeiten und erlaubt jedem die Auswertestrategien dieser Diplomarbeit ohne Kosten selbst umzusetzen.
Die Diplomarbeit konzentriert sich räumlich auf Vorarlberg, das in Österreich eine Vorreiterrolle in der flächendeckenden Landeserfassung mit Laserscanning einnimmt. Zur besseren Kontrolle der Ergebnisse und zur Reduktion der Rechenzeiten beschränkt sich die Arbeit auf ein 0.25 km^2 großes Untersuchungsgebiet im Vorarlberger Rheintal.
Die Tabelle 1.2 gibt die einzelnen Arbeitsschritte und ihre zeitliche Einordnung im Verlauf der

Diplomarbeit wieder.

Tätigkeit	**2004**					
	J	A	S	O	N	D
Literaturrecherche	X	X	X	X	X	X
Konzeption / Methodenentwicklung	X	X				
Auswahl der Testgebiete	X	X				
Datenbeschaffung	X	X				
Preprocessing der Daten		X				
Festlegung der Infrastruktur (Hard-, Software, Arbeitsplatz)	X					
Setup der Infrastruktur	X					
Festlegung der Schnittstellen für das Interface		X				
Aufbau der Datenbank		X	X			
Teilmodul Import		X	X			
Teilmodul Informationsgewinnung		X	X	X	X	
Teilmodul Export				X	X	X
Performancetests und Tuning				X	X	
Graphical User Interface (GUI) Erstellung				X	X	
Vergleich und Evaluierung der Ergebnisse					X	X
Tagungen	X			X		
Geländebegehungen			X			X

Tab. 1.2: Zeitplan der Diplomarbeit.

1.3 Aktueller Forschungsstand

Die Technologie und die Auswertestrategien im Bereich des flugzeuggestützten Laserscannings haben sich in den letzten 10 Jahren ständig verbessert. Dies hängt damit zusammen, dass immer mehr zivile Institutionen und Forschungseinrichtungen sich mit dieser Technologie beschäftigen und dass die Anzahl der Firmen, die Laserscanning kommerziell anbieten immer weiter steigt. Die Zusammenarbeit zwischen Forschung und Wirtschaft trägt sicher dazu bei, dass diese rasante Entwicklung in den nächsten Jahren weiter fortschreitet.

ACKERMANN (1999), HYYPÄÄ et. al. (2004), WEHR & LOHR (1999) und WEVER & LINDENBERGER (1999) geben einen guten Überblick über die Entwicklungen in den letzten 10 Jahren und führen in das Prinzip dieser Technologie ein. LINDENBERGER (1993) zeigt in seiner Dissertation auf, dass die Laserprofilmessung und die daraus abgeleiteten Geländemodelle auch in bisher schwer zu erreichenden bzw. zu erfassenden Gebieten (v.a. Wald) Erfolg bringen. Die darauf folgenden Schwerpunkte der Forschung lagen bei der Erstellung von flächigen Gelände- und Oberflächenmodellen. Einerseits wurde die Extraktion von Objekten, die bei der Geländemodellerstellung entfernt werden müssen, verstärkt in

Stadtgebieten (z.B. BRIESE et. al. 2001) und andererseits in bewaldeten Gebieten durchgeführt (z.B. KRAUS & PFEIFER 1998).

Zukünftige Forschungsschwerpunkte werden sich inhaltlich verstärkt mit der Auswertung des Intensitätssignals (LUTZ 2003), multitemporalen Analysen (GEIST et. al. 2003) und der Analyse in der Punktwolke (VOSSELMAN et. al. 2004) befassen. Die technischen Entwicklungen zeichnen einen Wandel von der Zwei-Puls-Aufnahme in Richtung *full-waveform* vor (WAGNER et. al. 2004). Ferner wird die Kombination von Laserscannerdaten und optischen Daten (z.B. digitale Luftbilder) stärker forciert und es werden noch bessere Analyseergebnisse erwartet (PERSSON et. al. 2004).

1.4 Aufbau der Arbeit

Die Diplomarbeit ist in 10 Kapitel gegliedert. Die Kapitel 1 bis 4 sollen den Leser in die Thematik des Laserscannings, in das Untersuchungsgebiet und die zur Verfügung stehenden Laserscannerdaten einführen. Das Kapitel 5 befasst sich mit der Grundkonzeption des Informationssystems und den einzelnen Komponenten, sowie mit der verwendeten Software. Die Kapitel 6 und 7 beschreiben die Konzeption der Datenverwaltung und den Datenimport. Die Kapitel 8 und 9 widmen sich der Datenanalyse und der Visualisierung der Ergebnisse. Das abschließende Kapitel 10 gibt einen Überblick über die gewonnen Erfahrungen, bewertet die Ergebnisse und gibt einen Ausblick, wie eine Weiterentwicklung des Informationssystems gestaltet werden könnte. Am Ende jedes Kapitels findet sich eine kurze Zusammenfassung, die die wichtigsten Punkte des jeweiligen Kapitels hervorhebt.

2 Grundlagen des flugzeuggestützten Laserscannings

2.1 Begriffserklärung

Zur Thematik des Laserscannings sind viele verschiedene Begriffe in der Literatur zu finden. In der englischsprachigen Literatur werden wieder ganz andere Begriffe bevorzugt wie in der deutschsprachigen. Ein klarer Überblick kann nicht gegeben werden, da es bis jetzt keine offizielle Terminologie gibt. Wer sich mit Laserscanning beschäftigen will, sollte aber auf jeden Fall die nachfolgend genannten Begriffe beherrschen, um Missverständnisse zu vermeiden.

Das Laserscanning ist in der Fernerkundung seit vielen Jahren unter dem Begriff *LIDAR* (LIght Detection And Ranging) im Einsatz. Das Akronym *LADAR* (LAser Detection And Ranging) präzisiert den Begriff *LIDAR* und gibt an, dass als Lichtquelle ein LASER (Light Amplification by Stimulated Emission of Radiation) verwendet wird (KRAUS 2004, 449; WEHR & LOHR 1999, 69).

Der Begriff „*flugzeuggestütztes Laserscanning*“ impliziert, dass der Laser an einem Flugzeug angebracht ist. Im Englischen wird der allgemeinere Begriff *Airborne Laser Scanning* (ALS) herangezogen, bei dem es offen bleibt, um welches Fluggefährt es sich handelt. Das Gegenstück zum ALS ist das terrestrische Laserscanning, wo das Aufnahmesystem vom Boden aus agiert.

Die ersten Systeme beschränkten sich auf die Aufnahme von Messpunkten entlang einer Linie in Flugrichtung. Dieses Verfahren wird *laser profiling* genannt (ECHELMEYER et. al. 1996, LINDENBERGER 1993). Beim *Laserscanning* wird die Erdoberfläche mit dem Ziel einer flächendeckenden Aufnahme mit einem Laserstrahl „gescannt“. Ein Scanvorgang erfasst sehr viele einzelne Punkte. Wird eine hohe Punktdichte erreicht, spricht man von einer flächenhaften oder flächigen Aufnahme.

2.2 Aufnahmeprinzip

Das Laserscanning zählt zu den aktiven Fernerkundungsverfahren. Der Sensor sendet aktiv einen Laserstrahl aus und empfängt die von der Oberfläche zurückgestreute Strahlung. Passive Systeme sind reine Empfänger, wie z.B. eine Luftbildkamera, die das von der Erdoberfläche reflektierte Sonnenlicht aufzeichnet. Daraus lässt sich auch schon ein großer Vorteil des Laserscannings ableiten. Als aktives System ist es nicht von den Lichtverhältnissen (Tag/Nacht, Wolken, diffuses Licht, Sonnenstand, Wolkenschatten, Abschattungseffekte, usw.) zur Zeit der Befliegung abhängig (WAGNER et. al. 2003).

Die Messeinrichtung befindet sich auf einer Flugplattform (z.B. Flugzeug oder Helikopter). Ein flugzeuggestütztes Laserscanning-System setzt sich aus drei Hauptkomponenten

zusammen: einem Laserscanner, einem „Global Positioning System“ (GPS) und einem Inertialen Navigationssystem (INS).

Der **Laserscanner** sendet Laserstrahlen, die von einem Spiegel senkrecht zur Flugrichtung abgelenkt werden, im Wellenlängenbereich des nahen Infrarots (hauptsächlich 800-1100 nm) zur Erdoberfläche und registriert den in Richtung Flugzeug zurückgestreuten Teil. Dabei wird die Zeit gemessen, die der Strahl für die Strecke Flugzeug-Oberfläche-Flugzeug benötigt (=Laufzeit). Aus der Zeit lässt sich der zurückgelegte Weg – ist gleich Gruppengeschwindigkeit mal Zeit – berechnen. Die Gruppengeschwindigkeit ist aufgrund von atmosphärischen Einflüssen um etwa 0,03 % geringer als die Lichtgeschwindigkeit im Vakuum (KRAUS 2004, 449). Die Distanz Flugzeug-Erdoberfläche entspricht der Hälfte des zurückgelegten Weges (siehe Formel 4.2 in Kapitel 4).

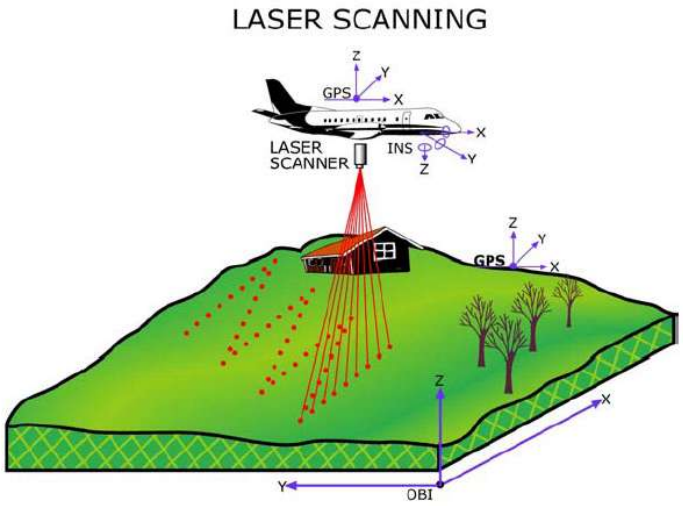


Abb. 2.1: Schematische Darstellung des flugzeuggestützten Laserscannings (AKEL et. al., 2003).

Die meisten Objekte auf der Erdoberfläche reflektieren den Laserstrahl diffus, d.h. eine Reflexion findet in alle Richtungen im Raum statt und somit auch zum Sensor. Gewisse Objekte, z.B. sehr glatte Oberflächen wie Glas oder Metall (z.B. Autodach), sind spektrale Reflektoren. Der Laserstrahl wird gerichtet reflektiert (Spiegelung) und erreicht den Sensor nicht. Das Reflexionsvermögen (Reflektivität) hängt von der Oberflächenbeschaffenheit ab. Generell kann gesagt werden, dass glatte und helle Objekte stärker reflektieren (WEVER 1999, 13; KRAUS 2004, 451). Gemessene Werte der Reflektivität natürlicher Oberflächen werden von WEHR & LOHR (1999, 74) und im Internet (www.riegl.com) aufgelistet. Die Reflexionseigenschaften von Wasserflächen sind je nach Einfallswinkel sehr unterschiedlich. Senkrecht auftreffende Strahlen werden fast völlig absorbiert und die Messung geht verloren. Je flacher der Einfallswinkel, desto größer ist die Chance, dass ein Teil der Strahlungsenergie zum Sensor zurückgestreut wird (KRAUS 2004, 451).
Der Laserstrahl weist einen bestimmten Öffnungswinkel (Strahldivergenz) auf und sollte

daher viel mehr als Lichtkegel verstanden werden. Bei einer Flughöhe von 1000 Metern über dem Boden und einer Strahldivergenz von 0,25 mrad ergibt sich ein Durchmesser von ca. 25 cm am Boden (WEVER 1999, 13). Dieser Abtastfleck wird in der Literatur meist als *footprint* bezeichnet.

Die räumliche Ausdehnung des Laserstrahls bringt die Möglichkeit von Mehrfachreflexionen mit sich. Das bedeutet, dass ein Lichtkegel auf dem Weg zum Boden viele Objekte treffen kann und somit mehrfach reflektiert wird (Abb. 2.2). Standard ist derzeit die Messung der ersten und letzten Reflexion, auch *first* und *last pulse* oder *echo* genannt. Die letzte Reflexion hat die höchste Wahrscheinlichkeit, die Reflexion des Strahls am Boden (=Bodenpunkt) zu sein. Das Verhältnis der Bodenpunkte zur Gesamtanzahl der Punkte wird als Durchdringungsrate (*penetration rate*) bezeichnet (LINDENBERGER 1993, 96). Je größer die Anzahl der Bodenpunkte, desto bessere Geländemodelle können erstellt werden. Das Laserscanning kann also bis zu einem gewissen Grade durch die Vegetation „hindurchblicken“ und das darunterliegende Gelände aufnehmen. Je dichter die Vegetation, desto weniger Bodenpunkte sind zu erwarten. Direkte Objekttreffer (z.B. Hausdach) oder Bodentreffer (z.B. Straße, Parkplatz) haben nur eine Reflexion.

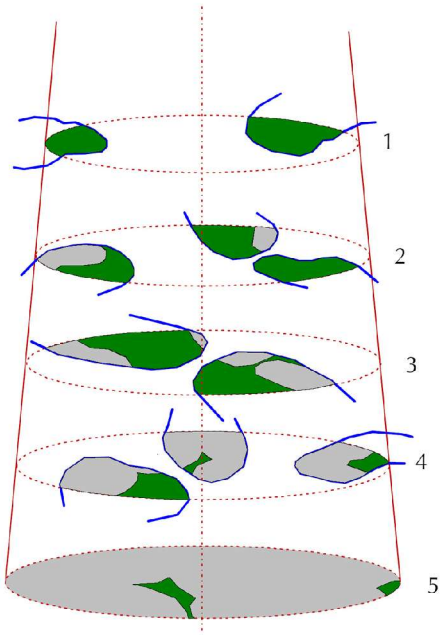


Abb. 2.2: Mehrfachreflexion (5 Reflexionen) bei Auftreffen eines Laserstrahls bzw. Lichtkegels auf Vegetation (KATZENBEISSER 2004, 11).

Die zweite Komponente des Laserscanning-Aufnahmesystems ist ein **Global Positioning System (GPS)**, das die genaue Position des Flugzeugs im Raum in sehr kurzen Zeitabständen erfasst. Gleichzeitig wird am Boden (Abb. 2.1) eine oder mehrere Referenzmessungen gemacht, um damit später die systematischen Fehler in der GPS-Messung zu korrigieren (differentielle Korrektur).

Die dritte Komponente ist das **Inertiale Navigationssystem (INS)**, auch *inertial measurement unit (IMU)* genannt, das die genaue Orientierung des Flugzeugs im Raum mit

den drei Winkeln Rollen, Nicken und Driften speichert (siehe Kapitel 4).
Mit diesen Informationen, der genauen Lage und Orientierung des Laserscanners im Raum, dem Ablenkwinkel des Laserstrahls am Spiegel und der zurückgelegten Distanz, kann der Vektor Laserscanner-Oberfläche für jede registrierte Reflexion bestimmt und somit die genaue Lage (x-,y-,z-Koordinate) der Reflexionsstelle angegeben werden.
Die zusätzlich zu jedem Punkt gespeicherte Signalintensität gibt Auskunft über die Oberflächenbeschaffenheit (Rauigkeit, Feuchte, Objektgröße, usw.), was den Reflexionseigenschaften des Objekts im Wellenlängenbereich des nahen Infrarots entspricht.
Die Höhengenauigkeit der Laserscannermessung hängt in erster Linie von der Genauigkeit der Laser-Entfernungsmesseinrichtung und der GPS-Höhenpositionierung ab. Die Lagegenauigkeit ist direkte Folge der GPS-Lagepositionierung, der Genauigkeit der INS-Messung und der Genauigkeit der Registrierung des Ablenkwinkels des Laserstrahls (KRAUS 2004, 454). Die Flughöhe (größere Flughöhe - größerer *footprint*) und das Relief (Neigung und Exposition) der Erdoberfläche spielen sowohl bei der Höhen- als auch bei der Lagegenauigkeit eine Rolle. Die Genauigkeit liegt im Durchschnitt im Zehnerzentimeterbereich. Mehr zu Genauigkeiten der Laserscannermessung sind in KRAUS (2004) und in LINDENBERGER (1993) zu erfahren.
Der Vollständigkeit halber ist zu erwähnen, dass das oben beschriebene Messprinzip für so genannte „*pulsed lasers*“, d.h. es wird mit Laserimpulsen gearbeitet, gilt. Eine andere Möglichkeit ist ein „*continuous wave*“ Laserscanner. Beim *continuous wave* Laserscanner wird über den Vergleich der Phasenlage der ausgesandten und der empfangenen intensitätsmodulierten Strahlung auf die Entfernung geschlossen (Abb. 2.3; vgl. WAGNER 2003).

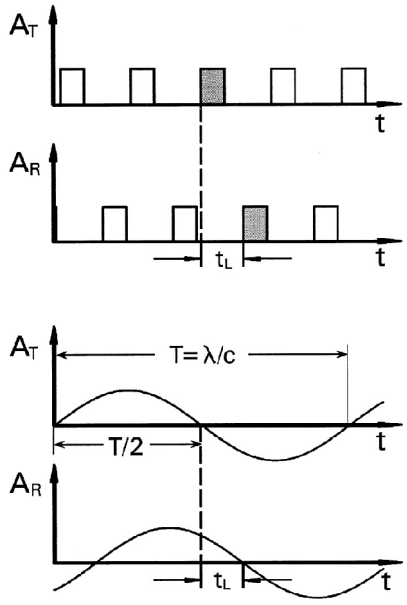


Abb. 2.3: Unterschied im Messprinzip zwischen *pulsed* (oben) und *continuous wave* Lasern (unten). A_T...Amplitude des ausgesendeten Signals, A_R...Amplitude des empfangenen Signals, t_L...Laufzeit, T...Schwingungsdauer des Signals (WEHR & LOHR 1999, 71).

2.3 Anwendungsbereiche

Das Laserscanning wurde ursprünglich für die topographische Geländeaufnahme in Waldgebieten entwickelt (siehe oben: Bodenpunkte in Waldgebieten). Die Nachteile der Luftbildphotogrammetrie werden zu den Vorteilen des Laserscannings. Überall dort, wo die Sichtverhältnisse eingeschränkt sind (z.B. dichter Bewuchs oder Bebauung) oder mangelnder Kontrast und Textur die Auswertung der Luftbilder erschweren (z.B. Gletscher, Strände), können mit dem Laserscanning brauchbare Ergebnisse erzielt werden.
Die Forschung beschäftigt sich sehr intensiv mit der Entwicklung neuer Auswerteverfahren und der Verbesserung der Laserscannermessung. Das Anwendungsspektrum ist sehr vielfältig.

Einige Anwendungsbeispiele

- Topographische Geländeaufnahme (Digitale Gelände- und Oberflächenmodelle)
- Forstwirtschaft: Vegetationshöhenbestimmung, Ermittlung von Waldgrenzen. usw.
- Digitale Stadtmodelle: Abgrenzung und 3D-Erfassung von Gebäuden
- Überwachung von Hochspannungsleitungen
- Multitemporale Auswertung von Massenbewegungen (z.B. Rutschungen, Gletscher)

2.4 Zusammenfassung

Vorteile

- gleichmäßige Punktverteilung, hohe Punktdichte und hohe Genauigkeit (die Probleme der Genauigkeit sind bekannt)
- hoher Automatisierungsgrad in der Erfassung und Auswertung der Daten
- wetterunabhängiger als Luftbildflüge
- Befliegung auch bei Nacht möglich
- Aufzeichnung von Mehrfachreflexionen (Geländeaufnahme in bewaldetem Gebiet möglich, Vegetations- und Gebäudehöhenbestimmung)
- Erstellung von Intensitätsbildern/-rastern möglich
- geringer Aufwand direkt bei der Befliegung: es werden keine *Ground Control Points* benötigt
- gutes Kosten-Nutzen-Verhältnis für große Befliegungsgebiete

Nachteile

- große Ungenauigkeiten in steilem Gelände (schleifende *footprints*)
- große Datenmenge erfordert eine gute Hard- und Software

- Klassifizierung der Punkte (was ist ein Geländepunkt?) ist noch nicht ausgereift
- hoher Aufwand für die Verarbeitung der Daten, die bei der Befliegung aufgezeichnet wurden (Prozessierung einer großen Datenmenge, händisches Nachkorrigieren der Punktklassifikation bei der Erstellung von Geländemodellen) → Kosten
- schlechtes Kosten-Nutzen-Verhältnis für kleine Befliegungsgebiete

3 Einführung in das Untersuchungsgebiet

3.1 Geographische Lage

Das Untersuchungsgebiet liegt im Südosten der Stadt Hohenems, die sich zwischen dem Rhein und dem Bregenzerwald im mittleren Vorarlberger Rheintal befindet (Abb. 3.1, Tab. 3.1). Im Norden wird Hohenems von der Stadt Dornbirn, im Osten von den Bergen Staufen (1465 m), Bocksberg (1461 m), Hoher Freschen (2004 m) und Hohe Kugel (1645 m), im Süden von den Gemeinden Götzis und Altach und im Westen vom Rhein, der Österreich von der Schweiz trennt, abgegrenzt. Das Zentrum der 14000 Einwohner zählenden Stadt liegt an der östlichen Rheintalseite. Der größte Teil der Gemeindefläche wird von gebirgigem Gelände, das langsam auf der östlichen Seite des Rheintals ansteigt, eingenommen.
Das 500 mal 500 m große Untersuchungsgebiet wird im Osten durch die Parzelle Berg, im Süden durch den Bodenweg, im Westen durch die Viktor-Scheffel-Straße und die Kaiser-Josef-Straße und im Norden durch die Lehenstraße begrenzt (vgl. INTERNET: Hohenems).

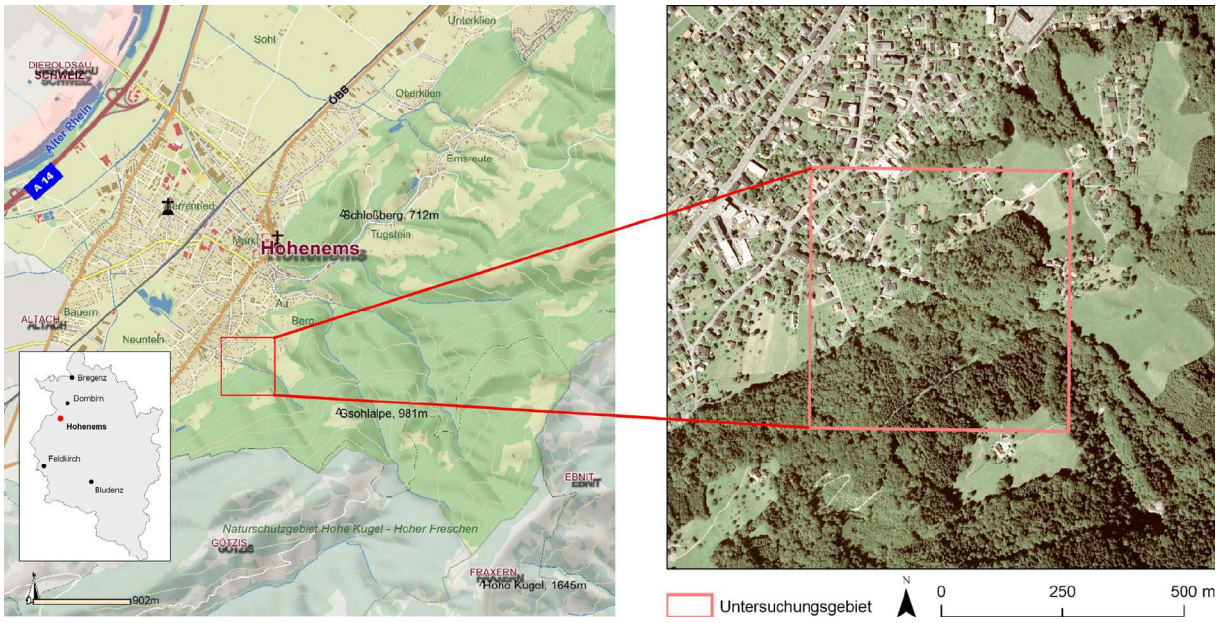


Abb. 3.1: Das Untersuchungsgebiet im Süden der Stadt Hohenems (Vorarlberg).
links: Quelle der Hintergrundkarte: www.hohenems.at
rechts: Quelle des Pseudoechtfarben-Luftbilds: © Land Vorarlberg (Landesvermessungsamt), 2001.

Geographische Koordinaten des Mittelpunktes	9° 41' östliche Länge, 47° 21' nördliche Breite
UTM-Koordinaten der Eckpunkte	NW: N 5244880, E 552110 NE: N 5244880, E 551610 SE: N 5244380, E 551610 SW: N 5244380, E 551110
Höhenerstreckung	436.74 m bis 688.58 m (mittlere Höhe: 551.79 m)
Fläche	500 x 500 m = 250.000 m² = 0.25 km^2

Tab. 3.1: Allgemeine Informationen zum Untersuchungsgebiet in Hohenems.

3.2 Situation

3.2.1 Topographie

In Abbildung 3.2 ist ein Digitales Geländemodell, das aus den Laserscannerdaten erstellt wurde, zu sehen. Bei der Berechnung des Geländemodells werden alle Objekte (z.B. Häuser, Vegetation) entfernt. Das Geländemodell erlaubt einen Blick durch die Vegetation und macht den Boden sichtbar. Deutlich zu erkennen ist das tief eingeschnittene Bachbett des Pelzreutebaches, der das Untersuchungsgebiet wesentlich prägt. Am Unterlauf des Baches zeigen die Höhenlinien einen Wechsel der Geländeform an. Das tief eingeschnittene Bachbett mündet in einen Schwemmfächer, der sich ins Tal ausbreitet. Ferner gut sichtbar sind die Straßen im Talbereich und die Forstwege im Hangbereich.
Das Untersuchungsgebiet erstreckt sich von ca. 435 m über dem Meer im Talbereich bis ca. 690 m im Hangbereich (vgl. Tab. 3.1), was einer Höhendifferenz von 255 m auf ca. 500 m horizontaler Distanz entspricht. Die genannten Höhenwerte entstammen den unklassifizierten Laserpunkten und können somit auch auf Objekten liegen.
Die Spannweite der Oberflächenbeschaffenheit ist im Untersuchungsgebiet recht groß. Es weist größere flache Bereiche ohne Objekte im Talbereich bis hin zu stark bewachsenen steilen Hängen auf.

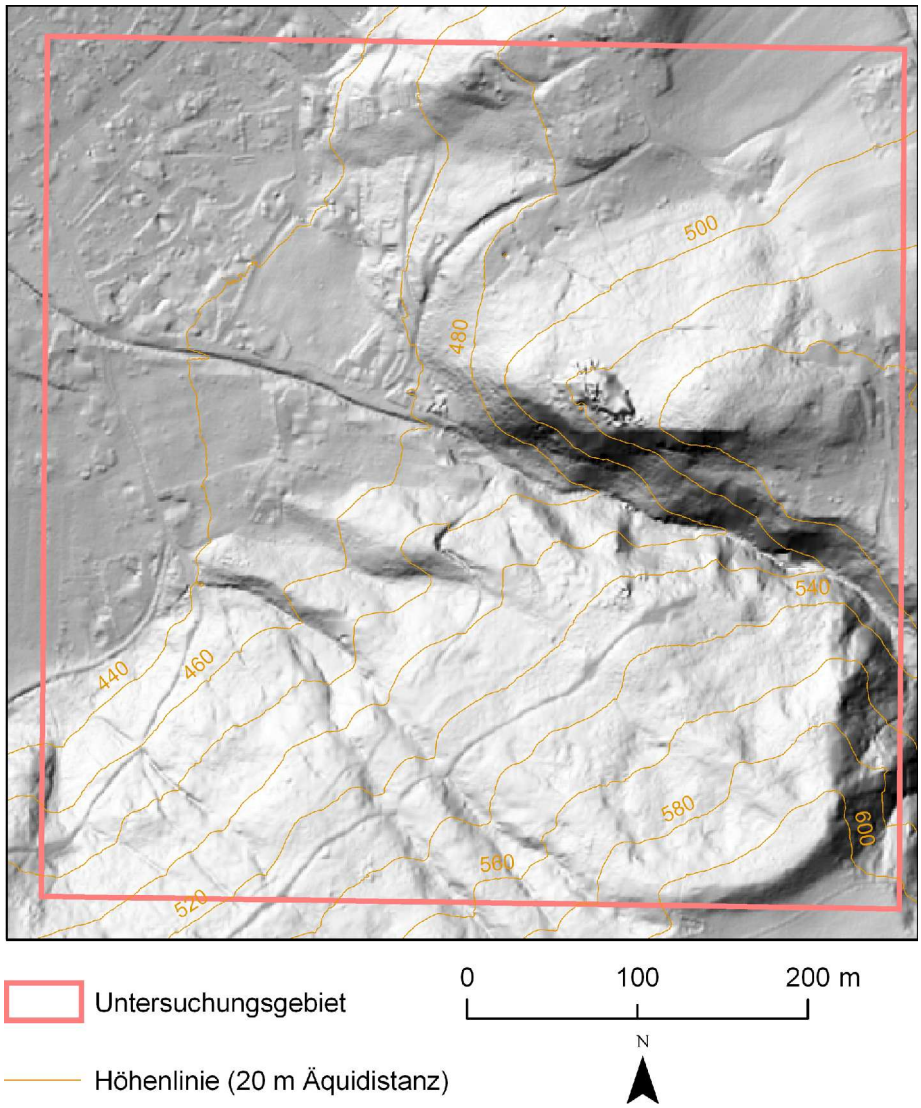


Abb. 3.2: *Hillshade* des Digitalen Geländemodells (1 m Rasterweite).
Quelle: © Land Vorarlberg (Landesvermessungsamt), 2003.

3.2.2 Siedlungsstruktur

Das Digitale Oberflächenmodell beinhaltet im Gegensatz zum Geländemodell alle Objekte, die sich auf dem Gelände befinden. Die Abbildung 3.3 (Digitales Oberflächenmodell) erlaubt wenig Sicht auf den Boden, v.a. in stark bewachsenen Gebieten. Es können jedoch Aussagen über die Objekte (v.a. Häuser, Vegetation) getroffen werden. Der Bodenweg, der Forstweg im südlichen bewaldeten Teil des Untersuchungsgebietes ist selbst im Oberflächenmodell immer noch recht deutlich zu erkennen. Das Flussbett des Pelzreutebaches verschwindet im bewaldeten Gebiet und wird erst wieder in kanalisierter Form im Talbereich sichtbar.
Im Untersuchungsgebiet sind kaum große Gebäude zu finden. Es handelt sich vorwiegend

um Einfamilienhäuser. Im südöstlichen Bereich des Untersuchungsgebietes („Bodner“) liegt ein Wohngebäude mit angeschlossenem Wirtschaftsgebäude. Die Wiesen rund um den Hof werden als Weideflächen genutzt.
Im Talbereich stehen die Einzelhäuser relativ dicht gedrängt (Abb. 3.4). Zwischen den Häusern sind immer wieder größere Gärten mit Obstbäumen angelegt. Die Häuser stehen bis direkt an den Wald, der mit dem ansteigenden Gelände Richtung Südosten beginnt, und direkt neben dem verbauten Wildbach.

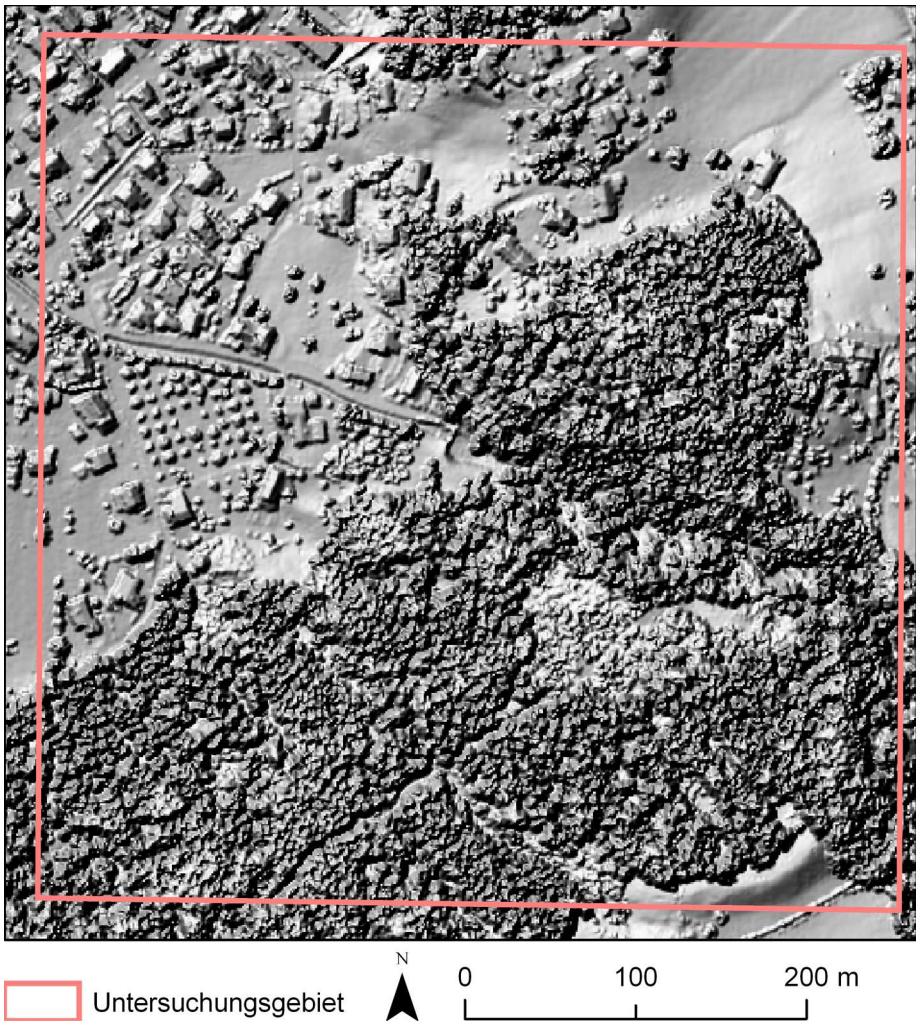


Abb. 3.3: *Hillshade* des Digitalen Oberflächenmodells (1 m Rasterweite).
Quelle: © Land Vorarlberg (Landesvermessungsamt), 2003.

Abb. 3.4: Einfamilienhäuser im Talbereich des Untersuchungsgebietes (Höfle, 25.12.2004).

3.2.3 Vegetation

Die Vegetation im Untersuchungsgebiet setzt sich aus landwirtschaftlich genutzten Mähwiesen, Obstbäumen, Rasenflächen und Waldflächen zusammen. In der Abbildung 3.5 zeigen sich die bewachsenen Flächen in rot. Anhand des Infrarotluftbildes können Obstbäume, Mähwiesen, Rasenflächen und Laub- und Nadelbäume gut voneinander unterschieden werden. Gleichzeitig sticht alles hervor, was nicht Vegetation ist (Häuser, Verkehrsflächen). Bei der Geländebegehung konnte ein vorherrschender Buchen-Tannenwald mit Fichten im Unterwuchs festgestellt werden (Abb. 3.6). Im großen Maßstabsbereich können Nadel- und Laubbaumbereiche voneinander abgegrenzt werden (vgl. RUTZINGER 2005).

Abb. 3.5: CIR (*coloured infrared*)-Luftbild (0.25 m Rasterweite).
Quelle: © Land Vorarlberg (Landesvermessungsamt), 2001.

Abb. 3.6: Buchen-Tannenwald (Höfle, 25.12.2004).

3.2.4 Naturgefahren

Am 6. August 2000 ereignete sich ein Murgang im Einzugsgebiet des Pelzreutebaches, der im Stadtgebiet von Hohenems schwere Schäden an Häusern und Grundstücken anrichtete. Das Anrissgebiet befindet sich außerhalb (östlich) des Untersuchungsgebietes. Das Muranrissgebiet im Oberlauf des Pelzreutebaches wurde unmittelbar nach dem Ereignis von der Wildbach- und Lawinenverbauung Vorarlberg stabilisiert. Im Jahr 2003 wurden am oberen Schwemmkegel ein Geschiebe- und Murauffangbecken mit Balkensperre und der Ausbau des Schwemmkegelgerinnes fertig gestellt (Abb. 3.7). Die Projektkosten für die Erbauung der Schutzmaßnahmen belaufen sich auf rund 835.000 €. Die Verbauungsmaßnahmen am Schwemmkegel stellen eine gewisse Verminderung des Gefährdungspotentials für die Objekte in der Nähe des Gerinnes dar. Die Prozesse im Oberlauf sind jedoch immer noch aktiv, wie ein kleiner Erdrutsch im Spätsommer 2002, der sich nach heftigen Regenfällen ca. 100 bis 150 m entfernt vom Anrissgebiet der Mure vom August 2000 löste, zeigt (INTERNET: Wildbach- und Lawinenverbauung Vorarlberg - Zwischenbilanz Oktober 2003, INTERNET: Die Emsigen – Pressemeldungen 09/2002).

Abb. 3.7: Geschiebeablagerungsbecken mit Balkensperre am oberen Schwemmkegel des Pelzreutebaches (Höfle, 25.12.2004).

3.3 Laserscannerbefliegungen

Das Land Vorarlberg hat die Erstellung eines landesweiten Digitalen Geländemodells basierend auf Laserscannerdaten in Auftrag gegeben. Der größte Teil des Landes wurde von der Firma Topscan GmbH. beflogen. Der Projektbericht bezüglich der Laserscannermessung Unterland und Vorderwald (TOPSCAN GmbH. 2004; © Land Vorarlberg – Landesvermessungsamt) gibt einen guten Überblick über die Befliegungskampagnen, die das Untersuchungsgebiet betreffen (vgl. Appendix A).

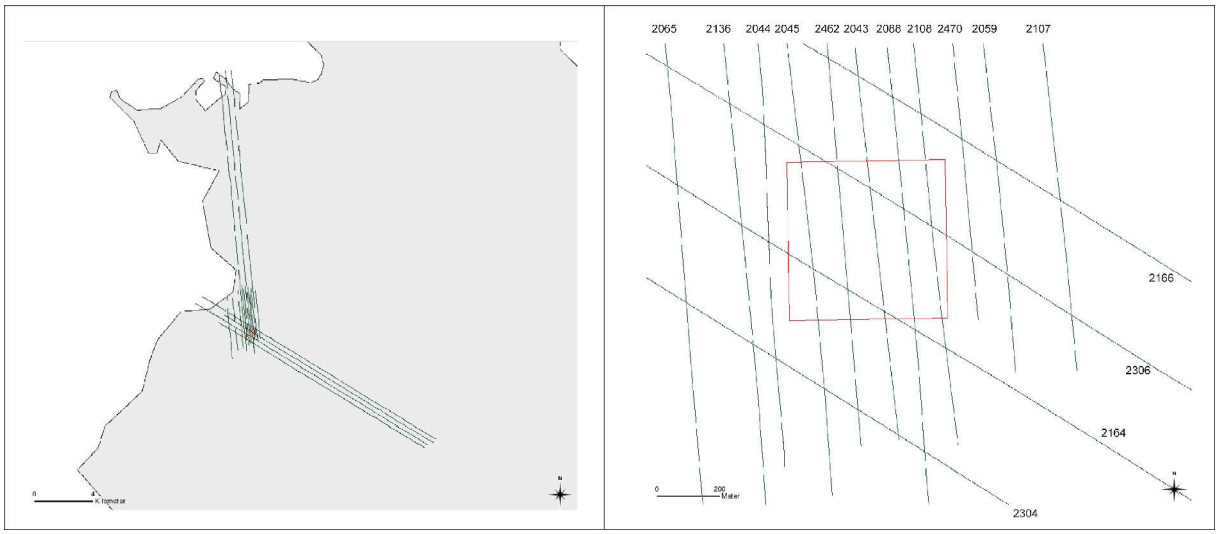


Abb. 3.8: Flugstreifenübersicht (eigener Entwurf).
links: Vorarlberger Unterland mit Flugstreifen von Hohenems bis zum Bodensee und von Hohenems bis zum Hohen Freschen.
rechts: Detailansicht der Flugstreifen, die das Untersuchungsgebiet schneiden.

ID	Nr	Datum	Laserpunkte	Flugzeugpositionen	Länge [m]	mittlere Fluggeschwindigkeit [m/s]
1	2306	05.11.2003	18926720	47868	16626.82	69.47
2	2304	05.11.2003	18567812	47724	16280.58	68.23
3	2166	05.11.2003	20538871	53400	18576.75	69.58
4	2164	05.11.2003	20294858	52465	18332.67	69.89
5	2470	04.11.2003	4390806	10600	3343.1	63.08
6	2462	04.11.2003	4975915	12000	3860.92	64.36
7	2136	04.11.2003	21254477	55874	18991.34	67.98
8	2108	04.11.2003	4351273	10500	3543.61	67.51
9	2107	04.11.2003	4193905	10250	3348.31	65.34
10	2093	04.11.2003	4322844	10600	3572.78	67.42
11	2088	04.11.2003	6042728	14800	4476.63	60.5
12	2065	04.11.2003	4719228	11551	3954.69	68.48
13	2059	04.11.2003	20187310	51788	18220.52	70.37
14	2046	04.11.2003	5196064	13228	4248.46	64.24
15	2045	04.11.2003	4892486	11800	3804.09	64.48
16	2044	04.11.2003	4778368	11500	4002.02	69.61
17	2043	04.11.2003	21505356	55555	18460.47	66.46
Gesamt			**189139021**	**481503**	**163643.76**	**66.88**
Untersuchungs-gebiet			**3797446**	**-**	**-**	**-**

Tab. 3.2: Flugstreifenübersicht (eigener Entwurf: Auszug aus dem Informationssystem).

An zwei Befliegungstagen wurden 17 Flugstreifen im Bereich des Untersuchungsgebietes aufgenommen (Abb. 3.8, Tab. 3.2). Da die Laserpunkte immer in ganzen Flugstreifen abgespeichert sind, müssen zuerst alle Punkte der 17 Flugstreifen zur Bearbeitung herangezogen werden (vgl. Kapitel *„4 Datenerfassung und Datengrundlagen“*). Diese 17 Flugstreifen, die mitunter bis zum Bodensee reichen, weisen ca. 190 Millionen Laserpunkte auf. Im Untersuchungsgebiet liegen aber nur ca. 3.8 Millionen Punkte (Tab. 3.3).

	Anzahl der Punkte	**Punktdichte [Punkte/m²]**	**Anteil an der Gesamtanzahl [%]**
first pulse	1835264	7.34	48.3
last pulse	1962182	7.85	51.7
Gesamt	3797446	15.19	100.0

Tab. 3.3: Überblick über die Laserpunkte im Untersuchungsgebiet.

3.4 Zusammenfassung

Das Untersuchungsgebiet erfüllt die meisten Kriterien, die ein ideales Testgebiet erfüllen muss. Das Gebiet ist ausreichend groß (500 x 500 m), um möglichst viele Oberflächenphänomene darin vorfinden zu können, und trotzdem bleibt die Datenmenge für die Leistung der Hardware akzeptabel. Das Testgebiet befindet sich am Schnittpunkt von zwei Befliegungskampagnen. Die Punktdichte ist mit ca. 15 Punkten pro Quadratmeter ausgesprochen hoch. Das Untersuchungsgebiet weist eine sehr heterogene Oberflächenstruktur auf. Es finden sich verschiedene Landnutzungstypen auf unterschiedlichstem Gelände. Von Verkehrsflächen und Häusern bis zu Obst- und Nadelbäumen, von flachem bis sehr steilem Gelände sind alle Variationen im Untersuchungsgebiet vorhanden.
Ein wichtiger Grund für die Auswahl eines Gebietes ist das Vorhandensein von Referenzdaten, die eine objektive Evaluierung der Ergebnisse erlauben. Dieses Kapitel zeigt die wichtigsten Referenzdaten (Luftbilder, Digitale Gelände- und Oberflächenmodelle, Fotos, Geländebegehungen) auf.

4 Datenerfassung und Datengrundlagen

4.1 Einleitung

Im folgenden Kapitel werden die verwendeten Daten und ihre Struktur beschrieben. Die Datensätze wurden dankenswerterweise vom Land Vorarlberg (Landesvermessungsamt) und der Universität Innsbruck zur Verfügung gestellt. Das Kapitel *„4 Datenerfassung und Datengrundlagen“* gliedert sich in zwei Teile. Für ein besseres Verständnis der Datenstruktur wird im ersten Teil auf die Erfassung und Vorverarbeitung (*pre-processing*) der Laserscannerdaten eingegangen. Im zweiten Teil wird die vorliegende Datenstruktur detailliert aufgezeigt.
Der Aufnahmeprozess des Laserscannings ist im Kapitel *„2 Grundlagen des flugzeuggestützten Laserscannings“* schematisch dargestellt und leicht verständlich beschrieben. Aufbauend auf den Grundlagen werden die bei der Befliegung erfassten Daten in diesem Kapitel erklärt.
Wichtig zu erwähnen bleibt, dass ausschließlich mit Datensätzen der Firma TopScan GmbH gearbeitet wird. Das Aufnahmesystem, die Datenauswertung und die Struktur der dabei gewonnenen Daten können von Firma zu Firma sehr verschieden sein.

4.2 Erfassung und Vorverarbeitung von Laserscannerdaten

4.2.1 Projektplanung

Jeder Befliegung geht eine genaue Projektplanung voraus. Je nach Topographie, Größe, Lage und Form des Befliegungsgebietes und je nach gewünschter Punktdichte muss von Seiten der Befliegungsfirma eine Strategie entwickelt und die Systemparameter entsprechend gewählt werden. Folgende Punkte beeinflussen das Ergebnis einer Laserscannerbefliegung:

- **Befliegungszeitpunkt**: Das flugzeuggestützte Laserscanning ist ein aktives System und ist daher nicht von tageszeitlichen Lichtschwankungen und Schattenwürfen beeinflusst. Das Wetter zum Befliegungszeitpunkt spielt eine wichtige Rolle. Bei Regen und starkem Wind kann nicht geflogen werden. Zu viel Wasser in der Atmosphäre zwischen Flugzeug und Boden stört den Laserstrahl. Wind führt zu einem unruhigen Flug und damit zu einer unregelmäßigen und lückenhaften Punktverteilung am Boden. Dient die Befliegung der Aufnahme von Vegetation, so sollte möglichst während der Vegetationsperiode geflogen werden. Sind Geländemodelle, das heißt eine möglichst hohe Durchdringungsrate mit möglichst vielen Bodenpunkten, das vorrangige Ziel, ist ein Zeitpunkt mit gering

entwickelter Vegetation zu wählen. Für die exakte Bestimmung von Vegetationshöhen sind ein gutes Gelände- und Oberflächenmodell ideal.

- **Scanwinkel:** Die möglichen Scanwinkel für die Messsysteme, die von TopScan verwendet werden, betragen 0 - ± 20°. Das aufgespannte Sichtfeld des Scanners (*field of view*) wird in der Literatur (BALTSAVIAS 1999, 214) als Scanwinkel bezeichnet. Der Scanwinkel (Gesamtöffnungswinkel) beträgt somit 40°. Je geringer der Scanwinkel bei gleich bleibender Scanfrequenz und Messrate, desto höher ist die Punktdichte am Boden, da mit einem geringeren Scanwinkel die Streifenbreite abnimmt. Ein zu großer Scanwinkel führt zu schwachen bis keinen Reflexionen am Rand des Scanstreifens und kann im stark geneigten Gelände zu schleifenden Winkeln des *footprint* und somit zu Ungenauigkeiten in der Messung führen.

- **Streifenbreite**: Die Streifenbreite ist direkt proportional zur Flughöhe und zum Scanwinkel und lässt sich mit Formel 4.1 berechnen (WEHR & LOHR 1999, 75; BALTSAVIAS 1999, 204):

$$b = 2 \cdot h \cdot \tan\left(\frac{\Theta}{2}\right) \tag{4.1}$$

b...Streifenbreite (m), h...relative Flughöhe (m), Θ...Scanwinkel (°)

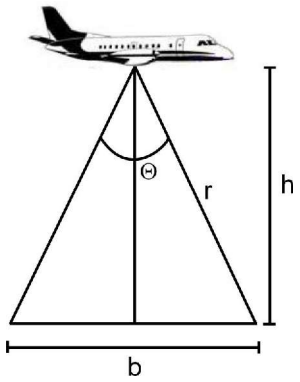


Abb. 4.1: Modellhafte Darstellung der Streifenbreite (b): Scanwinkel (Θ) und relative Flughöhe (h) (eigener Entwurf).

Die maximale Streifenbreite wird durch den maximalen Scanwinkel und durch die maximale Reichweite des Aufnahmesystems, die wiederum von der Stärke des ausgesendeten Laserstrahls, der Empfindlichkeit des Sensors und von der Reflektivität am Boden abhängt, limitiert (vgl. WEVER 1999). Die Streifenbreite wird in der Realität durch Abschattungseffekte aufgrund des Reliefs oder großer Objekte (z.B. Hochhäuser) sowie durch eine schwache Reflektivität der Oberfläche (geringere maximale Reichweite) vermindert.

h [m]	b [m]	r [m]
0	0	0
100	73	106
200	146	213
300	218	319
400	291	426
500	364	532
600	437	639
700	510	745
800	582	851
900	655	958
1000	**728**	**1064**
1100	801	1171
1200	874	1277
1300	946	1383
1400	1019	1490
1500	1092	1596
1600	1165	1703
1700	1237	1809
1800	1310	1916
1900	1383	2022
2000	1456	2128

Tab. 4.1: Relative Flughöhe, maximale Streifenbreite und Reichweite.
Die maximale Streifenbreite beträgt, bei einer relativen Flughöhe von 1000 m, ca. 730 m. Bei größeren Flughöhen werden die reflektierten Signale an den Streifenrändern zu schwach. Es wird ein maximaler Scanwinkel von 40° angenommen. Relative Flughöhe (h), maximale Streifenbreite (b) und Reichweite (r). Die Angaben gelten für den ALTM 1020 der Firma Optech (WEVER 1999).

- **Relative Flughöhe**: Wie in Formel 4.1 zu erkennen ist, kann über die Flughöhe die maximale Streifenbreite geregelt werden. Ferner ist der Durchmesser des Laserstrahls am Boden (*footprint*) unter anderem von der Flughöhe über Grund abhängig. Bei einer Strahldivergenz von 0.25 mrad und einer Flughöhe von 1000 m ergibt sich ein *footprint* mit ca. 25 cm Durchmesser auf einer horizontalen Fläche (WEVER 1999, 13). Die Größe des *footprint* entscheidet über die Messgenauigkeit und die Möglichkeit von Mehrfachreflexionen.

- **Messrate**: Die Messrate ist die Anzahl der gesendeten Laserimpulse pro Sekunde und wird in Hertz angegeben. Das von TopScan verwendete Messsystem, der ALTM 2050, hat eine Messrate von 50.000 Hz (Tab. 4.3). Je größer die Messrate, desto größer wird auch die Punktdichte am Boden sein.

- **Scanfrequenz**: Die Scanfrequenz gibt die Anzahl der vollständigen Scandurchläufe pro Sekunde (in Hertz) an. Ein vollständiger Scandurchlauf ist zum Beispiel das Abtasten des Scanstreifens von -20° bis +20° und wieder bis -20°. Das Sichtfeld des Scanners (40°) wird also zweimal, einmal von links nach rechts und retour, durchlaufen (Abb. 4.2 und BALTSAVIAS 1999, 204). Wird die Scanfrequenz erhöht, verringert sich der Abstand zwischen den Scanlinien. Der Punktabstand auf der Scanlinie nimmt aber gleichzeitig zu.

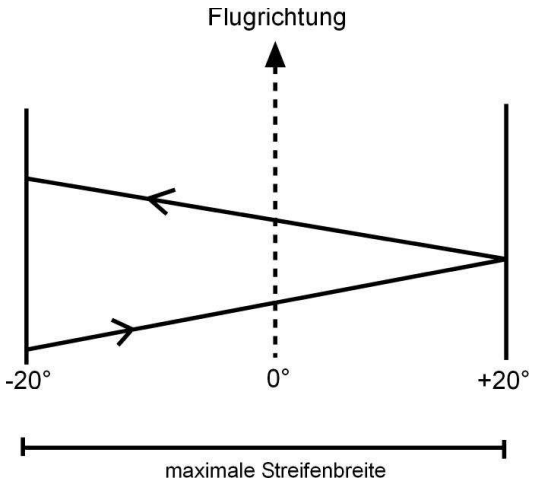


Abb. 4.2: Abgebildet ist ein vollständiger Scandurchlauf. Für 1 Scandurchlauf pro Sekunde würde die Scanfrequenz 1 Hz betragen (eigener Entwurf).

- **Fluggeschwindigkeit**: Die Geschwindigkeit des Flugzeugs entscheidet über die Abstände zwischen den einzelnen Scanlinien (die Abstände zwischen den Zick-Zack-Linien). Bei einer Fluggeschwindigkeit von 70 m/s und einer Scanfrequenz von 35 Hz (=70 Scanlinien) berechnet sich auf der Flugachse ein Abstand der Scanlinien von 1 m und am Streifenrand ein Abstand von 2 m (vgl. WEVER 1999, 14).

- **Streifenüberlappung (Querüberdeckung)**: Bei flächendeckenden Befliegungen wird streifenüberlappend geflogen. Damit werden die größeren Punktabstände an den Streifenrändern ausgeglichen. Je näher die einzelnen Fluglinien bei einander liegen, desto größer ist die Querüberdeckung und desto größer ist die mittlere Punktdichte im gescannten Gebiet. Die Lage der zu fliegenden Streifen wird vor Beginn der Befliegung festgelegt.

	ALTM 2050
Baujahr	2003
Messrate	50.000 Hz
Messmodi	erster und/oder letzter Impuls
Scanwinkel	0 - ± 20°
Flughöhen	210- 2000 m
Intensität	ja
Video	ja
Digitale Messkamera	4k x 4k Messkamera

Tab. 4.2: Messsystem ALTM 2050 der Firma TopScan GmbH (www.topscan.de).

Die zu erwartende Streifenbreite, die Abstände zwischen den Scanlinien, die Punktverteilung und die Punktdichte der Laserpunkte am Boden können durch die gewählten

Systemparameter (Messrate, Scanwinkel, Scanfrequenz) sowie die Flughöhe, die Fluggeschwindigkeit und die Streifenüberlappung sehr gut vor einer Befliegung geplant und berechnet werden. Die tatsächliche Punktverteilung und Punktdichte kann nicht vorhergesagt werden, da das Relief, die Objekte am Boden und die Bewegungen des Flugzeugs das Scanmuster modifizieren. Ein System, das *first* und *last pulse* unterstützt, kann bis zu doppelt so viele Laserpunkte aufnehmen (z.B. bei 50.000 Hz Messrate bis zu maximal 2 x 50.000 Punkte pro Sekunde).
Eine Beschreibung der flugzeuggestützten Laserscannermessung wird in WEHR & LOHR (1999) und WEVER (1999) vorgenommen. BALTSAVIAS (1999) schlüsselt die Messgrößen und abgeleiteten Größen der Laserscannermessung auf und beschreibt sie mit mathematischen Formeln.

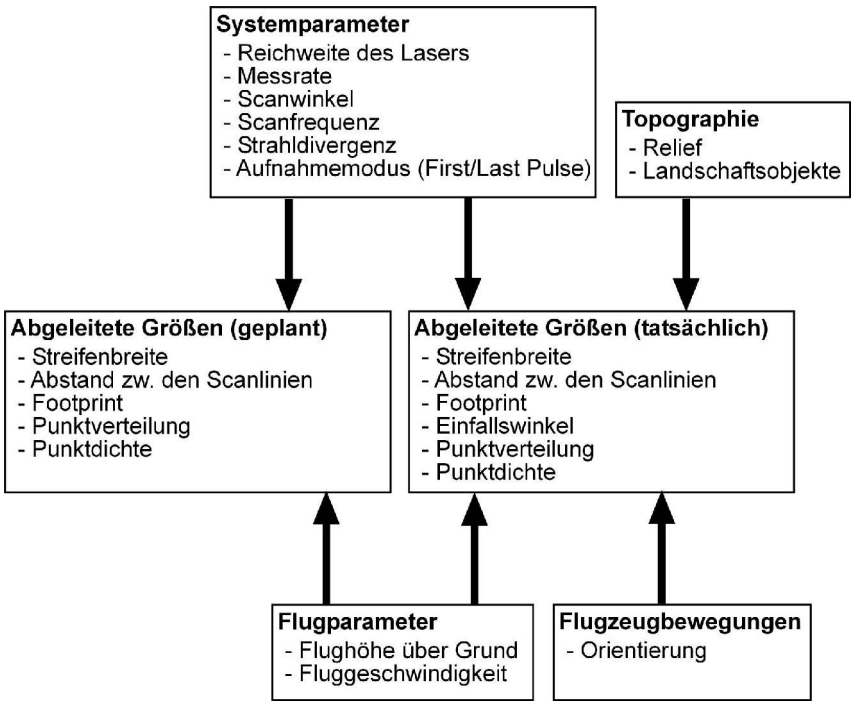


Abb. 4.3: Zusammenspiel der System- und Flugparameter mit der Topographie und den Bewegungen des Flugzeugs während der Befliegung und die daraus abgeleiteten Größen (eigener Entwurf).

4.2.2 Metadaten zur Befliegung

Metadaten zur Befliegung sind Informationen über die eigentlichen Messdaten. Metadaten sind für ein reibungsloses Arbeiten mit Laserscannerdaten unentbehrlich. Diese Zusatzinformationen werden ebenfalls von der Befliegungsfirma zur Verfügung gestellt und werden daher dem Punkt *„4.2 Erfassung und Vorverarbeitung von Laserscannerdaten“* zugeordnet. Folgende Informationen sollten von der Befliegungsfirma mitgeliefert werden:

- **Systemparameter**: Die Kenntnis mit welchem Laserscanner bzw. mit welchem Aufnahmesystem geflogen wurde, ist für die Weiterverarbeitung und Interpretation der

Daten sehr wichtig. Die Messrate des Scanners und der verwendete Messmodus (*first* und/oder *last pulse*) können bei der Betrachtung der Messdaten herausgefunden werden. Die Strahldivergenz, die für die Berechnung des *footprint* herangezogen wird, kann den Messdaten nicht entnommen werden und muss daher in den Metadaten angegeben sein. Ebenfalls Teil der Metadaten sollten der verwendete Scanwinkel und die Scanfrequenz sein.

- **Koordinatensystem**: Die Metadaten sollten Informationen über das Koordinatensystem, in welchem die räumlichen Daten vorliegen, enthalten. Die Projektionsparameter und der Bezugsellipsoid sind notwendige Metadaten und müssen jedem Datensatz eindeutig zugeordnet werden können. Der Genauigkeitsanspruch an die Laserscannerdaten erfordert die Berücksichtigung der Geoidundulationen. Die Metadaten sollten Auskunft darüber geben, ob und wie dies geschehen ist.

- **Befliegungszeitpunkt**: Das Datum der Befliegung kann beim ausgelieferten Format von TopScan aus den Messdaten entnommen werden. Die exakte zeitliche Einordnung jedes Laserpunkts und jeder Flugzeugposition ist möglich.

4.2.3 Flugzeugposition und -ausrichtung

Das Inertiale Navigationssystem (INS) arbeitet mit einer zeitlichen Auflösung von 200 Hz, d.h. 200 Aufzeichnungen pro Sekunde (mündl. Mitteilung GROSSER, 22.07.2004). Die Orientierung der Aufnahmeplattform im Raum wird mit den drei Winkeln Rollen (*roll*), Nicken (*pitch*) und Driften (*heading*) festgehalten. Ein vierter Winkel, der w*ander angle*, erlaubt die Berechnung des *true heading* (Abweichung von Nord in Grad).

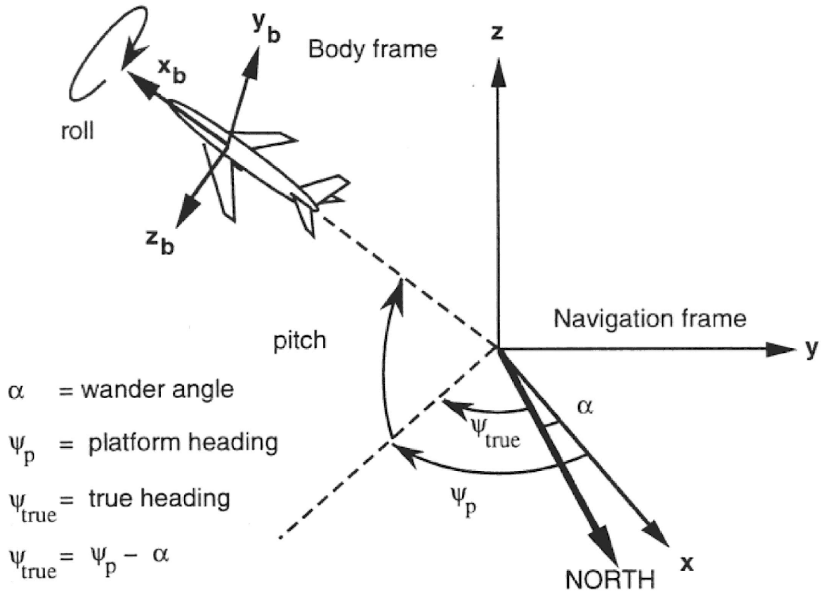


Abb. 4.4: Orientierung des Flugzeugs im Raum mit den Winkeln *roll*, *pitch* und *heading* (verändert nach SCHERZINGER, 1993).

Die absolute Flugzeugposition wird mit einem GPS-Empfänger zweimal in der Sekunde (2 Hz) im WGS84-Koordinatensystem bestimmt (mündl. Mitteilung GROSSER, 22.07.2004). Im Flugzeug befindet sich ein Bordcomputer, der die GPS-Positionen, die INS-Daten und die Daten des Scanners zeitsynchronisiert abspeichert (Abb. 4.5 und Wever 1999, 13).

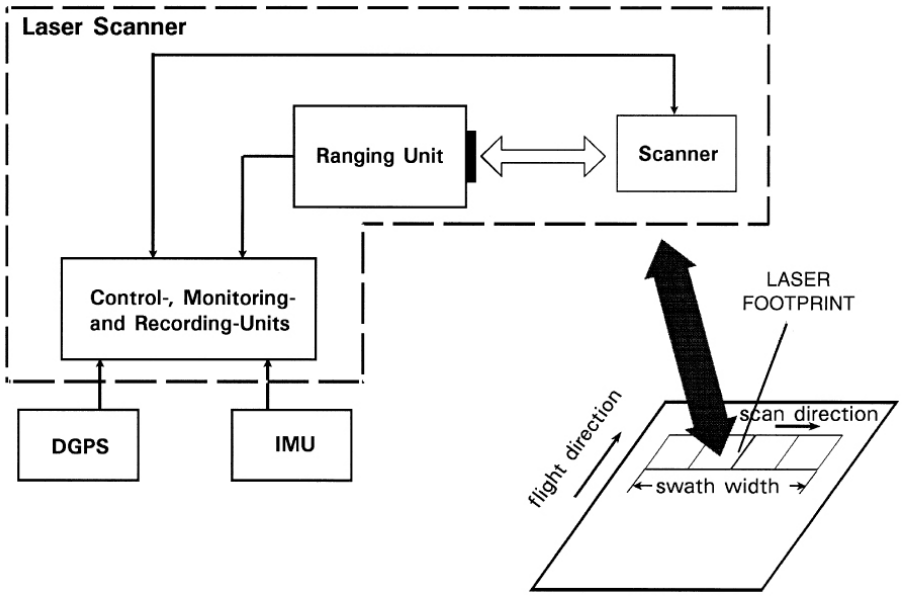


Abb. 4.5: Aufbau eines typischen ALS-Aufnahmesystems (WEHR & LOHR 1999, 69).

Nach der Befliegung beginnt der Auswerteprozess der aufgezeichneten Daten (Abb. 4.6). Die GPS-Daten des Flugzeugs werden mit Hilfe von GPS-Messungen von Bodenstationen, die während der Befliegung gemessen haben, differentiell korrigiert (→DGPS). Mit einem so genannten „Modell zur relativen kinematischen Positionsbestimmung unter Verwendung von doppelten Phasendifferenzen“ (WEVER 1999, 14; LINDENBERGER 1993, 28f.) wird eine *best estimated trajectory* der Fluglinie berechnet (mündl. Mitteilung LINDENBERGER, 10.08.2004). Dabei fließen die korrigierten Positionen der GPS-Antenne und die INS-Daten mit ein. Zusätzlich mit den Informationen zur Aufhängung der Messsysteme im Flugzeug und den Koordinatensystemen, in denen die einzelnen Messsysteme die Daten erfassen, kann für jeden Laserimpuls die entsprechende Startposition des Laserstrahls berechnet werden. Eine detaillierte Beschreibung der Positionsbestimmung geben LINDENBERGER (1993) und FAVEY (2001, 31ff.).

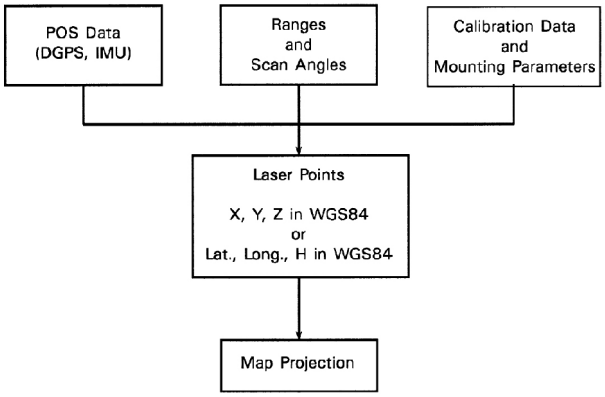


Abb. 4.6: Flussdiagramm zur Verarbeitung der Messdaten (verändert nach WEHR & LOHR 1999, 79).

4.2.4 Laserpunkte

Der Ursprung des Laservektors wird mit den GPS- und INS-Aufzeichnungen rekonstruiert (siehe oben). Über die Laufzeit des ausgesendeten Impulses – die Zeit von Aussendung des Impulses am Sender, Reflexion am Boden und Registrierung des reflektierten Impulses am Empfänger – kann die Länge des Vektors berechnet werden (Formel 4.2).

$$R = \frac{1}{2} \cdot c \cdot t_L \tag{4.2}$$

R...Länge des Vektors, *range* (m), c...Gruppengeschwindigkeit des Laserpulses (m/s), t_L...Laufzeit (s) (WEHR & LOHR 1999, 71).

Die Richtung des Vektors (Ablenkwinkel) wird für jeden Impuls aufgezeichnet. Nun können der Ursprung des Vektors, die Richtung und Länge kombiniert werden und die Koordinaten des Punktes an der Erdoberfläche berechnet werden.
First und *last pulse* weisen den selben Ursprung und die selbe Richtung auf, aber unterschiedliche Laufzeiten und in weiterer Folge auch unterschiedliche Koordinaten an der Erdoberfläche.
Bei jeder Befliegung wird am Boden die Höhe einer Kontrollfläche tachymetrisch und mit GPS vermessen. In der Regel handelt es sich dabei um eine große ebene Fläche mit geringer Oberflächenrauigkeit (z.B. Sportplätze, Parkplätze). Die Höheninformationen dieser Kontrollfläche werden mit den Werten der Laserscannermessung verglichen und zur Systemkalibrierung herangezogen (WEVER 1999, 14; LINDENBERGER 1993, 42ff.). Eine weitere Möglichkeit der Nachkorrektur der Messdaten stellt das Verfahren des *strip adjustment* dar. Dem liegt zugrunde, dass in den Überlappungsbereichen der Streifen die Erdoberfläche von zwei Flugzeugpositionen aus gescannt wurde. Damit lassen sich systematische Höhenfehler und Fehler im Rollwinkel kompensieren (vgl. MAAS 2001, BURMAN 2002 und FILIN 2003).

4.2.5 Signalintensität

Die ALS-Systeme, die TopScan im Einsatz hat, zeichnen die Signalintensität der ersten und letzten Reflexion jedes ausgesendeten Laserimpulses auf. Die neuesten Aufnahmesysteme auf dem Markt (z.B. LMS-Q569 der Firma Riegl oder ALTM 3100 der Firma Optech) erlauben das Speichern der gesamten Impulsform des reflektierten Laserstrahls (*full-waveform scanner*), was die Datenmenge um ein Vielfaches potenziert. Im Folgenden wird die Arbeitsweise eines Systems, das die erste und letzte Reflexion digitalisiert, beschrieben.
Die Verarbeitung des empfangenen Signals ist ein mehrstufiger Prozess (vgl. LINDENBERGER 1993, 22):

- Das einfallende Licht wird in der Empfangsoptik fokussiert und durch ein optisches Filter, das nur Licht der Wellenlänge des ausgestrahlten Laserstrahls durchlässt, geleitet.
- Ein Photodetektor (z.B. Photodiode) registriert die optischen Impulse und wandelt sie in elektrische Signale um.
- Das schwache Signal wird für die weitere Verarbeitung verstärkt.
- Die erste und letzte Reflexion wird herausgefiltert und zeitlich verortet.
- Mit der Zeitdifferenz zwischen Aussendung und Registrierung der Reflexion (=Laufzeit) wird die Länge des Laservektors (*range*) berechnet und abgespeichert (siehe Formel 4.2 und Abb. 4.7).
- Die Intensitäten der ersten und letzten Reflexion werden aus Speicherplatzgründen mit einer logarithmischen Funktion auf 8 oder 12 Bit heruntergerechnet und abgespeichert (mündl. Mitteilung LINDENBERGER, 10.08.2004).

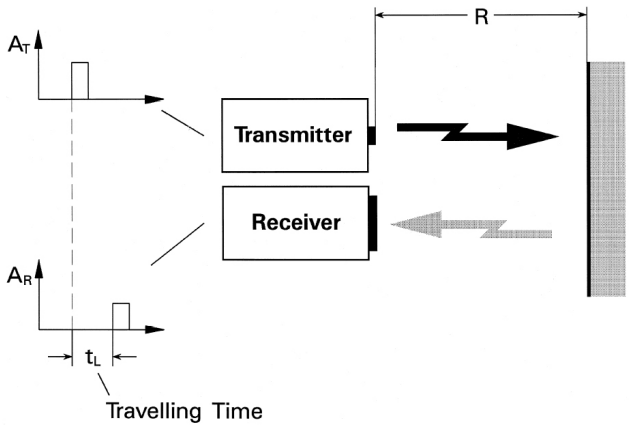


Abb. 4.7: Laufzeitmessung: A_T ist die Amplitude des gesendeten Signals, A_R die Amplitude des empfangenen Signals und t_L die gemessene Laufzeit (WEHR & LOHR 1999, 70).

Einige gängige Verfahren zur Detektion der ersten und letzten Reflexion aus dem gesamten empfangenen Signal sind in WAGNER et. al. (2003) aufgelistet. Die ursprünglich gesendete Impulsform und die Amplitude des Signals werden durch Interaktion des Laserstrahls mit der Atmosphäre und der Erdoberfläche verändert. Die Signalintensität stellt dabei die Amplitude der jeweiligen Reflexion dar.

4.3 Vorliegende Datenstruktur

4.3.1 Anordnung der Datenfiles

Die Beschreibung beschränkt sich auf die Flugzeug- und Laserpunktdaten und geht nicht auf eventuell mitgelieferte digitale Gelände- oder Oberflächenmodelle, Intensitätsraster usw. ein. Die dargestellten Auszüge aus den Originaldaten beziehen sich auf die Befliegungskampagnen in Vorarlberg (Höhenmodell © 2004 Land Vorarlberg). Abbildung 4.8 gibt einen Überblick über die Anordnung und Benennung der ausgelieferten Datenfiles einer Befliegungskampagne.
Eine Befliegungskampagne setzt sich in der Regel aus mehreren Befliegungen zusammen, die an einem oder mehreren Tagen stattgefunden haben und schlussendlich das gesamte Gebiet abdecken sollten.

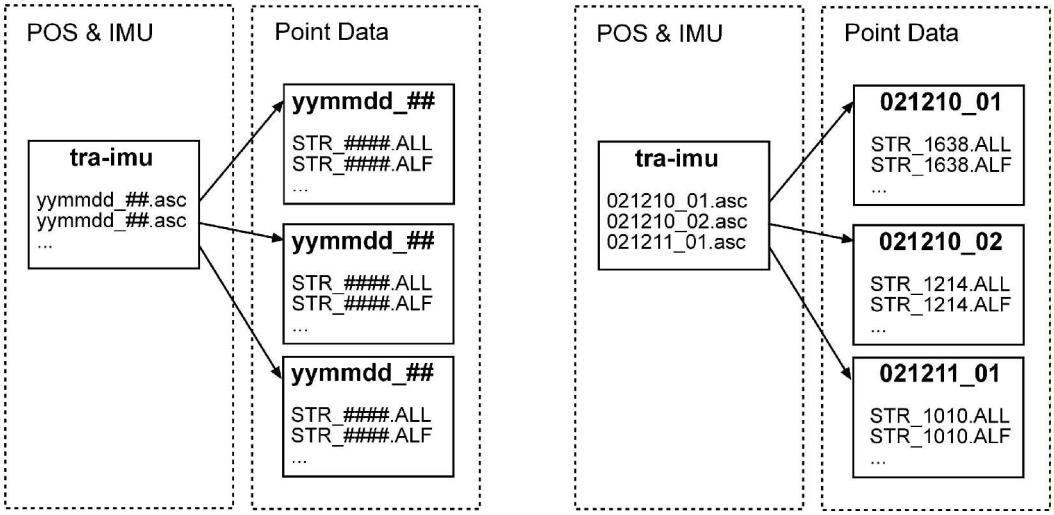


Abb. 4.8: Schema der vorliegenden Datenstruktur. Linke Seite: Theoretische Datenstruktur, Rechte Seite: Beispiel einer Befliegungskampagne (eigener Entwurf).

Alle Dateien liegen im ASCII-Format vor und können mit einem einfachen Texteditor betrachtet werden (Achtung Datenmenge!). Die **Flugzeugdaten** befinden sich alle in einem Verzeichnis („tra-imu“). Sie sind in Dateien organisiert, deren Name sich aus dem Datum der Befliegung, einer fortlaufenden Nummer und der Dateiendung „asc“ zusammensetzt. Es kann vorkommen, dass ein Gebiet mehrmals am Tag beflogen wurde, was am selben

Datum, aber einer unterschiedlichen Nummer (wird um 1 erhöht) zu erkennen ist.

Beispiel: **021210_01.asc**
→ File mit den Flugzeugdaten der ersten Befliegung am 10.12.2002.

Time	UTM-east	UTM-north	UTM-height	roll	pitch	heading	wander angl
125829.99456	32548188.958	5248153.734	2555.650	-0.314069	7.809263	120.460390	5.281013
125829.99957	32548189.248	5248153.634	2555.647	-0.310768	7.809922	120.458642	5.281009
125830.00457	32548189.538	5248153.534	2555.643	-0.307506	7.811673	120.456882	5.281006
125830.00956	32548189.828	5248153.435	2555.641	-0.299546	7.813061	120.455106	5.281149
125830.01456	32548190.119	5248153.335	2555.638	-0.293035	7.813939	120.453129	5.281146
125830.01956	32548190.409	5248153.235	2555.635	-0.291708	7.813945	120.451600	5.281143
125830.02456	32548190.699	5248153.135	2555.632	-0.286917	7.815043	120.449843	5.281140
125830.02956	32548190.990	5248153.036	2555.628	-0.278350	7.816361	120.448525	5.281137
125830.03456	32548191.280	5248152.936	2555.625	-0.273339	7.817677	120.446767	5.281134
125830.03956	32548191.570	5248152.836	2555.622	-0.271297	7.818342	120.445676	5.281132
125830.04456	32548191.861	5248152.736	2555.619	-0.264698	7.819879	120.444358	5.281129
125830.04956	32548192.151	5248152.637	2555.616	-0.256975	7.820106	120.443266	5.281126
125830.05456	32548192.441	5248152.537	2555.613	-0.253432	7.821207	120.441952	5.281123

Tab. 4.3: Auszug aus einem Flugzeugdatenfile mit 200 Hz zeitlicher Auflösung (021210_01.asc).

Tabelle 4.3 zeigt ein typisches Flugzeugdatenfile, das mit einer Headerzeile beginnt, auf die dann die eigentlichen Daten folgen. Jede Zeile entspricht einer zeitlich verorteten Flugzeugposition mit den dazugehörigen INS-Informationen.

Aus dem Header kann entnommen werden, um welche Daten es sich handelt. Die eigentlichen Daten zeigen mit welcher Genauigkeit die Flugzeugposition und die Orientierung des Flugzeugs vorliegen, und in welcher zeitlichen Auflösung. Fast alle Flugzeugdatenfiles haben eine zeitliche Auflösung von 200 Hz, wenige Files haben 50 Hz. Die meisten Flugzeugpositionen sind nicht direkt gemessen, sondern entsprechen der *best estimated trajectory*. Die Koordinaten sind in UTM (WGS84) projiziert. Die Zeit wird in Sekunden seit Wochenbeginn (Tab. 4.4) angegeben.

Wochentag	**Uhrzeit**	**Tag in der Woche**	**Anzahl der Sekunden seit Wochenbeginn**
Sonntag	00:00:00	1	0
Montag	00:00:00	2	86.400
Dienstag	00:00:00	3	172.800
Mittwoch	00:00:00	4	259.200
Donnerstag	00:00:00	5	345.600
Freitag	00:00:00	6	432.000
Samstag	00:00:00	7	518.400
Samstag	23:59:59	7	604.799
Sonntag	00:00:00	1	0

Tab. 4.4: Schema der zeitlichen Verortung aller Laserscannerdaten.

Beispiel: 207.953 Sekunden seit Wochenbeginn → Dienstag, 9h 45m 53s.

Eine Verknüpfung der einzelnen Flugzeugdatenfiles mit den dazugehörigen Laserpunkten ist über den Dateinamen des Flugzeugdatenfiles (Datum und Nummer) möglich. Die **Laserpunktdaten** jeder Befliegung liegen in einem Verzeichnis, dessen Name, gleich wie das Flugzeugdatenfile, sich aus dem Datum der Befliegung und einer fortlaufenden Nummer zusammensetzt (Abb. 4.8).
Die Dateinamen der Laserpunktdaten setzen sich aus einem Präfix „STR_“, einer Streifennummer, die vom Aufnahmesystem vergeben wird und einer Dateiendung, die angibt, ob es sich um *first* oder *last pulse* Daten handelt. Die Endung „ALL“ steht für *last* und „ALF“ für *first pulse*.

Beispiel: im Verzeichnis **021210_01** → **STR_114.ALL**
→ *last pulse* Daten des Befliegungsstreifens 114 vom 10.12.2002,
die Flugzeugposition ist in der Datei 021210_01.asc gespeichert.

Die Dateien der Laserpunkte haben, wie in Tabelle 4.5 ersichtlich, keine Headerzeile. Jede Zeile entspricht einem Messpunkt beginnend mit der zeitlichen Verortung. Darauf folgen die UTM-Koordinaten (East, North, Height) und der Intensitätswert in 8-Bit. Die Intensitäten können Werte zwischen 5 und 255 annehmen (vgl. LUTZ 2003, 11). Die zeitliche Auflösung beträgt 50.000 Hz.

207185.780820	32551495.16	5261535.22	441.58	5
207185.780860	32551493.36	5261535.04	441.60	17
207185.803600	32551499.54	5261533.77	441.53	8
207185.803660	32551502.40	5261534.06	441.66	5
207185.803760	32551507.12	5261534.51	441.46	6
207185.813900	32551507.89	5261533.79	441.54	8
207185.813980	32551503.92	5261533.39	441.58	12
207185.814100	32551498.16	5261532.80	441.59	10
207185.814120	32551497.25	5261532.71	441.62	5
207185.814260	32551490.62	5261532.03	441.57	5
207185.814280	32551489.53	5261531.93	441.65	10
207185.814320	32551487.75	5261531.74	441.56	10
207185.836580	32551483.96	5261529.53	441.63	5
207185.836620	32551485.92	5261529.73	441.70	8
207185.836720	32551490.60	5261530.18	441.58	7
207185.836920	32551500.13	5261531.12	441.58	5
207185.836940	32551501.02	5261531.21	441.60	19
207185.847260	32551506.86	5261530.97	441.56	6
...	...	...	...	...

Tab. 4.5: Auszug aus einem *first pulse* Datensatz (STR_1638.ALF).

Die unkomprimierten ASCII-Dateien verbrauchen sehr viel Speicherplatz (Tab. 4.6) und sind recht unhandlich. Das Öffnen und Betrachten der bis zu 1 GB großen Files kann sehr lange dauern und ist nicht mit jedem Texteditor möglich. Der Import mehrerer Punktdatenfiles in ein gängiges *Geographisches Informationssystem* (z.B. ESRI ArcMap) führt mit ziemlicher

Sicherheit zum Absturz.

Kampagne	Unterland (Vlbg.)		Gargellental (Vlbg.)		*Untersuchungsgebiet*	
	Files	Größe [GB]	Files	Größe [GB]	Files	Größe [GB]
Gesamt	421	84.7	154	32.7	30	8.7
First Pulse	204	41.1	76	16.2	14	4.2
Last Pulse	204	42.7	76	16.3	14	4.3
POS & IMU	13	0.9	2	0.2	2	0.2

Tab. 4.6: Anzahl der Files und Datenmenge (in Gigabyte) der einzelnen Befliegungskampagnen im Vorarlberger Unterland und im Gargellental, ein Seitental des Montafons. Die Rohdaten des Untersuchungsgebietes entsprechen allen Flugstreifen des Unterlandes, die durch diese 500 x 500 m laufen. Es wurden die Datensätze in UTM (WGS84) zur Zählung herangezogen.

4.3.2 Festlegung eines Standards

Die Anordnung der Datenfiles (Verzeichnisstruktur) und der innere Aufbau der Dateien sind für die weitere Verarbeitung, v.a. für das automatisierte Auslesen (siehe Kapitel *„6 Datenimport“*) der Daten, von großer Bedeutung. Die logische Verknüpfung der Daten (z.B. über das Befliegungsdatum oder über die zeitliche Verortung der einzelnen Messdaten) spielt dabei eine wichtige Rolle. Die Festlegung eines Standards für das Datenformat der „Rohdaten“ ist der erste Schritt bei der Entwicklung eines Datenverwaltungssystems.
Die oben vorgestellte Datenstruktur entspricht dem Format, das die Firma TopScan dem Land Vorarlberg ausgeliefert hat. Für zukünftige Befliegungskampagnen wurde mit der Firma TopScan dieses Format als Standardformat festgelegt (mündl. Mitteilung LINDENBERGER, 13.08.2004).

4.4 Zusammenfassung

Das Kapitel *„4 Datenerfassung und Datengrundlagen“* zeigt, welche Faktoren das Ergebnis einer flugzeuggestützten Laserscannermessung beeinflussen. Ferner wird die Struktur, der bei dieser Diplomarbeit verwendeten Daten, aufgezeigt. Bei der Betrachtung der vorliegenden „Rohdaten“ bleiben folgende Fragen im Raum stehen:

- Mit welcher Software können die Punktdaten visualisiert und verarbeitet werden?
- Was ist, wenn man nicht mit den Punktdaten des gesamten Streifens, sondern nur mit den Daten eines 100 x 100 m Bereiches arbeiten will?
- Was ist, wenn sich das Untersuchungsgebiet im Überlappungsbereich zweier Streifen befindet? Welche Streifen sind das?
- Wie findet man für ausgewählte Punkte die entsprechende Flugzeuginformation?
- Wie bekommt man die Punkte für eine bestimmte Profillinie oder ein vorgegebenes

Untersuchungsgebiet (z.B. Gletscherfläche)?

Die oben genannten Fragen zeigen, dass das Arbeiten mit den Rohdaten nahezu unmöglich ist. Im Laufe dieser Diplomarbeit sollen diese Fragen beantwortet werden und es soll dem Leser klar werden, wie wichtig und notwendig die Entwicklung eines Datenmanagementsystems ist, vor allem wenn man sich direkt mit den Punktinformationen der Laserscannermessung auseinandersetzt.

5 Entwicklung des Informationssystems

5.1 Einleitung

Das Kapitel *„5 Entwicklung des Informationssystems“* soll dazu dienen, den Leser in die Vorgehensweise bei der Entwicklung des Informationssystems einzuführen, den Aufbau des Systems zu skizzieren sowie die verwendete Software vorzustellen. Am Ende dieses Kapitels soll klar sein, wie das System in seinen Grundzügen funktioniert, welche Software mit einfließt, um was es sich bei dieser Software handelt und für welche Aufgaben die einzelnen Programme herangezogen werden. Die Frage „Warum gerade diese Programme und nicht Konkurrenzprodukte zum Zuge kommen?“ lässt sich in einem Satz beantworten: Es wird ausschließlich mit freier Software (*free & open source*) gearbeitet, wobei die verwendeten Programme, die jeweils Führenden in ihrem Bereich sind. „Führend“ bedeutet, dass sie am meisten angewendet werden und dass sie die größten Entwicklergemeinden vorzuweisen haben. Es kann von einer fortschreitenden und rasanten Entwicklung dieser Software ausgegangen werden.

5.2 Methodik

Zu Beginn jeder Systementwicklung steht die Frage „**Was soll mein geplantes System eigentlich leisten?**“ (vgl. JECKE et. al. 2003, 175ff.). Eine kurze und prägnante Antwort auf diese Frage wäre, dass ein System benötigt wird, das die Verwaltung und Analyse von Rohdaten der Laserscannermessung ermöglicht. Im Vordergrund der Diplomarbeit steht die Konzeption eines Prototyps. Darin enthalten sind die Festlegung der Systemkomponenten und die Definition der Schnittstellen. Mit Hilfe eines Anforderungskataloges (Tab. 5.1) an das System (≈ Eigenschaften und Funktionalität des Systems) kann ein erster Entwurf erstellt werden. Der Arbeitsablauf bei der Entwicklung des Informationssystems gliedert sich in mehrere Phasen (Tab. 5.2).
Bisherige Arbeiten am Institut für Geographie in Innsbruck beschäftigten sich mit der rasterbasierten Auswertung der Laserscannerdaten (vgl. LUTZ 2003). Aufgrund von Änderungen im Format der Rohdaten und aufgrund der fortschreitenden Entwicklung der Soft- und Hardware wird ein vektorbasiertes Modellierungsverfahren gewählt. Folgende Gründe sprechen für eine vektorbasierte Modellierung:

- Eine Verknüpfung zwischen Laserpunkten und IMU-Daten ist durch das vorliegende Datenformat der Rohdaten gegeben (Verknüpfung über die Zeit).
- Die Rekonstruktion der Aufnahmegeometrie ist ein Paradefall für die Anwendung der Vektorrechnung und der Trigonometrie.
- Das Vektordatenmodell ermöglicht eine maximale räumliche Genauigkeit.

- Jeder Laserpunkt – auch in Überlappungsbereichen der Flugstreifen – behält seine eigene Aufnahmegeometrie.
- Das Vektordatenmodell erlaubt die Anbindung von zusätzlichen Informationen an die exakte räumliche Position.
- Die fortschreitende Entwicklung von räumlichen Datenbanken erlaubt die Verarbeitung von großen Datenmengen – bei denen kommerzielle GIS-Programme längst aussteigen – in einer vertretbaren Zeit.
- Durch die hohe Punktdichte der Laserscannerdaten – es kann beinahe von einer flächigen Repräsentation gesprochen werden – ist eine Umwandlung in das Rasterdatenmodell durch einfache Interpolationsverfahren jederzeit gegeben. Wichtig sind der Export im Rasterformat für das Arbeiten in Fernerkundungssoftware und der Vergleich und die Evaluierung der Ergebnisse mit vorhandenen Datensätzen.
- Das Vektordatenmodell arbeitet direkt mit der Punktwolke. Analysen der Punktwolke treten immer mehr in das Interesse der internationalen Forschung (z.B. PFEIFER & WINTERHALDER 2004)
- Die Daten werden in einer räumlichen Datenbank verwaltet und können dort direkt als Objekte angesprochen werden. Die Daten werden dort prozessiert, wo sie gespeichert sind. Aufwändige Datentransfers sind nicht notwendig. Somit kann Zeit gespart werden, zusätzliche Fehlerquellen werden vermieden.

Eigenschaften	Funktionalität
• freie (und *open source*) Software • plattformunabhängiger Zugang • 4-dimensional: zeitliche (1-D) und räumliche Verortung (3-D) der Daten • multitemporal: durch die zeitliche Verortung gegeben • Speicherung der Daten in maximaler Genauigkeit (Vektor-Datenmodell) • ansprechende Performance bei großen Datenmengen (mehrere GByte) • modularer Aufbau: Erweiterungsmöglichkeit bei Bedarf • benutzerfreundliche Bedienung (Graphical User Interface)	• Import von Rohdaten der Laserscannerbefliegung (Punkte, IMU) • schnelle Selektion (Abfrage) der Daten über Raum, Zeit und Attribute (z.B. Signalintensität) • Berechnung (Rekonstruktion) der Aufnahmegeometrie: Verbindung zwischen Laserpunkt und Flugzeugposition • multitemporale Analysen (Veränderung des Raumes über die Zeit) • Funktionen eines GIS • Funktionen einer Statistiksoftware • Berechnung von streifenfreien Intensitätsrastern • Berechnung von Gelände- und Oberflächenmodellen • Attributierung der Laserpunkte über die Ergebnisse von Fernerkundungssoftware • Informationen über die Qualität des Datensatzes • Unterstützung des Raster-Datenmodells • Unterstützung von Metadaten • Unterstützung von unterschiedlichen Kartenprojektionen • Export der Daten und Analyseergebnisse in mehreren Formaten • Visualisierung der Daten und Ergebnisse

Tab. 5.1: Anforderungskatalog: Eigenschaften und Funktionen des Systems.

Die Visualisierung der Punktwolke und der daraus abgeleiteten Informationen ist nur in sehr wenigen kommerziellen (und sehr teuren) Softwareprodukten, die eine große Rechnerkapazität voraussetzen, zufriedenstellend gelöst (vgl. INTERNET: Leica Geosystems). Die Visualisierung und Interpretation der Ergebnisse muss bis dato den Umweg ins Rasterdatenmodell nehmen. Ein Raster stellt eine Vereinfachung und Reduktion der Daten dar. Die Prozessierung von Rasterdaten ist durch die einfache Topologie (festgelegte Nachbarschaftsbeziehungen) dieses Datenmodells sehr verbreitet. Es liegen viele Algorithmen für die Ableitung von Informationen (Datenveredelung) aus Rasterdaten vor. Sie arbeiten sehr robust und schnell. Diese Diplomarbeit stellt einen Versuch dar, Analysen in der Punktwolke, im Vektordatenmodell, mit freier Software und einfacher Hardware (Desktop PC, siehe *„5.4.1 Hardware“*) zu versuchen.

Erhebung des Ist-Zustandes
- Sichtung der Laserscannerdaten und Ableitung der Logik in der Datenstruktur - Erstellung eines Anforderungskataloges (siehe Tab. 5.1) - Sichtung vorhandener Software für Laserscannerdaten: was für Software gibt es und was kann sie? - Suche nach geeigneter Entwicklungs- und Anwendungssoftware ↓
Aufbau der Infrastruktur
- Installation des Betriebssystems - Installation der einzelnen Softwarepakete - Erweiterung der Hardware (Arbeits- und Festplattenspeicher) ↓
Entwicklung der Funktionalität
- Festlegung eines Standards für die Auslieferung der Rohdaten mit TopScan GmbH. (siehe Kapitel *„4 Datenerfassung und Datengrundlagen“*) - Entwurf der Datenbankstruktur (siehe Kapitel „*Datenverwaltung und -speicherung*“) - Importprogramm: Einlesen der Rohdaten in die Datenbank: gleichzeitiger Aufbau der Relationen und Ableitung erster Informationen (siehe Kapitel „*Datenimport*“) - absolute zeitliche Verortung der IMU- und Laserpunktdaten - Interpolation der Flugzeugposition: jedem Laserpunkt wird die Position des Flugzeugs (3D-Koordinaten) während der Aufnahme zugeteilt - Berechnung der relativen Flughöhe (Höhe über Grund) für jede Flugzeugposition - Berechnung der durchschnittlichen Flugzeuggeschwindigkeit für jeden Flugstreifen - Berechnung der Aufnahmegeometrie (Länge des Laservektors: *range*, Geländeneigung am Punkt, Exposition, Einfallswinkel des Laserstrahls, *footprint*) - Anbindung des Geographischen Informationssystems GRASS: Interpolation von Rastern - Anbindung der Statistiksoftware R: Korrelationsanalyse - Die Berechnung der deskriptiven Statistik (z.B. Anzahl, Mittelwert, Standardabweichung) und die Selektion der Daten anhand von Attributen oder Geometrie sind in der Datenbank gegeben ↓
Export und Visualisierung
- Auswahl der Exportformate (siehe Kapitel *„9 Datenexport und Visualisierung“*) - Visualisierung der Daten im GIS oder in der Statistiksoftware ↓
Graphical User Interface
- Entwicklung einer Arbeitsoberfläche und Anbindung von gewissen Funktionen für Testzwecke (siehe Kapitel *„9 Datenexport und Visualisierung“*) ↓
Performancetests und Tuning
- Verbesserung der Rechenzeiten durch Softwaretuning: Anpassung des Betriebssystem, des Datenbankservers (siehe Kapitel *„10 Evaluierung und Ausblick“*) und Optimierung der Algorithmen/Funktionen

Tab. 5.2: Arbeitsschritte bei der Entwicklung des Prototypens eines Informationssystems für Laserscannerdaten.

5.3 Systemkomponenten

Der Entwurf des Systems beinhaltet mehrere Komponenten, die nicht unabhängig von einander arbeiten, sondern deren Arbeitsabläufe sehr stark in einander übergehen. Das heißt, ein Arbeitsablauf gliedert sich in mehrere Prozesse, die gleichzeitig in einem oder mehreren Systemkomponenten ablaufen können. Die drei Hauptkomponenten sind das GIS, die räumliche Datenbank und die Statistiksoftware. In Abbildung 5.1 ist der Aufbau des Systems skizziert. Die Hauptkompontenten stehen eigentlich in einer Dreiecksbeziehung zueinander. Das heißt, dass jede Komponente auf die Daten oder Funktionalität der anderen zwei Komponenten zugreifen kann. Abbildung 5.1 zeigt eine Vereinfachung dieser Beziehung. Im Vordergrund steht der Zugriff des GIS (lesen und schreiben) auf die Daten der räumlichen Datenbank und der Transfer der Funktionalität der Statistiksoftware in die räumliche Datenbank. Die Schnittstellen zwischen den einzelnen Komponenten werden im Kapitel *„8 Datenverarbeitung“* detailliert beschrieben.

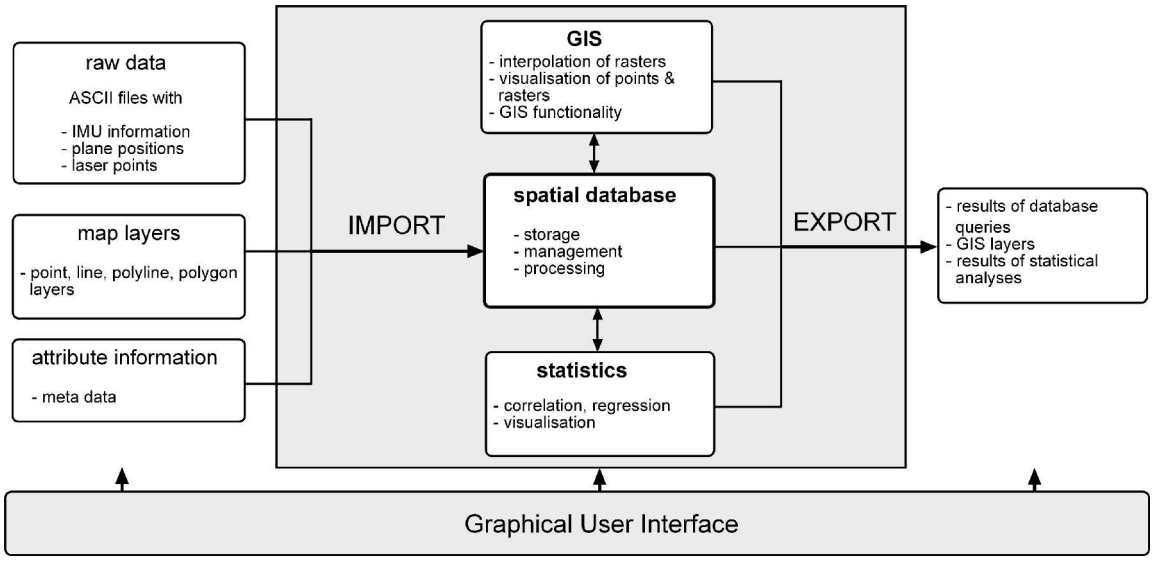


Abb. 5.1: Komponenten des Informationssystems (eigener Entwurf).

Der Datenimport wird mit der Programmiersprache (Python) realisiert (vgl. Kapitel *„7 Datenimport“*). Metadaten und GIS-Daten können direkt in die Datenbank importiert werden. Die räumliche Datenbank unterstützt z.B. den Import von ESRI s*hapefiles* (INTERNET: ESRI). Der gesamte Datenimport wird zentral über die Programmiersprache Python geregelt. Damit sind ein korrekter Import in das Datenbankschema und die Aufrechterhaltung korrekter Beziehungen zwischen den Datensätzen gewährleistet. Im Vergleich zum Datenimport kann der Datenexport direkt aus allen drei Hauptkompontenten betätigt werden. Die Datensätze der Datenbank können als Tabellen, aber auch als Vektoren (*shapefile*, SVG) ausgegeben werden. Im GIS stehen alle Exportvarianten, die das GIS unterstützt, zur Verfügung (z.B. ESRI GRID, Bildformate). Die Statistiksoftware bietet

vor allem die Möglichkeit die Zusammenhänge zwischen den Daten zu visualisieren und in verschiedenen Formaten (z.B. PDF, JPEG) zu exportieren. Berechnungen werden in allen drei Komponenten durchgeführt.

5.4 Entwicklungsumgebung

5.4.1 Hardware

Die verwendete Hardware entscheidet zum Großteil über die zu erwartende Performance des Informationssystems. Ein kleiner Teil der Performance kann durch Software-Tuning herausgeholt werden. Während der Arbeiten zur Diplomarbeit wurde der Computer von 512 MB auf 1 GB RAM aufgerüstet, was zu einer beachtlichen Verkürzung der Rechenzeiten führte. Für die Bewertung der Performancetests (siehe Kapitel *„10 Evaluierung und Ausblick“*) ist es wichtig, genau über die Hardware des Rechners Bescheid zu wissen. In Tabelle 5.3 sind die wichtigsten Hardwarekomponenten aufgelistet.

Computermodell	Dell OptiPlex GX260
Prozessor (CPU)	Intel Pentium 4 mit 1,8 GHz
Hauptspeicher (RAM)	1 GByte
Video	Intel 845G PCI Accelerated SVGA
Festplatte	160 GByte Maxtor

Tab. 5.3: Systemkomponenten des verwendeten Rechners.

5.4.2 Betriebssystem

Die gesamte Software eines Computers wird in zwei Bereiche untergliedert. Diese zwei Bereiche kann man sich als Ebenen (bzw. Schichten) vorstellen, die unterschiedliche Aufgaben zu übernehmen haben und die mit ihren Nachbarebenen kommunizieren, (Abb. 5.2). Das Betriebssystem stellt dabei die unterste Schicht dar.

„Ein Betriebssystem regelt den Zugriff auf die Hardware des Computers und verwaltet Daten, die im Rechner gespeichert sind. Es stellt eine Umgebung bereit, in der Benutzer Programme ausführen können.“

(WEIGEND 2004, 22)

Eine Ebene über dem Betriebssystem stehen die System- und Anwendungsprogramme. Solche Programme sind als zusätzliche Schnittstelle zwischen Benutzer und Betriebssystem zu sehen. Ein Systemprogramm ist z.B. der Python-Interpreter, der vom Benutzer geschriebene Python-Programme „interpretiert“ und das Betriebssystem beauftragt, die Befehle des Python-Programms auszuführen. Ein Anwendungsprogramm ist z.B. ein

Textverarbeitungsprogramm. Es zielt auf eine bestimmte Problemstellung ab, das Erstellen, Verändern und Speichern von Textdokumenten. Anwendungssoftware ist speziell für die Bedürfnisse des Benutzers geschrieben und soll die Arbeit am Computer erleichtern.
Die Entwicklung des Informationssystems soll die Arbeit mit Laserscannerdaten am Computer überhaupt erst ermöglichen und wird in der Zukunft vielleicht als benutzerfreundliches Anwendungsprogramm dienen.

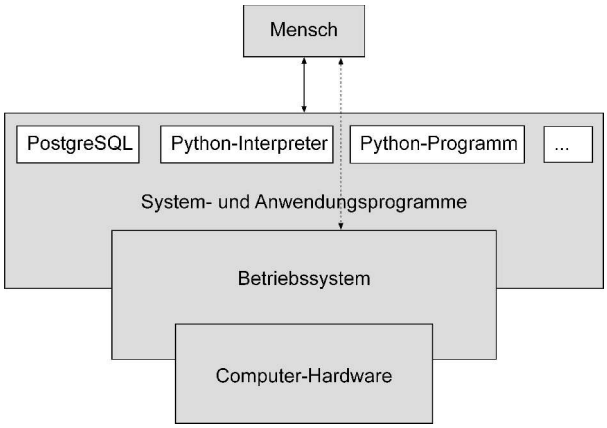


Abb. 5.2: Komponenten eines Computersystems (verändert nach WEIGEND 2004, 22).

Im Rahmen der Diplomarbeit wird auf einem Red Hat Linux Betriebssystem gearbeitet (vgl. www.redhat.com): **Red Hat Fedora Core release 2 (Tettnang) Kernel 2.6.6-1.435.23**
Die offene Struktur (siehe unten: *open source*) des gewählten Betriebssystems sowie das perfekte Zusammenspiel mit den restlichen Programmen (z.B. Datenbankserver) sind ein Pluspunkt von Linux. Ein wesentlicher Nachteil ist die lange Einlernphase in Linux, speziell für „Windows-Jünger“. Das freie Betriebssystem Linux ist wesentlich schwieriger zu handhaben als die Konkurrenzprodukte von Microsoft. Dafür bietet es für den versierten Benutzer sehr viele Einstellungs- und Manipulationsmöglichkeiten, die bei der Entwicklung von Software von großem Vorteil sind. Viele Probleme mit Linux werden durch den Zusammenhalt und die Hilfe innerhalb der Linux-Community wieder ausgeglichen (z.B. *newsgroups, mailing lists*, *online tutorials*, *online manuals*, usw.).

5.4.3 Datenbanksystem

5.4.3.1 PostgreSQL

PostgreSQL ist ein objektrelationales Datenbankmanagementsystem (ORDBMS), das auf POSTGRES basiert. Die Entwicklung von POSTGRES wurde 1986 an der Universität von Kalifornien in Berkeley begonnen. PostgreSQL ist ein Open-Source-Nachfolger von POSTGRES und unterstützt die *Structured Query Language* (SQL) (EISENTRAUT 2003, 19ff.). SQL wird im Punkt 5.4.6.1 näher beschrieben.

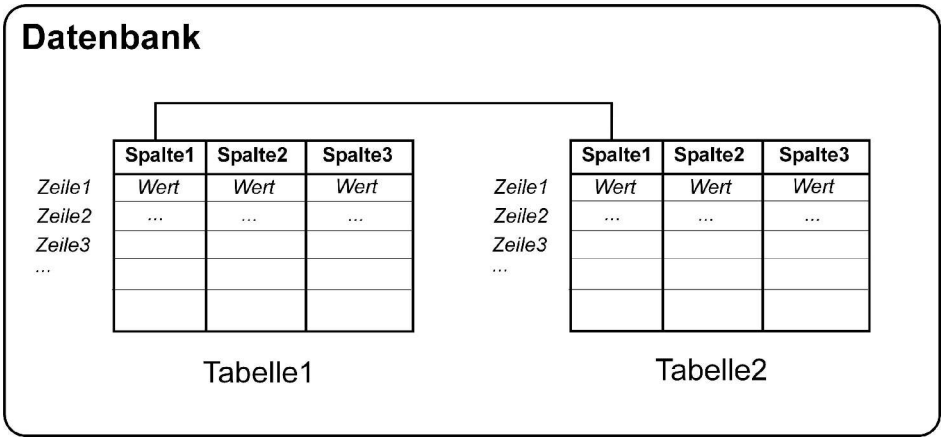


Abb. 5.3: Schema einer Datenbank mit zwei Tabellen (eigener Entwurf).

Ein relationales Datenbankmanagementsystem ist ein System, das Daten in Relationen speichert. „Relation“ ist ein mathematischer Begriff für eine Tabelle. Die Daten werden in Tabellen gespeichert, die aus Spalten und Zeilen bestehen (Abb. 5.3). Jede Spalte hat einen bestimmten Datentyp (z.B. Text, Datum, Integer) und die Anordnung (Reihenfolge) der Spalten in jeder Zeile ist festgelegt. Die Reihenfolge der Zeilen hängt vom Zeitpunkt ihrer Speicherung ab. Bei jeder SQL-Abfrage kann die Sortierung der Zeilen zusätzlich angegeben werden. Jede Tabelle hat einen Namen und kann über diesen angesprochen werden. Zwischen den Tabellen in einer Datenbank können Verbindungen und Abhängigkeiten aufgebaut werden (*constraints*). Eine Sammlung von Tabellen wird Datenbank genannt (Abb. 5.4). Mehrere Datenbanken, die unter einem PostgreSQL-Server laufen, werden in einem so genannten Datenbank-Cluster zusammengefasst (EISENTRAUT 2004, 35f.).

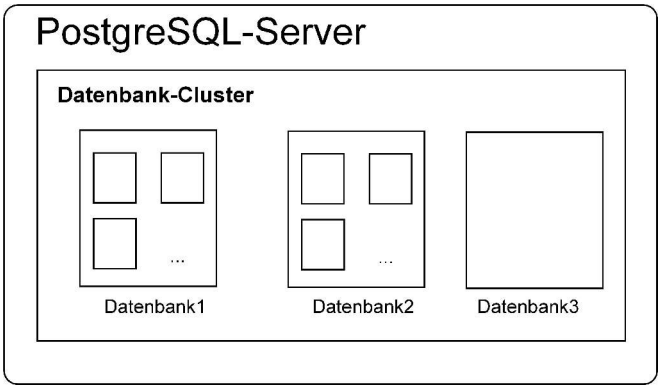


Abb. 5.4: Ein PostgreSQL-Server kann mehrere Datenbanken (=Cluster) verwalten (eigener Entwurf).

PostgreSQL ist *open source*. Das bedeutet, dass jeder Einblick in den Quellcode des Programms haben kann, ihn verändern und weitergeben darf. WHEELER (2004) zeigt in

seinem Artikel die wesentlichen Unterschiede zwischen *open source* und proprietärer Software auf. Die Einsicht in den Quelltext ist vor allem bei wissenschaftlichen Fragestellungen, bei denen die Nachvollziehbarkeit der Ergebnisse gewährleistet sein muss, ein großer Pluspunkt. So können alle Funktionen und Algorithmen eines *open source* Programms auf ihre Arbeitweise hin überprüft werden und gegebenenfalls umgeschrieben bzw. verbessert werden. Freie / *open source* Software unterliegt sehr wohl einer Lizenz, die dazu dient, die Grundsätze der *open source* Software zu schützen (Tab. 5.4). Die Definition von *open source* ist auf www.opensource.org zu finden.

0. freedom:	The freedom to run the program, for any purpose.
1. freedom:	The freedom to study how the program works, and adapt it to your needs.
2. freedom:	The freedom to redistribute copies.
3. freedom:	The freedom to improve the program, and release your improvements to the public, so that the whole community benefits.

Tab. 5.4: Die vier Freiheiten der *free / open source* Software (NETELER & MITASOVA 2002, 2).

PostgreSQL unterliegt der BSD Lizenz. Diese Lizenz ist sehr ähnlich wie die GNU General Public License (www.gnu.org), die wohl am häufigsten verwendete Lizenz für *open source* Software.

> „*People often ask why PostgreSQL is not released under the GNU General Public Licence. The simple answer is because we like the BSD licence and do not want to change it.*“
>
> (http://www.postgresql.org/licence.html, 2004)

PostgreSQL verwendet ein Client/Server-Modell (Abb. 5.5). Das Datenbankserverprogramm von PostgreSQL heißt *postmaster* und wird standardmäßig beim Start des Computers als Prozess gestartet. Es überwacht, akzeptiert und führt die Anfragen vom Client aus. Die Anfragen vom Client erfolgen in der Sprache SQL. Im Rahmen der Arbeiten zur Diplomarbeit wurde von verschiedenen Clients auf die Datenbank zugegriffen. Zum Beispiel gibt es Module für Python, die einen direkten Datenbankzugriff über die TCP/IP-Verbindung erlauben. Von GRASS aus können SQL-Statements ausgeführt und die Tabellen der Datenbank direkt angesprochen werden. Das wohl bekannteste Datenbankverwaltungsprogramm, auch ein Client, ist *pgAdmin* (www.pgadmin.org). *PgAdmin* ist eine graphische Benutzeroberfläche, die zur Verwaltung der Tabellen, Sichten, Funktionen usw. herangezogen wird. Die Client/Server-Architektur von PostgreSQL hat weiters zum Vorteil, dass sich Client und Server auf unterschiedlichen Rechnern befinden können und dass ein gleichzeitiger Zugriff von mehreren Clients (*multi-user* Betrieb) möglich ist. Die Daten befinden sich stets auf dem Server, der auch die ganzen Arbeitsprozesse

ausführt. Die Rechenleistung des Clients ist nicht entscheidend, da er nur die „Arbeit“ des Servers mit SQL-Statements steuert und selber keine Rechenoperationen ausführt. Der Datenimport/-export sollte jedoch direkt auf dem Server stattfinden, da die Verbindung zwischen Client und Server bei großen Datenmengen ein Hindernis darstellt (EISENTRAUT 2003, 29f.).

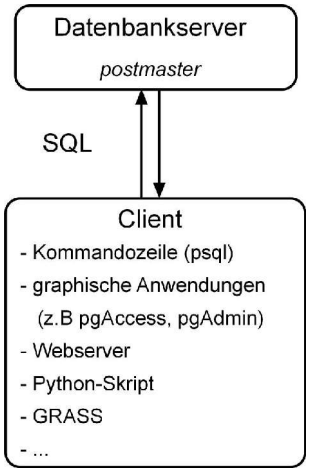


Abb. 5.5: Client/Server-Architektur von PostgreSQL (eigener Entwurf).

Die aktuelle Version von PostgreSQL kann von der offiziellen Homepage (www.postgresql.org) heruntergeladen werden. Diese Homepage beinhaltet ebenfalls eine sehr gute Installationsanleitung und ein Online-Tutorial. PostgreSQL läuft unter den meisten Betriebssystemen (Linux, Windows, MacOS, usw.). Ein guter Überblick der Fähigkeiten (*features*) von PostgreSQL ist auf www.postgres.de oder der offiziellen Homepage zu finden. Das Datenbanksystem PostgreSQL wurde aus vielen Gründen für die Anwendung im Informationssystem ausgewählt:

- Es ist *open source* Software und die neueste Version ist jederzeit kostenlos verfügbar.
- Die Client/Server-Architektur bringt entscheidende Vorteile für einen weiteren Ausbau des Informationssystems (z.B. *multi-user* Betrieb)
- PostgreSQL hat eine große Entwicklergemeinde und wird von vielen Firmen und wissenschaftlichen Institutionen bereits seit langer Zeit erfolgreich angewandt (z.B. BASF, siehe auch INTERNET: PostgreSQL Case Studies).
- Die Unterstützung von prozeduralen Sprachen (vgl. PL/pgSQL) und die einfache Anbindung des Datenbanksystems in externen Applikationen (z.B. GRASS, R oder Python-Programme) sind ein Kernpunkt für die Entwicklung des Informationssystems.
- PostgreSQL hat nahezu keine Einschränkungen (Limits) in puncto Größe der Datenbank (Tab. 5.5), was für das Arbeiten mit Laserscannerdaten sehr wichtig ist.
- Die Performance kann durch eingebaute Funktionen ständig überprüft und anschließend verbessert werden. Ein Tuning des Datenbanksystems, das die optimale Leistung aus der vorhandenen Hardware holt, ist möglich.

- PostGIS (siehe unten) macht PostgreSQL zu einer räumlichen Datenbank. PostgreSQL/PostGIS kann geometrische Datentypen verwalten und räumliche Abfragen und Berechnungen durchführen.

Maximum size for a database:	unlimited (4 TB databases exist)
Maximum size for a table:	16 TB on all operating systems → 23900 CDs
Maximum size for a row:	1.6 TB
Maximum size for a field:	1 GB
Maximum number of rows in a table:	unlimited
Maximum number of columns in a table:	250 - 1600 depending on column types
Maximum number of indexes on a table:	unlimited

Tab. 5.5: *Limitations of PostgreSQL* (http://www.postgresql.org/users-lounge/limitations.html).

5.4.3.2 PostGIS

PostGIS wird von der *Refractions Research Inc.* (postgis.refractions.net) als Zusatztool (*extension*) für das Datenbanksystem PostgreSQL entwickelt. PostGIS macht PostgreSQL zu einer räumlichen Datenbank. Welche Zusatzfunktionen bringt die Installation von PostGIS?

- Die Definition von zusätzlichen geometrischen Datentypen (z.B. Multipolygone) ist möglich.
- PostGIS stellt viele Funktionen für die räumliche Analyse (GIS-Grundfunktionalität) zur Verfügung (Tab. 5.6). Die PostGIS-Funktionen sind als SQL-Funktionen definiert und somit auch gleich zu verwenden. Die SQL-Funktion linkt auf die PostGIS-Bibliothek.

Funktion	**Beschreibung**		**Modul**
isSimple(geometry)	überprüft, ob eine Geometrie *simple* ist (siehe *OGC Simple Feature Spec.*)		GEOS
intersects(geometry, geometry)	überprüft, ob zwei Geometrien sich schneiden		GEOS
convexhull(geometry)	gibt die konvexe Hülle einer Geometrie zurück		GEOS
length(geometry)	Länge einer Linie		
transform(geometry, integer)	Transformation der Koordinaten in ein anderes Bezugssystem		Proj4

Tab. 5.6: Einige ausgewählte OpenGIS-Funktionen von PostGIS (vgl. RAMSEY, 2004).

- Die *Proj4 reprojection library* ermöglicht die Verwaltung von Projektionen sowie das Projizieren und Transformieren der räumlichen Daten. Proj4 (www.remotesensing.org/proj) muss zusätzlich am Rechner installiert sein, damit die Funktionen von PostGIS genützt werden können.
- PostGIS unterstützt *GEOS* (geos.refractions.net), das wie Proj4 ein eigenständiges Programm ist. Ist GEOS zusätzlich zur normalen PostGIS-Installation am Rechner installiert, können die GEOS-Funktionen (Tab. 5.6) über PostGIS ausgeführt werden.
- PostGIS verfolgt die *OpenGIS Simple Feature Specification for SQL*. Das OpenGIS Consortium (OGC) erarbeitet zusammen mit den Firmen ESRI, IBM, Oracle, MapInfo und Informix einen Standard für Geometriedatentypen. Die Spezifikation regelt, was *Simple Features* sind (Abb. 5.6), wie sie in der Datenbank abgespeichert werden (Abb. 5.7) und welche Funktionen auf diese Datentypen angewandt werden können (vgl. Tab. 5.6).

> „*This specification describes a standard set of SQL Geometry Types based on the OpenGIS Geometry Model, together with the SQL functions on those types.*“
>
> (OPENGIS CONSORTIUM 1999, 1-1)

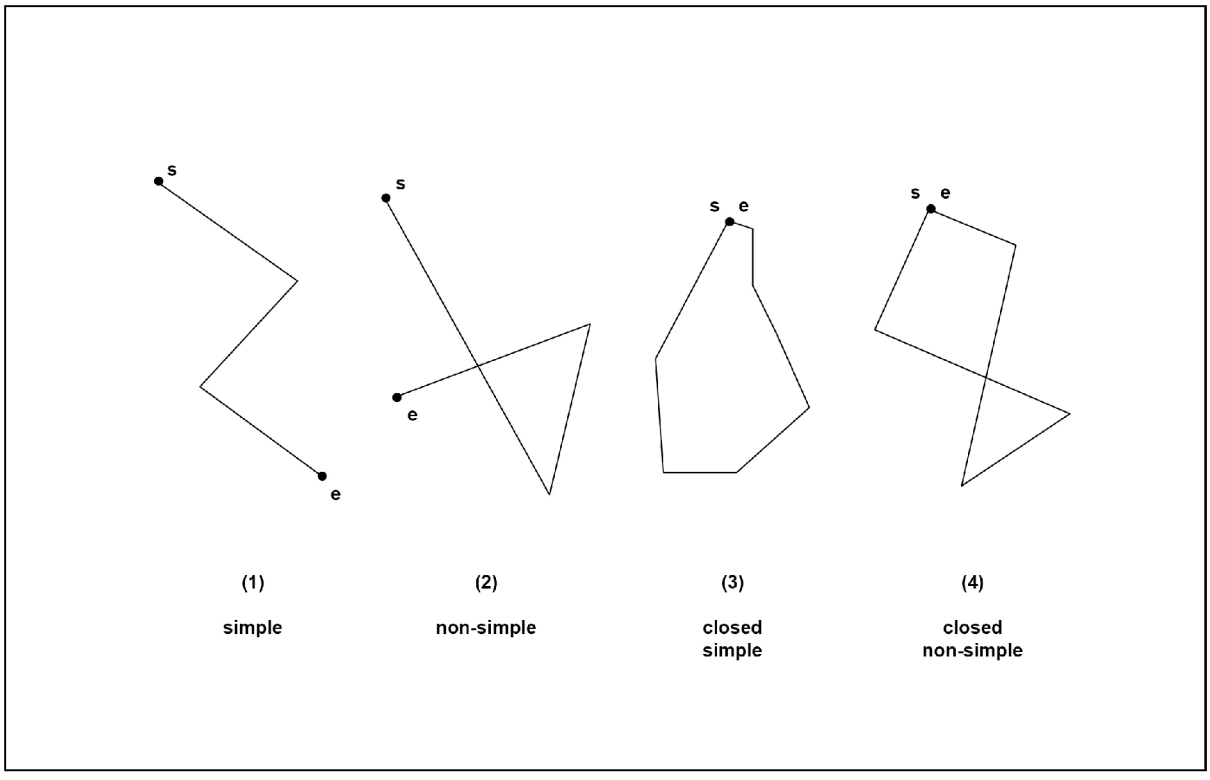


Abb. 5.6: Einfache und nicht-einfache Linien (s*imple and non-simple linestrings*) laut OGC (1999, 2-6).

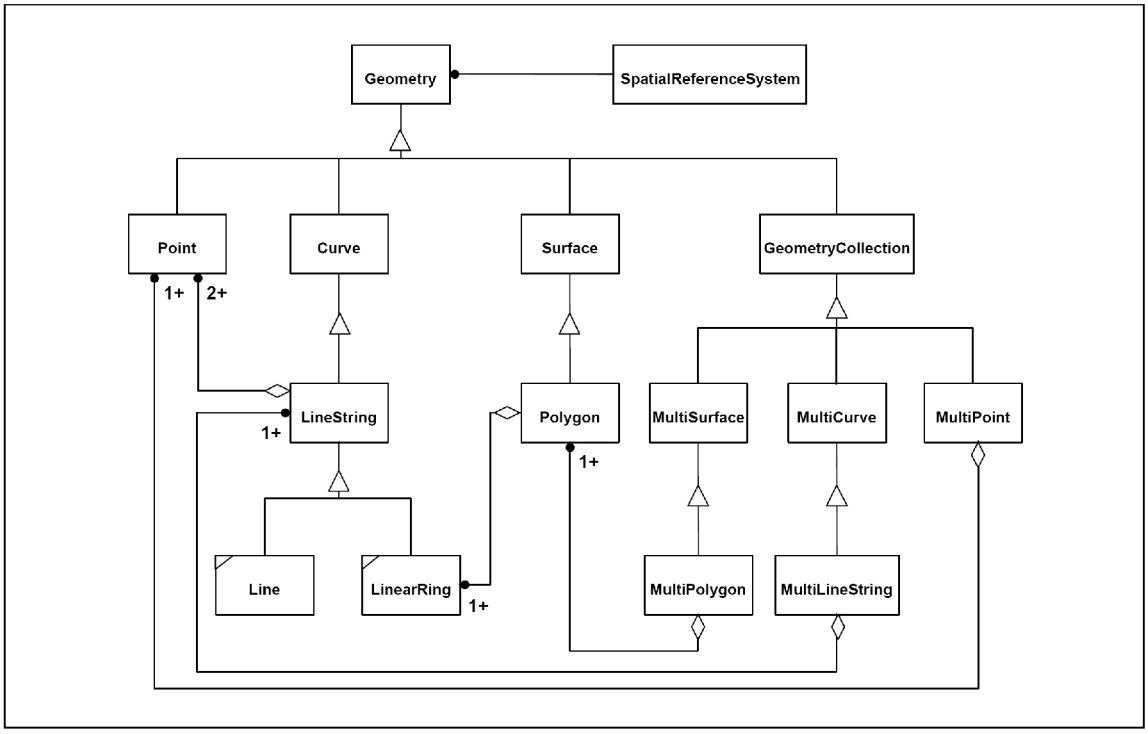


Abb. 5.7: Hierarchie der OGC-Geometrietypen (*Geometry Class Hierarchy*) (OGC 1999, 2-2).

- PostGIS verwendet räumliche Indices (*GiST-based R-Tree*). Indices helfen Abfragen in einer Datenbank zu beschleunigen, indem zusätzliche Informationen über die Daten abgespeichert werden. Mit Hilfe dieser zusätzlichen Informationen kann eine Abfrage viel gezielter und somit schneller ablaufen.
- Die Geometrie wird in einer eigenen Spalte (*AddGeometryColumn*) gespeichert. Jede Zeile einer Tabelle entspricht einem Objekt. Die Daten in Nicht-Geometriespalten entsprechen den Attributen des Objekts. Die Geometrie ist in der Geometriespalte gespeichert (Tab. 5.7).
- PostGIS ist *open source* und unterliegt der GNU General Public License. Die neueste Version kann von postgis.refractions.net heruntergeladen werden.

idscanpoint	idstrip	time	pulse	intens	geom_point
1	1	207185.780820	2	5	SRID=-1;POINT(32551495.14 5261535.23 441.59)
2	1	207185.780860	2	17	SRID=-1;POINT(32551493.34 5261535.04 441.63)
3	1	207185.803600	2	8	SRID=-1;POINT(32551499.51 5261533.78 441.59)
4	1	207185.803660	2	5	SRID=-1;POINT(32551502.38 5261534.06 441.66)
5	1	207185.803760	2	6	SRID=-1;POINT(32551507.09 5261534.52 441.53)

Tab. 5.7: Auszug aus einer Tabelle mit Geometrie. Jede Zeile ist ein Objekt.

Die Anleitung zu PostGIS von RAMSEY (2004) bezieht sich auf die aktuelle Version von PostGIS (0.9). Die Installation und einfache Anwendungen sind ausreichend erklärt. Anspruchsvollere räumliche Analysen werden jedoch nicht beschrieben und anschauliche Beispiele fehlen. Bei Problemen kann man sich über die PostGIS *Mailing List* direkt an die Entwickler wenden und bekommt in der Regel am gleichen Tag noch eine Antwort.

5.4.4 Geographisches Informationssystem GRASS

Das GIS GRASS (Geographic Resources Analysis Support System) wird im Informationssystem für Laserscannerdaten hauptsächlich für die Interpolation von Rastern aus den Vektoren der Datenbank eingesetzt. GRASS ist ein komplettes GIS, d.h. die Verwaltung, die Verarbeitung und die Visualisierung von Vektor- und Rasterdaten werden unterstützt.

> „*GIS is often described as integration of data, hardware, and software designed for management, processing, analysis and visualization of georeferenced data*."
>
> (NETELER & MITASOVA 2002, 1)

Viele Funktionen, die in der Regel ein Geographisches Informationssystem (GIS) übernimmt, können schon in der Datenbank durch PostGIS übernommen werden, wie z.B. Distanz- und Bufferfunktionen.
GRASS GIS läuft u.a. auf Linux, Windows, SUN-Solaris, Irix und MacOS X. GRASS ist ein Paradebeispiel einer *open source* Software (GNU GPL). Gerade im Bereich der Geographischen Informationssysteme ist die Kenntnis über die Algorithmen der Funktionen sehr nützlich. Wissenschafter und GIS-Begeisterte auf der ganzen Welt entwickeln GRASS ständig weiter. BROVELLI et. al. (2002) entwickelten einen Algorithmus für die Generierung von Digitalen Geländemodellen aus Laserscannerdaten und implementierten diesen in GRASS.
GRASS wurde von 1982 bis 1995 von den *U.S. Army Construction Engineering Research Laboratories (CERL)* entwickelt und half der U.S. Army bei Planungen (*land management and environmental planning*). Ende der 1980er wurde der gesamte Quellcode von den CERL im Internet zur Verfügung gestellt. Seit 1997 gibt es das „GRASS Development Team" mit Sitz in Trient (*Istituto Trentino di Cultura*).
GRASS unterstützt sehr viele Import- und Exportformate sowie einen direkten Zugriff auf die PostgreSQL/PostGIS-Datenbank. Die Daten können in der Datenbank gehalten werden und müssen nicht exportiert bzw. importiert werden.
Ein Download von GRASS ist von der offiziellen Homepage (grass.itc.it) möglich. Die Funktionalität von GRASS ist ebenfalls auf der offiziellen Homepage oder in NETELER & MITASOVA (2002, 2004) nachzulesen.

5.4.5 Statistiksoftware R

R wird in dieser Arbeit als „Statistiksoftware“ bezeichnet. R ist aber viel mehr als nur eine Statistiksoftware. Zum einen ist R eine Programmiersprache und zum anderen eine Arbeitsumgebung mit statistischem Schwerpunkt (DOLIĆ 2004, 2). Für Kenner und Könner soll erwähnt werden, dass R auf der Sprache S, die von den Bell Laboratories entwickelt wurde, basiert.
Die freie Software R wird als *open source* Projekt (www.r-project.org) von einer großen Entwicklergemeinde ständig verbessert. R lässt sich auf den häufigsten Betriebssystemen (Linux, Windows, ...) installieren. R unterliegt, so wie GRASS und PostGIS, der GNU General Public License.
Die Programmiersprache R ermöglicht dem Anwender eigene Funktionen für spezielle Fragestellungen zu schreiben, für die es noch keine oder keine befriedigende Lösung gibt. VENABLES & SMITH (2004, 2) charakterisieren die Arbeitsumgebung (*environment*) von R wie folgt:

> „*R is an integrated suite of software facilities for data manipulation, calculation and graphical display. Among other things it has*
> - *an effective data handling and storage facility,*
> - *a suite of operators for calculations on arrays, in particular matrices,*
> - *a large, coherent, integrated collection of intermediate tools for data analysis,*
> - *graphical facilities for data analysis and display either directly at the computer or on hardcopy, and*
> - *as well developed, simple effective programming language (called 'S') which includes conditionals, loops, user defined recursive functions and input and output facilities. (Indeed most of the system supplied functions are themselves written in the S language.)*“

Die Grundinstallation von R beinhaltet die meisten Funktionen, die für einfache statistische Auswertungen benötigt werden. Für R liegen sehr viele zusätzliche Pakete (*packages*) vor (INTERNET: CRAN), mit denen verschiedene Aufgaben bewältigt werden können (z.B. Cluster-Analyse). So gibt es z.B. ein Paket „Rdbi“, welches den Zugriff von der R Arbeitsumgebung auf die Daten einer relationalen Datenbank erlauben. Ein weiteres nützliches Paket ist „GRASS“. Damit können GRASS GIS Daten in R eingelesen und zur Analyse herangezogen werden.
Dieser flexible Einsatz von R in Kombination mit GIS Software und relationalen Datenbanken bringt sehr viele Vorteile und Einsatzmöglichkeiten mit sich.
Dem Informationssystem steht die komplette Arbeitsumgebung von R zur Verfügung. Statistische Analysen können auf zwei Arten gemacht werden. In R wird auf die räumliche Datenbank zugegriffen. Es findet ein Datentransfer von der Datenbank in die R Umgebung statt. Bei großen Datenmengen ist dies mit viel Rechenzeit (Datentransfer) und einer Verdoppelung der Datenmenge verbunden. Die Daten liegen in der Datenbank und in der R Umgebung. Die zweite Möglichkeit bietet die von JOSEPH E. CONWAY entwickelte Sprache

PL/R (siehe *„5.4.6.3 PL/R“*). Die Daten bleiben in der Datenbank und es findet ein „Funktionalitätstransfer“ von R zur Datenbank statt. Somit steht der Datenbank die komplette Funktionalität (inklusive Zusatzpaketen) von R zur Verfügung. Ein zeitraubender Datentransfer sowie eine Zunahme der Datenmenge werden verhindert.
Die Software R ist sehr gut dokumentiert. Auf der R Homepage (www.r-project.org) stehen nützliche Anleitungen und Dokumente, die den Einstieg erleichtern, zum Download bereit. DOLIĆ (2004) beschreibt in seinem sehr aktuellen Buch die Anwendung von R für einfache statistische Analysen (deskriptive Statistik, Grafiken mit R, Wahrscheinlichkeitstheorie, Wahrscheinlichkeits- und Verteilungsfunktionen, Schätztheorie, statistische Testverfahren, Zusammenhangsmaße, Regressionsanalyse) in der Wirtschafts- und Sozialwissenschaft.

5.4.6 Verwendete Sprachen

5.4.6.1 SQL (Structured Query Language)

SQL ist die Sprache, mit der der Client zum Datenbankserver (Abb. 5.5) spricht. Die SQL-Kommandos (auch SQL-Statements genannt) werden in drei Kategorien eingeteilt (vgl. JURGEIT 2003, 65f.; BOENIGK 2003, 6):

- **Datenbankkontrolle**: z.B. Datenbank erzeugen oder löschen, Rechteverwaltung
- **Datendefinition**: z.B. Tabelle erzeugen, verändern oder löschen, Index erzeugen, Beziehungen zwischen den Tabellen
- **Datenmanipulation**: z.B. Daten einfügen, verändern oder löschen

SQL wurde zweimal international standardisiert (nach ANSI/ISO-Standard). Die meisten Datenbanksysteme unterstützen den Standard SQL92. Der neuere Standard SQL99 wird noch nicht von allen Systemen unterstützt (JURGEIT 2003, 65). PostgreSQL „versteht“ beide Standards.
Von Datenbank- zu Datenbanksystem können die SQL-Befehle, trotz des Standards, im Detail unterschiedlich sein. Es empfiehlt sich, das Handbuch des jeweiligen Datenbanksystems durchzulesen. Die *OpenGIS Simple Feature Specification for SQL* ist eine Erweiterung der SQL-Funktionalität. Damit wird das Arbeiten mit räumlichen Daten über die Sprache SQL möglich.

5.4.6.2 PL/pgSQL

PostgreSQL bietet die Möglichkeit, SQL mit eigenen Funktionen zu erweitern. Folgende Arten von Funktionen stehen zur Verfügung (EISENTRAUT 2004, 448):

- in SQL geschriebene Funktionen
- in einer prozeduralen Sprache geschriebene Funktionen (z.B. PL/pgSQL, PL/R)
- interne Funktionen
- in C geschriebene Funktionen

Die Standarddistribution von PostgreSQL unterstützt die Programmiersprache PL/pgSQL, d.h. PostgreSQL kann Funktionen, die in PL/pgSQL geschrieben sind, übersetzen und ausführen. Bei der Installation von PostGIS muss PL/pgSQL sowieso installiert werden und steht dem Benutzer damit in der entsprechenden Datenbank zur Verfügung. Komplexe Berechnungen sind mit reinem SQL-Code nicht mehr möglich.

„Mit PL/pgSQL können Sie Berechnungen und Anfragen innerhalb des Datenbankservers zusammenfassen. Dadurch erhalten Sie die Fähigkeiten einer prozeduralen Sprache und die bekannte Funktionalität von SQL, sparen aber viel Zeit, weil die Verzögerung durch die Client/Server-Kommunikation wegfällt. Das kann die Leistung einer Datenbankverbindung erheblich verbessern. Mit PL/pgSQL können Sie außerdem alle Datentypen, Operatoren und Funktionen von SQL verwenden."

(EISENTRAUT 2004, 558)

Eine PL/pgSQL Funktion wird in der Diplomarbeit zum Beispiel für die Interpolation der Flugzeugposition für jeden Laserpunkt verwendet (siehe Kapitel *„8 Datenverarbeitung"*). EISENTRAUT (2004) listet die wichtigsten PL/pgSQL Befehle auf und gibt Beispiele, die den Einstieg in die Programmierung mit PL/pgSQL erleichtern.

5.4.6.3 PL/R

PL/R ist eine prozedurale Sprache und ermöglicht es, PostgreSQL Funktionen in der Programmiersprache R zu schreiben und dabei die Funktionalität von R zu nützen. Sie wird am Datenbankserver installiert. Funktionen können nur vom Datenbankadministrator geschrieben werden (*untrusted procedural language*), jedoch von allen Usern ausgeführt werden. Die Funktionen in PL/R werden wie normale SQL-Funktionen aufgerufen.
Die Daten können direkt in der Datenbank mit R analysiert werden. Eine Verminderung (Filterung) der Daten für statistische Analysen kann mit Hilfe von SQL-Statements erfolgen. Die Daten bleiben dort, wo sie gespeichert sind und müssen nicht zeit- und speicherplatzintensiv transferiert werden. PL/R Funktionen sind in der Regel langsamer als eingebaute vergleichbare SQL-Funktionen (z.B. Mittelwert, Summe, Standardabweichung, usw.). Sind jedoch entsprechende SQL-Funktionen nicht verfügbar oder nur sehr schwer nachzubauen, steht mit der Sprache PL/R fast der gesamte Funktionsumfang der Software R zur Verfügung. PL/R wird im Informationssystem zum Beispiel dazu verwendet Korrelationskoeffizienten zwischen zwei Spalten einer Datenbanktabelle zu berechnen. Die Homepage von JOSEPH E. CONWAY (www.joeconway.com/plr/) ist die derzeit beste Dokumentation zu PL/R.

5.4.6.4 Python

Python ist eine interpretierte, objektorientierte Programmiersprache und wird oft mit Java, Perl oder Tcl verglichen. Die Entwicklung von Python startete 1989 am Centrum voor Wiskunde en Informatica (CWI) in Amsterdam unter der Leitung von GUIDO VAN ROSSUM. Seit 2001 wird Python durch die Python Software Foundation (PSF) vertreten. Python ist eine nichtkommerzielle Software und unterliegt einer eigens definierten Lizenz (*PSF license*), die der *open source* Definition entspricht. Python ist eine *open source* Software und ist GPL-kompatibel.

„*GPL-compatible doesn't mean that we're distributing Python under the GPL. All Python licenses, unlike the GPL, let you distribute a modified version without making your changes open source. The GPL-compatible licenses make it possible to combine Python with other software that is released under the GPL; the others don't.*“

Auszug aus dem Python License Text (www.python.org, 2004)

Lizenzrechtlich ist es somit möglich, eine veränderte Version von Python oder eine mit Python entwickelte Software ohne Herausgabe des Quelltextes zu verbreiten.
Python ist plattformunabhängig. Das heißt, dass der Quelltext eines Pythonprogrammes auf jedem Betriebssystem, das Python unterstützt, funktionsfähig ist. Python läuft, wie die meisten *open source* Programme, auf Unix (Linux), Windows, OS/2, Mac, Amiga, uvm..
Folgende Gründe sprechen besonders für die Verwendung von Python als Programmiersprache für das Informationssystem:

- Python sowie das Modul Tkinter (siehe unten) sind plattformunabhängig.
- Die Einsatzmöglichkeiten von Python sind nahezu grenzenlos: z.B. dynamische Internetanwendungen, Softwareprogrammierung (OpenOffice wird mit Python entwickelt). Mit Python können große Projekte angegangen werden, aber auch nur sehr kleine Problemstellungen schnell und einfach gelöst werden.
- Python ist modular aufgebaut. Es gibt viele Zusatzmodule, wie z.B. das Modul „pg“. „pg“ ermöglicht den Zugriff auf eine PostgreSQL-Datenbank. Das Modul „RSPython“ erlaubt die Ausführung von R Befehlen in einer Pythonumgebung.
- Pythonprogramme sind sehr schnell und besitzen eine gute Speicherverwaltung.

WEIGEND (2003, 2004) sowie das Online-Tutorial auf www.python.org bieten einen leichten Einstieg in die Programmierung mit Python. Die integrierte Entwicklungsumgebung IDLE (www.python.org/idle) ist beim Download von Python automatisch dabei. IDLE ist die ideale Entwicklungsumgebung, speziell für Anfänger, die sich mit der Pythonsyntax vertraut machen wollen.

5.4.6.5 Tkinter

Tkinter („Tk interface“) ist ein Modul für die Programmiersprache Python. Tkinter ist das Standard Python-Interface zum Tk GUI Toolkit (INTERNET: Scriptics). Tk und Tkinter sind für die meisten Plattformen (Unix, Windows, Macintosh) verfügbar. Tkinter ermöglicht die Erstellung von Graphical User Interfaces (GUIs) mit Python. Das heißt, mit Python und Tkinter können plattformunabhängige Benutzeroberflächen geschrieben werden. Der ein und der selbe Quelltext einer Benutzeroberfläche kann auf Windows, Linux und MacOS ausgeführt werden und liefert eine an das Betriebssystem angepasste Benutzeroberfläche. Das Aussehen der Fenster, Buttons usw. wird dem Stil des jeweiligen Betriebssystems angeglichen (LUNDH 1999,1). Bei der Entwicklung von Applikationen mit Python ist es ein wesentlicher Vorteil, dass die Benutzeroberfläche ebenfalls mit Python geschrieben und direkt in den Quelltext der Applikation eingebaut werden kann. Der Prototyp der Benutzeroberfläche des Informationssystems wird in Kapitel *„9 Datenexport und Visualisierung“* näher beschrieben. LUNDH (1999) und WEIGEND (2003, 2004) geben eine gute Anleitung für die ersten Schritte bei der Entwicklung einer Benutzeroberfläche mit Tkinter.

5.5 Zusammenfassung

Das Kapitel *„5 Entwicklung des Informationssystems“* ist eines der größten Kapitel der Diplomarbeit. Dies hängt damit zusammen, dass die Kenntnis über den Aufbau des Informationssystems (Systemkomponenten) und die Kenntnis über die verwendete Software in den weiteren Kapiteln vorausgesetzt wird. Ein Kenner der Software wird sich in diesem Kapitel mehr auf die Methodik und den Aufbau des Systems konzentrieren und kann den Teil, in dem die Entwicklungsumgebung erklärt wird, überspringen.
Der Aufbau des Informationssystems lässt sehr viele Möglichkeiten für weitere Entwicklungen offen. Es können jederzeit neue Komponenten hinzugefügt werden (z.B. Applikation für die Erstellung von Gelände- und Oberflächenmodellen). Die ausnahmslose Verwendung von *open source* Software spart sehr viel Geld und ermöglicht es jedem, die Software zu installieren und ohne Verpflichtungen zu testen. Ferner ist der offene Zugang zum Quellcode dieser Software bei der Entwicklung von eigenen Applikationen von großem Vorteil. Ein evidenter Nachteil von *open source* Software ist mit Sicherheit die lange Einlernphase für diejenigen, die noch nie auf einem Linux-Betriebssystem gearbeitet haben. Ergänzend zu erwähnen bleibt, dass *open source* Software sehr wohl für Windows-Betriebssysteme zur Verfügung steht, aber ein integratives Zusammenspiel der Softwarepakete nur unter Unix-Betriebssystemen einwandfrei möglich ist.

Abschließend sind in Tabelle 5.8 die verwendeten Softwarepakete aufgelistet.

Software	Version	Homepage	Bemerkungen
Fedora	Core 2	fedora.redhat.com	Linux-Betriebssystem
GEOS	2.0.0	geos.refractions.net	zusätzliche räumliche Funktionen für PostGIS
GRASS	5.7 (CVS)	grass.itc.it	Geographisches Informationssystem (GIS)
pg	3.5	www.druid.net/pygresql/	Python-Modul: ermöglicht die Verbindung zum Datenbankser in einem Python-Skript
PL/R	0.6.1-alpha	www.joeconway.com/plr/	prozedurale Sprache: Funktionalität von R in der Datenbank
PostGIS	0.9	postgis.refractions.net	Funktionen und Datentypen einer räumlichen Datenbank
PostgreSQL	7.4.5	www.postgresql.org	Datenbankserver
Proj4	4.4.8	proj.maptools.org	Projektionen und Transformationen in PostGIS
Python	2.3.4	www.python.org	Import der Rohdaten in die Datenbank, Aufbau der Relationen
QGIS*	0.6.0 rc2	qgis.org	Visualisierung der PostGIS-Geometrien
R	2.0.0	www.r-project.org	Statistik- und Visualisierungssoftware
Tkinter	8.4	www.python.org/moin/TkInter	Erstellung eines Graphical User Interface (GUI)
XGobi*	latest	research.att.com/areas/stat/xgobi	3D-Visualisierung der Punktwolke

Tab. 5.8: Versionen der verwendeten Softwarepakete.

* werden in Kapitel *„9 Datenexport und Visualisierung“* näher beschrieben

6 Datenverwaltung und -speicherung

6.1 Einleitung

Das Kapitel *„6 Datenverwaltung und -speicherung“* widmet sich dem Kernbereich des Informationssystems. Eine Anforderung an das System ist es, die Rohdaten der Laserscannerbefliegungen so abzulegen, dass ein schneller Zugriff und die Selektion der Daten leicht möglich ist. Der Zugriff auf die Daten steuert nicht nur den Export, sondern ist auch bei Berechnungen entscheidend für die Performance. Der Datenbankserver PostgreSQL mit dem Zusatz PostGIS, der zusätzliche geometrische Datentypen und räumliche Funktionen mit sich bringt, werden für das Datenmanagement herangezogen. Bei der Speicherung von großen Datenmengen in einer PostgreSQL-Datenbank gibt es wenige Einschränkungen (Tab. 5.5).
Im Folgenden wird die Struktur der Datenbank (die Tabellen und ihre Beziehungen), und im speziellen die Speicherung von räumlichen Daten und Sachdaten, behandelt. Indices sind ein wichtiger Bestandteil von Datenbanken. Die Wahl der Indexierung – welche Daten mit welchem Typ von Index behaftet werden – kann viele Vorteile, aber auch Nachteile zur Folge haben. BOENIGK (2003) und KLEINSCHMIDT & RANK (2005) bieten einen guten Einstieg in die theoretische Datenmodellierung für relationale Datenbanksysteme und zeigen die praktische Umsetzung mit PostgreSQL auf.

6.2 Struktur der Datenbank

Aus dem Aufbau der Laserscannerrohdaten und dem Anforderungskatalog (Eigenschaften und Funktionalität) kann ein Entwurf der Datenbankstruktur abgeleitet werden. Es wird der Begriff „Entwurf“ verwendet, da die Struktur einer Datenbank nicht starr sein sollte, sondern durch die gesteigerten Bedürfnisse der Nutzer bzw. durch die fortschreitende Entwicklung der Hard- und Software einem ständigen Wandel unterliegt. So bietet z.B. die nächste Version von PostGIS (Version 1.0) neue Funktionen, die einen anderen Umgang mit den Daten in der Datenbank erlauben. Ein weiteres Beispiel ist die Anwendung eines neuen Laserscanners, der nicht nur zwei Reflexionen registriert, sondern die gesamte Impulsform des reflektierten Signals aufzeichnen kann. Der Aufbau der Datenbank muss mit diesen technischen Entwicklungen Schritt halten und ständig neu angepasst werden. Der Entwurf der Datenbankstruktur ist immer nur ein Modell der realen Welt zu einem bestimmten Zeitpunkt. Die Objekttypen (Entitäten), die Attribute, mit denen die Entitäten beschrieben werden und die Beziehungen, über die Entitäten miteinander verknüpft werden können, beschreiben die Realität in einer vereinfachten Form (vgl. BOENIGK, 6ff.). Beim Entwurf einer Datenbank gilt es diese Entitäten zu identifizieren, ihnen Attribute zuzuordnen und Beziehungen zwischen ihnen aufzubauen.

Anhand des konkreten Datenbankschemas wird auf diese drei Punkte näher eingegangen (Abb. 6.1). Die Datenbankstruktur wird so einfach wie möglich gehalten (nur 7 Tabellen!). Beim Entwerfen der Datenbankstruktur sollte man im Hinterkopf behalten, welche Operationen in der geplanten Anwendung auf den Tabellen arbeiten und welcher Datenumfang (Anzahl der Datensätze) in den jeweiligen Tabellen abgespeichert wird. Diese zwei Faktoren beeinflussen, ob es Sinn macht, eine vollkommene Normalisierung der Tabellen anzustreben (vgl. BOENIGK 2003, 30). Vor allem die große Datenmenge (mehrere hundert Millionen Datensätze) erfordert es, ein relativ starr wirkendes Datenbankschema anzuwenden. Die Verwendung von zusätzlichen Tabellen – sprich die Auslagerung zusammenhängender Attribute in eigenen Tabellen oder der Aufbau von n:m-Beziehungen (many to many) – würden den Speicherbedarf und die Rechenzeiten drastisch erhöhen. Eine Relationstabelle mit n x m Datensätzen sowie ein großer Aufwand bei der Verknüpfung der Tabellen wären die Folge.
Im Punkt 6.2.1 werden die modellierten Entitäten aufgezeigt, ihre Attribute beschrieben und die Beziehungen zwischen den Entitäten festgelegt.

6.2.1 Entitäten

Der aktuelle Stand der Technik des flugzeuggestützten Laserscannings erlaubt die Erstellung eines sehr allgemeinen Datenbankschemas, das für verschiedenste Aufnahmeverfahren Gültigkeit behält. Die Grundeinheit ist immer die **Befliegung**. Eine Befliegung weist immer ein oder mehrere **Flugstreifen** auf. Um ein Gebiet flächendeckend zu erfassen, werden sich überlappende Streifen geflogen, die schlussendlich das gesamte Gebiet abdecken sollten. Ein Flugstreifen besitzt viele **Laserpunkte**, die wiederum in *first* und l*ast pulse* Punkte unterteilt werden können. Die **Flugzeugpositionen** werden ursprünglich der Befliegung zugeordnet (siehe Kapitel *„4 Datenerfassung und Datengrundlagen“*). Mit Hilfe der Start- und Endzeit eines jeden Streifens können die Flugzeugpositionen mit den Flugstreifen verknüpft werden.

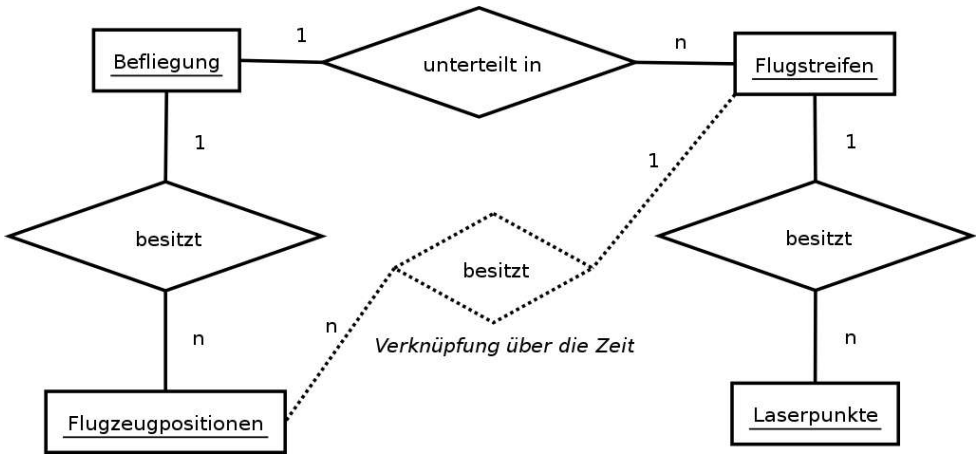


Abb. 6.1: Entitäten der Laserscannerbefliegung und ihre Beziehungen (eigener Entwurf).

Abbildung 6.1 zeigt die Tabellen, die Beziehungen zwischen den Tabellen und die Kardinalität der Beziehungen (z.B. 1:n-Beziehung). Ferner kann man in Abbildung 6.2 die Spalten der jeweiligen Tabelle entnehmen. Die Primärschlüssel (*primary ke*y – PK) – eine Spalte, die jede Instanz einer Entität eindeutig identifizierbar macht – sowie die Fremdschlüssel (*foreign key* – FK) sind eingezeichnet. Die Entitäten besitzen Attribute, die einerseits vorgegeben sind. Das heißt, diese Informationen werden aus den Rohdaten extrahiert und in die Datenbank importiert. Und andererseits besitzen sie Attribute, die sich aus anderen Attributen (z.B. Geometrie) der gleichen Entität oder einer anderen Entität ableiten lassen. Beispielsweise wird die durchschnittliche Fluggeschwindigkeit eines Flugstreifens aus der Geometrie der Flugzeugpositionen abgeleitet. Die Länge eines Flugstreifens wird aus seiner Geometrie berechnet. In Tabelle 6.1 sind die modellierten Entitäten mit ihren Tabellenbezeichnungen, Primärschlüsseln und Attributen aufgelistet.

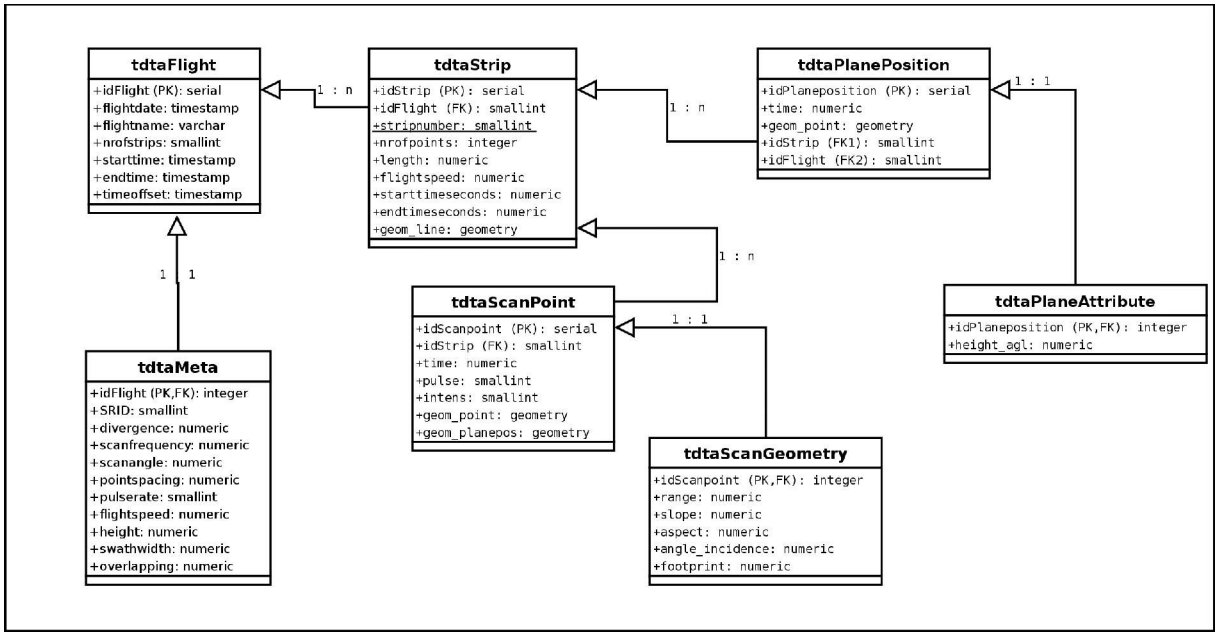


Abb. 6.2: Datenbankstruktur des Informationssystems. PK...*primary key*, FK...*foreign key* (eigener Entwurf).

Entität	Tabellenname	Primärschlüssel	Attribute
Befliegung	tdtaFlight	idFlight	Datum der Befliegung
			Name der Befliegung
			Anzahl der Streifen
			zeitlicher Beginn
			zeitliches Ende
			Datum des Wochenbeginns
Flugstreifen	tdtaStrip	idStrip	idFlight (Fremdschlüssel)
			Streifennummer
			Anzahl der Laserpunkte
			Länge des Streifens (in m)
			Fluggeschwindigkeit (m/s)
			zeitlicher Beginn
			zeitliches Ende
			Geometrie: Linie
Flugzeugposition	tdtaPlanePosition	idPlaneposition	Zeit (Sek. seit Wochenbeginn)
			Geometrie: Punkt
			idStrip (Fremdschlüssel)
			idFlight (Fremdschlüssel)
Laserpunkt	tdtaScanPoint	idScanpoint	idStrip (Fremdschlüssel)
			Zeit (Sek. seit Wochenbeginn)
			Reflexion (*first* oder *last pulse*)
			Intensität (5 bis 255)
			Geometrie: Punkt
			Geometrie: Punkt (Flugzeugposition)

Tab. 6.1: Übersicht der Objekttypen (Entitäten) der Laserscannerbefliegung. Kursiv geschriebene Attribute sind abgeleitete Größen (eigener Entwurf).

Die vier genannten Tabellen werden mit drei weiteren Tabellen ergänzt (Tab. 6.2). Die Metadaten einer Befliegung werden in einer eigenen Tabelle abgespeichert. In diesem Zusammenhang wird unter Metadaten alles verstanden, was von der Befliegungsfirma zusätzlich zu den Rohdaten ausgeliefert wird und dabei die Befliegung näher charakterisiert. Es handelt sich meist um die verwendeten Systemparameter oder geplante Ergebnisse der Befliegung (Tab. 6.2). Die Tabellen mit den Laserpunkten und den Flugzeugpositionen sind für die Speicherung großer Datenmengen konzipiert. Die Anzahl und Reihenfolge der Spalten ist fix festgelegt. Die Struktur dieser Tabellen darf nicht verändert werden, da die Daten beim Import als ganzes direkt in die Tabelle geschrieben werden (vgl. Kapitel *„7 Datenimport“*). Im Gegensatz dazu werden die Tabellen der Befliegung und der Flugstreifen direkt über die einzelnen Spalten angesprochen und „gefüttert“. Die Reihenfolge und Anzahl der Spalten spielt in diesem Falle keine Rolle. Der Dateninput in diese Tabellen ist nur sehr gering und kann daher mit dem einfacheren, aber langsameren Verfahren durchgeführt

werden. Im Kapitel *„7 Datenimport“* ist der Performanceunterschied zwischen den zwei Importvarianten erklärt. An die zwei großen Tabellen (Flugzeugpositionen, Laserpunkte) wird jeweils eine zusätzliche Tabelle mit einer 1:1-Beziehung angehängt. Dies erfolgt über die eindeutige Identifikation (ID) der Datensätze (Instanzen). Die Tabelle Laserpunkte wird mit einer Tabelle, die die Scangeometrie speichert, ergänzt. Jede Instanz der Tabelle Laserpunkte kann, aber muss nicht, maximal einen Wert in der Tabelle Scangeometrie haben. Dasselbe gilt für die Tabelle, die an die Tabelle Flugzeugpositionen angehängt wird. Sie enthält bis dato die Höhe über Grund (*height above ground level*) für ausgewählte Flugzeugpositionen.

Entität	Tabellenname	Primär-/Fremdschlüssel	Attribute
Befliegung	tdtaMeta	idFlight	Projektion
			Strahldivergenz
			Scanfrequenz
			Scanwinkel
			mittlerer Punktabstand
			Impulsrate
			Fluggeschwindigkeit
			Flughöhe
			Streifenbreite
			Streifenüberlappung
Flugzeugposition	tdtaPlaneAttribute	idPlaneposition	Flughöhe über Grund
Laserpunkt	tdtaScanGeometry	idScanpoint	Länge des Laservektors
			Neigung am Punkt
			Exposition
			Einfallswinkel des Vektors
			Fläche des Footprints

Tab. 6.2: Ausgelagerte Attribute der Entitäten Befliegung, Flugzeugposition und Laserpunkt.

6.2.2 Erstellen der Tabellen

Bei der Erstellung der Tabellen ist es wichtig von oben nach unten der Hierarchie der Tabellen zu folgen, damit die Verknüpfungen zwischen den Tabellen richtig hergestellt werden können. Die Auswahl der passenden Datentypen für die einzelnen Attribute der Objekttypen sollte gut überlegt werden, da damit viel Speicherplatz sowie Rechenzeit gespart werden kann. Die verfügbaren Datentypen der PostgreSQL-Datenbank sind in EISENTRAUT (2003, 105ff.) und BOENIGK (2003, 67ff.) im Detail erklärt. Alle verwendeten Datentypen sind in Tabelle 6.3 angegeben. Die maximale Genauigkeit und die Größenordnung der Daten, die im Informationssystem gespeichert werden, ist bekannt. Somit kann die passende Wahl des Datentyps mit entsprechender Angabe der Vor- und Nachkommastellen erfolgen.

Datentyp	Alias	Beschreibung
character varying(n)	varchar(n)	Zeichenkette mit variabler Länge: n...Höchstgrenze
geometry		wählbare PostGIS-Geometrietypen: *point, line, polygon, multipoint, multiline, multipolygon, geometrycollection*
integer	int, int4	4-Byte-Ganzzahl mit Vorzeichen: -2147483648 bis +2147483647
numeric [(p, s)]	decimal [(p, s)]	exakte Zahl mit wählbarer Präzision; p...Gesamtanzahl der Stellen, s...Stellen nach dem Komma
smallint	int2	2-Byte-Ganzzahl mit Vorzeichen: -32768 bis +32767
serial	serial4	selbstzählende 4-Byte-Ganzzahl: 1 bis 2147483647
timestamp [(p)]	timestamp	Datum und Zeit; p...Genauigkeit (maximal 1 Mikrosekunde)

Tab. 6.3: Auflistung der verwendeten Datentypen (vgl. EISENTRAUT 2003, 105ff.; RAMSEY 2004, 6ff.).

Die Erstellung einer Tabelle wird in den meisten Datenbanksystemen mit dem SQL-Statement CREATE vollzogen. Mit dem CREATE Statement wird der Name der Tabelle, die Spaltennamen und Datentypen sowie die *constraints* festgelegt. *Constraints* sind „Zwänge“ oder Bedingungen, die zusätzlich an die Spalten gebunden werden können. Im Falle der Tabelle *tdtaFlight* darf z.B. die Spalte Flugdatum nicht leer sein (NOT NULL) und der Flugname darf nur einmal in der ganzen Tabelle vorkommen (UNIQUE). Der Primärschlüssel (PRIMARY KEY) ist ebenfalls ein *constraint* und legt somit die Eindeutigkeit dieser Spalte fest, d.h. jeder Wert darf nur einmal vorkommen bzw. jede Instanz ist über den Primärschlüssel eindeutig identifizierbar.

```
CREATE TABLE tdtaFlight (                          --Tabelle der Befliegungen
      idFlight serial PRIMARY KEY,
      flightdate timestamp NOT NULL,
      flightname varchar(8) UNIQUE,
      nrofstrips smallint,
      starttime timestamp,
      endtime timestamp,
      timeoffset timestamp
)
```

```
CREATE TABLE tdtaMeta (                          --Metadaten der Befliegungen
      idFlight integer PRIMARY KEY,
      SRID integer DEFAULT -1,
      divergence NUMERIC(4,2) DEFAULT 0.25,
      scanfrequency NUMERIC(5,2),
      scanangle NUMERIC(5,2),
      pointspacing NUMERIC(4,2),
      pulserate INTEGER,
      flightspeed NUMERIC(5,2),
      height NUMERIC(6,2),
      swathwidth NUMERIC(6,2),
      overlapping NUMERIC(4,2),
```

```
      FOREIGN KEY (idFlight) REFERENCES tdtaFlight(idFlight) ON
      DELETE CASCADE ON UPDATE CASCADE
)
```

Die Tabelle *tdtaMeta* wird über einen Fremdschlüssel *constraint* (FOREIGN KEY) mit der Tabelle *tdtaFlight* verknüpft. Die Spalte *idFlight*, die in beiden Tabellen vorkommt, verbindet die zwei Tabellen. Wird eine Instanz in der Stammtabelle geändert oder gelöscht, wirkt sich das direkt auf die verknüpfte Instanz in der angehängten Tabelle aus (ON DELETE CASCADE ON UPDATE CASCADE).

```
CREATE TABLE tdtaStrip (                                    --Flugstreifen
      idStrip serial PRIMARY KEY,
      stripnumber smallint NOT NULL,
      nrofpoints integer,
      length NUMERIC(7,2),
      flightspeed NUMERIC(5,2),
      starttimesec NUMERIC(13,6),
      endtimesec NUMERIC(13,6),
      idFlight smallint NOT NULL,
      FOREIGN KEY (idFlight) REFERENCES tdtaFlight (idFlight) ON
      DELETE CASCADE ON UPDATE CASCADE,
      UNIQUE(stripnumber, idFlight)
)
```

Die Tabelle *tdtaStrip* wird, gleich wie die Tabelle *tdtaMeta*, mit der Tabelle *tdtaFlight* über die *idFlight* verknüpft. Für jede Befliegung darf eine Streifennummer nur einmal vergeben werden. Die Kombination aus Streifennummer und Befliegungsidentifikation (*idFlight*) wird durch einen *constraint* auf Eindeutigkeit überprüft (UNIQUE).

```
CREATE TABLE tdtaPlanePosition (                        --Flugzeugpositionen
      idPlanePosition serial PRIMARY KEY,
      time NUMERIC(12,5) NOT NULL,
      idFlight smallint NOT NULL,
      idStrip smallint DEFAULT NULL,
      FOREIGN KEY (idFlight) REFERENCES tdtaFlight (idFlight) ON DELETE
      CASCADE ON UPDATE CASCADE
)
```

```
CREATE TABLE tdtaPlaneAttribute (         --Attribute der Flugzeugpositionen
      idPlanePosition integer PRIMARY KEY,
      height_agl NUMERIC(6,2),
      FOREIGN KEY (idPlanePosition) REFERENCES tdtaPlanePosition
      (idPlanePosition) ON DELETE CASCADE ON UPDATE CASCADE
)
```

```
CREATE TABLE tdtaScanPoint (                                    --Laserpunkte
      idScanPoint serial PRIMARY KEY,
      idStrip smallint NOT NULL,
      time NUMERIC(13,6) NOT NULL,
      pulse smallint NOT NULL,
      intens smallint NOT NULL,
      FOREIGN KEY (idStrip) REFERENCES tdtaStrip (idStrip) ON DELETE
      CASCADE ON UPDATE CASCADE
)
```

```
CREATE TABLE tdtaScanGeometry (              --Scangeometrie der Laserpunkte
      idScanPoint integer PRIMARY KEY,
      range NUMERIC(6,2),
      slope NUMERIC(4,2),
      aspect NUMERIC(5,2),
      angle_incidence(5,2),
      footprint(5,2),
      FOREIGN KEY (idScanPoint) REFERENCES tdtaScanpoint (idScanpoint) ON
      DELETE CASCADE ON UPDATE CASCADE
)
```

6.2.3 Räumliche Daten - Geometriespalten

Bisher fehlen die Geometriespalten in den Tabellen. Die PostGIS-Geometriespalten müssen mit dem SQL-Befehl *AddGeometryColumn* einzeln in die entsprechenden Tabellen eingefügt werden. RAMSEY (2004, 6ff.) erläutert ausführlich, wie dieser Befehl im Detail auszuführen ist. In Abbildung 6.3 sind die von PostGIS erstellten Tabellen und ihre Spalten aufgelistet. Diese zwei Tabellen (*geometry_columns*, *spatial_ref_system*) sind interne Tabellen von PostGIS und dienen der Verwaltung aller Geometriespalten einer Datenbank. In der Tabelle *geometry_columns* sind die Definitionen aller Spalten, die einer PostGIS Geometrie entsprechen, enthalten. Die Tabelle *spatial_ref_system* ist für die Verwaltung aller Projektionen und Transformationsparameter verantwortlich. Wird eine Spalte mit dem Befehl *AddGeometryColumn* erzeugt, wird automatisch ein Eintrag in *geometry_columns* mit den entsprechenden Informationen über diese Geometriespalte eingefügt. In welcher Projektion diese Geometrie vorliegt, ist über das Attribut *SRID* (*spatial referencing system identifier*) festgelegt. Jede *SRID* entspricht einem Eintrag in der Tabelle *spatial_ref_system*. Zusätzlich können selbst definierte Projektions- und Transformationsparameter hinzugefügt werden. Die neu erzeugte Projektionsdefinition kann der Geometrie zugeteilt werden. Ferner ist die Dimension wichtig, in der die Daten vorliegen. Bei der Erstellung einer Geometriespalte muss festgelegt werden, ob es sich um 2-D- oder 3-D-Geometrie handelt.

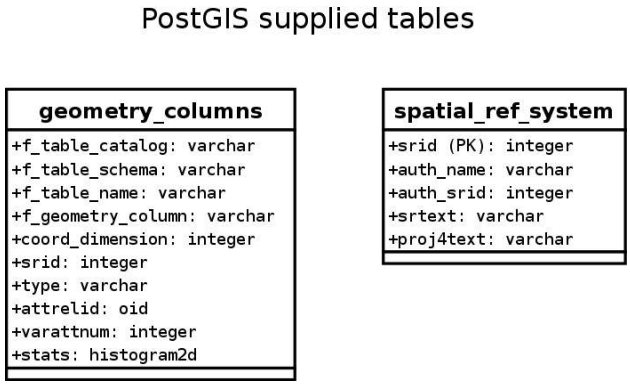


Abb. 6.3: Von PostGIS erstellte Tabellen, die die Geometriespalten einer Datenbank verwalten (eigener Entwurf).

```
--Hinzufügen der Geometriespalten

SELECT AddGeometryColumn('public','tdtastrip','geom_line',-1,
'LINESTRING',3);

SELECT AddGeometryColumn('public','tdtaplaneposition','geom_point',-1,
'POINT',3);

SELECT AddGeometryColumn('public','tdtascanpoint','geom_point',-1,
'POINT',3);

SELECT AddGeometryColumn('public','tdtascanpoint','geom_planepos',-1,
'POINT',3);
```

Als erstes muss angegeben werden, in welchem Datenbankschema sich die Tabelle befindet, dann folgt der Name der Tabelle, der Name der Spalte, die *SRID* (-1 steht für *undefined*), der Geometrietyp und letztlich die Koordinatendimension.
Der Befehl *AddGeometryColumn* erzeugt zusätzlich zwei *constraints* je Geometriespalte. Der erste *constraint* überprüft beim Einfügen neuer Objekte, ob die *SRID* der neuen Geometrie der *SRID* der Tabelle entspricht und der zweite *constraint* sorgt dafür, dass nur Objekte mit dem gleichen Geometrietyp (Punkt, Linie, Polygon,...), mit dem die Spalte definiert wurde, eingefügt werden können. Aus Performancegründen wird auf diese Überprüfung verzichtet, da jeder Check beim Einfügen von Daten Zeit kostet.

```
--Beispiel für das Entfernen eines constraints

ALTER TABLE tdtaPlaneposition DROP CONSTRAINT enforce_srid_geom_point;
```

6.3 Indexierung

„Die Optimierung der Anwendungen gehört zu den wichtigsten Aufgaben eines Entwicklers. Solange die Tabellen nicht viele Datensätze enthalten, mag das nicht so sehr ins Gewicht fallen. Bei großen Tabellen kann die Geschwindigkeitseinbuße allerdings erheblich sein, wenn die Datenbank nicht oder unzureichend optimiert wurde.“

(BOENIGK 2003, 247)

Das Zitat von BOENIGK (2003) zeigt recht deutlich, wie wichtig die Optimierung der Datenbank speziell bei großen Datenmengen ist. Die wohl beste Möglichkeit für die Verbesserung der Zugriffszeiten auf Daten in einer Tabelle stellt der so genannte **Index** dar (EISENTRAUT 2003, 187).
Grundsätzlich speichern Datenbanksysteme die Datensätze in derselben Reihenfolge, in der sie eingefügt werden. Eine Tabelle ist somit eine unsortierte Menge von Tupeln (vgl. BOENIGK 2003, 247ff.). Wird ein spezieller Datensatz gesucht, z.B. ein Laserpunkt mit der Intensität 120, muss die gesamte Tabelle – da unsortiert – durchsucht werden. Bei mehreren hundert Millionen Laserpunkten dauert die Suche einige Minuten. Wird die Suche für viele Punkte wiederholt, wird die Rechenzeit entsprechend länger. Auch wenn man die Tabelle nach einer bestimmten Spalte sortiert, um das Suchen zu vereinfachen, bleibt das Problem für die anderen Spalten ungelöst. Ferner müsste die Tabelle nach dem Einfügen eines Datensatzes neu sortiert werden. Indices verbessern die Leistung der Datenbank, bringen aber gleichzeitig einen hohen Verwaltungsaufwand für das Datenbanksystem mit sich. Einen Index kann man sich als Zusatztabelle vorstellen, die nach Werten der betreffenden Spalte sortiert ist. Sie enthält für jeden Wert eine Referenz auf den entsprechenden Datensatz in der Stammtabelle. Bei einer Suche mit Kriterien muss nicht die gesamte Stammtabelle „durchgeforstet“ werden, sondern die Suche im Index führt zu einem wesentlich schnelleren Ergebnis. Ein evidenter Nachteil von Indices ist die drastische Erhöhung der Datenmenge bzw. des Speicherplatzverbrauchs der Datenbank (siehe Kapitel *„10 Evaluierung und Ausblick“*). Wird für jede Spalte eine Indextabelle angelegt, verdoppelt sich die Datenmenge. Speziell bei großen Datenmengen kann man nicht ohne Index auskommen, da sonst die Suchzeiten zu lange wären. Man sollte aber stets darauf achten, dass nur sehr häufig frequentierte Spalten indexiert werden, damit der Speicherplatzverbrauch nicht zu groß wird. Werden Daten in die Tabelle eingefügt, muss der Index reorganisiert werden, was wiederum Zeit kostet (vgl. GESCHWINDE & SCHÖNIG 2002a, 158). Es ist auf jeden Fall schneller große Datenmengen ohne Index zu importieren, um erst anschließend den Index aufzubauen, als für jeden importierten Datensatz den Index zu erneuern (vgl. Kapitel *„7 Datenimport“*).

„Indexes are what make using a spatial database for large databases possible."

(RAMSEY 2004, 16)

Spalten, die als Primärschlüssel fungieren, erhalten automatisch einen Index. In PostgreSQL gibt es mehrere Indextypen, auf die an dieser Stelle nicht näher eingegangen wird (siehe EISENTRAUT 2003, 188f.). Es bleibt zu erwähnen, dass PostGIS eine Kombination von zwei Indextypen für die Indexierung der Geometriespalten verwendet: „*PostGIS uses an R-Tree index implemented on top of GiST (Generalized Search Trees) to index GIS data.*" (RAMSEY 2004, 16)

```
--Indexierung

CREATE INDEX plane_time on tdtaplaneposition (time);
CREATE INDEX scan_time on tdtascanpoint (time);
CREATE INDEX point_info on tdtapointinfo (idscanpoint);
CREATE INDEX strip_line on tdtastrip USING GIST (geom_line
gist_geometry_ops);

CREATE INDEX scan_point on tdtascanpoint USING GIST (geom_point
gist_geometry_ops);

--CREATE INDEX planepos_point on tdtascanpoint USING GIST (geom_planepos
gist_geometry_ops);

--CREATE INDEX plane_point on tdtaplaneposition USING GIST (geom_point
gist_geometry_ops);
```

Zusätzlich zu den Primärschlüsselspalten und den Geometriespalten werden die Spalten, die die zeitliche Verortung für jede Flugzeugposition und für jeden Laserpunkt enthalten, indexiert.

6.4 Zusammenfassung

Das Kapitel *„6 Datenverwaltung und -speicherung"* kommt mit gutem Grunde nach dem Kaptitel *„4 Datenerfassung und Datengrundlagen"* und vor dem Kapitel *„7 Datenimport"*. Die Kenntnis über die vorliegenden Daten sowie der Anforderungskatalog erlauben es dem Entwickler eine passende Datenbankstruktur aufzubauen. Der Entwickler muss dabei vor allem die großen Datenmengen im Hinterkopf behalten und darauf achten, dass die Geschwindigkeit der Berechnungen bzw. Abfragen nicht zu sehr leidet und gleichzeitig sollte der Speicherplatzverbrauch möglichst gering gehalten werden. Erst eine Indexierung von häufig abzufragenden Tabellenspalten ermöglicht es, mit diesen Datenmengen sinnvoll arbeiten zu können. Die Erweiterung PostGIS für PostgreSQL Datenbanken enthält einen Indexierungsalgorithmus, der speziell für Spalten mit PostGIS-Geometrie entwickelt wurde. Nachdem die Datenbankstruktur steht, kann sich der Entwickler dem Import der Rohdaten widmen.

7 Datenimport

7.1 Methodik

Ein direkter Import der Laserscannerdaten, wie sie derzeit von der Befliegungsfirma geliefert werden, ist nicht möglich. Das vorliegende Datenformat (siehe Kapitel *„4 Datenerfassung und Datengrundlagen“*) wird mit einem Python-Skript gelesen, ausgewertet und in die Datenbank geschrieben. Das Einlesen der Rohdaten lässt sich leicht verallgemeinern. Im Detail jedoch gibt es viele Möglichkeiten, wie sich dieser Prozess gestalten kann. Beispielsweise kann man mehrere Varianten des Programmstarts (z.B. Kommandozeile, Benutzeroberfläche) umsetzen. Das Schreiben der Daten in die Datenbank kann auf mehrere Arten geschehen (z.B. über ein SQL-Dumpfile oder direkt in den Server Socket). Aus diesen vielen Möglichkeiten der Umsetzung mit der Programmiersprache Python (www.python.org) soll die schnellste Variante gewählt werden. Wie schon oft vorher in dieser Diplomarbeit erwähnt, soll nochmals darauf aufmerksam gemacht werden, dass speziell bei großen Datenmengen kleine Unterschiede in der Geschwindigkeit einer Berechnung (eines Prozesses) große Zeitunterschiede in der Gesamtsumme ergeben. Ferner muss beachtet werden, dass nicht zu große temporäre Datenmengen entstehen. Zur Vereinfachung ist der Datenimport in Abbildung 7.1 in drei Stufen dargestellt.

Der **Programmstart** kann über die Kommandozeile am Linux-Server oder über eine graphische Benutzeroberfläche erfolgen. Beim Start des Python-Skripts in der Kommandozeile müssen die Verzeichnisse, in denen die zu importierenden Daten liegen, als Argumente angegeben werden (eine weitere Möglichkeit wäre eine interaktive Abfrage dieser Verzeichnisse). Die graphische Oberfläche erlaubt es dem Benutzer, die Verzeichnisse mit Hilfe eines betriebssystemtypischen Verzeichnisbaumes auszuwählen (vgl. Kapitel *„9 Datenexport und Visualisierung“*). Die ausgewählten Verzeichnisse werden überprüft, ob deren Namen und die Dateinamen der Files in diesen Verzeichnissen dem vorgegebenen Datenschema (siehe Kapitel *„4 Datenerfassung und Datengrundlagen“*) entsprechen:

- Die Namen der Verzeichnisse müssen einem Datumsformat entsprechen. Zum Bespiel: 041224_1 → erste Befliegung am 24.12.2004
- Die Dateinamenserweiterungen (*file extensions*) müssen mit den definierten Typen übereinstimmen. Zum Beispiel: *.all → für *last pulse* Daten
- Im Stammverzeichnis der Daten muss ein Verzeichnis vorhanden sein, das die Flugzeugpositionsdateien enthält. Zum Beispiel: tra-imu

Nicht passende Verzeichnisse bzw. Dateien werden beim Import ignoriert.

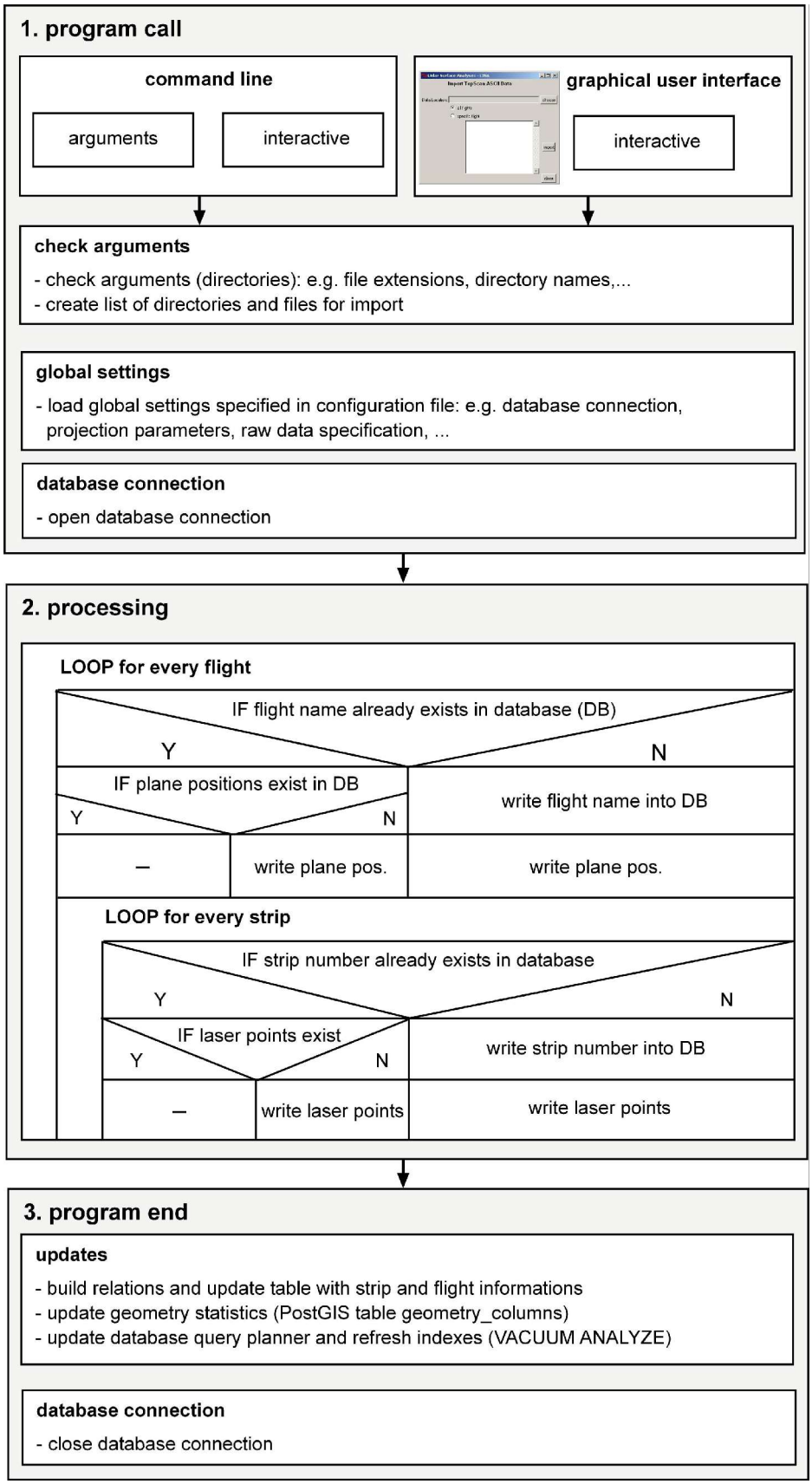


Abb. 7.1: *Data Import Workflow* (eigener Entwurf).

Die so genannten *global settings* sind in einer separaten Datei abgespeichert und können mit einem einfachen Texteditor jederzeit verändert werden. Diese Einstellungen (Variablen) werden vom Hauptprogramm eingelesen. Darin sind die erlaubten Dateinamenserweiterungen und der Name des Verzeichnisses mit den Flugzeugpositionen festgelegt. Ferner definiert werden die Zugriffsparameter für den Datenbankserver, sowie die Namen der Tabellen. Die Projektionsparameter der Rohdaten sind ebenfalls in dieser Konfigurationsdatei zu definieren. Mit dem Python-Modul *pg* (www.druid.net/pygresql/) wird eine Verbindung zum PostgreSQL-Server geöffnet (BOENIGK 2003, 299ff.). Dieses Zusatzmodul stellt viele Methoden für das Arbeiten mit einer PostgreSQL-Datenbank zur Verfügung (z.B. Anwendung von SQL-Statements, Einfügen von Daten, Löschen von Daten, Daten aktualisieren, usw.).
Die **Verarbeitung** der Rohdaten – das Einlesen und Schreiben der Daten in die Datenbank – ist bei großen Datenmengen ein sehr zeit- und rechenintensiver Prozess. Die Programmiersprache Python bietet einen schnellen Zugriff auf große Dateien (Lese- und Schreibzugriff). Die Abfolge bzw. Logik des Datenimports ist in Abbildung 7.1 in Form eines Flussdiagramms aufgezeigt.
Es gibt zwei Möglichkeiten Daten in eine PostgreSQL-Datenbank zu schreiben, d.h. Datensätze in eine Tabelle einfügen. Zum einen kann das SQL-Statement INSERT und zum anderen der Befehl COPY, der nicht im SQL-Standard enthalten ist, verwendet werden (EISENTRAUT 2003, 85f.; BOENIGK 2003, 164ff.). Beim INSERT-Statement sind die Spalten, in die geschrieben wird, explizit anzugeben. Es wird immer nur ein Datensatz eingefügt. Mit COPY kann man Daten aus einer Datei direkt in eine PostgreSQL-Tabelle schreiben. Der Datensatz in der Datei muss gleich viele Spalten wie die Zieltabelle aufweisen, sonst wird ein Fehler ausgegeben. Die Wahl des Trennzeichens zwischen den Spalten steht frei. COPY eignet sich somit besser für große Datenmengen. Der Geschwindigkeitsunterschied zwischen den beiden Methoden ist in „*7.4 Performancetests*“ aufgezeigt. Beide Methoden haben Vor- und Nachteile und eignen sich für unterschiedliche Aufgaben, auf die an dieser Stelle nicht näher eingegangen wird (vgl. EISENTRAUT 2003, BOENIGK 2003, GESCHWINDE & SCHÖNIG 2002a, www.postgresql.org).

Beispiel: INSERT INTO tdtastrip (stripnumber, idflight) VALUES (1100, 1);

In die Flugstreifen-Tabelle wird ein neuer Streifen eingefügt. Es werden Werte in die Spalten Streifennummer und die Spalte Befliegungs-ID eingefügt.
Alle Tabellen, außer den datenintensiven Tabellen mit den Flugzeugpositionen und den Laserpunkten, werden mit **INSERT** gefüllt. Für das Einfügen der großen Datenmengen wird der Befehl **COPY** verwendet. Der Befehl COPY kann Daten nicht nur aus Dateien, sondern auch vom Standardeingabegerät (*stdin*) lesen. Dies bedeutet, dass man Daten mit der Tastatur eingeben oder mit einem Befehl bzw. Skript in das Standardeingabegerät schreiben kann. Das Python-Modul *pg* verfügt über Methoden (*putline()*, *endcopy()*), die die Verwendung des COPY-Befehls in einem Python-Programm ermöglichen (GESCHWINDE & SCHÖNIG 2002a, 304).

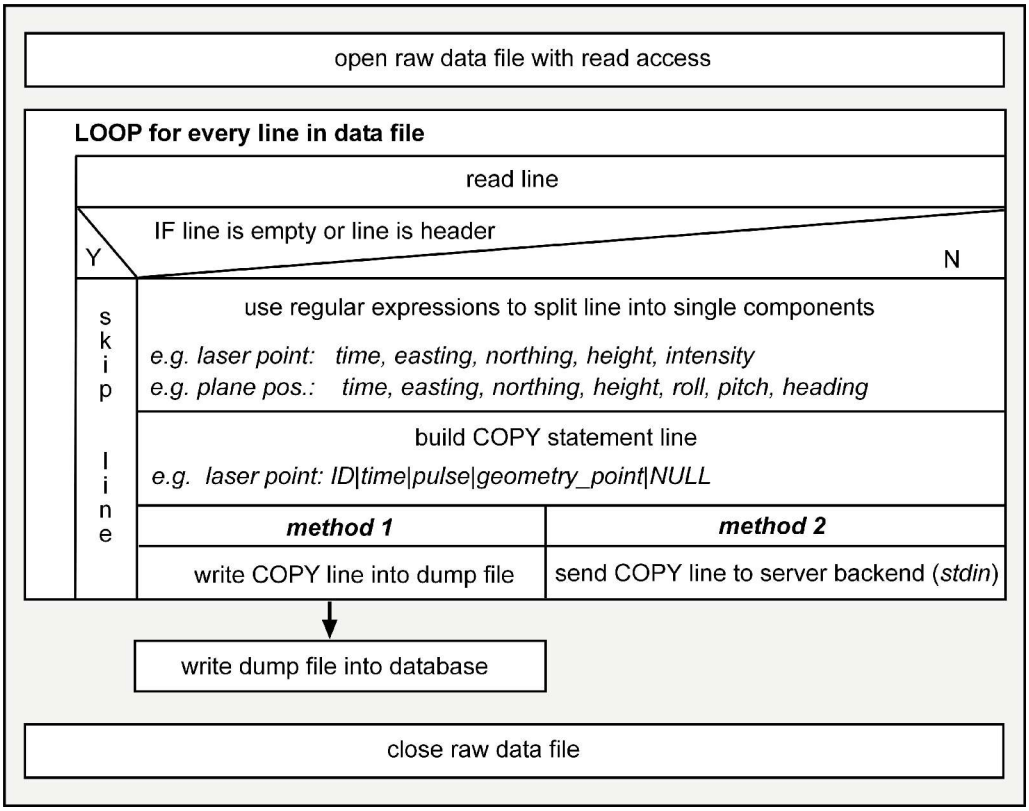


Abb. 7.2: *Import Process in Details* (eigener Entwurf).

In Abbildung 7.2 ist der Ablauf des Imports der Flugzeugpositionen und der Laserpunkte vereinfacht dargestellt. Die einzelnen Rohdatenfiles werden mit Lesezugriff geöffnet. Alle Zeilen in dieser ASCII-Datei werden systematisch gelesen. Es findet eine Überprüfung statt, ob die Zeile leer ist (also keine Daten enthält), ob es sich um eine Headerzeile handelt (z.B. Bezeichnung der Spalten) oder ob es sich um eine Zeile mit Daten handelt, die für den Import in die Datenbank in Frage kommen. Das festgelegte Rohdatenformat (siehe Kapitel *„4 Datenerfassung und Datengrundlagen“*) erlaubt es, die Zeile mit Hilfe von so genannten „regulären Ausdrücken“ in ihre Komponenten zu zerlegen (Python-Modul *re*: WEIGEND 2004, 322ff.). Bei den Flugzeugpositionen wird zum Beispiel die Zeit, die 3-dimensionalen Koordinaten (*easting*, *northing*, *height*) herausgefiltert. Die restlichen Informationen werden bis data vernachlässigt (z.B. *roll*, *pitch*, *heading*). Die herausgefilterten Daten werden zu einer Zeichenkette, die von der Datenbank gelesen werden kann, neu zusammengesetzt.

Beispiel: regulärer Ausdruck (Flugzeugpositionen)

```
"([  ]+)(\d+[.]\d+)([  ]+)(\d+[.]\d+)([  ]+)(\d+[.]\d+)([  ]+)(\d+
[.]\d+)([  ]+)([-]*\d+[.]\d+)([  ]+)([-]*\d+[.]\d+)([  ]+)([-]*\d+
[.]\d+)([ ]+)([-]*\d+[.]\d+)"
```

Beispiel: neu zusammengesetzte Zeichenkette (Laserpunkt)

```
1|1|207185.780820|2|5|SRID=-1;POINT(32551495.14 5261535.23 441.59)|\N
```

von links nach rechts:

ID des Punktes | Streifen-ID | Zeit | Art des Pulses (1..first, 2..last) | Intensität | Geometrie des Punktes | Geometrie der Flugzeugposition für den Punkt (hier \N → NODATA)

Die fortlaufende Nummerierung der Datensätze (IDs) wird vom Python-Skript übernommen, da beim Import mit COPY die Sequenzen der Primärschlüssel nicht fortgeführt werden.

Es sind beide Möglichkeiten des Imports mit COPY – über eine Datei oder über das Modul *pg* – in lauffähigen Programmen umgesetzt. Die erste Methode fügt die neu erzeugten Zeichenketten in eine Datei ein. Ist das Lesen der Rohdatendatei beendet, wird diese Datei (*dump file*) komplett in die Datenbank importiert. Diese Methode ist zwar absturzsicherer, bringt aber einige Nachteile mit sich. Das *dump file* hat mindestens die Größe der ursprünglichen Rohdatendatei. Der Schreibprozess in diese temporäre Datei bedeutet mehr Rechen- und somit auch Zeitaufwand. Die zweite Methode nützt die Funktionen des *pg*-Moduls. Die erzeugte Zeichenkette wird nicht wie bei der ersten Methode in eine Datei geschrieben, sondern direkt an das Server-Backend gesendet. Wenn das Ende der Rohdatendatei erreicht ist, importiert der Datenbankserver die erhaltenen Daten.

7.2 Aufbau der Relationen

Das Kapitel *„6 Datenverwaltung und -speicherung“* beschreibt die Entitäten der Datenbank und wie diese Entitäten in Beziehung zu einander stehen. Beim Import der Rohdaten in die Datenbank gilt es, diese Beziehungen aufzubauen. Ein Datensatz – eine Zeile in einer Rohdatendatei mit Laserpunkten – isoliert betrachtet, gibt keine Auskunft, zu welcher Befliegung oder zu welchem Flugstreifen diese Instanz gehört, oder um welche Reflexion es sich handelt. In der Datenbank werden alle Laserpunkte in einer Tabelle verwaltet. Diese Tatsache macht es notwendig, den einzelnen Instanzen zusätzliche Informationen anzuhängen, die es erlauben, Verknüpfungen (Relationen) zwischen den Entitäten herzustellen. Wie oben gezeigt, wird jedem Laserpunkt die passende Flugstreifen-ID zugeteilt. Diese Flugstreifen wiederum sind über ein Attribut (Fremdschlüssel) mit den Befliegungen verknüpft. Es reicht also, wenn jeder Laserpunkt ein Attribut Flugstreifen-ID besitzt. Die Entität Laserpunkt ist über die Entität Flugstreifen mit der Entität Befliegung verknüpft (Abb. 6.1). Das heißt, dass man für jeden Laserpunkt, die entsprechende Befliegung abfragen kann, ohne dass diese Information mit jedem Punkt abgespeichert wird. Die Rohdaten mit den Flugzeugpositionen sind differenzierter zu betrachten. Die Dateien mit den Flugzeugpositionen sind auf Ebene der Befliegungen abgelegt (siehe Kapitel *„4 Datenerfassung und Datengrundlagen“*). Pro Befliegung gibt es eine Datei mit den Positionen des Flugzeugs. Beim Auslesen der Rohdaten ist somit eine direkte Verknüpfung

zwischen den Flugstreifen und den Flugzeugpositionen nicht möglich. Die Relation zwischen Flugstreifen und Flugzeugposition kann erst nach dem Import aufgebaut werden. Beim Import eines Flugstreifens wird zusätzlich die minimale und die maximale Zeit (in Sekunden seit Wochenbeginn) aller Laserpunkte dieses Streifens abgespeichert. Die Flugzeugpositionen erhalten beim Import ein zusätzliches Attribut Befliegungs-ID. Nachdem eine Befliegung (Flugzeugpositionen und Laserpunkte) komplett in die Datenbank importiert wurde, kann den Flugzeugpositionen mit Hilfe einer PL/pgSQL Funktion eine eindeutige Flugstreifen-ID zugeteilt werden. Eine Flugzeugposition, die im Zeitintervall minimale bis maximale Zeit eines Flugstreifens liegt und die selbe Befliegungs-ID aufweist (d.h. am selben Tag aufgenommen wurde), kann eindeutig diesem Flugstreifen zugeordnet werden.

Beispiel: PL/pgSQL Funktion für die Verknüpfung von Flugzeugposition und Flugstreifen über das Attribut Zeit und das Attribut Befliegungs-ID:

```
FOR strip IN SELECT * FROM tdtastrip order by idstrip LOOP
      UPDATE tdtaplaneposition SET idstrip=strip.idstrip WHERE
      tdtaplaneposition.time BETWEEN strip.starttimesec-1 AND
      strip.endtimesec+1 AND strip.idflight=tdtaplaneposition.idflight;
END LOOP;
```

Für die Interpolation der Flugzeugposition für jeden Laserpunkt (siehe Kapitel *„8 Datenverarbeitung“*) ist es unbedingt notwendig, eine Flugzeugposition vor und nach der Aufnahme des Laserpunkts heranziehen zu können. Aus diesem Grund wird das Zeitintervall in beide Richtungen um 1 Sekunde verlängert. Dies ist möglich, da zwischen den einzelnen Flugstreifen in der Regel größere Zeitabstände liegen und es daher zu keinen zweideutigen Zuordnungen kommt.
Der Aufbau der Relationen – die Abspeicherung zusätzlicher Information für jeden Datensatz – führt logischerweise zu einem höheren Speicherplatzverbrauch, als es bei den Rohdaten der Fall ist.
Nach dem Import großer Datenmengen wird der PostgreSQL-Befehl VACUUM ANALYZE gestartet. EISENTRAUT (2003, 291) gibt folgende Gründe an, diesen Befehl regelmäßig auszuführen:

- um den von aktualisierten oder gelöschten Zeilen belegten Speicherplatz wiederzugewinnen
- um die vom PostgreSQL-Anfrageplaner verwendeten Datenstatistiken zu aktualisieren
- um sich gegen den Verlust sehr alter Daten durch Überlauf der Transaktionsnummern zu schützen.

Verlorene Datensätze, die z.B. durch einen Programmabsturz während des Imports entstanden sind und Speicherplatz verbrauchen, können durch VACUUM entfernt werden. Der PostGIS-Befehl *update_geometry_stats()* aktualisiert die Informationen über alle Geometriespalten einer Datenbank und speichert sie in der Tabelle *geometry_columns* (RAMSEY 2004, 17).

7.3 Berechnung erster Informationen

Nachdem die Daten in die entsprechenden Tabellen eingefügt wurden, steht die mächtige Funktionalität der PostgreSQL-Datenbank für die Analyse der Daten bereit. Es könnten schon im Zuge des Python-Programmes für den Datenimport die meisten Berechnungen (siehe Kapitel *„8 Datenverarbeitung“*) automatisch für die neu importierten Datensätze durchgeführt werden (z.B. Berechnung der Scangeometrie für jeden Laserpunkt). Um den Prozess des Datenimports vom Prozess der eigentlichen Datenverarbeitung zu trennen, werden beim Datenimport lediglich die Relationen aufgebaut (siehe oben) sowie die Attribute der Entitäten Flugstreifen und Befliegung berechnet (vgl. Tab. 6.1).

7.3.1 Attribute der Befliegung

Die Attribute der Befliegung, die nicht direkt importiert werden, sondern aus der Entität Flugstreifen abzuleiten sind, werden mit mehreren PL/pgSQL Funktionen berechnet. Die Relation zwischen der Befliegung und den dazugehörigen Flugstreifen (über die Befliegungs-ID) ermöglicht ein einfaches und vor allem eindeutiges Ableiten dieser Informationen.

```
UPDATE tdtaflight SET nrofstrips=(SELECT COUNT(idstrip) FROM tdtastrip
     WHERE tdtastrip.idflight=tdtaflight.idflight);
```

Das Attribut *Anzahl der Streifen* je Befliegung wird durch einfaches Zählen der Flugstreifen mit der selben Befliegungs-ID in der Tabelle Flugstreifen erreicht.

```
UPDATE tdtaflight SET timeoffset=timeofweek(flightdate)
```

Das Attribut *Datum des Wochenbeginns* wird für die Berechnung des Attributs *zeitlicher Beginn* (Datum), *zeitliches Ende* (Datum) der Befliegung und in weiterer Folge für die Berechnung der absoluten zeitlichen Verortung aller Datensätze, die als Attribut Zeit in Sekunden seit Wochenbeginn haben, herangezogen. Die Funktion *timeofweek()* gibt für ein beliebiges Datum den Wochenbeginn – sprich das Datum des davorliegenden Sonntags – zurück.

```
UPDATE tdtaflight SET starttime=timeoffset+((SELECT MIN
     (tdtastrip.starttimesec) FROM tdtastrip WHERE
     tdtastrip.idflight=tdtaflight.idflight)::INTEGER|| '
     seconds')::INTERVAL;
```

```
UPDATE tdtaflight SET endtime=timeoffset+((SELECT MAX
     (tdtastrip.starttimesec) FROM tdtastrip WHERE
     tdtastrip.idflight=tdtaflight.idflight)::INTEGER|| '
```

```
      seconds')::INTERVAL;
```

Die Attribute *zeitlicher Beginn* und *zeitliches Ende* ergeben sich, wenn man zum Wochenbeginn die minimale Zeit aller Flugstreifen der jeweiligen Befliegung bzw. die maximale Zeit addiert. Die Attribute stellen ein absolutes Datum mit dazugehöriger Uhrzeit dar.

7.3.2 Attribute des Flugstreifens

Die *Anzahl der Laserpunkte* je Flugstreifen werden nach dem gleichen Prinzip wie die *Anzahl der Flugstreifen* je Befliegung aus den vorliegenden Datensätzen abgeleitet.

```
UPDATE tdtastrip SET nrofpoints=(SELECT COUNT(idscanpoint) FROM
      tdtascanpoint WHERE tdtascanpoint.idstrip=tdtastrip.idstrip);
```

Da bei den Laserpunkten kein Index auf dem Attribut Flugstreifen-ID liegt, muss die gesamte Tabelle mit den Laserpunkten sequentiell durchlaufen und die Anzahl der Laserpunkte je Flugstreifen gezählt werden. Je nach Größe der Tabelle kann dieser Vorgang mehrere Minuten in Anspruch nehmen. Auf die Unterscheidung zwischen *first* und *last pulse* wird verzichtet.

Die *Geometrie der Flugstreifen* – die Repräsentation des Flugstreifens in Form einer dreidimensionalen Linie – ergibt sich aus den Flugzeugpositionen als Stützpunkte dieser Linie. Über das Attribut Flugstreifen-ID können die Flugzeugpositionen des jeweiligen Flugstreifens abgefragt werden. Die zeitliche Verortung der Flugzeugpositionen ermöglicht ein korrektes Zusammensetzen der Linie (richtige Reihenfolge!). Die Flugstreifen werden nahezu geradlinig geflogen. Mit dem Douglas-Peucker-Algorithmus mit einer frei wählbaren Toleranz werden die Stützpunkte stark ausgedünnt, was zu einer Reduktion der Datenmenge, aber zu keinem wesentlichen Verlust der Lagegenauigkeit der Flugstreifenlinie, führt. Als Parameter werden dem Algorithmus die Geometrie (Linie) und eine Toleranz [m], die den Grad der Generalisierung bestimmt, übergeben. Linienstützpunkte, die den Linienverlauf nicht wesentlich beeinflussen, werden entfernt. Die ursprüngliche Linienform bleibt erhalten (RAMSEY 2004, 40; EKLUND et. al. 2001, 107f.; DOUGLAS & PEUCKER 1973). In Tabelle 7.1 sind drei Generalisierungsstufen (Toleranzen: 0.5 m, 1 m und 2 m) gegenübergestellt.

id	Laser-punkte	Flugzeug-positionen	Länge (3-D) [m]	$\bar{v}$ [m/s]	Länge (3-D) [m]	Diff. [m]	n	Länge (3-D) [m]	Diff. [m]	n	Länge (3-D) [m]	Diff. [m]	n
					Toleranz 0.5 m			Toleranz 1 m			Toleranz 2 m		
1	18926720	47868	16626.82	69.47	16626.69	0.13	50	16626.61	0.21	33	16626.47	0.35	20
2	18567812	47724	16280.58	68.23	16280.49	0.09	44	16280.44	0.14	28	16280.30	0.28	14
3	20538871	53400	18576.75	69.58	18576.65	0.10	52	18576.59	0.16	33	18576.47	0.28	21
4	20294858	52465	18332.67	69.89	18332.55	0.12	56	18332.47	0.20	35	18332.39	0.28	22
5	4390806	10600	3343.10	63.08	3343.08	0.02	9	3343.07	0.03	7	3343.05	0.05	5
6	4975915	12000	3860.92	64.36	3860.83	0.09	12	3860.80	0.12	8	3860.77	0.15	6
7	21254477	55874	18991.34	67.98	18991.05	0.29	58	18990.94	0.40	40	18990.67	0.67	22
8	4351273	10500	3543.61	67.51	3543.58	0.03	14	3543.56	0.05	9	3543.51	0.10	6
9	4193905	10250	3348.31	65.34	3348.27	0.04	12	3348.26	0.05	8	3348.20	0.11	5
10	4322844	10600	3572.78	67.42	3572.74	0.04	14	3572.72	0.06	9	3572.70	0.08	7
11	6042728	14800	4476.63	60.50	4476.52	0.11	16	4476.48	0.15	12	4476.41	0.22	6
12	4719228	11551	3954.69	68.48	3954.45	0.24	10	3954.36	0.33	8	3954.34	0.35	6
13	20187310	51788	18220.52	70.37	18220.40	0.12	54	18220.34	0.18	37	18220.23	0.29	24
14	5196064	13228	4248.46	64.24	4248.39	0.07	16	4248.33	0.13	9	4248.30	0.16	7
15	4892486	11800	3804.09	64.48	3804.05	0.04	16	3804.02	0.07	11	3804.00	0.09	9
16	4778368	11500	4002.02	69.61	4001.92	0.10	16	4001.87	0.15	12	4001.79	0.23	8
17	21505356	55555	18460.47	66.46	18460.21	0.26	52	18460.07	0.40	35	18459.73	0.74	20

Tab. 7.1: Gegenüberstellung von drei Generalisierungsstufen (0.5 m, 1 m und 2 m Toleranz) mit dem Douglas-Peucker-Algorithmus.
Von links nach rechts: Flugstreifen-ID, Anzahl der Laserpunkte, Anzahl der Flugzeugpositionen, 3-D-Länge des Flugstreifens und mittlere Fluggeschwindigkeit anhand aller Flugzeugpositionen, Länge der generalisierten Linie, Längendifferenz zur Originallinie, Anzahl der Stützpunkte (n) der generalisierten Linie, usw.

Geometrie, Länge, Fluggeschwindigkeit der Flugstreifen	
Toleranz [m]	**Rechenzeit in ms**
0.1	76222.991
0.5	72386.405
1	72538.625
2	72502.016
5	73067.206

Tab. 7.2: Rechenzeiten für die Berechnung und Speicherung der Attribute Geometrie, Länge und mittlere Fluggeschwindigkeit der Flugstreifen in Tab. 7.1. Die Toleranz für die Generalisierung der Geometrie mit dem Douglas-Peucker-Algorithmus wird variiert.

Gleichzeitig mit der Berechnung der Liniengeometrie für die Flugstreifen wird die *Länge des Flugstreifens* und die *mittlere Fluggeschwindigkeit* berechnet und abgespeichert. Die Länge des Flugstreifens wird bis dato ohne Berücksichtigung des Ellipsoids (Länge auf dem Ellipsoid) angegeben. Die mittlere Fluggeschwindigkeit ergibt sich aus dem Quotienten Länge durch zeitliche Dauer des Flugstreifens (in Metern pro Sekunde). Die PL/pgSQL-Funktion für die Berechnung der Geometrie der Streifen ist in Appendix B.2 zu finden.

7.4 Performancetests

Die unten angeführten Performancetests können nur eine relative Aussage über die Geschwindigkeiten der verschiedenen Importmethoden geben, da die Hardware und die Softwareeinstellungen (Softwaretuning) die Rechenzeiten maßgeblich steuern.

7.4.1 INSERT versus COPY

Datensätze (Index)	COPY	INSERT	PUTLINE (pg)
100.000 ohne GIST	7s	3m 37s	8s
100.000 mit GIST	12s	3m 50s	11s
1.000.000 mit GIST	1m 46s	35m 33s	1m 50s
1.000.000 ohne GIST	1m 17s	33m 27s	1m 20s
50.000.000 mit GIST	1h 54m 57s	ca. 30h*	-
50.000.000 ohne GIST	1h 03m 51s	ca. 28,5 h*	-

Tab. 7.3: Einfache Importszenarien mit drei verschiedenen Methoden und einer unterschiedlichen Anzahl von Datensätzen (mit und ohne Index GIST auf der Geometriespalte). * nicht getestet, sondern hochgerechnet.

Die Tests in Tabelle 7.3 zeigen die Anzahl der importierten Datensätze, die Importmethode und die Zeit, die für den Import benötigt wird. Die Datensätze werden in einem Python-Skript dynamisch erzeugt (siehe Appendix B.1). Es zeigt sich deutlich der Unterschied zwischen den Importzeiten zwischen COPY und INSERT. Beim Import mit COPY werden die Datensätze zuerst in eine Datei gespeichert, die dann in die Datenbank geschrieben wird. Ferner ist zu sehen, dass die Funktion *putline()* des pg-Moduls, obwohl sie die Daten direkt in die Datenbank schreibt, nicht schneller als der Import aus einer Datei ist. Logischerweise ist das Schreiben von Daten in eine indexierte Spalte langsamer als in eine Spalte ohne Index. Der Aufbau des Indexes erst nach dem Import dauert nur unwesentlich kürzer als die Zeitdifferenz zwischen Import mit und ohne Index. Die Erstellung des Indexes für 50 Millionen Datensätze dauert 35m 47s (Zeitdifferenz aus Tab. 7.3: 51m 6s). Die oben genannten Importzeiten beziehen sich auf eine idealisierte Tabelle, die vor dem Import keine Daten enthält und keine Relationen zu anderen Tabellen aufweist.

7.4.2 Import der Rohdaten

Der Import der Rohdaten in das in Kapitel *„6 Datenverwaltung und -speicherung“* vorgestellte Datenbankschema nimmt mehr Zeit in Anspruch als die Tests (Tab. 7.3) vermuten lassen. Dies hängt damit zusammen, dass Daten nicht nur in eine, sondern in mehrere Tabellen geschrieben werden, die einen oder mehrere Indizes und Relationen besitzen. Die Relationen müssen nach jedem Import neu überprüft werden. Die Indizes müssen nach dem Einfügen von neuen Datensätzen auf den neuesten Stand gebracht werden, damit nachfolgende Abfragen möglichst schnell ablaufen können. In Tabelle 7.4 sind die

Rechenzeiten für den Import von unterschiedlich vielen Laserpunkten und Flugzeugpositionen angegeben. Sehr augenscheinlich ist der Unterschied zwischen dem Import mit oder ohne Softwaretuning.

Laserpunkte	Flugzeugpositionen	Dauer	Bemerkungen
2.308.848	591.387	35m 12s	ohne Softwaretuning*
2.308.848	591.387	18m 51s	mit Softwaretuning*
45.623.162	692.648	10h 32m 31s	mit Softwaretuning*, inkl. update_geometry_stats(), VACUUM ANALYZE.
189.139.021	1.976.602	43h 26m 4s	mit Softwaretuning*, inkl. update_geometry_stats(), VACUUM ANALYZE (ca. 9h)

Tab. 7.4: Importzeiten der Rohdaten mit *putline()*. ** siehe Kapitel „10 Evaluierung und Ausblick“*

Zusammenfassung

In diesem Kapitel wurde ein Übergang zwischen dem theoretischen Aufbau der Datenverwaltung (Datenbankstruktur) und der Datenverarbeitung im nachfolgenden Kapitel geschaffen. Das derzeitige Datenformat der Rohdaten – wie die Daten von der Befliegungsfirma geliefert werden – macht ein komplexes Auslesen der Datenfiles notwendig. Die Daten können nicht direkt in die Datenbank importiert werden. In mehreren Schleifen, die in einem Python-Skript verwirklicht wurden, werden die einzelnen Dateien ausgelesen, wobei es gleichzeitig zum Aufbau der Relationen zwischen den Entitäten kommt. Ausführliche Performancetests zeigten die schnellste Importmethode, die sich speziell für große Datenmengen eignet. Es handelt sich dabei um den PostgreSQL-Befehl COPY, der direkt aus einem Pythonprogramm mit Hilfe des pg-Moduls ausgeführt werden kann. Im Zuge des Datenimports werden die Daten nicht nur in die Tabellen der Datenbank eingefügt, sondern es werden auch schon erste Attribute der Entitäten abgeleitet.
In Zukunft ist sicherlich anzustreben, dass die Befliegungsfirma ihr Exportformat so wählt, dass die Relationen zwischen den einzelnen Entitäten von vornherein gegeben sind und ein direkter Import möglich ist, was zu einer wesentlichen Reduktion der Importzeiten führen wird.

8 Datenverarbeitung

8.1 Einleitung

Im Kapitel *„8 Datenverarbeitung“* wird die Zusammenführung der drei Hauptkomponenten des Informationssystems (GIS, räumliche Datenbank und Statistiksoftware) beschrieben. Der größte Teil der Datenverarbeitung findet in der räumlichen Datenbank statt. Die prozedurale Sprache PL/pgSQL erlaubt die Entwicklung von komplexen Funktionen mit einfachen Methoden. Dieses Kapitel soll dem Leser erklären, wie multitemporale Analysen möglich werden, wie die Aufnahmegeometrie für jeden Laserpunkt rekonstruiert wird, welche statistische Auswertemethoden im Informationssystem zur Verfügung stehen und wie die Berechnung von Rasterdaten umgesetzt ist. Anhand von konkreten Beispielen wird die Leistungsfähigkeit der einzelnen Funktionen aufgezeigt. Da der geschriebene Quelltext keinen Lizenzbestimmungen unterliegt, muss auf die Veröffentlichung des Quelltextes verzichtet werden. Die folgenden Erläuterungen jedoch sollen die Nachvollziehbarkeit der einzelnen Arbeitsschritte gewährleisten.

8.2 Absolute zeitliche Verortung

Für multitemporale Auswertungen der Laserscannerdaten ist es unbedingt notwendig eine absolute zeitliche Verortung aller Datensätze vorliegen zu haben. Wie in Kapitel *„4 Datenerfassung und Datengrundlagen“* beschrieben ist, besitzen die Flugzeugpositionen und die Laserpunkte ein Attribut *Zeit*. Die Zeit wird in Sekunden seit Wochenbeginn angegeben. Dieses Zeitformat hat seine Berechtigung, da viel Speicherplatz gespart wird, wenn nicht mit jedem Laserpunkt ein absolutes Datum abgespeichert werden muss. Die absolute zeitliche „Verankerung“ wird einmal für jede Befliegung (Attribut *Datum des Wochenbeginns*) abgelegt (vgl. *„7.3.1 Attribute der Befliegung“*). Jedem Laserpunkt und jeder Flugzeugposition kann über die eindeutige Beziehung zum Flugstreifen und dessen eindeutige Beziehung zur Befliegung eine absolute zeitliche Verankerung zugewiesen werden (vgl. Abb. 6.2). Werden nun zum *Datum des Wochenbeginns* die Sekunden seit Wochenbeginn des entsprechenden Datensatzes dazugezählt, erhält man Datum und Uhrzeit in der Form **dd.mm.yyyy hh:mm:ss** (Bsp.: 05.11.2003 11:14:44.94872). Die Uhrzeit kann bis auf eine Mikrosekunde genau angegeben werden (EISENTRAUT 2003, 113). Das ausgegebene Datumsformat hängt in erster Linie von den Einstellungen der Datenbank ab. Es sollte darauf geachtet werden, dass ein europäisches Datumsformat eingestellt ist, damit der Tag vor dem Monat im *timestamp* aufgeführt wird (SET DATESTYLE TO 'European'; EISENTRAUT 2003, 114).

Beispiel: Wochenbeginn + Anzahl der Sekunden seit Wochenbeginn
02.11.2003 00:00:00 + 208209.55325 = 04.11.2003 09:50:09.55325

SQL-Befehl für die absolute zeitliche Verortung einer relativen Zeitangabe

```
abs_time(Sekunden seit Wochenbeginn::NUMERIC, Flugstr.-ID::INTEGER)
```

Beispiel:

```
SELECT idplaneposition, time, abs_time(time, idstrip) FROM
tdtaplaneposition LIMIT 10;
```

idplaneposition	time	abs_time
1526230	208209.55325	04.11.2003 09:50:09.55325
1526231	208209.55825	04.11.2003 09:50:09.55825
1526232	208209.56325	04.11.2003 09:50:09.56325
1526233	208209.56825	04.11.2003 09:50:09.56825
1526234	208209.57325	04.11.2003 09:50:09.57325
1526235	208209.57825	04.11.2003 09:50:09.57825
1526236	208209.58325	04.11.2003 09:50:09.58325
1526237	208209.58825	04.11.2003 09:50:09.58825
1526238	208209.59325	04.11.2003 09:50:09.59325
1526239	208209.59825	04.11.2003 09:50:09.59825

Tab. 8.1: Berechnung der absoluten Zeit für 10 Flugzeugpositionen.

absolute zeitliche Verortung	
Anzahl der Datensätze	Rechenzeit in ms
10	3.906 (= 0.003906 s)
100	98.090
1000	439.507
10000	3794.002
100000	43481.163
1000000	213675.130

Tab. 8.2: Rechenzeiten für die absolute zeitliche Verortung einer unterschiedlichen Anzahl von Datensätzen.

In Tabelle 8.1 ist der Einsatz der neu entwickelten SQL-Funktion für die Berechnung der absoluten Zeit von 10 Flugzeugpositionen dargestellt. Die Rechenzeiten für die *on-the-fly* Berechnung der absoluten zeitlichen Verortung sind in Tabelle 8.2 aufgelistet.

8.3 Aufnahmegeometrie

8.3.1 Gründe für die Rekonstruktion der Aufnahmegeometrie

LUTZ (2003) rekonstruierte die Aufnahme- bzw. Scangeometrie einer Laserscannerbefliegung eines norwegischen Gletschers (Svartisheibreen) auf Rasterbasis. Er setzte die Aufnahmegeometrie in Beziehung zu den Intensitätswerten der Oberflächentypen Schnee, Eis, Wasser und Fels. Als Ergebnis dieser Untersuchung kann festgehalten werden, dass die Intensitätswerte der einzelnen Laserpunkte zum größten Teil auf die Art der Oberfläche und zu einem kleineren Teil auf die Aufnahmegeometrie (Länge des Laservektors, Einfallswinkel) zurückzuführen sind.
Wenn man nun den Einfluss der Aufnahmegeometrie quantifiziert und in weiterer Folge entfernt, erhält man ein ungestörtes Bild der Reflexionseigenschaften der gescannten Oberfläche. Der nächste Schritt ist die Ableitung von Oberflächeneigenschaften (z.B. Oberflächenrauigkeit) auf Basis der Intensitätswerte. Diese Art von Rauigkeit könnte man als „*innere Rauigkeit*“ bezeichnen, da sie auf die Fläche des *footprints* beschränkt ist. Diese *innere Rauigkeit* in Kombination mit der topographischen Information am Laserpunkt, aus der man die Struktur des Geländes ableiten kann, also auf die Oberfläche zwischen den einzelnen *footprints* schließt („*äußere Rauigkeit*“), ermöglicht eine Beschreibung und vielleicht sogar eine Abgrenzung und Identifizierung der Phänomene dieser Oberfläche.
Im Rahmen dieser Diplomarbeit soll der Grundstein für das oben beschriebene Szenario gesetzt werden, indem die Aufnahmegeometrie für jeden Laserpunkt in einem rein vektorbasierten Modell berechnet wird. Die größten Vorteile dieser neuen Vorgehensweise sind die Beibehaltung der maximalen Genauigkeit, das Heranziehen der unveränderten Punktdaten (keine Interpolation !) und die Prozessierung der Daten an Ort und Stelle, d.h. in der Datenbank, wo sie gespeichert sind.

8.3.2 Rekonstruktion der Flugzeugposition für jeden Laserpunkt

Über das Attribut *Flugstreifen-ID* kann eine Verbindung zwischen Flugzeugpositionen und Laserpunkten hergestellt werden. Die Laserpunkte werden mit einer Messrate von 50.000 Punkten pro Sekunde (Hz) aufgezeichnet. Die Flugzeugpositionen, die ursprünglich mit 2 Hz GPS-Positionen aufgezeichnet wurden, werden von der Befliegungsfirma mit Hilfe der INS-Daten auf 200 Hz gerechnet. Damit ergibt sich ein Verhältnis von 250 Laserpunkten auf eine Flugzeugposition. Der Zeitabstand zwischen den Flugzeugpositionen beträgt somit 0.005 Sekunden. Aufgrund der Tatsache, dass das Flugzeug in diesen 0.005 Sekunden keine wesentlichen Richtungs- oder Geschwindigkeitsänderungen machen kann, wird für die Berechnung der Flugzeugposition für jeden Laserpunkt ein linearer Interpolationsansatz gewählt. Bei einer angenommenen Fluggeschwindigkeit von 70 m/s beträgt der Abstand zwischen den Flugzeugpositionen 35 cm (Abb. 8.1). In weiterer Folge werden 250 zusätzliche Flugzeugpositionen auf diese Strecke von 35 cm interpoliert. Der Abstand zwischen den interpolierten Flugzeugpositionen beläuft sich auf 0.0014 m (=1.4 mm). Dieses

Rechenbeispiel soll verdeutlichen, in welchem Maßstabsbereich sich die Interpolation abspielt.

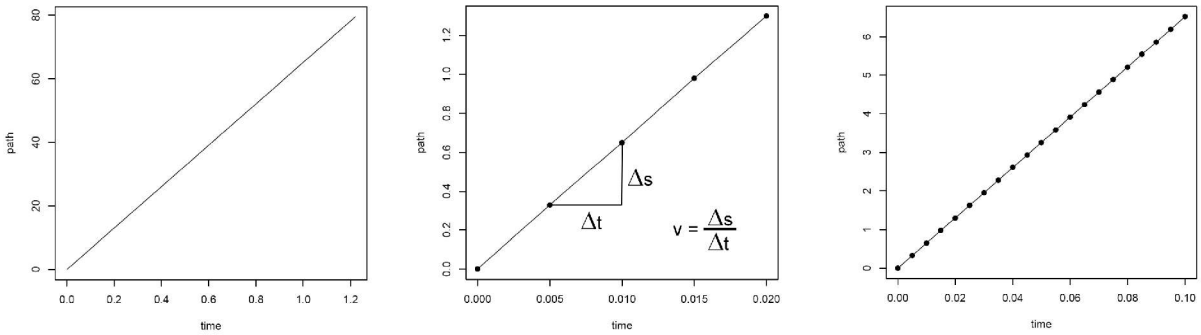


Abb. 8.1: Weg-Zeit-Diagramme des Flugzeugs in unterschiedlichen Skalen (eigener Entwurf).

Ausgehend vom Laserpunkt können alle Flugzeugpositionen desselben Flugstreifens selektiert werden (gleiche Flugstreifen-ID !). Je nach Frequenz in der die Flugzeugpositionen (200 Hz) vorliegen, wird ein unterschiedlich großes zeitliches Suchfenster verwendet, um die zwei zeitlich benachbarten Flugzeugpositionen für den entsprechenden Laserpunkt zu finden. Durch die Rundung der letzten Stelle des Zeitattributs der Flugzeugpositionen kann es zu Verschiebungen in Richtung der Zeit des Laserpunkts (Bsp. 3, siehe unten) oder zeitlich weg vom Laserpunkt (Bsp. 2) kommen. Dieses Problem kann durch die Vergrößerung des zeitlichen Suchfensters – Erhöhung der letzten Kommastelle um 1 – gelöst werden. Bei 200 Flugzeugpositionen pro Sekunde (200 Hz) werden alle Flugzeugpositionen, die 0.0051 Sekunden vor oder nach dem Laserpunkt aufgezeichnet wurden, selektiert.

Beispiel 1:	Zeit des Laserpunkts:	125829.9965**6**	
	1. Stützpunkt:	125829.9945**6**	
	2. Stützpunkt:	125829.9995**6**	*exakt 200 Hz (0.005 s)*
Beispiel 2:	Zeit des Laserpunkts:	125829.9965**6**	
	1. Stützpunkt:	125829.9945**6**	
	2. Stützpunkt:	125829.9995**7**	*Rundung weg vom Laserpunkt !!!*
Beispiel 2:	Zeit des Laserpunkts:	125829.9965**6**	
	1. Stützpunkt:	125829.9945**6**	
	2. Stützpunkt:	125829.9995**5**	*Rundung zum Laserpunkt !!!*

Folgende Szenarien können sich bei der Selektion der Flugzeugpositionen ergeben:

1) Für die Zeit des Laserpunkts wird keine Flugzeugposition von 0.0051 Sekunden davor bis 0.0051 Sekunden danach gefunden. Die Daten wurden entweder nicht importiert oder in den Rohdaten fehlen sie ganz. Oder das zeitliche Suchfenster wurde zu klein gewählt. Die Frequenz der Flugzeugpositionen ist geringer als angenommen. → Keine Interpolation möglich!

2) Es wird nur *eine* Flugzeugposition gefunden. Der Laserpunkt befindet sich am Rand einer Datenlücke der Flugzeugpositionen. → Die Flugzeugposition könnte durch die Fortführung eines Trends berechnet werden. Auf eine Interpolation wird verzichtet.
3) Es werden zwei Flugzeugpositionen gefunden. Eine vor der Aufnahme des Laserpunktes und eine danach. → Die Flugzeugposition für diesen Laserpunkt wird linear interpoliert.
4) Es werden mehr als drei Flugzeugpositionen gefunden. Das zeitliche Suchfenster bzw. die Frequenz wurde falsch gewählt. Die Flugzeugpositionen liegen höher frequentiert vor. → Die zwei zeitlich nächsten Positionen werden als Stützpunkte für die Interpolation herangezogen.
5) Der Laserpunkt wurde zur exakt gleichen Zeit wie eine Flugzeugposition aufgenommen. → Eine Interpolation ist nicht nötig. Dem Laserpunkt wird diese Position zugewiesen.

$$ratio = \frac{t_L - t_1}{t_2 - t_1} \tag{8.1}$$

t_1...Zeit des ersten Stützpunktes, t_2...Zeit des zweiten Stützpunktes, t_L...Zeit des Laserpunktes

Das Verhältnis (*Ratio*) Zeitdifferenz zwischen dem Laserpunkt und dem ersten Stützpunkt zu Zeitdifferenz zwischen dem zweiten Stützpunkt und dem ersten Stützpunkt (Formel 8.1) wird dazu verwendet den dreidimensionalen Vektor von Stützpunkt 1 zu Stützpunkt 2 proportional zu verkürzen (Formeln 8.2, 8.3, 8.4). Der *Ratio* sorgt dafür, dass es egal ist, in welcher zeitlichen Reihenfolge die Stützpunkte 1 und 2 liegen ($t_1 > t_2$ oder $t_1 < t_2$) und wie groß der zeitliche Abstand zwischen ihnen ist. Voraussetzung ist, dass $t_1 < t_L < t_2$ oder $t_2 < t_L < t_1$ und dass $t_1 \neq t_2$, da es sonst zu einer Division durch Null kommt.

$$x_L = x_1 + (x_2 - x_1) \cdot ratio \qquad y_L = y_1 + (y_2 - y_1) \cdot ratio \qquad z_L = z_1 + (z_2 - z_1) \cdot ratio \tag{8.2, 8.3, 8.4}$$

(x_L, y_L, z_L)...Koordinaten des Laserpunktes, (x_1, y_1, z_1)...Koordinaten des ersten Stützpunktes,
(x_2, y_2, z_2) ...Koordinaten des zweiten Stützpunktes

Die Koordinaten der interpolierten Flugzeugpositionen werden auf Zentimeter gerundet. Mehr Nachkommastellen würden das Ergebnis nicht besser machen und v.a. den Speicherplatzverbrauch erhöhen. Die Rundung der Koordinaten führt bei den geringen zeitlichen Abständen (1 / 50.000 s) zu gleichen Flugzeugpositionen für Laserpunkte, die zeitlich unmittelbar nebeneinander liegen. Gleichzeitig kommt es zu Rundungssprüngen in den Koordinaten und somit zur Versetzung der interpolierten Punkte um maximal die Hälfte des Wertes der zurundenden Kommastelle. Im großen Maßstab betrachtet, liegen die linear interpolierten Punkte, die auf Zentimeter gerundete Koordinaten besitzen, nicht auf einer Linie (Abb. 8.2).

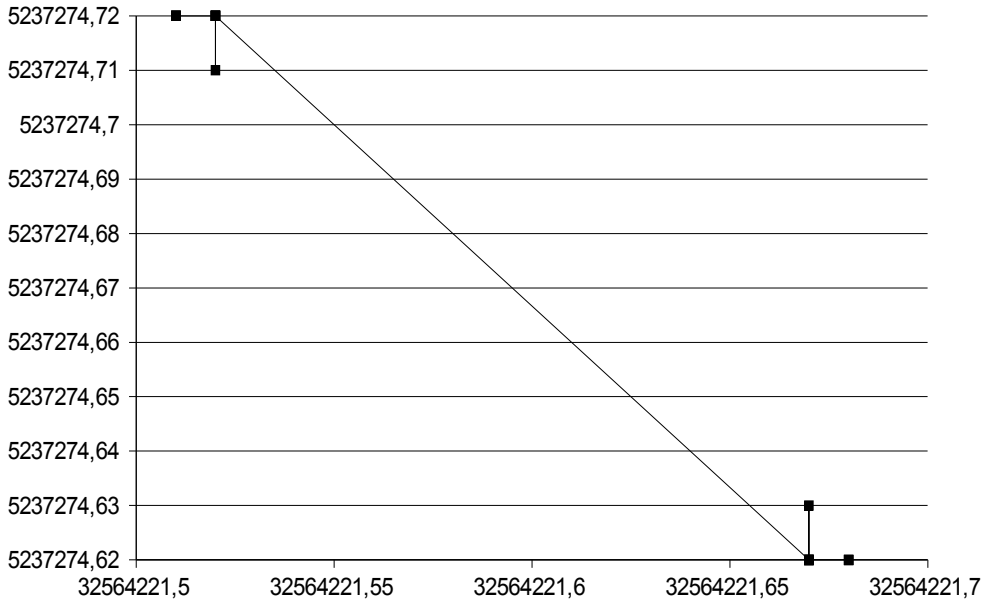


Abb. 8.2: Beispiel von 20 interpolierten Flugzeugpositionen (UTM-East, UTM-North) (eigener Entwurf).

In Abb. 8.2 sind die Konzentration mehrerer Punkte auf eine Position und die Sprünge einzelner Punkte, beides durch die Rundung der Koordinaten auf Zentimeter begründet, zu erkennen.

SQL-Befehl für die lineare Interpolation der Flugzeugposition für einen Laserpunkt

```
planepos(Sekunden seit Wochenbeginn::NUMERIC, Flugstr.-ID::INTEGER)
```

Beispiel:

```
SELECT idscanpoint, time, planepos(time, idstrip) FROM tdtascanpoint
ORDER BY idscanpoint LIMIT 10;
```

idscanpoint	time	planepos
1	299684.948720	SRID=-1;POINT(32564223.400 5237273.570 2540.060)
2	299684.948740	SRID=-1;POINT(32564223.400 5237273.570 2540.060)
3	299684.948760	SRID=-1;POINT(32564223.400 5237273.570 2540.060)
4	299684.979220	SRID=-1;POINT(32564221.680 5237274.620 2540.060)
5	299684.979240	SRID=-1;POINT(32564221.680 5237274.620 2540.060)
6	299684.979260	SRID=-1;POINT(32564221.680 5237274.620 2540.060)
7	299684.979280	SRID=-1;POINT(32564221.670 5237274.620 2540.060)
8	299684.979300	SRID=-1;POINT(32564221.670 5237274.620 2540.060)
9	299684.979320	SRID=-1;POINT(32564221.670 5237274.620 2540.060)
10	299684.979340	SRID=-1;POINT(32564221.670 5237274.620 2540.060)

Tab. 8.3: Berechnung der Flugzeugposition für ausgewählte Laserpunkte.

Wie in Tabelle 8.3 ersichtlich, gibt die Funktion *planepos* die interpolierte Flugzeugposition in Form der so genannten *Well-Known Text (WKT)* Repräsentation einer Geometrie, die der OpenGIS-Spezifikation entspricht, zurück (Ramsey 2004, 9). Die Funktion gibt die Geometrie mit der *SRID (spatial referencing system identifier)* zurück, in der die Flugzeugpositionen vorliegen.

Interpolation der Flugzeugposition	
Anzahl der Datensätze	**Rechenzeit in ms**
10	5.380
100	151.696
1000	573.904
10000	4822.474
100000	50984.743
1000000	536023.787

Tab. 8.4: Rechenzeiten für die lineare Interpolation der Flugzeugposition für ausgewählte Laserpunkte.

Die Rechenzeiten in Tabelle 8.4 enthalten den gesamten Prozess der Suche nach den passenden Stützpunkten, die Berechnung des *Ratios,* die Konstruktion eines verkürzten Vektors, der schlussendlich die interpolierte Flugzeugposition berechnen lässt, sowie die Rückgabe des Ergebnisses in Form einer *WKT-Geometrie*.

8.3.3 Länge des Laservektors (*range*)

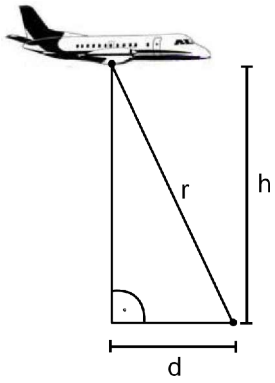


Abb. 8.3: Die Länge des Laservektors (r) als Entfernung zwischen der Flugzeugposition und dem Laserpunkt (eigener Entwurf).

Die Entfernung zwischen dem Laserpunkt und seiner Flugzeugposition (Position des Flugzeugs bei der Aufnahme des Laserpunkts) wird aus der horizontalen und vertikalen Distanz zwischen diesen beiden Punkten berechnet.

$$r = \sqrt{d^2 + h^2} \tag{8.5}$$

r...Länge des Laservektors, d...horizontale Distanz, h...vertikale Distanz (vgl. Abb. 8.3)

SQL-Befehl für die Berechnung der Länge des Laservektors (= Entfernung zwischen zwei Punkten im dreidimensionalen Raum)

```
range(Position des Laserpunkts::GEOMETRY,Flugzeugposition::GEOMETRY)
```

Beispiel:

```
SELECT idscanpoint, geom_point, geom_planepos, range(geom_point,
geom_planepos) FROM tdtascanpoint ORDER BY idscanpoint LIMIT 10;
```

idscanpoint	geom_point	geom_planepos	range
1	SRID=-1;POINT (32564342.775237585.842007.3)	SRID=-1;POINT (32564223.45237273.572540.06)	628.96
2	SRID=-1;POINT (32564342.45237585.222007.25)	SRID=-1;POINT (32564223.45237273.572540.06)	628.63
3	SRID=-1;POINT (32564342.055237584.632007.14)	SRID=-1;POINT (32564223.45237273.572540.06)	628.36
4	SRID=-1;POINT (32564338.35237582.782007.34)	SRID=-1;POINT (32564221.685237274.622540.06)	626.38

5	SRID=-1;POINT (32564338.665237583.392007.39)	SRID=-1;POINT (32564221.685237274.622540.06)	626.71
6	SRID=-1;POINT (32564338.975237583.952007.36)	SRID=-1;POINT (32564221.685237274.622540.06)	627.10
7	SRID=-1;POINT (32564339.285237584.482007.38)	SRID=-1;POINT (32564221.675237274.622540.06)	627.37
8	SRID=-1;POINT (32564339.585237584.992007.42)	SRID=-1;POINT (32564221.675237274.622540.06)	627.64
9	SRID=-1;POINT (32564339.95237585.542007.41)	SRID=-1;POINT (32564221.675237274.622540.06)	627.99
10	SRID=-1;POINT (32564340.255237586.142007.5)	SRID=-1;POINT (32564221.675237274.622540.06)	628.27

Tab. 8.5: Berechnung der Länge des Laservektors für ausgewählte Laserpunkte.

Länge des Laservektors	
Anzahl der Datensätze	**Rechenzeit in ms**
10	0.694
100	3.636
1000	79.359
10000	521.790
100000	3881.148
1000000	36969.029

Tab. 8.6: Rechenzeiten für die Berechnung der Länge des Laservektors (*range*) bei gegebener Flugzeugposition.

WAGNER et. al. (2003) zeigen, von welchen Einflussfaktoren die Intensität des reflektierten Laserstrahls bestimmt wird. Die aus der Mikrowellenfernerkundung entlehnte Radargleichung (Formel 8.6) schlüsselt die Einflussfaktoren auf.

$$P_E = \frac{P_S D_E^{\,2}}{4\pi R^4 \beta_S^{\,2}} \cdot \eta_{SYS}\, \eta_{ATM} \cdot \sigma \tag{8.6}$$

Radargleichung (WAGNER et. al. 2003, 230):

P_E...Empfangene Leistung
P_S...Gesendete Leistung
D_E...Apertur des Empfängers
R...Entfernung
β_S...Öffnungswinkel Laserstrahl
η_{SYS}...Wirkungsgrad des Laserscanners
η_{ATM}...Transmissionsfaktor der Atmosphäre
σ...Streuquerschnitt

Formel 8.6 zeigt den Einfluss der Länge des Laservektors – die Entfernung zwischen Aufnahmesystem und aufgenommener Erdoberfläche – auf die empfangene Leistung. Viele der Variablen in Formel 8.6 können für eine Laserscannerbefliegung als konstant angenommen werden (z.B. der Öffnungswinkel des Laserscanners). Der *range* vermindert die Intensität des reflektierten Signals in vierter Potenz. Ferner ist der *range* für die Fläche des Streuquerschnitts (vgl. *footprint*) mit verantwortlich. Dieser doppelte Einfluss des *range* sollte bei der Arbeit mit den Intensitätswerten im Hinterkopf behalten werden. Wenn man diesen „geometrischen Anteil" aus dem Intensitätswert entfernt, erhält man ein „ungetrübtes" Bild der Reflektionseigenschaften der Oberfläche. Auf eine geometrische Korrektur der Intensitätswerte kann im Rahmen dieser Diplomarbeit nicht näher eingegangen werden.

8.3.4 Geländemodellierung

8.3.4.1 Methodik

Im Rasterdatenmodell kann man durch die Nachbarschaftsumgebung einer Rasterzelle auf die Geländeparameter dieser Zelle schließen. Eine Zelle isoliert betrachtet, gibt noch keine Auskunft über die Neigung oder die Exposition an dieser Stelle. Erst wenn man diese Zelle in Beziehung zu den Nachbarzellen setzt, kann mit verschiedenen Algorithmen auf ein „mittleres" Gelände geschlossen werden.
Ein ähnliches Problem liegt an, wenn man Geländeparameter aus Punkten, die das Gelände repräsentieren, ableiten will. Die Berechnung von Geländeparametern an einem Punkt erfordert es – vergleichbar mit dem Raster – diesen Punkt in Beziehung zu seinen Nachbarn zu setzen. In der Vektorwelt ist die *Delauney-Triangulation*, die wohl bekannteste Methode, um aus Punkten ein Gelände abzuleiten (EKLUNDH et. al. 2001, 184). Die Punkte werden dabei als Eckpunkte für aufgespannte Dreiecksflächen herangezogen. Diese Fläche, deren Neigung und Exposition von den drei Stützpunkten abhängt, repräsentiert das Gelände. Aber genau an diesen Stützpunkten, die den Laserpunkten entsprechen, können keine Geländeparameter abgeleitet werden. Eine Mittelwertsbildung aus allen angrenzenden Dreiecksflächen oder die Berechnung einer komplexen mathematischen Trendfläche mit mehreren Nachbarpunkten, führt bei genaueren Überlegungen zu keiner befriedigenden Lösung. Der große Rechenaufwand, der für jeden Laserpunkt wiederholt werden müsste, und das unveränderte Problem bei Spezialfällen (z.B. Dachkante, Vegetation, usw.), führten zur Wahl eines sehr einfachen Verfahrens (Abb. 8.4). Zur Demonstration wie wichtig die Rechenzeiten sind, soll gezeigt werden, wie sehr sich augenscheinlich kleine Unterschiede in der Rechenzeit für eine große Anzahl an Datensätzen in Summe auswirken können (Tab. 8.7).

Rechenzeit pro Laserpunkt [Sekunden]	summierte Zeit für 200.000.000 Punkte [Tage]
0.001	2.3
0.01	23.1
0.1	231.5
0.5	1157.4
1	2314.8 (≈ 6.4 Jahre)

Tab. 8.7: Auswirkung der Rechenzeit für einen Laserpunkt auf die Summe der Rechenzeit für 200 Millionen Punkte.

Die einfachste Repräsentation eines lokalen Geländes ist die Konstruktion einer Ebene, die sich durch den Laserpunkt, dessen Geländeparameter gesucht werden, und seine zwei nächstliegenden Nachbarn im dreidimensionalen Raum definiert. Für eine flächendeckende Darstellung des Geländes oder die Interpolation von Geländepunkten eignet sich diese Methode nicht.

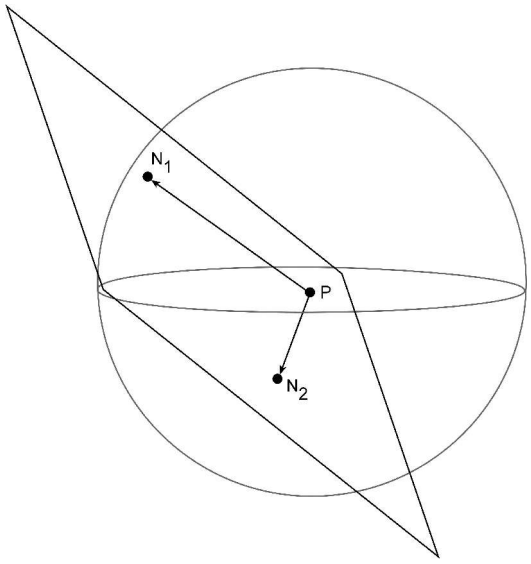


Abb. 8.4: Konstruktion des Geländes an einem Punkt mit Hilfe einer Ebene, die sich durch den Laserpunkt (P) und seine zwei nächsten Nachbarn (N_1, N_2) definiert (eigener Entwurf).

In Abbildung 8.4 sind diese Ebene, die Stützpunkte dieser Ebene und das räumliche Suchkriterium abgebildet. Um den Laserpunkt wird eine Kugel im Raum mit einem definierten Radius bzw. Durchmesser konstruiert. Von allen gefundenen Punkten in dieser Kugel werden die zwei „Nachbarpunkte“ mit der geringsten dreidimensionalen Distanz zum Laserpunkt (vgl. Funktion *range*) für die Konstruktion der Ebene herangezogen. Der Suchradius sollte sich am mittleren Punktabstand der Laserpunkte im Untersuchungsgebiet

orientieren. Je geringer der maximale Suchradius gewählt wird, desto mehr Punkte werden keine zwei Nachbarn in dieser Distanz aufweisen können. Überall wo große Höhenunterschiede auf eine kurze horizontale Distanz treffen (z.B. Häuser, Bäume) oder wo Laserpunkte aufgrund der fehlenden Rückstrahlung des Signals fehlen (z.B. Gewässer), kann es sein, dass in einem kleinen Suchradius keine Nachbarn gefunden werden. Wird der Suchradius vergrößert, ändert sich für die Punkte, die schon zwei Nachbarn hatten, nichts, da immer die zwei nächsten Punkte für die Ebene verwendet werden und es daher egal ist, wenn zusätzlich weiter entfernte Punkte im Suchradius liegen.
In den Abbildungen 8.5 bis 8.8 ist eine klare Verbindung zwischen hoher Vegetation (Nadel- und Laubbäume) und fehlenden Nachbarpunkten bei der Konstruktion der Geländeebene zu erkennen (Abb. 8.6 ist im Appendix D.3 größer dargestellt). Ferner können Gebäudekanten festgestellt werden, an denen keine Nachbarpunkte gefunden werden konnten. Je größer der Suchradius, desto geringer wird die Anzahl der Laserpunkte mit weniger als zwei Nachbarn (vgl. Tab. 8.8).

Abb. 8.5: Infrarot-Luftbild zur Veranschaulichung der hohen Vegetation (© Land Vorarlberg, Landesvermessungsamt, 2001).

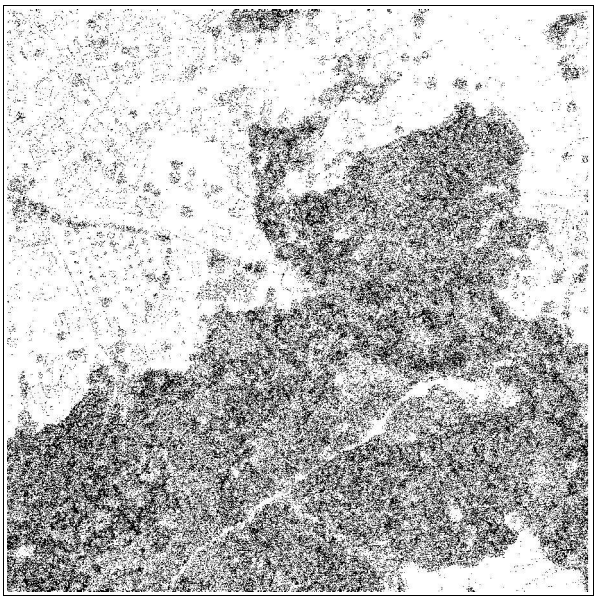


Abb. 8.6: Bereiche (in schwarz), die bei einem 3-D-Suchradius von 0.5 m weniger als 2 Nachbarpunkte haben (eigener Entwurf).

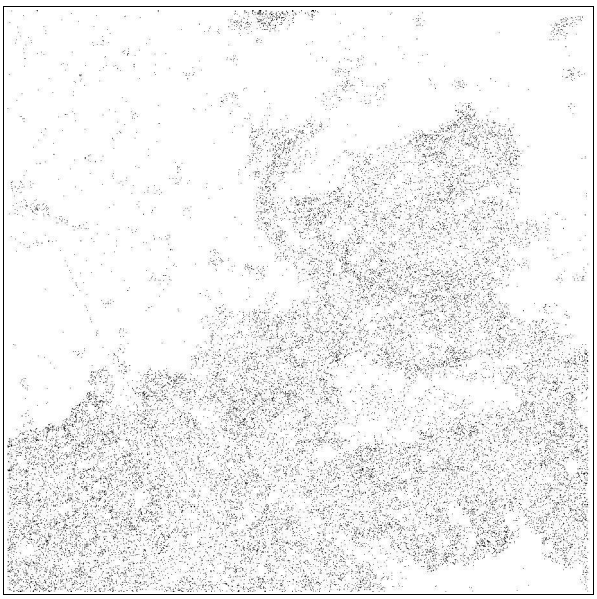


Abb. 8.7: Bereiche (in schwarz), die bei einem 3-D-Suchradius von 1 m weniger als 2 Nachbarpunkte haben (eigener Entwurf).

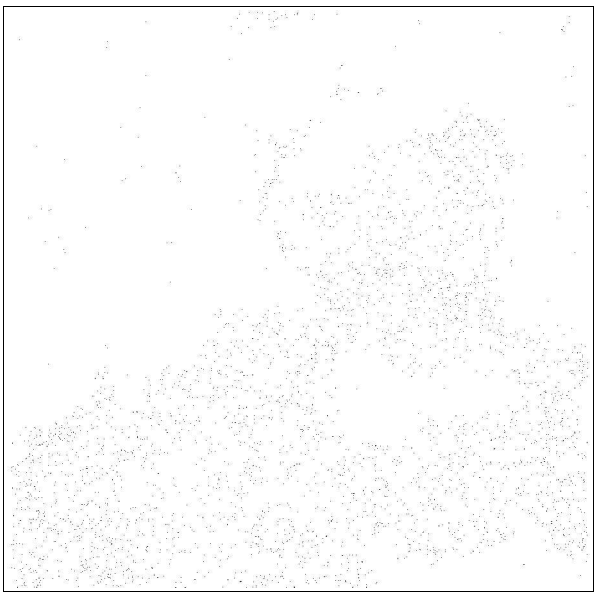


Abb. 8.8: Bereiche (in schwarz), die bei einem 3-D-Suchradius von 2 m weniger als 2 Nachbarpunkte haben (eigener Entwurf).

Bei der Modellierung der Aufnahmegeometrie kommen für die Konstruktion der Geländeebene alle Punkte, d.h. von allen Streifen *first* und *last pulse*, als mögliche Nachbarpunkte in Frage. Es wäre möglich, nur Punkte des gleichen Streifens oder nur Punkte der gleichen Art der Reflexion (erste oder letzte) heranzuziehen. Es macht keinen Sinn diese Einschränkungen zu treffen, da die höchstmögliche Anzahl der Punkte für eine möglichst exakte Darstellung der Erdoberfläche im Modell gegeben sein sollte.
Mathematisch gesehen kann eine Ebene durch drei Punkte beschrieben werden, sofern die Punkte nicht exakt auf einer Geraden liegen. Ausgehend vom Laserpunkt, dessen Geländegeometrie gesucht wird, können zwei Vektoren ($\overrightarrow{PN_1}, \overrightarrow{PN_2}$) angegeben werden (Abb. 8.9). Diese zwei Vektoren spannen die gesuchte Ebene auf. Die weiteren Berechnungen werden der Einfachheit halber mit dem Normalvektor dieser Ebene $\vec{n}$, der sie eindeutig beschreibt, gemacht (Abb. 8.9).

$$\vec{n} = \overrightarrow{PN_1} \times \overrightarrow{PN_2} \tag{8.7}$$

$\vec{n}$ ...Normalvektor der Ebene als Kreuzprodukt aus den beiden Vektoren:

$\overrightarrow{PN_1}, \overrightarrow{PN_2}$ ...Vektoren, die die Ebene aufspannen. Vektor vom Laserpunkt zum Nachbarpunkt 1 bzw. 2.

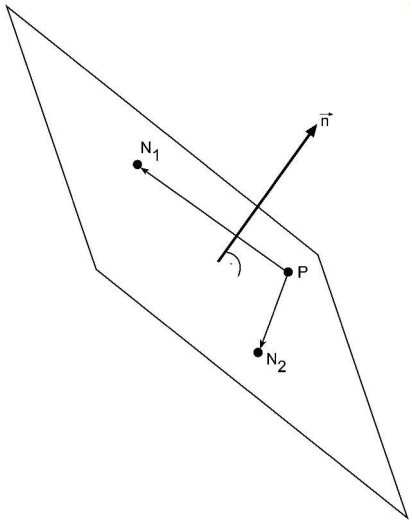


Abb. 8.9: Normalvektor $\vec{n}$ (eigener Entwurf).

Anzahl der Punkte mit weniger als zwei Nachbarn im *Suchradius*			**Anzahl der Punkte, die mit ihren zwei nächsten Nachbarn auf einer Geraden liegen**			**Gesamtanzahl der Punkte im Untersuchungsgebiet**
0.5 m	***1 m***	***2 m***				
731981 (19.28 %)	118395 (3.12 %)	13863 (0.37 %)	117	117	117	3797446 (100 %)

Tab. 8.8: Anzahl der Punkte, die bei der Rekonstruktion der Aufnahmegeometrie „Problemfälle“ darstellen.

Je größer der Suchradius, desto mehr Punkte erhalten mindestens zwei Nachbarn, aber auch desto ungenauer wird die Rekonstruktion des Geländes (vgl. Tab. 8.8). Der Spezialfall, dass alle drei Punkte, die für die Rekonstruktion der Ebene in Frage kommen, auf einer Geraden liegen, kann zum Beispiel dann zustande kommen, wenn ein Punkt neben einem Punkt liegt, dessen erste und letzte Reflexion an der exakt gleichen Stelle stattgefunden hat (*first pulse = last pulse*). Unter „exakt“ wird verstanden, dass die Koordinaten auf Zentimeter gerundet dieselben sind. Bei der Suche nach Nachbarpunkten werden alle Punkte, die eine Distanz von 0 m zum ursprünglichen Punkt haben, ausgeschlossen. Werden zwei Nachbarpunkte mit den gleichen Koordinaten gefunden, wird ein drittnächster Punkt, der ungleiche Koordinaten hat, zur Berechnung verwendet. Werden für einen Laserpunkt nicht mindestens zwei ungleiche Nachbarpunkte gefunden, kann die Berechnung der Aufnahmegeometrie mit dem vorgegebenen Suchradius nicht durchgeführt werden.

8.3.4.2 Neigung (*slope*)

Die Neigung (α) des konstruierten Geländes entspricht dem Winkel zwischen der Horizontalen (Vektor [1,1,0]) und der Ebene. Ebenso gilt, dass die Geländeneigung dem Winkel zwischen dem Normalvektor der Ebene und der Vertikalen (Vektor [0,0,1]) entspricht (Abb. 8.10).

$$\cos\alpha = \frac{\vec{n}\cdot\vec{v}}{|\vec{n}|\cdot|\vec{v}|} \tag{8.8}$$

Die Formel 8.8 (BÜRGER et. al. 1997, 17), die das Winkelmaß zwischen zwei Vektoren angibt, wird für die Berechnung des Winkels zwischen dem Normalvektor $\vec{n}$ der Ebene und dem Vektor $\vec{v}$ herangezogen.

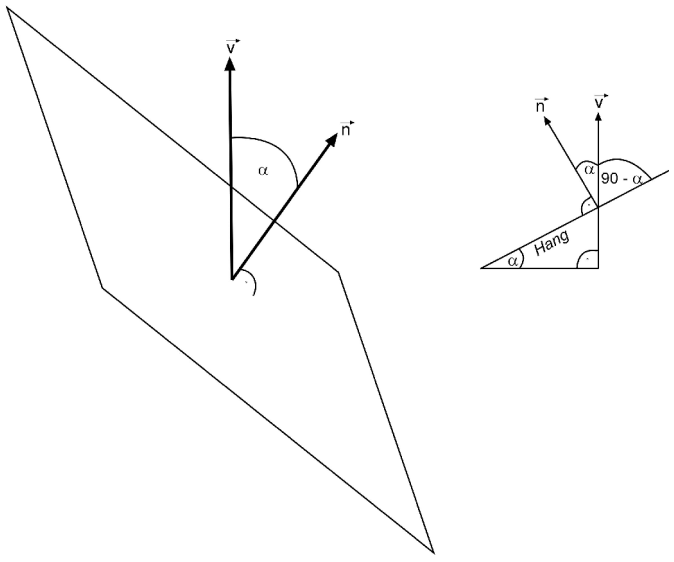


Abb. 8.10: Neigung der Ebene als Winkel zwischen dem Normalvektor und dem Vektor $\vec{v}$ [0, 0, 1], der senkrecht nach oben zeigt. Auf der rechten Seite ist ein Querschnitt durch einen Hang mit der Neigung α dargestellt (eigener Entwurf).

```
slope(ID des Laserpunkts::INTEGER, Position des
Laserpunkts::GEOMETRY, 3D-Suchradius [m])
```

Beispiel:

```
SELECT idscanpoint, slope(idscanpoint, geom_point, 1.0) FROM
tdtascanpoint ORDER BY idscanpoint LIMIT 10;
```

idscanpoint	slope
1	10.07
2	34.61
3	26.91
4	30.94
5	20.72
6	48.77
7	25.60
8	26.06
9	34.67
10	23.29

Tab. 8.9: Berechnung der Hangneigung für ausgewählte Laserpunkte (3D-Suchradius = 1 m).

Die Neigung kann Werte zwischen 0° und 90° annehmen. Die Modellierung von überhängendem Gelände wird ausgeschlossen. Eine Neigung von 90° würde einer senkrechten Wand entsprechen.

Neigung am Laserpunkt			
Anzahl der Datensätze	**Rechenzeit in ms**		
	Suchradius		
	0.5 m	***1 m***	***2 m***
10	31.941	449.708	144.662
100	273.839	1550.733	708.856
1000	2042.035	4828.579	7317.115
10000	20700.527	50681.387	79035.854
100000	209326.972	416571.186	643269.416
1000000	3278591.558	4578679.788	8203998.432

Tab. 8.10: Rechenzeiten für die Berechnung der Hangneigung am Laserpunkt

Die Rechenzeiten (Tab. 8.10) sind proportional zum Suchradius. Dies hängt damit zusammen, dass je größer die räumliche Ausdehnung ist, desto mehr Punkte werden in dieser Umgebung gefunden und desto mehr Abstandsmessungen müssen vorgenommen werden. Es gilt auch, je mehr Punkte im Suchradius liegen, desto mehr Punkte müssen nach der Distanz sortiert werden und desto länger wird im Endeffekt die Rechenzeit.

8.3.4.3 Exposition (*aspect*)

Die Exposition der Ebene (ω) – d.h. in welche Himmelsrichtung der Normalvektor der Ebene zeigt – wird folgendermaßen codiert:

Exposition	**in °**
Nord	0
Ost	90
Süd	180
West	270
flaches Gelände (α=0°)	-1

Tab. 8.11: Beispiel für die Exposition in Grad für die vier Haupthimmelsrichtungen und für ein Gelände mit 0° Neigung.

In Abbildung 8.11 ist auf der linken Seite ein Beispiel einer Ebene mit Exposition ca. SE (Südost) in Schrägansicht dargestellt. Es ist zu erkennen, dass für die Bestimmung der Exposition wiederum der Normalvektor verwendet wird.

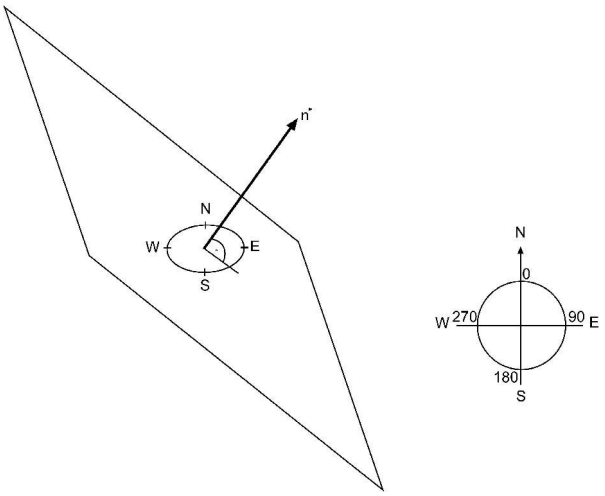
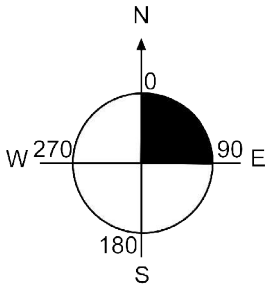
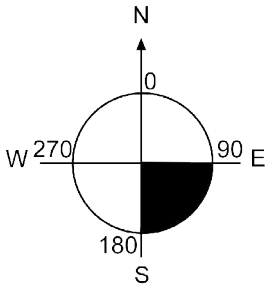
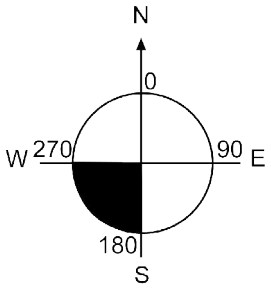
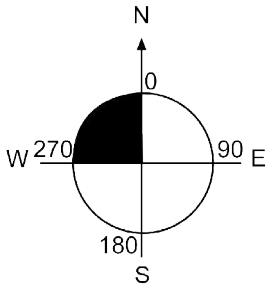
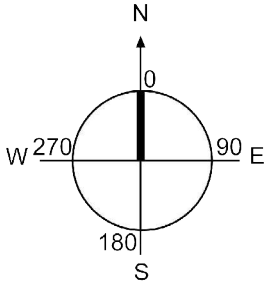
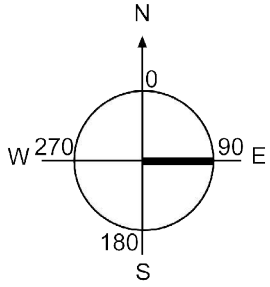
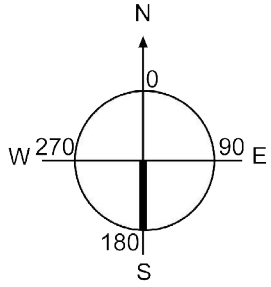
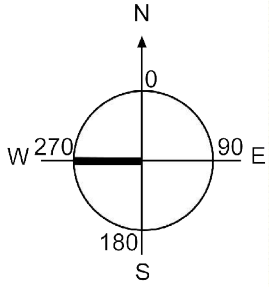
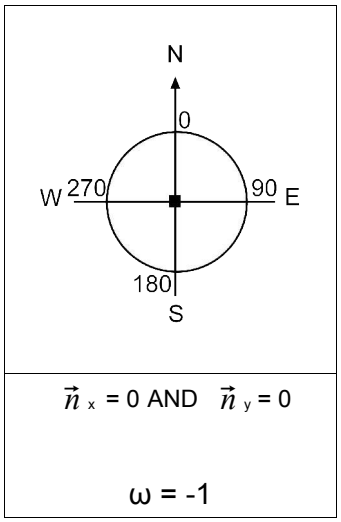


Abb. 8.11: Exposition der Ebene als horizontale Ausrichtung des Normalvektors. Auf der rechten Seite ist die Gradcodierung der Himmelsrichtungen dargestellt. $\vec{n}_{x,y,z}$...Koordinaten des Normalvektors (eigener Entwurf).

			
$\vec{n}_x > 0$ AND $\vec{n}_y > 0$	$\vec{n}_x > 0$ AND $\vec{n}_y < 0$	$\vec{n}_x < 0$ AND $\vec{n}_y < 0$	$\vec{n}_x > 0$ AND $\vec{n}_y > 0$
$\omega = 90 - atan(\frac{\vec{n}_y}{\vec{n}_x})$	$\omega = 90 + atan(\frac{\lvert\vec{n}_y\rvert}{\vec{n}_x})$	$\omega = 180 + (90 - atan(\frac{\vec{n}_y}{\vec{n}_x}))$	$\omega = 270 + atan\frac{\vec{n}_y}{(\lvert\vec{n}_x\rvert)}$
			
$\vec{n}_x = 0$ AND $\vec{n}_y > 0$	$\vec{n}_x > 0$ AND $\vec{n}_y = 0$	$\vec{n}_x = 0$ AND $\vec{n}_y < 0$	$\vec{n}_x < 0$ AND $\vec{n}_y = 0$
ω = 0	ω = 90	ω = 180	ω = 270

Tab. 8.12: Berechnungsschema für die Exposition der Ebene. Die Bedingungen sind mit einem logischen UND (AND), d.h. beide Bedingungen müssen erfüllt sein, verknüpft (eigener Entwurf).

Bevor die Bedingungen in Tabelle 8.12 abgefragt werden, muss überprüft werden, ob es sich um ein flaches Gelände mit der Neigung 0° handelt, da für diesen Spezialfall keine Exposition berechnet werden kann. Der Normalvektor zeigt senkrecht nach oben (Tab. 8.13). In Anlehnung an die Codierung von ESRI (INTERNET: ESRI) wird hierfür die Exposition -1 vergeben.



Tab. 8.13: Gelände mit Neigung 0°. Der Normalvektor [0,0,1] zeigt senkrecht nach oben (eigener Entwurf).

```
aspect(ID des Laserpunkts::INTEGER, Position des
Laserpunkts::GEOMETRY, 3D-Suchradius [m])
```

Beispiel:

```
SELECT idscanpoint, aspect(idscanpoint, geom_point, 1.0) FROM
tdtascanpoint ORDER BY idscanpoint LIMIT 10;
```

idscanpoint	aspect
1	173.66
2	87.48
3	146.98
4	92.70
5	105.95
6	218.08
7	116.57
8	81.96
9	121.65
10	95.44

Tab. 8.14: Berechnung der Exposition für ausgewählte Laserpunkte (3D-Suchradius = 1 m).

Exposition	
Anzahl der Datensätze	**Rechenzeit in ms**
	Suchradius 0.5 m
10	37.812
100	258.780
1000	1967.376
10000	20218.720
100000	193013.489
1000000	2812333.983

Tab. 8.15: Rechenzeiten für die Berechnung der Exposition des Laserpunkts.

8.3.4.4 Einfallswinkel (*angle of incidence*)

Der Einfallswinkel (φ) ist in der Physik als Winkel zwischen dem einfallenden Strahl und der Normalen (dem Lot an der Einfallsstelle) definiert (TIPLER & MOSCA 2004, 1007). Der einfallende Strahl entspricht dem Laservektor $\vec{l}$ und die Normale dem Normalvektor $\vec{n}$ der Ebene (Abb. 8.12). Im dreidimensionalen Raum ist dies der kleinste Winkel zwischen diesen beiden Vektoren $\vec{l}, \vec{n}$ und wird analog der Formel 8.8 berechnet (LUTZ 2003, 32).

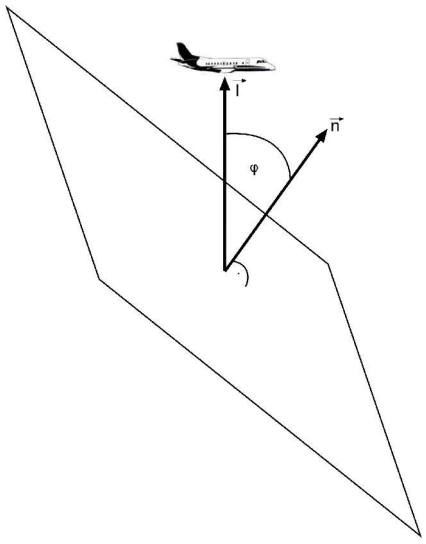


Abb. 8.12: Einfallswinkel (φ) des Laservektors $\vec{l}$. $\vec{n}$ ...Normalvektor der Ebene (eigener Entwurf).

```
incidence(ID des Laserpunkts::INTEGER, Position des
Laserpunkts::GEOMETRY, Flugzeugposition::GEOMETRY, 3D-Suchradius
[m])
```

Beispiel:

```
SELECT idscanpoint, incidence(idscanpoint, geom_point,
geom_planepos, 1.0) FROM tdtascanpoint ORDER BY idscanpoint LIMIT
10;
```

idscanpoint	incidence
1	23.57
2	54.70
3	26.16
4	49.78
5	38.80
6	20.17
7	37.94
8	49.44
9	41.04
10	43.71

Tab. 8.16: Berechnung des Einfallswinkels für ausgewählte Laserpunkte (3D-Suchradius = 1 m).

Ein Einfallswinkel von 0° kann bedeuten, dass der Scanwinkel des Aufnahmesystems 0° und die Neigung des Geländes 0° sind. Ist die Geländeebene zum Laservektor mit dem gleichen Winkel wie der aktuelle Scanwinkel geneigt, erhält man ebenfalls einen Einfallswinkel von 0°. Alle Geländeebenen, die einen Einfallswinkel < 90° haben, werden von vorne vom Laserstrahl getroffen. Für Geländeebenen, die vom Aufnahmesystem weg geneigt sind (φ >= 90°), d.h. sie liegen im Sichtschatten des Aufnahmesystems, kann in weiterer Folge keine Fläche des Lichtkegels am Boden (*footprint*) berechnet werden (vgl. Abb. 8.13).

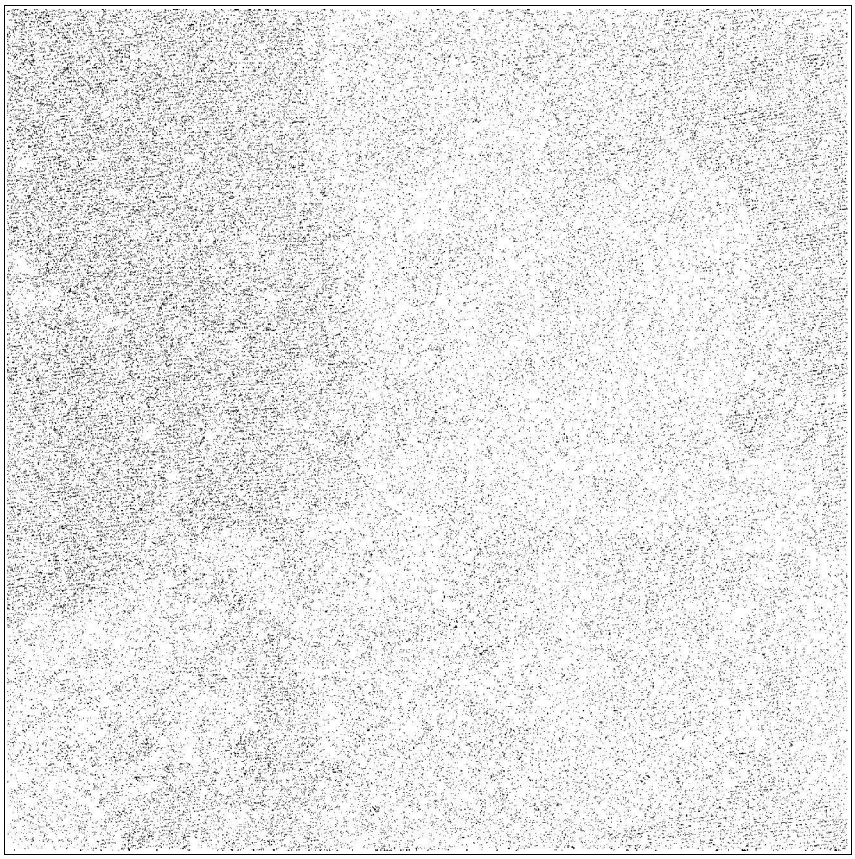


Abb. 8.13: Schattenbereiche (Einfallswinkel >= 90°) im Untersuchungsgebiet bei einem 2D-Suchradius von 0.5 m (eigener Entwurf).

Die Schattenbereiche werden stark reduziert, wenn für einen Laserpunkt nur Nachbarpunkte desselben Flugstreifens in Frage kommen. Dafür sinkt die Punktdichte, v.a. in Streifenüberlappungsbereichen, und somit auch die Anzahl der Punkte, die bei einem kleinen Suchradius (z.B. 0.5 m) gefunden werden.

Einfallswinkel	
Anzahl der Datensätze	**Rechenzeit in ms**
	Suchradius 0.5 m
10	89.902
100	282.401
1000	2345.720
10000	22772.352
100000	231623.198
1000000	3359752.127

Tab. 8.17: Rechenzeiten für die Berechnung des Einfallswinkels des Laservektors an der Geländeebene.

8.3.4.5 Fläche des Lichtkegels am Boden (*footprint*)

Die Fläche des Lichtkegels am Boden (*footprint*) ist eine Funktion der Laserstrahldivergenz (γ), der Länge des Laservektors (*range*) und dem Einfallswinkel (φ) (Abb. 8.14). Die Strahldivergenz ist der Öffnungswinkel des Laserstrahls. Dieser Öffnungswinkel hängt vom gewählten Aufnahmesystem ab. Es gibt Aufnahmesysteme, die mehrere Winkeleinstellungen zulassen. Die Befliegungen in Vorarlberg wurden mit einer Strahldivergenz von 0.25 mrad durchgeführt, d.h. dieser Wert ist für alle Laserpunkte konstant.

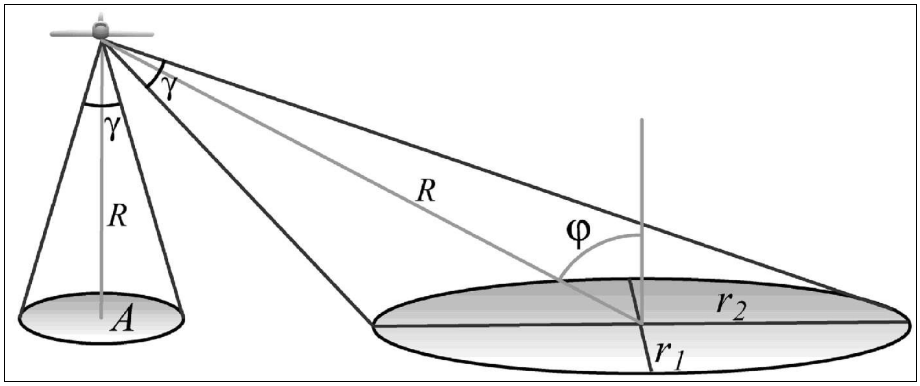


Abb. 8.14: Der *footprint* wird durch *range* R, Einfallswinkel φ und Laserstrahldivergenz γ festgelegt (LUTZ 2003, 33).

WEHR & LOHR (1999, 75) zeigen wie der *footprint* auf einer relieflosen Oberfläche (ohne Objekte und mit 0° Neigung) als eine Funktion von Scanwinkel, Flughöhe und Strahldivergenz berechnet werden kann. Die Oberflächenstruktur, die die Größe des Streuquerschnitts beeinflusst, kann bei der Modellierung der Fläche des Lichtkegels am Boden nicht berücksichtigt werden. Es wird angenommen, dass der Lichtkegel des Laserstrahls auf eine strukturlose Fläche trifft (vgl. Abb. 8.15).

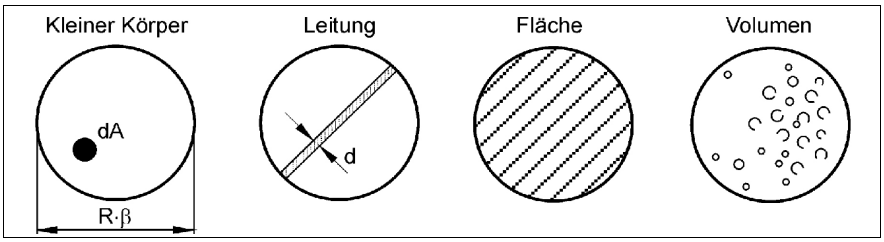


Abb. 8.15: Streuung an verschiedenen Objekten (WAGNER et. al. 2003, 231).

Der *footprint* ist kreisförmig, wenn der Einfallswinkel gleich 0° beträgt. Je größer der Einfallswinkel ist, desto mehr wird der Lichtkegel in Form einer Ellipse projiziert. LUTZ (2003, 33f.) wählte eine Annäherung dieser Projektion der Kreisfläche auf die Ebene, die er mit einer Ellipse beschreibt (Abb. 8.16).

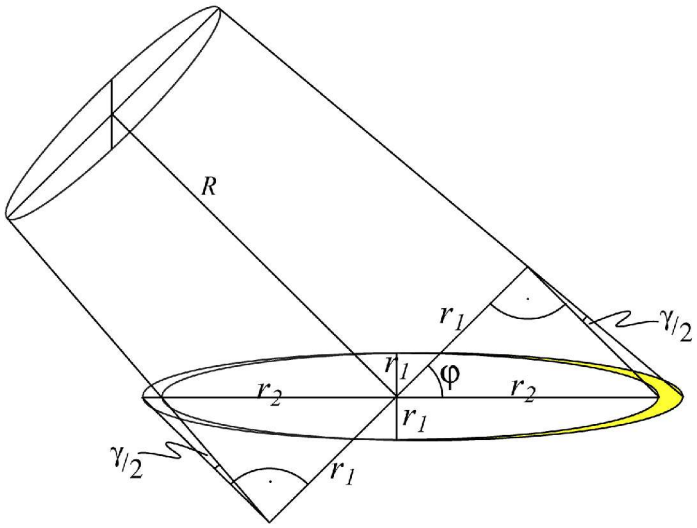


Abb. 8.16: Die Fläche des Lichtkegels am Boden bei einem Einfallswinkel größer als 0° (LUTZ 2003, 34).

Die kleine Halbachse (r_1) der Ellipse in Abbildung 8.16 ist eine Funktion von *range* (R), Strahldivergenz (γ) und Laserapertur (Austrittsöffnung des Laserstrahls). Da die Größe der Laserapertur im Vergleich zum *range* sehr gering ist, wird sie in Formel 8.9 vernachlässigt.

$$r_1 = R \cdot \tan\left(\frac{\gamma}{2}\right) \tag{8.9}$$

Unter der Annahme, dass der *footprint* ellipsenförmig ist, kann die große Halbachse (r_2) aus r_1 und dem Einfallswinkel (φ) berechnet werden.

$$r_2 = \frac{r_1}{\cos\varphi} \tag{8.10}$$

Die Formel 8.10 vernachlässigt die Strahldivergenz (γ) ab dem Zeitpunkt, an dem das Zentrum des Laserstrahls die Oberfläche erreicht. Die Fläche, die an der entfernteren Seite der Ellipse verloren geht, kommt auf der flugzeugnäheren Seite dazu (Abb. 8.16). Bei einer Strahldivergenz von 0.25 mrad sind die dadurch verursachten Längenunterschiede des Radius nur sehr gering (kleiner als 1 mm; LUTZ 2003, 34).

Die Fläche des Lichtkegels am Boden entspricht somit der Fläche der Ellipse.

$$A = \pi \cdot r_1 \cdot r_2 \tag{8.11}$$

```
footprint(ID des Laserpunkts::INTEGER, Position des
Laserpunkts::GEOMETRY, Flugzeugposition::GEOMETRY, 3D-Suchradius
[m])
```

Beispiel:

```
SELECT idscanpoint, footprint(idscanpoint, geom_point,
geom_planepos, 1.0) FROM tdtascanpoint ORDER BY idscanpoint LIMIT
10;
```

idscanpoint	footprint
1	0.021
2	0.034
3	0.022
4	0.030
5	0.025
6	0.021
7	0.024
8	0.030
9	0.026
10	0.027

Tab. 8.18: Berechnung der Fläche des Lichtkegels am Boden (*footprint*) in m² für ausgewählte Laserpunkte (3D-Suchradius = 1 m).

Fläche des Lichtkegels am Boden	
Anzahl der Datensätze	Rechenzeit in ms
	Suchradius 0.5 m
10	33.482
100	288.765
1000	2219.085
10000	25053.745
100000	287233.787
1000000	3374953.784

Tab. 8.19: Rechenzeiten für die Berechnung der Fläche des Lichtkegels am Boden.

Bei sehr flachen Einfallswinkeln ($\varphi \approx 90°$) kommt es zu so genannten *„schleifenden Winkeln“*. Die Fläche des *footprints* nimmt dementsprechend zu. In der Realität sind Signale, die einen *footprint* größer als 5 m² haben, zu schwach und können vom Aufnahmesystem nicht mehr registriert werden. Solche extremen Ausreißer müssen als Artefakte der Geländemodellierung angesehen werden (vgl. LUTZ 2003, 34).

8.3.4.6 Relative Flughöhe (*height above ground level*)

Die Modellierung der relativen Flughöhe – die Höhendifferenz zwischen Flugzeug und Oberfläche – ist gesondert von den übrigen Geländeparametern zu betrachten. Dies hängt damit zusammen, dass ein anderer Suchradius (2D) bei der Modellierung der relativen Flughöhe verwendet wird und damit ein Zusammenführen der Berechnung der relativen Flughöhe in dieselbe Funktion mit den anderen Parametern keinen Sinn macht.
Mit einem zweidimensionalen kreisförmigen Suchradius werden alle Laserpunkte, die direkt unterhalb des Flugzeugs liegen (Scanwinkel ca. 0°), ausgewählt (Abb. 8.17). Der Abstand der Laserpunkte zur Flugzeugposition wird im zweidimensionalen Raum (x,y) bestimmt.
Die minimale Höhe aller ausgewählten Laserpunkte wird für die Berechnung der Höhendifferenz herangezogen. Damit soll gewährleistet sein, dass der Abstand Flugzeug zu Gelände und nicht zu einem Objekt auf Oberfläche angegeben wird.

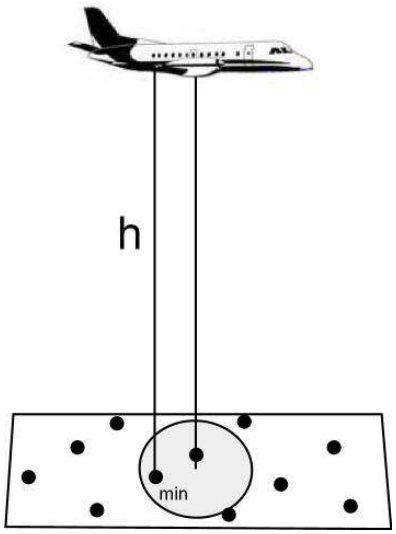


Abb. 8.17: Selektion der Laserpunkte für die Berechnung der relativen Flughöhe mit einem zweidimensionalen Suchradius (eigener Entwurf).

```
height_agl(ID des Laserpunkts::INTEGER, Flugzeugposition::GEOMETRY,
2D-Suchradius [m])
```

Beispiel:

```
SELECT idscanpoint, height_agl(idscanpoint, geom_planepos, 1.0) FROM
tdtascanpoint ORDER BY idscanpoint LIMIT 10;
```

idscanpoint	height_agl
8196223	1853.55
8196224	1853.55
8196225	1853.55
8197013	1854.44
8197014	1854.44
8197015	1854.44
8197016	1854.44
8197017	1854.44
8197433	1854.23
8197434	1854.23

Tab. 8.20: Berechnung der relativen Flughöhe für ausgewählte Laserpunkte (2D-Suchradius = 1 m).

Relative Flughöhe	
Anzahl der Datensätze	Rechenzeit in ms
	Suchradius 0.5 m
10	30.495
100	77.568
1000	302.727
10000	3225.849
100000	93770.510
1000000	1855684.467

Tab. 8.21: Rechenzeiten für die Berechnung der relativen Flughöhe (2-D-Suchradius = 2 m).

8.3.4.7 Zusammenfassung

Der Rechenaufwand für die Modellierung der einzelnen Geländeparameter (Neigung, Exposition, Einfallswinkel, Fläche des Lichtkegels am Boden) nimmt, wenn man sie jeweils einzeln berechnet, viel Zeit in Anspruch. Dies kann dadurch begründet werden, dass die Suche der Nachbarpunkte im Suchradius, die den Hauptteil der Zeit erfordert, immer wieder neu durchgeführt werden muss. Um dies zu vermeiden, sind alle Berechnungsschritte in einer Funktion zusammengefasst, die als Ergebnis alle Geländeparameter eines Laserpunktes zurückgibt. Die Rechenzeit der Funktion, die alle Parameter hintereinander berechnet (Tab. 8.22), ist wesentlich kürzer als die Summe der Rechenzeiten der einzelnen Funktionen.

Gesamte Scangeometrie			
Anzahl der Datensätze	**Rechenzeit in ms**		
	Suchradius		
	0.5 m	***1 m***	***2 m***
10	29.397	32.768	53.621
100	319.331	388.570	339.707
1000	3427.446	4895.869	3438.363
10000	35549.331	49084.762	35784.330
100000	312578.208	324521.882	311006.254
1000000	4727169.369	4713991.060	4764554.491

Tab. 8.22: Rechenzeiten für die Berechnung von Neigung, Exposition, Einfallswinkel und Fläche des Lichtkegels am Boden in einer Funktion.

8.4 Statistische Auswertung

8.4.1 Einleitung

Die große Anzahl der Datensätze (v.a. Flugzeugpositionen und Laserpunkte) macht es notwendig, Methoden anzuwenden, die eine verständliche und objektive Beschreibung der Daten erlauben. Mit Hilfe der deskriptiven Statistik können die Attribute der Entitäten (z.B. Intensität, absolute Höhe, usw.) näher beschrieben werden. Die daraus abgeleiteten statistischen Kennwerte können zum Vergleich zwischen Datensätzen (oder räumlichen Bereichen) verwendet werden. Ein Vergleich von Datensätzen kann nicht nur über die gewonnenen statistischen Kennwerte geschehen, sondern mit direkten statistischen Methoden (Zusammenhangsmaße) durchgeführt werden. Die für die Interpretation wohl wichtigste Methode ist die Visualisierung der Variablen in Diagrammen.
Die Datenbank besitzt in ihrer Grundinstallation so genannte *Aggregatfunktionen*, die aus einer Sammlung von Eingabewerten ein einzelnes Ergebnis berechnen, wie zum Beispiel die Summe der Werte einer Spalte (EISENTRAUT 2003, 170f.). Für die Beschreibung der Daten reicht die Funktionalität der Datenbank aus. Die Zusammenhangsstatistik wird erst durch den Funktionalitätstransfer der Statistiksoftware R in die Datenbank möglich (vgl. „*5.4.6.3 PL/R*“). Die Statistiksoftware R weist ein reichhaltiges Angebot an Diagrammen und Graphiken auf, die ebenfalls auf die Daten in der Datenbank angewandt werden können (DOLIĆ 2004, 75ff.).

8.4.2 Deskriptive Statistik

Die Aggregatfunktionen von PostgreSQL (Tab. 8.23) erlauben die direkte Berechnung von statistischen Kennwerten. Die Eingabewerte, für die ein Kennwert berechnet werden soll, können mit *SQL-Statements* ausgewählt werden. Dadurch können Eingabewerte durch bestimmte Abfragekriterien vom gesamten Datensatz extrahiert werden und nur die

ausgewählten Datensätze fließen als Eingabewerte in die Funktion ein.

Funktion	Beschreibung
avg()	Durchschnitt (arithmetisches Mittel) aller Eingabewerte
count()	Anzahl der Eingabewerte
max()	Maximalwert aller Eingabewerte
min()	Minimalwert aller Eingabewerte
stddev()	Standardabweichung der Eingabewerte
sum()	Summe der Eingabewerte
variance()	Varianz der Eingabewerte (Quadrat der Standardabweichung)

Tab. 8.23: Aggregatfunktionen in PostgreSQL (EISENTRAUT 2003, 170f.).

Beispiel:

```
SELECT avg(intens), min(intens), max(intens) FROM tdtascanpoint
WHERE idstrip=1;
```

Es wird das arithmetische Mittel, der Minimalwert und der Maximalwert der Intensitäten aller Datensätze, die die Flugstreifen-ID=1 haben, berechnet.
Gleich wie im oben angeführten Beispiel können die Datensätze über das Attribut Zeit oder die räumlichen Koordinaten selektiert werden.

Beispiel:

```
SELECT avg(intens), min(intens), max(intens) FROM tdtascanpoint
WHERE abs_time(time, idstrip) BETWEEN '01.01.2004' AND '01.01.2005';
```

Die Intensitätswerte aller Datensätze die in einem bestimmten Zeitintervall (absolut zeitlich verortet) liegen, werden herangezogen.

Beispiel:

```
SELECT avg(intens), min(intens), max(intens) FROM tdtascanpoint
WHERE EXPAND(GEOMETRYFROMTEXT('POINT(32564342.77 5237585.84
2007.3)', -1), 10) && geom_point;
```

Die Datensätze, die in die Berechnung der Kennwerte einfließen, werden räumlich selektiert. Es wird ein Quadrat mit 10 m Kantenlänge um den gegebenen Punkt gelegt und alle Punkte selektiert, die in diesem Quadrat liegen.

Die Funktionen, die standardmäßig in der Datenbank zur Verfügung stehen, werden durch

die Funktionalität von R ergänzt. Um die Funktionen von R nützen zu können, müssen eigene Aggregatfunktionen in der Sprache PL/R geschrieben werden. Die internen Aggregatfunktionen von PostgreSQL sind wesentlich schneller als die in PL/R geschriebenen Funktionen. Die Berechnung der Standardabweichung ist in einer eigenständigen PL/R-Funktion ca. dreimal so langsam wie die interne Funktion von PostgreSQL. Eine Funktion für die Berechnung der Standardabweichung, die bereits als Aggregatfunktion in PostgreSQL vorliegt, wurde zum Vergleich der Rechenzeit zusätzlich in PL/R umgesetzt (Appendix B.2). Die Berechnung des Medians erfolgt ebenfalls in PL/R, da PostgreSQL keine derartige Funktion besitzt (Appendix B.2).
Die Bildung von Klassen und die Beschreibung dieser Klassen (z.B. relative Häufigkeit) können durch die Formulierung von SQL-Statements durchgeführt werden.

Beispiel:

```
SELECT avg(intens) FROM tdtascanpoint GROUP BY idstrip;
```

Im oben angeführten Beispiel werden die durchschnittlichen Intensitätswerte für jeden Flugstreifen berechnet.

Die Tabelle 8.24 zeigt die statistischen Kennwerte des Intensitätssignals aller Laserpunkte im Untersuchungsgebiet. Es ist deutlich zu erkennen, dass nur Werte von 5 bis 255 vorkommen. Die Intensitätswerte werden von der Befliegungsfirma auf 8-Bit (0-255) umgerechnet. Alle Werte unter 5 liefern ein zu unsicheres Signal, das nicht für die Auswertung herangezogen wird.

n	arithm. Mittel	Summe	Minimal-wert	Maximal-wert	Median	Standard-abweichung	Varianz
3797446	56.26	213625415	5	255	40	46.51	2163.09

Tab. 8.24: Deskriptive statistische Werte der Intensität berechnet mit PostgreSQL-Funktionen (der Median wird mit einer PL/R-Funktion berechnet).

Die visuelle Interpretation eines Datensatzes kann sehr viel Aufschluss über die Charakteristik einer Variablen geben. Im Informationssystem werden die Plot-Funktionen von R in Form von PL/R-Funktionen genützt. DOLIĆ (2004, 75ff.) gibt eine Einführung in alle Plot-Varianten der Statistiksoftware R. Für das Informationssystem wurde eine Funktion entwickelt, die es erlaubt, mehrere Variablen gegeneinander in einem Diagramm aufzutragen. Die Funktion *r_plot_jpeg* erhält als Argument ein SQL-Statement, das die Variablen aus den Tabellen selektiert, die dann zur Graphikerstellung herangezogen werden. Das Ergebnis dieser Funktion ist ein Diagramm als JPEG-Graphik. Das Ausgabeformat dieser Funktion kann jederzeit geändert werden (z.B. PDF, PNG, usw.). Das PDF-Format eignet sich nur für geringe Datenmengen, da jeder Punkt als Vektor abgespeichert wird.

```
r_plot_jpeg(SQL-Statement::TEXT)
```

Beispiel:

```
SELECT r_plot_jpeg('select range(geom_point, geom_planepos), intens
from tdtascanpoint');
```

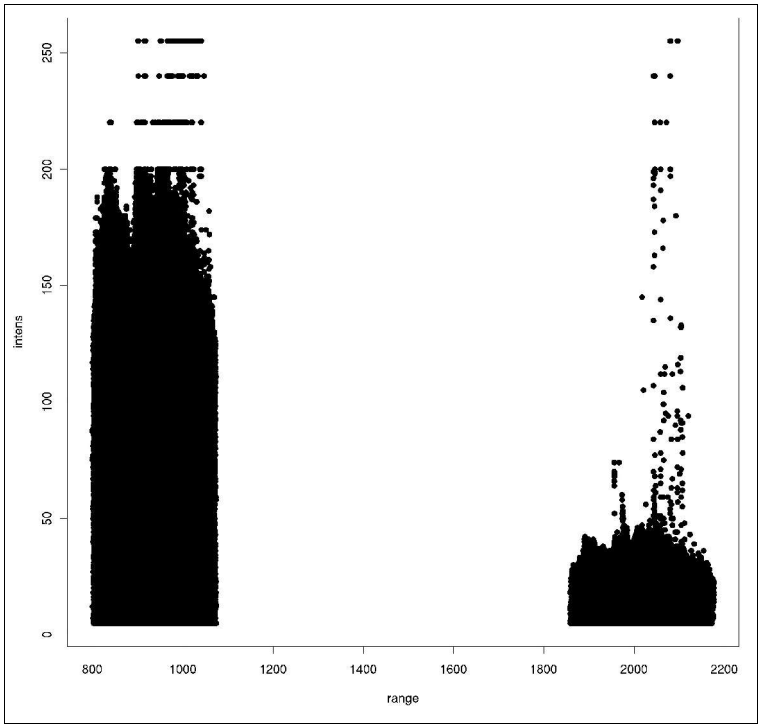


Abb. 8.18: Plot der Intensität (*y*) und der Länge des Laservektors (*x*) für alle Punkte des Untersuchungsgebietes (eigener Entwurf).

In Abbildung 8.18 ist eine Zweiteilung der Länge des Laservektors (*range*) zu erkennen. Dies begründet sich durch die zwei Befliegungen des Untersuchungsgebietes, die in einer unterschiedlichen Höhe geflogen wurden. Die Intensitätswerte der Laserpunkte mit einem kürzeren *range* erreichen durchschnittlich höhere Werte.

Die Funktionen *r_hist_rel* und *r_hist_abs* erstellen ein Histogramm einer Variablen, in relativen bzw. absoluten Häufigkeiten (vgl. Abb. 8.19). Die Histogramme werden im PDF-Format ausgegeben. Der gesamte Wertebereich der Variablen wird in mehrere Abschnitte (Intervalle oder Klassen) unterteilt. Die Klassengrenzen können beim Aufruf der Funktion angegeben werden (DOLIĆ 2004, 96ff.). Das Histogramm gibt Auskunft über die Streuung

(z.B. Schiefe) der Variablen.

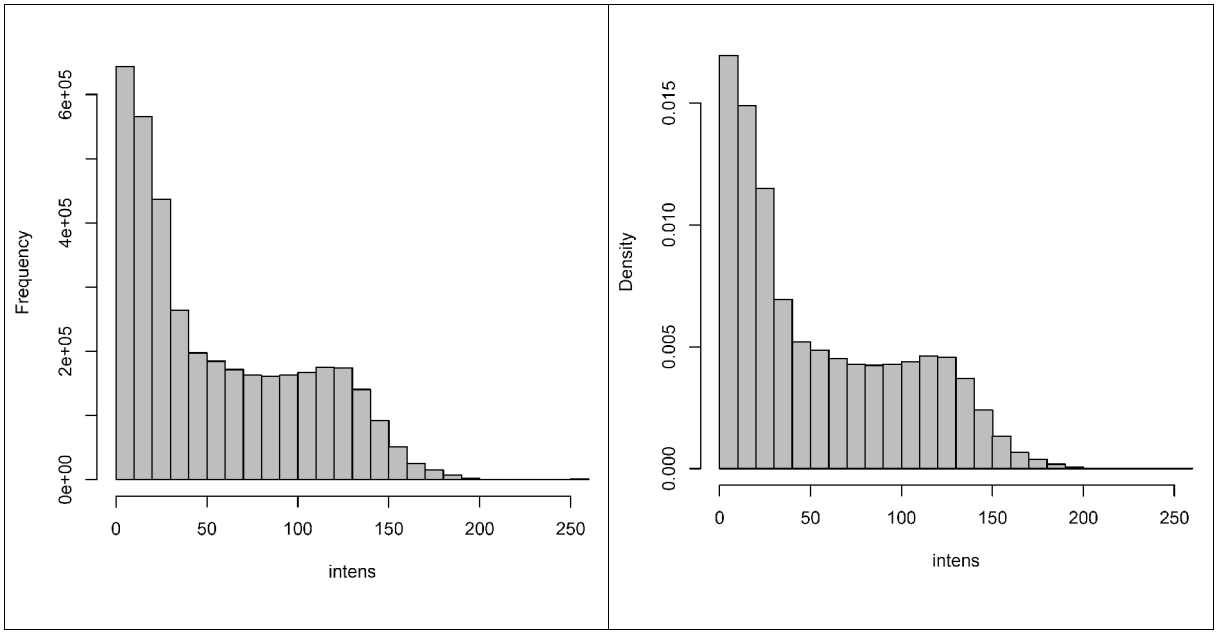


Abb. 8.19: Histogramm der Intensität aller Laserpunkte im Untersuchungsgebiet (links: absolute Häufigkeiten, rechts: relative Häufigkeiten) (eigener Entwurf).

Die Verteilung des Intensitätssignals weist eine positive Schiefe auf (Abb. 8.19). Niedrige Intensitätswerte sind viel häufiger als hohe Werte. Eine weitere Möglichkeit der Visualisierung wird mit der Funktion *r_boxplot* umgesetzt. Das so genannte Box-and-Whiskers-Plot (kurz: Boxplot) eignet sich besonders um Verteilungen im Vergleich darzustellen (vgl. Abb. 8.20; DOLIĆ 2004, 100).

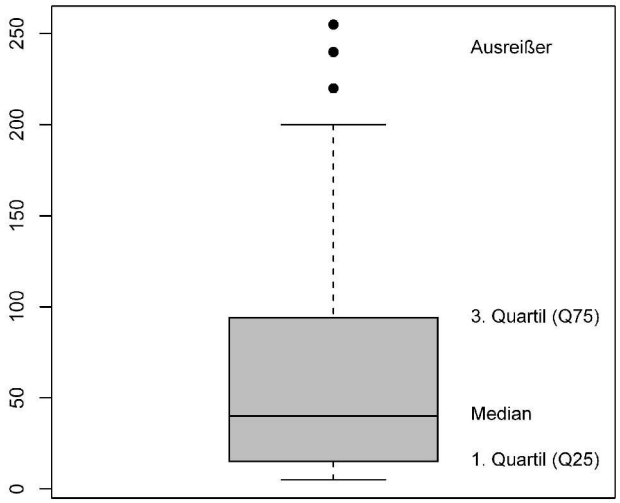


Abb. 8.20: Boxplot der Intensität aller Punkte im Untersuchungsgebiet. Datenpunkte, die um das 1.5-fache des Interquartilabstandes (Q75-Q25) über die Box reichen, werden als einzelne Punkte dargestellt (eigener Entwurf).

8.4.3 Zusammenhangsmaße

8.4.3.1 Korrelationskoeffizient

Der Korrelationskoeffizient misst die Stärke des linearen Zusammenhangs zwischen den Ausprägungen der Variablen *x* und den Ausprägungen der Variable *y*. Bei der Korrelationsanalyse wird kein Unterschied zwischen einer abhängigen und einer unabhängigen bzw. einer Ausgangs- und einer Zielgröße getroffen. Bei der Berechnung kann die Reihenfolge der Eingangsvariablen beliebig erfolgen. Ein Punkt im Korrelationsdiagramm hat einen Wert für *x* und einen Wert für *y*. Der Korrelationskoeffizient, der Werte zwischen -1 und +1 annehmen kann, hat folgende Bedeutung (zitiert nach DOLIĆ 2004, 203):

- Ein negativer Wert in der Nähe von -1 bedeutet eine starke negative Beziehung zwischen den Variablen. Wird also die Variable *x* größer, wird gleichzeitig die Variable *y* kleiner.
- Ein Wert um 0 bedeutet, dass keine lineare Beziehung zwischen den beiden Variablen besteht. Wenn die Werte der Variablen *x* also größer werden, können die Werte der Variablen *y* mal größer mal kleiner werden oder auch gleich bleiben.
- Ein positiver Wert in der Nähe von +1 bedeutet eine starke positive Korrelation. Wenn die Variable *x* größer wird, wird die Variable *y* mit hoher Wahrscheinlichkeit auch größer.

Eine hohe Korrelation sagt nichts über den Kausalzusammenhang zwischen den beiden Variablen aus. Eine niedrige Korrelation kann auch dann zustande kommen, wenn zwei Variablen in einer logischen Beziehung zu einander stehen, aber eine der beiden Variablen stark von anderen Einflüssen abhängt und somit die Werte sehr stark verändert werden. Ein Beispiel ist die Korrelation zwischen Signalintensität und Länge des Laservektors. Es besteht ein eindeutiger physikalischer Zusammenhang zwischen den beiden Variablen. Die Länge des Laservektors ist nur *ein* beeinflussender Faktor der Intensität. Würden alle Laserpunkte unter gleichen Bedingungen (Einfallswinkel, Geländeneigung, Oberflächenbeschaffenheit, usw.) aufgenommen werden, müsste eine sehr starke negative Korrelation das Ergebnis sein. Im Intensitätssignal stecken zusätzlich Informationen über die Reflexionseigenschaften der Oberfläche, den tatsächlichen Streuquerschnitt, usw.. Zwischen Intensität und *range* berechnet sich, unter Einbeziehung aller Punkte des Untersuchungsgebietes, ein Korrelationskoeffizient von r = -0.42. Dieser Wert zeigt den negativen Zusammenhang zwischen den beiden Variablen recht deutlich. Der Betrag des Korrelationskoeffizienten ist jedoch zu niedrig, als dass man eine alleinige Abhängigkeit der Intensität vom *range* ableiten könnte.

Die Korrelationsanalyse ist im Informationssystem mit Hilfe der Funktionalität von R in der Datenbank umgesetzt (→PL/R). R verwendet als Standardkorrelationskoeffizienten die

Produkt-Moment-Korrelation von Pearson (DOLIĆ 2004, 205). Die PostgreSQL-Funktion *random()* kann in Kombination mit LIMIT herangezogen werden, um Zufallsstichproben zu nehmen. Wird die Tabelle mit *random* sortiert (ORDER BY RANDOM) und danach auf eine bestimmte Anzahl von Datensätzen (die Zählung beginnt mit Anfang der Tabelle) beschränkt (LIMIT), kann eine beliebige Anzahl von zufälligen Datensätzen ausgewählt werden.

Beispiel:

```
SELECT idscanpoint FROM tdtascanpoint ORDER BY random() LIMIT 10;
```

Von der ganzen Tabellen werden, durch die Zufallszahl *random* gesteuert, 10 Datensätze (Ausgabe der ID) ausgewählt.

```
r_cor(Variable x (Spaltenname)::TEXT, Variable y::TEXT,
Tabellenname::TEXT, Zufallsstichprobe::INTEGER)
```

Beispiel:

```
SELECT r_cor('range(geom_point, geom_planepos)', 'intens',
'tdtascanpoint', 10000);
```

Korrelationskoeffizient zwischen *range* und Intensität	
Zufallsstichprobe	**r**
10	-0.743
100	-0.351
1000	-0.407
10000	-0.420
100000	-0.416

Tab. 8.25: Korrelationskoeffizienten zwischen Länge des Laservektors (*range*) und Intensität für unterschiedlich große Zufallsstichproben.

Die Rechenzeiten für die Bestimmung des Korrelationskoeffizienten hängen in erster Linie von der Größe der Grundgesamtheit, der Größe der Zufallsstichprobe und von der Art der Variablen ab. Die Berechnung ist schneller, wenn die Variablen bereits in einer Tabelle gespeichert sind. Bei der Korrelation zwischen *range* und Intensität wird der *range* on-the-fly berechnet (vgl. Beispiel oben), was zu längeren Rechenzeiten führt. Für die Bestimmung der Zufallsstichprobe muss die Grundgesamtheit zufällig sortiert werden. Je größer die Grundgesamtheit, desto länger wird dieser Prozess in Anspruch nehmen. Die Größe der

Zufallsstichprobe beeinflusst die Dauer der Berechnung des Korrelationskoeffizienten. Je größer die Zufallsstichprobe, desto mehr Punkte müssen verglichen werden, aber desto repräsentativer ist der berechnete Koeffizient. Die Berechnung des Korrelationskoeffizienten zwischen *range* und Intensität beläuft sich bei einer Grundgesamtheit von n=3797446 und einer Zufallsstichprobe von 10000 auf ca. 3 m28 s. Wird die Stichprobe auf 50000 erhöht, beträgt die Rechenzeit ca. 5 m 45 s.

8.4.3.2 Regressionsgerade

Die Regressionsanalyse untersucht die Art des Zusammenhangs (z.B. linear) zwischen Variablen. Der einfachste Fall ist die so genannte lineare Einfachregression. Die lineare Einfachregression setzt zwei Variablen miteinander in Beziehung und gibt den Zusammenhang in Form einer Geradengleichung zurück. Im Gegensatz zur Korrelationsanalyse gibt es bei der Regressionsanalyse eine abhängige und eine unabhängige Variable. Die Intensität des reflektierten Laserstrahls hängt unter anderem (siehe Formel 8.6) von der Länge des Laservektors ab. Somit ist die Intensität die Zielgröße bzw. die abhängige Variable *y* und die Länge des Laservektors die Ausgangsgröße bzw. die unabhängige Variable *x*. Man spricht von einer Regression von *y* nach *x* (BAHRENBERG & GIESE, 1975, 128ff.).
Im Folgenden wird die lineare Einfachregression mit zwei Variablen behandelt. Zusätzlich zum Korrelationskoeffizienten soll eine Regressionsgerade ($y = a + bx$), die die Punktwolke am besten repräsentiert, berechnet und im Korrelationsdiagramm eingezeichnet werden. Die Unbekannte *b* wird als Regressionskoeffizient bezeichnet. Die Variablen *a* und *b* der Geradengleichung werden wie folgt berechnet:

$$b = \frac{cov(x, y)}{var(x)} \tag{8.12}$$

Die Steigung b der Regressionsgeraden wird als Kovarianz dividiert durch Varianz von *x* ausgedrückt (DOLIĆ 2004, 217).

$$a = \bar{y} - \frac{cov(x, y)}{var(x)} \cdot \bar{x} \tag{8.13}$$

Der Ordinatenabstand *a* der Gerade wird durch Freistellen in der Geradengleichung und durch Einsetzen der Mittelwerte von *x* und *y* berechnet.

Die PL/R-Funktion *r_corplot* ergänzt die Funktion *r_cor*, indem die Regressionsgerade der Werte berechnet wird. Das Korrelationsdiagramm mit der Regressionsgeraden wird in einer JPEG-Datei abgespeichert.

```
r_corplot(Variable x (Spaltenname)::TEXT, Variable y::TEXT,
Tabellenname::TEXT, Zufallsstichprobe::INTEGER, Titel des
Diagramms::TEXT)
```

Beispiel:

```
SELECT r_corplot('range(geom_point, geom_planepos)', 'height_agl',
'tdtascanpoint', 1000, 'correlation between range and height above
ground level');
```

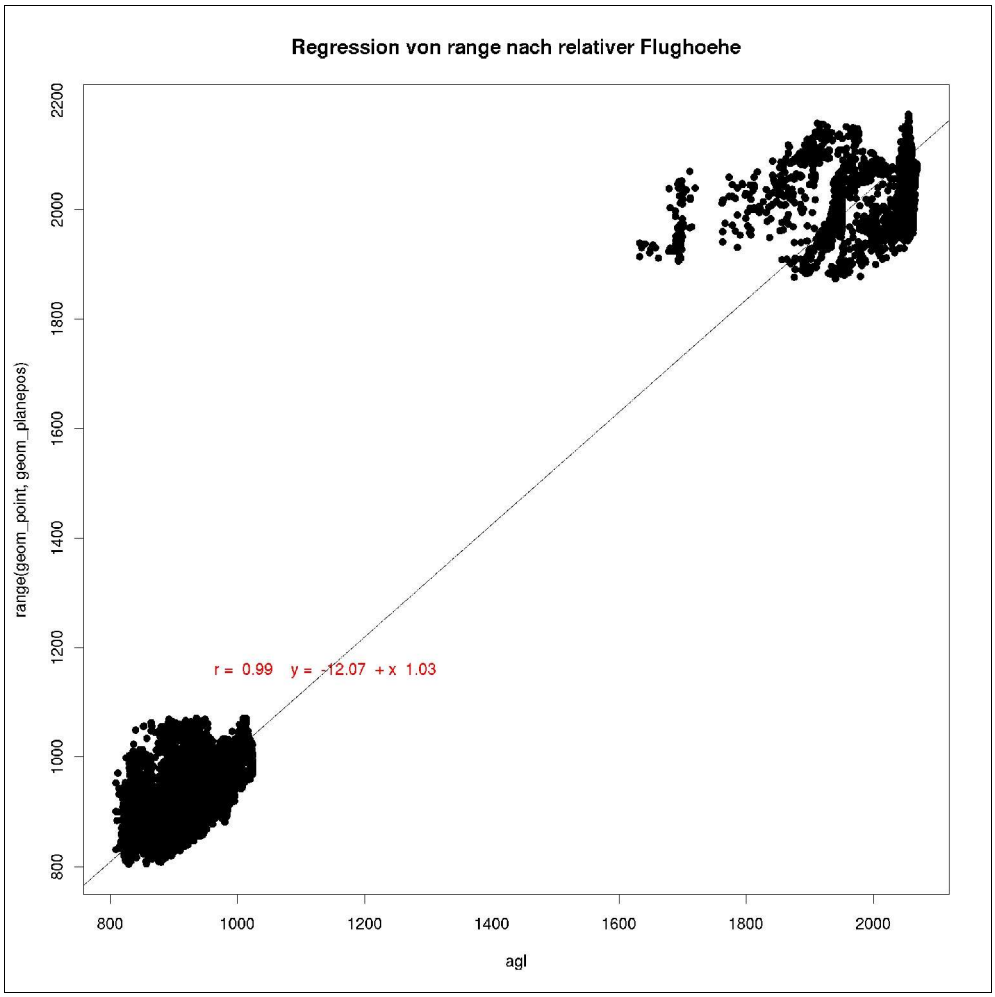


Abb. 8.21: Korrelationsdiagramm mit eingezeichneter Regressionsgeraden zwischen Länge des Laservektors (*range*) und relativer Flughöhe. n=10000 (eigener Entwurf).

Die Korrelation zwischen Länge des Laservektors (*range*) und relativer Flughöhe (*height above ground level*) ist in der Regel nur wenig von äußeren Umständen beeinflusst. Der *range* ist eine Funktion von relativer Flughöhe und Scanwinkel (vgl. Abb. 8.3). Der

Korrelationskoeffizient (r = 0.99 bei n = 3787446) bestätigt diesen Zusammenhang. Es wird eine Regressiongerade y = -12.07 + 1.03 x für die Regression *range* (y) nach relativer Flughöhe (x) bei einer Zufallsstichprobe von n = 10000 berechnet (vgl. Abb. 8.21).

8.5 Berechnung von Rasterdaten

8.5.1 Methodik

Das Rasterdatenmodell bietet eine gute Möglichkeit kontinuierlich verlaufende, räumliche Phänomene zu visualisieren. Die Laserpunkte liegen im Vektordatenmodell vor und repräsentieren den Raum nur punktuell. Das Geographische Informationssystem GRASS bietet die Möglichkeit, Punktdaten mit Hilfe von Interpolationsverfahren in das Rasterdatenmodell zu überführen (Abb. 8.22).
GRASS 5.7 stellt zwei Funktionen für den direkten Zugriff auf die PostgreSQL-Datenbank zur Verfügung (INTERNET: GRASS Online Manual). Zuerst muss mit **db.connect** die Verbindung zur Datenbank definiert werden:

Beispiel:

```
db.connect driver=pg database='host=localhost,user=laser,dbname=laser'
```

Die Funktion **v.external** setzt einen *read-only* Link zu den Geometriedaten in der PostgreSQL-Datenbank. Die Geometrie- und Attributdaten bleiben in der Datenbank. Mit den verlinkten Vektoren kann nun in GRASS gearbeitet werden. Die verlinkten Vektoren besitzen keine echte Topologie. Es wird eine so genannte *Pseudo-Topologie* verwendet (INTERNET: GRASS Online Manual).

Beispiel:

```
v.external dsn='PG:host=localhost user=laser dbname=laser'
layer=tdtascanpoint output=grass_vector
```

Die zweite Möglichkeit stellt die Funktion **v.in.db** dar. Die Funktion erstellt einen GRASS-Punktvektor aus den x-, y-, z-Koordinaten einer PostGIS-Tabelle. Die Geometrie und die ID werden ins GRASS importiert und die Topologie wird aufgebaut. Die Attributdaten bleiben in der Datenbank und sind über die ID mit dem GRASS-Vektor verknüpft (INTERNET: GRASS Online Manual).

Beispiel:

```
v.in.db driver=pg database='host=localhost,user=laser,dbname=laser'
table=tdtascanpoint x='x(geom_point)' y='y(geom_point)' z='z
(geom_point)' key=idscanpoint output=grass_vector
```

Will man nur bestimmte Laserpunkte im GIS zur Verfügung haben, kann in der Datenbank eine Sicht (*view*) erzeugt werden, die bestimmte Laserpunkte mit Hilfe von SQL-Statements auswählt. In weiterer Folge wird diese Sicht als Tabelle in der Funktion *v.in.db* angegeben.

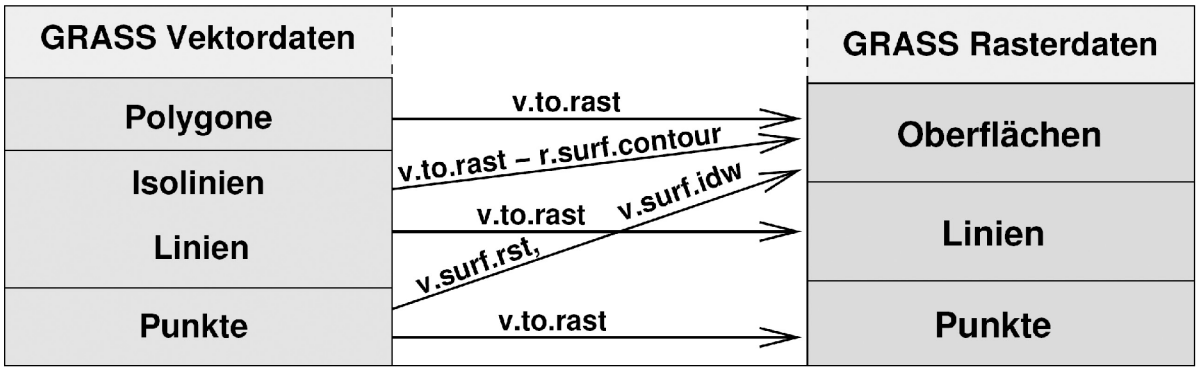


Abb. 8.22: GRASS-Module zur Umwandlung von Vektor- in Rasterdaten (DASSAU et. al. 2004, 98).

In Abbildung 8.22 sind die GRASS-Funktionen für die Interpolation von Vektor- in Rasterdaten zu sehen. Für die Interpolation von Punktvektoren in eine Rasteroberfläche sind die Funktionen **v.surf.rst** und **v.surf.idw** vorgesehen.

Die Funktion *v.surf.idw* ist eine lokal wirkende Interpolationsmethode, die auf dem Prinzip der Inversen Distanzgewichtung (*inverse distance weighted*, IDW) basiert. DASSAU et. al. (2004, 100f.) beschreiben IDW wie folgt:

„Je näher ein zu interpolierender Punkt an einem Punkt mit bekanntem Wert liegt, desto ähnlicher ist der Wert des zu interpolierenden Punktes zum bekannten Wert in der Nähe. Dabei wird zuerst die Distanz zwischen dem gesuchten Punkt und den umliegenden Stützpunkten berechnet. Anschließend erfolgt die Berechnung der gesuchten Punkte als Mittelwert der Umgebungsstützpunkte, und zwar gewichtet mit dem Kehrwert der Distanzen.“

Wird die Anzahl der zur Interpolation zu verwendenden Stützpunkte auf 1 gesetzt, berechnet *v.surf.idw* Thiessenpolygone (vgl. DASSAU et. al. 2004, 102). Die Standardeinstellungen von IDW sind 12 Nachbarpunkte und die Gewichtung mit dem Kehrwert der Distanz zum Quadrat ($1/d^2$).

Die Funktion *v.surf.rst* (*regularized splines with tension*, RST) ist eine Spline-Interpolationsmethode. Die Spline-Interpolation, die vor allem für die Schließung von großen Datenlücken geeignet ist, ist nur der Vollständigkeit halber erwähnt und wurde in der Diplomarbeit nicht verwendet (siehe NETELER & MITASOVA 2004, 160ff.).
Mit Hilfe der Sprache PL/pgSQL und den Funktionen von PostGIS können Rasterdaten direkt in der Datenbank erzeugt werden. Dieser Schritt ist im Rahmen der Diplomarbeit nicht durchgeführt worden und kann nur theoretisch beschrieben werden.
Legt man einen beliebigen Polygonraster – regelmäßig oder unregelmäßig – über die Laserpunkte und selektiert ein bestimmtes Attribut (z.B. maximaler Intensitätswert aller Punkte in der Zelle) und weist es den Polygonzellen zu, kann dieser Polygonraster mit den zugewiesenen Attributen in einem weiteren Schritt in Form eines Punktrasters exportiert werden. Dieses Prinzip basiert darauf, dass für jede Zelle bzw. jedes Polygon des Rasters eine räumliche Abfrage gemacht wird und die Punkte ausgewählt werden, die räumlich gesehen in dieser Fläche liegen. Nach der Auswahl der Punkte wird der gewünschte Attributwert berechnet und in den Datensatz dieser Zelle geschrieben. Dieser Polygonraster, der keine Topologie besitzt, kann entweder als Vektor oder in Form eines Punktrasters, der den Mittelpunkten der Polygonzellen entspricht, weiterverarbeitet werden.
Ein anderer Weg wäre die Generierung eines Punktrasters (Punkte die regelmäßig über den Raum verteilt sind). Mit Hilfe der Distanzfunktion kann um jeden Punkt des Punktrasters ein Buffer gelegt werden, der die Laserpunkte in dieser Bufferzone selektiert. Im Gegensatz zum Polygonraster wird den Stützpunkten des Punktrasters der berechnete Wert zugewiesen. Die Datenmenge bzw. die Anzahl der Koordinaten des Punktrasters ist um das fünffache geringer als die Datenmenge des Polygonrasters.
Dieser Punktraster, dessen Datenmenge im Vergleich zu den Laserpunkten um ein Vielfaches geringer ist, kann nun als Vektor in das GIS exportiert und dort in einen Raster interpoliert werden. Die Export- und Rechenzeiten sind wesentlich geringer als beim Export aller Laserpunkte.

8.5.2 Intensitätsraster

Das Ergebnis der Rasterinterpolation des Intensitätsattributs der Laserpunkte hängt von den gewählten Einstellungen der IDW-Methode, der Rasterweite und den Laserpunkten, die als Stützpunkte herangezogen werden, ab. Die Rasterweite wird auf 0.2 m festgelegt, damit die Ergebnisse verglichen werden können. Einerseits wurde die Anzahl der Nachbarpunkte der IDW-Interpolation variiert (1, 2 und 10 Nachbarpunkte) und andererseits wurden unterschiedliche Stützpunktvarianten eingesetzt (alle Laserpunkte, nur *first pulse*, nur *last pulse*, nur Befliegung ID=1, nur Befliegung ID=2; vgl. Appendix F). Dem Histogramm der Intensität kann entnommen werden, dass ab ca. Intensität > 190 nur noch Ausreißer liegen (Median=40). Bei einem kontinuierlichen Farbverlauf von Schwarz nach Weiß über alle Werte der Intensität, wäre ein stark verdunkeltes Bild das Ergebnis. In GRASS wird den Intensitätsrastern ein frei definierter Farbverlauf zugewiesen. Alle Werte über 190 (ca. 75 %

von 255) werden weiß eingefärbt. Zwischen 0 und 190 wird ein kontinuierlicher Farbverlauf von Schwarz nach Weiß eingestellt (Abb. 8.23).

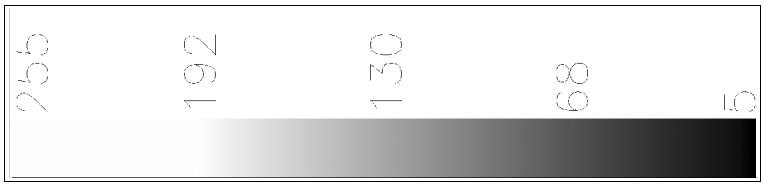


Abb. 8.23: Legende für die Intensitätsraster (schwarz...niedrige Intensitätswerte, weiß...hohe Intensitätswerte; eigener Entwurf).

Abb. 8.24: Intensitätsraster des Untersuchungsgebietes berechnet mit allen Laserpunkten (IDW: 1 Nachbarpunkt; eigener Entwurf).

In den helleren Bereichen von Abbildung 8.24 fallen regelmäßig gemusterte schwarze Punkte, d.h. Punkte mit niedriger Intensität, auf. Die statistischen Plots der Intensität (vgl. Appendix D.1) zeigen, dass die Befliegungskampagne mit der ID=1 des Untersuchungsgebietes wesentlich niedrigere Intensitäten, aufgrund der größeren Flughöhe, als die zweite Befliegung (ID=2) aufweist. Die Befliegung mit der ID=2 diente der Verdichtung der Messpunkte im Stadtgebiet von Hohenems. Wird nun aus allen Laserpunkten (erste und zweite Befliegung) ein Raster gerechnet, kommt es zu dem oben genannten störenden Einfluss im Intensitätsbild.

Abb. 8.25: Intensitätsraster des Untersuchungsgebietes berechnet mit den Laserpunkten der Befliegung ID=2 (3058889 Punkte; IDW: 1 Nachbarpunkt; eigener Entwurf).

Die Abbildung 8.25 wurde aus den Punkten (3058889 Punkte) der zweiten Befliegung (ID=2) gerechnet. Das Intensitätsbild erscheint nun ohne das Rauschen der ersten Befliegung.

Wenn die Intensitäten der ersten Befliegung (höhere Befliegung) geometrisch korrigiert werden würden, könnte man alle Laserpunkte für die Erstellung eines rauschfreien Bildes heranziehen. Dieses Problem der Vergleichbarkeit der Intensitätswerte aus mehreren Befliegungen bzw. Flugstreifen, die eine unterschiedliche Aufnahmegeometrie aufweisen, wird im alpS-Forschungsprojekt ein Gegenstand der Untersuchung sein. Inwieweit die statistischen Zusammenhangsmaße (v.a. Regressionsanalyse) zur Lösung dieses Problems beitragen können, kann zum jetzigen Zeitpunkt nur schwer abgeschätzt werden. Die Ergebnisse der statistischen Auswertung, die den negativen Zusammenhang zwischen *range* und Intensität bestätigen (vgl. Appendix D.1), sind nur für dieses Untersuchungsgebiet repräsentativ. Weiterführende Vergleiche der statistischen Ergebnisse zwischen mehreren Untersuchungsgebieten bzw. Datensätzen, sollten Aufschluss darüber geben, ob ein einfaches statistisches Korrekturverfahren für die Intensität einsetzbar ist.
Zum besseren Vergleich der beiden Befliegungen wurde ein Intensitätsraster mit demselben Farbverlauf (Abb. 8.23) wie die vorher gezeigten Bilder und ein Raster mit einer modifizierten Farbskala berechnet (Abb. 8.26). Für die erste Befliegung (ID=1), deren Intensitätswerte deutlich unter den Werten der zweiten Befliegung (ID=2) liegen, wird ein Schwellenwert von 15 % (Intensität ca. 38) angenommen. Das heißt, dass alle Werte über 38 auf weiß gesetzt werden und von 0 bis 38 ein Farbverlauf von Schwarz nach Weiß Anwendung findet (Abb. 8.26).

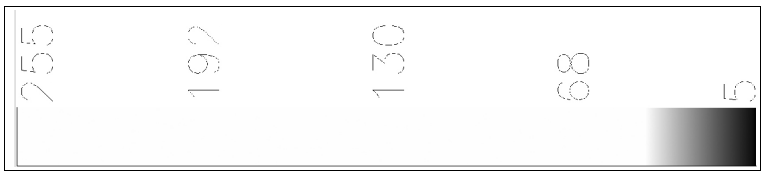


Abb. 8.26: Modifizierte Legende für die Intensitätsraster der Befliegung ID=1(schwarz...niedrige Intensitätswerte, weiß...hohe Intensitätswerte; eigener Entwurf).

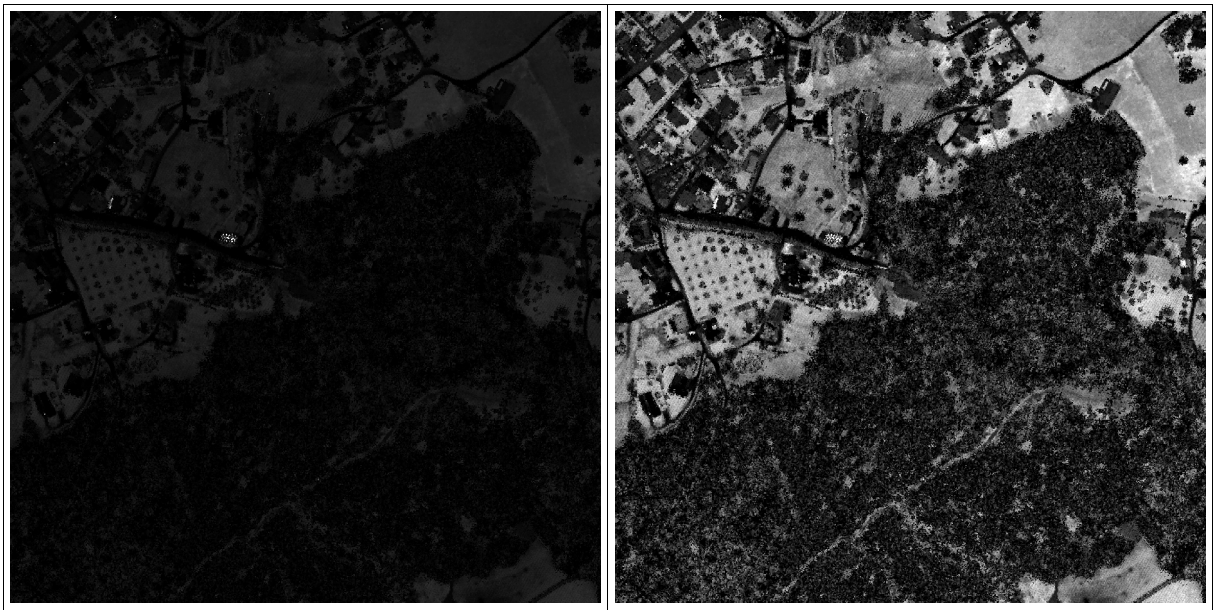


Abb. 8.27: Intensitätsraster des Untersuchungsgebietes berechnet mit den Laserpunkten der Befliegung ID=1 (738557 Punkte; IDW: 1 Nachbarpunkt). links: Farbschema laut Abb. 8.23, rechts: Farbschema laut Abb. 8.26 (eigener Entwurf).

8.6 Zusammenfassung

Die Entwicklung neuer Funktionen für die Datenanalyse steht im Zentrum des Kapitels *„8 Datenverarbeitung“*. Dabei kommen die drei wichtigsten Komponenten des Informationssystems, die räumliche Datenbank, die Statistiksoftware und das Geographische Informationssystem (GIS) zum Einsatz. Die absolute zeitliche Verortung, die vor allem für multitemporale Analysen benötigt wird, kann mit einer einfachen Datenbankfunktion (PL/pgSQL), die das relative Zeitformat der einzelnen Datensätze mit der absolut verorteten Befliegung verknüpft, umgesetzt werden. Dieses Kapitel soll gezeigt haben, wie wichtig die Rekonstruktion der Aufnahmegeometrie für weiterführende Analysen, wie zum Beispiel die geometrische Intensitätskorrektur, ist. Die Umsetzung einer vektorbasierten Modellierung des Geländes bringt Vor- und Nachteile. Die Laserpunkte können im Vektorformat, in dem sie bereits vorliegen, prozessiert werden. Bei der Modellierung werden ausschließlich Datenbankfunktionen angewendet. Es findet kein Datentransfer statt, da die Funktionen direkt auf die einzelnen Objekte zugreifen. Dies kann über ein beliebiges Attribut geschehen. Bei der Rekonstruktion der Flugzeugposition werden die Datensätze über die Zeit und bei der Modellierung der Geländeebene über ein räumliches Suchkriterium ausgewählt. Die dreidimensionale, räumliche Suche nach Nachbarpunkten bei der Konstruktion der Geländeebene bringt als Nebeneffekt die Identifizierung von Punkten an Objektkanten und von Punkten in hoher Vegetation (höher als das Suchkriterium), mit sich.

Die Modellierung der Aufnahmegeometrie unterscheidet nicht zwischen Gelände- und Objektpunkten, sondern geht immer nach dem Prinzip der minimalen Distanz vor. Das Ergebnis der Modellierung sollte viel mehr als Modell der Oberfläche, die der Lichtkegel bei der Abtastung vorgefunden hat, verstanden werden. Wird zum Beispiel die Ebene für einen Bodenpunkt berechnet, ist anzunehmen, dass die zwei nächsten Nachbarpunkte ebenfalls Bodenpunkte sind. Somit wird eine Ebene, die das „Gelände“ im Sinne der Fachtermini repräsentiert, konstruiert. Wird jedoch eine Ebene für einen Punkt in einer Baumkrone abgeleitet, ist es am wahrscheinlichsten, dass die zwei nächsten Punkte ebenfalls in der Baumkrone liegen. Dies könnte als Ebene, die die „Oberfläche“ repräsentiert, bezeichnet werden. Im Maßstab des gesamten Untersuchungsgebietes ergibt sich eine stark heterogene Verteilung von „Gelände- und Oberflächenebenen“. Eine Ableitung von Digitalen Gelände- bzw. Oberflächenrastern kann mit dieser Methode nicht durchgeführt werden.
Die statistische Auswertung der Punktwolke zeigt sehr viel Potential für weiterführende Anwendungen auf. Die deskriptiven Kennwerte der einzelnen Variablen (z.B. Intensität, *range*, usw.) und die Zusammenhangsstatistik helfen die theoretisch bekannten physikalischen Zusammenhänge am Beispiel eines realen Datensatzes besser zu verstehen. Man kann dabei noch einen Schritt weitergehen und versuchen diese Zusammenhänge bzw. Beziehungen in empirische Modelle zu fassen.
Die aus der statistischen Auswertung gewonnen Erfahrungen konnten bei der Berechnung der Intensitätsraster bzw. -bilder umgesetzt werden. Die große Punktdichte im Untersuchungsgebiet (ca. 15 Punkte/m^2) erlaubt die Verwendung eines einfachen Interpolationsverfahrens. Je mehr Nachbarpunkte für die Interpolation mit dem IDW-Verfahren verwendet werden, desto stärker wird das Bild geglättet. Dadurch kommt es aber zur Veränderung (zur Interpolation) der Intensitätswerte, was für gewisse Anwendungen (z.B. Klassifikation des Rasters in einer Fernerkundungssoftware) nicht erwünscht ist. Die Interpolation mit nur einem Nachbarpunkt (vgl. Thiessenpolygone) ergibt einen Raster mit nur wenig veränderten Werten. Die Trennung der einzelnen Laserpunkte nach bestimmten Kriterien (z.B. *range*, *first* oder *last pulse*) vor der Interpolation, führt zu wesentlich besseren Ergebnissen, rein visuell betrachtet, als die Interpolation aller Laserpunkte. Die bei der statistischen Auswertung gewonnen Histogramme der Intensität ermöglichen eine optimale Anpassung der Farbskala (Farbcodierung der einzelnen Werte). Vor der Erstellung der Intensitätsbilder muss der Zweck der Anwendung festgelegt werden. Werden die ursprünglichen Intensitätswerte oder Bilder, die eine visuelle Interpretation erlauben, benötigt?

9 Datenexport und Visualisierung

9.1 Methodik

Der Export der Daten des Informationssystems ist in die verschiedensten Datenformate (z.B. GIS-Formate und Graphik-Formate bzw. Vektor- und Rasterformate, etc.) möglich. Die Entwicklergemeinden von *open source* Software achten sehr darauf, dass ihre Programme möglichst offen und kompatibel mit anderer Software gestaltet werden. Die meisten Programmiersprachen ermöglichen den Zugriff auf die PostgreSQL-Datenbank und die meisten *open source* GIS-Programme können PostGIS-Tabellen lesen und direkt importieren. Es gibt zwei grundlegende Exportvarianten. Einerseits können die Daten von der Datenbank in ein entsprechendes Format geschrieben werden (**„direkter Datenbankexport“**), das vom Zielprogramm gelesen werden kann, und andererseits kann mit geeigneter Software direkt auf die Daten in der Datenbank zugegriffen werden. Das Zielprogramm liest die Daten ein und kann sie dann wiederum in ein anderes Format umwandeln (**„indirekter Datenbankexport“**). Dieses Kapitel behandelt nur eine kleine Auswahl von Exportmöglichkeiten, die für die Arbeit mit den Punktdaten des Laserscannings von besonderem Interesse sind.
Abbildung 9.1 gibt einen Überblick über die wichtigsten Exportvarianten. Die Daten können direkt mit der Exportfunktionalität von PostgreSQL, PostGIS oder von PL/R (R-Funktionalität) aus der Datenbank geschrieben werden. Die indirekten Möglichkeiten gehen über die Geographischen Informationssysteme GRASS und QGIS, über die Statistiksoftware R und über das Visualisierungsprogramm GGobi.
Die unten angeführten Befehle beziehen sich auf die aktuellen Softwareversionen (vgl. Tab. 5.8) und können sich in einer anderen Version unterscheiden.

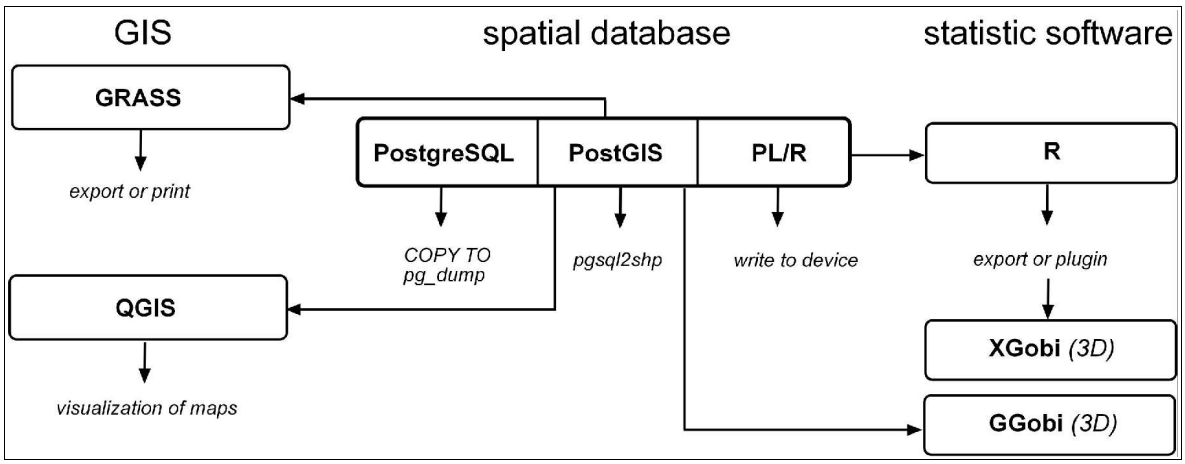


Abb. 9.1: Ausgewählte Exportmöglichkeiten (eigener Entwurf).

9.2 Direkter Datenbankexport

9.2.1 PostgreSQL

Der Datenbankserver PostgreSQL besitzt standardmäßig zwei effiziente Funktionen für den Export von großen Datenmengen. Der Befehl **COPY**, der bereits im Kapitel 7 beschrieben wurde, kann die Daten einer Datei importieren, aber auch die Daten der Tabellen in eine Datei exportieren. COPY kann immer nur eine Tabelle gleichzeitig exportieren. Beim Export kann zwischen dem ASCII-Format und einem binären Format gewählt werden. Zusätzlich kann das Trennzeichen, das die Spalten der Tabellen voneinander trennt, beliebig eingestellt werden. Das Standardtrennzeichen ist der Tabulator (vgl. BOENIGK 2003, 164ff.).

Beispiel:

```
COPY tdtastrip TO '/tmp/ausgabedatei' USING DELIMITERS ',' WITH NULL AS
'-9999';
```

Die Flugstreifen-Tabelle wird in eine ASCII-Datei geschrieben. Als Trennzeichen wird der Beistrich verwendet und NODATA-Werte werden mit -9999 codiert.

Die wohl einfachste Möglichkeit eine gesamte Datenbank zu exportieren bzw. zu sichern, ist mit dem Befehl **pg_dump** gegeben. Der Befehl wird in der Kommandozeile des Betriebssystems ausgeführt. Es werden alle Daten und SQL-Statements, die für die Wiederherstellung der kompletten Datenbank benötigt werden, in eine Datei geschrieben. Die so archivierten Tabellen mit ihren Daten können mit dem Befehl *psql -f* rekonstruiert werden.

Beispiel:

```
pg_dump -h localhost -Fc -f dumpfile.tar.gz laser
```

Die Datenbank *laser* wird vom lokalen Server (*localhost*) in eine komprimierte Archivdatei geschrieben.

Probleme beim Export mit *pg_dump* können dann auftreten, wenn die Archivdatei die maximal erlaubte Dateigröße des Betriebssystems überschreitet. Bei großen Datenmengen ist eine Kombination von *pg_dump* und *COPY* anzuwenden. Das Datenbankschema – alle SQL-Statements für das Erzeugen der Tabellen – kann mit *pg_dump* exportiert werden. Die einzelnen Tabellen können dann mit COPY in mehrere Dateien geschrieben werden.
Bei der Wiederherstellung der Datenbank ist darauf zu achten, dass der Ziel-Datenbankserver softwaremäßig gleich ausgestattet ist, wie der Server, von dem die Daten

stammen. Eine Rekonstruktion von PostGIS-Geometriespalten zum Beispiel kann nur dann erfolgen, wenn PostGIS (am besten in der gleichen Version) am Zielserver installiert ist.

9.2.2 PostGIS

Die Installation von PostGIS enthält die Befehle **pgsql2shp** und **shp2pgsql**. *pgsql2shp* kann Tabellen mit PostGIS-Geometrie in ESRI s*hapefiles* umwandeln. *shp2pgsql* geht den umgekehrten Weg und ermöglicht den Import von *shapefiles* in die Datenbank, wobei eine PostGIS-Tabelle erzeugt wird.

Beispiel:

```
pgsql2shp -f name_shapefile -h localhost -g geom_line -d laser
tdtastrip
```

Die Flugstreifen-Tabelle (*tdtastrip*) wird vom lokalen Datenbankserver in ein dreidimensionales *shapefile* (*-d*) unter Verwendung der PostGIS-Spalte *geom_line* als Geometriefeld geschrieben. Eine PostGIS-Tabelle kann mehrere Geometriespalten haben, ein *shapefile* jedoch nur eine.
Der Datenexport mit *pgsql2shp* erreicht ansprechende Exportzeiten auch für große Datenmengen. Der Export aller Laserpunkte im Untersuchungsgebiet (3797446) dauert ca. 4 Minuten und schreibt ein *shapefile* mit ca. 450 MB.
PostGIS bietet noch weitere SQL-Funktionen, wie z.B. die Funktion *AsSVG*, die die Geometrie in eine *Scalable Vector Graphics (SVG)* konvertiert (www.svg.cc).

9.2.3 Profile mit PL/R

Die Projektion der Laserpunkte auf eine Fläche, macht ab einem bestimmten Maßstab keinen Sinn mehr, da die Symbole der Punkte die komplette Karte bzw. den ganzen Bildschirm ausfüllen (Abb. 9.2). Erst eine starke Reduktion der Punkte nach bestimmten Kriterien führt zu einem befriedigenden Ergebnis. Diese Ausdünnung der Punkte kann entweder nach Attributwerten oder nach räumlichen Kriterien erfolgen. Ersteres könnte zum Beispiel durch die Selektion von Punkten, die einen bestimmten Intensitätswert unter- bzw. überschreiten, geschehen. Letzteres wurde mit der Entwicklung einer Funktion, die Punkte entlang einer Linie auswählt und dann in einem **Profildiagramm** darstellt, umgesetzt. Die Auswahl der Punkte wird mit einer PL/pgSQL-Funktion durchgeführt. Die ausgewählten Punkte werden einer PL/R-Funktion übergeben, die dann die Visualisierung des Diagramms übernimmt. Dabei kommen die graphischen Fähigkeiten von R (in Form einer PL/R-Funktion) zum Einsatz.

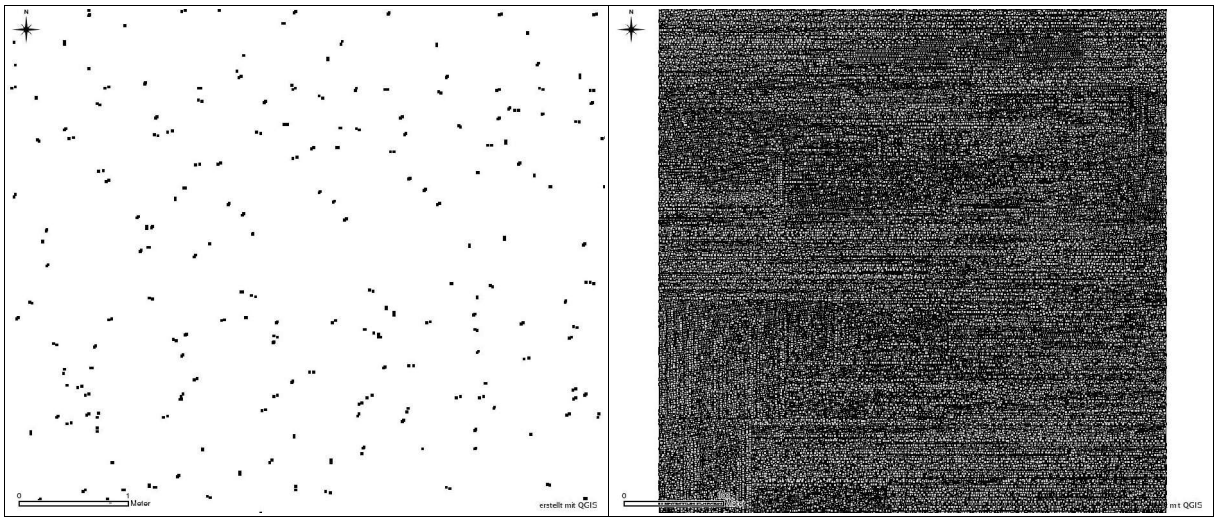


Abb. 9.2: Visualisierung der Laserpunkte in QGIS. links: ca. 5 x 4 m großer Ausschnitt, rechts: gesamtes Untersuchungsgebiet (eigener Entwurf).

Wird eine zweidimensionale Profillinie durch die Punktwolke gezogen, liegen meist nur sehr wenige Punkte mathematisch exakt auf dieser Linie. Aus diesem Grund wird ein Toleranzbereich definiert, der zusätzlich alle Punkte selektiert, die in diese Toleranz bzw. Distanz zur Profillinie fallen (Abb. 9.3). Korrekter müsste man also von einem „Profilstreifen“ sprechen. Die Profillinie wird zuerst an beiden Enden um den Toleranzbereich verkürzt. Dann werden alle Punkte selektiert, deren zweidimensionale Distanz zur verkürzten Profillinie geringer als die angegebene Toleranz ist.

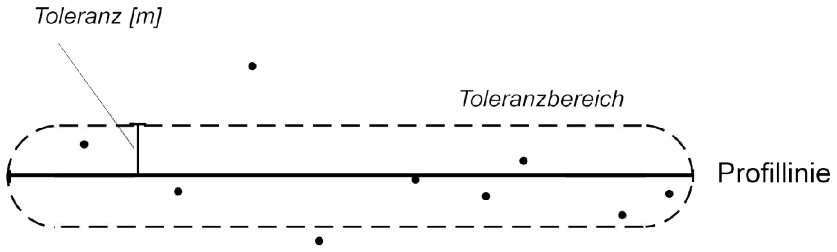


Abb. 9.3: Profillinie mit eingezeichnetem Toleranzbereich (eigener Entwurf).

Ein zweidimensionales Profil setzt sich aus einer *x*- und *y-Koordinate* zusammen. Die x-Koordinate entspricht der zurückgelegten Entfernung auf der Profillinie und die y-Koordinate der absoluten Höhe des Laserpunkts. Durch die Einführung einer Toleranz müssen die Punkte, die nicht direkt auf der Linie liegen, auf diese projiziert werden. Dies geschieht mit dem Normalabstand des Punktes zur unverkürzten Profillinie (Abb. 9.4).

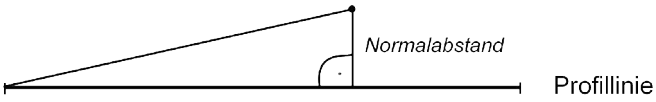


Abb. 9.4: Entfernung des Laserpunkts vom Startpunkt des Profils und Normalabstand zur Profillinie (eigener Entwurf).

Das Informationssystem bietet zwei verschiedene SQL-Funktionsaufrufe für die Selektion der Punkte, die für die Erstellung von zweidimensionalen Profilen herangezogen werden.

```
profile2d(x-Koordinate des Startpunkts::FLOAT, y-Koordinate des
Startpunkts::FLOAT, x-Koordinate des Endpunkts::FLOAT, y-Koordinate des
Endpunkts::FLOAT, Toleranz [m]::FLOAT)
```

Die x-Koordinate entspricht dem UTM-Rechtswert und die y-Koordinate dem UTM-Hochwert. Die Funktion gibt die Entfernung des Punktes auf der Profillinie und die absolute Höhe zurück.

Die Erfahrungen bei der GPS-Messung im Gelände haben gezeigt, dass die Aufnahme von zwei Messpunkten im steilen Gelände aufgrund der Erreichbarkeit und im bewaldeten Gebiet aufgrund des GPS-Empfangs nur sehr schwer möglich ist. Die folgende Funktion erleichtert die Festlegung von Profillinien im Gelände. Der Startpunkt des Profils kann an einem gut erreichbaren Punkt eingemessen werden. Mit dem Kompass kann die Himmelsrichtung der Profillinie bestimmt werden. Die Länge der Profillinie ist im Gelände nur schwer abzuschätzen. Die ungefähre Länge des Profils kann einer guten Kartengrundlage (z.B. Orthofoto) entnommen werden.

```
profile2d_exp(x-Koordinate des Startpunkts::FLOAT, y-Koordinate des
Startpunkts::FLOAT, Himmelsrichtung [Grad], Länge der Profillinie [m]::
FLOAT, Toleranz [m]::FLOAT)
```

Die Richtung des Profils wird in Grad zur Nordrichtung (0°) dem Uhrzeigersinn folgend angegeben (Ost = 90°, Süd = 180°, West = 270°).

Die Erstellung des Profildiagramms wird mit einer PL/R-Funktion, die in mehreren Ausführungen vorliegt, durchgeführt:

```
profile2d_pdf(x-Koordinate des Startpunkts::FLOAT, y-Koordinate des
Startpunkts::FLOAT, x-Koordinate des Endpunkts::FLOAT, y-Koordinate des
Endpunkts::FLOAT, Toleranz [m]::FLOAT)
```

Profildiagramm als PDF-Datei.

```
profile2d_jpeg(x-Koordinate des Startpunkts::FLOAT, y-Koordinate des
Startpunkts::FLOAT, x-Koordinate des Endpunkts::FLOAT, y-Koordinate des
Endpunkts::FLOAT, Toleranz [m]::FLOAT)
```

Profildiagramm als JPEG-Bilddatei.

```
profile2d_png(x-Koordinate des Startpunkts::FLOAT, y-Koordinate des
Startpunkts::FLOAT, x-Koordinate des Endpunkts::FLOAT, y-Koordinate des
Endpunkts::FLOAT, Toleranz [m]::FLOAT)
```

Profildiagramm als PNG-Bilddatei.

```
profile2d_txt(x-Koordinate des Startpunkts::FLOAT, y-Koordinate des
Startpunkts::FLOAT, x-Koordinate des Endpunkts::FLOAT, y-Koordinate des
Endpunkts::FLOAT, Toleranz [m]::FLOAT)
```

Ausgabe der selektierten Punkte in einer ASCII-Textdatei.

```
profile2d_exp_jpeg(x-Koordinate des Startpunkts::FLOAT, y-Koordinate
des Startpunkts::FLOAT, Himmelsrichtung [Grad], Länge der Profillinie
[m]::FLOAT, Toleranz [m]::FLOAT)
```

Ausgabe als JPEG-Bilddatei.

Beispiel: SQL-Statement für die Erzeugung von Profil 1 (Abb. 9.5):

```
SELECT * FROM profile2d_jpeg(32551768, 5244763, 32551815, 5244769,
1.0);
```

Abb. 9.5: Standort des Profils Nr. 1 (eigner Entwurf; Orthofoto: Land Vorarlberg; Foto: Höfle 25.12.2004).

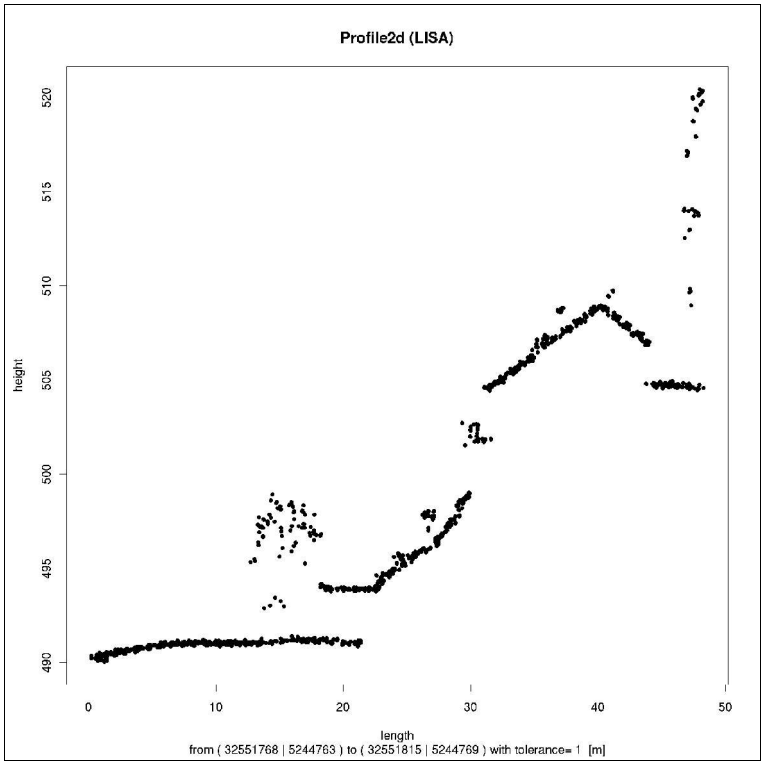


Abb. 9.6: Profil 1 mit einer Toleranz von 1 m (eigener Entwurf).

Das Profildiagramm wird immer auf einen quadratischen Ausschnitt gebracht. Die Skalierung der Achsen (somit die Überhöhung) hängt vom Verhältnis Länge der Profillinie zu Höhenunterschieden im Verlauf des Profils ab. Es besteht zusätzlich die Möglichkeit die Skalierung der Achsen festzulegen. Die absoluten Höhenwerte können leicht in relative Höhenunterschiede umgerechnet werden (z.B. Höhenunterschied zur Minimalhöhe im Profil).
Die Statistiksoftware R besitzt eine Fülle von Zusatzpaketen, wie z.B. das Paket *„Scatterplot3d“* (vgl. INTERNET: CRAN). Dieses Paket besitzt eine Funktion für die Erstellung von dreidimensionalen Diagrammen für Punktwolken. Die Funktionalität dieses Pakets kann mit der Profilfunktion verknüpft werden.

```
profile3d_scatter(x-Koordinate des Startpunkts::FLOAT, y-Koordinate des
Startpunkts::FLOAT, x-Koordinate des Endpunkts::FLOAT, y-Koordinate des
Endpunkts::FLOAT, Toleranz [m]::FLOAT)
```

Dreidimensionaler Scatterplot als JPEG-Bilddatei.

Beispiel:

```
SELECT * FROM profile3d_scatter(32551768, 5244763, 32551815,
5244769, 5.0);
```

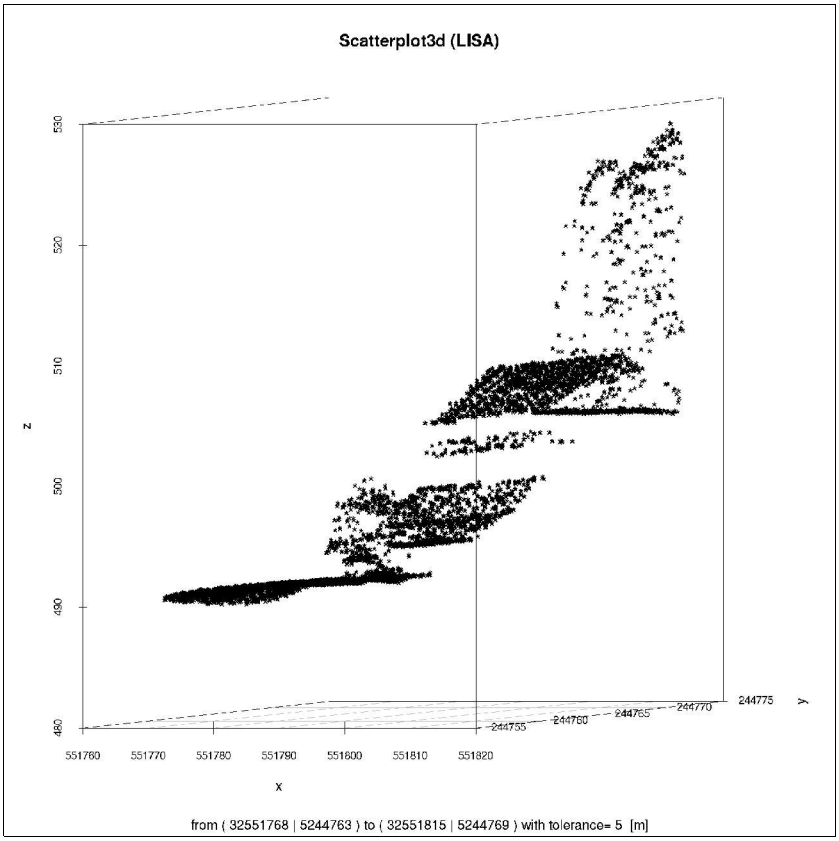


Abb. 9.7: 3D-Scatterplot des Profils 1 mit einer Toleranz von 5 m (eigener Entwurf).

Im Appendix E finden sich weitere Beispiele für Profildiagramme mit unterschiedlichen Toleranzen.

Ein Diskussionspunkt ist die Umsetzung von Profillinien mit mehreren Segmenten (Abb. 9.8). Anstelle von zwei Punkten (Start- und Endpunkt) könnte man eine Liniengeometrie als Input für die Funktion angeben. Die segmentweise Abarbeitung dieser Linie nach dem oben beschriebenen Prinzip ist möglich, wenn in den Überschneidungsflächen der Toleranzbereiche die Punkte eindeutig einem Segment zugeteilt werden.

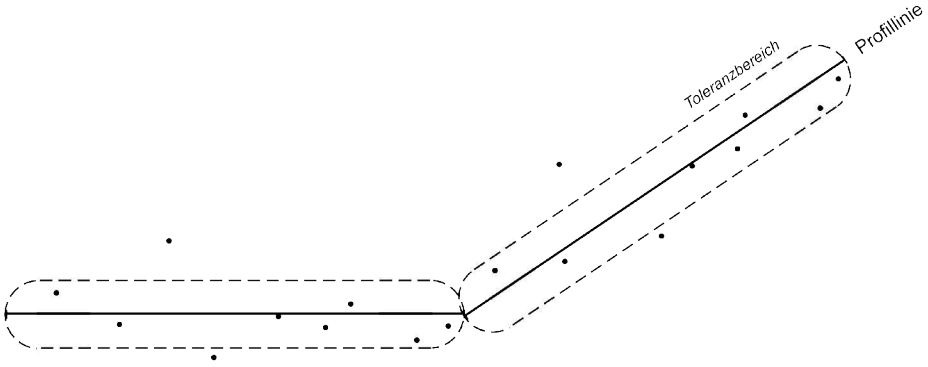


Abb. 9.8: Schematische Darstellung einer Profillinie mit zwei Segmenten (eigener Entwurf).

9.3 Indirekter Datenbankexport

9.3.1 GRASS

Der indirekte Datenexport dient vor allem der Visualisierung der Vektoren der Datenbank und der Berechnung von Rasterdaten aus diesen Vektoren. Im Kapitel *„8 Datenverarbeitung“* wird auf die Zugriffsmöglichkeiten von GRASS auf die PostgreSQL/PostGIS-Datenbank und die Interpolation von Rasterdaten aus den Laserpunkten näher eingegangen. Nach dem Import bzw. nach der Verknüpfung der Laserscannerdaten in die GRASS-Umgebung, steht die gesamte Exportpalette von GRASS zur Verfügung (Tab. 9.1).

Vektordaten	Rasterdaten
ESRI shapefile, E00-Format, Ungenerate-Format, GRASS ASCII Vektorformat, SDTS-Format, DXF-Format, MapInfo-Format	*GRASS ASCII Raster, ASCII GRID (für Surfer, Modflow, etc.), ARC/INFO ASCII GRID, BIL, Binary Array, PPM, MPEG, TIFF (georeferenziert), TARGA, ERDAS LAN, ASCII 3D file, Vis5D file*

Tab. 9.1: Ausgewählte GRASS-Exportformate (vgl. NETELER & MITASOVA 2004).

9.3.2 QGIS

Quantum GIS (QGIS) ist ein sehr einfaches Geographisches Informationssystem für Linux/Unix, aber auch Windows (www.qgis.org). QGIS dient hauptsächlich der Visualisierung von Vektor- und Rasterdaten und bietet einfache Möglichkeiten der räumlichen Analyse (z.B. räumliche Selektion). QGIS besitzt wenig Exportmöglichkeiten, aber kann dafür viele Formate lesen, visualisieren und die daraus entstandenen Karten als Bild speichern. QGIS liest z.B. *ESRI shapefiles*, *ArcInfo coverages*, *MapInfo layers* und PostGIS-Tabellen. Die Abbildung 3.8 wurde mit QGIS erzeugt. Dabei wird direkt auf die Datenbank zugegriffen und es findet kein Datentransfer statt. Bei großen Tabellen sollte vor der Visualisierung ein Filterkriterium (z.B. Laserpunkte eines bestimmten Flugstreifen) angegeben werden, da die Wartezeit sonst mehrere Minuten betragen kann. Zusätzlich zur Visualisierung von PostGIS-Geometrien können in der graphischen Benutzeroberfläche von QGIS ESRI *shapefiles* in die PostgreSQL/PostGIS-Datenbank geschrieben werden (*„Shapefile in PostGIS Import Tool“*).

9.3.3 R

9.3.3.1 XGobi

XGobi ist eine Software für die Visualisierung von mehrdimensionalen Daten (INTERNET: XGobi). Es kann als Standalone-Software oder als Plugin für die Statistiksoftware R verwendet werden. Die Weiterentwicklung von XGobi wurde eingestellt, da mit GGobi eine neu überarbeitete Version von XGobi, die die gesamte Funktionalität übernommen hat, zur Verfügung steht.
Im Informationssystem dient es dazu, die dreidimensionale Punktwolke zu visualisieren. Ein großer Vorteil von XGobi ist, dass es nicht nur die Punkte in einer dreidimensionalen Darstellung anzeigt, sondern auch die Identifikation und die Abfrage von Attributen für einzelne Punkte erlaubt. XGobi besitzt eine Fülle von verschiedenen Darstellungsarten, wie z.B. 1-D-Plots, XY-Plots, 3-D-Rotation (Abb. 9.9), Korrelationsdiagramme, usw.. Das „Highlight“ von XGobi ist sicherlich die Rotation der Punktwolke. Der Benutzer kann durch eine virtuelle Welt der Laserpunkte wandern.
Wie in Abb. 9.1 gezeigt wird, kann XGobi nicht direkt auf die Datenbank zugreifen. Entweder man exportiert die Laserpunkte und ihre entsprechenden Attribute in eine ASCII-Datei und liest sie dann in XGobi ein, oder man greift in R auf die Datenbank zu und übergibt die Daten an das XGobi-Plugin (vgl. Swayne et. al. 1998).

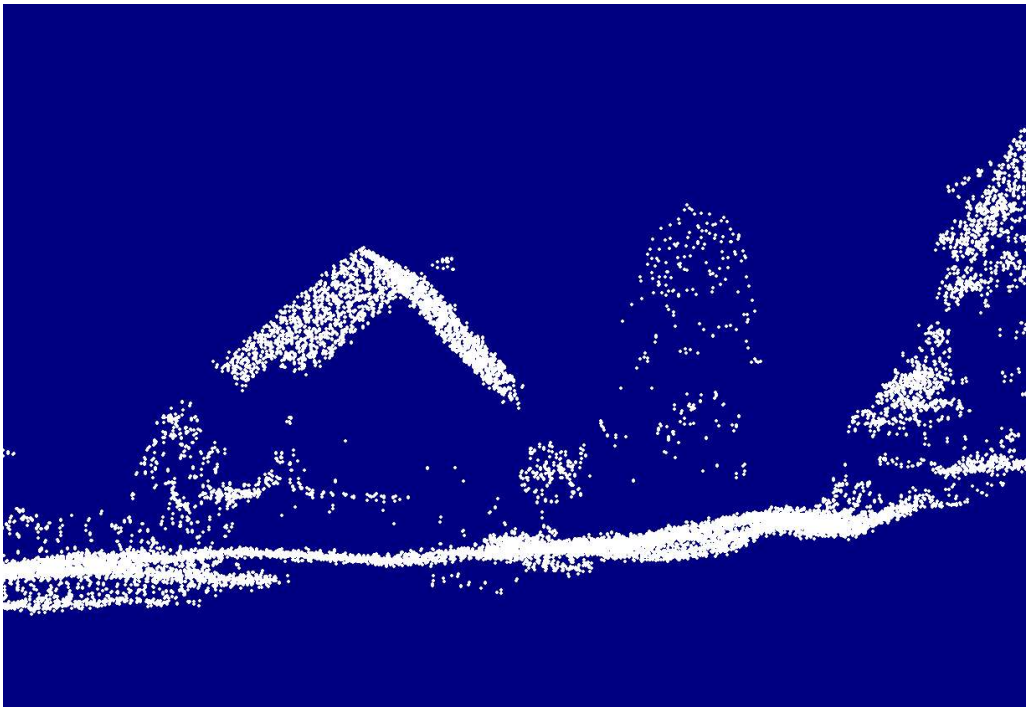


Abb. 9.9: Screenshot einer 3-D-Visualisierung in XGobi (eigener Entwurf).

Mit PL/pgSQL und PL/R wurde eine Funktion entwickelt, die eine räumliche Selektion der Laserpunkte wiederum mit einer Profillinie und einer angegebenen Toleranz durchführt. Ferner können die Laserpunkte mit einer bestehenden Polygongeometrie räumlich ausgewählt werden. Im Gegensatz zur Profillinie werden die x-, y- und z-Koordinaten der Punkte und ausgewählte Attribute in eine ASCII-Datei geschrieben.

```
profile3d_txt(x-Koordinate des Startpunkts::FLOAT, y-Koordinate des
Startpunkts::FLOAT, x-Koordinate des Endpunkts::FLOAT, y-Koordinate des
Endpunkts::FLOAT, Toleranz [m]::FLOAT)
```

Ausgabe der Laserpunkte (x, y, z), die in einem definierten Streifen liegen, in eine ASCII-Datei.

```
poly3d_txt(Polygon::GEOMETRIE, Toleranz [m]::FLOAT)
```

Ausgabe der Laserpunkte (x, y, z), die in einem definierten Polygon liegen, in eine ASCII-Datei.

```
profile3d(x-Koordinate des Startpunkts::FLOAT, y-Koordinate des
Startpunkts::FLOAT, x-Koordinate des Endpunkts::FLOAT, y-Koordinate des
Endpunkts::FLOAT, Toleranz [m]::FLOAT)
```

Räumliche Selektion der Punkte auf einer Profillinie mit entsprechender Toleranz.

```
poly3d(Polygon::GEOMETRIE, Toleranz [m]::FLOAT)
```

Räumliche Selektion der Punkte in einem Polygon mit entsprechender Toleranz.

Die beiden oberen Funktionen (PL/R) greifen auf die beiden unteren Funktionen (PL/pgSQL) zurück und dienen rein dem Export der Punkte in eine Datei.
Die Funktionen *profile3d* und *poly3d* können in jedes SQL-Statement eingebunden werden. So kann man z.B. in R auf die Datenbank zugreifen und dann mit einem der beiden Befehle die 3D-Koordinaten von Punkten abfragen.

9.3.3.2 GGobi

Die Software GGobi ist der Nachfolger von XGobi (www.GGobi.org). Im Gegensatz zu **XGobi**, dessen Benutzeroberfläche auf dem **X**-Windows-System basiert, wird die Oberfläche von **GGobi** mit dem GIMP Toolkit (**G**TK) erstellt (www.gtk.org). Eine hilfreiche Neuerung von GGobi ist die Unterstützung von direkten Datenbankabfragen. Der Umweg des Exports der Daten in eine Datei, die dann in XGobi eingelesen wird, bleibt erspart. In GGobi kann man die Daten mit SQL-Statements (inklusive PostGIS) direkt aus der PostgreSQL-Datenbank importieren. Die oben genannten Funktionen *profile3d* und *poly3d* können bei der Auswahl der Laserpunkte angewendet werden. Die Visualisierungsmöglichkeiten von GGobi unterscheiden sich nicht wesentlich von XGobi (vgl. LANG 2004)

9.4 Graphische Benutzeroberfläche

Die Entwicklung einer graphischen Benutzeroberfläche (G*raphical User Interface – GUI*) nimmt sehr viel Zeit in Anspruch und es muss gut durchdacht vorgegangen werden. Eine Benutzeroberfläche sollte den Anforderungen des Users entsprechen und möglichst alle Anwenderaktionen kontrollieren, d.h. alle Ausnahmen bzw. Fehlaktionen müssen vom Programm abgefangen werden.
Die Benutzeroberfläche, die im Rahmen der Diplomarbeit erstellt wird, beschränkt sich auf einen einfachen Prototypen. Die wichtigsten Elemente sind die Menüleiste und der Statusmonitor im Zentrum des Hauptfensters (Abb. 9.10). Die Menüleiste ist vorgesehen, um alle Funktionen des Informationssystems zu steuern. Im Statusmonitor werden die Meldungen der gestarteten Programme angezeigt und können als Datei abgespeichert werden (vgl. Appendix C).
Die Benutzeroberfläche ist in Python geschrieben und nützt die Funktionen des Python-Moduls Tkinter (LAUER 2002). Python kann direkt auf die PostgreSQL-Datenbank zugreifen und SQL-Statements ausführen. Die Ergebnisse von Datenbankabfragen können von Python weiterverarbeitet werden. Das GUI leitet alle Meldungen, die in die Betriebssystem-Standardausgabe geschrieben werden, an den Statusmonitor weiter. Der User erhält alle Informationen, die von den laufenden Programmen ausgegeben werden (z.B. Fehlermeldungen, Rechenergebnisse, usw.).
Da Python eine plattformunabhängige Programmiersprache ist, kann das GUI auf Linux sowie Windows gestartet werden. Die Benutzeroberfläche wird am Client gestartet und steuert die Arbeitsprozesse, die am Linux-Server (Datenbankserver) ablaufen. Die Auslastung am Client-Rechner ist nur sehr gering. Die „Rechenarbeit“ wird vom Server übernommen. Ein Multi-User-Zugriff ist durch das gute Client-Management von PostgreSQL möglich. Die Performance der einzelnen Prozesse sinkt drastisch, wenn mehrere User gleichzeitig am Server arbeiten und somit die Rechnerkapazität aufgeteilt werden muss. Der Datenimport, bei dem große Datenmengen transferiert werden, kann ausschließlich direkt am Server erfolgen.

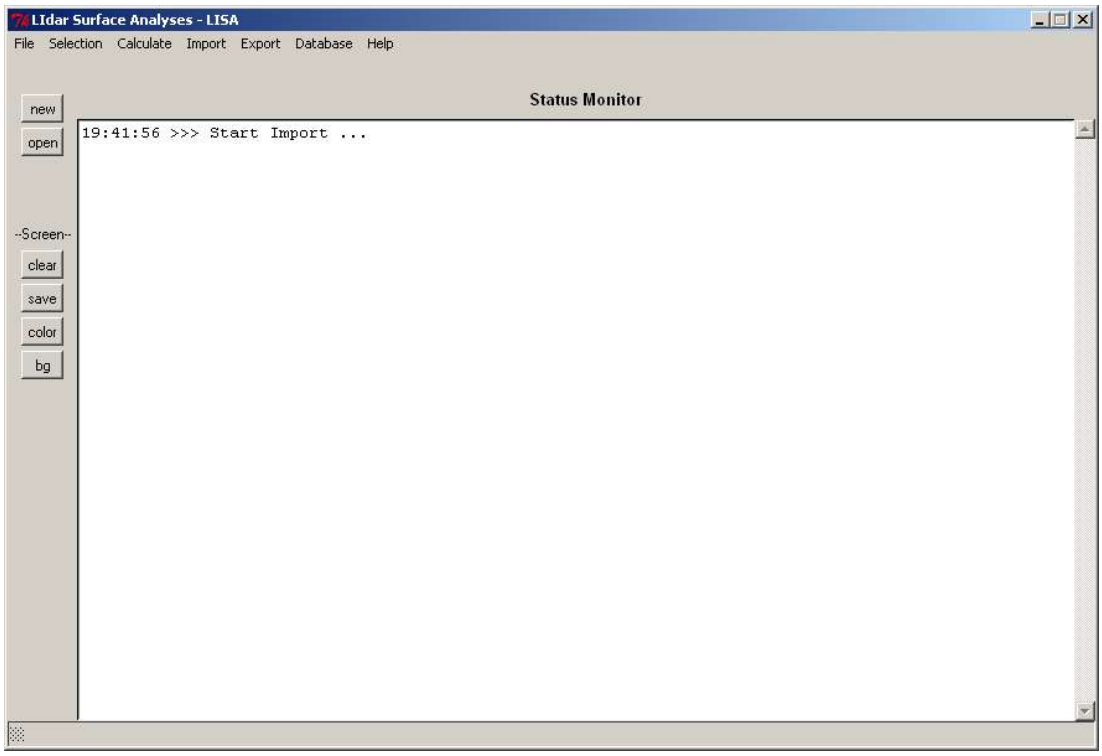


Abb. 9.10: Hauptfenster der graphischen Benutzeroberfläche des Informationssystems (eigener Entwurf).

Die Funktion des Datenimports ist vollständig im Prototypen umgesetzt (Abb. 9.11). Die Rohdaten des Untersuchungsgebietes wurden mit der Benutzeroberfläche importiert und die dabei ausgegebenen Meldungen im Statusmonitor in eine Log-Datei gespeichert (vgl. Appendix C).

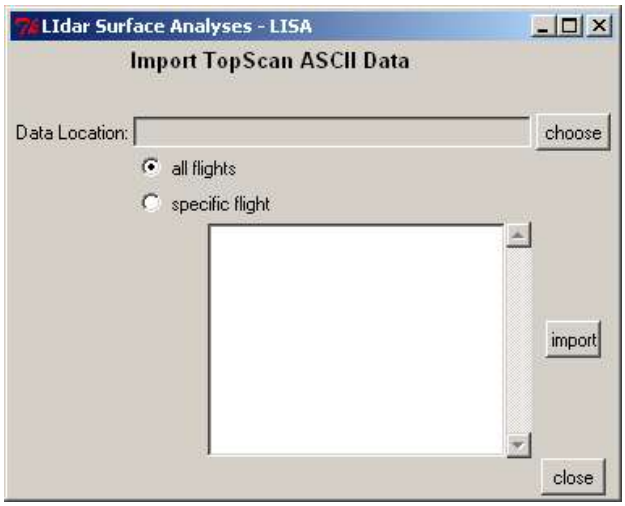


Abb. 9.11: Fenster für den Datenimport der Rohdaten (eigener Entwurf).

9.5 Zusammenfassung

Die räumliche Datenbank, die die Laserscannerdaten speichert und verwaltet, steht im Zentrum des Datenexports. Der direkte Zugriff vieler *open source* Softwarepakte auf die Geometriedaten der PostgreSQL/PostGIS-Datenbank löst das Problem des Austauschformats und vermeidet Fehler, die beim Datenexport aus der Datenbank bzw. beim Datenimport in eine externe Software entstehen können. Ein direkter Zugriff verringert die Anzahl der Rechenschritte und somit die Rechenzeit. Der Export der Geometriedaten in das ESRI *shapefile* Format, das von den gängigsten Geographischen Informationssystemen gelesen werden kann, bietet eine breite Schnittstelle zu kommerzieller Software. Die Hersteller kommerzieller Software gehen sogar einen Schritt weiter. Dies zeigt die neueste Version von ESRI ArcGIS, die das direkte Lesen, Importieren und Schreiben von PostGIS-Tabellen unterstützt (INTERNET: ArcGIS Data Interoperability).
Die Funktionalität der Statistiksoftware R, eingebaut in die Datenbank, ermöglicht die Visualisierung der räumlichen Daten ohne Datentransfer und mit voller Unterstützung der SQL-Sprache, die unter anderem für die Selektion der Daten zuständig ist. R ist dafür konzipiert große Datenmengen zu verarbeiten. Die Erstellung der Profildiagramme in einem Graphikformat (z.B. JPEG, PDF) erfolgt in wenigen Sekunden, auch bei einer großen Anzahl von Punkten (z.B. 1 Million).
GRASS wurde dazu genutzt, die Attribute der Laserpunkte (z.B. Intensität oder ein Parameter der Scangeometrie) in das Rasterdatenmodell umzuwandeln. Die Rechenzeiten werden dabei in erster Linie durch die Anzahl der Punkte, die Auflösung des Rasters und durch die gewählte Interpolationsmethode beeinflusst. GRASS greift direkt auf die Laserpunkte in der Datenbank zu. Die Attribute der Laserpunkte können in der Datenbank berechnet bzw. geändert werden, auf die dann GRASS bei der Interpolation zurückgreift.
Mit der verbesserten Version von XGobi, die nun GGobi heißt, kann auch die dreidimensionale, interaktive Darstellung der Laserpunkte ohne Datentransfer stattfinden.
Die graphische Benutzeroberfläche für das Informationssystem stellt einen Prototypen dar, der das Potential von Python bei der Entwicklung von graphischen Oberflächen aufzeigen soll. Die wohl aufwändigste Rechenoperation außerhalb der Datenbank, der Datenimport, ist bereits voll in die Oberfläche implementiert.

10 Evaluierung und Ausblick

10.1 Diskussion

Die Diskussion wird anhand von drei Aspekten, die diese Arbeit aus verschiedenen Blickwinkeln betrachten, durchgeführt werden. Die Ergebnisse der Diplomarbeit können aus einer *technischen*, einer *inhaltlichen* und einer *wirtschaftlichen Betrachtungsweise* diskutiert werden.

Zu Beginn der Entwicklung des Informationssystems stand die Ermittlung des *State-of-the-Art* bereits bestehender kommerzieller und nichtkommerzieller Systeme für die Verwaltung von Laserscannerdaten. Es gibt kommerzielle Software, die sich auf unterschiedliche Bereiche der Bearbeitung von Laserscannerdaten spezialisiert haben, wie zum Beispiel das Softwarepaket SCOP++ (INTERNET: SCOP++), das sich auf die Erstellung von Rasterdaten (v.a. Digitale Gelände- und Oberflächenmodelle) konzentriert, oder die Software Terrasolid (INTERNET: Terrasolid), die Geländemodelle erstellen und Punktwolken dreidimensional darstellen kann. Kommerzielle Software sieht nicht vor, den Quelltext, der für das Verständnis und eventuelle Modifikationen der Arbeitsabläufe hilfreich wäre, offen zu legen. Ferner ist die Entwicklung meist auf nur *eine* Plattform (hauptsächlich Microsoft Windows) beschränkt. Der Schritt ein eigenes System zu entwerfen, das einen plattformunabhängigen Zugriff erlaubt, wurde durch die Erfahrungen mit *open source* Software während des Studiums erleichtert. Die Kombination und Zusammenführung der drei Hauptkomponenten des Informationssystems – die räumliche Datenbank, das Geographische Informationssystem und die Statistiksoftware – für die Anwendung mit Laserscannerdaten ist absolutes Neuland. Das Ergebnis dieser Diplomarbeit sollte nicht als Entwicklung einer eigenständigen Software verstanden werden. Viel mehr sollte gezeigt werden, dass mit bereits bestehenden, *open source* Programmen eine Verwaltung, Verarbeitung und Visualisierung von großen Datenmengen möglich ist.

Erst die Verbesserung der Leistung der Computer in den letzten Jahren macht die Verarbeitung von Laserscannerdaten im Vektordatenmodell möglich. Die zwei Hauptdiskussionspunkte bleiben jedoch immer noch die **Performance** (die Rechenkapazität pro Zeit) und der **Speicherplatz**, der durch die damit verbundenen hohen Kosten limitiert bleibt. Die Untersuchungen der Diplomarbeit haben ergeben, dass die Komponenten des Informationssystems sehr wohl in der Lage sind, die Datenmengen von großen Befliegungskampagnen (über eine Milliarde Punkte) zu handhaben. Die beste Logik eines Algorithmus reicht nicht aus, wenn die Hardware nicht den Erfordernissen entspricht. Mit gezieltem Softwaretuning des Betriebssystems (vgl. INTERNET: Fedora Softwaretuning) und einer verbesserten Einstellung der einzelnen Systemkomponenten (v.a. Datenbanktuning) kann die Performance effektiv gesteigert werden. Die neu geschriebenen

Funktionen (vgl. *Kapitel „7 Datenimport“*, Kapitel *„8 Datenverarbeitung“*) wurden zahlreichen Tests unterzogen und optimiert, um eine möglichst kurze Berechnungszeit zu erreichen. Ein Vergleich der Performance mit veränderter Hardware konnte nur mit einer Aufrüstung des Arbeitsspeichers durchgeführt werden, da der Zentrale Informatikdienst (ZID) der Universität Innsbruck im Zeitraum der Diplomarbeit keinen Großrechner für umfassende Performancetests zur Verfügung stellen konnte. Diese Erweiterung des Arbeitsspeichers führte zu einer drastischen Verkürzung der Rechenzeiten (Aufstockung des Arbeitsspeichers von 512 MB auf 1024 MB: ca. 60 % der ursprünglichen Importzeit). Die in Kapitel *„8 Datenverarbeitung“* vorgelegten Rechenzeiten zeigen recht deutlich, dass die Analyse der Laserpunkte im Untersuchungsgebiet (ca. 3.8 Millionen Laserpunkte) mit der vorhandenen Hardware (vgl. Kapitel *„5 Entwicklung des Informationssystems“*) zufriedenstellend ablaufen kann. Durch die Speicherung der räumlichen Daten in einer Datenbank, die erst durch den Aufbau von Indices auf die relevanten Informationen sinnvoll einsetzbar ist, führt zu einem erhöhten Speicherplatzverbrauch im Gegensatz zu den Rohdaten. Es muss ein Kompromiss zwischen schnellem Zugriff (bzw. Abfrage) auf die Daten und einem *„Overload“* an Datenmenge gefunden werden. Die aktuelle Version von PostGIS (Version 0.9), das die räumlichen Daten in der Datenbank verwaltet, ist nicht für eine große Anzahl von Punktvektoren ausgerichtet. So wird zum Beispiel für jeden Punkt ein Rechteck (*bounding box*), das die Ausmaße des Objektes angibt, abgespeichert. Das Rechteck wird vom räumlichen Index bei der Suche herangezogen. Dieses Vorgehen ist vor allem für die schnelle räumliche Selektion von Polygongeometrien gedacht. In der weltweiten Diskussionsgruppe von PostGIS (INTERNET: PostGIS Mailing List) wird bereits an einer *Light Weight Geometry* (LWGEOM) gearbeitet, die das unnötige Abspeichern der *bounding box* für Punkte verhindern soll. Nach dem Import der Rohdaten des Untersuchungsgebietes (ca. 10 GB) beträgt die Größe der Datenbank ca. 60 GB, also das Sechsfache. Ein Teil dieser Vervielfachung ist auf den Aufbau der Relationen zurückzuführen, wie zum Beispiel die Verknüpfung zwischen Flugzeugposition und Flugstreifen, die in den Rohdaten nicht gegeben ist. Ein weiterer Teil wird sicher durch die Indexierung des Zeitattributs für jedes Objekt (Laserpunkt und Flugzeugposition) hervorgerufen. Das vorgelegte Datenbankschema (vgl. Kapitel *„6 Datenverwaltung und -speicherung“*), das zwar den Speicherplatzverbrauch der Laserscannerdaten im Gegensatz zu den Rohdaten erhöht, beinhaltet wesentlich mehr Informationen über das einzelne Objekt, wie es in den Rohdaten der Fall ist (z.B. Flugzeugposition für jeden Laserpunkt).

Zusammenfassend zu den technischen Aspekten soll angemerkt werden, dass die reibungslose Zusammenarbeit der drei Hauptkomponenten des Informationssystems (Datenbank, GIS, Statistiksoftware) ein großes Potential für wissenschaftliche Anwendungen bereitstellt. Die offene Struktur des *Informationssystems* bzw. der *Komponenten des Informationssystems* erlaubt die Entwicklung eigener Funktionalität sowie die Anbindung zusätzlich erforderlicher Programme, die dem *open source* Standard entsprechen (vgl. Appendix G). Das Informationssystem wurde für Laserscannerdaten entwickelt, ungeachtet von welcher Befliegungsfirma oder von welchem System (z.B. terrestrische Aufnahme oder System mit mehr als zwei Pulsen) die Daten stammen. Das Grundcharakteristikum, dass ein

Laserpunkt eine zeitliche und eine räumliche Verortung sowie zusätzliche Attribute besitzt, ist allgemein gültig.

Inhaltlich konzentriert sich die Diplomarbeit auf die Rekonstruktion der Scangeometrie, die Ableitung von statistischen Kennwerten aus den Attributen der Laserpunkte und auf die Visualisierung der Punktwolke im Vektordatenmodell sowie die Visualisierung des Intensitätssignals im Rasterdatenmodell. Diese drei Bereiche können nicht isoliert betrachtet werden, sondern gehen stark ineinander über. Die Rekonstruktion der Scangeometrieparameter mit Hilfe der Modellierung einer *Ebene*, die aus drei benachbarten Laserpunkten abgeleitet wird, kann als Basis für die Interpretation des Intensitätssignals angesehen werden. Die Intensität der Reflexion eines Laserimpulses wird unter anderem durch die Scangeometrie (v.a. Länge des Laservektors und Einfallswinkel) modifiziert. Die statistischen Methoden, die in der Diplomarbeit Verwendung finden (deskriptive Statistik, Zusammenhangsmaße und graphische Darstellung der Variablen) bieten die Möglichkeit der Überprüfung der berechneten Scangeometrieparameter sowie die Möglichkeit der Analyse des Zusammenhangs zwischen Intensität und Scangeometrie.
Die Berechnung der Scangeometrie ist im Vektordatenmodell mit einfachen mathematischen Funktionen umsetzbar. Die Einbeziehung von räumlichen Suchkriterien steuert im Wesentlichen die Rechenzeit. Bei der Berechnung der Ebenengleichung wird der Großteil der Zeit von der Suche nach Nachbarpunkten, und nicht von der Ableitung des Normalvektors, eingenommen. Je größer der Suchradius gewählt wird, desto mehr Punkte müssen verarbeitet werden. Die gewählte Methode der Geländemodellierung ist leicht nachvollziehbar. Inwiefern die Ergebnisse der Realität entsprechen, kann nicht gesagt werden. Zuerst müsste geklärt werden, mit was die Ergebnisse verglichen werden sollen. Eine Rekonstruktion von komplexen Oberflächenstrukturen, wie sie zum Beispiel die Vegetation darstellt, ist mit dieser Methode nicht gegeben. Sehr wohl können einfache Oberflächen, wie zum Beispiel ein Hausdach oder eine Straße, mit diesem einfachen Ansatz korrekt abgebildet werden. Eine Modellierung der realen Welt, die der Lichtkegel beim Auftreffen auf die Erdoberfläche vorfindet, führt bei genaueren Überlegungen immer zu mehr „Problemfällen“ als zu Regelmäßigkeiten. Ein Ansatz, der einige dieser „Problemfälle“ abfängt und in das Modell übernimmt, kann nur im sehr großen Maßstab für wenige Laserpunkte beispielhaft umgesetzt werden (vgl. PFEIFER & WINTERHALDER 2004, die Baumstammobjekte aus den Laserpunkten ableiten). Der Vergleich der modellierten Scangeometrie mit den Ergebnissen einer rasterbasierten Rekonstruktion der Scangeometrie (vgl. LUTZ 2003) ist nicht möglich. Im vorliegenden vektorbasierten Modell wird nicht zwischen Boden- und Objektpunkt unterschieden. Es wird vom Prinzip der minimalen räumlichen Distanz ausgegangen. In der unmittelbaren Umgebung eines Bodenpunkts werden sich wahrscheinlicher ebenso Bodenpunkte wie Objektpunkte finden lassen, die dann zur Konstruktion einer Ebene, die den Boden bzw. das Gelände repräsentiert, herangezogen werden. Vergleichbar kann es bei Objektpunkten, zum Beispiel einem Punkt im Kronendach eines Baumes, angenommen werden, der viel wahrscheinlicher Baumpunkte wie Bodenpunkte in seiner nächsten Umgebung hat. Die Interpolation von

Rasterdaten aus den Attributen der Scangeometrie (z.B. Neigung) liefert nur dann ein visuell ansprechendes Bild, wenn Bodenpunkte und Objektpunkte getrennt werden. Werden alle Laserpunkte für die Interpolation verwendet, erhält man ein stark rauschendes Bild, das die „Geländeebenen“ mit den „Objektebenen“ vermischt. Die Analyse aller Laserpunkte des Untersuchungsgebiets mit dem 3D-Suchkriterium weist ca. 20 % aller Punkte aus, weniger als zwei Nachbarpunkte in einer räumlichen Distanz von 0.5 m zu haben. Der visuelle Vergleich mit dem Infrarotluftbild hat ergeben, dass es sich größtenteils um Punkte in der Vegetation oder um Punkte an Objektkanten (z.B. Dachkante) handelt. In weiteren Untersuchungen müsste eine Verifikation dieser Vermutung folgen. Können mit dieser Methode schon 20 % aller Punkte mit rein geometrischen Kriterien (x, y ,z) als Objektpunkte klassifiziert werden? (vgl. Kapitel *„8 Datenverarbeitung*“).
Die Länge des Laservektors (*range*) ist ein wesentlicher Bestandteil der Scangeometrie. Die Entfernung zwischen Flugzeug und Laserpunkt wird nicht von der Modellierung der Geländeebene beeinflusst. Die restlichen Scangeometrieparameter (Neigung, Exposition, Einfallswinkel, Fläche des Lichtkegels am Boden) sind Ableitungen der konstruierten Ebene. Diese Ableitungen sind immer nur so gut wie ihr Ausgangsprodukt. Eine Ebene, die das Gelände einer Straße recht gut wiedergibt, führt zu guten Ergebnissen der daraus abgeleiteten Parameter der Scangeometrie. Die Exposition, die zum Beispiel aus einer Ebene, die in einem Baumkronendach liegt, berechnet wird, muss sehr kritisch interpretiert werden (vgl. „Problemfall“). Wie oben erwähnt, wird der *range* unabhängig vom Gelände berechnet. Die Korrelationskoeffizienten zwischen *range* und Intensität aller Laserpunkte des Untersuchungsgebietes ($r = -0.42$) bzw. für ausgewählte Befliegungen (Befliegung ID=1→ $r = 0.14$; Befliegung ID=2→ $r = 0.07$) geben keinen eindeutigen Zusammenhang an. Begründet werden kann dies mit dem starken Einfluss der Reflexionseigenschaften der Oberfläche auf die Intensität. Die Korrelationsanalyse muss auf Bereiche mit einer homogenen Oberflächenstruktur beschränkt werden, damit der Einfluss der Oberfläche verringert wird (vgl. LUTZ 2003). Die Regression von Intensität (abhängige Variable) nach *range* (unabhängig) zeigt die Abnahme der Intensität mit zunehmender Entfernung recht deutlich auf ($y = 108.58 – 0.05 x$).
Die Punktdichte im Untersuchungsgebiet ist mit über 15 Punkten pro Quadratmeter ausgesprochen hoch. Ein Vorteil der räumlichen Datenbank ist es, dass die Punkte immer in der höchsten Punktdichte angesprochen werden können, aus denen dann angepasst an die jeweilige Fragestellung (z.B. Erstellung einer Waldmaske) bestimmte Attribute in geringerer Auflösung abgeleitet werden können. Die Arbeit mit ungefilterten Punkten erhöht den Rechenaufwand und führt vielleicht zu keinen signifikant besseren Ergebnissen. Die beste räumliche Genauigkeit (Vektordatenmodell) und die beste räumliche Auflösung (höchste Punktdichte) sollte aber immer die Ausgangsbasis für Analysen in der Punktwolke sein. Im Kapitel *„8 Datenverarbeitung*“ wurde ein theoretischer Ansatz für die Erstellung von Rasterdaten direkt in der Datenbank andiskutiert. Ein Vorteil dieses Ansatzes ist es, dass große Datenmengen, ohne in „Kacheln“ (z.B. 5 x 5 km) zu unterteilt werden und ohne die Probleme an den Blattschnitten, in Rasterdaten überführt werden können.
Die Möglichkeit für multitemporale Analysen wurde mit der absoluten zeitlichen Verortung

geschafften. Die Datensätze des Landes Vorarlberg wurden zwar in mehreren Befliegungskampagnen aufgenommen, aber zeitlich unmittelbar hintereinander (z.B. 1 Tag liegt zwischen den zwei Befliegungen des Untersuchungsgebiets). Somit waren keine multitemporalen Analysen, die große Veränderungen in der Punktwolke nachweisen können, möglich. Das Institut für Geographie der Universität Innsbruck besitzt über zehn zeitlich verschiedene Datensätze des Gletschers *Hintereisferner* (Tirol), der durch Akkumulations- und Ablationsprozesse einem ständigen Wandel unterliegt. Die Überführung dieser Daten in ein Format, das den Spezifikationen des Importskripts entspricht, muss angestrebt werden. Ein Gletscher, dessen Oberfläche mit dem Gelände gleichzusetzen ist, da fast keine Objekte (z.B. Vegetation, Häuser) zu finden sind, bietet die ideale Testfläche für Algorithmen. Es kann vermutet werden, dass mit dem 3D-Suchkriterium Punkte an Gletscherspalten und scharfen Geländekanten identifiziert werden können. Die multitemporale Analyse dieser identifizierten Objekte (Erzeugung von Objektlinien oder -flächen) könnte Aufschluss über die Bewegung des Gletschers geben.

Der letzte Teil der Diskussion befasst sich mit der **wirtschaftlichen Betrachtungsweise**, die nur sehr allgemein behandelt werden kann. Die Ausführungen der Diplomarbeit sollten gezeigt haben, dass die Verarbeitung von geographischen Daten mit *„gratis“* Software nicht unbedingt schlechter als mit *„kommerzieller“* Software sein muss, sondern sogar mehr Möglichkeiten bietet, die beim Einsatz im wissenschaftlichen Arbeitsbereich an Bedeutung gewinnen. Gerade die finanziellen Kürzungen für Wissenschaft und Forschung erfordern es, Alternativen zu suchen. Alternativen müssen nicht immer ein Kompromiss sein, sondern können den Horizont erweitern. Die Zusammensetzung des *Informationssystems* mit kommerziellen Komponenten (z.B. Oracle Spatial Datenbank, ESRI ArcGIS mit ArcSDE, S PLUS Statistiksoftware) würde Kosten in der Höhe mehrerer Zehntausend Euro verursachen. Die gewählten *open source* Softwareprodukte können in ihrem Aufgabengebiet leicht mit kommerzieller Software mithalten. Die ständige Weiterentwicklung der Software durch die Zusammenarbeit einer weltweiten Entwicklergemeinde aus Programmierern und Wissenschaftlern gewährleistet eine fortschreitende Entwicklung der Funktionalität und Performance dieser Software. Die Internet *mailing groups* stellen eine große Hilfe (gratis Support) bei der Arbeit mit dieser Software dar. Für jede Problemstellung – von den Anfangsschwierigkeiten bis zu einem spezifischen technischen Problem – findet sich in der Regel binnen eines Tages eine Lösung.
Die Inwertsetzung der Rohdaten (bzw. der Punktwolke), die in dieser Diplomarbeit ausführlich thematisiert (und visualisiert) wurde, bringt einen Mehrwert für die meisten Kunden von Laserscannerbefliegungen, die bis jetzt ausschließlich mit den abgeleiteten Digitalen Gelände- und Oberflächenmodellen gearbeitet haben.

10.2 Ausblick

Folgende Fragestellungen ergeben sich aus der Diskussion der Ergebnisse der Diplomarbeit *„Entwicklung eines Informationssystems für Laserscannerdaten mit open source Software“*:

- Das *Informationssystem* läuft bisher auf einem Desktop-Computer. Wie wirkt sich eine optimale Hardware (leistungsfähiger Linux-Server) auf die Rechenzeiten der Funktionen aus?
- Die Datenbank PostgreSQL und das Tool PostGIS werden demnächst in einer neuen Version (PostgreSQL 8 und PostGIS 1.0) erscheinen. Welche neue Funktionalität kann in das Informationssystem eingebaut werden?
- Das Importskript könnte dahingehend verbessert werden, dass schon beim Import Punkte, die nicht im geforderten räumlichen Ausschnitt liegen, weggelassen werden. Dadurch könnte die Importzeit und vor allem der Speicherplatzverbrauch reduziert werden.
- Die automatische Generierung von Profillinien in einem bestimmten Abstand über ein gesamtes Gebiet könnte dazu benützt werden, eine Art „Tomographie“ der Punktwolke zu erstellen. Die einzelnen Profile aneinander gereiht, ergeben einen Film, der die Bewegung durch die Punktwolke simuliert.
- Können die statistischen Auswerte-Tools dazu beitragen, ein empirisches Modell für die geometrische Korrektur des Intensitätssignals zu entwickeln?
- Geostatistische Analysen, die Analyse der Verteilung der Punkte im Raum (x, y, z), wurden bisher ausgeklammert, da der Rechenaufwand mit der Anzahl der Punkte zu groß wäre. Gibt es einfache geostatistische Verfahren, die es erlauben, die Datenmenge des Laserscannings zu verarbeiten und welchen Mehrwert bringen sie?
- Die Segmentierung und Klassifikation in der Punktwolke verdrängt immer mehr die klassischen Analysen im Rasterdatenmodell. Können Segmentierungsalgorithmen auf die Punktwolke übertragen werden? Bleibt der Rechenaufwand für große Untersuchungsgebiete (z.B. 1 km^2) annehmbar?
- Die Analysen in der Punktwolke, die Herangehensweise an die Entwicklung eines Algorithmus, hängt in erster Linie vom Maßstab der Fragestellung ab. Mit welchen Filtermethoden bzw. wie stark können die Punktdaten reduziert werden ohne einen signifikanten Genauigkeitsverlust zu verursachen?
- Die zusätzliche Anbindung von optischen Informationen (z.B. Luftbild: RGB-Werte) oder von Ergebnissen einer Fernerkundungssoftware als Attribute an die Laserpunkte könnte die Klassifizierung der Laserpunkte anhand rein geometrischer Kriterien ergänzen bzw. evaluieren.
- Die Laserpunkte werden in der Vorverarbeitung bei der Befliegungsfirma in Boden- und Nichtbodenpunkte klassifiziert, da hauptsächlich die gerasterten Gelände- und Oberflächenmodellen von den Kunden gefordert werden. Eine zusätzliche Anbindung dieser Information als Attribut an die Punkte, würde die Geländemodellerstellung im

Informationssystem ermöglichen. Die räumlichen Selektionsmöglichkeiten der Datenbank könnten dazu genutzt werden, Geländemodelle für unterschiedliche Maßstabsbereiche durch Ausschöpfung aller Punktdaten zu erstellen.

- Im Vordergrund der Forschung steht die Entwicklung neuer Methoden und Strategien für die Analyse von Laserscannerdaten. In Zusammenarbeit mit einer IT-Firma könnte der Prototyp des Informationssystems in Form einer Software realisiert werden.

Literaturverzeichnis

Ackermann, F., 1999. Airborne laser scanning – Present status and future expectations. -*in*: ISPRS Journal of Photogrammetry and Remote Sensing, 54, 64-67.

Akel, N.A., Zilberstein, O., Doytsher, Y., 2003. Automatic DTM Extraction from Dense Raw LIDAR Data in Urban Areas. -*in*: Proceedings of the FIG (International Federation of Surveyors) Working Week 2003, Paris, 1-10.

Bahrenberg, G., Giese, E., 1975. Statistische Methoden und ihre Anwendung in der Geographie. Stuttgart.

Baltsavias, E.P., 1999. Airborne laser scanning: basic relations and formulas. -*in*: ISPRS Journal of Photogrammetry and Remote Sensing, 54, 199-214.

Baltsavias, E.P., 1999. Airborne laser scanning: existing systems and firms and other resources. -*in*: ISPRS Journal of Photogrammetry and Remote Sensing, 54, 164-198.

Boenigk, C., 2003. PostgreSQL. Grundlagen, Praxis, Anwendungsentwicklung mit PHP. Heidelberg.

Briese, Ch., Belada, P., Pfeifer, N., 2001. Digitale Geländemodelle im Stadtgebiet aus Laser- Scanner-Daten. -*in*: Österreichische Zeitschrift für Vermessung & Geoinformation, 89. Jahrgang, Heft 2, 83-91.

Brovelli, M.A., Cannata, M., Longoni, U.M., 2002. Managing and processing LIDAR data within GRASS. -*in*: Proceedings of the GRASS Users Conference 2002, Trento, 1-29.

Bürger, H., Unfried, H., Haschkovitz, F., 1997. Mathematische Formelsammlung. Wien.

Burman, H., 2002. Laser Strip Adjustment for Data Calibration and Verification. -*in*: International Archives of Photogrammetry and Remote Sensing, 34(3A), Graz, 67-72.

Dassau, O., Holl, S., Neteler, M., Redslob, M., 2004. Eine Einführung in den praktischen Umgang mit dem Freien Geographischen Informationssystem GRASS 5.7. *Internet*: http://www.gdf-hannover.de/dl.php?download=gdf_grass57_v10.pdf

Dolić, D., 2004. Statistik mit R. Einführung für Wirtschafts- und Sozialwissenschaftler. München-Wien.

Douglas, D.H., Peucker, T.K., 1973. Algorithms for the reduction of the number of points required to represent digitized line or its charicature. -*in*: The Canadian Cartographer, 10, 112-122.

Echelmeyer, K.A., Harrision, W.D., Larsen, C.F., Sapiano, J., Mitchell, J.E., DeMallie, J., Rabus, B., Aðalgeirsdóttir, G., Sombardier, L., 1996. Airborne surface profiling of glaciers: a case-study in Alaska. -*in*: Journal of Glaciology, 42(142), 538-547.

Eisentraut, P., 2003. PostgreSQL – Das offizielle Handbuch. Bonn.

Eklundh, L. (Hg.), Arnberg, W., Arnborg, S., Harrie, L., Hauska, H., Olsson, L., Pilesjö, P., Rystedt, B., Sandgren, U., 2001. Geografisk informationsbehandling – metoder och tillämpningar. Stockholm.

Favey, E., 2001. Investigation and Improvement of Airborne Laser Scanning Technique for Monitoring Surface Elevation Changes of Glaciers. Dissertation ETH Nr. 14045.

Filin, S., 2003. Analysis and Implementation of a laser strip adjustment model. *-in*: International Archives of Photogrammetry and Remote Sensing, 34(3/W13), Dresden, 65-70.

Geist, Th., Lutz, E., Stötter, H., 2003. Airborne Laser Scannning Technology and its Potential for Applications in Glaciology. *-in*: International Archives of Photogrammetry and Remote Sensing, 34(3/W13), Dresden, 101-106.

Geschwinde, E., Schönig, H.J., 2002a. Datenbank-Anwendungen mit PostgreSQL. Einführung in die Programmierung mit SQL, Java, C/C++, Perl, PHP und Delphi. München.

Geschwinde, E., Schönig, H.J., 2002b. PostgreSQL Developer's Handbook. Indianapolis.

Hyyppä, J., Hyyppä, H., Litkey, P., Yu, X., Haggrén, H., Rönnholm, P., Pyysalo, U., Pitkänen, J., Maltamo, M., 2004. Algorithms and Methods of Airborne Laser Scanning for Forest Measurements. *-in*: International Archives of Photogrammetry and Remote Sensing, 36(8/W2), Freiburg, 82-89.

Jeckle, M, Rupp, Ch., Hahn, J., Zengler, B., Queins, S., 2003. UML 2 glasklar. München. *Internet*: http://www.jeckle.de/uml-glasklar/UseCaseDiagramm.pdf

Jurgeit, F., 2003. Rauminformationssystem mit PostgreSQL und SVG am Beispiel des Prototypen „UniRIS“. Unveröffentlichte Diplomarbeit, Institut für Geographie, Universität Innsbruck.

Katzenbeißer, R., 2003. Technical-Note on Echo Detection. *Internet*: www.toposys.com (public download area).

Kleinschmidt, P., Rank, C., 2005. Relationale Datenbanksysteme. Heidelberg.

Kraus, K., 2004. Photogrammetrie – Geometrische Informationen aus Photographien und Laserscanneraufnahmen. Bd. 1, Berlin.

Kraus, K., Pfeifer, N., 1998. Determination of Terrain Models in Wooded Areas with Airborne Laser Scanner Data. *-in*: ISPRS Journal of Photogrammetry and Remote Sensing, 53(4),193-203.

Lang, D.T., 2004. GGobi and Database Management Systems from GGobi Reading data from MySQL & Postgres. *Internet*: http://www.ggobi.org/DBMS.pdf

Lauer, M., 2002. Python und GUI-Toolkits. Bonn.

Ligges, U., 2005. Programmieren mit R. Berlin.

Lindenberger, J., 1993. Laser-Profilmessung zur topographischen Geländeaufnahme. Deutsche Geodätische Kommission bei der Bayerischen Akademie der Wissenschaften, Reihe C, Nr. 400, München.

Lundh, F., 1999. An Introduction to Tkinter. *Internet*: http://www.pythonware.com/library/tkinter/an-introduction-to-tkinter.pdf

Lutz, E.R., 2003. Investigations of Airborne Laser Scanning Signal Intensity on Glacial Surfaces – Utilizing Comprehensive Laser Geometry Modeling and Surface Type Modeling (a Case Study: Svartisheibreen, Norway). Unveröffentlichte Diplomarbeit, Institut für Geographie, Universität Innsbruck.

Maas, H., 2001. On the Use of Pulse Reflectance Data for Laserscanner Strip Adjustment. -*in*: International Archives of Photogrammetry and Remote Sensing, 34(3W/4), Annapolis/Maryland.

Neteler, M., Mitasova, H., 2002. Open Source GIS: A GRASS GIS Approach. 1. Auflage, Boston, Dordrecht.

Neteler, M., Mitasova, H., 2004. Open Source GIS: A GRASS GIS Approach. 2. Auflage, Boston, Dordrecht.

OpenGIS Consortium Inc. (OGC), 1999. OpenGIS Simple Features Specification For SQL. Revision 1.1. *Internet*: http://www.opengeospatial.org/docs/99-049.pdf

Persson, Å., Holmgren, J., Söderman, U., Olsson, H., 2004. Tree Species Classification of Individual Trees in Sweden by Combining High Resolution Laser Data with High Resolution Near-Infrared Digital Images. -*in*: International Archives of Photogrammetry and Remote Sensing, 36(8/W2), Freiburg, 204-206.

Pfeifer, N., Winterhalder, D., 2004. Modelling of Tree Cross Sections from Terrestrial Laser Scanning Data with Free-Form Curves. -*in*: International Archives of Photogrammetry and Remote Sensing, 36(8/W2), Freiburg, 76-81.

Ramsey, P., 2004. PostGIS 0.9 Manual. *Internet*: http://postgis.refractions.net/docs/postgis.pdf

Rutzinger, M., 2005. Identifikation und Klassifikation von Objekten und Oberflächeneigenschaften aus Laserscannerdaten – Auswertestrategien mit eCognition für forstliche Fragestellungen. Unveröffentlichte Diplomarbeit, Institut für Geographie, Universität Innsbruck.

Scherzinger, B.M., 1993. A Report on Navigation Systems Integration and on the Benefits and Costs of Integration. Unpublished report of the Applied Analytics Corporation, Markham, Ontario, Canada. Prepared for the Directorate of Avionics, Systems and Photography, Department of National Defense of Canada.

Swayne, D.F., Cook, D., Buja, A., 1998. XGobi: Interactive Dynamic Data Visualization in the X Windows System. *-in*: Journal of Computational and Graphical Statistics, 7(1), 1-20. *Internet*: http://www.research.att.com/areas/stat/xgobi/papers/miss.ps.gz

Tipler P.A., Mosca, G., 2004. Physik – Für Wissenschaftler und Ingenieure. München.

Topscan GmbH, 2004. Unveröffentlichter Projektbericht bezüglich der Laserscannermessung Unterland und Vorderwald im Auftrag des Landesvermessungsamtes Feldkirch.

Venables, W.N., Smith, D.M., 2004. An Introduction to R. *Internet*: http://www.r-project.org

Vosselman, G., Gorte, B.G.H., Sithole, G., Rabbani, T. 2004. Recognising Structure in Laser Scanner Point Clouds. *-in*: International Archives of Photogrammetry and Remote Sensing, 36(8/W2), Freiburg, 33-38.

Wagner, W., A. Ullrich, A., Melzer, T., Briese, C., Kraus, K., 2004. From Single-pulse to Full-waveform Airborne Laser Scanners: Potential and Practical Challenges. *-in*: International Archives of Photogrammetry and Remote Sensing, 25(B3), Istanbul.

Wagner, W., Ullrich, A., Briese, C., 2003. Der Laserstrahl und seine Interaktion mit der Erdoberfläche. *-in*: Österreichische Zeitschrift für Vermessung & Geoinformation, 4, 223-235.

Wehr, A., Lohr, U., 1999. Airborne laser scanning – an introduction and overview. *-in*: ISPRS Journal of Photogrammetry and Remote Sensing, 54, 68-82.

Weigend, M., 2003. Python GE-PACKT. Bonn.

Weigend, M., 2004. Objektorientierte Programmierung mit Python. Bonn.

Wever, Ch., 1999. Laserscannermessungen – ein Verfahren setzt sich durch. *-in*: Geo-Informationssysteme: Zeitschrift für raumbezogene Informationen und Entscheidungen, 12, 12-16.

Wever, Ch., Lindenberger, J., 1999. Laser Scanning – A Mapping Method Gains Ground. *-in*: Photogrammetrische Woche 1999, 125-132.

Wheeler, D.A., 2004. Why Open Source Software/Free Software (OSS/FS)? Look at the Numbers! *Internet*: http://www.dwheeler.com/oss_fs_why.html

Mündliche Mitteilungen

Grosser Martin, TopScan Gmbh. (Steinfurt, D). Telefonische Konsultation zu den Metadateninformationen der Befliegungen (v.a. IMU). 22.07.2004.

Lindenberger Joachim, TopScan Gmbh. (Steinfurt, D). Besprechung der vorhandenen Befliegungsdaten und Anforderungen an das Informationssystem. Innsbruck, 10.08.2004.

Lindenberger Joachim, TopScan Gmbh. (Steinfurt, D). Telefonische Konsultation zur Festlegung eines Standardformats für Laserscannerdaten. 13.08.2004.

Internetressourcen (Stand 01/2005)

ArcGIS Data Interoperability: http://www.esri.com/software/arcgis/extensions/datainteroperability/about/supported-formats.pdf
AsSVG: http://www.svg.cc
BASF & PostgreSQL: http://www.postgresql.org/files/about/casestudies/wcgcasestudyonpostgresqlv1.2.pdf
BASF: http://www.basf.com
CRAN (R Pakete): http://cran.r-project.org
CWI: http://www.cwi.nl
Die Emsigen – Pressemeldungen 09/2002: http://www.emsige.at/archiv/2002/09geolga.html
ESRI: http://www.esri.com
Features von PostgreSQL: http://www.postgres.de/features.html
Fedora Softwaretuning: http://www.europe.redhat.com/documentation/rhdb/admin_user/kernel-resources.php3
Free Software: http://www.gnu.org/philosophy/free-software-for-freedom.html
FreeGIS: http://www.freegis.org/
Geopage: http://uibk.com
GEOS: http://geos.refractions.net/
GGobi: http://www.ggobi.org
GiST indexing: http://www.sai.msu.su/~megera/postgres/gist
GNU General Public License: http://www.gnu.org/copyleft/gpl.html
GRASS Mailing Lists: http://grass.itc.it/support.html
GRASS Online Manual: http://www.grass.itc.it/grass57/
GRASS: http://grass.itc.it
GTK: http://www.gtk.org
Hohenems: http://www.hohenems.at
IDLE: http://www.python.org/idle/

Institut für Geographie, Universität Innsbruck: http://geowww.uibk.ac.at
Institut für Photogrammetrie und Fernerkundung der TU Wien: http://www.ipf.tuwien.ac.at
ISPRS: http://www.isprs.org
Leica Geosystems: http://www.hds.leica-geosystems.com
LWGEOM: http://postgis.refractions.net/pipermail/postgis-users/2004-February/003999.html
NATSCAN: http://www.natscan.de
OGR Simple Feature Library: http://www.remotesensing.org/gdal/ogr/
Online GRASS Manuals: http://grass.itc.it/gdp/online.html
Open Geospatial Consortium (OGC): http://www.opengeospatial.org/
Open Source: http://www.opensource.org/
OpenGIS Consortium: http://www.opengis.org
OpenGIS Simple Features Specification for SQL: http://www.opengis.org/techno/specs/99-049.pdf
Optech Incorporated: http://www.optech.on.ca
pgAdmin: http://www.pgadmin.org
PL/R Mailing List: http://gborg.postgresql.org/mailman/listinfo/plr-general
PL/R: http://www.joeconway.com/plr/
PostGIS Mailing List: http://postgis.refractions.net/mailman/listinfo/postgis-users
PostGIS: http://postgis.refractions.net.
PostgreSQL Case Studies: http://www.postgresql.org/about/casestudies/
PostgreSQL Features: http://www.postgresql.org/users-lounge/features.html
PostgreSQL Lizenz: http://www.postgresql.org/licence.html
PostgreSQL: http://www.postgresql.org
Proj4: http://www.remotesensing.org/proj oder http://proj.maptools.org
PSF: http://www.python.org/psf/
Python pg-Mod. Dokumentation: http://www.postgresql.org/docs/7.3/interactive/pygresql.html
Python pg-Modul (pygreSQL): http://www.druid.net/pygresql/
Python: http://www.python.org
QGIS: http://qgis.org
R Mailing Lists: http://www.r-project.org/mail.html
R Projekt: http://www.r-project.org
Red Hat: http://www.redhat.com/fedora/
Redhat Fedora: http://fedora.redhat.com
Relationale Datenbanksysteme: http://www.rz.uni-passau.de/db-buch
Riegl Laser Measurement Systems: http://www.riegl.com
SCOP++: http://www.inpho.de/scop.htm
Scriptics (Tk GUI Toolkit): http://www.scriptics.com
SQL FAQ: http://epoch.cs.berkeley.edu:8000/sequoia/dba/montage/FAQ/SQL_TOC.html
SQL Infos: http://www.sql.org/
Tcl/Tk: http://www.tcl.tk/
Terrasolid: http://www.terrasolid.fi
Tkinter: www.python.org/moin/TkInter

TopScan Gmbh.: http://www.topscan.de
UML 2: http://www.uml-glasklar.de
Uni Stuttgart: http://www.ifp.uni-stuttgart.de
Wildbach- und Lawinenverbauung Vorarlberg - Zwischenbilanz Oktober 2003:
http://www.vorarlberg.at/doc/wlv_bericht_pku.doc
XGobi: http://research.att.com/areas/stat/xgobi

Abbildungsverzeichnis

2 Grundlagen des flugzeuggestützten Laserscannings

Abb. 2.1: Schematische Darstellung des flugzeuggestützten Laserscannings.
Abb. 2.2: Mehrfachreflexion (5 Reflexionen) bei Auftreffen eines Laserstrahls bzw. Lichtkegels auf Vegetation.
Abb. 2.3: Unterschied im Messprinzip zwischen *pulsed* und *continuous wave* Lasern.

3 Einführung in das Untersuchungsgebiet

Abb. 3.1: Das Untersuchungsgebiet im Süden der Stadt Hohenems (Vorarlberg).
Abb. 3.2: *Hillshade* des Digitalen Geländemodells (1 m Rasterweite).
Abb. 3.3: *Hillshade* des Digitalen Oberflächenmodells (1 m Rasterweite).
Abb. 3.4: Einfamilienhäuser im Talbereich des Untersuchungsgebietes.
Abb. 3.5: CIR (*coloured infrared*)-Luftbild (0.25 m Rasterweite).
Abb. 3.6: Buchen-Tannenwald.
Abb. 3.7: Geschiebeablagerungsbecken mit Balkensperre am oberen Schwemmkegel des Pelzreutebaches.
Abb. 3.8: Flugstreifenübersicht.

4 Datenerfassung und Datengrundlagen

Abb. 4.1: Modellhafte Darstellung der Streifenbreite.
Abb. 4.2: Vollständiger Scandurchlauf.
Abb. 4.3: Zusammenspiel der System- und Flugparameter mit der Topographie und den Bewegungen des Flugzeugs während der Befliegung und die daraus abgeleiteten Größen.
Abb. 4.4: Orientierung des Flugzeugs im Raum mit den Winkeln *roll*, *pitch* und *heading.*
Abb. 4.5: Aufbau eines typischen ALS-Aufnahmesystems.
Abb. 4.6: Flussdiagramm zur Verarbeitung der Messdaten.
Abb. 4.7: Laufzeitmessung.
Abb. 4.8: Schema der vorliegenden Datenstruktur.

5 Entwicklung des Informationssystems

Abb. 5.1: Komponenten des Informationssystems.
Abb. 5.2: Komponenten eines Computersystems.
Abb. 5.3: Schema einer Datenbank mit zwei Tabellen.
Abb. 5.4: Ein PostgreSQL-Server kann mehrere Datenbanken (=Cluster) verwalten.
Abb. 5.5: Client/Server-Architektur von PostgreSQL.
Abb. 5.6: Einfache und nicht-einfache Linien (s*imple and non-simple linestrings*) laut OGC.
Abb. 5.7: Hierarchie der OGC-Geometrietypen (*Geometry Class Hierarchy*).

6 Datenverwaltung und -speicherung

Abb. 6.1: Entitäten der Laserscannerbefliegung und ihre Beziehungen.
Abb. 6.2: Datenbankstruktur des Informationssystems.
Abb. 6.3: Von PostGIS erstellte Tabellen, die die Geometriespalten einer Datenbank verwalten.

7 Datenimport

Abb. 7.1: *Data Import Workflow.*
Abb. 7.2: *Import Process in Details.*

8 Datenverarbeitung

Abb. 8.1: Weg-Zeit-Diagramme des Flugzeugs in unterschiedlichen Skalen.
Abb. 8.2: Beispiel von 20 interpolierten Flugzeugpositionen.
Abb. 8.3: Die Länge des Laservektors als Entfernung zwischen der Flugzeugposition und dem Laserpunkt.
Abb. 8.4: Konstruktion des Geländes an einem Punkt mit Hilfe einer Ebene.
Abb. 8.5: Infrarot-Luftbild zur Veranschaulichung der hohen Vegetation.
Abb. 8.6: Bereiche, die bei einem 3-D-Suchradius von 0.5 m weniger als 2 Nachbarpunkte haben.
Abb. 8.7: Bereiche, die bei einem 3-D-Suchradius von 1 m weniger als 2 Nachbarpunkte haben.
Abb. 8.8: Bereiche, die bei einem 3-D-Suchradius von 2 m weniger als 2 Nachbarpunkte haben.
Abb. 8.9: Normalvektor $\vec{n}$.
Abb. 8.10: Neigung der Ebene als Winkel zwischen dem Normalvektor und dem Vektor $\vec{v}$ [0, 0, 1], der senkrecht nach oben zeigt.
Abb. 8.11: Exposition der Ebene als horizontale Ausrichtung des Normalvektors.
Abb. 8.12: Einfallswinkel (φ) des Laservektors $\vec{l}$.
Abb. 8.13: Schattenbereiche (Einfallswinkel >= 90°) im Untersuchungsgebiet bei einem 2D-Suchradius von 0.5 m
Abb. 8.14: Der *footprint* wird durch *range* R, Einfallswinkel φ und Laserstrahldivergenz γ festgelegt
Abb. 8.15: Streuung an verschiedenen Objekten.
Abb. 8.16: Die Fläche des Lichtkegels am Boden bei einem Einfallswinkel größer als 0°.
Abb. 8.17: Selektion der Laserpunkte für die Berechnung der relativen Flughöhe mit einem zweidimensionalen Suchradius.
Abb. 8.18: Plot der Intensität (*y*) und der Länge des Laservektors (*x*) für alle Punkte des Untersuchungsgebietes.
Abb. 8.19: Histogramm der Intensität aller Laserpunkte im Untersuchungsgebiet.
Abb. 8.20: Boxplot der Intensität aller Punkte im Untersuchungsgebiet.
Abb. 8.21: Korrelationsdiagramm mit eingezeichneter Regressionsgeraden zwischen Länge des Laservektors (*range*) und relativer Flughöhe.
Abb. 8.22: GRASS-Module zur Umwandlung von Vektor- in Rasterdaten.
Abb. 8.23: Legende für die Intensitätsraster.
Abb. 8.24: Intensitätsraster des Untersuchungsgebietes berechnet mit allen Laserpunkten.
Abb. 8.25: Intensitätsraster des Untersuchungsgebietes berechnet mit den Laserpunkten der Befliegung ID=2.
Abb. 8.26: Modifizierte Legende für die Intensitätsraster der Befliegung ID=1.
Abb. 8.27: ntensitätsraster des Untersuchungsgebietes berechnet mit den Laserpunkten der Befliegung ID=1.

9 Datenexport und Visualisierung

Abb. 9.1: Ausgewählte Exportmöglichkeiten.
Abb. 9.2: Visualisierung der Laserpunkte in QGIS.
Abb. 9.3: Profillinie mit eingezeichnetem Toleranzbereich.
Abb. 9.4: Entfernung des Laserpunkts vom Startpunkt des Profils und Normalabstand zur Profillinie.
Abb. 9.5: Standort des Profils Nr. 1.
Abb. 9.6: Profil 1 mit einer Toleranz von 1 m.
Abb. 9.7: 3D-Scatterplot des Profils 1 mit einer Toleranz von 5 m.
Abb. 9.8: Schematische Darstellung einer Profillinie mit zwei Segmenten.
Abb. 9.9: Screenshot einer 3-D-Visualisierung in XGobi.
Abb. 9.10: Hauptfenster der graphischen Benutzeroberfläche des Informationssystems.
Abb. 9.11: Fenster für den Datenimport der Rohdaten.

Appendix

Abb. D.1: Boxplot der Intensität aller Punkte im Untersuchungsgebiet.
Abb. D.2: Plot mit berechnetem Korrelationskoeffizienten (im Titel) zwischen Intensität und Länge des Laservektors (*range*).
Abb. D.3: Plot zwischen Intensität und *range* mit eingezeichneter Regressionsgeraden.
Abb. D.4: Histogramm der Intensität aller Laserpunkte im Untersuchungsgebiet.
Abb. D.5: Histogramm der Länge des Laservektors für alle Laserpunkte im Untersuchungsgebiet.
Abb. D.6 Histogramme der Länge des Laservektors augeschlüsset nach Befliegungskampagnen.
Abb. D.7: Boxplots der Länge des Laservektors.
Abb. D.8: Histogramm der Neigung.
Abb. D.9: Bereiche, die bei einem 3-D-Suchradius von 0.5 m weniger als 2 Nachbarpunkte haben.
Abb. D.10: Histogramm der Exposition.
Abb. D.11: Histogramm des Einfallswinkels.
Abb. D.12: Boxplot des *footprints* für Werte < 5 m².
Abb. D.13: Histogramm des *footprints* aller Punkte mit einem Wert <5 m².
Abb. D.14: Histogramm der relativen Flughöhe für alle Laserpunkte (2D-Suchradius = 1 m).
Abb. D.15: Regression *von range* (y) nach relativer Flughöhe (x). n=3787446.
Abb. E.1: Standortsübersicht der drei Profillinien im Untersuchungsgebiet.
Abb. E.2: Standort des Profils Nr. 1.
Abb. E.3: Detailansicht des Hauses in Profil 1.
Abb. E.4: Profil 1 mit 0.5 m Toleranz.
Abb. E.5: Profil 1 mit 1 m Toleranz und mit Linien verbundenen Punktsymbolen.
Abb. E.6: Profil 1 mit 2 m Toleranz.
Abb. E.7: Profil 1 mit 5 m Toleranz.
Abb. E.8: Standort des Profils Nr. 2.
Abb. E.9: Profil 2 mit 0.5 m Toleranz.
Abb. E.10: Profil 2 mit 1 m Toleranz.
Abb. E.11: Profil 2 mit 2 m Toleranz.
Abb. E.12: Profil 2 mit 5 m Toleranz.
Abb. E.13: Standort des Profils Nr. 3.
Abb. E.14: Profil 3 mit 0.5 m Toleranz.
Abb. E.15: Profil 3 mit 1 m Toleranz.
Abb. E.16: Profil 3 mit 2 m Toleranz.
Abb. E.17: 3D-Scatterplot des Profils 1 mit einer Toleranz von 2 m und mit nicht gefüllten Punktsymbolen.
Abb. E.18: 3D-Scatterplot des Profils 1 mit einer Toleranz von 5 m und mit einem Stern (*) als Punktsymbol.
Abb. E.19: 3D-Scatterplot des Profils 1 mit einer Toleranz von 5 m und mit gefüllten Punktsymbolen.
Abb. E.20: 3D-Scatterplot des Profils 1 mit einer Toleranz von 10 m und mit gefüllten Punktsymbolen.
Abb. E.21: 3D-Scatterplot des Profils 3 mit einer Toleranz von 10 m und mit einem Punkt '.' als Punktsymbol.
Abb F.1: Intensitätsraster aller Laserpunkte des Untersuchungsgebietes mit unterschiedlichen Einstellungen des IDW-Interpolationsverfahrens.
Abb. F.2: Intensitätsraster aller *first pulse* Laserpunkte des Untersuchungsgebietes mit unterschiedlichen Einstellungen des IDW-Interpolationsverfahrens.
Abb. F.3: Intensitätsraster aller *last pulse* Laserpunkte des Untersuchungsgebietes mit unterschiedlichen Einstellungen des IDW-Interpolationsverfahrens.
Abb. F.4: Intensitätsraster aller Laserpunkte des Untersuchungsgebietes der Befliegung ID=1 mit unterschiedlichen Einstellungen des IDW-Interpolationsverfahrens.
Abb. F.5: Intensitätsraster aller Laserpunkte des Untersuchungsgebietes der Befliegung ID=2 mit unterschiedlichen Einstellungen des IDW-Interpolationsverfahrens.

Tabellenverzeichnis

1 Einleitung

Tab. 1.1: Auszug aus der Projektbeschreibung des alpS-Forschungsprojekts A 2.5.
Tab. 1.2: Zeitplan der Diplomarbeit.

3 Einführung in das Untersuchungsgebiet

Tab. 3.1: Allgemeine Informationen zum Untersuchungsgebiet in Hohenems.
Tab. 3.2: Flugstreifenübersicht.
Tab. 3.3: Überblick über die Laserpunkte im Untersuchungsgebiet.

4 Datenerfassung und Datengrundlagen

Tab. 4.1: Relative Flughöhe, maximale Streifenbreite und Reichweite.
Tab. 4.2: Messsystem ALTM 2050 der Firma TopScan GmbH.
Tab. 4.3: Auszug aus einem Flugzeugdatenfile mit 200 Hz zeitlicher Auflösung.
Tab. 4.4: Schema der zeitlichen Verortung aller Laserscannerdaten.
Tab. 4.5: Auszug aus einem *first pulse* Datensatz.
Tab. 4.6: Anzahl der Files und Datenmenge (in Gigabyte) der einzelnen Befliegungskampagnen im Vorarlberger Unterland und im Gargellental.

5 Entwicklung des Informationssystems

Tab. 5.1: Anforderungskatalog: Eigenschaften und Funktionen des Systems.
Tab. 5.2: Arbeitsschritte bei der Entwicklung des Prototypens eines Informationssystems für Laserscannerdaten.
Tab. 5.3: Systemkomponenten des verwendeten Rechners.
Tab. 5.4: Die vier Freiheiten der *free / open source* Software.
Tab. 5.5: *Limitations of PostgreSQL.*
Tab. 5.6: Einige ausgewählte OpenGIS-Funktionen von PostGIS.
Tab. 5.7: Auszug aus einer Tabelle mit Geometrie.
Tab. 5.8: Versionen der verwendeten Softwarepakete.

6 Datenverwaltung und -speicherung

Tab. 6.1: Übersicht der Objekttypen (Entitäten) der Laserscannerbefliegung.
Tab. 6.2: Ausgelagerte Attribute der Entitäten Befliegung, Flugzeugposition und Laserpunkt.
Tab. 6.3: Auflistung der verwendeten Datentypen.

7 Datenimport

Tab. 7.1: Gegenüberstellung von drei Generalisierungsstufen (0.5 m, 1 m und 2 m Toleranz) mit dem Douglas-Peucker-Algorithmus.
Tab. 7.2: Rechenzeiten für die Berechnung und Speicherung der Attribute Geometrie, Länge und mittlere Fluggeschwindigkeit der Flugstreifen.
Tab. 7.3: Einfache Importszenarien mit drei verschiedenen Methoden und einer unterschiedlichen Anzahl von Datensätzen.
Tab. 7.4: Importzeiten der Rohdaten mit *putline().*

8 Datenverarbeitung

Tab. 8.1.: Berechnung der absoluten Zeit für 10 Flugzeugpositionen.
Tab. 8.2: Rechenzeiten für die absolute zeitliche Verortung einer unterschiedlichen Anzahl von Datensätzen.
Tab. 8.3.: Berechnung der Flugzeugposition für ausgewählte Laserpunkte.
Tab. 8.4: Rechenzeiten für die lineare Interpolation der Flugzeugposition für ausgewählte Laserpunkte.
Tab. 8.5.: Berechnung der Länge des Laservektors für ausgewählte Laserpunkte.
Tab. 8.6: Rechenzeiten für die Berechnung der Länge des Laservektors (*range*) bei gegebener Flugzeugposition.
Tab. 8.7: Auswirkung der Rechenzeit für einen Laserpunkt auf die Summe der Rechenzeit für 200 Millionen Punkte.
Tab. 8.8: Anzahl der Punkte, die bei der Rekonstruktion der Aufnahmegeometrie „Problemfälle“ darstellen.
Tab. 8.9.: Berechnung der Hangneigung für ausgewählte Laserpunkte (3D-Suchradius = 1 m).
Tab. 8.10: Rechenzeiten für die Berechnung der Hangneigung am Laserpunkt.
Tab. 8.11: Beispiel für die Exposition in Grad für die vier Haupthimmelsrichtungen und für ein Gelände mit 0° Neigung.
Tab. 8.12: Berechnungsschema für die Exposition der Ebene.
Tab. 8.13: Gelände mit Neigung 0°.
Tab. 8.14: Berechnung der Exposition für ausgewählte Laserpunkte (3D-Suchradius = 1 m).
Tab. 8.15: Rechenzeiten für die Berechnung der Exposition des Laserpunkts.
Tab. 8.16: Berechnung des Einfallswinkels für ausgewählte Laserpunkte (3D-Suchradius = 1 m).
Tab. 8.17: Rechenzeiten für die Berechnung des Einfallswinkels des Laservektors an der Geländeebene.
Tab. 8.18: Berechnung der Fläche des Lichtkegels am Boden (*footprint*) in m² für ausgewählte Laserpunkte (3D-Suchradius = 1 m).
Tab. 8.19: Rechenzeiten für die Berechnung der Fläche des Lichtkegels am Boden.
Tab. 8.20: Berechnung der relativen Flughöhe für ausgewählte Laserpunkte (2D-Suchradius = 1 m).
Tab. 8.21: Rechenzeiten für die Berechnung der relativen Flughöhe (2-D-Suchradius = 2 m).
Tab. 8.22: Rechenzeiten für die Berechnung von Neigung, Exposition, Einfallswinkel und Fläche des Lichtkegels am Boden in einer Funktion.
Tab. 8.23: Aggregatfunktionen in PostgreSQL.
Tab. 8.24: Deskriptive statistische Werte der Intensität berechnet mit PostgreSQL-Funktionen.
Tab. 8.25: Korrelationskoeffizienten zwischen Länge des Laservektors (*range*) und Intensität für unterschiedlich große Zufallsstichproben.

9 Datenexport und Visualisierung

Tab. 9.1: Ausgewählte GRASS-Exportformate.

Appendix

Tab. A.1: Flug- und Systemparameter.
Tab. A.2: Befliegungskampagnen des Untersuchungsgebietes.
Tab. D.1: Flugstreifen des Untersuchungsgebietes aufgeschlüsselt nach Befliegungen und minimaler Länge des Laservektors (*range*).

Appendix

A Befliegungskampagne Vorarlberg

	Unterland / Vorderwald	Verdichtungsbereiche
Abstand der Flugachsen	300 - 400 m	100 m
mittl. Flughöhe über Grund	1000 m	1000 m
mittl. Fluggeschwindigkeit	70 m/s	70 m/s
Lasermessrate	50 000 Hz	50 000 Hz
Scanfrequenz	30 Hz	30 Hz
Scanwinkel	± 20 deg	± 20 deg
mittl. Streifenbreite	728 m	728 m
mittl. Streifenüberlappung	328 - 428 m	628 m
theoretische Anzahl Laserpunkte/m² *	1.8 - 2.4	6.9

Tab. A.1: Flug- und Systemparameter mit dem Aufnahmesystem ALTM 2050 von Optech (Topscan Gmbh. 2004, 1) * ohne Mehrfachreflexionen.

ID	Datum der Befliegung	Name der Befliegung	Anzahl der Streifen	Anzahl der Laserpunkte	zeitlicher Beginn	zeitliches Ende	Datum des Wochenbeginns
1	05.11.2003	031105_1	4	738557	05.11.2003 11:01:21	05.11.2003 11:25:25	2.11.2003
2	04.11.2003	031104_1	13	3058889	04.11.2003 09:17:18	04.11.2003 12:04:58	2.11.2003

Tab. A.2: Befliegungskampagnen des Untersuchungsgebietes (Auszug aus dem Informationssystem).

B Quelltextbeispiele

B.1 Performancetests Datenimport

Python-Skript, das 50 Millionen Datensätze erzeugt, in ein File schreibt und es dann mit COPY in die PostgreSQL-Datenbank importiert (copy.py):

```
#! /usr/bin/env python
#-*- coding: cp1252 -*-
"""Script tests the time usage for import data with COPY statement"""
import os
from runtime import *

#start timing
counttime=runtime()
counttime.start()
out_file = file("copy.sql","w")
out_file.write("BEGIN;\n")
out_file.write("COPY test_copy from STDIN;\n")
geom="SRID=-1;POINT(10000.23 8098908.22 1000.24)"
for i in range(0, 50000000):
    outstr="%s\ttest term\t1\t9999\t%s\n" %(i, geom)
    out_file.write(outstr)
out_file.write("\.\nCOMMIT;")
out_file.close()
os.system("psql -f copy.sql")
#end timing
counttime.end()
```

Python-Skript, das 50 Millionen Datensätze dynamisch erzeugt und mit INSERT in die PostgreSQL-Datenbank schreibt (insert.py):

```
#! /usr/bin/env python
#-*- coding: cp1252 -*-
"""Script tests the time usage for import data with INSERT statement"""
import psycopg
from runtime import *

#start timing
counttime=runtime()
counttime.start()
cnx = psycopg.connect("user=laser dbname=laser host=localhost")
cr = cnx.cursor()
geom="SRID=-1;POINT(10000.23 8098908.22 1000.24)"
for i in range(0, 50000000):
      sql="INSERT INTO test_insert(term, min, max, geom)  values ('test term',%s,
      10,'%s')" %(i, geom)
      cr.execute(sql)
      cnx.commit()
cr.close()
cnx.close()
```

```
#end timing
counttime.end()
```

Python-Skript, das 50 Millionen Datensätze dynamisch erzeugt und mit PUTLINE in die PostgreSQL-Datenbank schreibt (putline.py):

```
#! /usr/bin/env python
import pg
conn=pg.connect(dbname='laser', host='localhost' , user='laser')
conn.query("copy test_copy from stdin")
geom="SRID=-1;POINT(10000.23 8098908.22 1000.24)"

for i in range(50000000):
       outstr="%s\ttest term\t1\t9999\t%s\n" %(i, geom)
       conn.putline(outstr)
#      if i%1000==0:
#             conn.putline('\\.\n')
#             conn.endcopy()
 #            conn.query("copy test_copy from stdin")


conn.putline('\\.\n')
conn.endcopy()
```

B.2 Datenbankfunktionen

PL/pgSQL-Funktion für die Berechnung der Liniengeometrie eines Flugstreifens aus den Flugzeugpositionen:

```
CREATE OR REPLACE FUNCTION update_strip(FLOAT)
RETURNS TEXT AS '
DECLARE
       strip RECORD;
       planepos RECORD;
       geom_new GEOMETRY;
       geom_old GEOMETRY;
       range_ float;
       length_ float:=0.0;
       maxtime NUMERIC(9,3):=0;
       mintime NUMERIC(9,3):=0;
       speed NUMERIC(5,2);
       num INTEGER:=0;
       sql TEXT;
       back TEXT;
       line TEXT:='''';
       line_temp TEXT:='''';
       line_geom GEOMETRY;
       factor ALIAS FOR $1;
BEGIN
       RAISE NOTICE ''start calculating the average flight speed...'';
```

```
	CREATE OR REPLACE FUNCTION simplyline(text, float)
	RETURNS TEXT AS ''
	DECLARE
		sline TEXT;
		sline_geom GEOMETRY;

	BEGIN
		sline:=substr($1, 1, length($1)-1)||'''')'''';
		sline_geom:=simplify(geometryfromtext(sline::text, -1), $2);
		sline:=astext(sline_geom);
		sline:=substr(sline, 1, length(sline)-1)||'''','''';
		RETURN sline;

	END;
	'' LANGUAGE plpgsql;

	FOR strip IN SELECT * FROM tdtastrip WHERE speed IS NULL LOOP
		maxtime:=0;
		mintime:=0;
		length_:=0;
		Geom_old:=NULL;
		num:=0;
		range_:=0;
		line:=''LINESTRING('';
		line_temp:='''';
		FOR planepos IN SELECT * FROM tdtaplaneposition WHERE
idstrip=strip.idstrip ORDER BY time ASC LOOP
			geom_new:=planepos.geom_point;
			IF num=0 THEN
				mintime:=planepos.time;
			END IF;
			IF planepos.time > maxtime THEN
				maxtime:=planepos.time;
			END IF;
			IF geom_old IS NOT NULL THEN
				range_:= range(geom_new, geom_old);
			END IF;
			geom_old:=geom_new;
			length_:=length_+range_;
			RAISE NOTICE ''%\t%'', planepos.time,length_;
			num:=num+1;
			line_temp:=x(geom_new)::text||'' ''||y(geom_new)::text||'' ''||z
(geom_new)::text||'','';
			line:=line||line_temp;
			--Simplify Line every 1000th point
			IF num%1000=0 THEN
				line:=simplyline(line, factor);
			END IF;
		END LOOP;
		speed:=length_/(maxtime-mintime);
		sql:=''UPDATE tdtastrip SET flightspeed=''||speed::text||'' WHERE
idstrip=''||strip.idstrip::text;
		EXECUTE sql;
		sql:=''UPDATE tdtastrip SET length=''||round(length_::numeric,2)::
```

```
text||'' WHERE idstrip=''||strip.idstrip::text;
             EXECUTE sql;
             line_temp:=simplyline(line, factor);
             line:=substr(line_temp, 1, length(line_temp)-1)||'')'';
             sql:=''UPDATE     tdtastrip     SET     geom_line=linefromtext(''''''||
line::text||'''''', -1) WHERE idstrip=''||strip.idstrip::text;
             EXECUTE sql;
             RAISE NOTICE ''Strip (ID:%): % m/s'', strip.idstrip, speed;
      END LOOP;
             back:=''done'';
             RETURN back;

END;
' LANGUAGE plpgsql;
```

PL/R-Funktion für die Berechnung des Korrelationskoeffizienten zweier Variablen:

```
CREATE OR REPLACE FUNCTION r_cor(text, text, text, integer)
  RETURNS float AS
'
      sql <-paste(''select '',arg1,'' as x, '',arg2,'' as y from '',arg3,'' order
by random() limit '',arg4)
      str <- pg.spi.exec (sql);
      attach(str)
      coeff<-cor(x, y, use = ''complete.obs'');
      return(round(coeff,3))
'
LANGUAGE 'plr';
```

PL/R-Funktion für die Berechnung des Medians:

```
CREATE OR REPLACE FUNCTION r_med(_float8)
  RETURNS float AS
'
      round(median(arg1),2)
'
LANGUAGE 'plr';

CREATE AGGREGATE median(
      sfunc = plr_array_accum,
      basetype = float8,
      stype = _float8,
      finalfunc = r_med
);
```

PL/R-Funktion für die Berechnung der Standardabweichung:

```
CREATE OR REPLACE FUNCTION r_sd(_float8)
  RETURNS float AS
'
      round(sd(arg1),2)
```

```
'
LANGUAGE 'plr';

CREATE AGGREGATE sd(
       sfunc = plr_array_accum,
       basetype = float8,
       stype = _float8,
       finalfunc = r_sd
);
```

C Logfile des Datenimports

Import aller Laserpunkte und Flugzeugpositionen, die das Untersuchungsgebiet betreffen (ca. 190 Millionen Laserpunkte):

```
11:00:19 >>> PostgreSQL 7.4.5 on i686-redhat-linux-gnu, compiled by GCC gcc (GCC)
3.3.3 20040412 (Red Hat Linux 3.3.3-7)
11:00:19 >>> open database connection...
11:00:19 >>> checking FLIGHT DATA...
11:00:19 >>> write FLIGHT with name "031105_1" into DB...
11:00:19 >>> write IMU data "031105_1.asc"
11:08:11 >>> 1285000 plane positions for "031105_1.asc" successfully written into
DB.
11:08:11 >>> checking STRIP DATA...
11:08:12 >>> write STRIP number "2306" into DB.
11:08:12 >>> write point data "str_2306.all"
12:17:51 >>> 9668305 laser points for "str_2306.all" successfully written into DB.
12:17:51 >>> checking STRIP DATA...
12:17:51 >>> STRIP number "2306" already exists in DB.
12:19:27 >>> write point data "str_2306.alf"
13:30:40 >>> 9258415 laser points for "str_2306.alf" successfully written into DB.
13:30:40 >>> checking STRIP DATA...
13:30:40 >>> write STRIP number "2304" into DB.
13:30:40 >>> write point data "str_2304.all"
14:52:51 >>> 9445806 laser points for "str_2304.all" successfully written into DB.
14:52:52 >>> checking STRIP DATA...
14:52:52 >>> STRIP number "2304" already exists in DB.
14:56:39 >>> write point data "str_2304.alf"
16:24:38 >>> 9122006 laser points for "str_2304.alf" successfully written into DB.
16:24:38 >>> checking STRIP DATA...
16:24:38 >>> write STRIP number "2166" into DB.
16:24:38 >>> write point data "str_2166.all"
18:00:46 >>> 10494412 laser points for "str_2166.all" successfully written into DB.
18:00:46 >>> checking STRIP DATA...
18:00:46 >>> STRIP number "2166" already exists in DB.
18:06:41 >>> write point data "str_2166.alf"
19:45:29 >>> 10044459 laser points for "str_2166.alf" successfully written into DB.
19:45:30 >>> checking STRIP DATA...
19:45:30 >>> write STRIP number "2164" into DB.
19:45:30 >>> write point data "str_2164.all"
21:41:56 >>> 10353353 laser points for "str_2164.all" successfully written into DB.
21:41:56 >>> checking STRIP DATA...
```

```
21:41:56 >>> STRIP number "2164" already exists in DB.
21:48:53 >>> write point data "str_2164.alf"
23:42:07 >>> 9941505 laser points for "str_2164.alf" successfully written into DB.
23:42:07 >>> checking FLIGHT DATA...
23:42:07 >>> write FLIGHT with name "031104_1" into DB...
23:42:07 >>> write IMU data "031104_1.asc"
23:47:26 >>> 691602 plane positions for "031104_1.asc" successfully written into
DB.
23:47:26 >>> checking STRIP DATA...
23:47:26 >>> write STRIP number "2470" into DB.
23:47:26 >>> write point data "str_2470.all"
00:04:58 >>> 2230769 laser points for "str_2470.all" successfully written into DB.
00:04:58 >>> checking STRIP DATA...
00:04:58 >>> STRIP number "2470" already exists in DB.
00:12:01 >>> write point data "str_2470.alf"
00:30:09 >>> 2160037 laser points for "str_2470.alf" successfully written into DB.
00:30:09 >>> checking STRIP DATA...
00:30:09 >>> write STRIP number "2462" into DB.
00:30:09 >>> write point data "str_2462.all"
00:53:04 >>> 2522673 laser points for "str_2462.all" successfully written into DB.
00:53:04 >>> checking STRIP DATA...
00:53:04 >>> STRIP number "2462" already exists in DB.
00:59:48 >>> write point data "str_2462.alf"
01:22:10 >>> 2453242 laser points for "str_2462.alf" successfully written into DB.
01:22:10 >>> checking STRIP DATA...
01:22:10 >>> write STRIP number "2136" into DB.
01:22:10 >>> write point data "str_2136.all"
03:07:30 >>> 10723930 laser points for "str_2136.all" successfully written into DB.
03:07:31 >>> checking STRIP DATA...
03:07:31 >>> STRIP number "2136" already exists in DB.
03:16:03 >>> write point data "str_2136.alf"
05:04:14 >>> 10530547 laser points for "str_2136.alf" successfully written into DB.
05:04:14 >>> checking STRIP DATA...
05:04:14 >>> write STRIP number "2108" into DB.
05:04:14 >>> write point data "str_2108.all"
05:24:25 >>> 2210080 laser points for "str_2108.all" successfully written into DB.
05:24:25 >>> checking STRIP DATA...
05:24:25 >>> STRIP number "2108" already exists in DB.
05:33:47 >>> write point data "str_2108.alf"
05:53:31 >>> 2141193 laser points for "str_2108.alf" successfully written into DB.
05:53:31 >>> checking STRIP DATA...
05:53:31 >>> write STRIP number "2107" into DB.
05:53:31 >>> write point data "str_2107.all"
06:11:21 >>> 2149647 laser points for "str_2107.all" successfully written into DB.
06:11:21 >>> checking STRIP DATA...
06:11:21 >>> STRIP number "2107" already exists in DB.
06:20:16 >>> write point data "str_2107.alf"
06:38:24 >>> 2044258 laser points for "str_2107.alf" successfully written into DB.
06:38:24 >>> checking STRIP DATA...
06:38:24 >>> write STRIP number "2093" into DB.
06:38:24 >>> write point data "str_2093.all"
06:57:15 >>> 2190677 laser points for "str_2093.all" successfully written into DB.
06:57:15 >>> checking STRIP DATA...
06:57:15 >>> STRIP number "2093" already exists in DB.
```

```
07:06:30 >>> write point data "str_2093.alf"
07:25:50 >>> 2132167 laser points for "str_2093.alf" successfully written into DB.
07:25:50 >>> checking STRIP DATA...
07:25:50 >>> write STRIP number "2088" into DB.
07:25:50 >>> write point data "str_2088.all"
07:55:07 >>> 3071397 laser points for "str_2088.all" successfully written into DB.
07:55:07 >>> checking STRIP DATA...
07:55:07 >>> STRIP number "2088" already exists in DB.
08:04:51 >>> write point data "str_2088.alf"
08:32:52 >>> 2971331 laser points for "str_2088.alf" successfully written into DB.
08:32:52 >>> checking STRIP DATA...
08:32:52 >>> write STRIP number "2065" into DB.
08:32:52 >>> write point data "str_2065.all"
08:55:45 >>> 2394898 laser points for "str_2065.all" successfully written into DB.
08:55:45 >>> checking STRIP DATA...
08:55:45 >>> STRIP number "2065" already exists in DB.
09:05:52 >>> write point data "str_2065.alf"
09:28:41 >>> 2324330 laser points for "str_2065.alf" successfully written into DB.
09:28:41 >>> checking STRIP DATA...
09:28:41 >>> write STRIP number "2059" into DB.
09:28:41 >>> write point data "str_2059.all"
11:11:17 >>> 10174936 laser points for "str_2059.all" successfully written into DB.
11:11:17 >>> checking STRIP DATA...
11:11:17 >>> STRIP number "2059" already exists in DB.
11:23:04 >>> write point data "str_2059.alf"
13:07:45 >>> 10012374 laser points for "str_2059.alf" successfully written into DB.
13:07:46 >>> checking STRIP DATA...
13:07:46 >>> write STRIP number "2046" into DB.
13:07:46 >>> write point data "str_2046.all"
13:34:51 >>> 2633876 laser points for "str_2046.all" successfully written into DB.
13:34:51 >>> checking STRIP DATA...
13:34:51 >>> STRIP number "2046" already exists in DB.
13:47:30 >>> write point data "str_2046.alf"
14:14:22 >>> 2562188 laser points for "str_2046.alf" successfully written into DB.
14:14:22 >>> checking STRIP DATA...
14:14:22 >>> write STRIP number "2045" into DB.
14:14:22 >>> write point data "str_2045.all"
14:43:22 >>> 2479913 laser points for "str_2045.all" successfully written into DB.
14:43:22 >>> checking STRIP DATA...
14:43:22 >>> STRIP number "2045" already exists in DB.
14:56:38 >>> write point data "str_2045.alf"
15:25:56 >>> 2412573 laser points for "str_2045.alf" successfully written into DB.
15:25:56 >>> checking STRIP DATA...
15:25:56 >>> write STRIP number "2044" into DB.
15:25:56 >>> write point data "str_2044.all"
15:56:53 >>> 2417001 laser points for "str_2044.all" successfully written into DB.
15:56:53 >>> checking STRIP DATA...
15:56:53 >>> STRIP number "2044" already exists in DB.
16:09:33 >>> write point data "str_2044.alf"
16:40:20 >>> 2361367 laser points for "str_2044.alf" successfully written into DB.
16:40:20 >>> checking STRIP DATA...
16:40:20 >>> write STRIP number "2043" into DB.
16:40:20 >>> write point data "str_2043.all"
18:54:55 >>> 10832056 laser points for "str_2043.all" successfully written into DB.
```

```
18:54:55 >>> checking STRIP DATA...
18:54:55 >>> STRIP number "2043" already exists in DB.
19:11:56 >>> write point data "str_2043.alf"
21:34:21 >>> 10673300 laser points for "str_2043.alf" successfully written into DB.
21:34:21 >>> UPDATE GEOMETRY STATS
06:26:23 >>> processing time: 43 h, 26 m 4 s
06:26:23 >>> ...close database connection
```

D Statistische Datenplots

D.1 Intensität

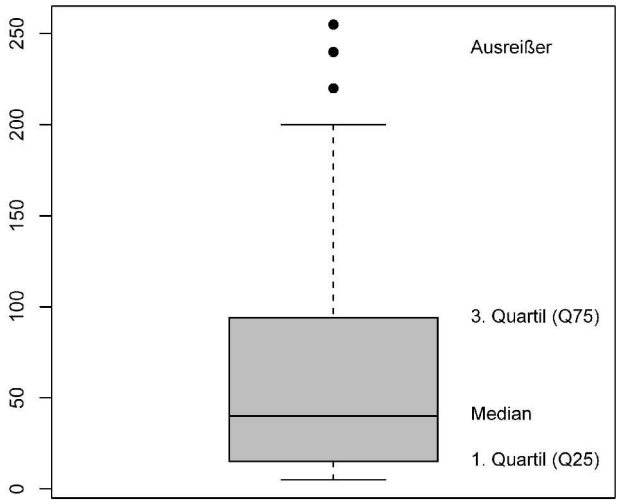


Abb. D.1: Boxplot der Intensität aller Punkte im Untersuchungsgebiet. Datenpunkte, die um das 1.5-fache des Interquartilabstandes (Q75-Q25) über die Box reichen, werden als einzelne Punkte dargestellt (eigener Entwurf).

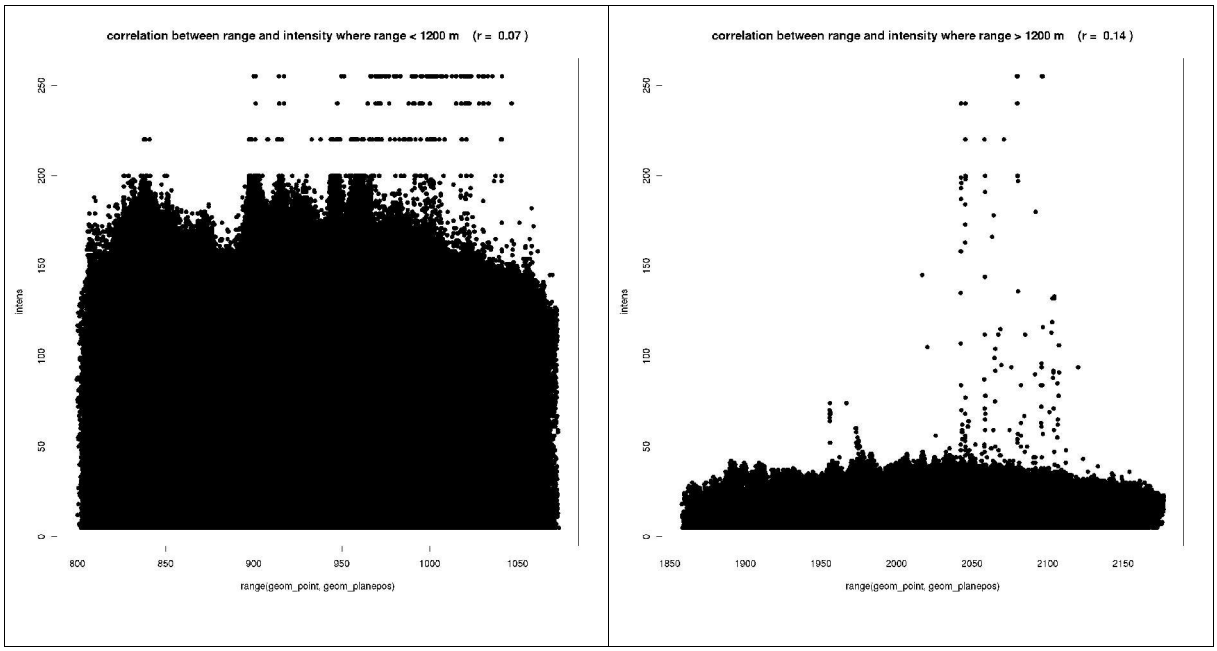


Abb. D.2: Plot mit berechnetem Korrelationskoeffizienten zwischen Intensität und Länge des Laservektors (*range*). links: Befliegung ID=2 (r = 0.07), rechts: Befliegung ID=1 (r = 0.14) (vgl. Tab. A.2; eigener Entwurf).

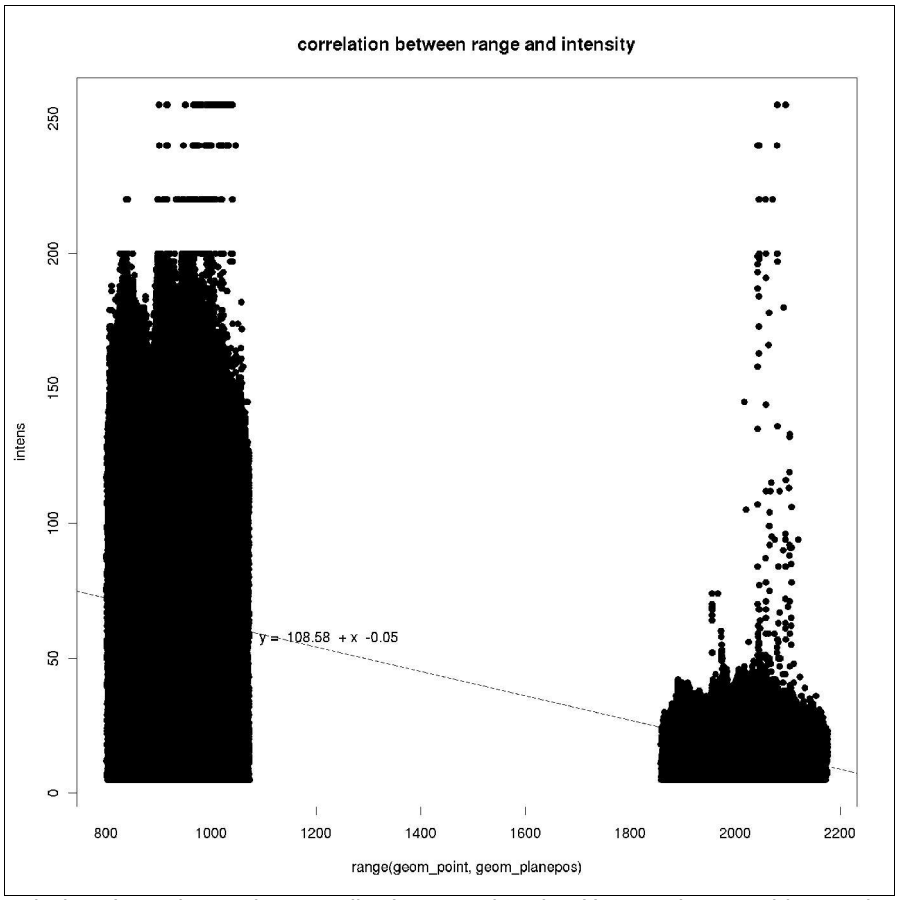


Abb. D.3: Plot zwischen Intensität und *range* aller Laserpunkte des Untersuchungsgebietes mit eingezeichneter Regressionsgeraden (vgl. Abb. D2; eigener Entwurf).

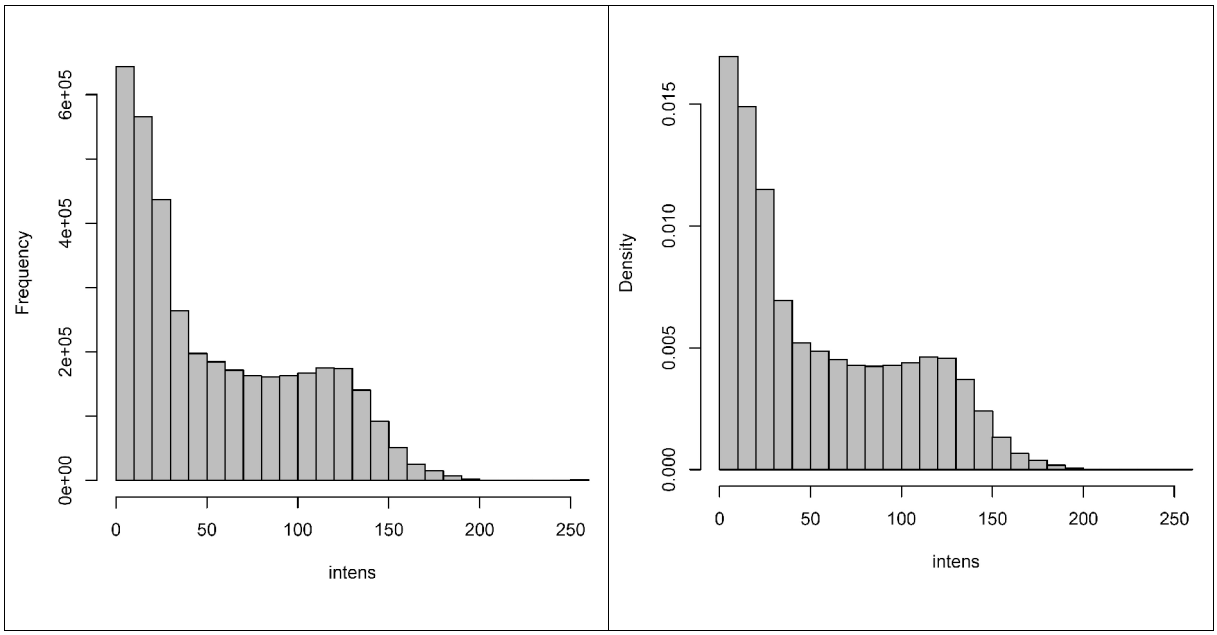


Abb. D.4: Histogramm der Intensität aller Laserpunkte im Untersuchungsgebiet (links: absolute Häufigkeiten, rechts: relative Häufigkeiten) (eigener Entwurf).

D.2 Länge des Laservektors

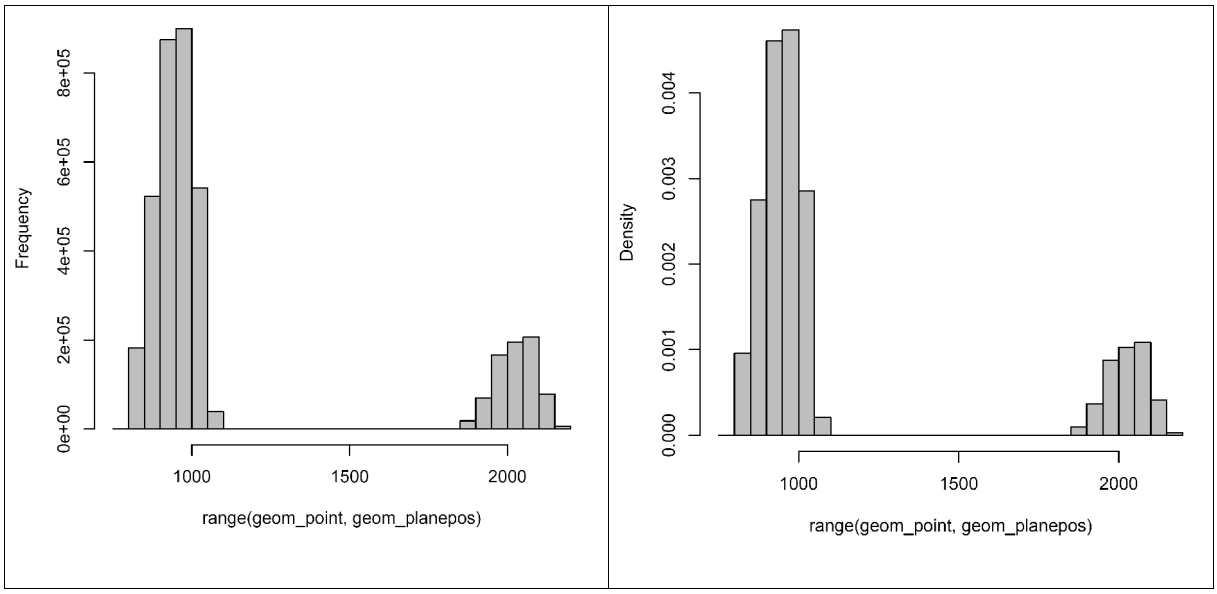


Abb. D.5: Histogramm der Länge des Laservektors für alle Laserpunkte im Untersuchungsgebiet (eigener Entwurf).

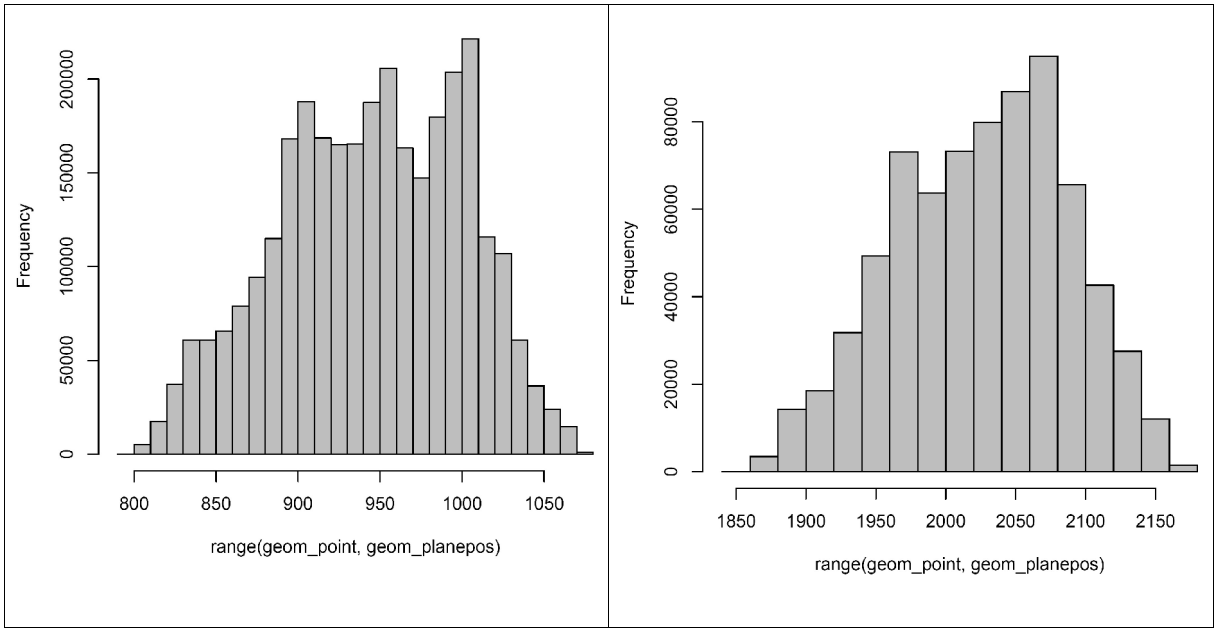


Abb. D.6: Histogramme der Länge des Laservektors aufgeschlüsselt nach Befliegungskampagnen. links: Befliegung ID=2, rechts: Befliegung ID=1 (eigener Entwurf).

Befliegungs-ID	Streifen-ID	*minimaler range [m]*
1	1	1870
1	2	1901
1	3	1922
1	4	1858
2	5	808
2	6	821
2	7	918
2	8	802
2	11	801
2	12	961
2	13	832
2	15	842
2	16	879
2	17	799

Tab. D.1: Flugstreifen des Untersuchungsgebietes aufgeschlüsselt nach Befliegungen und minimaler Länge des Laservektors (*range*) (vgl. Tab. A.2).

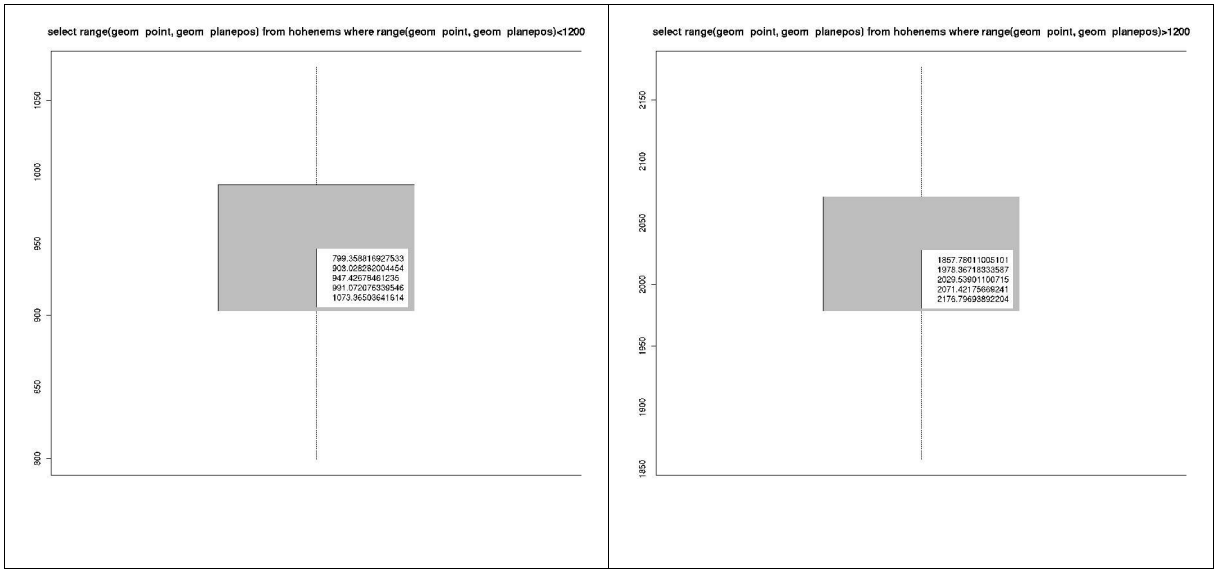


Abb. D.7: Boxplots der Länge des Laservektors. links: Befliegung ID=2, rechts: Befliegung ID=1. Die Zahlen im weißen Kästchen entsprechen von oben nach unten: Minimum, unteres Quartil, Median, oberes Quartil und Maximum (eigener Entwurf, vgl. DOLIĆ 2004, 101).

D.3 Neigung

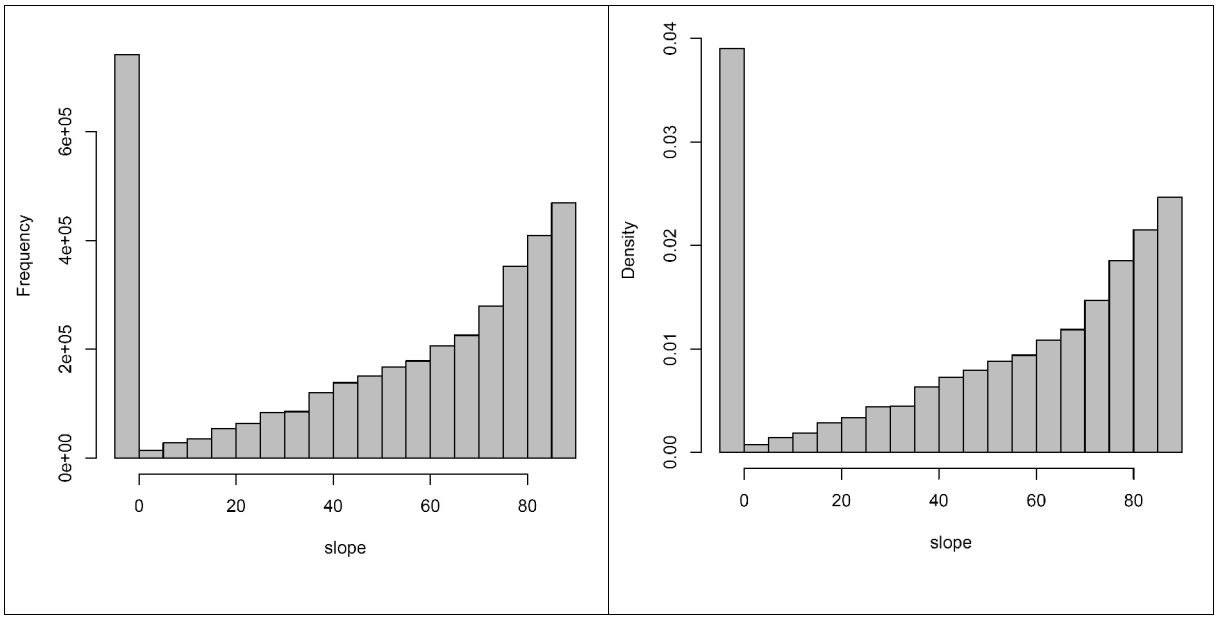


Abb. D.8: Histogramm der Neigung. Alle Werte unter 0 stellen Punkte dar, die nicht zur Berechnung herangezogen werden konnten (**Suchradius 0.5 m**) (eigener Entwurf).

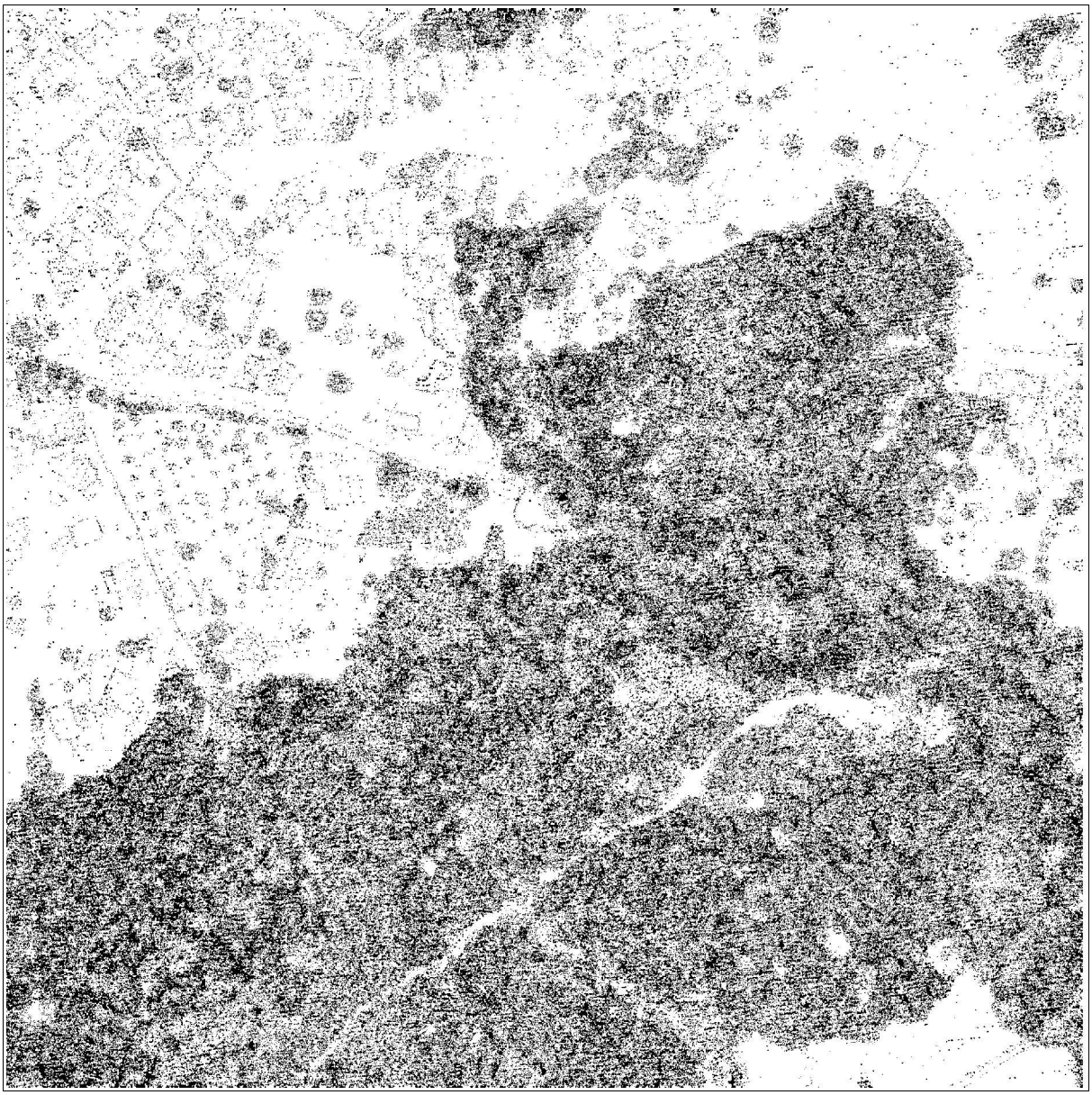


Abb. D.9: Bereiche (in schwarz), die bei einem 3-D-Suchradius von 0.5 m weniger als 2 Nachbarpunkte haben (vgl. Kapitel *„8 Datenverarbeitung“*; eigener Entwurf).

D.4 Exposition

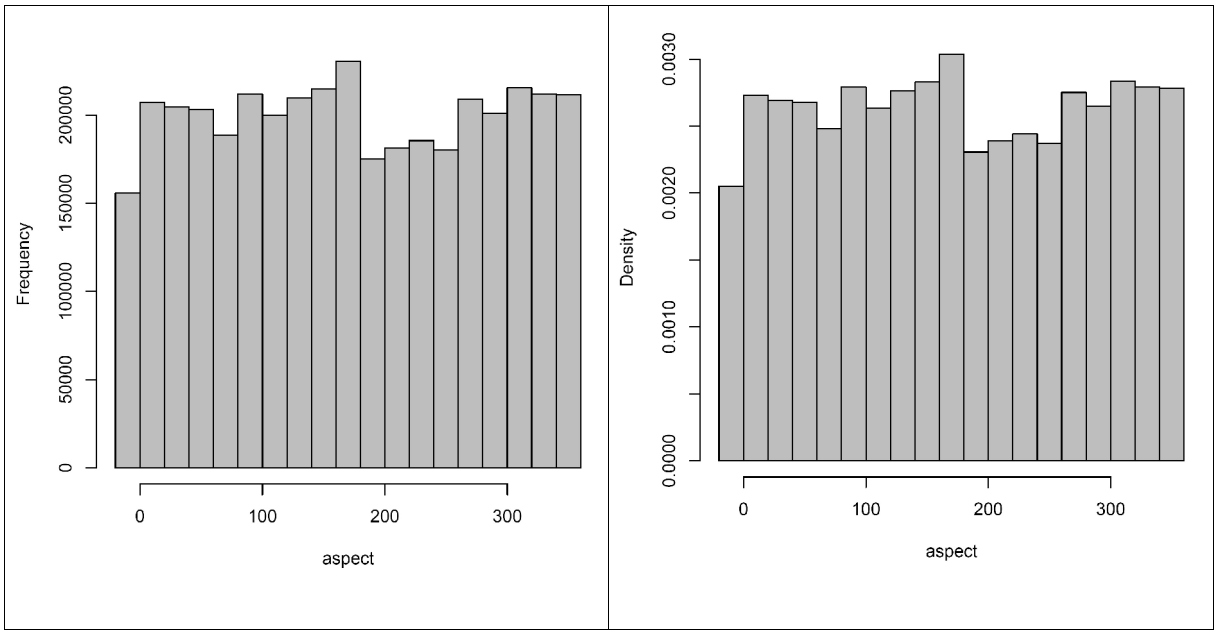


Abb. D.10: Histogramm der Exposition. Alle Werte unter 0 stellen Punkte dar, die nicht zur Berechnung herangezogen werden konnten oder eine Neigung von 0° aufweisen (**Suchradius 1 m**) (eigener Entwurf).

D.5 Einfallswinkel

Laserpunkte, die einen Einfallswinkel >= 90° bei einem Suchradius von 1 m aufweisen:

307624 (8.1 % aller Laserpunkte)

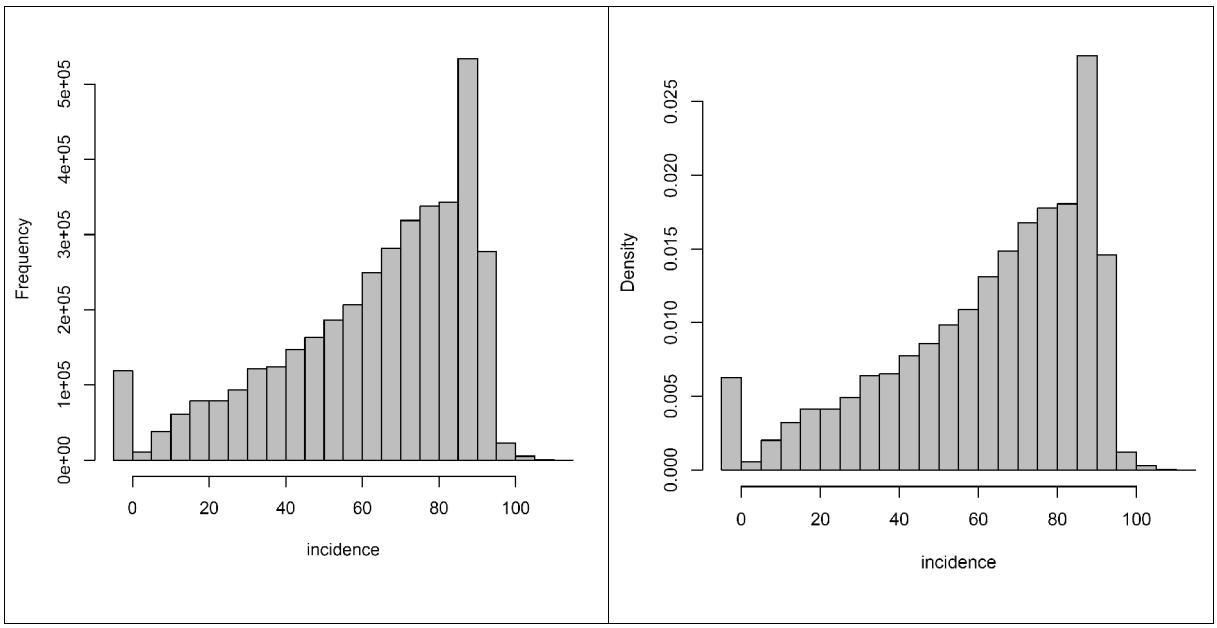


Abb. D.11: Histogramm des Einfallswinkels. Alle Werte unter 0 stellen Punkte dar, die nicht zur Berechnung herangezogen werden konnten (**Suchradius 1 m**) (eigener Entwurf).

D.6 Fläche des Lichtkegels am Boden

Laserpunkte, die einen *footprint* >= 5 m^2 bei einem Suchradius von 0.5 m aufweisen:

75647 (2 % aller Laserpunkte)

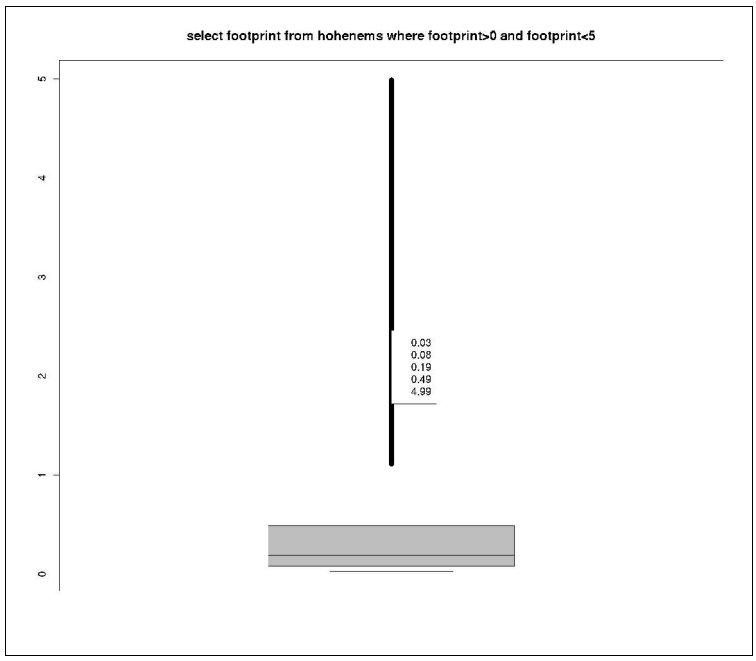


Abb. D.12: Boxplot des *footprints* für Werte < 5 m². Die Zahlen im weißen Kästchen entsprechen von oben nach unten: Minimum, unteres Quartil, Median, oberes Quartil und Maximum (eigener Entwurf).

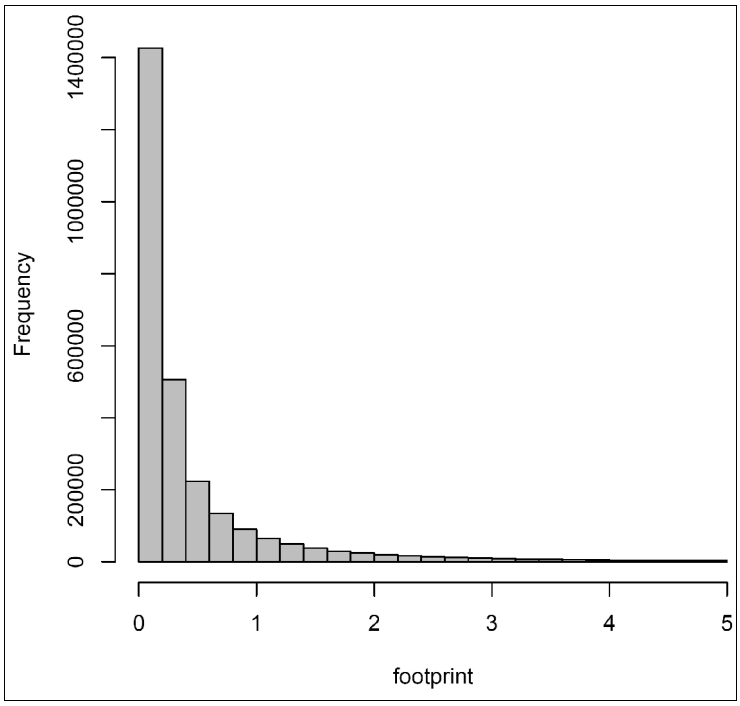


Abb. D.13: Histogramm des *footprints* aller Punkte mit einem Wert <5 m² (eigener Entwurf).

D.7 Relative Flughöhe

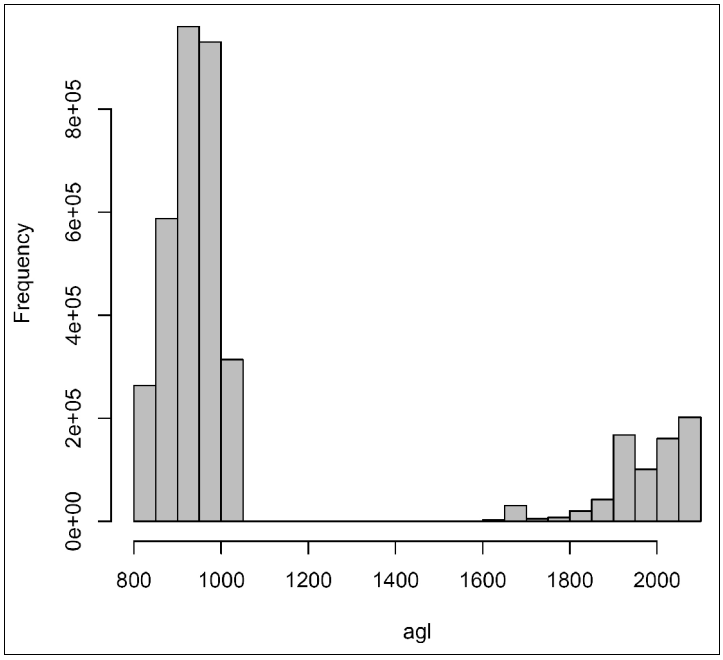


Abb. D.14: Histogramm der relativen Flughöhe für alle Laserpunkte (2D-Suchradius = 1 m; eigener Entwurf).

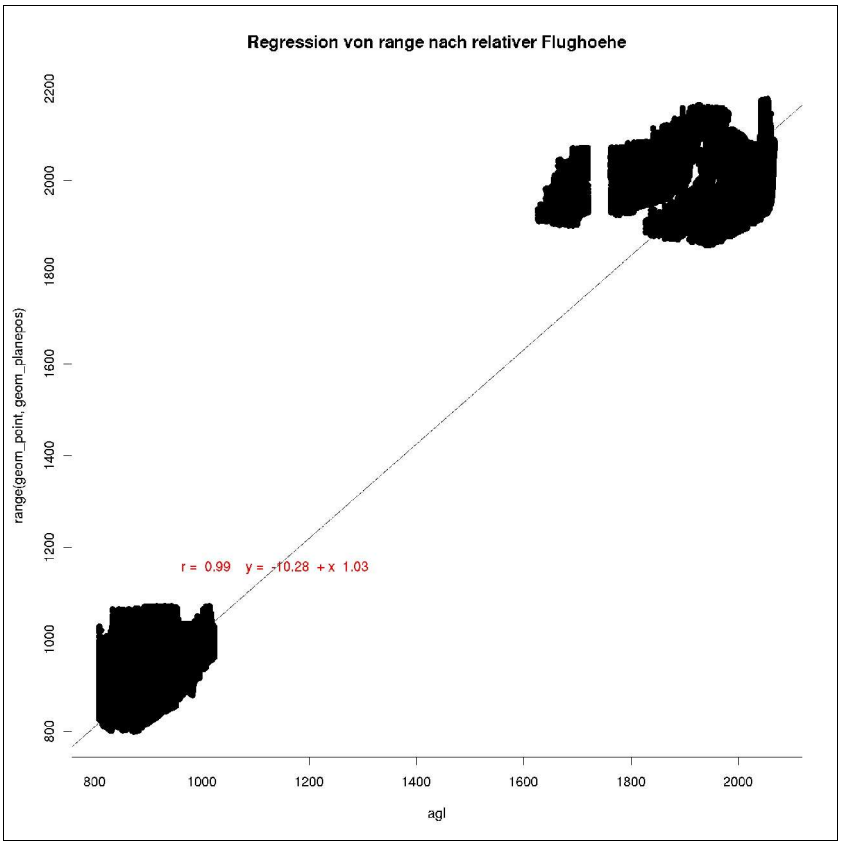


Abb. D.15: Regression von *range* (y) nach relativer Flughöhe (x). n=3787446 (eigener Entwurf).

E Profillinien

E.1 Übersicht

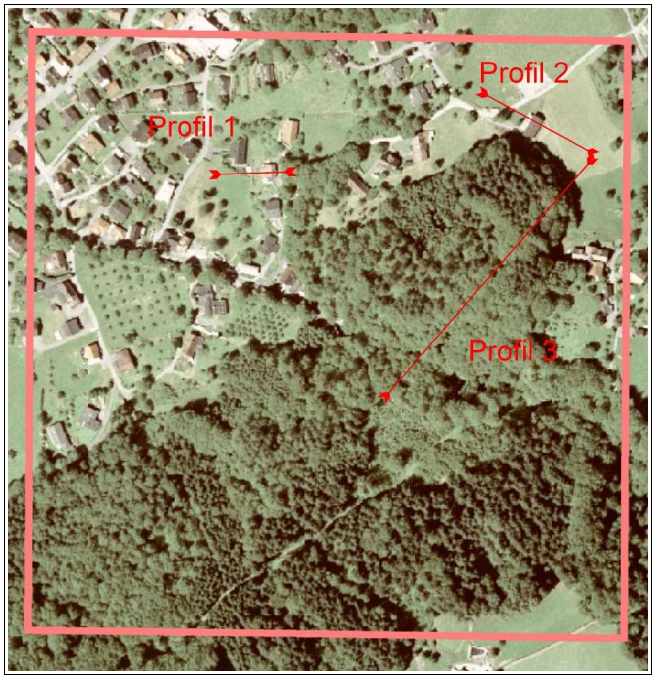


Abb. E.1: Standortsübersicht der drei Profillinien im Untersuchungsgebiet (eigener Entwurf).

E.2 Zweidimensionale Profile

Abb. E.2: Standort des Profils Nr. 1 (eigener Entwurf; Orthofoto: Land Vorarlberg; Foto: Höfle 25.12.2004).

Abb. E.3: Detailansicht des Hauses in Profil 1(25.12.2004).

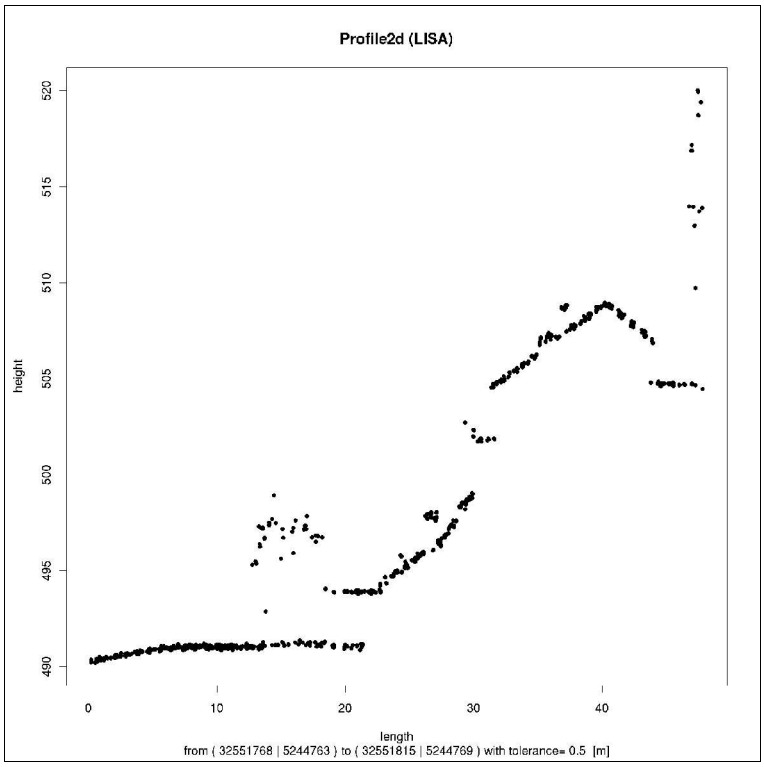


Abb. E.4: Profil 1 mit 0.5 m Toleranz (eigener Entwurf).

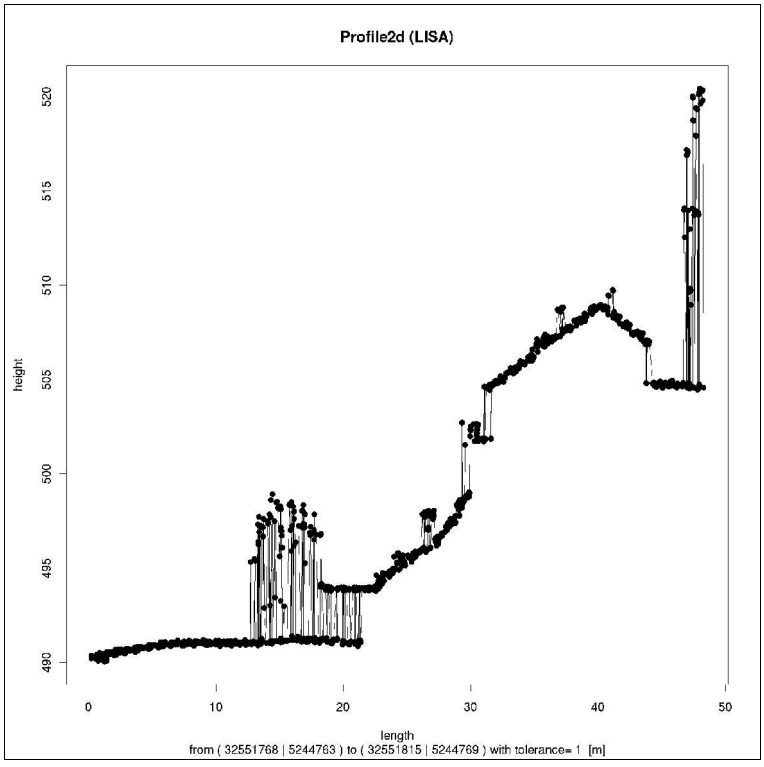


Abb. E.5: Profil 1 mit 1 m Toleranz und mit Linien verbundenen Punktsymbolen (eigener Entwurf).

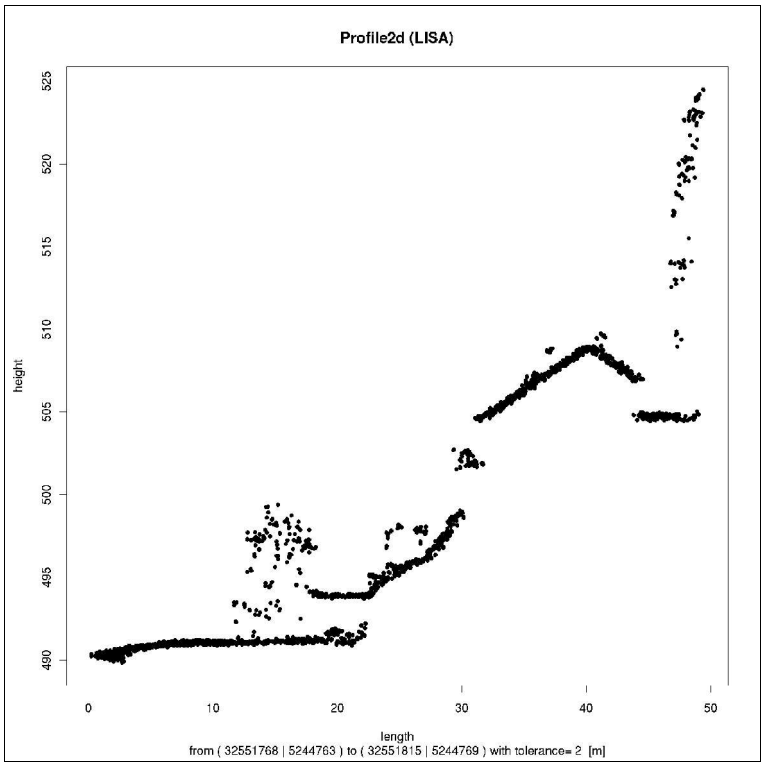


Abb. E.6: Profil 1 mit 2 m Toleranz (eigener Entwurf).

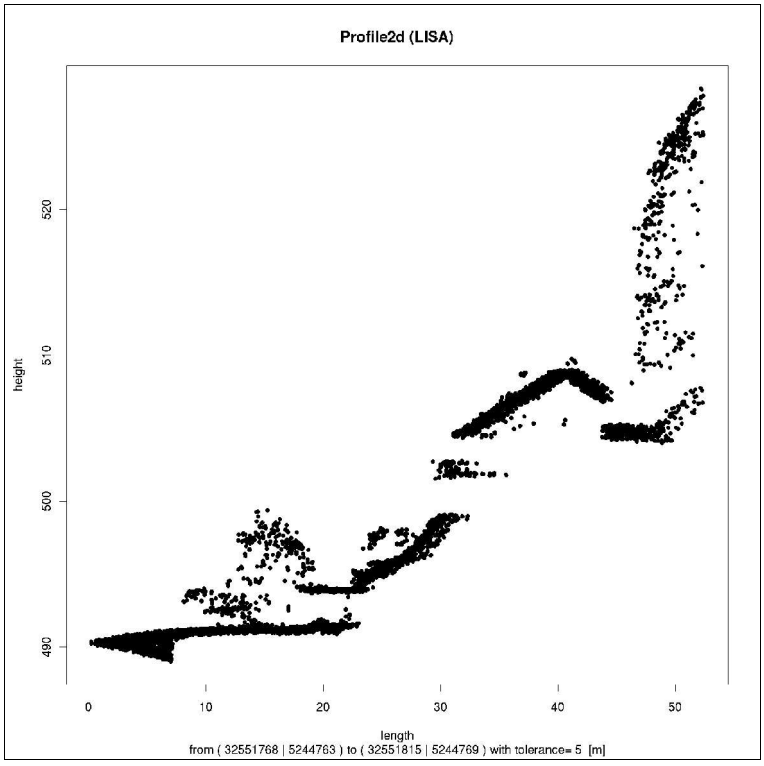


Abb. E.7: Profil 1 mit 5 m Toleranz (eigener Entwurf).

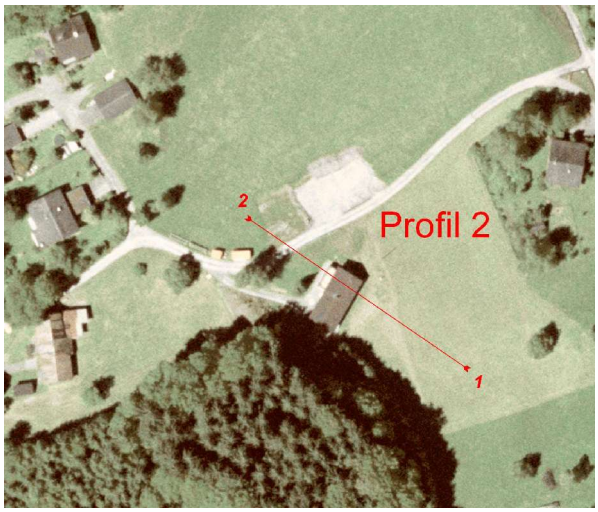


Abb. E.8: Standort des Profils Nr. 2 (eigener Entwurf; Orthofoto: Land Vorarlberg; Foto: Höfle 25.12.2004).

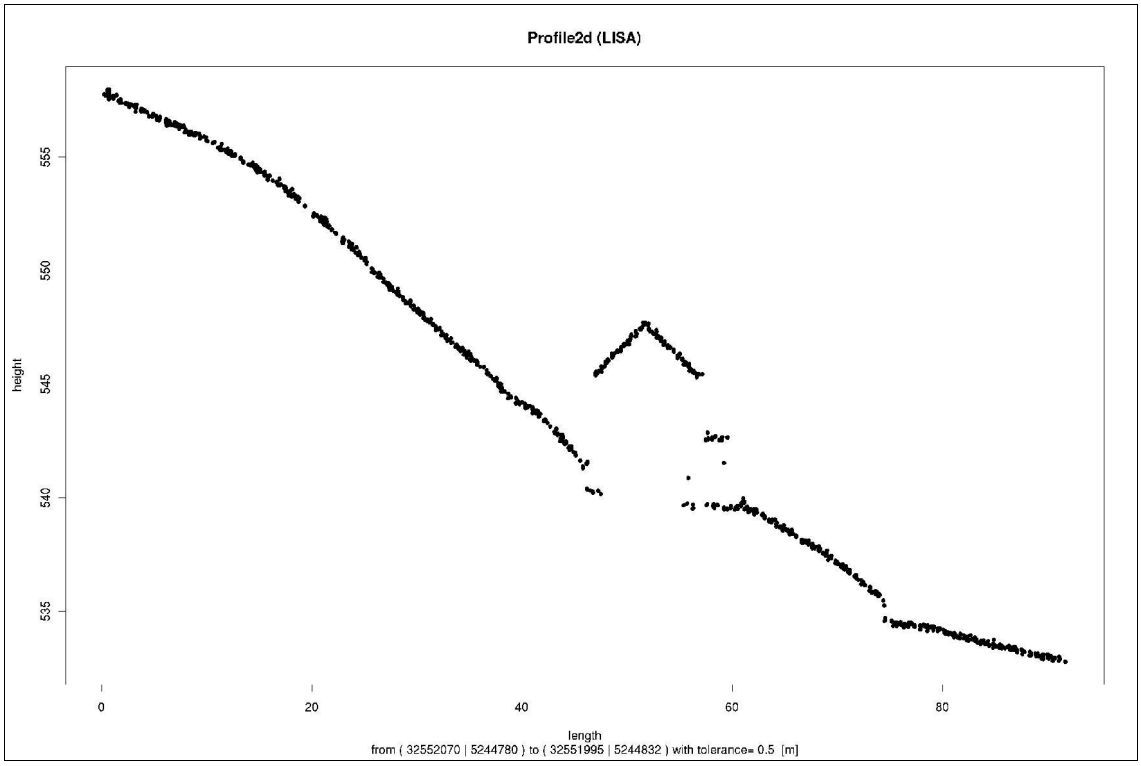


Abb. E.9: Profil 2 mit 0.5 m Toleranz (eigener Entwurf).

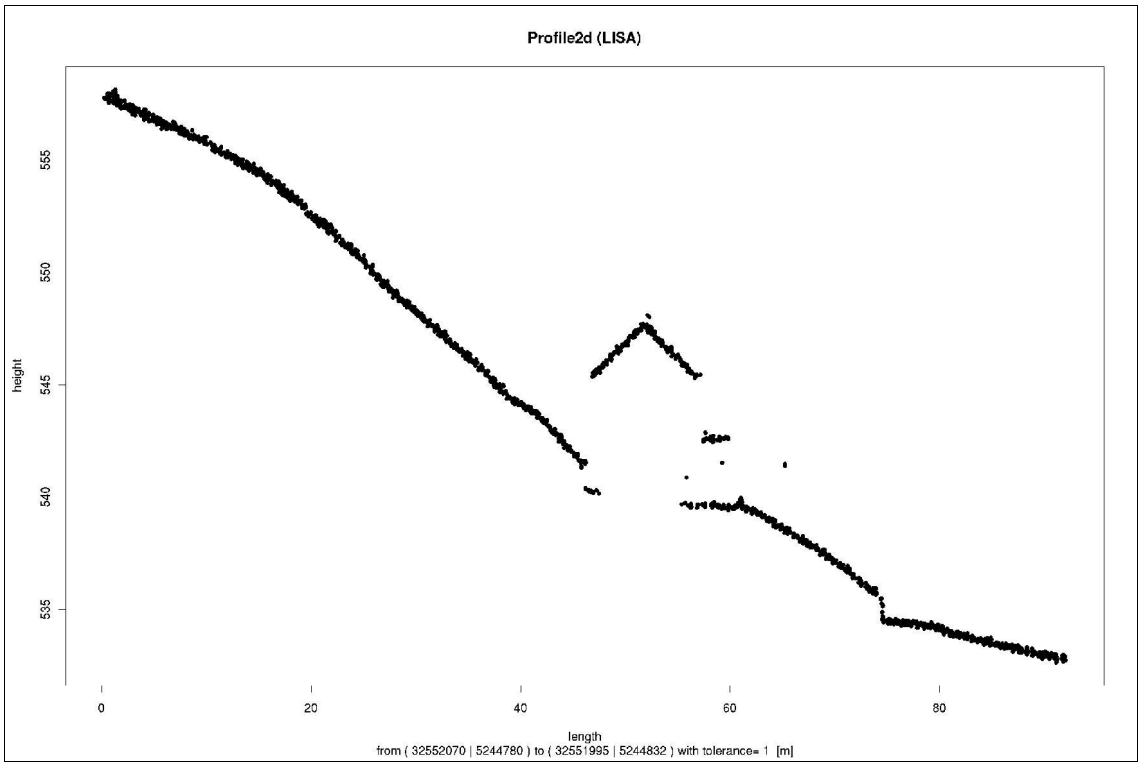


Abb. E.10: Profil 2 mit 1 m Toleranz (eigener Entwurf).

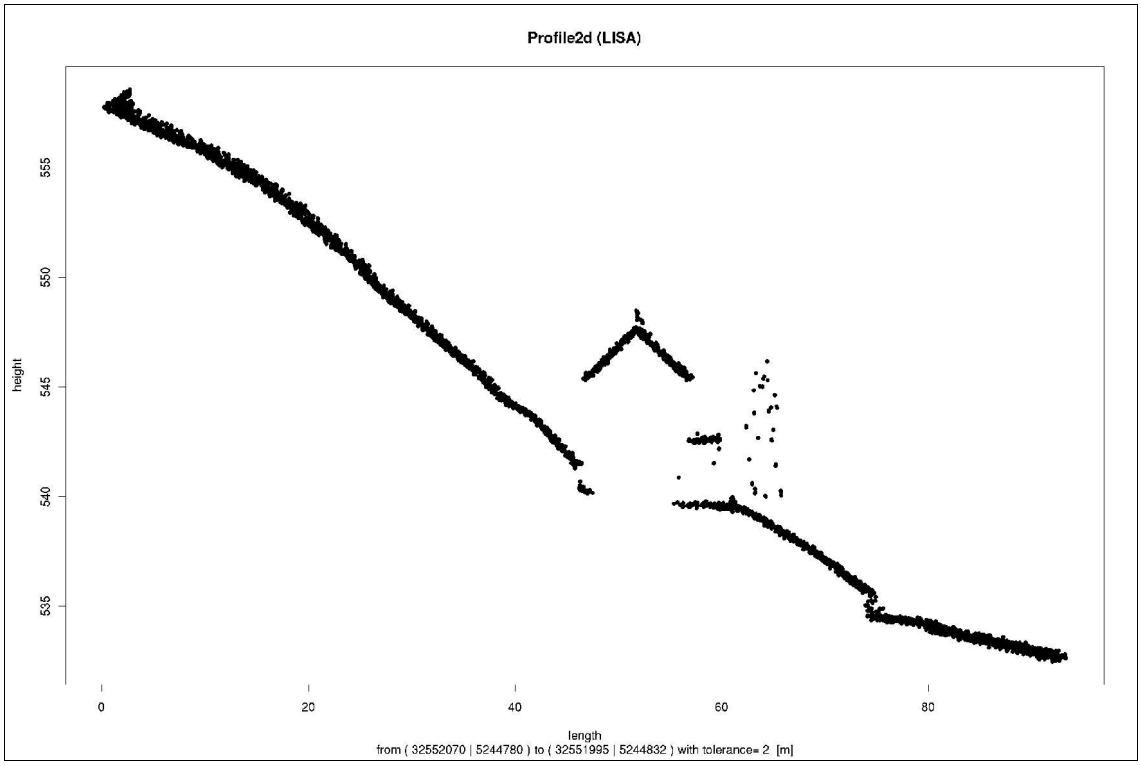


Abb. E.11: Profil 2 mit 2 m Toleranz (eigener Entwurf).

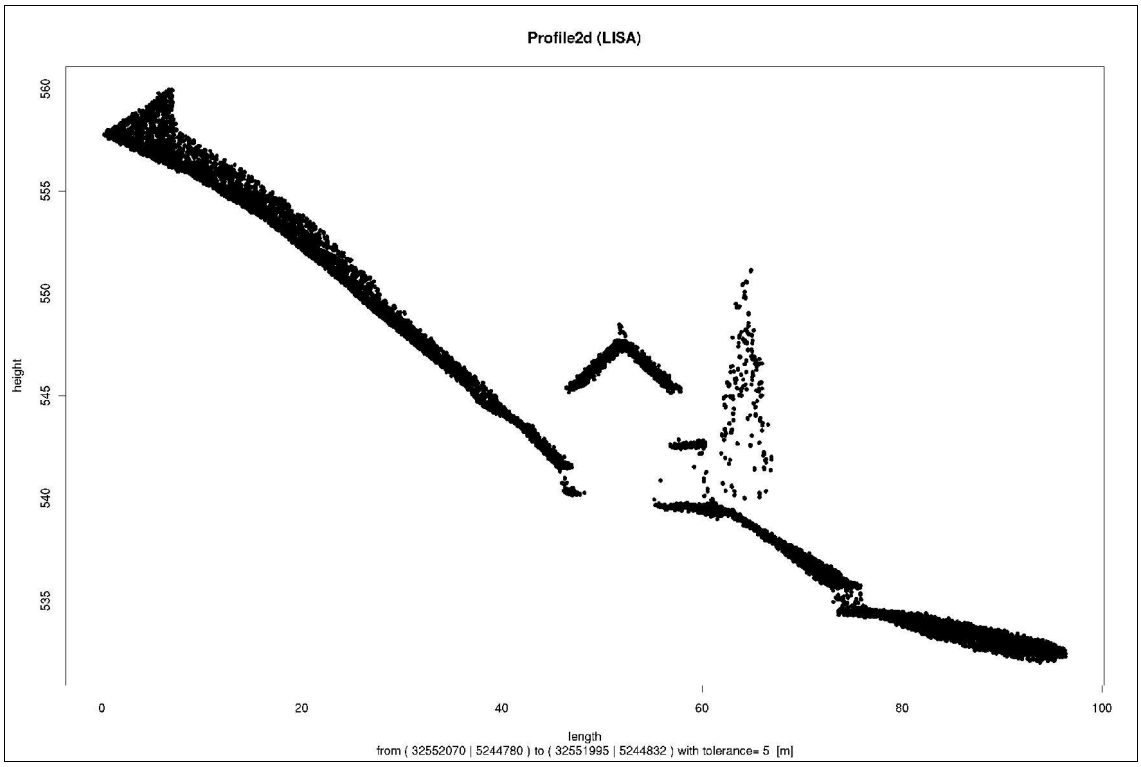


Abb. E.12: Profil 2 mit 5 m Toleranz (eigener Entwurf).

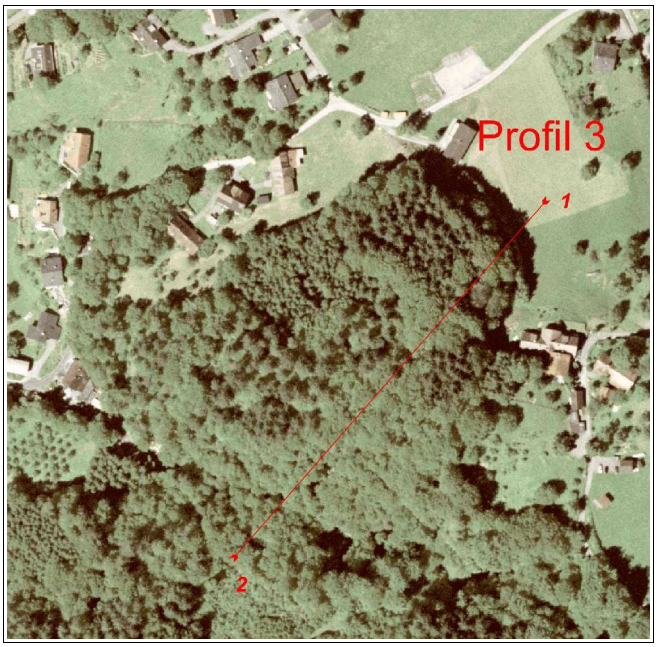


Abb. E.13: Standort des Profils Nr. 3 (eigener Entwurf).

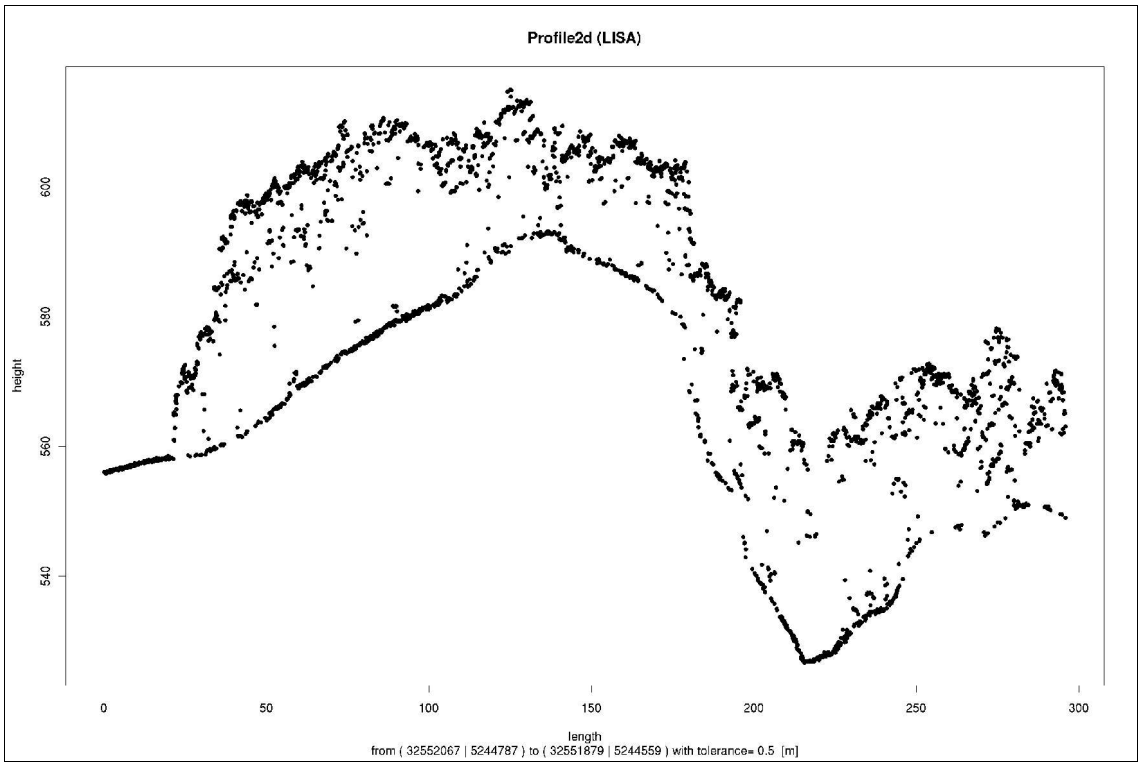


Abb. E.14: Profil 3 mit 0.5 m Toleranz (eigener Entwurf).

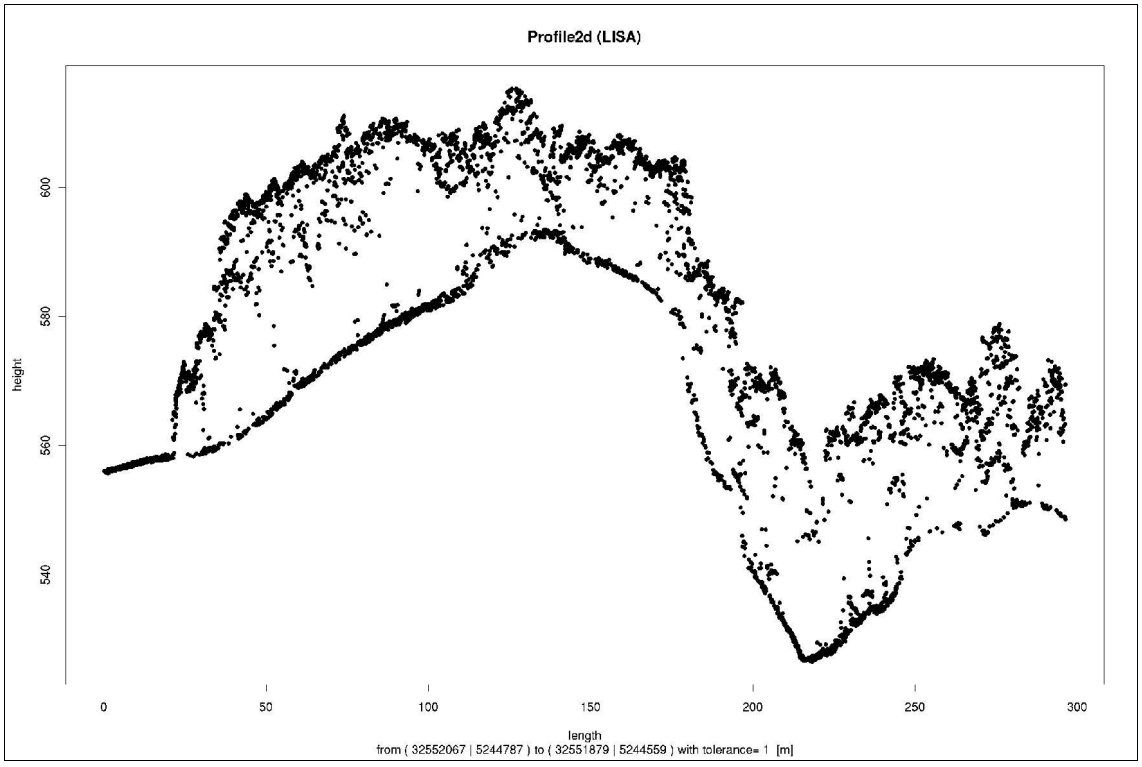


Abb. E.15: Profil 3 mit 1 m Toleranz (eigener Entwurf).

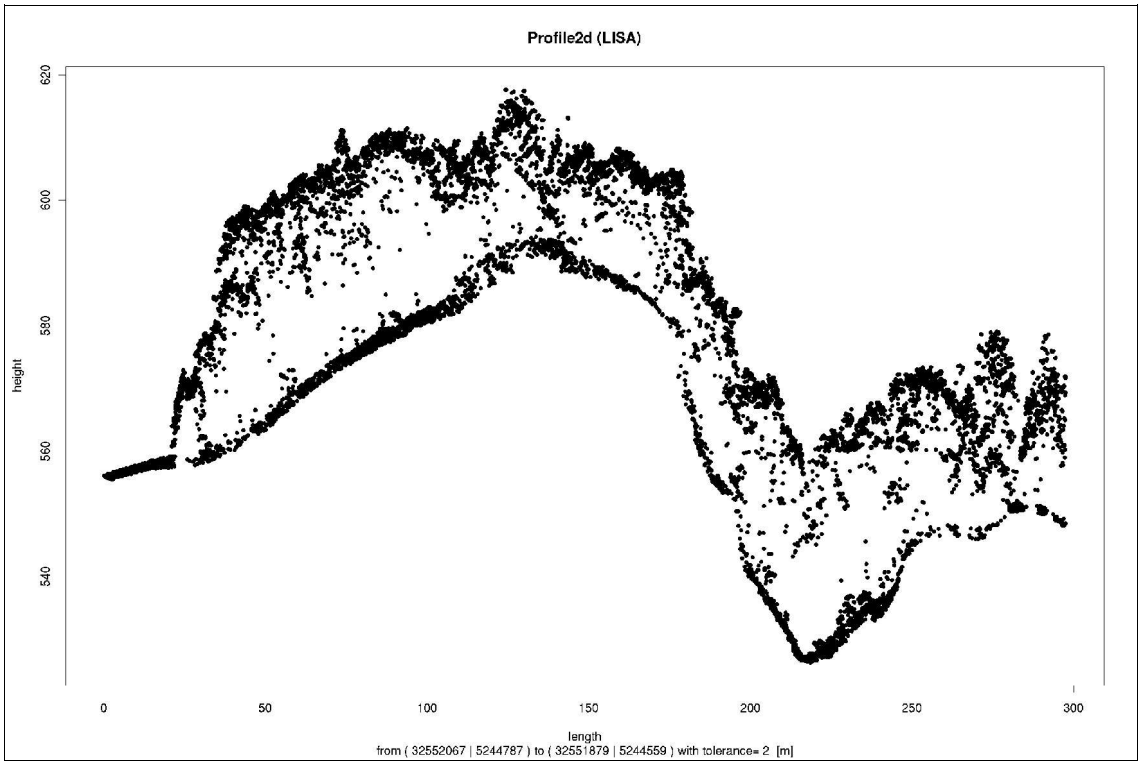


Abb. E.16: Profil 3 mit 2 m Toleranz (eigener Entwurf).

E.3 Dreidimensionale Profile

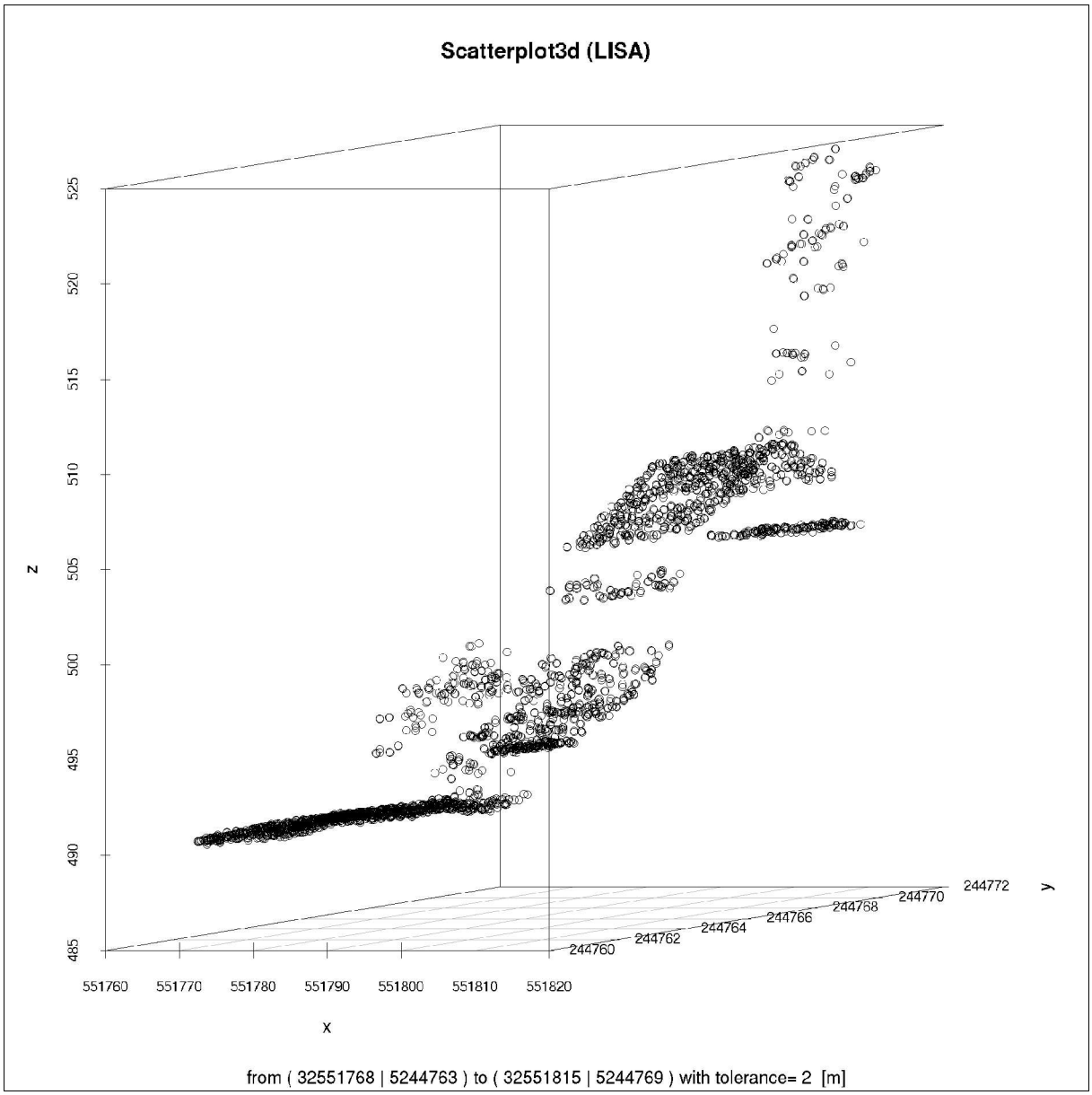


Abb. E.17: 3D-Scatterplot des Profils 1 mit einer Toleranz von 2 m und mit nicht gefüllten Punktsymbolen (eigener Entwurf).

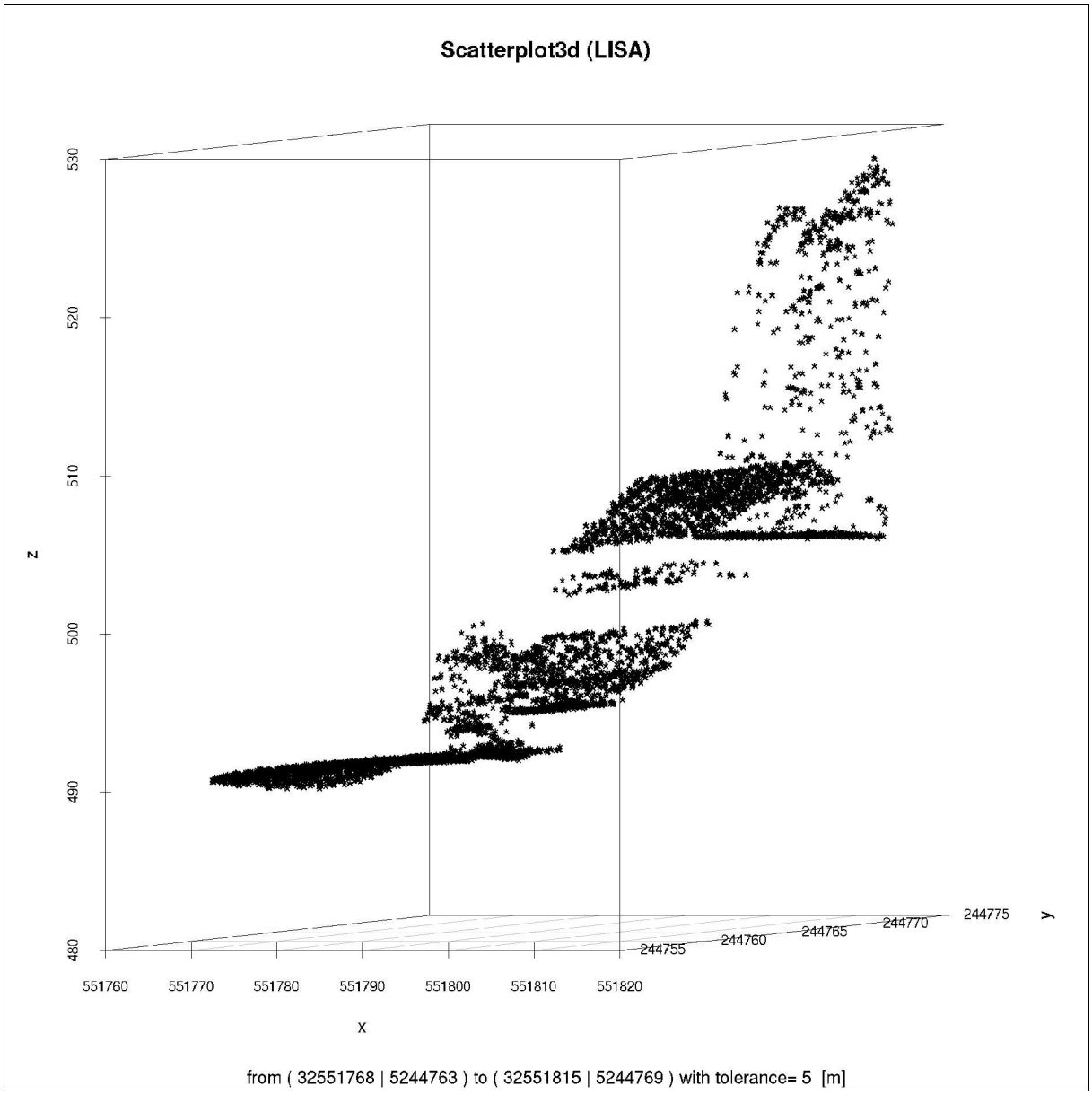


Abb. E.18: 3D-Scatterplot des Profils 1 mit einer Toleranz von 5 m und mit einem Stern (*) als Punktsymbol (eigener Entwurf).

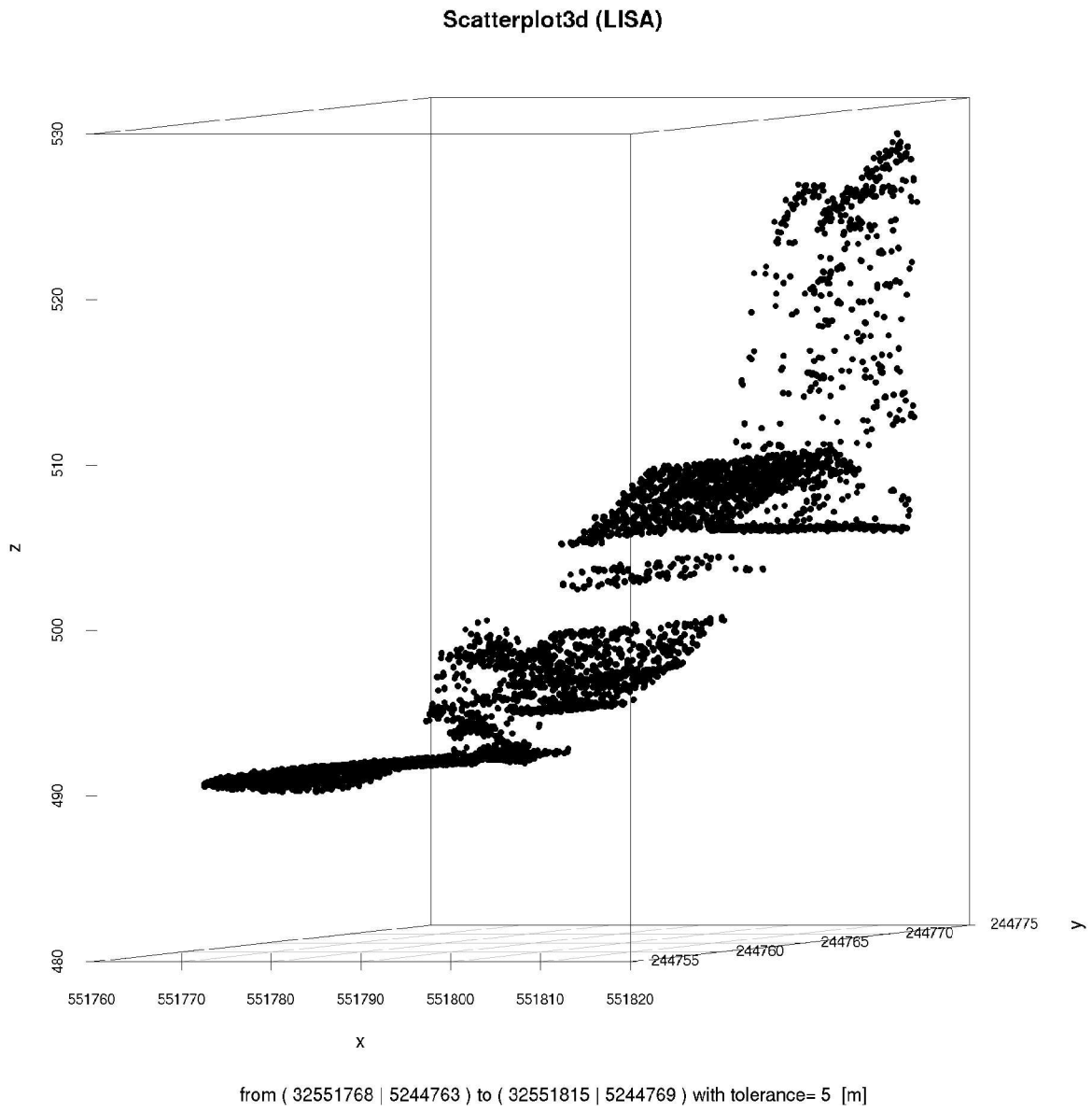


Abb. E.19: 3D-Scatterplot des Profils 1 mit einer Toleranz von 5 m und mit gefüllten Punktsymbolen (eigener Entwurf).

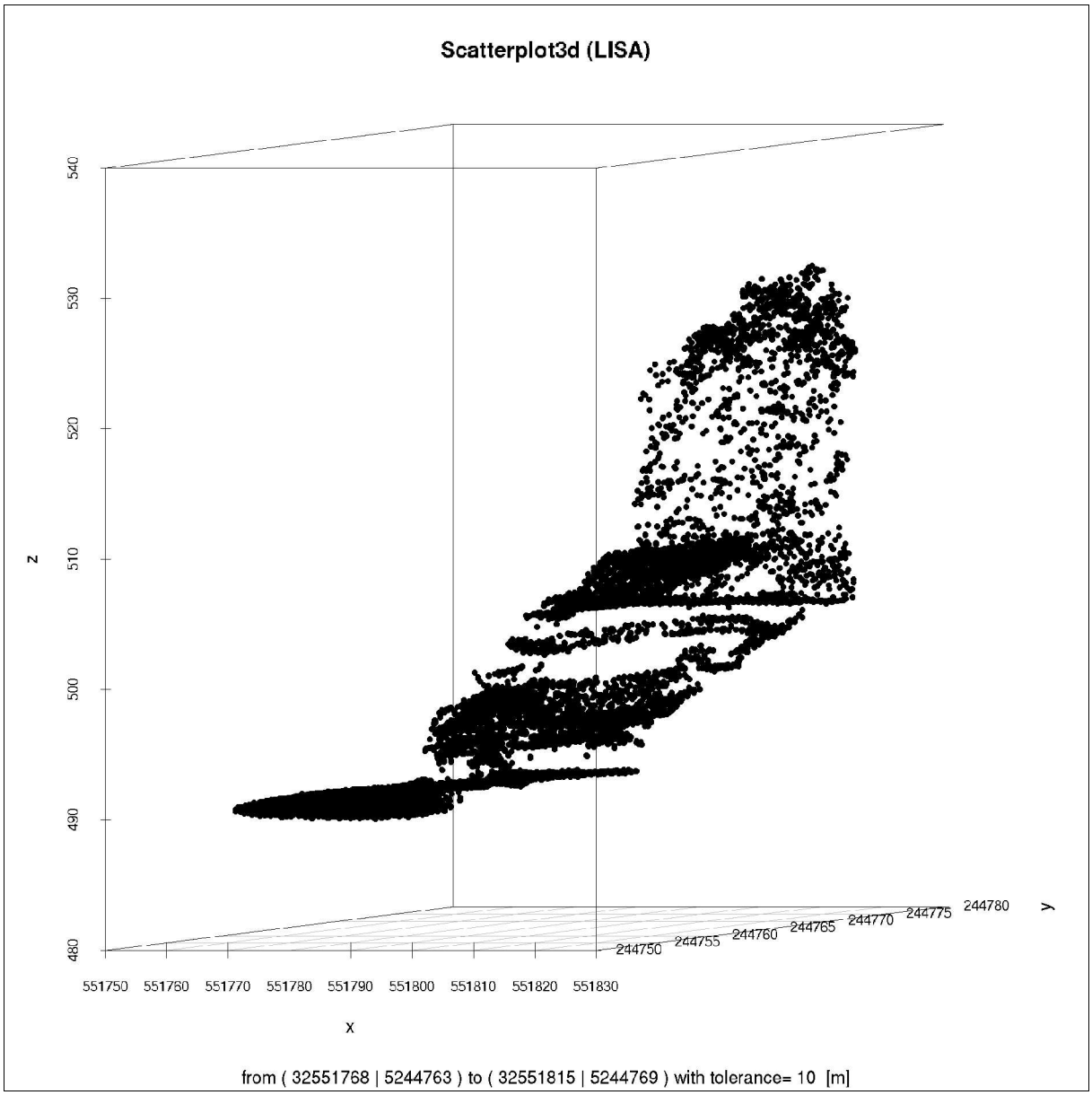


Abb. E.20: 3D-Scatterplot des Profils 1 mit einer Toleranz von 10 m und mit gefüllten Punktsymbolen (eigener Entwurf).

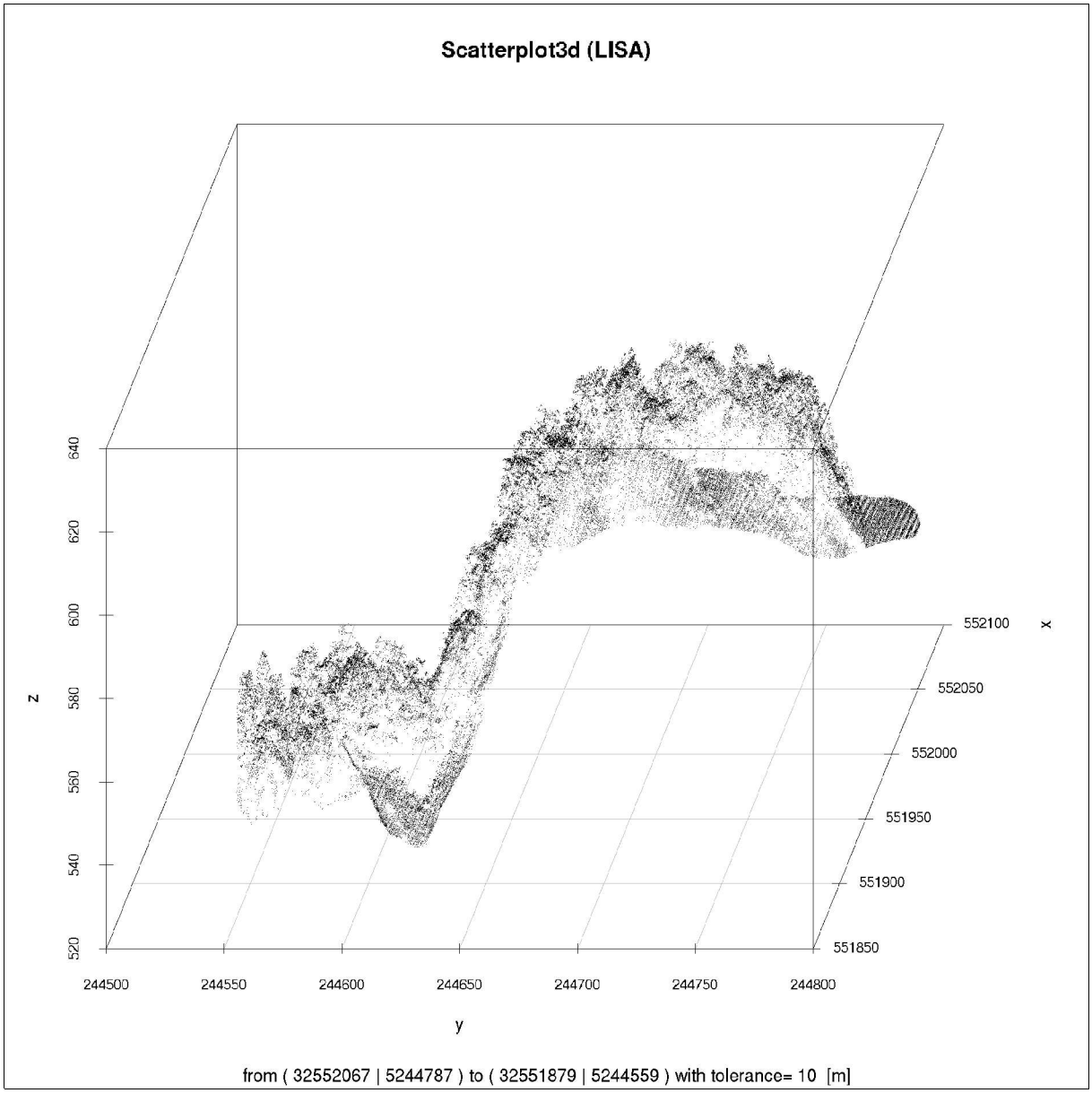
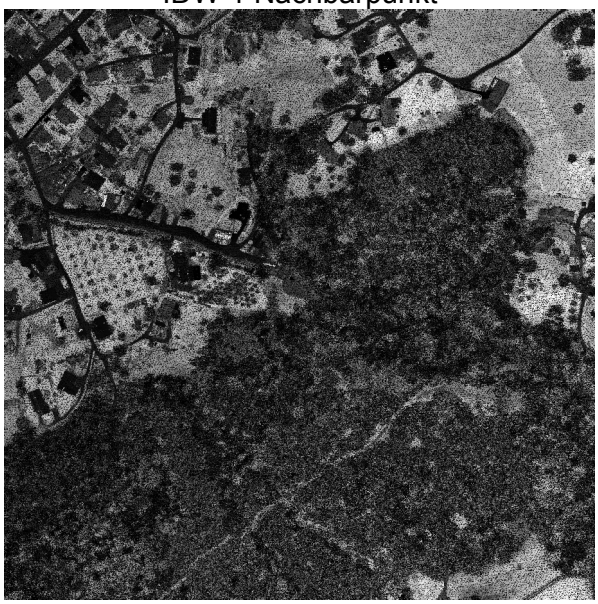


Abb. E.21: 3D-Scatterplot des Profils 3 mit einer Toleranz von 10 m und mit einem Punkt (.) als Punktsymbol (eigener Entwurf).

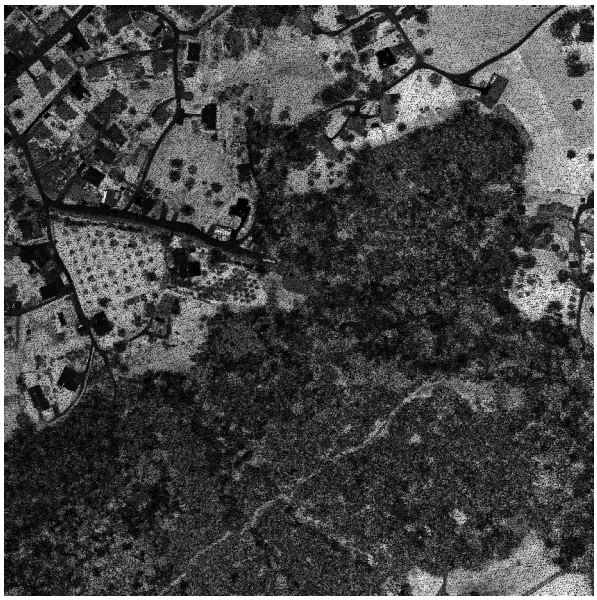
F. Intensitätsraster

alle Laserpunkte (3797446)

IDW 1 Nachbarpunkt



IDW 2



IDW 10

Abb F.1: Intensitätsraster aller Laserpunkte des Untersuchungsgebietes mit unterschiedlichen Einstellungen des IDW-Interpolationsverfahrens (eigener Entwurf).

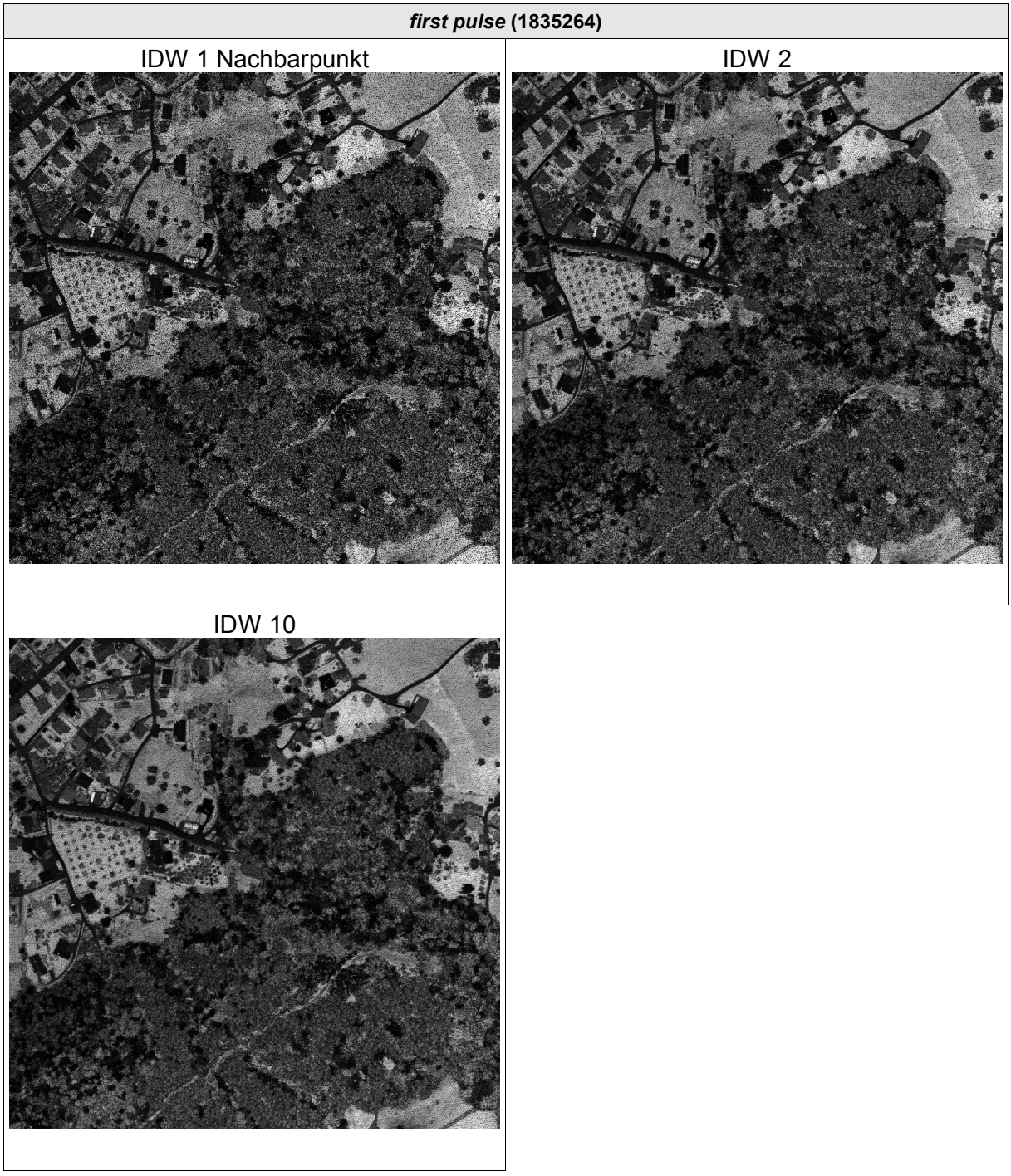


Abb. F.2: Intensitätsraster aller *first pulse* Laserpunkte des Untersuchungsgebietes mit unterschiedlichen Einstellungen des IDW-Interpolationsverfahrens (eigener Entwurf).

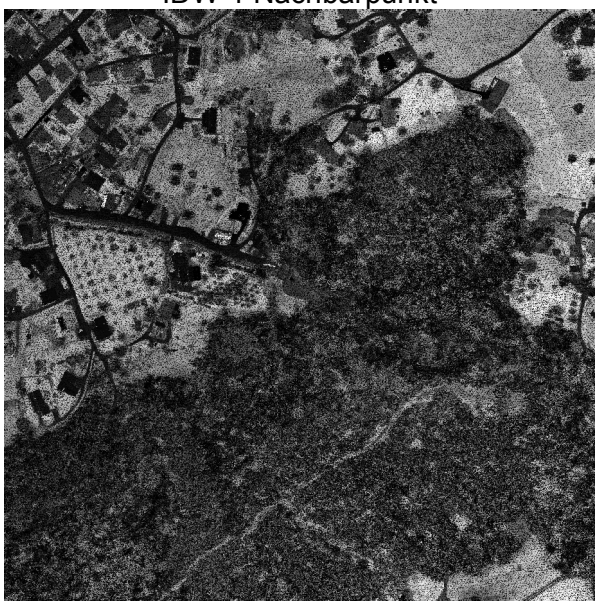
***last pulse* (1962182)**	
IDW 1 Nachbarpunkt	IDW 2
	
IDW 10	
	

Abb. F.3: Intensitätsraster aller *last pulse* Laserpunkte des Untersuchungsgebietes mit unterschiedlichen Einstellungen des IDW-Interpolationsverfahrens (eigener Entwurf).

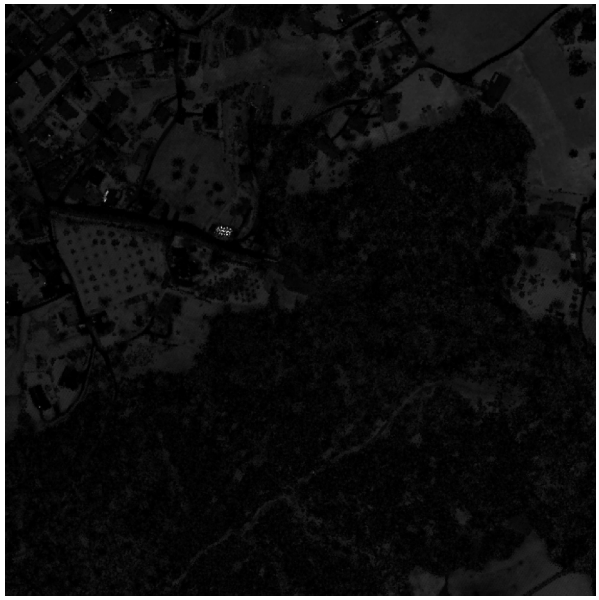
Befliegung ID=1 (738557)	
IDW 1 Nachbarpunkt mit modifizierter Farbskala	IDW 2 mit modifizierter Farbskala
	
IDW 10 mit modifizierter Farbskala	IDW 10 ohne modifizierter Farbskala
	

Abb. F.4: Intensitätsraster aller Laserpunkte des Untersuchungsgebietes der Befliegung ID=1 mit unterschiedlichen Einstellungen des IDW-Interpolationsverfahrens (eigener Entwurf).

Befliegung ID=2 (3058889)

IDW 1 Nachbarpunkt

IDW 2

IDW 10

Abb. F.5: Intensitätsraster aller Laserpunkte des Untersuchungsgebietes der Befliegung ID=2 mit unterschiedlichen Einstellungen des IDW-Interpolationsverfahrens (eigener Entwurf).

G Lizenzen

G.1 PostgreSQL

www.postgresql.org/about/licence:

```
PostgreSQL is released under the BSD license.
PostgreSQL Database Management System
(formerly known as Postgres, then as Postgres95)

Portions Copyright (c) 1996-2002, The PostgreSQL Global Development Group

Portions Copyright (c) 1994, The Regents of the University of California

Permission to use, copy, modify, and distribute this software and its documentation
for any purpose, without fee, and without a written agreement is hereby granted,
provided that the above copyright notice and this paragraph and the following two
paragraphs appear in all copies.

IN NO EVENT SHALL THE UNIVERSITY OF CALIFORNIA BE LIABLE TO ANY PARTY FOR DIRECT,
INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES, INCLUDING LOST PROFITS,
ARISING OUT OF THE USE OF THIS SOFTWARE AND ITS DOCUMENTATION, EVEN IF THE
UNIVERSITY OF CALIFORNIA HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

THE UNIVERSITY OF CALIFORNIA SPECIFICALLY DISCLAIMS ANY WARRANTIES, INCLUDING, BUT
NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A
PARTICULAR PURPOSE. THE SOFTWARE PROVIDED HEREUNDER IS ON AN "AS IS" BASIS, AND THE
UNIVERSITY OF CALIFORNIA HAS NO OBLIGATIONS TO PROVIDE MAINTENANCE, SUPPORT,
UPDATES, ENHANCEMENTS, OR MODIFICATIONS.
```

G.2 GRASS und R

GRASS und R unterliegen der GNU General Public License (www.gnu.org/copyleft/gpl.html , www.gnu.de/gpl-ger.html):

```
GNU General Public License
Deutsche Übersetzung der Version 2, Juni 1991

Copyright © 1989, 1991 Free Software Foundation, Inc.
59 Temple Place, Suite 330, Boston, MA 02111-1307, USA

Es ist jedermann gestattet, diese Lizenzurkunde zu vervielfältigen und unveränderte
Kopien zu verbreiten; Änderungen sind jedoch nicht erlaubt.

Diese Übersetzung ist kein rechtskräftiger Ersatz für die englischsprachige
Originalversion!
```

Vorwort
Die meisten Softwarelizenzen sind daraufhin entworfen worden, Ihnen die Freiheit zu nehmen, die Software weiterzugeben und zu verändern. Im Gegensatz dazu soll Ihnen die GNU General Public License, die Allgemeine Öffentliche GNU-Lizenz, ebendiese Freiheit garantieren. Sie soll sicherstellen, daß die Software für alle Benutzer frei ist. Diese Lizenz gilt für den Großteil der von der Free Software Foundation herausgegebenen Software und für alle anderen Programme, deren Autoren ihr Datenwerk dieser Lizenz unterstellt haben. Auch Sie können diese Möglichkeit der Lizenzierung für Ihre Programme anwenden. (Ein anderer Teil der Software der Free Software Foundation unterliegt stattdessen der GNU Library General Public License, der Allgemeinen Öffentlichen GNU-Lizenz für Bibliotheken.) [Mittlerweile wurde die GNU Library Public License von der GNU Lesser Public License abgelöst – Anmerkung des Übersetzers.]

Die Bezeichnung „freie" Software bezieht sich auf Freiheit, nicht auf den Preis. Unsere Lizenzen sollen Ihnen die Freiheit garantieren, Kopien freier Software zu verbreiten (und etwas für diesen Service zu berechnen, wenn Sie möchten), die Möglichkeit, die Software im Quelltext zu erhalten oder den Quelltext auf Wunsch zu bekommen. Die Lizenzen sollen garantieren, daß Sie die Software ändern oder Teile davon in neuen freien Programmen verwenden dürfen – und daß Sie wissen, daß Sie dies alles tun dürfen.

Um Ihre Rechte zu schützen, müssen wir Einschränkungen machen, die es jedem verbieten, Ihnen diese Rechte zu verweigern oder Sie aufzufordern, auf diese Rechte zu verzichten. Aus diesen Einschränkungen folgen bestimmte Verantwortlichkeiten für Sie, wenn Sie Kopien der Software verbreiten oder sie verändern.

Beispielsweise müssen Sie den Empfängern alle Rechte gewähren, die Sie selbst haben, wenn Sie – kostenlos oder gegen Bezahlung – Kopien eines solchen Programms verbreiten. Sie müssen sicherstellen, daß auch die Empfänger den Quelltext erhalten bzw. erhalten können. Und Sie müssen ihnen diese Bedingungen zeigen, damit sie ihre Rechte kennen.

Wir schützen Ihre Rechte in zwei Schritten: (1) Wir stellen die Software unter ein Urheberrecht (Copyright), und (2) wir bieten Ihnen diese Lizenz an, die Ihnen das Recht gibt, die Software zu vervielfältigen, zu verbreiten und/oder zu verändern.

Um die Autoren und uns zu schützen, wollen wir darüberhinaus sicherstellen, daß jeder erfährt, daß für diese freie Software keinerlei Garantie besteht. Wenn die Software von jemand anderem modifiziert und weitergegeben wird, möchten wir, daß die Empfänger wissen, daß sie nicht das Original erhalten haben, damit irgendwelche von anderen verursachte Probleme nicht den Ruf des ursprünglichen Autors schädigen.

Schließlich und endlich ist jedes freie Programm permanent durch Software-Patente bedroht. Wir möchten die Gefahr ausschließen, daß Distributoren eines freien Programms individuell Patente lizensieren – mit dem Ergebnis, daß das Programm proprietär würde. Um dies zu verhindern, haben wir klargestellt, daß jedes Patent entweder für freie Benutzung durch jedermann lizenziert werden muß oder überhaupt nicht lizenziert werden darf.

Es folgen die genauen Bedingungen für die Vervielfältigung, Verbreitung und Bearbeitung:

Allgemeine Öffentliche GNU-Lizenz
Bedingungen für die Vervielfältigung,
Verbreitung und Bearbeitung
§0. Diese Lizenz gilt für jedes Programm und jedes andere Datenwerk, in dem ein entsprechender Vermerk des Copyright-Inhabers darauf hinweist, daß das Datenwerk unter den Bestimmungen dieser General Public License verbreitet werden darf. Im folgenden wird jedes derartige Programm oder Datenwerk als „das Programm" bezeichnet; die Formulierung „auf dem Programm basierendes Datenwerk" bezeichnet das Programm sowie jegliche Bearbeitung des Programms im urheberrechtlichen Sinne, also ein Datenwerk, welches das Programm, auch auszugsweise, sei es unverändert oder verändert und/oder in eine andere Sprache übersetzt, enthält. (Im folgenden wird die Übersetzung ohne Einschränkung als „Bearbeitung" eingestuft.) Jeder Lizenznehmer wird im folgenden als „Sie" angesprochen.

Andere Handlungen als Vervielfältigung, Verbreitung und Bearbeitung werden von dieser Lizenz nicht berührt; sie fallen nicht in ihren Anwendungsbereich. Der Vorgang der Ausführung des Programms wird nicht eingeschränkt, und die Ausgaben des Programms unterliegen dieser Lizenz nur, wenn der Inhalt ein auf dem Programm basierendes Datenwerk darstellt (unabhängig davon, daß die Ausgabe durch die Ausführung des Programmes erfolgte). Ob dies zutrifft, hängt von den Funktionen des Programms ab.

§1. Sie dürfen auf beliebigen Medien unveränderte Kopien des Quelltextes des Programms, wie sie ihn erhalten haben, anfertigen und verbreiten. Voraussetzung hierfür ist, daß Sie mit jeder Kopie einen entsprechenden Copyright-Vermerk sowie einen Haftungsausschluß veröffentlichen, alle Vermerke, die sich auf diese Lizenz und das Fehlen einer Garantie beziehen, unverändert lassen und desweiteren allen anderen Empfängern des Programms zusammen mit dem Programm eine Kopie dieser Lizenz zukommen lassen.

Sie dürfen für den eigentlichen Kopiervorgang eine Gebühr verlangen. Wenn Sie es wünschen, dürfen Sie auch gegen Entgelt eine Garantie für das Programm anbieten.

§2. Sie dürfen Ihre Kopie(n) des Programms oder eines Teils davon verändern, wodurch ein auf dem Programm basierendes Datenwerk entsteht; Sie dürfen derartige Bearbeitungen unter den Bestimmungen von Paragraph 1 vervielfältigen und verbreiten, vorausgesetzt, daß zusätzlich alle im folgenden genannten Bedingungen erfüllt werden:

Sie müssen die veränderten Dateien mit einem auffälligen Vermerk versehen, der auf die von Ihnen vorgenommene Modifizierung und das Datum jeder Änderung hinweist.

Sie müssen dafür sorgen, daß jede von Ihnen verbreitete oder veröffentlichte Arbeit, die ganz oder teilweise von dem Programm oder Teilen davon abgeleitet ist, Dritten gegenüber als Ganzes unter den Bedingungen dieser Lizenz ohne Lizenzgebühren zur Verfügung gestellt wird.

Wenn das veränderte Programm normalerweise bei der Ausführung interaktiv Kommandos einliest, müssen Sie dafür sorgen, daß es, wenn es auf dem üblichsten Wege für solche interaktive Nutzung gestartet wird, eine Meldung ausgibt oder ausdruckt, die einen geeigneten Copyright-Vermerk enthält sowie einen Hinweis, daß es keine Gewährleistung gibt (oder anderenfalls, daß Sie Garantie leisten), und daß die Benutzer das Programm unter diesen Bedingungen weiter verbreiten dürfen. Auch muß der Benutzer darauf hingewiesen werden, wie er eine Kopie dieser Lizenz ansehen

kann. (Ausnahme: Wenn das Programm selbst interaktiv arbeitet, aber normalerweise keine derartige Meldung ausgibt, muß Ihr auf dem Programm basierendes Datenwerk auch keine solche Meldung ausgeben).

Diese Anforderungen gelten für das bearbeitete Datenwerk als Ganzes. Wenn identifizierbare Teile des Datenwerkes nicht von dem Programm abgeleitet sind und vernünftigerweise als unabhängige und eigenständige Datenwerke für sich selbst zu betrachten sind, dann gelten diese Lizenz und ihre Bedingungen nicht für die betroffenen Teile, wenn Sie diese als eigenständige Datenwerke weitergeben. Wenn Sie jedoch dieselben Abschnitte als Teil eines Ganzen weitergeben, das ein auf dem Programm basierendes Datenwerk darstellt, dann muß die Weitergabe des Ganzen nach den Bedingungen dieser Lizenz erfolgen, deren Bedingungen für weitere Lizenznehmer somit auf das gesamte Ganze ausgedehnt werden – und somit auf jeden einzelnen Teil, unabhängig vom jeweiligen Autor.

Somit ist es nicht die Absicht dieses Abschnittes, Rechte für Datenwerke in Anspruch zu nehmen oder Ihnen die Rechte für Datenwerke streitig zu machen, die komplett von Ihnen geschrieben wurden; vielmehr ist es die Absicht, die Rechte zur Kontrolle der Verbreitung von Datenwerken, die auf dem Programm basieren oder unter seiner auszugsweisen Verwendung zusammengestellt worden sind, auszuüben.

Ferner bringt auch das einfache Zusammenlegen eines anderen Datenwerkes, das nicht auf dem Programm basiert, mit dem Programm oder einem auf dem Programm basierenden Datenwerk auf ein- und demselben Speicher- oder Vertriebsmedium dieses andere Datenwerk nicht in den Anwendungsbereich dieser Lizenz.

§3. Sie dürfen das Programm (oder ein darauf basierendes Datenwerk gemäß Paragraph 2) als Objectcode oder in ausführbarer Form unter den Bedingungen der Paragraphen 1 und 2 kopieren und weitergeben – vorausgesetzt, daß Sie außerdem eine der folgenden Leistungen erbringen:

Liefern Sie das Programm zusammen mit dem vollständigen zugehörigen maschinenlesbaren Quelltext auf einem für den Datenaustausch üblichen Medium aus, wobei die Verteilung unter den Bedingungen der Paragraphen 1 und 2 erfolgen muß. Oder:

Liefern Sie das Programm zusammen mit einem mindestens drei Jahre lang gültigen schriftlichen Angebot aus, jedem Dritten eine vollständige maschinenlesbare Kopie des Quelltextes zur Verfügung zu stellen – zu nicht höheren Kosten als denen, die durch den physikalischen Kopiervorgang anfallen –, wobei der Quelltext unter den Bedingungen der Paragraphen 1 und 2 auf einem für den Datenaustausch üblichen Medium weitergegeben wird. Oder:

Liefern Sie das Programm zusammen mit dem schriftlichen Angebot der Zurverfügungstellung des Quelltextes aus, das Sie selbst erhalten haben. (Diese Alternative ist nur für nicht-kommerzielle Verbreitung zulässig und nur, wenn Sie das Programm als Objectcode oder in ausführbarer Form mit einem entsprechenden Angebot gemäß Absatz b erhalten haben.)

Unter dem Quelltext eines Datenwerkes wird diejenige Form des Datenwerkes verstanden, die für Bearbeitungen vorzugsweise verwendet wird. Für ein ausführbares Programm bedeutet „der komplette Quelltext“: Der Quelltext aller im Programm enthaltenen Module einschließlich aller zugehörigen Modulschnittstellen-Definitionsdateien sowie der zur Compilation und Installation verwendeten Skripte.

Als besondere Ausnahme jedoch braucht der verteilte Quelltext nichts von dem zu enthalten, was üblicherweise (entweder als Quelltext oder in binärer Form) zusammen mit den Hauptkomponenten des Betriebssystems (Kernel, Compiler usw.) geliefert wird, unter dem das Programm läuft – es sei denn, diese Komponente selbst gehört zum ausführbaren Programm.

Wenn die Verbreitung eines ausführbaren Programms oder von Objectcode dadurch erfolgt, daß der Kopierzugriff auf eine dafür vorgesehene Stelle gewährt wird, so gilt die Gewährung eines gleichwertigen Zugriffs auf den Quelltext als Verbreitung des Quelltextes, auch wenn Dritte nicht dazu gezwungen sind, den Quelltext zusammen mit dem Objectcode zu kopieren.

§4. Sie dürfen das Programm nicht vervielfältigen, verändern, weiter lizenzieren oder verbreiten, sofern es nicht durch diese Lizenz ausdrücklich gestattet ist. Jeder anderweitige Versuch der Vervielfältigung, Modifizierung, Weiterlizenzierung und Verbreitung ist nichtig und beendet automatisch Ihre Rechte unter dieser Lizenz. Jedoch werden die Lizenzen Dritter, die von Ihnen Kopien oder Rechte unter dieser Lizenz erhalten haben, nicht beendet, solange diese die Lizenz voll anerkennen und befolgen.

§5. Sie sind nicht verpflichtet, diese Lizenz anzunehmen, da Sie sie nicht unterzeichnet haben. Jedoch gibt Ihnen nichts anderes die Erlaubnis, das Programm oder von ihm abgeleitete Datenwerke zu verändern oder zu verbreiten. Diese Handlungen sind gesetzlich verboten, wenn Sie diese Lizenz nicht anerkennen. Indem Sie das Programm (oder ein darauf basierendes Datenwerk) verändern oder verbreiten, erklären Sie Ihr Einverständnis mit dieser Lizenz und mit allen ihren Bedingungen bezüglich der Vervielfältigung, Verbreitung und Veränderung des Programms oder eines darauf basierenden Datenwerks.

§6. Jedesmal, wenn Sie das Programm (oder ein auf dem Programm basierendes Datenwerk) weitergeben, erhält der Empfänger automatisch vom ursprünglichen Lizenzgeber die Lizenz, das Programm entsprechend den hier festgelegten Bestimmungen zu vervielfältigen, zu verbreiten und zu verändern. Sie dürfen keine weiteren Einschränkungen der Durchsetzung der hierin zugestandenen Rechte des Empfängers vornehmen. Sie sind nicht dafür verantwortlich, die Einhaltung dieser Lizenz durch Dritte durchzusetzen.

§7. Sollten Ihnen infolge eines Gerichtsurteils, des Vorwurfs einer Patentverletzung oder aus einem anderen Grunde (nicht auf Patentfragen begrenzt) Bedingungen (durch Gerichtsbeschluß, Vergleich oder anderweitig) auferlegt werden, die den Bedingungen dieser Lizenz widersprechen, so befreien Sie diese Umstände nicht von den Bestimmungen dieser Lizenz. Wenn es Ihnen nicht möglich ist, das Programm unter gleichzeitiger Beachtung der Bedingungen in dieser Lizenz und Ihrer anderweitigen Verpflichtungen zu verbreiten, dann dürfen Sie als Folge das Programm überhaupt nicht verbreiten. Wenn zum Beispiel ein Patent nicht die gebührenfreie Weiterverbreitung des Programms durch diejenigen erlaubt, die das Programm direkt oder indirekt von Ihnen erhalten haben, dann besteht der einzige Weg, sowohl das Patentrecht als auch diese Lizenz zu befolgen, darin, ganz auf die Verbreitung des Programms zu verzichten.

Sollte sich ein Teil dieses Paragraphen als ungültig oder unter bestimmten Umständen nicht durchsetzbar erweisen, so soll dieser Paragraph seinem Sinne nach angewandt werden; im übrigen soll dieser Paragraph als Ganzes gelten.

Zweck dieses Paragraphen ist nicht, Sie dazu zu bringen, irgendwelche Patente oder andere Eigentumsansprüche zu verletzen oder die Gültigkeit solcher Ansprüche zu bestreiten; dieser Paragraph hat einzig den Zweck, die Integrität des Verbreitungssystems der freien Software zu schützen, das durch die Praxis öffentlicher Lizenzen verwirklicht wird. Viele Leute haben großzügige Beiträge zu dem großen Angebot der mit diesem System verbreiteten Software im Vertrauen auf die konsistente Anwendung dieses Systems geleistet; es liegt am Autor/Geber, zu entscheiden, ob er die Software mittels irgendeines anderen Systems verbreiten will; ein Lizenznehmer hat auf diese Entscheidung keinen Einfluß.

Dieser Paragraph ist dazu gedacht, deutlich klarzustellen, was als Konsequenz aus dem Rest dieser Lizenz betrachtet wird.

§8. Wenn die Verbreitung und/oder die Benutzung des Programms in bestimmten Staaten entweder durch Patente oder durch urheberrechtlich geschützte Schnittstellen eingeschränkt ist, kann der Urheberrechtsinhaber, der das Programm unter diese Lizenz gestellt hat, eine explizite geographische Begrenzung der Verbreitung angeben, in der diese Staaten ausgeschlossen werden, so daß die Verbreitung nur innerhalb und zwischen den Staaten erlaubt ist, die nicht ausgeschlossen sind. In einem solchen Fall beinhaltet diese Lizenz die Beschränkung, als wäre sie in diesem Text niedergeschrieben.

§9. Die Free Software Foundation kann von Zeit zu Zeit überarbeitete und/oder neue Versionen der General Public License veröffentlichen. Solche neuen Versionen werden vom Grundprinzip her der gegenwärtigen entsprechen, können aber im Detail abweichen, um neuen Problemen und Anforderungen gerecht zu werden.

Jede Version dieser Lizenz hat eine eindeutige Versionsnummer. Wenn in einem Programm angegeben wird, daß es dieser Lizenz in einer bestimmten Versionsnummer oder „jeder späteren Version“ (“any later version”) unterliegt, so haben Sie die Wahl, entweder den Bestimmungen der genannten Version zu folgen oder denen jeder beliebigen späteren Version, die von der Free Software Foundation veröffentlicht wurde. Wenn das Programm keine Versionsnummer angibt, können Sie eine beliebige Version wählen, die je von der Free Software Foundation veröffentlicht wurde.

§10. Wenn Sie den Wunsch haben, Teile des Programms in anderen freien Programmen zu verwenden, deren Bedingungen für die Verbreitung anders sind, schreiben Sie an den Autor, um ihn um die Erlaubnis zu bitten. Für Software, die unter dem Copyright der Free Software Foundation steht, schreiben Sie an die Free Software Foundation; wir machen zu diesem Zweck gelegentlich Ausnahmen. Unsere Entscheidung wird von den beiden Zielen geleitet werden, zum einen den freien Status aller von unserer freien Software abgeleiteten Datenwerke zu erhalten und zum anderen das gemeinschaftliche Nutzen und Wiederverwenden von Software im allgemeinen zu fördern.

Keine Gewährleistung
§11. Da das Programm ohne jegliche Kosten lizenziert wird, besteht keinerlei Gewährleistung für das Programm, soweit dies gesetzlich zulässig ist. Sofern nicht anderweitig schriftlich bestätigt, stellen die Copyright-Inhaber und/oder Dritte das Programm so zur Verfügung, „wie es ist“, ohne irgendeine Gewährleistung, weder ausdrücklich noch implizit, einschließlich – aber nicht begrenzt auf – Marktreife oder Verwendbarkeit für einen bestimmten Zweck. Das volle Risiko bezüglich Qualität und Leistungsfähigkeit des Programms liegt bei Ihnen. Sollte sich das Programm als fehlerhaft herausstellen, liegen die Kosten für notwendigen Service, Reparatur oder Korrektur bei Ihnen.

```
§12. In keinem Fall, außer wenn durch geltendes Recht gefordert oder schriftlich
zugesichert, ist irgendein Copyright-Inhaber oder irgendein Dritter, der das
Programm wie oben erlaubt modifiziert oder verbreitet hat, Ihnen gegenüber für
irgendwelche Schäden haftbar, einschließlich jeglicher allgemeiner oder spezieller
Schäden, Schäden durch Seiteneffekte (Nebenwirkungen) oder Folgeschäden, die aus
der Benutzung des Programms oder der Unbenutzbarkeit des Programms folgen
(einschließlich – aber nicht beschränkt auf – Datenverluste, fehlerhafte
Verarbeitung von Daten, Verluste, die von Ihnen oder anderen getragen werden
müssen, oder dem Unvermögen des Programms, mit irgendeinem anderen Programm
zusammenzuarbeiten), selbst wenn ein Copyright-Inhaber oder Dritter über die
Möglichkeit solcher Schäden unterrichtet worden war.

Ende der Bedingungen
```

G.3 Python

www.python.org/doc/Copyright.html:

```
A. HISTORY OF THE SOFTWARE
==========================

Python was created in the early 1990s by Guido van Rossum at Stichting
Mathematisch Centrum (CWI, see http://www.cwi.nl) in the Netherlands
as a successor of a language called ABC.  Guido remains Python's
principal author, although it includes many contributions from others.

In 1995, Guido continued his work on Python at the Corporation for
National Research Initiatives (CNRI, see http://www.cnri.reston.va.us)
in Reston, Virginia where he released several versions of the
software.

In May 2000, Guido and the Python core development team moved to
BeOpen.com to form the BeOpen PythonLabs team.  In October of the same
year, the PythonLabs team moved to Digital Creations (now Zope
Corporation, see http://www.zope.com).  In 2001, the Python Software
Foundation (PSF, see http://www.python.org/psf/) was formed, a
non-profit organization created specifically to own Python-related
Intellectual Property.  Zope Corporation is a sponsoring member of
the PSF.

All Python releases are Open Source (see http://www.opensource.org for
the Open Source Definition).  Historically, most, but not all, Python
releases have also been GPL-compatible; the table below summarizes
the various releases.

    Release         Derived     Year        Owner       GPL-
                    from                                compatible? (1)

    0.9.0 thru 1.2              1991-1995   CWI         yes
    1.3 thru 1.5.2  1.2         1995-1999   CNRI        yes
```

```
    1.6             1.5.2       2000        CNRI        no
    2.0             1.6         2000        BeOpen.com  no
    1.6.1           1.6         2001        CNRI        yes (2)
    2.1             2.0+1.6.1   2001        PSF         no
    2.0.1           2.0+1.6.1   2001        PSF         yes
    2.1.1           2.1+2.0.1   2001        PSF         yes
    2.2             2.1.1       2001        PSF         yes
    2.1.2           2.1.1       2002        PSF         yes
    2.1.3           2.1.2       2002        PSF         yes
    2.2.1           2.2         2002        PSF         yes
    2.2.2           2.2.1       2002        PSF         yes
    2.2.3           2.2.2       2003        PSF         yes
    2.3             2.2.2       2002-2003   PSF         yes
    2.3.1           2.3         2002-2003   PSF         yes
    2.3.2           2.3.1       2002-2003   PSF         yes

Footnotes:

(1) GPL-compatible doesn't mean that we're distributing Python under
    the GPL.  All Python licenses, unlike the GPL, let you distribute
    a modified version without making your changes open source.  The
    GPL-compatible licenses make it possible to combine Python with
    other software that is released under the GPL; the others don't.

(2) According to Richard Stallman, 1.6.1 is not GPL-compatible,
    because its license has a choice of law clause.  According to
    CNRI, however, Stallman's lawyer has told CNRI's lawyer that 1.6.1
    is "not incompatible" with the GPL.

Thanks to the many outside volunteers who have worked under Guido's
direction to make these releases possible.


B. TERMS AND CONDITIONS FOR ACCESSING OR OTHERWISE USING PYTHON
===============================================================

PSF LICENSE AGREEMENT FOR PYTHON 2.3
------------------------------------

1. This LICENSE AGREEMENT is between the Python Software Foundation
("PSF"), and the Individual or Organization ("Licensee") accessing and
otherwise using Python 2.3 software in source or binary form and its
associated documentation.

2. Subject to the terms and conditions of this License Agreement, PSF
hereby grants Licensee a nonexclusive, royalty-free, world-wide
license to reproduce, analyze, test, perform and/or display publicly,
prepare derivative works, distribute, and otherwise use Python 2.3
alone or in any derivative version, provided, however, that PSF's
License Agreement and PSF's notice of copyright, i.e., "Copyright (c)
2001, 2002, 2003 Python Software Foundation; All Rights Reserved" are
retained in Python 2.3 alone or in any derivative version prepared by
Licensee.
```

```
3. In the event Licensee prepares a derivative work that is based on
or incorporates Python 2.3 or any part thereof, and wants to make
the derivative work available to others as provided herein, then
Licensee hereby agrees to include in any such work a brief summary of
the changes made to Python 2.3.

4. PSF is making Python 2.3 available to Licensee on an "AS IS"
basis.  PSF MAKES NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR
IMPLIED.  BY WAY OF EXAMPLE, BUT NOT LIMITATION, PSF MAKES NO AND
DISCLAIMS ANY REPRESENTATION OR WARRANTY OF MERCHANTABILITY OR FITNESS
FOR ANY PARTICULAR PURPOSE OR THAT THE USE OF PYTHON 2.3 WILL NOT
INFRINGE ANY THIRD PARTY RIGHTS.

5. PSF SHALL NOT BE LIABLE TO LICENSEE OR ANY OTHER USERS OF PYTHON
2.3 FOR ANY INCIDENTAL, SPECIAL, OR CONSEQUENTIAL DAMAGES OR LOSS AS
A RESULT OF MODIFYING, DISTRIBUTING, OR OTHERWISE USING PYTHON 2.3,
OR ANY DERIVATIVE THEREOF, EVEN IF ADVISED OF THE POSSIBILITY THEREOF.

6. This License Agreement will automatically terminate upon a material
breach of its terms and conditions.

7. Nothing in this License Agreement shall be deemed to create any
relationship of agency, partnership, or joint venture between PSF and
Licensee.  This License Agreement does not grant permission to use PSF
trademarks or trade name in a trademark sense to endorse or promote
products or services of Licensee, or any third party.

8. By copying, installing or otherwise using Python 2.3, Licensee
agrees to be bound by the terms and conditions of this License
Agreement.
```