DHBW
Duale Hochschule
Baden-Württemberg

Verarbeitung von TerraSAR-X/TanDEM-X-Radaraufnahmen zu Zeitserien mittels Bildkoregistrierungs- und Geokodierungsalgorithmen

aus dem Studiengang Informationstechnik

an der Dualen Hochschule Baden-Württemberg in Mannheim

von

Nane Götte

26.08.2025

Matrikelnummer, Kurs:	6297164, TINF22IT1
Unternehmen:	Deutsches Zentrum für Luft- und Raumfahrt e.V., Oberpfaffenhofen
Betreuer im Unternehmen:	Philipp Posovszky
Betreuer an der DHBW:	Arne Sachtler

Selbstständigkeitserklärung

Gemäß Ziffer 1.1.13 der Anlage 1 zu §§ 3, 4 und 5 der Studien- und Prüfungsordnung für die Bachelorstudiengänge im Studienbereich Technik der Dualen Hochschule Baden-Württemberg vom 29.09.2017.

Ich versichere hiermit, dass ich meine Bachelorarbeit zum Thema:

Verarbeitung von TerraSAR-X/TanDEM-X-Radaraufnahmen zu Zeitserien mittels Bildkoregistrierungs- und Geokodierungsalgorithmen

selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Ich versichere zudem, dass alle eingereichten Fassungen übereinstimmen.

Oberpfaffenhofen 26.08.2025

Nane Götte

Abstract

As part of its public relations work, the Microwaves and Radar Institute at the German Aerospace Center (DLR) processes radar images from the TerraSAR-X/TanDEM-X mission into presentation material. Employees manually produce the videos and images, without relying on automation to select or to process the images. This thesis describes the development of a software application called StackSAR. The latter will support the production of presentation material by automating the analysis of the acquisition data and the image co-registration of the radar images, as well as their processing into video and image material. The software implementation is based on a requirements analysis carried out at the start of the project and the resultant requirements documentation. Test-driven development is the programming approach used to implement the requirements. Finally, a system test and a usability test were conducted to examine and to document the final developmental stage of the StackSAR software. The system test showed that the implementation fulfilled nearly all functional requirements. In the usability test, the stakeholder feedback yielded a score of 85/100 on the System Usability Scale. Based on these results, StackSAR is evaluated as a successful initial version of an application software to automate the creation of presentation material. It is expected that future development endeavors of StackSAR will benefit from the software's testability, so that successfull integration of the required geocoding functionality and improvement of usability are enabled.

Zusammenfassung

Im Rahmen seiner Öffentlichkeitsarbeit verarbeitet das Institut für Hochfrequenztechnik und Radarsysteme am Deutschen Zentrum für Luft- und Raumfahrt Radaraufnahmen der TerraSAR-X/TanDEM-X-Mission zu Präsentationsmaterial. Die zuständigen Mitarbeitenden produzieren die Videos und Bilder manuell, ohne sich bei der Auswahl der Aufnahmen oder ihrer Verarbeitung auf Automatisierung zu stützen. Die vorliegende Arbeit beschreibt die Entwicklung einer Softwareanwendung namens StackSAR. Diese wird die Erstellung des Präsentationsmaterials durch Automatisierung der Analyse des Aufnahmebestands, der Bildkoregistrierung der Radaraufnahmen sowie ihrer Verarbeitung zu Video- und Bildmaterial unterstützen. Die Implementierung der Anwendungssoftware stützt sich auf eine zu Projektbeginn durchgeführte Anforderungsanalyse und die daraus angefertigte Anforderungsdokumentation. Zur Umsetzung der Anforderungen wurde der Programmierungsansatz der testgetriebenen Entwicklung verfolgt. Abschließend wurde der finale Entwicklungsstand der StackSAR-Software in einem Systemtest sowie in einem Usability-Test untersucht und festgehalten. Der Systemtest ergab, dass nahezu alle funktionalen Anforderungen durch die Implementierung erfüllt werden konnten. Im Usability-Test ergab das Feedback durch den Stakeholder eine Punktzahl von 85/100 auf der System Usability Scale. StackSAR wird auf Basis der Ergebnisse als eine gelungene erste Ausführung einer Anwendungssoftware bewertet, die die Erstellung von Präsentationsmaterial durch Automatisierung künftig effizienter machen wird. Es wird erwartet, dass die zukünftige Weiterentwicklung von StackSAR von der Testbarkeit seiner Software profitiert, insbesondere bei der notwendigen Integration von Geokodierungsverfahren sowie der Verbesserung der Bedienbarkeit.

Inhaltsverzeichnis

1 Einleitung 1
1.1 Problemstellung 2
1.2 Ziele 2
1.3 Vorgehensweise 4
1.4 Sprachliche Hinweise 5

2 Grundlagen 6
2.1 Theoretische Grundlagen 6
2.1.1 Synthetic Aperture Radar 6
2.1.2 TerraSAR-X und TanDEM-X 10
2.1.3 Bildkoregistrierung 13
2.1.4 Geokodierung 17
2.2 Methodische Grundlagen 18
2.2.1 Anforderungsanalyse 18
2.2.2 Test Driven Development 22

3 Anforderungen 25
3.1 Anforderungsanalyse 25
3.2 Anforderungsdokumentation 25
3.2.1 Einleitung und Übersicht 26
3.2.2 Referenzen 31
3.2.3 Anforderungen 31

4 Algorithmenstudie und -auswahl 36
4.1 Literaturrecherche 36
4.2 Versuch 37
4.3 Auswertung 41

5 Implementierung 43
5.1 TDD und Versionsverwaltung 45
5.2 Nutzungsanleitung 47
5.3 Automatisierte Analyse des Aufnahmebestands 47
5.4 Automatisierung der Bildkoregistrierung 53
5.5 Automatisierung der Herstellung von Präsentationsmaterial 54

6 Tests 59
6.1 Systemtest 59
6.2 Usability-Test 65

7 Diskussion **69**

8 Ausblick **73**

Literatur ..... **a**

Abbildungsverzeichnis

Abbildung 1 Nachstellung der deutschen Fernerkundungssatelliten TerraSAR-X und TanDEM-X im Überflug der Erdoberfläche 1

Abbildung 2 Ausschnitt eines Bild-Outputs der Anwendungssoftware 3

Abbildung 3 Das elektromagnetische Spektrum mit den SAR-Frequenzbändern [1] 7

Abbildung 4 *Down-looking* Vorgehensweise zur Erhebung von Radardaten (links) und *side-looking* (rechts) [2] 8

Abbildung 5 Die geometrische Formation einer SAR-Aufnahme [1] 9

Abbildung 6 Darstellung der durchschnittlichen Änderungsgeschwindigkeiten des Drygalski-Gletschers, abgeleitet aus 29 TerraSAR-X-Aufnahmen. [3] 11

Abbildung 7 Die Aufnahmemodi ScanSAR, StripMap und Spotlight [4] 12

Abbildung 8 Zwei Bilder der gleichen Szene aus verschiedenen Perspektiven – vor der Bildkoregistrierung [5] 13

Abbildung 9 Zwei Bilder der gleichen Szene aus verschiedenen Perspektiven – nach der Feature Point Detection [5] 14

Abbildung 10 Zwei Bilder der gleichen Szene aus verschiedenen Perspektiven – vor (oben) und nach (unten) der Bildkoregistrierung [5] 16

Abbildung 11 Geokodierung als Übertragung einer Szene aus einem zweidimensionalen Bild in ein dreidimensionales Modell [6] 17

Abbildung 12 Der Kommunikationsprozess zwischen den an der Anforderungsanalyse beteiligten Parteien [7] 18

Abbildung 13 Formulierungsschablone für funktionale Anforderungen nach Pohl und Rupp [8], übersetzt ins Deutsche in Anlehnung an Pohl und Rupp [9] 20

Abbildung 14 Systementwurf des Stakeholders aus dem projektinternen Referenzdokument 28

Abbildung 15 Use-Case-Diagramm zur Darstellung der StackSAR-Systemfunktionalitäten 29

Abbildung 16 Die im Referenzdokument enthaltene Beispielgrafik als Vorschau dessen, wie die Parameterdaten der Datatake Requests in Form einer Heatmap dargestellt werden können. Die dargestellte Heatmap beinhaltet manuell hinzugefügte Markierungen von Supersites, welche mit StackSAR automatisch aus den Daten identifiziert und in der Heatmap markiert werden sollen. 34

Abbildung 17 Die im Referenzdokument enthaltene Beispielgrafik als Vorschau dessen, wie die farbliche Darstellung der in den SAR-Aufnahmen eines Stacks enthaltenen Veränderungen über die Zeit umgesetzt werden kann 35

Abbildung 18 Darstellung der gelungenen Zuordnungen von Merkmalsvektoren zweier Radaraufnahmen. Jede grüne Linie verbindet einen ORB-Merkmalsvektoren in der linken Radaraufnahme mit einem ORB-Merkmalsvektoren in der rechten Radaraufnahme. 41

Abbildung 19 Darstellung der gelungenen Zuordnungen von Merkmalsvektoren zweier Radaraufnahmen. Jede grüne Linie verbindet einen KAZE-Merkmalsvektoren in der linken Radaraufnahme mit einem KAZE-Merkmalsvektoren in der rechten Radaraufnahme. 42

Abbildung 20 Reduziertes Klassendiagramm vom Aufbau der StackSAR-Software 44

Abbildung 21 Flussdiagramm vom Entwicklungsprozess einer StackSAR-Funktion 46

Abbildung 22 Datenflussdiagramm von der Verarbeitung der Datatake Requests zur Heatmap durch StackSAR 49

Abbildung 23 Anzeige der aus den Metadaten der Datatake Requests generierten Heatmap 50

Abbildung 24 Heatmap-Ausschnitt 52

Abbildung 25 Ein schwarz-weißes Bild vor dem morphologischen Öffnen (links) und nach dem Öffnen (rechts) 56

Abbildung 26 Resultat der Bildgenerierung mit StackSAR zur Darstellung der Veränderungen in einem Stack von Aufnahmen in roter Farbe 57

Tabellenverzeichnis

Tabelle 1 Die sechs Qualitätsmerkmale nach ISO/IEC 9126, mit dazugehörigen Beschreibungen und Stichwörtern aus Grande [10] 19

Tabelle 2 Übersicht über verschiedene Arten des Testens eines Softwaresystems 23

Tabelle 3 Die Kontextaspekte des StackSAR-Systems, zugeordnet zu den Aspekttypen nach Pohl und Rupp [9] 27

Tabelle 4 Die an StackSAR gerichteten funktionalen Anforderungen 31

Tabelle 5 Die an die Qualitätsmerkmale (nach ISO/IEC 9126) von StackSAR gerichteten nicht-funktionalen Anforderungen 33

Tabelle 6 Durchschnittliche Anzahlen der gelungenen Zuordnungen von Kennzeichnern der Bilder zum Referenzbild pro Bildordner. 41

Tabelle 7 Liste der kritischen Aufnahmeparameter, deren Übereinstimmung einen geeigneten Stack charakterisiert. 47

Tabelle 8 Die Protokollierungstabelle für Anwendungsfall Nr. 1 des Systemtests 61

Tabelle 9 Die Protokollierungstabelle für Anwendungsfall Nr. 2 des Systemtests 62

Tabelle 10 Die Protokollierungstabelle für Anwendungsfall Nr. 3 des Systemtests 62

Tabelle 11 Die Protokollierungstabelle für Anwendungsfall Nr. 4 des Systemtests 64

Tabelle 12 Wiedergabe des im Usability-Test durch den Probanden ausgefüllten SUS-Fragebogens 67

Abkürzungsverzeichnis

AVI	Audio Video Interleave
CSV	Comma-Separated Values
DBSCAN	Density-Based Spatial Clustering of Applications with Noise
DEM	Digital Elevation Model
DLR	Deutsches Zentrum für Luft- und Raumfahrt e.V.
FLANN	Fast Library for Approximate Nearest Neighbors
GIF	Graphics Interchange Format
GIS	Geographic Information System
ID	Identifikationsnummer
IDL	Interactive Data Language
LTDB	Langzeitdatenbank
Radar	Radio Detection and Ranging
SAR	Synthetic Aperture Radar
TDD	Test Driven Development

1 Einleitung

Am Institut für Hochfrequenztechnik und Radarsysteme des Deutschen Zentrums für Luft- und Raumfahrt (DLR) in Oberpfaffenhofen wird die Mission TerraSAR-X/TanDEM-X betreut. Seit 18 und 15 Jahren erheben die deutschen Satelliten TerraSAR-X und TanDEM-X Aufnahmen der Erdoberfläche anhand von Radar mit synthetischer Apertur (SAR) [11], die für eine Vielzahl kommerzieller und wissenschaftlicher Anwendungen zum Einsatz kommen [12].

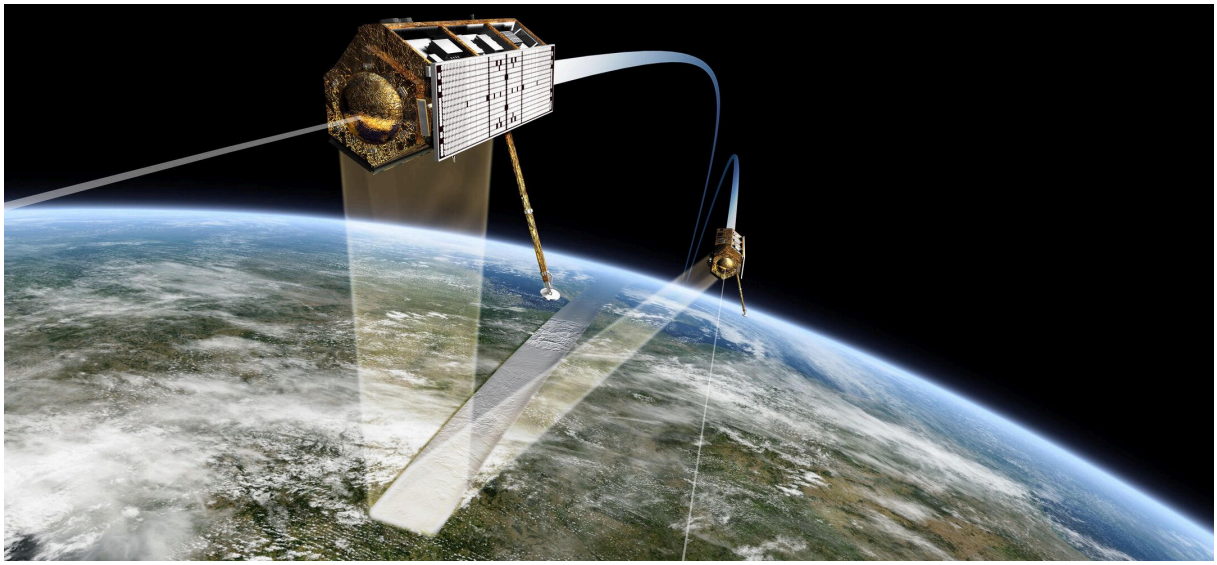


Abbildung 1 — Nachstellung der deutschen Fernerkundungssatelliten TerraSAR-X und TanDEM-X im Überflug der Erdoberfläche

Die durch die TerraSAR-X/TanDEM-X-Mission bereitgestellten SAR-Daten können kombiniert werden, sodass Höheninformationen über die Erdoberfläche daraus abgeleitet werden können [13]. Eine Darstellung der Aufnahme von Höhendaten mit TerraSAR-X und TanDEM-X ist durch Abbildung 1 gegeben. Die gewonnenen SAR-Daten werden der Allgemeinheit, der wissenschaftlichen Gemeinschaft sowie der Geschäftswelt präsentiert, indem aus ihnen Präsentationsmaterial für die Öffentlichkeitsarbeit des DLR erstellt werden. Dabei handelt es sich insbesondere um Videos, die Zeitserien von Radaraufnahmen eines Ortes auf der Erde zusammenhängend wiedergeben und so die aufgenommenen Veränderungen an diesem Ort sichtbar machen. Mitunter werden auch Bilder generiert, in denen die in den Aufnahmen enthaltenen Veränderungen farblich hervorgehoben sind.

1.1 Problemstellung

Die Arbeit, die die Erstellung des Präsentationsmaterials für die Öffentlichkeitsarbeit beansprucht, wird von einzelnen Mitarbeitenden in den DLR-Projekten übernommen. Da keine Anwendungssoftware zur digitalen Unterstützung dieser Arbeit existiert, verarbeiten die zuständigen Mitarbeitenden die SAR-Aufnahmen manuell zu Videos und Bildern, ohne Automatisierung. Diese Vorgehensweise hat vor allem Ineffizienz zum Nachteil und beginnt bereits mit der aufwändigen Suche nach Radaraufnahmen, die den selben Ort über einen ausreichend großen Zeitraum hinweg abbilden, ohne zeitlich zu eng oder zu entfernt zu liegen. Ergibt die Suche geeignete Radaraufnahmen, müssen Mitarbeitende diese im nächsten Schritt einzeln beschaffen. Nach der Beschaffung der Bilddaten werden diese händisch in ihrer zeitlichen Reihenfolge zu einer Serie in Videoform zusammengesetzt. Da die Radaraufnahmen auch unter den gleichen Aufnahmebedingungen nicht perfekt kongruent sind, müssen sie einander zunächst angeglichen –- koregistriert –- werden, sodass der darin dargestellte Ort für Betrachtende über das Video hinweg unbewegt erscheint. Der Arbeitsprozess stellt einen großen Aufwand für die zuständigen Mitarbeitenden dar, der durch Anwendungssoftware mit Automatisierungen erheblich reduziert werden könnte. Die bestehende Notwendigkeit der Automatisierung wiederkehrender SAR-Verarbeitungsaufgaben wird in Tsokas et al. [14] ebenso für das gesamte SAR-Arbeitsfeld verdeutlicht.

1.2 Ziele

Ziel der vorliegenden Arbeit ist die Entwicklung einer Anwendungssoftware, welche die Mitarbeitenden des Instituts für Hochfrequenztechnik und Radarsysteme durch Automatisierung bei der Erstellung von Präsentationsmaterial aus Radaraufnahmen unterstützt. Der Schwerpunkt der Arbeit ist die Analyse von Koregistrierungsverfahren, um ihre Eignung für die Angleichung der Radaraufnahmen von TerraSAR-X und TanDEM-X festzustellen. Von besonderem Interesse ist zudem die automatisierte Erzeugung einer Heatmap durch die Anwendungssoftware. Anhand der Heatmap soll vor allem die globale räumliche Verteilung aller in der TerraSAR-X/TanDEM-X-Mission getätigten Radaraufnahmen ersichtlich werden. Für die Entwicklung der Anwendungssoftware stehen zu Projektbeginn die folgenden Ziele und Teilziele fest.

1.2.1 Automatisierte Analyse des Aufnahmebestands

a) Es sind Eignungskriterien zu formulieren, die festlegen, wann eine Reihe von Radaraufnahmen (ein sogenannter Stack) geeignet ist für die Erstellung von Präsentationsmaterial. Wann bilden Aufnahmen eine passende Zeitreihe? Wie nah müssen die Koordinaten der Aufnahmen beieinander liegen? Zu allen gemachten Radaraufnahmen gibt es Aufzeichnungen (Metadaten), die algorithmisch zu analysieren sind.

b) In einer auf die Weltkarte projizierten Heatmap sollen die analysierten Metadaten dargestellt werden, sodass die globale Verteilung aller in der Datenbank aufgezeichneten Radaraufnahmen sichtbar wird. Darin soll insbesondere erkennbar werden, von welchen Orten sich über die Lebenszeit der Mission eine größere Zahl von Radaraufnahmen angesammelt hat.

1.2.2 Automatisierte Bildkoregistrierung

a) Ein Koregistrierungs- oder Geokodierungsverfahren, dass sich für SAR-Aufnahmen von Gletscherbewegung, Stadtentwicklung und Bergbau eignet, ist auszuwählen und zu implementieren.

b) Ein Koregistrierungs- oder Geokodierungsverfahren, dass sich für DEM-Aufnahmen von Gletschern, Vulkanen und Landwirtschaft eignet, ist auszuwählen und zu implementieren.

1.2.3 Automatisierte Herstellung von Präsentationsmaterial

a) Die automatisierte Verarbeitung von Radaraufnahmen zu Videos im Graphics-Interchange-Format (GIF) oder im Audio-Video-Interleave-Format AVI ist in die Anwendungssoftware zu integrieren.

b) Die automatisierte Verarbeitung von Radaraufnahmen zu Bildern wie in Abbildung 2, in denen die Unterschiede beziehungsweise Veränderungen zwischen den Aufnahmen farblich eingezeichnet ist, soll in die Anwendungssoftware integriert werden.

Abbildung 2 — Ausschnitt eines Bild-Outputs der Anwendungssoftware

1.3 Vorgehensweise

Im Vorgehen zur Erreichung der aufgestellten Implementierungsziele (siehe Kapitel 1.2) sind die nachfolgenden Methoden als verbindlich für die vorliegende Arbeit zu betrachten. Kapitel 2.2 dient hierzu als Wissensbasis. Das Durchführen einer Literaturreche, das Testen des Implementierungsergebnisses sowie die abschließende Diskussion des Implementierungserfolgs werden als grundsätzliche Bestandteile der wissenschaftlichen Arbeit vorausgesetzt.

1.3.1 Anforderungsmanagement

a) Anhand einer Anforderungsanalyse sollen die Vorstellungen derjenigen ermittelt werden, die die zu entwickelnde Anwendung zukünfitg nutzen werden.

b) In einer Anforderungsdokumentation sollen die Ergebnisse der Anforderungsanalyse vollständig dokumentiert und festgehalten werden.

1.3.2 Vergleichsstudie

a) Gegenwärtige Fachliteratur heranziehen, um geeignete Algorithmen und Verfahren zur Bildkoregistrierung von SAR-Aufnahmen identifizieren.

b) Die identifizierten Methoden anhand von Stichproben aus dem ca. 1.000.000 Radaraufnahmen umfassenden Bestand erproben und eine begründete Auswahl für die Implementierung treffen.

1.3.3 Testgetriebene Entwicklung & Versionsverwaltung

a) Bei der Umsetzung der geforderten Software soll der Ansatz der testgetriebenen Entwicklung verfolgt werden, sodass insbesondere zu den Hauptfunktionalitäten bestandene Tests vorliegen.

b) Die Software soll während der Entwicklung mit einem Gitlab-Repository synchronisiert werden, sodass Versionsverwaltung betrieben werden kann und der Entwicklungsstand für andere Mitwirkende im Projekt sichtbar ist.

1.4 Sprachliche Hinweise

1.4.1 Erklärung zur genderkonformen Schreibweise

Die vorliegende Bachelorarbeit nutzt zur genderkonformen Schreibweise den Leitfaden der Dualen Hochschule Baden-Württemberg (DHBW) über gendersensible Sprache [15]. Dem Leitfaden entsprechend bedient sich die Arbeit geschlechterneutraler Formulierungen und Schreibweisen, um alle Geschlechtsidentitäten sprachlich einzubeziehen, ohne dabei die Lesbarkeit einzuschränken.

1.4.2 Erklärung zu englischen Fachbegriffen

Um die originale Bedeutung englischsprachiger Fachbegriffe beizubehalten und ungenaue oder unpassende Übersetzungen ins Deutsche zu vermeiden, sieht die vorliegende Arbeit von einer zwangsläufigen Übersetzung der fachbezogenen Fremdwörter ab. Insbesondere sind das Zielsystem beschreibende Diagramme in Englisch verfasst worden, um sprachlich nicht vom Quellcode abzuweichen.

2 Grundlagen

In diesem Kapitel ist das Grundlagenwissen dargelegt, welches zum tieferen Verständnis der vorliegenden Arbeit nötig ist.

2.1 Theoretische Grundlagen

Das erste Grundlagenkapitel widmet sich der Erklärung der theoretischen Grundlagen. Zunächst erfolgen Ausführungen zur TerraSAR-X/TanDEM-X-Mission und den durch sie bereitgestellten Radaraufnahmen. Im Anschluss werden die Funktionsweisen erklärt, die der Bildkoregistrierung und der Geokodierung zugrundeliegen.

2.1.1 Synthetic Aperture Radar

Synthetic Aperture Radar (SAR) ist eine radarbasierte Fernerkundungstechnologie [1], die die Produktion hochauflösender Radaraufnahmen ermöglicht [16]. Radio Detection and Ranging Radar bezeichnet das technologische Konzept der funkgestützten Erkennung und Lokalisierung von Objekten, das seit den 1940er Jahren in der Erdbeobachtung eingesetzt wird [17]. In der Fernerkundung werden zwei Arten von Erdbeobachtungsmethoden unterschieden - aktive Methoden und passive Methoden [2]. SAR zählt zu den aktiven Beobachtungsmethoden [1], [2], bei denen das Instrument zunächst einen Impuls aussendet, der mit der Erdoberfläche interagiert, und den zurückgestreuten Energieanteil dann wieder aufnimmt [2]. Im Gegensatz dazu wird bei passiver Erdbeobachtung durch passive Sensoren Energie gemessen, die von der Erdoberfläche oder Atmosphäre ausgestrahlt wird oder von einer anderen Quelle auf die Erdoberfläche trifft [2]. Dieses Vorgehen wird als passiv bezeichnet, da Energie aufgenommen wird, ohne dass dem beobachteten System zuvor gezielt Energie hinzugefügt wurde [2].

In der Fernerkundung eingesetzte Radarinstrumente sind so konstruiert, dass sie in bestimmten Frequenzbereichen des elektromagnetischen Spektrums arbeiten, abhängig von ihrer designierten Aufgabe. Auf dem elektromagnetischen Spektrum ist die für die SAR-Technologie genutzte Strahlung im langwelligen Bereich, ihre Wellen sind länger als die des sichtbaren Lichts beziehungsweise ihre Frequenzen niedriger. Folglich besteht ein energetischer Unterschied zwischen sichtbarem Licht und Mikrowellen, der bedingt, dass die Eigenschaften der Erdoberfläche in Radaraufnahmen anders erscheinen als in optischen Bildern. [2]

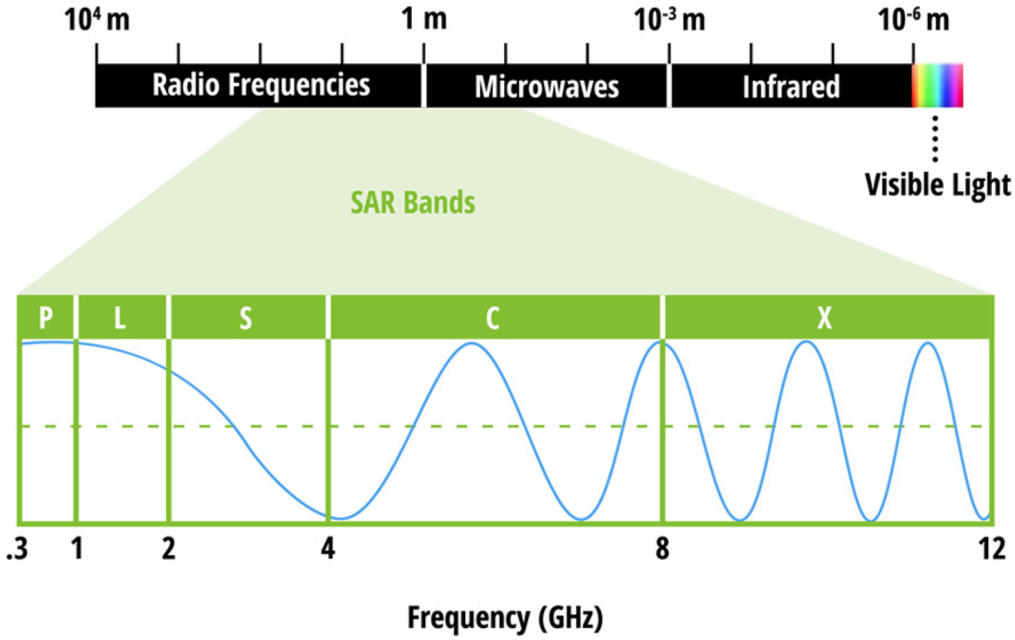


Abbildung 3 — Das elektromagnetische Spektrum mit den SAR-Frequenzbändern [1]

In Abbildung 3 ist der Bereich des elektromagnetischen Spektrums markiert, in dem die Wellenlängen der in der SAR-Technologie genutzten elektromagnetischen Strahlung liegen – im Bereich weniger Zentimeter bis einiger Dezimeter [16]. Dieser wird typischerweise weiter unterteilt in sogenannte Frequenzbänder, als „SAR Bands“ bezeichnet in Abbildung 3. Die Frequenzbänder stellen eine Kategorisierung der elektromagnetischen Strahlung dar, die die konkrete Zuschreibung von Anwendungszwecken nach Wellenlängen erlaubt: Zum Beispiel dringt beim Einsatz des X-Band-Radars für Erdbeobachtungsaufnahmen das Signal kaum in die Vegetation auf der Erdoberfläche ein [1]. Wird hingegen beispielsweise L-Band- oder P-Band-Radar eingesetzt, dringt das Signal tiefer in die Vegetation ein und interagiert mit dieser, sodass der zurückgeworfene Impuls Informationen über den Vegetationsbestand liefert [1]. Auch bei Böden oder Eisflächen bestimmt die Wellenlänge der Radarstrahlung, mit der die Erdoberfläche abgetastet wird, wie weit diese eindringen kann, bevor sie zum Radarinstrument zurückgeworfen wird [1], [2].

Der entscheidende Vorteil von SAR-Technologie gegenüber optischer Bildgebung ist, dass die Radaraufnahmen bei nahezu allen Witterungsbedingungen möglich sind und kein Tageslicht brauchen [2]. Dadurch ermöglicht der Einsatz von SAR auch das regelmäßige Abbilden von

Regionen, die üblicherweise stark von Wolken bedeckt sind, wie die tropischen Regenwälder, oder von Regionen, die wenig Tageslicht erleben, wie die Polkappen [17].

Anders als bei optischen Bildgebungsverfahren zeigt der Radarsensor eines Fernerkundungssystems nicht senkrecht auf die Erdoberfläche, sondern ist um einen bestimmten Winkel geneigt [17]. Dieser Unterschied ist in Abbildung 4 dargestellt. Radaraufnahmen werden üblicherweise in dem *side-looking* genannten Modus gemacht, da ansonsten nicht zwischen gleichzeitig abgetasteten Punkten auf der Erdoberfläche unterschieden werden kann [2]. Das liegt daran, dass die von ihnen rückgestreuten Signale im *down-looking* Modus zeitgleich auf die Radarantenne eintreffen würden [2] und dann in den aufgenommenen Daten keine Unterscheidung der gleichzeitig erfassten Punkte mehr möglich wäre. Dies ist links in Abbildung 4 anhand der Punkte beziehungsweise der Objekte *a* und *b* dargestellt. Im Gegensatz dazu können im rechts in Abbildung 4 gezeigten *side-looking* Modus die Punkte oder Objekte *a* und *b* nach der Aufnahme noch unterschieden werden, da die von ihnen zurückgestreuten Signale nicht gleichzeitig wieder auf das Radarinstrument eintreffen.

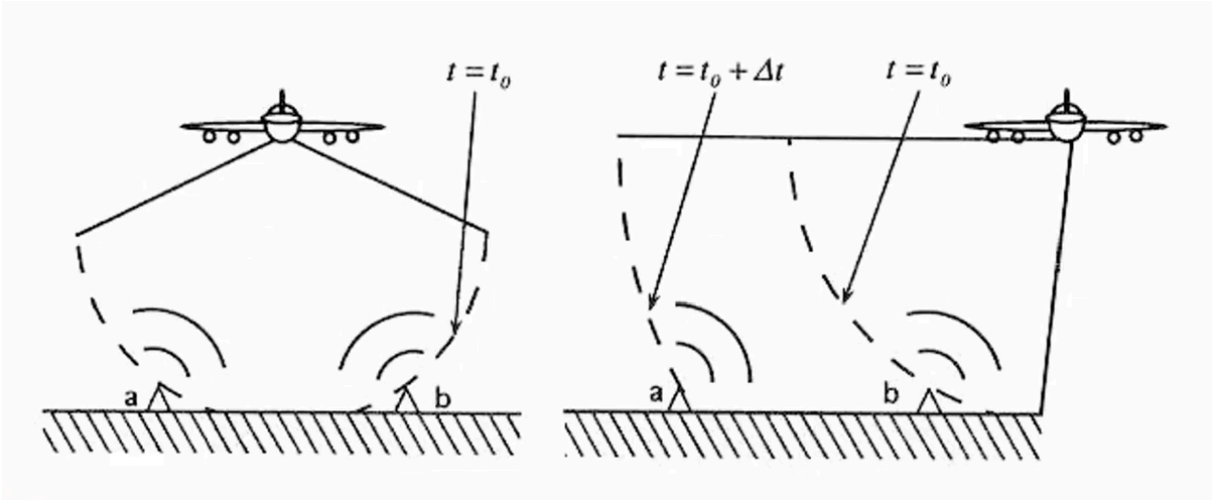


Abbildung 4 — *Down-looking* Vorgehensweise zur Erhebung von Radardaten (links) und *side-looking* (rechts) [2]

Bei der Datenerhebung mittels Radar wird der vom Instrument geformte Radarstrahl auf dem Weg zur Erdoberfläche um ein Vielfaches breiter [16]. Dargestellt ist dies in Abbildung 5. Darin ist mit grünen Rechtecken eine einzelne Radarantenne dargestellt, die sich entlang einer Bahn x bewegt. Sie fertigt die Aufnahmen *x1* bis *x2* an, wobei der Winkel β die Breite des auf der Erdoberfläche eintreffenden Radarstrahls vorgibt [17]. Je größer β ist, desto größer ist die

Fläche, die zu jedem Aufnahmezeitpunkt beleuchtet wird, und dessen rückgestreutes Signal dann wieder gebündelt auf die Radarantenne eintrifft, und desto geringer ist die Auflösung der Aufnahme [16].

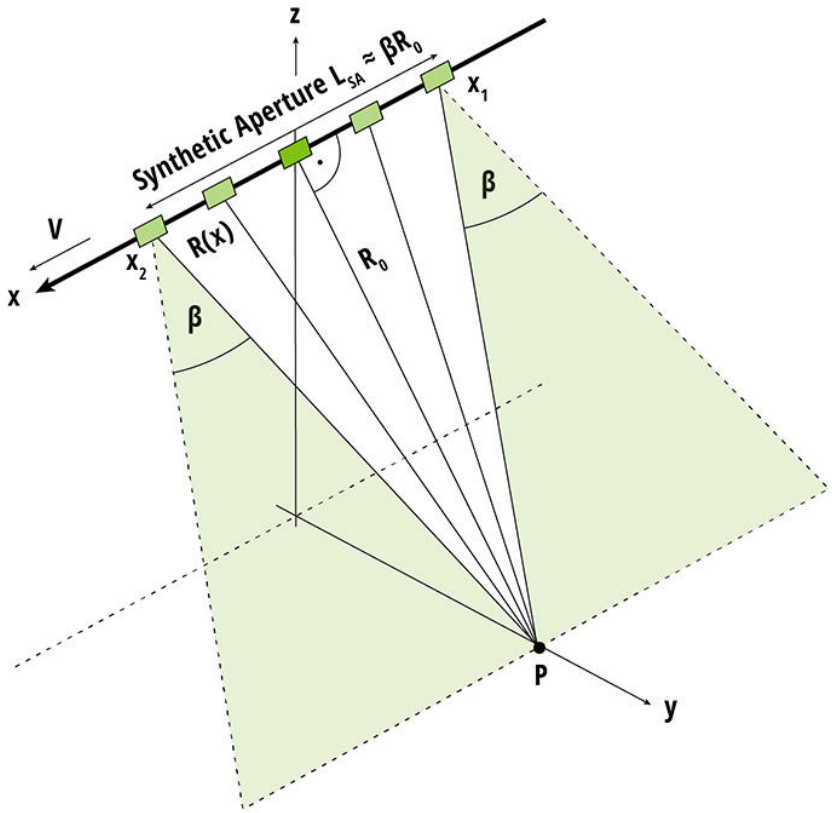


Abbildung 5 — Die geometrische Formation einer SAR-Aufnahme [1]

SAR löst das Problem der geringen Auflösung, indem das SAR-Instrument sich auf einem Pfad fortbewegt (*x* in Abbildung 5) und dabei kontinuierlich Radarimpulse aussendet (*x1* bis *x2* in Abbildung 5) und die entsprechenden Rückantworten aufnimmt [16]. Dadurch wird jeder Punkt im Aufnahmebereich mehrmals durch die Radarstrahlen abgetastet, wobei die Distanz zum Punkt bei jeder Aufnahme leicht verschieden ist [16]. Diese Distanzveränderungen können in den Daten schließlich aus der Phase der aufgenommenen Radarimpulse ausgelesen und interpretiert werden [16]. Dafür werden bei der Datenverarbeitung die einzelnen Aufnahmen zu einem Datensatz zusammengefasst und so kombiniert, dass sie dem Resultat einer Radaraufnahme mit größerer Antenne entsprechen [17]. *Synthetisch* ist SAR folglich deshalb, weil mit einer kleineren Antenne eine größere simuliert wird, indem eine Vielzahl von Radaraufnahmen so kombiniert wird, dass sich eine höhere Auflösung ergibt [1].

Aufgrund der beschriebenen Besonderheiten des SAR-Konzepts gibt es eine Vielzahl von industriellen, wissenschaftlichen und militärischen Anwendungen der Technologie. Die Anwendungsgebiete beinhalten unter anderem die Klassifizierung von Landflächen oder Feldfrüchten, die Detektion von Überschwemmungen oder Ölverschmutzungen oder die Kontrolle der Bodenfeuchtigkeit [2] sowie die Überwachung von Flughäfen, satellitengestützte Höhenmessungen, die Kontrolle von Wasseraufnahme, Städteentwicklung, Eis- oder Schneebeständen, oder die Kartierungen der Landnutzung, der Erdoberflächenbeschaffenheit oder der Vegetationsbeschaffenheit [14].

Die Abdeckung dieser Vielzahl von Anwendungsgebieten ist deshalb möglich, weil es beim Einsatz von SAR viele technologische Parameter gibt, mit der die Konfiguration einer Radaraufnahme an ihr Anwendungsziel angepasst werden an. Konkret kann unter anderem spezifiziert werden: das Frequenzband der eingesetzten Radarstrahlung, der *Look Angle* – das ist der Neigungswinkel des SAR-Instruments zur Erdoberfläche –, oder auch die Polarisation der elektromagnetischen Wellen – diese ist meist horizontal oder vertikal und bestimmt, wie die Welle mit den Zielobjekten oder -flächen interagiert [14].

Aufgrund der vielen Modi, in denen SAR-Sensoren operieren können, sind auch SAR-Daten entsprechend vielfältig und stehen in einer Vielzahl verschiedener Datenformate und Verarbeitungsstadien zur Verfügung [14]. Folglich ergibt sich für die Arbeit mit SAR-Daten die Limitierung, dass umgangreiche Nachverarbeitungschritte nötig sind, die insbesondere Korrekturen und Filterungen unerwünschter Effekte umfassen [1]. Beispielhaft dafür ist der *Look Angle*, der sichtbaren Einfluss auf SAR-Aufnahmen nimmt [2]. Wegen der Anwinkelung im *side-looking* Modus führt das Abtasten von unebenen Oberflächen mittels SAR zu geometrischen Verzerrungen in den Daten. Insbesondere erscheinen Erhebungen – wie zum Beispiel Berge – aufgrund des schiefen Blickwinkels, als wären sie in Richtung des Aufnahmesensors geneigt (*Foreshortening*) oder verdecken in der Aufnahme sogar die Bereiche auf der anderen, abgewandten Seite des Berges beziehungsweise der Erhebung (*Layover*) [17].

2.1.2 TerraSAR-X und TanDEM-X

Der deutsche Erdbeobachtungssatellit TerraSAR-X erhebt seit 2007 mit einem Synthetic-Aperture-Radar-Sensoren hochwertige Fernerkundungsdaten der Erdoberfläche [11], [18]. TerraSAR-X fliegt auf einer polaren Umlaufbahn in 514 Kilometern Höhe und hat für die Neigung der Radarantenne einen Schwenkbereich zwischen 20 und 60 Grad [11]. Die

Umlaufbahn heißt polar, da TerraSAR-X wiederkehrend über die Polargebiete fliegt, wobei sich die Erde unterhalb der Bahn ständig weiterdreht, sodass eine streifenweise Aufnahme der Erdoberfläche entsteht [19]. Auf diese Weise stellt die TerraSAR-X Mission hochwertige Synthetic-Aperture-Radar-Daten im X-Band mit einer Auflösung von bis zu einem Meter für die Forschung und Entwicklung sowie für wissenschaftliche und kommerzielle Anwendungen bereit [11].

Seit 2010 fliegt in enger Formation mit TerraSAR-X der baugleiche Satellit TanDEM-X auf gleicher Höhe, sodass zwischen ihnen stets nur wenige hundert Meter Abstand sind [12]. Das Ziel der TanDEM-X Mission – die Erstellung eines DEMs – konnte bereits 2016 erfüllt werden, da die beiden Satelliten von 2011 bis 2015 die gesamte Landoberfläche der Erde mehrfach vollständig ausgemessen haben [12].

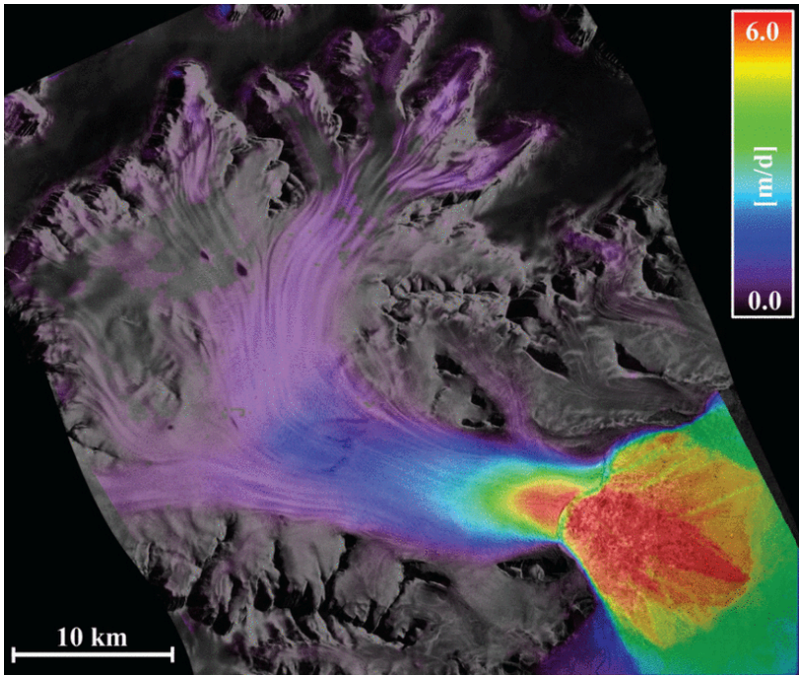


Abbildung 6 — Darstellung der durchschnittlichen Änderungsgeschwindigkeiten des Drygalski-Gletschers, abgeleitet aus 29 TerraSAR-X-Aufnahmen. [3]

Die von TerraSAR-X beziehungsweise TanDEM-X gemachten SAR-Aufnahmen können für eine Vielzahl verschiedener Anwendungen eingesetzt werden, wie zum Beispiel für die Beobachtung von Veränderungen in der Eisdynamik von Gletschern [18]. Abbildung 6 zeigt beispielsweise den Drygalski-Gletscher, aufgenommen mit dem TerraSAR-X-Satelliten. Die Aufnahmen des Gletschers sind kombiniert worden, sodass die Änderungsgeschwindigkeiten

in den Bildpunkten berechnet und farblich markiert werden konnten. Die Aufnahmen von TerraSAR-X eignen sich aus mehrerehn Gründen besonders für Anwendungen wie die zuvor beschriebene: Die Umlaufbahn von TerraSAR-X kann sehr genau vorausgesagt werden, die Auflösung der Aufnahmen wird als hoch gewertet und die kurzen Zeitabstände zwischen wiederkehrenden Aufnahmen derselben Position auf der Erdoberfläche ermöglichen präzise Berechnungen der Bewegung von Objekten [3].

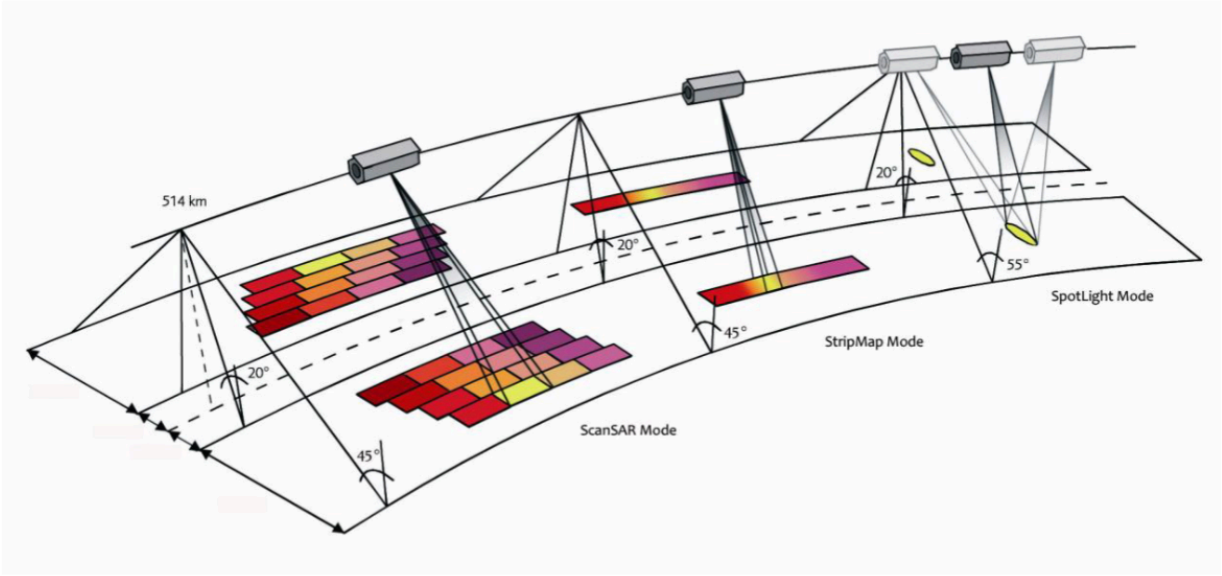


Abbildung 7 — Die Aufnahmemodi ScanSAR, StripMap und Spotlight [4]

Die Vielfalt der Anwendungsmöglichkeiten der TerraSAR-X/TanDEM-X-Radaraufnahmen wird zusätzlich dadurch bedingt dass sie flexibel in verschiedenen Modi (*Imaging Modes*) betrieben werden und daher verschiedenste Anforderungen an Radaraufnahmen erfüllen können [4]. Das X-Band-Radarinstrument von TerraSAR-X und TanDEM-X kann in verschiedenen Bildgebungsmodi eingesetzt werden, von denen in Abbildung 7 drei Ausführungen dargestellt sind. Im ScanSAR-Modus werden leicht überlappende Bereiche der Erdoberfläche aus verschiedenen Winkeln abgetastet, sodass aus den resultierenden Einzelaufnahmen die Szene zusammengesetzt werden kann [4]. Im StripMap-Modus – dem Standardmodus von TerraSAR-X und TanDEM-X – wird ein Streifen auf der Erdoberfläche abgetastet, der der Flugbahn des Satelliten folgt, sodass sich eine Radaraufnahme mit gleichbleibender Auflösung ergibt [4]. Im SpotLight-Modus wird ein begrenzter Bereich auf der Erdoberfläche durch kontinuierliche Veränderung des Aufnahmewinkels entlang der Flugrichtung im Überflug mit hoher Auflösung abgetastet [4].

2.1.3 Bildkoregistrierung

Der Prozess, zwei oder mehr Bilder desselben Objekts oder derselben Szene aneinander anzugleichen, heißt Koregistrierung. Im Englischen werden als äquivalente Begriffe dazu unter anderem „Image Registration“ und „Image Matching“ verwendet. Koregistrierung wird häufig dann angewandt, wenn Bilder zu verschiedenen Zeitpunkten, aus verschiedenen Perspektiven und/oder durch verschiedene Sensoren aufgenommen worden sind und zur Erhöhung des Informationsgehalts kombiniert werden sollen. In der Fernerkundung wird Koregistrierung beispielsweise dazu eingesetzt, ein Gesamtbild der Erde aus Radaraufnahmen zusammenzusetzen, die aus verschiedenen Perspektiven gemacht worden sind. [20]

Am Beispiel der in Abbildung 8 gezeigten Bilder desselben Bergs aus verschiedenen Perspektiven soll in den nachfolgenden Schritten der Ablauf einer Bildkoregistrierung erklärt werden.

Abbildung 8 — Zwei Bilder der gleichen Szene aus verschiedenen Perspektiven – vor der Bildkoregistrierung [5]

Der Prozess der Koregistrierung kann nach Saleem et al. [21] in die folgenden drei fundamentalen Prozessschritte unterteilt werden:

i. Merkmalspunkte detektieren (Feature Point Detection)

ii. Merkmalspunkte kennzeichnen (Feature Point Description)

iii. Merkmalsvektoren zuordnen (Feature Vector Matching)

Feature Point Detection

Die Detektion von Merkmalspunkten (Feature Points) ist grundlegender Bestandteil von Anwendungen der Computer Vision in der Fernerkundung, in der computergestützten Luftfahrt und in der astronomischen Beobachtung [22]. Die in den Bildern enthaltenen Merkmale oder Features, die üblicherweise anhand von Algorithmen detektiert werden, sind unter anderem Stufenkanten, Linien, Ecken, Knotenpunkte oder Dachkanten [23]. Der Grund für die Nutzung von Merkmalen wie den oben genannten ist, dass diese einfach zu lokalisieren sind, verglichen mit anderen Merkmalen [22]. Seit 1980 wurde eine Vielzahl von Koregistrierungsalgorithmen entwickelt [21], von denen einige ausschließlich der Detektion von Merkmalspunkten dienen, einige nur der Kennzeichnung von Merkmalspunkten dienen und wiederum einige beide Prozesschritte beinhalten [22]. Für die Arbeit mit Detektionsalgorithmen ist insbesondere zu beachten, dass sie für unterschiedliche Arten von Bildern und Merkmalspunkten unterschiedlich effektiv sind [22]. Für die Beispielbilder des Bergs könnte die Verteilung der detektierten Merkmalspunkte wie in Abbildung 9 aussehen.

Feature Point Description

Die Kennzeichnung (Description) dient dazu, die zuvor detektierten Merkmalspunkte zu charakterisieren und von ihrer unmittelbaren Umgebung abzugrenzen [22]. Dieser Prozessschritt ist notwendig, da die bloße Lokalisierung von Merkmalspunkten in einem Bild noch nicht ausreicht, um Verbindungen zu den Merkmalspunkten der anderen, zu koregistrierenden Bilder

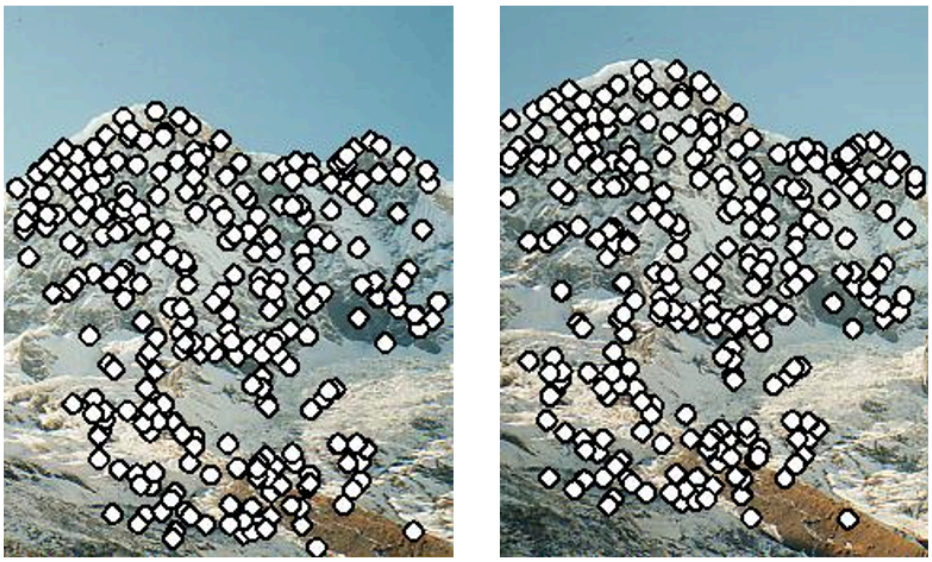


Abbildung 9 — Zwei Bilder der gleichen Szene aus verschiedenen Perspektiven – nach der Feature Point Detection [5]

herzustellen [21]. Es wurde daher eine Vielzahl von Methoden entwickelt, mittels derer ein Merkmalspunkt beschrieben werden kann, indem seiner definierten Merkmalsregion (Neighborhood) ein Merkmalsvektor (Feature Vector) zugeordnet wird, der den Merkmalspunkt repräsentiert [21]. Die simpelste Methode zur Kennzeichnung eines Merkmalspunktes ist die, die festgelegte Merkmalsregion durch einen Vektor ihrer Pixelwerte zu repräsentieren [22]. Nach der Feature Point Description wäre jedem der in Abbildung 9 eingezeichneten Merkmalspunkte ein Merkmalsvektor zu geordnet, der ihn auf eine bestimmte Weise beschreibt, sodass er von den anderen Merkmalspunkten (und deren Merkmalsvektoren) unterscheidbar wird.

Feature Vector Matching
Für das Zuordnen der Merkmalsvektoren eines Bildes zu denen eines anderen Bildes gibt es drei verschiedene grundlegende Herangehensweisen. Erstens können die Vektoren basierend auf einem Schwellenwert (Threshold) einander zugeordnet werden. Bei diesem Ansatz wird ein Merkmalsvektor des einen Bildes einem Merkmalsvektoren des anderen Bildes genau dann zugeordnet, wenn die Größe des Unterschieds (auch Distanz genannt) einen bestimmten Schwellenwert nicht überschreitet. Zweitens können Merkmalsvektoren einander nach dem Prinzip Nearest Neighbor zugeordnet werden. Hierbei gilt auch das Schwellenwertkriterium aus dem ersten Ansatz, aber zusätzlich muss gelten, dass ein Merkmalsvektor (nach einer festgelegten Metrik) dem zugeordneten Merkmalsvektor am nächsten ist, verglichen mit allen anderen. Drittens können zwei Merkmalsvektoren durch ein berechnetes Distanzverhältnis (Distance Ratio) als Nearest Neighbors identifiziert und einander zugeordnet werden, wobei das Distanzverhältnis unter einen festgelegten Schwellenwert fallen muss. Da es bei der Anwendung einfacher Zuordnungsverfahren häufig dazu kommt, dass Merkmalsvektoren fehlzugeordnet werden, können im Anschluss an das Feature Vector Matching Algorithmen zur Eliminierung der Fehlzuordnungen genutzt werden. [22]

In Abbildung 10 ist gezeigt, wie die originalen Bilder aus Abbildung 10 koregistriert und überlagert aussehen. Dafür sind im letzten Schritt die in Abbildung 9 gezeigten Merkmalspunkte der jeweiligen Bilder möglichst passend zugeordnet worden, sodass im Idealfall zu jedem Merkmalspunkt des linken Bildes ein Merkmalspunkt des rechten Bildes zugeordnet wurde. Für das Identifizieren, welche Merkmalspunkte einander am besten entsprechen, wurden wiederum die Merkmalsvektoren der Punkte herangezogen und verglichen. Schließlich wurde eine Unterschiedsmatrix berechnet und mit ihr das rechte Bild transformiert.

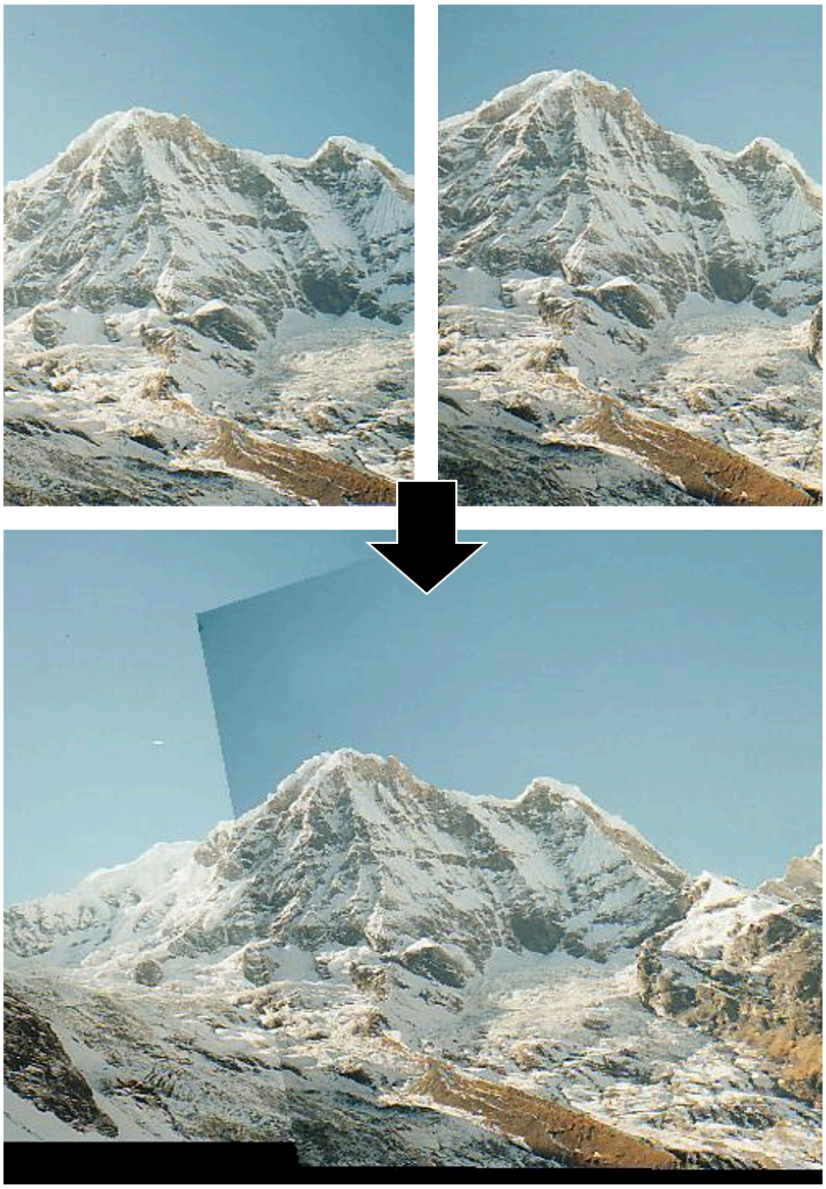


Abbildung 10 — Zwei Bilder der gleichen Szene aus verschiedenen Perspektiven – vor (oben) und nach (unten) der Bildkoregistrierung [5]

Zusammenfassend gibt es bei der Koregistrierung von Bildern keine einheitliche Methode oder Vorgehensweise, auf die standardmäßig zurückgegriffen werden kann. Deshalb muss für jeden Anwendungsfall stets anhand der Prioritäten (beispielsweise Rechenzeit, Bildmerkmale, Speicherverfügbarkeit, Genauigkeit) neu abgewogen werden, welche Algorithmen sich für die Durchführung der Featrue Point Detection, der Feature Point Description und des Feature Vector Matching am ehesten eignen und auszuwählen sind. [20]

2.1.4 Geokodierung

Wenn die Informationsgehälter mehrerer SAR-Aufnahmen kombiniert werden sollen, erfordert dies genaue Maßnahmen zur Registrierung. Das gilt insbesondere dann, wenn die Aufnahmen einen größeren Zeitraum überspannen oder sich aufgrund der Aufnahmekonfiguration, zum Beispiel der genutzten Frequenzbänder, unterscheiden. [24]

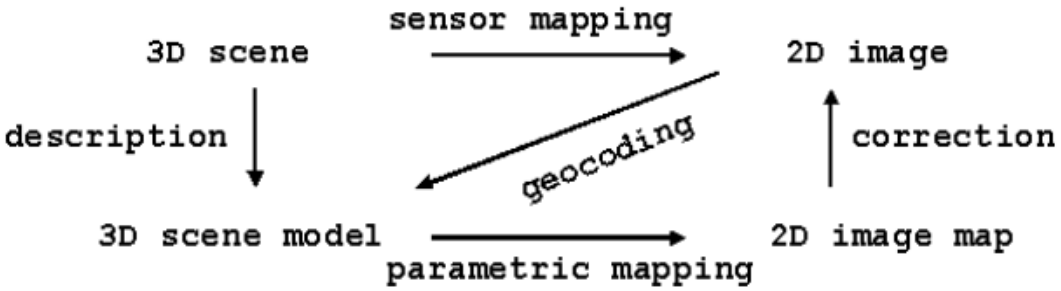


Abbildung 11 — Geokodierung als Übertragung einer Szene aus einem zweidimensionalen Bild in ein dreidimensionales Modell [6]

Eine solche Registrierungsmaßnahme ist die Karte-zu-Bild-Registrierung, meist Geokodierung genannt, die dazu dient, die Unterschiede zwischen Aufnahmen zu korrigieren [6]. Dabei wird dem in einer zweidimensionalen Aufnahme dargestellten Ort (die sogenannte Szene) ein Koordinatenpaar zugeordnet, indem mit Referenzdaten dessen reale Position im dreidimensionalen Modell der Erdoberfläche abgeleitet wird [25]. Dafür müssen Adresseinträge übersetzt werden und in den Referenzdaten auf der Karte muss nach der wahrscheinlichsten Position gesucht werden [25]. Auf diese Weise wird die in einem zweidimensionalen Bild dargestellte Szene in ein dreidimensionales Modell übertragen (siehe Abbildung 11). Geokodierung ermöglicht die Kombination mehrerer SAR-Aufnahmen letztendlich, indem diese auf ein gemeinsames dreidimensionales Referenzmodell zurückgeführt werden und entsprechend weiterverarbeitet werden können.

Es gibt verschiedene Arten der Geodierung, die ihre eigenen Methoden und Limitierungen mitbringen [25]. Für die SAR-Verarbeitung ist die Geokodierung von erheblicher Bedeutung: Nahezu alle Lieferanten von SAR-Bilddaten nutzen Standardprozeduren der Geokodierung [6].

2.2 Methodische Grundlagen

Im zweiten Grundlagenkapitel werden die Methoden erläutert, welche die Vorgehensweise, die für die vorliegende Arbeit festgelegt worden ist (siehe Kapitel 1.3), zusammensetzen.

2.2.1 Anforderungsanalyse

Eine Anforderungsanalyse ist ein systematisches Vorgehen zur Bestimmung der Anforderungen an ein zu entwickelndes System, bei der die Bedürfnisse derjenigen untersucht werden, die in die Systementwicklung involviert sind und/oder die Zielgruppe des Systems sind [26]. Letztere sind die sogenannten Stakeholder des jeweiligen Entwicklungsprojekts. Stakeholder sind konkret diejenigen Personen, Personengruppen oder Organisationen, die an dem zu entwickelnden System ein Interesse haben und/oder dessen Entwicklung mitbeeinflussen [27]. Die Untersuchung der Stakeholderbedürfnisse stellt, wie in Abbildung 12 abgebildet, einen Kommunikationsprozess dar zwischen den Stakeholdern auf der einen Seite und denjenigen, die die Anforderungen an das System verwalten, auf der anderen Seite [7].

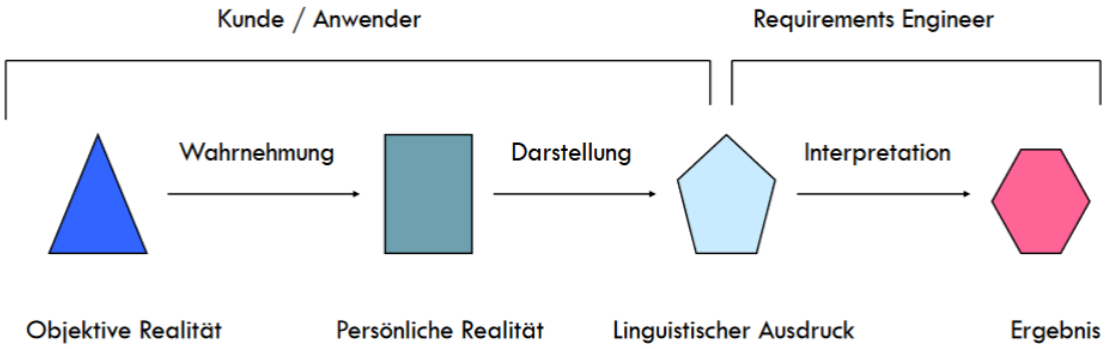


Abbildung 12 — Der Kommunikationsprozess zwischen den an der Anforderungsanalyse beteiligten Parteien [7]

Bei ebendiesem Kommunikationsprozess handelt es sich um die Anforderungserhebung [7], die zum Ziel hat, die Vision des zu entwickelnden Systems zu spezifizieren und das Ergebnis (siehe Abbildung 12) in Form von Anforderungen festzuhalten [27]. Bei der Anforderungserhebung werden dafür mithilfe von Befragungstechniken präzise und möglichst unverzerrte Aussagen von der Stakeholderperson erlangt [8] und interpretiert [7].

Das Ergebnis der Anforderungerhebung sind die Anforderungen an die Hardware und die Software des zu entwickelnden Systems. Anforderungen sind dem IEEE-Standard 610.12-1990 zufolge Bedingungen oder Fähigkeiten, die ein System erfüllen beziehungsweise haben muss, um seinen Anwendenden das Lösen eines bestimmten Problems oder das Erreichen eines bestimmten Ziels zu ermöglichen [26]. Dabei kann grundsätzlich zwischen zwei Arten von Anforderungen unterschieden werden: funktionale Anforderungen und nicht-funktionale Anforderungen. Funktionale Anforderungen beschreiben die Funktionalitäten und das Verhalten eines Produkts [10] und spezifizieren konkrete Funktionen, die ein System erfüllen muss [26]. Nicht-funktionale Anforderungen sind hingegen alle Anforderungen, die keine funktionalen Anforderungen sind [26] und werden auch als Qualitätsanforderungen bezeichnet [27]. Nach ISO/IEC 9126 werden bei der Anforderungsverwaltung die nachfolgenden sechs Qualitätsmerkmale unterschieden, welche in Grande [10] so beschrieben werden:

Qualitätsmerkmal	**Beschreibung**	**Stichwörter**
Änderbarkeit	Wie viel Aufwand zur Umsetzung von Änderungen an der Software sind nötig?	Modifizierbarkeit, Testbarkeit
Benutzbarkeit	Wie gut kann ein Benutzer mit der Software umgehen? Wie schnell kann er den Umgang erlernen?	Verständlichkeit, Bedienbarkeit
Effizienz	Wie effizient ist die Software in Bezug auf Betriebsmittel und Leistung?	Zeitverhalten, Ressourcenverbrauch
Funktionalität	Erfüllt die Software spezifizierte Funktionen?	Richtigkeit, Sicherheit
Übertragbarkeit	Mit welchem Aufwand kann die Software in eine andere Umgebung übertragen werden?	Anpassbarkeit, Installierbarkeit
Zuverlässigkeit	Erfüllt die Software ein bestimmtes Leistungsniveau?	Fehlertoleranz, Wiederherstellbarkeit

Tabelle 1 — Die sechs Qualitätsmerkmale nach ISO/IEC 9126, mit dazugehörigen Beschreibungen und Stichwörtern aus Grande [10]

Die beschriebenen Qualitätsmerkmale sind bei der Anforderungsanalyse in gleichem Maße zu berücksichtigen wie die funktionalen Anforderungen, da Nachlässigkeiten in Bezug auf Qualitätsmerkmale den Erfolg von Entwicklungsprojekten gefährden und die Akzeptanz des Endprodukts bei den Stakeholdern verringern kann [8]. Von den Qualitätsmerkmalen und den funktionalen Anforderungen werden oftmals noch die Randbedingungen beziehungsweise Einschränkungen – organisatorische oder technologische Aspekte, die die Möglichkeiten zur Realisierung des Zielsystems im Projekt einschränken – abgegrenzt [8].

Da in Entwicklungsprojekten die Anzahl der funktionalen und nicht-funktionalen Anforderungen (sowie Einschränkungen) an das Zielsystem zumeist hoch ist, sollten diese für alle Projektbeteiligten in strukturierter und verständlicher Form dokumentiert werden [8]. Dabei lohnt sich der Einsatz von Satzschablonen zur Formulierung von Anforderungen, denn diese können dabei helfen, sprachliche Effekte wie missverständliche oder mehrdeutge Formulierungen zu reduzieren [9].

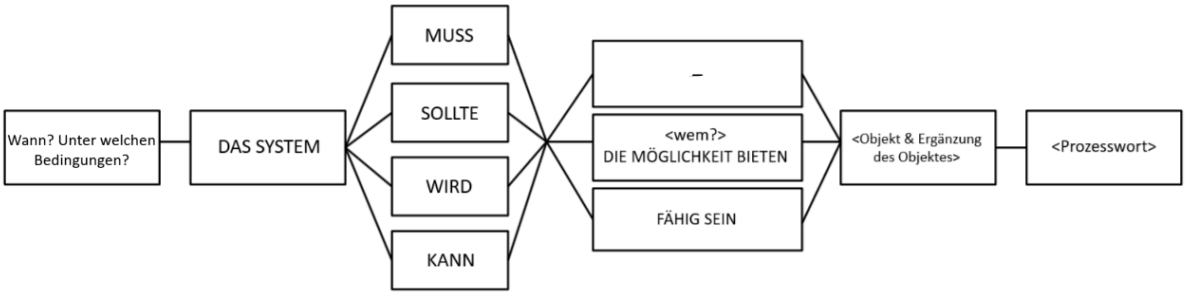


Abbildung 13 — Formulierungsschablone für funktionale Anforderungen nach Pohl und Rupp [8], übersetzt ins Deutsche in Anlehnung an Pohl und Rupp [9]

Abbildung 13 zeigt eine Satzschablone für die natürliche Formulierung von funktionalen Anforderungen, deren Anwendung universell verständliche Anforderungssätze ergeben sollte [8]. *Muss* drückt dabei meist eine zwingend zu erfüllende Anforderung aus, während *wird* eher einen Fakt oder einen Zweck darlegt und *sollte* oftmals ein Ziel beschreibt [28]. Die ausformulierten Anforderungen müssen letztlich an einer zentralen Stelle für alle Projektbeteiligten zugänglich dokumentiert werden [8].

Die Anforderungsdokumentation, bei der Sorgfalt als entscheidender Faktor für die Qualität und Nutzbarkeit des geplanten Systems gilt, sollte insbesondere auch die Dokumentation des Systemkontexts umfassen [27]. Hier können Standardentwürfe helfen. Diese geben

als Vorlagen für die Anforderungsdokumentation eine grobe Struktur vor und können beziehungsweise sollten an die jeweiligen Bedingungen des betreffenden Entwicklungsprojekts angepasst werden [8]. Der Nutzung von Standardentwürfen für Anforderungsdokumentationen werden in der Arbeit von Pohl und Rupp [8] die folgenden konkreten Vorteile allgemein zugeschrieben:

1. *Standardentwürfe vereinfachen die Eingliederung neuer Mitarbeitender in den Arbeitsprozess.*
2. *Standardentwürfe ermöglichen ein einfacheres Auffinden gewünschter Anforderungsinhalte.*
3. *Standardentwürfe ermöglichen selektives Lesens und Validieren von Anforderungsdokumenten.*
4. *Standardentwürfe ermöglichen die automatische Verifizierung von Anforderungsdokumenten (zum Beispiel bezüglich ihrer Vollständigkeit).*
5. *Standardentwürfe ermöglichen eine einfachere Wiederverwendung der Inhalte von Anforderungsdokumenten.*

Der ISO/IEC/IEEE Standard 29148:2011 enthält einen solchen Standardentwurf für die Dokumentation von Softwareanforderungen [8]. In der aktuellen Version des Standards (29148:2018) werden vor allem Prozesse und Leitlinien definiert, die sich auf Entwicklungsarbeiten mit (Software-)Systemen als Produkt und die Arbeit mit Anforderungen beziehen [29]. In beiden Versionen sieht der Standard die folgenden fünf Hauptkapitel für die Anforderungsdokumentation vor [9], [29], [30].

1. Einleitung und generelle Beschreibung der Software
2. Auflistung aller Dokumente, die in der Anforderungsdokumentation referenziert werden
3. Vollständige Auflistung aller konkreten Systemanforderungen
4. Auflistung der geplanten Verifikationsmaßnahmen
5. Anhänge

Unabhängig davon, welcher Standardentwurf für die Dokumentation von Anforderungen in einem Softwareprojekt gewählt worden ist, sollten uner anderem die Systemumgebung sowie die Systemfunktionalität darin festgehalten sein. Zur Beschreibung der Systemumgebung wird überlicherweise eine System- und Kontextabgrenzung durchgeführt. Die Systemgrenze separiert das geplante System von seinem Systemkontext [9]. Der Systemkontext ist jener Teil der Umgebung eines Systems, der für die Definition und das Verständnis der Anforderungen an dieses System relevant ist, wie beispielsweise bestimmte Benutzendeneigenschaften, interagierende Systeme oder Geschäftsprozesse [9].
Die Beschreibung der Funktionalität eines Systems hingegen erfolgt häufig anhand von Use-Cases (Anwendungsfällen). Der Use-Case stellt eine Interaktion einer Person oder eines anderen Systems mit dem betreffenden dar, die zu einem bestimmten Mehrwert führt. [9]

2.2.2 Test Driven Development

Test Driven Development (TDD) ist ein Konzept der Programmierung, welches Entwickelnde dazu befähigt, den Entwicklungsprozess aufs Testen zu fokussieren [31]. Das Ziel bei der Anwendung von TDD ist es, durch das vorangehende Schreiben von Softwaretests zuverlässigen, einfach gestalteten und sauberen Code zu produzieren [32].

Das Testen selbst ist in eine Praktik, die der IEEE-Standard 610.12-1990 im wesentlichen definiert als den Prozess, ein System oder eine Systemkomponente unter spezifizierten Bedingungen zu betreiben, die Ergebnisse dessen zu beobachten und aufzunehmen und darauf basierend eine Evaluation vorzunehmen [26]. Bei diesen spezifizierten Bedingungen handelt es sich um sogenannte Testfälle, die es vor dem Testen zu planen, zu analysieren und zu gestalten gilt [33]. Jeder Testfall sollte so gestaltet sein, dass er einen Input, einen Output und eine Ausführungsreihenfolge besitzt [33]. Obwohl auf diese Weise theoretisch jeder mögliche Testfall definiert werden kann, ist es grundsätzlich nicht möglich, alle Aspekte des Verhaltens einer Software zu testen [33]. Stattdessen werden sogenannte Unit-Tests vollzogen, die die einzelnen, wesentlichen Funktionen einer Software auf dem grundlegendsten Level verifizieren [31].

Beim Testen wird allgemein unterschieden zwischen dem Development Testing und dem Operational Testing. Beim Development Testing wird das Softwaresystem fortwährend in seinem Entwicklungsprozess getestet [26]. Beim Operational Testing hingegen wird das System oder eine Systemkomponente nach der Implementierung im Einsatz evaluiert [26].

Darüber hinaus können weitere Testarten unterschieden werden, von denen vier in Tabelle 2 zusammengefasst sind.

Testart	**Beschreibung**
Unit-Tests	Werden durchgeführt mit dem Ziel, die grundlegenden Funktionalitäten einer Anwendung zu testen, indem einzelne Codeabschnitte voneinander isoliert und unter vorbestimmten Bedingungen getestet werden. [31]
Regressionstests	Werden durchgeführt mit dem Ziel, nachzuweisen, dass die Modifikation einer Software keine Auswirkungen auf dessen bestehende Funktionalitäten hat. [34]
Integrationstests	Werden durchgeführt mit dem Ziel, die Interaktionsfähigkeit von Software- und/oder Hardwarekomponenten zu evaluieren, indem diese in Kombination getestet werden. [26]
Systemtests	Werden durchgeführt mit dem Ziel, die Einhaltung der Anforderungen eines Systems zu evaluieren, indem das vollständig entwickelte, integrierte System gestestet wird. [26]

Tabelle 2 — Übersicht über verschiedene Arten des Testens eines Softwaresystems

Die Softwareentwicklung mit TDD integriert das Testen in sein Vorgehen laut Beck [32] wie folgt:

1. Schreibe einen Test, der der aktuellen Vorstellung dessen entspricht, wie der Code funktionieren soll.
2. Lasse den Test laufen, um zu bestätigen, dass er fehlschlägt (*rot* anzeigt).
3. Schreibe den nötigen Code, um den Test *grün* anzeigen zu lassen, ohne dabei auf sauberen Code zu achten.
4. Schreibe den akzeptierten Code um in sauberen, den Konventionen entsprechenden Code und entferne Redundanzen, ohne dass der Test wieder auf *rot* umschlägt.

Neben der Produktion von sauberem Code hat TDD den Vorteil, dass besonders modularer Code durch das schrittweise Vorgehen entsteht. Das liegt daran, dass die Entkopplung von Funktionen im Code durch TDD gefördert wird, was ein modulares Design ergibt [35]. Darüber hinaus dienen die geschriebenen Tests zusätzlich als den Code erklärende Dokumentation und reduzieren im Allgemeinen die Zeit, die zum Debugging gebraucht wird [35].

3 Anforderungen

Die Erhebung und Dokumentierung der Anforderungen an die im Zentrum dieser Arbeit stehenden Anwendungssoftware sind Gegenstand dieses Kapitels. Die erarbeitete Anforderungsdokumentation hat die beiden Funktionen, die gemeinsamen Vorstellungen von Stakeholderseite und Entwicklungsseite über das Zielsystem der vorliegenden Arbeit in finaler Version darzustellen und nach vollendeter Entwicklung als Maßstab für den Implementierungserfolg zur Verfügung zu stehen.

3.1 Anforderungsanalyse

Zu Beginn des Entwicklungsprozesses wurde ein initiales Stakeholder-Interview durchgeführt, in welchem durch Erfragung der an die Anwendungssoftware gerichteten Vorstellungen des Stakeholders ein geteiltes Grundverständnis der mit der Anwendung zu erreichenden Ziele geschaffen wurde. Über den gesamten Entwicklungsprozess hinweg wurde mit wenigen Ausnahmen wöchentlich ein Stakeholdergespräch gehalten, in welchem die Zwischenergebnisse und der Projektfortschritt diskutiert wurden und hinzukommende Erwartungen und Aufgaben besprochen wurden. Auf diese Weise sind die an das Zielsystem gerichteten Anforderungen stets aktuell gehalten worden. Nach den ersten vier Wochen des Entwicklungsprozesses sind keine Anforderungen mehr verändert worden und keine weiteren Anforderungen sind hinzugenommen worden, sodass von diesem Zeitpunkt an die Softwareentwicklung auf Grundlage der final vereinbarten Anforderungen erfolgen konnte.

3.2 Anforderungsdokumentation

In enger Orientierung an die im ISO/IEC/IEEE-Standard 29148:2011 enthaltenen Leitlinien sind in den nachfolgenden Abschnitten die dokumentierten Anforderungen an die Software vollständig beschrieben worden. In der Anforderungsdokumentation sind ausschließlich diejenigen Anforderungen und Systemeigenschaften aufgeführt, die der finalen Vereinbarung zwischen Stakeholderseite und Entwicklungsseite entsprechen. Am Ende von Kapitel 3.2.1 sind die Definitionen einiger in der Anforderungsdokumentation enthaltener Fachbegriffe und Konzepte aufgeführt.

3.2.1 Einleitung und Übersicht

Zweck
Die Anforderungsdokumentation hat die strukturierte Wiedergabe der angestrebten Systemeigenschaften sowie der vereinbarten Systemanforderungen zum Zweck. Sie dient als *Single Source of Truth* über das Entwicklungsziel für alle am Projekt beteiligten Personen.

Systemumfang
Zu entwickeln ist ein Anwendungsprogramm namens StackSAR. Die Anwendung hat zum Ziel, die Auffindung und die Koregistrierung von Radaraufnahmen sowie die Generierung von Bild- und Videomaterial zu automatisieren und dadurch Produktionsprozesse in der Öffentlichkeitsarbeit des Instituts effizienter zu gestalten.

Stakeholder
Als Stakeholder gelten in erster Linie alldiejenigen Mitarbeitenden des Instituts für Hochfrequenztechnik und Radarsysteme, die für die Erstellung von Präsentationsmaterial aus SAR-Aufnahmen verantwortlich sind. Für sie stellt die Gruppenleitung der Fachgruppe *Missions Engineering* repräsentativ die zentrale Stakeholderfigur dar. Diese Fachgruppe leistet strategische Arbeit für die Erdbeobachtung und kontrolliert die Qualität der SAR-Aufnahmen [36]. Zusätzlich gilt das IT-Management des Instituts für Hochfrequenztechnik und Radarsysteme als Stakeholder, da dieses mit der Betreuung des Projekts vertraut ist und zukünftig anfallende Aufgaben der Wartung und Weiterentwicklung der Software übernehmen wird. Weitere Stakeholder des Entwicklungsprojekts sind die Mitarbeitenden, deren Arbeit zur Produktion und Verarbeitung von SAR-Aufnahmen beiträgt, denn durch die Öffentlichkeitsarbeit wird die Qualität der Arbeit dieser Mitarbeitenden aktiv beworben.

Systemumgebung
Zur Beschreibung der Systemumgebung von StackSAR erfolgte eine System- und Kontextabgrenzung. Das nach außen hin abgegrenzte StackSAR-System soll durch die in Kapitel 3.2.3 enthaltenen Anforderungen beschrieben werden. Der Systemkontext ergibt sich aus den in Tabelle 3 aufgeführten Kontextaspekten, die zu dem StackSAR-System eine Beziehung haben. Die Kategorien der Kontextaspekte in Tabelle 3 sind aus den in Pohl und Rupp [9] vorgeschlagenen Typen von Aspekten im Systemkontext ausgewählt worden.

Zusammenfassend ergibt sich der Systemkontext von StackSAR aus den das System bedienenden, weiterentwickelnden und wartenden Personen sowie aus dem Quellsystem der für die Systemfunktionalität relevanten Daten.

Aspekttyp	Kontextaspekt von StackSAR
Personen	Das System zukünftig nutzende Stakeholder, das System zukünftig wartende oder weiterentwickelnde Stakeholder
Systeme im Betrieb	Langzeitdatenbank mit allen Datatake Requests und Quicklooks
Alt-/Vorgängersysteme	IDL-Code zur Geokodierung
(Geschäfts-)Prozesse	Öffentlichkeitsarbeit
Dokumente	Referenzdokument mit den initialen Stakeholdervorstellungen zum Produkt

Tabelle 3 — Die Kontextaspekte des StackSAR-Systems, zugeordnet zu den Aspekttypen nach Pohl und Rupp [9]

Ein Aspekt des StackSAR-Systems, welcher nicht zum Systemkontext sondern zur (irrelevanten) Systemumgebung gehört, ist das interne Netzwerk des DLR, denn dieses muss automatisch genutzt werden, damit über StackSAR auf die Metadaten der SAR-Aufnahmen zugegriffen werden kann und beeinflusst daher die übrigen Anforderungen nicht.

Architekturbeschreibung
Der Systementwurf des zentralen Stakeholders sieht eine Kombination aus vier Systemkomponenten vor. In Abbildung 14 ist der originale Entwurf gezeigt. Von den darin gezeigten vier Systemkomponenten sollen im Rahmen der vorliegenden Arbeit die folgenden drei Komponenten entwickelt werden.

1. *Find Stacks of SAR images or DEMs:*
   Unterstützt Anwendende bei der Analyse der zeitlichen und räumlichen Verteilung aller gemachten Datatake Requests, sodass diese für die Öffentlichkeitsarbeit geeignete Stacks anhand ihrer aufbereiteten Aufnahmeparameter erkennen können.

2. *Coregistration:*
   Koregistriert die SAR-Aufnahmen eines Stacks anhand von Koregistrierungsverfahren nach dem Prinzip der Merkmalsdetektion und -zuordnung oder anhand von Geokodierung, sodass diese aneinander angeglichen sind.

3. *Movie (and image) generator:*
   Produziert aus einem Stack ein Video oder ein Bild, welches die Veränderungen in den enthaltenen Aufnahmen über die Zeit darstellt. Videos geben die im Stack enthaltene Zeitserie als chronologische Aneinanderreihung von Aufnahmen wieder, während Bilder den Grad der Veränderung zwischen den Aufnahmen der Zeitserie farblich darstellen.

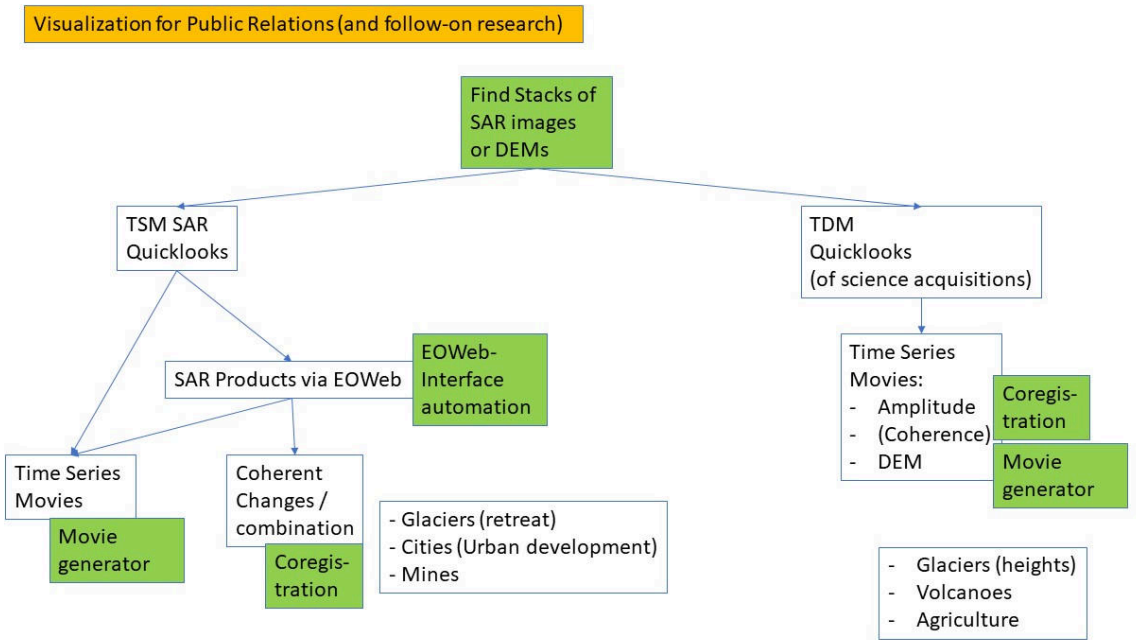


Abbildung 14 — Systementwurf des Stakeholders aus dem projektinternen Referenzdokument

Die erste Systemkomponente ist von der zweiten und der dritten Systemkomponente entkoppelt, weil die in Abbildung 14 vorhandene EOWeb-Interface Automation für diese Arbeit nicht angedacht ist und daher vorerst nicht implementiert wird.

Systemfunktionalitäten

Die geplante Funktionalität des StackSAR-Systems ist anhand des in Abbildung 15 dargestellten Use-Case-Diagramms festgehalten. Den primären Akteur stellt die das System nutzende Person dar. Diese kann zwei Aktivitäten des Systems auslösen, welche wiederum eine Reihe von Folgeaktivitäten bedingen. Der logischen Aktionsfolge nach gehend würde der Akteur zunächst *Find stacks of SAR images or DEMs* aufrufen. StackSAR zeigt dann die Heatmap mit den in den Daten identifizierten Stacks an und gibt zeitgleich die Tabelle aus, die die Informationen zu diesen Stacks enthält.

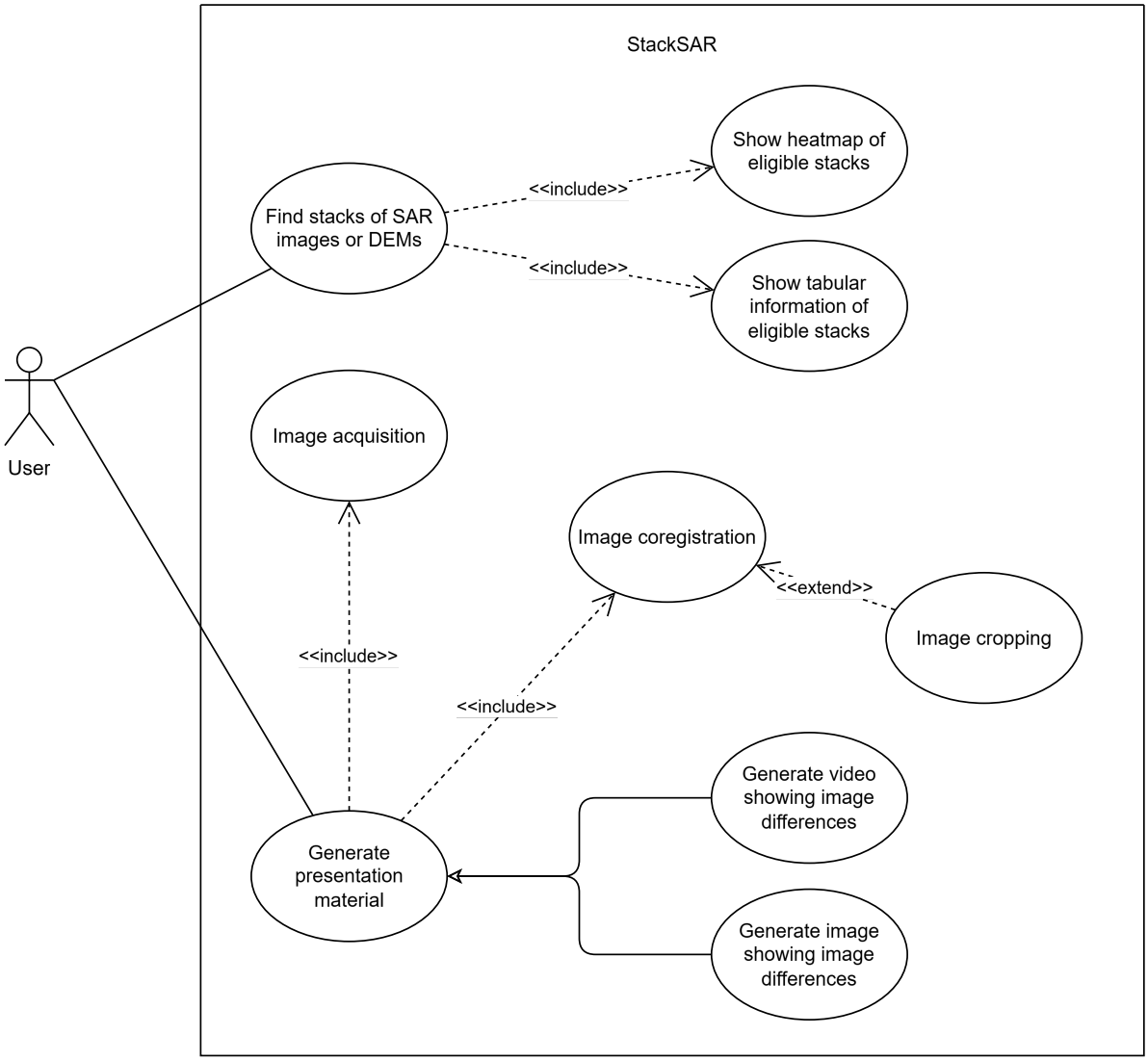


Abbildung 15 — Use-Case-Diagramm zur Darstellung der StackSAR-Systemfunktionalitäten

Im Anschluss an die *Find stacks of SAR images or DEMs*-Aktivität besteht die Möglichkeit, die Generierung von Präsentationsmaterial auszulösen. Dies führt umgehend zu der Bildkoregistrierung („Image Coregistration“ in Abbildung 15) der vorhandenen Quicklooks. Die für StackSAR vorgesehene Funktionalität der Bildbeschaffung („Image Acquisition“ in Abbildung 15) nach der Stack-Auswahl soll im aktuellen Projekt noch nicht umgesetzt werden. Möglicherweise erfolgt im Anschluss an die Koregistrierung auch ein Zuschneiden der angeglichenen Bilder („Image Cropping“ in Abbildung 15). Abschließend erfolgt die Herstellung des Präsentationsmaterials in Form eines Videos oder eines Bildes („Generate Presentation Material“ in Abbildung 15).

Einschränkungen
Der Lösungsraum, der für StackSAR zur Bearbeitung der Problemstellung des Entwicklungsprojekts möglich ist, ist insbesondere durch den zeitlichen Rahmen der vorliegenden Arbeit eingeschränkt. Für den Stakeholderentwurf des StackSAR-Systems sind noch einige weitere Funktionalitäten denkbar und auch erwünscht, davon wurden aufgrund der vorgegebenen Bearbeitungszeit jedoch nicht alle in die Anforderungsdokumentation aufgenommen.

Ein weiterer, den Lösungsraum des Projekts einschränkender Faktor ist das Satellitendatensicherheitsgesetz SatDSiG. Es handelt sich dabei um das Gesetz zum Schutz vor Gefährdung der Sicherheit der Bundesrepublik Deutschland durch das Verbreiten von hochwertigen Erdfernerkundungsdaten [37], welches die Wahrung der sicherheits- und außenpolitischen Interessen zum Ziel hat und insbesondere die Verbreitung der durch die Satelliten TerraSAR-X und TanDEM-X aufgenommenene Fernerkundungsdaten reguliert [38]. Die Implementierung von Verarbeitungsfunktionen mit hochaufgelösten SAR-Aufnahmen ist daher im Rahmen des aktuellen Projekts nicht möglich.

Annahmen
Es wird angenommen, dass die zukünftigen Nutzenden von StackSAR als wissenschaftliche Mitarbeitende des DLR ein grundlegendes Verständnis von beziehungsweise erste Erfahrungen mit der Bedienung von Anwendungen auf der Kommandozeile besitzen.

Definitionen
Ein *Datatake* ist eine Radaraufnahme.

Datatake Requests sind Anfragen von SAR-Aufnahmen bestimmter Positionen auf der Erdoberfläche. Sie gehen gewöhnlich von Mitgliedern der wissenschaftlichen Gemeinschaft oder von wirtschaftlich agierenden Parteien aus und werden in der Langzeitdatenbank des Instituts für Hochfrequenztechnik und Radarsysteme mitsamt der Aufnahmeparameter festgehalten.

Ein *Stack* ist eine Ansammlung von SAR-Aufnahmen einer bestimmten Position oder Region auf der Erdoberfläche.

Supersites sind per Referenzdokument Standorte, die über die Lebenszeit der TerraSAR-X/TanDEM-X-Mission regelmäßig aufgezeichnet worden sind.

Quicklooks sind Bilddateien, die eine SAR-Aufnahme in geringerer Qualität darstellen.

Eine *Zeitserie* ist eine im Aufnahmebestand vorhandene Gruppe von SAR-Aufnahmen, deren Aufnahmezeitpunkte eine chronologische Reihe ergeben, sodass Informationen über die Veränderungen im aufgenommenen Ort abgeleitet werden können.

3.2.2 Referenzen

Dem Projekt stehen als Referenzen neben der in der Arbeit zitierten Fachliteratur insbesondere eine projektinterne Powerpoint-Präsentation mit Vorschlägen und Wünschen des Stakeholders zur Verfügung. Zudem existert in der Programmiersprache Interactive Data Language (IDL) verfasster Quellcode, der im Institut bisher zur Geokodierung eingesetzt worden ist.

3.2.3 Anforderungen

Funktionale Anforderungen

	Anforderung
A1	Das StackSAR-System muss dem Zielgruppenpersonal die Möglichkeit bieten, aus SAR-Aufnahmen der TerraSAR-X/TanDEM-X-Mission Präsentationsmaterial zu erstellen.
A2	Das StackSAR-System muss dem/der Nutzenden die Möglichkeit bieten, die Anwendung über eine Konfigurationsdatei zu steuern.
A3	StackSAR-System muss dem/der Nutzenden die Möglichkeit bieten, geeignete Daten zur Erstellung des Präsentationsmaterials auszuwählen.
A4	Wenn der/die Nutzende die Heatmap-Routine aufruft, muss das StackSAR-System die in der Datenbank enthaltenen Aufnahmeparameter in Form einer Heatmap ausgeben.
A5	Wenn die Heatmap ausgegeben wird, muss das StackSAR-System darauf die existierenden Supersites anzeigen.
A6	Das StackSAR-System muss die in der Datenbank enthaltenen Aufnahmedaten in geografische Cluster einteilen.

	Anforderung
A7	Wenn der/die Nutzende die Heatmap-Routine aufruft, muss das StackSAR-System die Cluster mit den dazugehörigen Aufnahmeparametern in Form einer tabellarischen Liste ausgeben.
A8	Das StackSAR-System muss die ausgewählten Bilddaten zur Erstellung des Präsentationsmaterials einlesen.
A9	Das StackSAR-System muss die SAR-Aufnahmen der Zeitserie in der chronologisch korrekten Reihenfolge zusammensetzen.
A10	Das StackSAR-System muss mit Koregistrierungs- oder Geokodierungsalgorithmen die SAR-Aufnahmen aus der Zeitserie aneinander angleichen.
A11	Wenn der/die Nutzende die Video-Routine aufruft, muss das StackSAR-System aus den eingelesenen SAR-Aufnahmen ein Video generieren, in welchem diese der Zeitserie entsprechend nacheinander eingeblendet werden.
A12	Das StackSAR-System muss dem/der Nutzenden die Möglichkeit bieten, die Einblendungsdauer der Aufnahmen im Video zu konfigurieren.
A13	Das StackSAR-System muss dem/der Nutzenden die Möglichkeit bieten, den Grad der Transparenz bei den Übergängen im Video zu konfigurieren.
A14	Das StackSAR-System muss dem/der Nutzenden die Möglichkeit bieten, die SAR-Aufnahmen durch die Angabe von Eckkoordinaten zuzuschneiden.
A15	Wenn der/die Nutzende die Image-Routine aufruft, muss das StackSAR-System aus den SAR-Aufnahmen der Zeitserie in Verarbeitung ein überlagertes Bild generieren, in welchem die veränderlichen Bereiche entsprechend des Veränderungsgrades eingefärbt sind.

Tabelle 4 — Die an StackSAR gerichteten funktionalen Anforderungen

Nicht-Funktionale Anforderungen

Qualitätsmerkmal	Anforderungen
Änderbarkeit	Q1 StackSAR soll über das aktuelle Projekt hinaus weiterentwickelt werden und muss daher problemlos modifizierbar sein. Q2 StackSAR soll langfristig nutzbar sein und muss daher wartbar und entsprechend testbar sein.
Benutzbarkeit	Q3 Die Verständlichkeit von StackSAR soll durch Anweisungen für Nutzende sowie nachvollziehbaren Code unterstützt werden. Q4 StackSAR soll über die Kommandozeile bedient werden und muss für diese Anwendungsform nutzungsfreundlich sein.
Effizienz	Q5 StackSAR ist keine zeitkritische Anwendung und darf Laufzeiten von mehreren Stunden bis zu wenigen Tagen aufweisen. Q6 Der Ressourcenverbrauch durch die Nutzung von StackSAR ist hinsichtlich des Einlesens der Daten vom Server nicht limitiert.
Funktionalität	Q7 StackSAR soll die Daten so verarbeiten, dass die Richtigkeit der ausgegebenen Informationen stets gewährleistet ist.
Übertragbarkeit	-
Zuverlässigkeit	Q8 StackSAR soll reproduzierbare, deterministische Ergebnisse liefern.

Tabelle 5 — Die an die Qualitätsmerkmale (nach ISO/IEC 9126) von StackSAR gerichteten nicht-funktionalen Anforderungen

3.2.4 Testplanung

Development Testing

StackSAR ist nach dem Prinzip des Test-Driven Development zu entwickeln. Dabei sind zunächst Unit-Tests zu schreiben, die jeweils einer Funktion oder Funktionalität gewidmet

sind. Die Implementierung jeder neuen Funktion eines Software-Moduls ist abzuschließen mit einem Regressionstest, der sicherstellt, dass alle bisgerigen für das Modul geschriebenen Tests auch nach Einfügen der neuen Funktion noch positiv ausfallen. Fertige oder vorerst fertige Module werden dann einem Integrationstest unterzogen, indem sie in die aktuellste Version von StackSAR integriert werden und eine Ausfürung des Systems durch den geplanten Anwendungsmechanismus auf der Kommandozeile stattfindet.

Operational Testing
Am Ende der Entwicklung von StackSAR ist ein Systemtest durchzuführen, sodass eingeschätzt werden kann, welche Anforderungen an das System im bis dorthin erfolgten Entwicklungsprozess erfüllt worden sind und welche weiteren Entwicklungsschritte notwendig sein werden. Zusätzlich wird die zentrale Stakeholderperson das StackSAR-System in einem Usability-Test anwenden und eine Einschätzung der Benutzertauglichkeit geben.

3.2.5 Anhänge

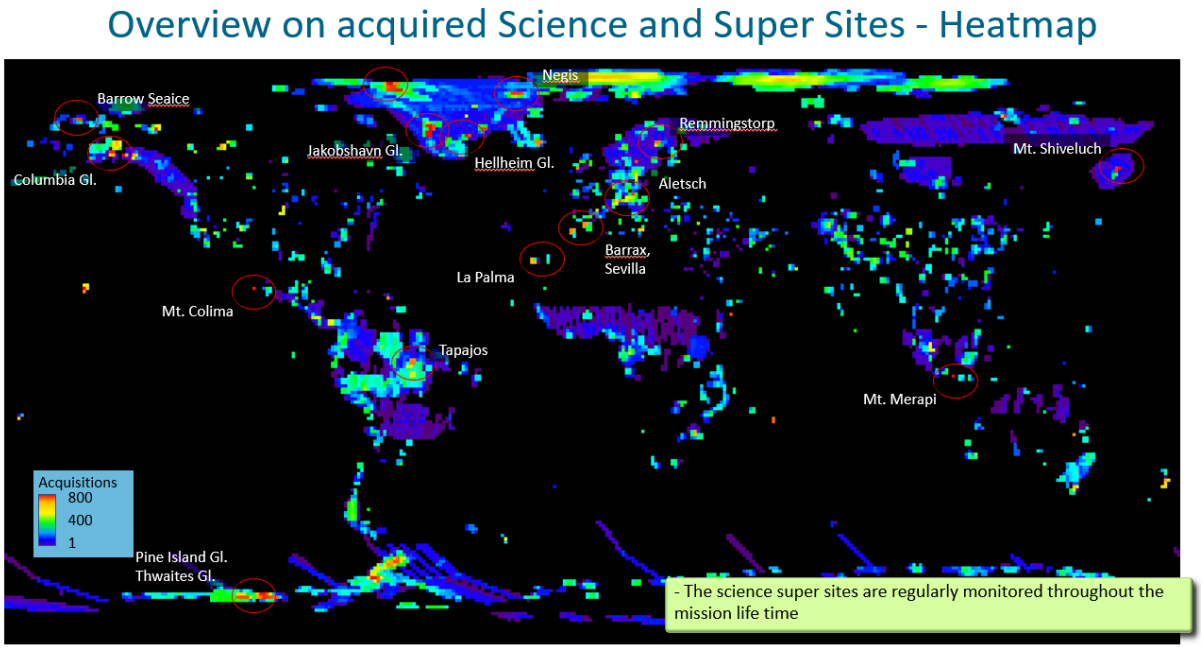


Abbildung 16 — Die im Referenzdokument enthaltene Beispielgrafik als Vorschau dessen, wie die Parameterdaten der Datatake Requests in Form einer Heatmap dargestellt werden können. Die dargestellte Heatmap beinhaltet manuell hinzugefügte Markierungen von Supersites, welche mit StackSAR automatisch aus den Daten identifiziert und in der Heatmap markiert werden sollen.

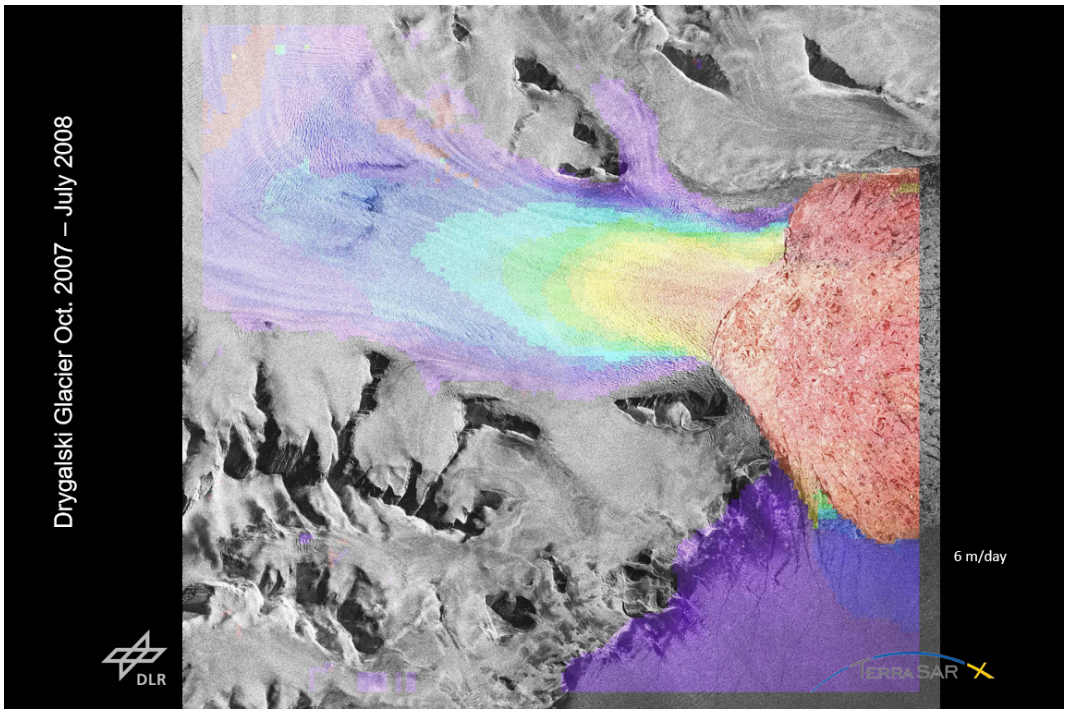


Abbildung 17 — Die im Referenzdokument enthaltene Beispielgrafik als Vorschau dessen, wie die farbliche Darstellung der in den SAR-Aufnahmen eines Stacks enthaltenen Veränderungen über die Zeit umgesetzt werden kann

4 Algorithmenstudie und -auswahl

Die Bearbeitung des Teilziels, geeignete Algorithmen zur Implementierung der automatisierten Bildkoregistrierung auszuwählen (siehe Kapitel 1.2), ist Gegenstand dieses Kapitels. Die automatisierte Koregistrierung ist für die Anwendungssoftware StackSAR von erheblicher Bedeutung, da sie als Vorverarbeitungsschritt der Erzeugung von Präsentationsmaterial aus Radaraufnahemn zwingend erfolgen muss. Zur Automatisierung der Bildkoregistrierung werden Algorithmen zur Detektion, zur Deskription und zur Zuordnung der Merkmalspunkte von SAR-Aufnahmen benötigt. Um geeignete Algorithmen für die Implementierung von StackSAR auszuwählen, wurde eine vergleichende Algorithmusstudie durchgeführt. Gesucht wurde darin nach einer Vorgehensweise zur Bildkoregistrierung, bei der die Detektion und die Deskription der Merkmalspunkte durch einen einschlägigen Algorithmus umgesetzt werden können. Folglich wurden in der Algorithmusstudie die Detektions- und Deskriptionsalgorithmen nicht getrennt betrachtet, sondern in Kombination untersucht.

In den nachfolgenden Abschnitten wird zunächst beschrieben, auf welche Weise die untersuchten Koregistrierungsalgorithmen recherchiert worden und als geeignet für die Projektarbeit identifiziert worden sind. Anschließend wird der Versuch dargelegt, mit welchem die Algorithmen erprobt worden sind und auf dessen Basis letztendlich der zu implementierende Algorithmus ausgewählt wurde.

4.1 Literaturrecherche

Im Vorfeld des Versuchs wurde gegenwärtige Fachliteratur zum aktuellen Forschungsstand der Koregistrierung von SAR-Aufnahmen recherchiert. Anhand der gefundenen Veröffentlichungen wurden dann Koregistrierungsalgorithmen identifiziert, bei denen von einer Eignung für die Umsetzung der automatisierten Koregistrierung mit StackSAR ausgegangen werden kann. Dafür wurden die in den Veröffentlichungen enthaltenen Beschreibungen der verschiedenen Koregistrierungsmethoden analysiert und die beschriebenen Studienergebnisse berücksichtigt.

In der umfangreichen systematischen Übersichtsarbeit von Liu et al. [22] werden sechs verschiedene herkömmliche und häufig herangezogene Algorithmen zur Detektion und sechs

zur Deskription von Merkmalspunkten jeweils untereinander verglichen. Die in der Übersichtsarbeit enthaltene Vergleichsstudie an gewöhnlichen Lichtbildern hatte zum Ergebnis, dass die Detektionsalgorithmen ORB und KAZE – erklärt im nachfolgenden Abschnitt – zuverlässiger Merkmalspunkte identifizieren als die vier anderen untersuchten Algorithmen [22]. Insbesondere an Bilderpaaren mit verschieden starker Beleuchtung, unterschiedlichen Kompressionsraten oder variierter Bildschärfe zeigte sich ihr Einsatz als vorteilhaft. Darüber hinaus zeigte sich ORB robuster im Umgang mit Bildpaaren unterschiedlicher Perspektive, Skalierung oder Rotation [22].

ORB steht für *Oriented FAST and Rotated BRIEF* und basiert auf den Detektionsalgorithmen FAST (*Features from Accelerated Segment Test* [39]) und BRIEF (*Binary Robust Independent Elementary Features* [40]) [41]. Der ORB-Detektionsalgorithmus wurde entwickelt mit dem Ziel, Merkmalpunkte in Bildern trotz vorhandenem Rauschen oder unterschiedlichen Rotationsgraden zuverlässig detektieren zu können [41]. KAZE wiederum ist japanisch für „Wind“ und basiert auf einer neuen Herangehensweise, die sich von den klassischen Merkmalsdetektoren wie SIFT (*Scale Invariant Feature Transform* [42]) fundamental unterscheidet [43]. KAZE wurde entwickelt mit dem Ziel, bei der Detektion die Objektgrenzen in den Bildern präzise zu erfassen und dadurch die Genauigkeit bei der Lokalisierung von Merkmalspunkten sowie deren Unterscheidbarkeit zu erhöhen [43].

Die vergleichende Untersuchung der sechs Algorithmen zur Deskription von Merkmalen in Liu et al. [22] lieferte im Gegensatz zur Untersuchung der Detektionsalgorithmen kein einschlägiges Ergebnis darüber, welcher oder welche der Algorithmen sich an den eingesetzten Lichtbildern als grundsätzlich vorteilhafter erwiesen.

4.2 Versuch

Die Ergebnisse der Literaturrecherche suggerieren, dass die Wahl des Algorithmus zur Merkmalsdetektion einen größeren Einfluss nimmt auf den Erfolg der Bildkoregistrierung als die Wahl des Algorithmus zur Deskription der gefunden Merkmalspunkte. Deshalb wurde die Entscheidung getroffen, im Versuch die zusammenhängende Merkmalsdetektion und -deskription mittels ORB-Algorithmus zu vergleichen mit der zusammenhängenden Merkmalsdetektion und -deskription mittels KAZE-Algorithmus. Einzelne Vergleiche der

jeweiligen Detektion mit ORB- und KAZE-Algorithmus sowie der jeweiligen Deskription mit ORB- und KAZE-Algorithmus waren für den Versuch nicht vorgesehen.

An die Auswahl der zu vergleichenden Algorithmen zur Detektion und zur Deskription von Merkmalspunkten schloss sich noch die Wahl eines Algorithmus zur gegenseitigen Zuordnung der Merkmalsvektoren der einzelnen Bilder an. Zum Abgleichen und Zuordnen der Merkmalsvektoren im Versuch wurde die Fast Library for Approximate Nearest Neighbors (FLANN) [44] herangezogen. Dessen Suchalgorithmen ordnen nach dem Nearest-Neighbor-Prinzip die Punkte zweier Mengen möglichst passend einander zu [44]. Weil dadurch sowohl die mit ORB ermittelten Merkmalsvektoren als auch die mit KAZE ermittelten Merkmalsvektoren zweier Bilder einander zugeordnet werden können, war die FLANN geeignet, im Versuch Vergleichbarkeit der beiden Algorithmen herzustellen. In den nachfolgenden Abschnitten ist beschrieben, wie der Versuch konkret aufgebaut war, welche Materialien und Software zum Einsatz kamen und wie sich die Versuchsdurchführung gestaltete.

4.2.1 Bildmaterial

Für die Vergleichsstudie stellte das Institut für Hochfrequenztechnik und Radarsysteme TerraSAR-X-Aufnahmen in Form von Quicklooks zur Verfügung. Darin ist jeweils die Region um den Bodenkalibrierungstransponder der Mission Biomass dargestellt, die sich in New Norcia (Australien) befindet [45]. Insgesamt handelt es sich um 28 zur Verfügung gestellte Aufnahmen. Diese liegen in Paketen vor und sind zu unterschiedlichen Zeitpunkten in den Jahren 2023 und 2024 aufgenommen worden. Alle Aufnahmen eines Pakets entstanden ungefähr im selben Zeitraum, besitzen jedoch verschiedene Aufnahmekonfigurationen. Das bedeutet, dass die Aufnahmeparameter wie zum Beispiel die Position, aus der aufgenommen wurde, oder der Winkel, in dem aufgenommen wurde, oder die Dauer der Aufnahme sich zu einem gewissen Grad unterscheiden. Für den Versuch wurden nur solche Aufnahmen berücksichtigt, deren Aufnahmekonfiguration auch auf Aufnahmen in anderen Paketen zutrafen, sodass auf Basis der Konfigurationen Aufnahmegruppen erstellt werden konnten. Die fünf verschiedenen Aufnahmekonfigurationen *Spot 031*, *Spot 034*, *Spot 073*, *Spot 109* und *Spot 112* kamen in jedem der Pakete bei genau einer Aufnahme vor, sodass für den Versuch pro Konfiguration jeweils fünf Radaraufnahmen mit derselben Aufnahmekonfiguration zur Verfügung standen.

4.2.2 Software

Für die Versuchsdurchführung ist im Vorfeld ein Softwaretool entwickelt worden, welches zur automatisierten Durchführung der Koregistrierungen mit den auszutestenden Algorithmen diente. Das Zuordnen der Merkmalsvektoren wurde mit der bereits erwähnten FLANN sowie mit der Open-Source-Bibliothek OpenCV (Open Computer Vision) implementiert. OpenCV ist eine Softwarebibliothek, die mehrere hundert Computer-Vision-Algorithmen beinhaltet und mit Python kompatibel ist [46]. Mit diesen Bibliotheken konnte das Matching so implementiert werden, dass es sowohl mit den ORB-Merkmalsvektoren als auch mit den KAZE-Merkmalsvektoren kompatibel ist. Dabei war lediglich zu spezifizieren, mit welcher Art von Indizes der FLANN-Algorithmus zum Suchen der Matches arbeiten wird [47]. Das ist deshalb nötig, weil die ORB-Deskriptoren binär sind [41], während KAZE-Deskriptoren mit Gleitkommazahlen dargestellt werden [43]. Dementsprechend unterschieden sich die Implementierungen der Merkmalszuordnungen mit FLANN für die ORB-Deskriptoren und die KAZE-Deskriptoren gezwungenermaßen in diesem Punkt voneinander [44].

Anhand des für den Versuch implementierten Softwaretools konnten die koregistrierten Aufnahmen auch zu Videos zusammengesetzt und als solche gespeichert werden. Das Tool wurde in der Programmiersprache Python implementiert, welche im Institut für Hochfrequenztechnik und Radarsysteme die Standardsprache für Anwendungsentwicklung ist. Für die Umsetzung der ausgewählten Koregistrierungsalgorithmen und der Zusammensetzung der Zeitreihen zu Videos wurde ebenso auf OpenCV zurückgegriffen.

4.2.3 Messung

Die grundlegende Annahme des Versuchs ist, dass die Anzahl gelungener Zuordnungen der Merkmalsvektoren eines Bildes zu denen eines zweiten Bildes ein Indikator dafür ist, wie geeignet der zur Merkmalsdetektion und -deskription eingesetzte Algorithmus ist. Folglich wurde im vorliegenden Versuch der Vergleich von ORB- und KAZE-Vorgehen zur Bildkoregistrierung anhand der Anzahl gelungener Zuordnungen mittels FLANN-Matcher gezogen. Zur Bestimmung eines gelungenen Matches, also einer gelungenen Zuordnung zweier Merkmalsvektoren, wurde dabei der Schwellenwerttest nach Lowe herangezogen, der wie folgt entscheidet: Seien q ein Kennzeichner von Bild A und p_1 und p_2 der am besten und der am zweitbesten zu q passende Kennzeichner von Bild B, basierend auf der Distanz d und dem Schwellenwert T. Dann wird p_1 der Menge der gelungenen Zuordnungen bei Zutreffen der folgenden Bedingung zugeordnet [48]:

$$\frac{d(q, p_1)}{d(q, p_2)} < T \tag{1}$$

Das bedeutet, dass die Distanz vom besten Match zum Vektor q um einen festgelegten Grad kleiner sein muss als die Distanz vom zweitbesten Match zu q, je nach Schwellenwert T, der in diesem Versuch bei 0.7 liegt. Aus ORB und KAZE soll hier diejenige Vorgehensweise als besser geeignet bewertet werden, aus deren detektierten und als Merkmalsvektoren gekennzeichneten Merkmalspunkten sich die größere Menge gelungener Matches ergibt.

4.2.4 Durchführung

Für die Erprobung der Bildkoregistrierung mittels ORB und KAZE und die anschließende Erstellung des Videomaterials wurde zunächst das Tool in Python entwickelt. Die Implementierungen der Bildkoregistrierung und der Videogenerierung in diesem Tool orientierten sich an den OpenCV-Python-Leitfäden [49]. Mit dem fertigen Tool wurde die Versuchsdurchführung vorgenommen. Vor der Versuchsdurchführung waren die Radaraufnahmen der Region um den Bodenkalibrierungstransponder der Mission Biomass in fünf Ordner organisiert worden, entsprechend ihrer Aufnahmekonfigurationen: *Spot 031*, *Spot 034*, *Spot 073*, *Spot 109* und *Spot 112*.

Das Tool war so implementiert worden, dass vor jeder Ausführung im Programmcode der Pfad zum Quellordner mit den zu koregistrierenden Radaraufnahmen angegeben werden musste sowie der Name der einzusetzenden Koregistrierungsmethode (wobei nur „ORB“ und „KAZE“ als Angaben zulässig waren). Der Versuch wurde dann durchgeführt, indem mit dem Tool für jeden der fünf Bildordner eine Bildkoregistrierung mit der ORB-Methode und eine Bildkoregistrierung mit der KAZE-Methode ausgeführt wurde. Im Detail wurden die Koregistrierungen so abgewickelt, dass das Tool zunächst das größte Bild aus dem angegebenen Ordner als Referenzbild definierte und daran dann alle weiteren Bilder des Ordners einzeln anglich, unter Verwendung der angegebenen Koregistrierungsmethode. Bei jeder Koregistrierung wurde bei der Zuordnung der ORB- beziehungsweise KAZE-Merkmalsvektoren festgehalten, wie viele gelungene Zordnungen der Lowe-Test ergab. Die angeglichenen Bilder wurden gespeichert. Im letzten Schritt jeder Ausführung generierte das Tool aus den angeglichenen Bildern ein Video, indem es sie in alphanumerischer Reihenfolgte aneinanderfügte.

4.3 Auswertung

Die für jede Koregistrierung aufgezeichneten Anzahlen der gelungenen Zuordnungen wurden mittels Durchschnittsbildung ausgewertet. Pro Bildordner und Koregistrierungsmethode wurde ein Durchschnitt berechnet, wobei in der Berechnung außenvorgelassen wurde, wie viele gelungene Zuordnungen das Matching des Referenzbildes mit sich selbst ergab. Die Durchschnittswerte der gelungenen Zuordnungen pro Bildgruppe sind in der folgenden Tabelle wiedergegeben.

Aufnahmekonfiguration	**ORB**	**KAZE**
Spot_031	470.00	94.50
Spot_034	886.50	100.25
Spot_073	861.50	203.25
Spot_109	139.00	40.50
Spot_112	1335.75	747.25

Tabelle 6 — Durchschnittliche Anzahlen der gelungenen Zuordnungen von Kennzeichnern der Bilder zum Referenzbild pro Bildordner.

In der Tabelle ist ersichtlich, dass die Koregistrierung unter Anwendung von ORB zur Detektion und Kennzeichnung von Merkmalspunkten für jede Bildgruppe im Durchschnitt mehr gelungene Zuordnungen ergab als unter Anwendung von KAZE. Eine bildliche Darstellung dieses Unterschieds soll durch die beiden nachfolgenden Abbildungen gegeben sein, in denen für zwei Beispielaufnahmen derselben Aufnahmekonfiguration, also derselben Bildgruppe, die gelungenen Zuordnungen anhand von Linien dargestellt sind.

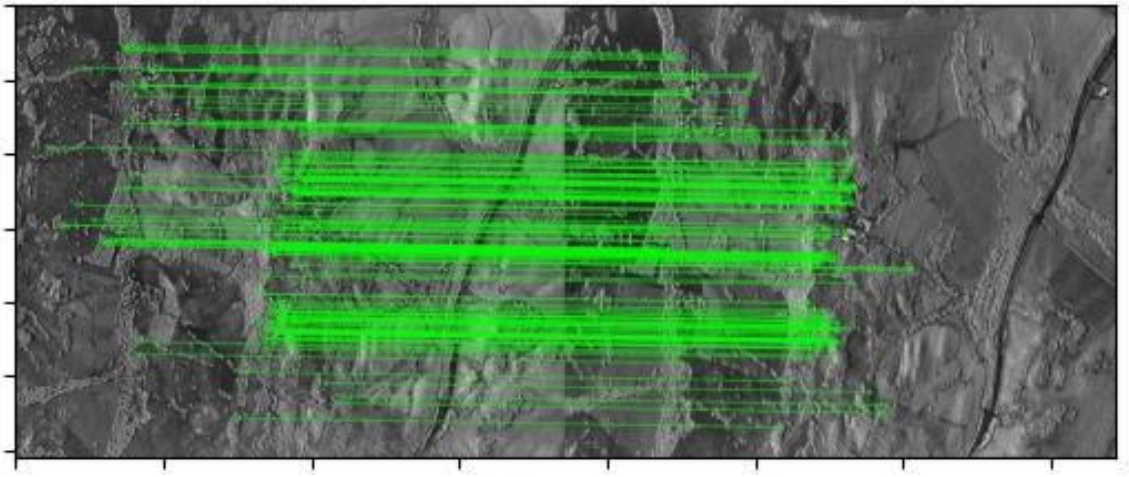


Abbildung 18 — Darstellung der gelungenen Zuordnungen von Merkmalsvektoren zweier Radaraufnahmen. Jede grüne Linie verbindet einen ORB-Merkmalsvektoren in der linken Radaraufnahme mit einem ORB-Merkmalsvektoren in der rechten Radaraufnahme.

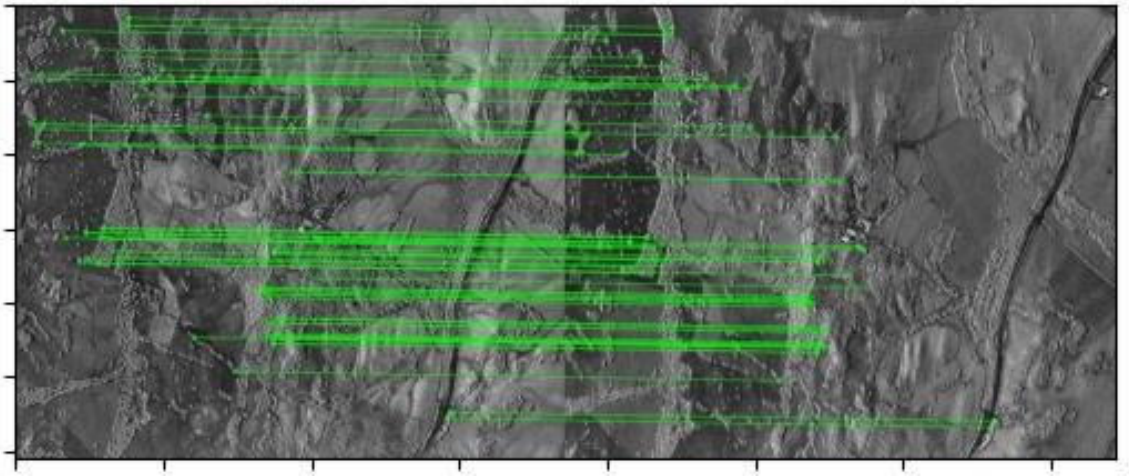


Abbildung 19 — Darstellung der gelungenen Zuordnungen von Merkmalsvektoren zweier Radaraufnahmen. Jede grüne Linie verbindet einen KAZE-Merkmalsvektoren in der linken Radaraufnahme mit einem KAZE-Merkmalsvektoren in der rechten Radaraufnahme.

Die Unterschiede zwischen den Anzahlen der gelungenen Zuordnungen bei der Verwendung von ORB gegenüber KAZE ist in den nach den Koregistrierungen generierten Videos mit dem Auge nicht erkennbar. Alle in den Versuchen generierten Videos entsprachen in ihrer Qualität den Ansprüchen der zuständigen Mitarbeitenden am Institut für Hochfrequenztechnik und Radarsysteme.

5 Implementierung

Das aktuelle Kapitel gibt wieder wie in der Anwendungsentwicklung zur Erreichung der Implementierungsziele (siehe Kapitel 1.2) vorgegangen wurde. Der Entwicklungsprozess von StackSAR folgte der geplanten, in Kapitel 1.3 beschriebenen Vorgehensweise und orientierte sich inhaltlich an den in Kapitel 3.2.3 beschriebenenen funktionalen und nicht-funktionalen Anforderungen. Der Einsatz von TDD zur Implementierung von StackSAR reultierte in einem inkrementellen Entwicklungsprozess, bei dem die Anforderungsdefinition sowie das Design, die Implementierung und das Testen der Software der Definition [26] entsprechend auf überlappende, iterative Weise erfolgten. Dadurch ergab sich eine schrittweise Fertigstellung der StackSAR-Software, die die nachfolgend beschriebenen Ergebnisse hat.

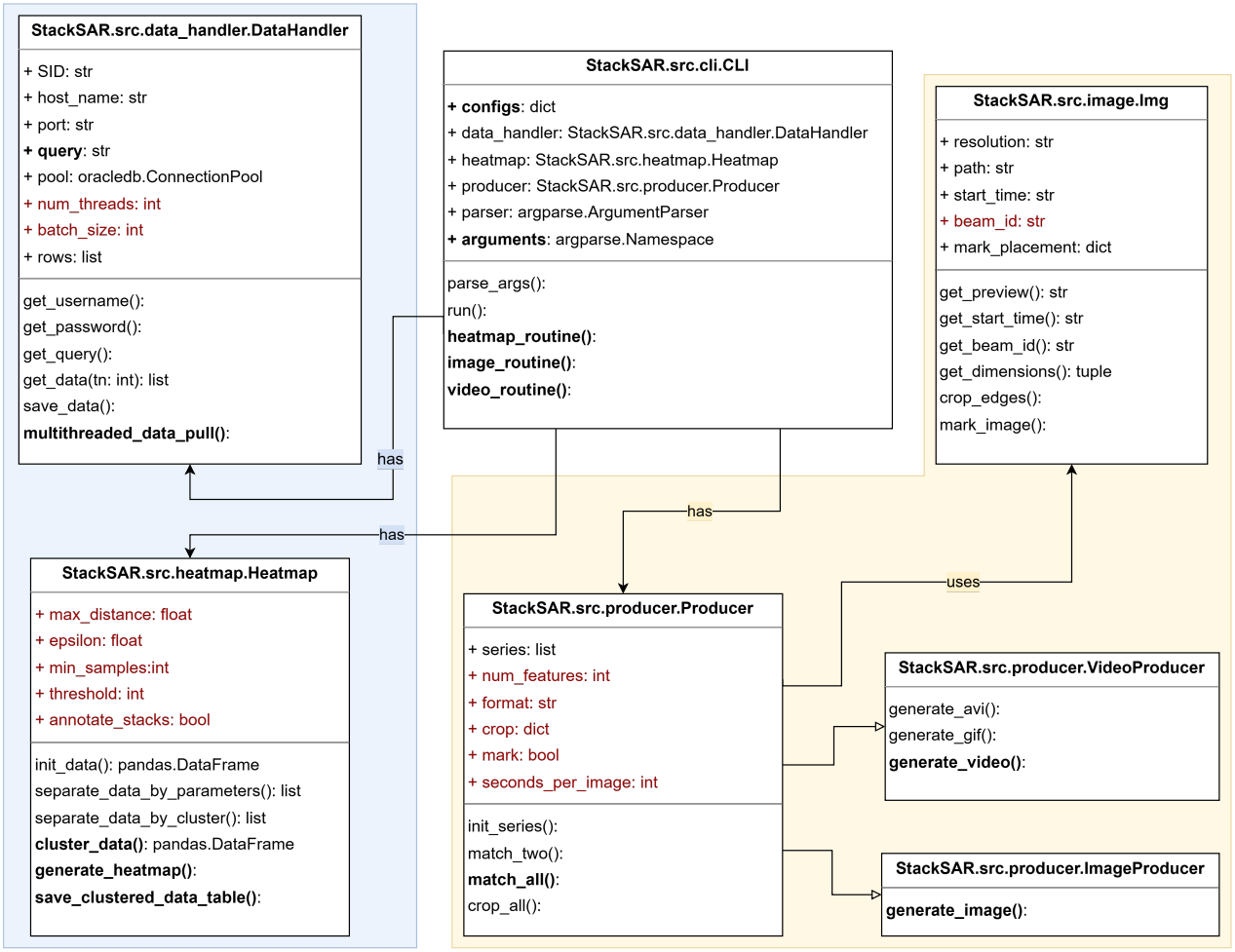


Abbildung 20 — Reduziertes Klassendiagramm vom Aufbau der StackSAR-Software

Das in Abbildung 20 gezeigte Klassendiagramm gibt den Entwicklungsstand der Anwendungssoftware StackSAR zum Ende der Bearbeitungsphase dieses Projekts wieder. Der Kern der Software ist vollständig aus Klassen zusammengesetzt, welche sich aus der hier genutzten objektorientierten Programmierung ergeben. Objektorientiertes Softwaredesign zeichnet sich dadurch aus, dass ein System oder eine Systemkomponente in in Objekte und Objektverbindungen übersetzt wird [26]. Für StackSAR eignete sich dieser Ansatz, da die in der Anforderungsdokumentation festgehaltenen, geforderten Funktionalitäten bereits klar in zwei übergeordnete Hauptfunktionalitäten abgegrenzt waren, wie im Use-Case-Diagramm (Abbildung 15) in Kapitel 3.2.1 ersichtlich ist. Die zwei Hauptfunktionalitäten – das Auffinden von Stacks und die Generierung von Präsentationsmaterial – wurden mit dem objektorientierten Ansatz in der Form mehrerer separater Klassen implementiert. Die Klassen stellen klar voneinander getrennte Software-Einheiten dar, die jeweils über bestimmte Attribute (klasseneigene Variablen) und Methoden (klasseneigene Funktionen) verfügen. Aus dem Klassendiagramm in Abbildung 20 sind für die Erklärung der Implementierung irrelevante Attribute und Methoden entfernt worden.

Im Zentrum steht die `CLI`-Klasse (Command-Line-Interface-Klasse), welche die Nutzungsschnittstelle auf der Kommandozeile repräsentiert. Sie implementiert die übergeordneten Methoden (Routinen), die ausgeführt werden, wenn StackSAR genutzt wird. Konkret gibt es drei verschiedene Routinen, die über die Kommandozeile von Nutzenden aufgerufen werden können: die Heatmap-Routine, die Image-Routine und die Video-Routine. Erstere stellt die Umsetzung der Hauptfunktionalität *Auffinden von Stacks* dar, wobei die letzteren beiden die Umsetzung der Hauptfunktionalität *Generierung von Präsentationsmaterial* darstellen. Die Methode `heatmap_routine()` der `CLI`-Klasse definiert die Heatmap-Routine und nutzt dazu Instanzen der blau hinterlegten Klassen. Die Methoden `image_routine()` und `video_routine()` definieren die Image- und die Video-Routine und nutzen dazu Instanzen der gelb hinterlegten Klassen. Unabhängig davon, welche Routine von StackSAR Anwendende über die Kommandozeile aufrufen, wird zunächst ein `CLI`-Objekt erzeugt. Dieses verarbeitet zunächst das Routine-Argument, welches beim Aufrufen über die Kommandozeile mitgegeben wird und entweder „heatmap“, „image“ oder „video“ lauten kann. Bevor das `CLI`-Objekt dann die der genannten Routine entsprechende Methode aufruft, liest es aus der Konfigurationsdatei die Konfigurationsoptionen ein, damit diese an ein Objekt der `Heatmap`-Klasse, der `ImageProducer`-Klasse oder der `VideoProducer`-Klasse weitergegeben werden können.

5.1 TDD und Versionsverwaltung

Wie im Testplan der Anforderungsdefinition (siehe Kapitel 3.2.4) gefordert, ist bei der Softwareentwicklung nach TDD-Prinzip vorgegangen worden. Darüber hinaus ist im Entwicklungsprozess fortwährend Versionsverwaltung mit Git betrieben worden. Die zu entwickelnden Funktionalitäten wurden zu umfassenden Features zusammengefasst und jedes Feature auf einem eigenen Git-Branch entwickelt. Auf den Git-Branches ist jeweils an den Python-Modulen gearbeitet worden, deren enthaltene Klassen das Feature realisieren sollten. Zu jedem Python-Modul wurde ein Test-Modul angelegt und darin in einer `unittest.TestCase`-Klasse die dazugehörigen Tests geschrieben.

Anhand von einem Flussdiagramm wird in Abbildung 21 gezeigt, wie TDD und die Versionsverwaltung mit Git zu dem Workflow der Entwicklung von StackSAR kombiniert worden sind. Aufgrund der TDD-üblichen Vorgehensweise kann dieser genutzte Workflow grob auf die fortwährende Entwicklung einzelner Funktionen heruntergebrochen werden. Das Flussdiagramm in Abbildung 21 stellt den Ablauf bei der Entwicklung einer neuen Einzelfunktion von StackSAR dar. Für jede neue Funktion wurde zunächst ein einzelner Unit-Test geschrieben, der die zu entwickelnde Funktion aufruft und dabei zu einer bestimmten Eingabe ein bestimmtes Ergebnis sicherstellt. Dieser Test sollte beim ersten Ausführen rot sein, also fehlschlagen, und eine entsprechende Fehlermeldung liefern. Solange der Unit-Test fehlschlug, wurde aus den jeweiligen Fehlermeldungen der zu schreibende Code abgeleitet und geschrieben, und anschließend erneut getestet. Sobald der Unit-Test grün war, also bestanden war, wurde der Commit geschrieben. Anschließend erfolgte das *Refactoring*, das Aufräumen des geschriebenen Codes, wobei auch hier das Bestehen des Unit-Tests weiterhin sichergestellt wurde, bevor ein erneuter Commit erfolgte.

Im nächsten Schritt wurde in einem Regressionstest überprüft, ob die bis zu diesem Zeitpunkt geschriebenen Tests der aktuellen Test-Klasse noch immer grün sind und nicht durch das hinzukommen der neuen Funktionen umgeschlagen sind. Bestand der Regressionstest, folgte ein Integrationstest, bei dem der geschriebene Code in die bestehende Funktionalität von StackSAR integriert und über die Kommandozeile als Teil der Routine ausgeführt wurde. Für jeden der zuvor erwähnten Testdurchläufe gilt, dass bei einem Nicht-Bestehen im Workflow zurückgesprungen und der betreffende Code nachgebessert wurde, sodass alle Testdurchläufe dann erneut erfolgten.

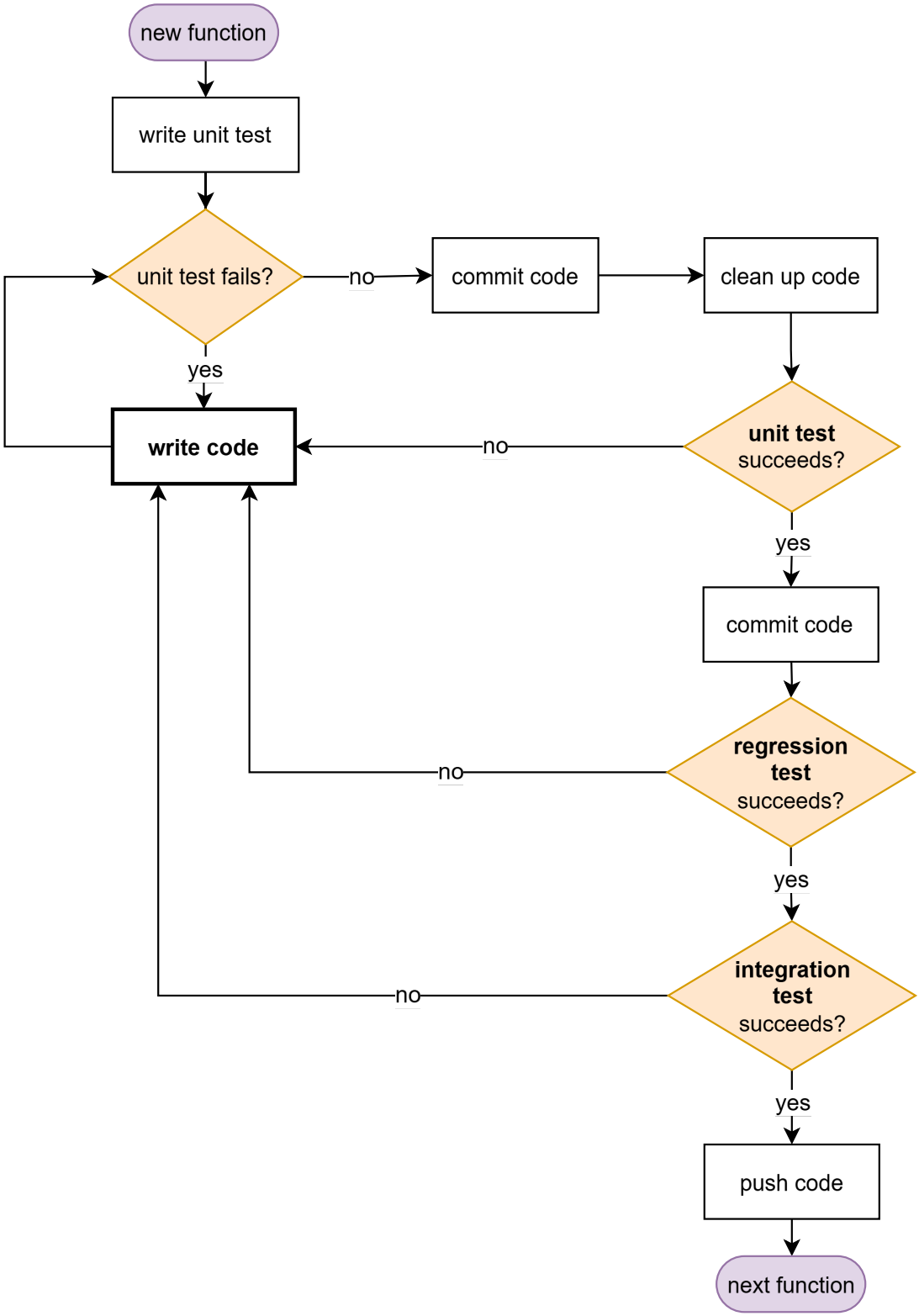


Abbildung 21 — Flussdiagramm vom Entwicklungsprozess einer StackSAR-Funktion

5.2 Nutzungsanleitung

Um zukünftigen Anwendenden die Installation und die Nutzungsweise von StackSAR zugänglich zu machen, wurden in mehreren Instanzen Nutzungsanleitungen implementiert. Zunächst wurde in der Nutzungsschnittstelle eine Hilfe-Funktionalität implementiert. Diese kann von der Kommandozeile aus mit dem Argument „–help“ oder „-h“ aufgerufen werden. Nach dem Aufruf wird ein kurzer Erklärungstext zu StackSAR in die Kommandozeile ausgegeben sowie ein Hinweis zur Nutzungsweise.

Eine weitere Nutzungsanleitung stellt die README-Datei dar, die dem Git-Repository des StackSAR-Projekts in Gitlab hinzugefügt wurde. Sie enthält vor allem konkrete Anweisungen zur korrekten Installation von StackSAR aus Gitlab heraus. Darüber hinaus sind darin detaillierte Hinweise zur Nutzung von StackSAR enthalten.

Abschließend beinhaltet StackSAR eine Konfigurationsdatei mit Kommentaren, die Anwendenden Hinweise zur Steuerung geben. Die Konfigurationsdatei stellt für die StackSAR-Software die Quelle der Konfigurationsoptionen dar. Diese Optionen werden aus der Konfigurationsdatei eingelesen, sobald StackSAR angewendet wird. Die Optionen betreffen alle der drei Hauptfunktionalitäten von StackSAR und geben vor, wie StackSAR die Metadaten und Bilddaten verarbeitet.

5.3 Automatisierte Analyse des Aufnahmebestands

Das erste Ziel des Entwicklungsprozesses war die Automatisierung der Analyse und Bereitstellung der Metadaten über die TerraSAR-X/TanDEM-X-Radaraufnahmen. Durch diese Automatisierung sollte das Identifizieren von Stacks im Datenbestand, die sich für die Herstellung von Präsentationsmaterial eignen, von den DLR-Mitarbeitenden abgenommen werden. Dafür wurde zunächst vom Stakeholder in Erfahrung gebracht, welche Kriterien für ihn bestimmen, wann ein Stack von Radaraufnahmen sich für die Darstellung von Veränderungen in Form von Präsentationsmaterial eignet. Die Befragung ergab eine konkrete Liste an Aufnahmeparametern, die im Idealfall für einen gesamten Stack übereinstimmen sollten, damit sich dieser für die Darstellung von Veränderungen in Videos oder Bildern eignet. In Tabelle 7 sind diese Parameter aufgeführt und in kurzer Form erklärt.

Aufnahmeparameter	Erklärung
imagingMode	Bezeichnet unter anderen mit SpotLight („SL“), High Resolution SpotLight („HS“) oder StripMap („SM“) den Modus der Radaraufnahme (siehe Kapitel 2.1.2 für ausführlichere Erklärungen).
beamID	Bezeichnet mit „spot“, „strip“, „stripNear“, „stripFar“ (unter anderen) und einer daran angehangenen Identifikationsnummer die Strahlkonfiguration der Radaraufnahme.
polarization	Bezeichnet mit „VV“, „HH“, „HV“ oder „VH“ die Polarisation. SAR-Signale können vertikal oder horizontal zur Erdoberfläche ausgesendet und empfangen werden, wodurch sich die vier Kombinationsmöglichkeiten ergeben [50].
looking	Bezeichnet mit Links („L“) oder Rechts („R“) die Satelliten-Blick-Richtung, aus der die Radaraufnahme gemacht wurde.

Tabelle 7 — Liste der kritischen Aufnahmeparameter, deren Übereinstimmung einen geeigneten Stack charakterisiert.

Wie aus Tabelle 7 hervorgeht, beschreiben die kritischen Aufnahmeparameter, die die Eignung eines Stacks bestimmen, insbesondere die Konfiguration der einzelnen Radaraufnahmen. Sie sind ein Ausschnitt der Gesamtheit an Daten über die Datatake Requests (Metadaten), die zu Beginn der Heatmap-Routine aus der TerraSAR-X/TanDEM-X-Langzeitdatenbank geholt werden. In Abbildung 22 ist dargestellt, welchem Fluss diese Daten in der implementierten Heatmap-Routine von StackSAR folgen.

Rufen Anwendende von StackSAR die Heatmap-Routine über die Kommandozeile auf, wird zunächst die Beschaffung der Metadaten („Threaded Data Pull“ in Abbildung 22) aus der Langzeitdatenbank („LTDB“ in Abbildung 22) gestartet. Dafür werden von drei externen Entitäten Informationen abverlangt – aus der LTDB die zu beschaffenden Metadaten, von der das System anwendenden Person („User“ in Abbildung 22) die Zugangsdaten zur LTDB und von der Konfigurationsdatei noch zusätzliche Angaben, die für den gesamten Datenfluss in der

Heatmap-Routine relevant sind. Um auf diese Informationen zuzugreifen, wird zu Beginn der Routine die Konfigurationsdatei eingelesen und ein Objekt der Klasse `DataHandler` erzeugt. Dieses Objekt besitzt alle zur Datenbeschaffung benötigten Attribute und Methoden (siehe Abbildung 20) zur Erlangung der Zugangsdaten, zum Abfragen der Datenbank und zum Speichern der noch unverarbeiteten Metadaten. Anhand des `DataHandler`-Objekts werden als erster Schritt der Heatmap-Routine die Metadaten mit den Aufnahmeparametern der Datatake Requests beschafft und gespeichert, sodass diese als Comma-Separated-Values-Datei (CSV-Datei) lokal vorliegen.

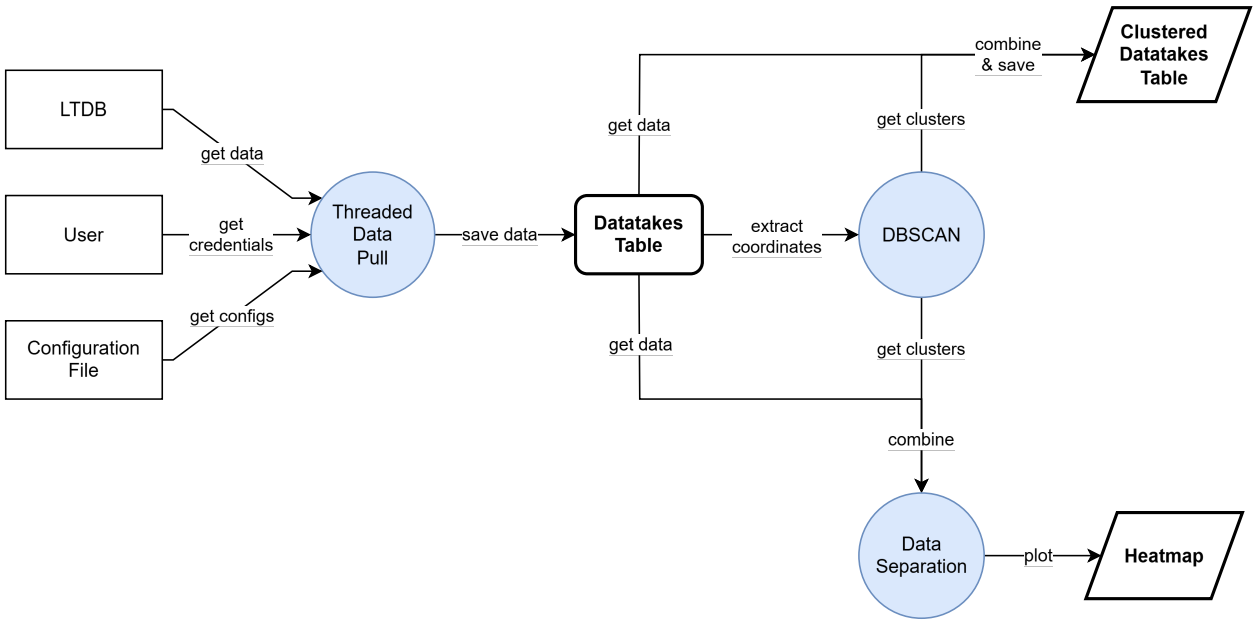


Abbildung 22 — Datenflussdiagramm von der Verarbeitung der Datatake Requests zur Heatmap durch StackSAR

Im nächsten Schritt der Heatmap-Routine werden die nun lokal vorliegenden Metadaten der Datatake Requests weiterverarbeitet, um die Heatmap und die Tabelle zu erzeugen. Der erste Verarbeitungsschritt ist das Clustering („DBSCAN“ in Abbildung 22), mit dem die in den Metadaten enthaltenen Koordinaten aller Aufnahmen zu Clustern zugeordnet werden. Dabei werden die Koordinaten zuerst extrahiert und daraufhin der Density-Based-Spatial-Clustering-of-Applications-with-Noise-Algorithmus (DBSCAN-Algorithmus) auf sie angewandt. Der DBSCAN-Algorithmus wurde für das Clustering der Koordinatendaten ausgewählt, da dieser spezifisch für große räumliche Datenmengen entwickelt wurde [51]. Der Einsatz von DBSCAN birgt insbesondere den Vorteil, dass Datenpunkte, die aufgrund eines festgelegten

Maximalabstands in den Clustern keinem Cluster eindeutig zugeordnet werden können, als Rauschen ignoriert werden [51]. Dadurch konnte bei der Implementierung StackSAR-Anwendenden die Möglichkeit eingeräumt werden, die Maximalgröße (Radius) der Cluster eigens vorzubestimmen. Auch die Mindestanzahl der Datenpunkte, die in einem Cluster enthalten sein sollen, kann beim Einsatz von DBSCAN zuvor festgelegt werden [51]. Sie kann ebenso wie die Maximalgröße der Cluster von StackSAR-Anwendenden durch Anpassen der in der Konfigurationsdatei enthaltenen Optionen konfiguriert werden.

Das Clustering ergibt eine Liste, die jeder in den DBSCAN-Algorithmus gegebenen Koordinate eine Cluster-Identifikationsnummer (Cluster-ID) zuordnet. Der nächste Schritt in der Heatmap-Routine besteht darin, die Cluster-IDs mit den aus der Langzeitdatenbank beschafften Metadaten zu kombinieren. Dies wird mit der `pandas.concatenate()`-Operation der Open-Source-Bibliothek Pandas bewerkstelligt, aus der auch alle weiteren in der StackSAR-Implementierung genutzten Tabellen-Operationen und -Datenstrukturen stammen. Wie in Abbildung 22 zu sehen, teilt sich die Heatmap-Routine an dieser Stelle auf.

Einerseits wird nun die zusammengesetzte Datatakes-Tabelle mit den hinzugefügten Clusterzugehörigkeiten in dem in der Konfigurationsdatei angegebenen Output-Ordner gespeichert,

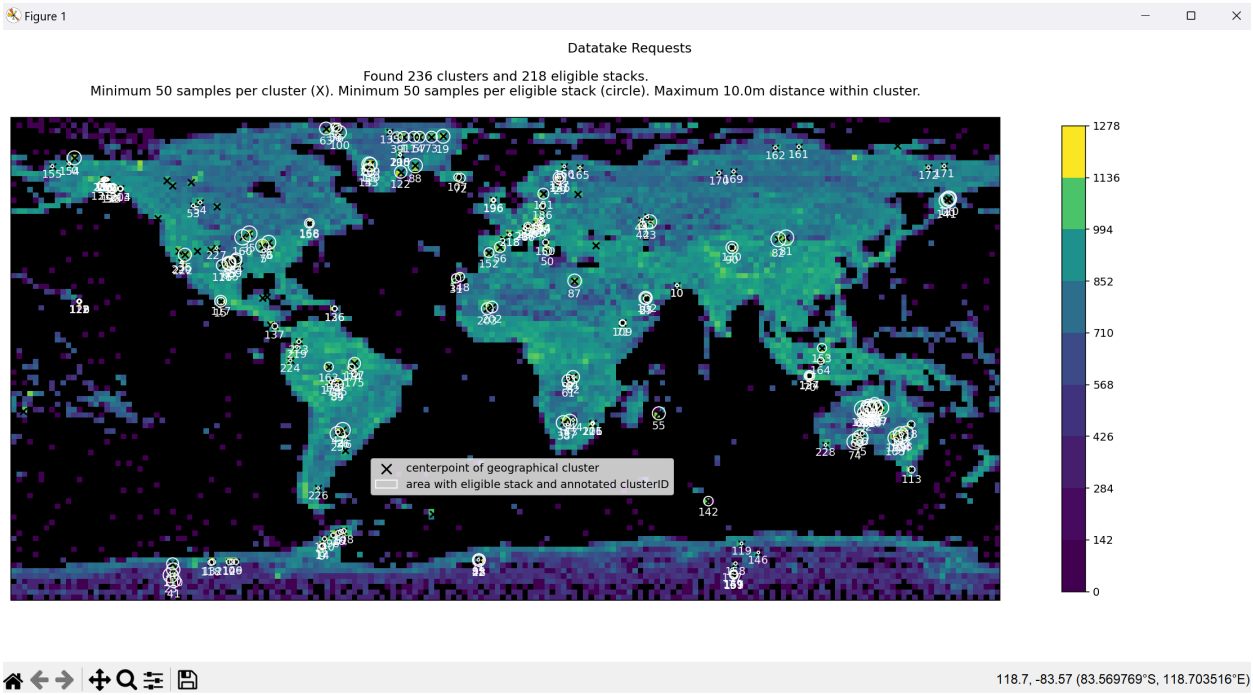


Abbildung 23 — Anzeige der aus den Metadaten der Datatake Requests generierten Heatmap

sodass sie dort von StackSAR-Anwendenden aufgerufen werden kann. Andererseits erfolgt die Generierung der Heatmap, die zum Schluss der Heatmap-Routine entsprechend Abbildung 23 angezeigt wird. Die Anzeige der Heatmap beinhaltet neben der Heatmap selbst Informationen über die Anzahl der berechneten Cluster und die Anzahl der identifizierten Stacks und gibt noch einmal die vor der Ausführung der Routine festgelegten Parameter wieder. Darüber hinaus sind links am unteren Rand Widgets vorhanden, mit denen Anwendende in der Ansicht zoomen und schieben können und die aktuelle Darstellung speichern können. Am unteren Rand rechts werden Anwendenden zu jedem Zeitpunkt die Koordinaten angezeigt, denen die Position des Cursors auf der Heatmap entspricht. Eine Legende, die mit der Maus über die Heatmap bewegt werden kann, sodass Anwendende alle Bereiche ansehen können, gehört ebenfalls zur Anzeige der Heatmap.

Die Heatmap selbst setzt sich zusammen aus einem zweidimensionalen Histogramm (2D-Histogramm), welches die Häufigkeiten der Koordinateneinträge aus der Metadatentabelle anzeigt sowie aus den darauf eingezeichneten Clusterzentren und Stacks. Die Umsetzung beider Komponenten der Heatmap wurde mit Datenstrukturen und Methoden aus der Open-Source-Bibliothek matplotlib implementiert. Die aus den Metadaten extrahierten Koordinatendaten werden zunächst zu einer $n \times 2$-Matrix umgeformt und dann als 2D-Histogramm kartiert. Das 2D-Histogramm besteht aus 180×90 quadratischen Containern, die jeweils einen Farbwert annehmen. Die Einfärbung ist heller, je mehr Aufnahmen in dem entsprechenden Bereich auf der Erdoberfläche erfolgt sind. In Abbildung 24 ist die Heatmap mit größerem Zoom gezeigt. Wie anhand der Farbskala der Heatmap in Abbildung 23 und Abbildung 24 zu erkennen ist, ist für die Einfärbung der Histogramm-Container eine logarithmische Darstellung gewählt worden. Dadurch wurde bewirkt, dass die dichter befüllten Container überbetont und die weniger dicht befüllten Container unterbetont werden. In Folge dessen werden die Bereiche auf der Erdoberfläche, die aufgrund der Vielzahl an vorhandenen Aufnahmen für StackSAR-Anwendende interessanter sind, auf der Karte besser erkennbar, als dies bei einer linearen Darstellung der Farbwerte der Fall wäre.

Wenn die Kartierung des 2D-Histogramms erfolgt ist, werden abschließend die Cluster und Stacks darauf eingezeichnet. Die Cluster werden jeweils durch ihr Clusterzentrum dargestellt. Als Clusterzentrum wird mit Hilfe der Open-Source-Bibliothek shapely jene Koordinate eines Clusters ausgewählt, die sich am nächsten zum tatsächlichen geografischen Mittelpunkt des Clusters befindet.

Die Clusterzentren werden in der Heatmap mit den schwarzen Kreuzen markiert. Die in unterschiedlichen Größen eingezeichneten weißen Kreise hingegen stellen die Stacks dar, die zunächst in einem abschließenden Verarbeitungsschritt der Heatmap-Routine – der Datenseparierung („Data Separation“ in Abbildung 22) – berechnet werden müssen. Bei der Datenseparierung werden die als `pandas.DataFrame` vorliegenden Metadaten mit den ergänzten Cluster-IDs in mehreren Schritten zu kleinen DataFrames separiert.

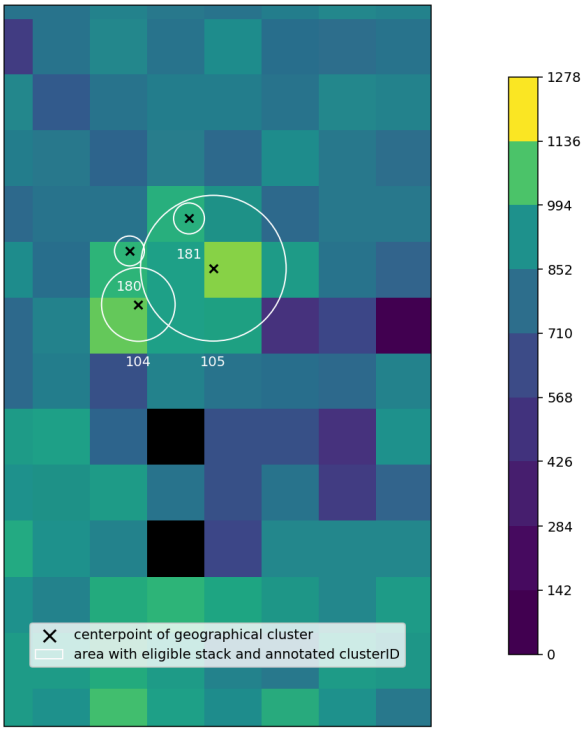


Abbildung 24 — Heatmap-Ausschnitt

Zunächst wird dabei der Frame aufgrund der Cluster-IDs separiert, sodass eine Liste mit DataFrames entsteht, die je einem Cluster zugeordnet sind. Die in dieser Liste einthaltenen DataFrames werden nun nach selbigem Prinzip wieder separiert, auf Grundlage ihrer Einträge für die Parameter *imagingMode*, *beamID*, *polarization* und *looking*. Entsteht bei der Separierung ein DataFrame, dessen Anzahl an Datenpunkten kleiner ist als ein in der Konfigurationsdatei festgelegter Schwellenwert, wird dieser fallen gelassen. Nach der Datenseparierung liegt eine Liste an DataFrames vor, in der jeder DataFrame eine einzigartige Kombination aus Cluster-ID und kritischen Parametern (siehe Tabelle 7) besitzt und den Schwellenwert der mindestens enthaltenen Datenpunkte nicht unterschreitet. Jeder dieser DataFrames repräsentiert einen Stack von Aufnahmen, aus denen Präsentationsmaterial generiert werden kann, denn die enthaltenen Daten stimmen jeweils in den kritischen Parametern überein und gehören demselben geografischen Cluster an. Jeder Stack wird auf der Heatmap als ein weißer Kreis eingezeichnet. Anwendende von StackSAR können vor der Ausführung der Heatmap-Routine in der Konfigurationsdatei festlegen, ob die Kreise mit der ClusterID ihres Clusters annotiert werden sollen. Die Ausführung der Heatmap-Routine ist dann beendet, wenn das Fenster mit der Anzeige der Heatmap geschlossen wurde.

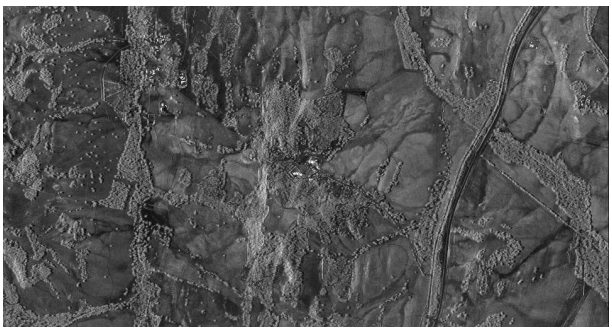
5.4 Automatisierung der Bildkoregistrierung

Das zweite Implementierungsziel für die StackSAR-Anwendungssoftware (siehe Kapitel 1.2) war die Automatisierung der Bildkoregistrierung zur Angleichung von für Präsentationsmaterial ausgewählten Aufnahmen. Zur Implementierung der Merkmalsdetektion, Merkmalsdeskription und Merkmalspunktzuordnung wurde auf die in Kapitel 4 beschriebene Software zur Durchführung der Vergleichsstudie zurückgegriffen. Die darin implementierte Bildkoregistrierung mit ORB-Algorithmus wurde in StackSAR übertragen sowie die Zuordnung der Merkmalsvektoren mittels FLANN-Algorithmus.

Die Bildkoregistrierung findet bei der Anwendung von StackSAR immer dann statt, wenn entweder die Image-Routine oder die Video-Routine ausgeführt wird. Dabei wird zunächst ein Objekt der `Producer`-Klasse erzeugt, welches die unter dem in der Konfigurationsdatei angegebenen Quellpfad zur Verfügung stehenden Radaraufnahmen nach ihrer alphanumerischen Reihenfolge einliest. Je nachdem, ob die Image- oder die Video-Routine aufgerufen worden ist, handelt es sich um ein `ImageProducer`- oder `VideoProducer`-Objekt. Aufgrund der Vererbung verfügen beide Ausführungen über die in der `Producer`-Klasse implementierten Koregistrierungsmethoden.

Anhand jeder eingelesenen Aufnahme wird ein Objekt der `Img`-Klasse (siehe Abbildung 20) initialisiert. Die `Img`-Objekte ergeben eine Bildserie, von der das erste Bild als Referenz festgelegt wird. Alle weiteren Bilder der Serie werden dann nacheinander koregistriert, indem sie an das Referenzbild angeglichen werden. Als Beispiel hierzu dienen die Abbildungen 19a) - c). Abbildung 19a) ist das Referenzbild, an welches Abbildung 19b) anzugleichen ist. Die Koregistrierung läuft so ab, dass zunächst die ORB-Merkmalspunkte der Bilder detektiert und dann durch Vektoren deskribiert werden. Der ebenfalls im

a) Referenzbild



b) Anzugleichendes Bild

Versuch in Kapitel 4 genutzten FLANN-Matcher findet dann die Zuordnungen der Merkmalspunkte von Referenzbild und anzugleichendem Bild. Aus den Zuordnungen wird im nächsten Schritt eine Transformationsmatrix berechnet und damit das anzugleichende Bild transformiert. Abbildung 19c) ist das Ergebnis der Koregistrierung von Bild 19b) zu Bild 19a).

c) Angeglichenes und markiertes Bild b)

Abbildung 19 – Beispiel einer Koregistrierung

Der schwarze Rand in Abbildung 19c) ist dadurch entstanden, dass bei der Koregistrierung zu Abbildung 19a) eine Verschiebung erfolgt ist. Diese Verschiebung ist gewünscht und für die Angleichung nötig, erzeugt jedoch aufgrund der Beibehaltung der originalen Bildgröße des anzugleichenden Bildes bei der Transformation den „leeren“ Bildrand. Aufgrund dessen ist als Teil der Image- und Video-Routine von StackSAR eine Bildzuschneidung implementiert worden, bei der die betreffenden Aufnahmen nach der Koregistrierung auf in der Konfigurationsdatei angegebene Pixelwerte zugeschnitten werden. Abbildung 19c) weist neben dem schwarzen Rand zusätzlich auch eine Markierung in der oberen linken Bildecke auf. Auch dies ist Resultat einer Funktionalität von StackSAR, die vor der Ausführung der Image- oder Video-Routine über die Konfigurationsdatei aktiviert oder deaktiviert werden kann. Bei der Markierung handelt es sich um die Startzeit der im Bild dargestellten Radaraufnahme.

5.5 Automatisierung der Herstellung von Präsentationsmaterial

Mit StackSAR können zwei Arten von Präsentationsmaterial generiert werden – Bilder, die in Aufnahmen enthaltene Veränderungen über die Zeit darstellen und Videos, die selbiges tun. In StackSAR ist jeweils eine Routine implementiert worden, die von Anwendenden ausgeführt werden kann, um das entsprechende Präsentationsmaterial zu erzeugen. Wie bereits im vorherigen Abschnitt beschrieben worden ist, findet zu Beginn der Image-Routine und der Video-Routine zunächst die Bildkoregistrierung und Zuschneidung der Aufnahmen statt. In den nachfolgenden Abschnitten ist beschrieben, wie bei der Ausführung der Image- und der Video-Routine aus den koregistrierten und zugeschnittenen Aufnahmen das entsprechende Präsentationsmaterial generiert wird.

5.5.1 Video-Routine

Rufen Anwendende über die Kommandozeile die Video-Routine von StackSAR auf, wird zunächst ein `VideoProducer`-Objekt erzeugt, welches die durch das `CLI`-Objekt eingelesenen Konfigurationsoptionen erhält. Das `VideoProducer`-Objekt initialisiert daraufhin die Bildserie, ruft dann die Koregistrierunsmethode und dann die Zuschneidungsmethode auf. Schließlich wird `generate_video()` ausgeführt, um aus den koregistrierten und zugeschnittenen Aufnahmen den Video-Output zusammenzusetzen. Zuerst erfolgt die Überprüfung, ob der Ouput-Ordner unter dem in der Konfigurationsdatei angegebenen Pfad existiert. Ist dies nicht der Fall, wird der Ordner anhand des Pfades an gewünschter Stelle erzeugt. Im nächsten Schritt besteht die Möglichkeit, die Bilder der Serie mit dem Startzeitpunkt ihrer Aufnahme zu markieren, sodass diese in weißer Aufschrift auf dem Bild erscheint. Falls dies in der Konfigurationsdatei aktiviert worden ist, werden als nächstes alle Bilder der Bildserie mit der Startzeit ihrer jeweiligen Aufnahme markiert. In Abhängigkeit davon, welches Videoformat in den Konfigurationsoptionen gefordert ist, wird die Videoproduktion nun mit `generate_avi()` oder `generate_gif()` ausgeführt. Ist in der Konfigurationsdatei ein ungültiges Dateiformat eingegeben worden, wird automatisch eine AVI-Datei als Video-Output generiert.

Die Generierung eines Videos im AVI-Format erfolgt mit der durch die OpenCV-Bibliothek zur Vefügung gestellten `VideoWriter`-Klasse. Das `VideoWriter`-Objekt wird initialisiert, indem ihm der vorbestimmte Output-Pfad übergeben wird sowie die Dauer, für die ein Bild im Video jeweils eingeblendet werden soll. Mit der Methode `VideoWriter.write()` werden schließlich die Bilder der Bildserie nacheinander dem Video hinzugefügt und dieses gespeichert.

Die Generierung eines Videos im GIF-Format erfolgt hingegen mit der durch die imageio-Bibliothek zur Verfügung gestellten Methode `get_writer()`, die ein `Writer`-Objekt zurückgibt. Auch `get_writer()` werden der Output-Pfad und die Einblendungsdauer pro Bild übergeben. Schließlich fügt das `Writer`-Objekt die Bilder der Bildserie nacheinander dem Video hinzu und speichert dieses.

5.5.2 Image-Routine

Analog zum Ablauf der Video-Routine findet auch in der Image-Routine zunächst die Initialisierung des `ImageProducer`-Objekts mit Übergabe der Konfigurationsoptionen statt. Daraufhin wird hier ebenso die Koregistrierung und Zuschneidung der zu nutzenden Bilder abgewickelt. Im Unterschied zur Video-Routine erfolgt die eventuelle Markierung der Bilder

mit der Startzeit ihrer Aufnahme an diesem Punkt noch nicht. Das liegt daran, dass die Generierung des Bild-Outputs zunächst noch Berechnungen mit den koregistrierten und zugeschnittenen Bildern erfordert und folglich erst am Ende der Image-Routine das entstandene Endprodukt zu markieren ist, falls dies per Konfigurationsdatei vorgegeben wurde.

Die Generierung des Bildprodukts erfolgt unabhängig vom in der Konfigurationsdatei angegebenen Bildformat gleich, wobei lediglich beim Speichern des Bildes ein vollständiger Pfad mit Dateiname und -endung übergeben wird. Die eingefügte Endung entspricht der vom Anwendenden eingetragenenen Konfigurationsoption, wobei nur „PNG“ und „JPG“ als Angaben gültig sind. Ist in der Konfigurationsdatei ein anderes Format angegeben, wird automatisch eine PNG-Datei generiert. In Ausführung beginnt `generate_image()` mit der Bestimmung des Referenzbildes aus der Bildserie und der Erzeugung einer Null-Matrix, die in ihrer Größe dem Referenzbild entspricht. Dann wird durch die übrigen Bilder iteriert, wobei für jedes Bild die folgenden Schritte abgearbeitet werden:

1. Berechnung der Graustufendifferenz zum vorangegangenen Bild.
2. Morphologisches Opening des Differenzbildes.
3. Gewichtetes Aufaddieren des Differenzbildes auf die Matrix.

Die Berechnung der Graustufendifferenz erfolgt für jedes Bild der Serie durch das Subtrahieren seiner Graustufenwerte von denen des Vorgängerbildes in der Serie. Diese Subtraktion ergibt ein sogenanntes Differenzbild – eine Matrix, die die Differenz der Graustufen in jedem

Abbildung 25 — Ein schwarz-weißes Bild vor dem morphologischen Öffnen (links) und nach dem Öffnen (rechts)

Bildpunkt hält. Das berechnete Differenzbild wird dann der morphologischen Operation namens Opening unterzogen. Abbildung 25 soll als Beispiel zeigen, wie die das Ausgangsbild (links) nach dem Opening als Resultat (rechts) aussieht. Beim morphologischen Opening wird ein Filter nach und nach über das gesamte Bild bewegt. Wenn der Filter eine Kante erkennt – in Abbildung 25 einen Übergang zwischen schwarz und weiß – wird je nach Breite des Filters ein Teil der Kante gelöscht (Ersosion) [52]. Anschließend erfolgt der selbe Prozess nochmal für das gesamt Bild, wobei diesmal jedoch die Kanten vergrößert werden (Dilation) [52]. Dadurch wird wie in Abbildung 25 bewirkt, dass kleine Hervorhebungen im Bild entfernt werden, denn wenn von ihrer Kante ein Teil gelöscht wird, verschwinden sie mitunter ganz, sodass in der zweiten Phase keine Kante mehr dilatiert werden kann. Für die Produktion des Bildmaterials wurde diese Operation implementiert, um Rauschen in den Differenzbildern zu reduzieren und dadurch im Endergebnis die zusammenhängenden Differenzbereiche bevorzugt darzustellen gegenüber einzelnen Differenzpunkten.

Nach dem morpholigschen Öffnen eines Differenzbildes wird dieses auf die zu Beginn initialisierte Matrix gewichtet aufaddiert, die am Ende die Gesamtdifferenz hält. Die Gewichtung leitet sich aus der Anzahl der berechneten Differenzbilder ab. Bei einer Bildserie aus n Aufnahmen werden insgesamt $n - 1$ Differenzbilder berechnet und jeweils nach dem Öffnen mit einer Gewichtung von $\frac{1}{n-1}$ auf die zu Beginn initialisierte Matrix aufaddiert. Sind die drei beschrieben Vorgehensschritte für die Bilder der Bildserie abgearbeitet worden, liegt als Ergebnis die Matrix mit der aufaddierten Gesamtdifferenz vor. Diese wird im nächsten Schritt normiert (Min-Max-Skalierung) und dann als Farbmaske auf das Referenzbild aufgetragen, sodass sich ein Endprodukt wie in Abbildung 26 ergibt. Die Matrix wird deshalb zuvor normiert, damit die resultierende Farbmaske den gesamten Wertebereich des Farbschemas bedient.

Abbildung 26 zeigt das mit der Image-Routine von StackSAR erzielte Ergebnis einer Generierung von Präsentationsmaterial in Bildform. Unter dem in der Konfigurationsdatei angegebenen Pfad befanden sich Radaraufnahmenpakete mit derselben Beam-ID (*spot_073*). Genutzt wurden entsprechend der Konfigurationsdatei jeweils die besser aufgelösten Quicklooks der Aufnahmen. Die Differenzbereiche sind in Abbildung 26 rot eingefärbt, wobei ein intensiverer Farbwert eine stärkere kumulative Veränderung indiziert. Auf eine quantitative Darstellung der Veränderungen, beispielsweise durch eine Farbskala, ist verzichtet worden, da die berechneten Differenzbilder nicht auf eine reale physikalische Größe zurückgeführt

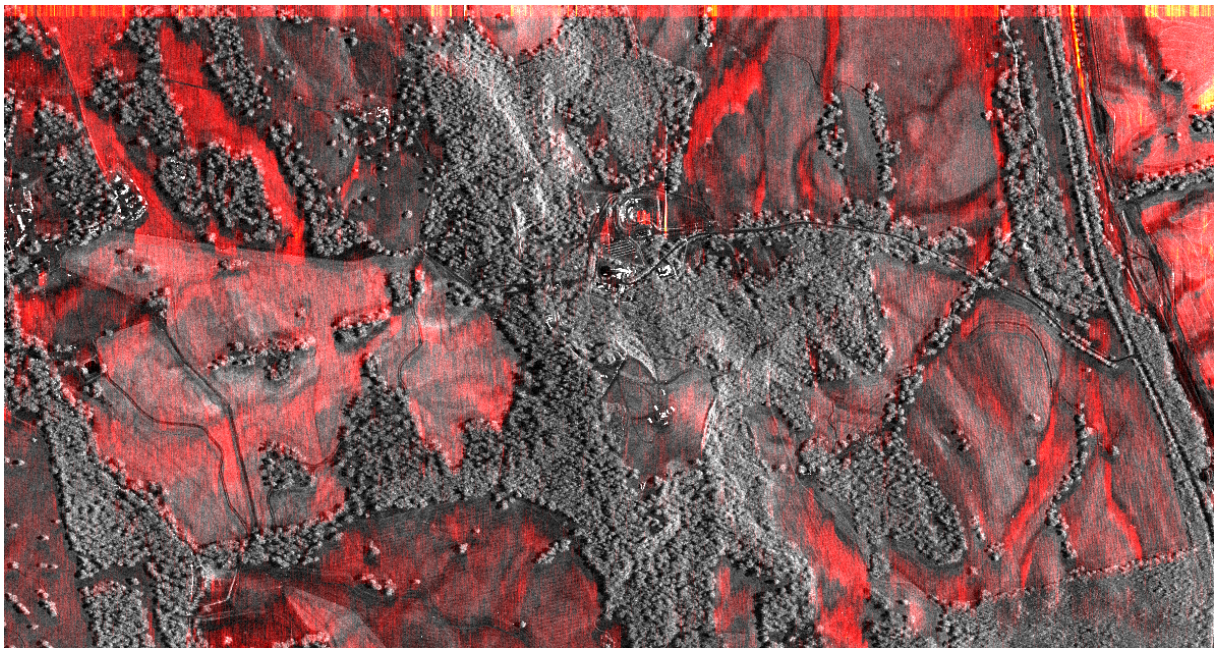


Abbildung 26 — Resultat der Bildgenerierung mit StackSAR zur Darstellung der Veränderungen in einem Stack von Aufnahmen in roter Farbe

werden können, wie dies beispielsweise in Abbildung 17 in Kapitel 3.2.3 der Fall war. Eine Interpretationsmöglichkeit des Einfärbungsergebnisses in Abbildung 26 ist die, dass insbesondere in den Wiesenflächen der dargestellten Landschaft (New Norcia) die Veränderung der Feuchtigkeit in den Radaraufnahmen sichtbar geworden ist, denn die verarbeitete Zeitserie enthielt Aufnahmen, die zu verschiedenen Jahreszeiten gemacht worden sind.

6 Tests

Der letzte Schritt zur Entwicklung von StackSAR im Rahmen der vorliegenden Arbeit war das Testen der finalen Softwareversion. Der in der Anforderungsdokumentation enthaltenen Testplanung in Kapitel 3.2.4 entsprechend wurden ein Systemtest sowie ein Usability-Test durchgeführt. Hintergrund dieser Testauswahl ist, dass mit dem Systemtest ein Test des Gesamtsystems aus Entwickelndensicht erfolgt und mit dem Usability-Test ein Test des Gesamtsystems aus Anwendendensicht. Auf diese Weise wird durch die Tests der Entwicklungsstand von StackSAR aus beiden genannten Perspektiven abgebildet. In den nachfolgenden Abschnitten sind die in den Tests eingesetzten Instrumente, die Vorgehensweisen und die Ergebnisse dargelegt.

6.1 Systemtest

Zur Untersuchung des Entwicklungsstandes von StackSAR aus Entwickelndenperspektive wurde ein Systemtest wie in den nachfolgenden Abschnitten beschrieben vorbereitet und mit den unten stehenden Zielen durchgeführt:

1. Die Funktionalität des Gesamtsystems StackSAR nach der vollständigen Integration der Systemkomponenten zu untersuchen.
2. Die Funktionalität von StackSAR unter (annähernd) realen Bedingungen zu untersuchen.
3. Den Anteil der umgesetzten funktionalen Anforderungen an StackSAR zu ermitteln.
4. Vorhandene Implementierungsfehler aufzudecken.
5. Den aktuellen Entwicklungsstand von StackSAR zum Ende der Bearbeitungsphase des Projekts aufzuzeichnen.
6. Den im Rahmen der Bachelorarbeit erreichten Implementierungserfolg für StackSAR abzuschätzen.

Der Systemtest wurde als anwendungsfallbasierter Test geplant und vorbereitet. Es wurden zunächst drei Anwendungsfälle definiert, die jeweils durch ein Nutzungsziel beschrieben werden. Die folgenden Anwendungsfälle beziehungsweise Nutzungsziele wurden für den Systemtest aufgestellt:

1. Heatmap anzeigen lassen und gespeicherte Tabelle erhalten.
2. Stack auswählen und Bilder beschaffen.
3. Image-Output generieren.
4. Video-Output generieren.

Im nächsten Schritt wurden den jeweiligen Anwendungsfällen diejenigen funktionalen Anforderungen zugeordnet, deren Erfüllung bei der Erreichung des Nutzungsziels eintreten sollte. Auf Grundlage der Zuordnung der funktionalen Anforderungen sind daraufhin die Tabellen Tabelle 8 bis Tabelle 11 angelegt worden. In den Tabellen sind zu jeder der funktionalen Anforderungen eine oder mehrere erwartete Beobachtungen festgehalten worden. Das Eintreten der erwarteten Beobachtungen für eine funktionale Anforderungen indizierte im Systemtest die Erfüllung dieser Anforderung. Für die Überprüfung, ob eine erwartete Beobachtung eingetreten war, wurden insbesondere die im Code enthaltenen Logging-Statements genutzt, sodass nach dem Durchlaufen der Anwendungsfälle aus den Einträgen in der Log-Datei entsprechende Schlussfolgerungen gezogen werden konnten. Darüber hinaus dienten die Eigenschaften der erzeugten Output-Dateien (zum Beispiel die Dateigröße) sowie Ausgaben von Datenbank-Abfragen als Indikatoren für das Eintreten oder Nicht-Eintreten der erwarteten Beobachtungen. Da im ersten Durchlauf des Systemtest zwei Überprüfungen nicht erfolgen konnten („n. ü.“) in Tabelle 10 und Tabelle 11, wurde der Systemtest zweimal durchgeführt.

6.1.1 Testprotokoll

Die Ausgangssituation für den Systemtest wurde hergestellt, indem die Installation des Python-Projekts StackSAR aus dem Gitlab Repository entsprechend der Installationsanleitung in der zugehörigen README-Datei erfolgte. Vor den Durchläufen der Anwendungsfälle erfolgten jeweils Anpassungen der in der Konfigurationsdatei enthaltenen Pfade, sodass Inputs und Outputs korrekt einlesbar beziehungsweise speicherbar waren.

1. Ziel: Heatmap anzeigen lassen und gespeicherte Tabelle erhalten
Die folgenden Vorgehensschritte wurden zum Erreichen des Ziels vorgenommen:

1. Den korrekten Pfad zu dem Ordner, in dem die lokal gespeicherte Datei mit der enthaltenen SQL-Query ist, in der Konfigurationsdatei angeben.
2. Die korrekten Pfade für die Zielordner zum Speichern der Input- und der Output-Tabelle angeben.

3. `py -13.3 main.py heatmap` ausführen.

Die folgenden Beobachtungen sind eingetreten beziehungsweise nicht eingetreten:

Anforderung	Erwartete Beobachtungen	Eingetreten	
A2	Die in der Heatmap-Anzeige wiedergegebenen Parameter entsprechen den in der Konfigurationsdatei gemachten Angaben.	ja	ja
	Die Input-Tabelle ist in dem Ordner gespeichert worden, zu dem der in der Konfigurationsdatei angegebene Pfad führt.	ja	ja
	Die Größe und die Annotationen der Stacks entsprechen den Angaben in der Konfigurationsdatei.	ja	ja
	Die Output-Tabelle ist in dem Ordner gespeichert worden, zu dem der in der Konfigurationsdatei angegebene Pfad führt.	ja	ja
A3	Die Stacks indizierenden Kreise sind annotiert mit der Cluster-ID.	ja	ja
A4	Der aufgerufene Prozess endet mit der Anzeige der nicht-leeren Heatmap.	ja	ja
A5	Auf der angezeigten Heatmap sind mit Kreisen diejenigen Stacks von Aufnahmen eines Ortes markiert, die den in der Konfigurationsdatei festgelegten Schwellenwert überschreiten.	ja	ja
A6	In der angezeigten Heatmap sowie in der gespeicherten Output-Tabelle sind Cluster verzeichnet.	ja	ja
A7	Die Output-Tabelle ist gespeichert worden.	ja	ja

Tabelle 8 — Die Protokollierungstabelle für Anwendungsfall Nr. 1 des Systemtests

2. Ziel: Stack auswählen und Bilder beschaffen

Die folgenden Vorgehensschritte wurden zum Erreichen des Ziels vorgenommen:

1. Einen in der Heatmap angezeigten Stack auswählen.

2. In der Output-Tabelle nach der Cluster-ID des Stacks filtern, um das Cluster zu erhalten.
3. Aus dem Cluster einen Stack auswählen.
4. Acquisition-Item-IDs nutzen, um aus der Langzeitdatenbank die Bilder zu beschaffen.
5. Ausgewählte Bilder lokal speichern.

Anforderung	Erwartete Beobachtungen	Eingetreten	
A3	Jede Zeile in der Output-Tabelle hat eine Acquisition-Item-ID und eine Cluster-ID.	ja	ja
	Die Acquisition-Item-IDs der Aufnahmen im ausgewählten Stack existieren in der Langzeitdatenbank.	ja	ja

Tabelle 9 — Die Protokollierungstabelle für Anwendungsfall Nr. 2 des Systemtests

3. Ziel: Image-Output generieren

Die folgenden Vorgehensschritte wurden zum Erreichen des Ziels vorgenommen:

1. Den korrekten Pfad zu dem Ordner, in dem die lokal gespeicherten Bilder sind, in der Konfigurationsdatei angeben.
2. Den korrekten Pfad zu dem Ordner, in dem der Bild-Output gespeichert werden soll, in der Konfigurationsdatei angeben.
3. `py -13.3 main.py image` ausführen.

Anforderung	Erwartete Beobachtungen	Eingetreten	
A1	Am Ende des aufgerufenen Prozesses liegt ein gespeicherter Image-Output vor.	ja	ja
A2	Die Bilder werden von dem Ordner des in der Konfigurationsdatei angegebenen Pfades eingelesen.	ja	ja
	Die Bilder besitzen nach dem Einlesen die in der Konfigurationsdatei angegebenen Optionen als Attribute.	ja	ja
	Die Koregistrierung erfolgt anhand der in der Konfigurationsdatei angegebenen Anzahl an Merkmalspunkten.	ja	ja
	Die Größe des Bild-Outputs entspricht den Angaben zum Zuschneiden in der Konfigurationsdatei.	ja	nein

Anforderung	Erwartete Beobachtungen	Eingetreten	
	Die Markierung des Bild-Outputs entspricht der Angabe in der Konfigurationsdatei.	ja	nein
	Der Image-Output ist in dem Ordner gespeichert worden, zu dem der in der Konfigurationsdatei angegebene Pfad führt.	ja	nein
A8	Die im Prozess eingelesene Bildserie gleicht der Menge an Bildern, die in dem Ordner vorliegen, zu dem der in der Konfigurationsdatei angegebene Pfad führt und deren *beamID*-Attribut der Angabe in der Konfigurationsdatei entspricht.	ja	ja
A9	Die Reihenfolge der in die Bildserie eingelesenen Bilder entspricht der alphanumerischen Reihenfolge ihrer Namen.	n. ü.	ja
A10	Die in die Bildserie eingelesenen Bilder wurden mit dem implementierten Vorgehen an das Referenzbild in der Bildserie angeglichen.	ja	ja
A14	Die funktionale Anforderung ist aus zeitlichen Gründen wissentlich nicht implementiert worden. Stattdessen erfolgt das Zuschneiden der Bilder pixelbasiert und nicht koordinatenbasiert.		
A15	Am Ende des aufgerufenen Prozesses liegt ein gespeicherter Image-Output vor.	ja	nein

Tabelle 10 — Die Protokollierungstabelle für Anwendungsfall Nr. 3 des Systemtests

4. Ziel: Video-Output generieren

Die folgenden Vorgehensschritte wurden zum Erreichen des Ziels vorgenommen:

1. Den korrekten Pfad zu dem Ordner, in dem die lokal gespeicherten Bilder sind, in der Konfigurationsdatei angeben.
2. Den korrekten Pfad zu dem Ordner, in dem der Bild-Output gespeichert werden soll, in der Konfigurationsdatei angeben.
3. `py -13.3 main.py video` ausführen.

Anforderung	Erwartete Beobachtungen	Eingetreten	
A1	Am Ende des aufgerufenen Prozesses liegt ein gespeicherter Video-Output vor.	ja	nein
A2	Die Bilder werden von dem Ordner des in der Konfigurationsdatei angegebenen Pfades eingelesen.	ja	ja
	Die Bilder besitzen nach dem Einlesen die in der Konfigurationsdatei angegebenen Optionen als Attribute.	ja	ja
	Die Koregistrierung erfolgt anhand der in der Konfigurationsdatei angegebenen Anzahl an Merkmalspunkten.	ja	ja
	Die Größe der im Video-Output verwendeten Bilder entspricht den Angaben zum Zuschneiden in der Konfigurationsdatei.	ja	nein
	Die Markierung der im Video-Output verwendeten Bilder entspricht der Angabe in der Konfigurationsdatei.	ja	nein
	Der Video-Output ist in dem Ordner gespeichert worden, zu dem der in der Konfigurationsdatei angegebene Pfad führt.	ja	nein
A8	Die im Prozess eingelesene Bildserie gleicht der Menge an Bildern, die in dem Ordner vorliegen, zu dem der in der Konfigurationsdatei angegebene Pfad führt und deren *beamID*-Attribut der Angabe in der Konfigurationsdatei entspricht.	ja	ja
A9	Die Reihenfolge der in die Bildserie eingelesenen Bilder entspricht der alphanumerischen Reihenfolge ihrer Namen.	n. ü.	ja
A10	Die in die Bildserie eingelesenen Bilder wurden mit dem implementierten Vorgehen an das Referenzbild in der Bildserie angeglichen.	ja	ja
A11	Der gespeicherte Video-Output beinhaltet alle Bilder der eingelesenen Zeitserie, welche darin nacheinander eingeblendet werden.	ja	nein

Anforderung	Erwartete Beobachtungen	Eingetreten	
A12	Die Einblendungsdauer der im Video-Output enthaltenen Bilder entspricht den Angaben in der Konfigurationsdatei.	ja	nein
A13	Die funktionale Anforderung ist aus zeitlichen Gründen wissentlich nicht implementiert worden.		
A14	Die funktionale Anforderung ist aus zeitlichen Gründen wissentlich nicht implementiert worden. Stattdessen erfolgt das Zuschneiden der Bilder pixelbasiert und nicht koordinatenbasiert.		

Tabelle 11 — Die Protokollierungstabelle für Anwendungsfall Nr. 4 des Systemtests

In der zweiten Durchführung der Vorgehensschritte zu Ziel Nummer Drei und Ziel Nummer 4 traten mehrere erwartete Beobachtungen nicht ein. (Siehe dazu die jeweils rechten Spalten in Tabelle 10 und Tabelle 11.) Es wurde beobachtet, dass die Anwendung abbrach und eine Fehlermeldung ausgab, die eine erfolglose Koregistrierung indizierte. Bei der Betrachtung der im vorherigen Schritt ausgewählten Quicklooks fiel auf, dass diese sehr dunkel und sehr kontrastarm waren, verglichen mit den bisher in der Entwicklung von StackSAR genutzten Quicklooks. Da ein weiterer Versuch, Ziele Nummer Drei und Vier zu erfüllen, mit den im ersten Testdurchlauf genutzten Quicklooks schließlich gelang, wird davon ausgegangen, dass das unerwartete Verhalten durch die dunklen, kontrastarmen SAR-Aufnahmen ausgelöst worden war.

6.2 Usability-Test

Das Gesamtsystem StackSAR wurde in einem Usability-Test auf seine Nutzungstauglichkeit hin untersucht. Der Usability-Test wurde durchgeführt mit den folgenden Zielen:

1. Den Schwierigkeitsgrad einzuschätzen, den Anwendende bei der Erfüllung ihres Anwendungsziels mit StackSAR erleben.
2. Die Tauglichkeit von StackSAR für seinen intendierten Anwendungszweck zu messen.
3. Den wirtschaftlichen Vorteil von StackSAR einschätzen.
   (Annahme: Nur wenn StackSAR nutzungstauglich ist, kann damit gerechnet werden, dass durch die implementierten Automatisierungen Arbeitszeit von Mitarbeitenden eingespart werden wird.)

Der Usability-Test gliederte sich in zwei Phasen, die mit dem Stakeholder des Entwicklungsprojekts als testweisen Anwender (Proband) durchgeführt wurden. In der ersten Phase erhielt der Proband eine Liste mit Zielen, die er durch die Nutzung von StackSAR erfüllen sollte. In der zweiten Phase erhielt der Proband einen Fragebogen zum Ausfüllen, der die Einschätzung der Nutzungstauglichkeit von StackSAR aus Sicht des Probanden ermittelte. Als Fragebogen wurde die *System Usability Scale* (SUS) ausgewählt. Dabei handelt es sich um ein Testinstrument, welches 1986 in Orientierung an der Definition von *Usability* des derzeitigen ISO-9241-11-Standards entwickelt worden ist [53]. Die Skala setzt sich zusammen aus zehn Aussagen, zu denen ein Zustimmungswert von 1 (*Strongly disagree*) bis 5 (*Strongly agree*) zu geben ist [53]. Die Testaussagen betreffen verschiedene Aspekte der Gebrauchstauglichkeit des Systems, wie zum Beispiel das Bedürfnis nach Unterstützung bei dessen Anwendung oder die Komplexität der Anwendung [53]. In Schmidt et al. [54] ist die Eignung von SUS als geeignetes Instrument zur schnellen, kostengünstigen und einfachen Evaluierung der Gebrauchstauglichkeit digitaler Assistenzsysteme bestätigt worden.

Das konkrete Vorgehen bei der Durchführung des Usability-Tests war wie folgt: Zunächst wurde sichergestellt, dass der PC des Probanden über die nötigen Voraussetzungen verfügt, um StackSAR installieren zu können. Anschließend erfolgte die Installation von StackSAR auf dem Rechner des Probanden, wobei die Versuchsleitung zur Hilfestellung anwesend war. Daraufhin erhielt der Proband alle Instruktionen über den Versuchsablauf und einen Ausdruck der von ihm zu erfüllenden Anwendungsziele (siehe Anhang). Der Ausdruck enthielt ausdrücklich keine Nutzungsanleitung oder Hinweise zum Umgang mit StackSAR, denn auch die in StackSAR eingebauten Hilfestellungen (siehe Kapitel 5.2) waren Teil der zu testenden Gebrauchstauglichkeit. Nach der Übergabe verließ die Versuchsleitung den Raum. Der Proband gab ein Signal, wenn er die Anwendungsziele abgearbeitet hatte. Erst dann übergab ihm die Versuchsleitung den SUS-Fragebogen auf Papier gedruckt und verließ erneut den Raum, bis der Proband den Fragebogen vollständig ausgefüllt hatte. So wie es in der Dokumentation der SUS [53] vorgegeben ist, wurde der Proband vor dem Ausfüllen des Fragebogens instruiert, die Items möglichst unmittelbar zu beantworten, ohne intensiv dabei nachzudenken. Ebenso wurde der Proband gebeten, in dem Falle, dass er keine Anwort auf ein Item geben könne, die mittlere Option (3) anzukreuzen.

6.2.1 Testprotokoll

Item	(1) Strongly disagree	(2)	(3)	(4)	(5) Strongly agree
1. I think that I would like to use this system frequently				X	
2. I found the system unnecessarily complex	X				
3. I thought the system was easy to use				X	
4. I think that I would need the support of a technical person to be able to use this system		X			
5. I found the various functions in this system were well integrated					X
6. I thought there was too much inconsistency in this system		X			
7. I would imagine that most people would learn to use this system very quickly				X	
8. I found the system very cumbersome to use	X				
9. I felt very confident using the system					X
10. I needed to learn a lot of things before I could get going with this system		X			

Tabelle 12 — Wiedergabe des im Usability-Test durch den Probanden ausgefüllten SUS-Fragebogens

6.2.2 Auswertung

Die Auswertung des durch den Probanden ausgefüllten SUS-Fragebogens erfolgte gemäß den Vorgaben in der SUS-Dokumentation [53]:

Die Auswertung des SUS ergibt immer eine Punktzahl zwischen 0 und 100. Zunächst wird aus den angekreuzten Antworten der Items eine Summe (siehe Gleichung 2) berechnet, wobei jedes Item 0 bis 4 Punkte zu der Summe beträgt.

$$\begin{aligned} \text{sum} = &(\#(\text{item1}) - 1) + (5 - \#(\text{item2})) + \\ &(\#(\text{item3}) - 1) + (5 - \#(\text{item4})) + \\ &(\#(\text{item5}) - 1) + (5 - \#(\text{item6})) + \\ &(\#(\text{item7}) - 1) + (5 - \#(\text{item8})) + \\ &(\#(\text{item9}) - 1) + (5 - \#(\text{item10})) \end{aligned} \tag{2}$$

Die eine Hälfte der Items ist positiv kodiert (zum Beispiel „3. I thought the system was easy to use“) und die andere Hälfte negativ kodiert (zum Beispiel „8. I found the system very cumbersome to use“). Die Antworten der positiv kodierten Items (1, 3, 5, 7 und 9) fließen in die Summe ein, indem von ihnen 1 abgezogen wird und die Antworten der negativ kodierten Items fließen in die Summe ein, indem sie von 5 abgezogen werden.

$$\text{result} = \text{sum} \times 2.5 \tag{3}$$

Schließlich wird die entstandene Summe mit 2.5 multipliziert (siehe Gleichung 3), sodass als Ergebnis eine Punktzahl zwischen 0 und 100 entsteht.

Die Antworten des Stakeholders auf die Items der SUS ergeben nach vorgegebener Berechnungsweise die folgende Punktzahl:

$$\begin{aligned} &((4 - 1) + (5 - 1) + \\ &(4 - 1) + (5 - 2) + \\ &(5 - 1) + (5 - 2) + \\ &(4 - 1) + (5 - 1) + \\ &(5 - 1) + (5 - 2)) \times 2.5 = \underline{\underline{85}} \end{aligned} \tag{4}$$

7 Diskussion

Im Rahmen der vorliegenden Bachelorarbeit wurde eine Anwendung namens StackSAR zur Unterstützung der Erstellung von Präsentationsmaterial aus TerraSAR-X/TanDEM-X-Aufnahmen entwickelt. Das Anwendungskonzept ist in Form einer umfassenden Anforderungsdokumentation erarbeitet worden und die Wahl des implementierten Koregistrierungsalgorithmus wurde durch eine eigens durchgeführte experimentelle Untersuchung fundiert begründet. Am Ende der Arbeit wurde der Entwicklungsstand von StackSAR in einem Systemtest und in einem Usability-Test ermittelt.

Zu Beginn des Entwicklungsprozesses wurde eine vollständige Anforderungsdokumentation auf Grundlage der Vorstellungen des Stakeholders erarbeitet. Bei der Erarbeitung war berücksichtigt worden, dass die Arbeit an der Entwicklung von StackSAR über das vorliegende Projekt hinaus weiterlaufen wird. Die Anforderungsdokumentation beinhaltet deshalb auch Ideen, die in diesem Projekt nicht umzusetzten waren, jedoch für die Vollständigkeit der Beschreibung von StackSAR nötig waren. Daraus resultierte im Entwicklungsprozess mitweilen eine Uneindeutigkeit darüber, welche der Anforderungen verbindlich im vorliegenden Projekt zu implementieren sind. In Retrospektive ist daher anzumerken, dass eine Priorisierung der erarbeiteten Anforderungen, beispielsweise durch die Nutzung unterschiedlicher Schlüsselbegriffe [55], anhand des Kano-Modells [56] oder nach der MoSCoW-Priorisierung [57] für den Entwicklungsprozess förderlich gewesen wäre. Als die Umsetzung schon fortgeschritten war, kamen noch weitere Anforderungen hinzu, sodass die verfügbaren zeitlichen Ressourcen überschritten wurden. Hier hätte eine Priorisierung der Anforderungen entgegenwirken und eine gezieltere Umsetzung der wichtigsten Funktionalitäten fördern können.

Das Ergebnis der Algorithmenstudie war, dass die Merkmalsdetektion und -deskription mittels ORB-Algorithmus für die eingesetzten SAR-Aufnahmen mehr gelungene Zuordnungen von Merkmalsvektoren ergibt als das Vorgehen mittels KAZE-Algorithmus. Eine mögliche Ursache dafür ist die Robustheit von ORB im Umgang mit Bildpaaren unterschiedlicher Perspektive, Skalierung und Rotation, verglichen mit anderen Methoden zur Detektion von Merkmalspunkten [22]. Den in den Koregistrierungsversuchen erfolgten Messungen lag die Annahme zugrunde, dass die Anzahl der gelungenen Zuordnungen von Merkmalsvektoren

ein gültiger Indikator für die Effektivität einer Bildkoregistrierung ist. Dieser vermutete Zusammenhang zwischen Messgröße und Qualität des Verfahrens ist jedoch nicht wissenschaftlich abgesichert worden. Stattdessen wurde angenommen, dass zwei Aufnahmen, zu denen eine größere Anzahl an Merkmalsvektoren mit gelungener Zuordnung vorliegt, besser angeglichen werden können als Aufnahmen mit einer geringeren Anzahl solcher Zuordnungen. Da die aus den angeglichenen Bildern generierten Videos keine mit dem Auge sichtbaren Qualitätsunterschiede aufweisen, liegt die Vermutung nahe, dass zwischen der Anzahl der gelungenen Zuordnungen und der Qualität der Koregistrierung kein oder zumindest kein linearer Zusammenhang besteht. Stattdessen kann vermutbar ab einer bestimmten Anzahl gelungener Zuordnungen von Merkmalsvektoren von einer erfolgreichen Bildkoregistrierung ausgegangen werden, ohne dass zusätzliche Merkmalspunkte das Ergebnis noch maßgeblich verbessern könnten. Ein weiteres Qualitätsmaß, welches zum Vergleich der beiden Koregistrierungsalgorithmen hätte untersucht werden können, ist ihre Laufzeit. Letztendlich hat jedoch der Zeitaufwand bei der Koregistrierung mit StackSAR entsprechend der Qualitätsmerkmale (siehe Kapitel 3.2.3) keine Priorität, und ist daher bei der Algorithmenauswahl vernachlässigbar.

Die in Kapitel 1.2 aufgestellten Implementierungsziele konnten im Rahmen der vorliegenden Arbeit erreicht werden. Der durch die Algorithmenstudie legitimierte ORB-Algorithmus erwies sich auch in der Implementierung als zuverlässig, scheiterte jedoch an Radaraufnahmen, die insgesamt dunkel und von geringem Kontrast waren. Als besonders vorteilhaft erwies sich die testgetriebene Entwicklung (TDD). Die daraus resultierte Entkopplung der Funktionalitäten ergab testbaren Code, von dem für zukünftige Weiterentwicklungsarbeiten Wartbarkeit und Modifizierbarkeit erwartet wird. Wegen Mangels an Dokumentation und zeitlichen Ressourcen konnte im Rahmen dieser Arbeit keine Übersetzung und Integrierung des existieren IDL-Codes zur Geokodierung von SAR-Aufnahmen erfolgen, was die aktuellen Anwendungsmöglichkeiten des Tools denkbar einschränkt. Im nachfolgenden Abschnitt wird diese Vermutung durch ein Beispiel bestätigt.

Das Erreichen der Implementierungsziele wurde festgestellt durch die abschließend durchgeführten Tests. Der Systemtest und der Usability-Test stellen eine Erweiterung der im TDD-Vorgehen durchgeführten Tests auf das Gesamtsystem StackSAR dar. Der Systemtest ergab ein überwiegend postives Bild des Entwicklungsstands der Anwendungssoftware unter (annähernd) realen Bedingungen. Gleichzeitig zeigte er die Unzuverlässigkeit des ORB-Algorithmus

bei der Koregistrierung dunkler, kontrastarmer Radaraufnahmen auf, sodass die Schwachstelle künftig behoben werden kann. Es ist zu bemerken, dass der Systemtest nicht alle möglichen Anwendungsszenarien und Konfigurationsmöglichkeiten von StackSAR abdeckt und demzufolge davon auszugehen ist, dass mehr Testdurchläufe mit unterschiedlichen Bedingungen weitere Schwächen in der Implementierung aufgezeigt hätten. Beispielsweise lag der Fokus beim Systemtest vollständig auf der Überprüfung der funktionalen Anforderungen und die Qualitätsmerkmale von StackSAR wurden außen vor gelassen. Die Ursache hinter dieser Entscheidung liegt darin, dass die Qualitätsmerkmale wenig quantifizierbar formuliert worden sind, wodurch ihre Überprüfung stark erschwert wäre.

Der Usability-Test ergab ein überwiegend positives Feedback über die Gebrauchstauglichkeit von StackSAR. Anzumerken ist jedoch, dass es in der Testdurchführung mehrmals zu Fehlermeldungen kam, die die Hilfestellung der Versuchsleitung erforderten. Die Fehlermeldungen waren kein Resultat der Bedienung des Tools durch den Probanden, sondern zeigten weitere bestehende Schwachstellen im aktuellen Entwicklungsstand der Software auf. Für die Weiterentwicklung hat sich dadurch ein Mehrwert ergeben, jedoch muss davon ausgegangen werden, dass das Antwortverhalten des Probanden durch die unerwünschten Unterbrechungen des Tests beeinflusst wurde [53]. Auch kann nicht ausgeschlossen werden, dass die Rolle des Probanden als Stakeholder im Entwicklungsprozess von StackSAR sich möglicherweise auf seine Einschätzung der Bedienbarkeit ausgewirkt hat. Abschließend gibt die Testdokumentation der SUS keine Interpretationsvorgabe der errechneten Punktzahl her, sodass diese für sich steht.

Der wirtschaftliche Vorteil, der durch die Nutzung von StackSAR im Institut zu erwarten ist, kann im aktuellen Entwicklungsstand und anhand der durchgeführten Tests nur ansatzweise abgeschätzt werden. Das bereits zum jetzigen Entwicklungsstand im Usability-Test erhaltene, positive Stakeholderfeedback kann jedoch als ein Indikator für eine rege zukünftige Nutzung der Anwendung herangezogen werden. Eine genaue Abschätzung der Zeiteinsparungen, die bei der Nutzung von StackSAR zur Produktion von Präsentationsmaterial zu erwarten ist, kann erst in einem späteren Entwicklungsstand ermittelt werden. Insgesamt zeugen die Ergebnisse von Systemtest und Usability-Test bereits jetzt davon, dass StackSAR seine funktionalen Anforderungen sowie die Erwartungen an seine Bedienbarkeit überwiegend erfüllt. Daraus wird geschlossen, dass der aktuelle Entwicklungsstand die gelungene Umsetzung eines

funktionierenden Konzepts zur Erreichung der aktuellen und hinzukommenden Implementierungsziele darstellt.

8 Ausblick

StackSAR stellt eine erfolgreiche erste Umsetzung der zur Erstellung von Präsentationsmaterial geforderten Anwendung dar. Dennoch bestehen offene Implementierungsdetails und Visionen, die es in Zukunft zu ergänzen gilt. Das sind in erster Linie die bislang unerfüllten funktionalen Anforderungen – die Konfigurierbarkeit der Bildübergänge in den Videos sowie die koordinatenbasierte Bildzuschneidung. Von besonderer Priorität sollte auch die Implementierung der EOWeb-Interface-Automation sein, die der Originalentwurf von StackSAR (siehe Abbildung 14) für die Beschaffung der Radaraufnahmen vorsieht. Aktuell müssen Nutzende ihre ausgewählten Radaraufnahmen manuell aus der Langzeitdatenbank abrufen und auf ihrem eigenen System lokal zwischenspeichern. Diesen Aufwand zu reduzieren, sollte ein zentrales Ziel bei der anstehenden Weiterentwicklung von StackSAR sein. Sollte die Implementierung der EOWeb-Interface-Automation sich aufgrund der bestehenden rechtlichen Einschränkungen weiterhin verzögern, könnte übergangsweise ein Workaround-Tool integriert werden, welches die durch Nutzende ausgewählten Radaraufnahmen automatisch anhand eines Identifiers aus dem Netzwerk auf das ausführende System kopiert.

Um StackSAR von einem „Entwicklungsmodus“ in eine *Ready-to-Use*-Anwendung zu überführen, sind zukünftig Verbesserungen seiner Bedienbarkeit zu erzielen. Zunächst sollte die Steuerbarkeit über die Kommandozeile weiter ausgebaut werden, sodass die Kommandos intuitiver und kürzer sind und nicht den technischen Unterbau des Tools wiederspiegeln. Zusätzlich sollten weitere Kommandos implementiert werden, die die Notwendigkeit einer häufigen Aktualisierung der Konfigurationsdatei durch Anwendende reduzieren. Letztendlich ist auch die Implementierung eines umfassenden Error-Handlings von ernormer Bedeutung, wenn die Bedienbarkeit von StackSAR verbessert werden soll. Um in der Zukunft einen gezielten Ausbau der Bedienbarkeit einzuleiten, wäre zunächst ein weiterer Systemtest angebracht, der den Fokus auf die Qualitätsmerkmale von StackSAR legt, um dahingehende Schwachstellen zu identifizieren.

Es wird erwartet, dass der bestehende Quellcode von StackSAR inklusive seiner Test-Module eine zufriedenstellende Grundlage darstellt, auf der zukünftig weitere Funktionalitäten aufgebaut werden können. Für den Zweck der Erstellung von Präsentationsmaterial ist eine

Vielzahl von Funktionalitäten denkbar, von deren Implementierung das Zielpersonal profitieren könnte. Eine solche Funktionalität wäre beispielsweise ein in die Heatmap-Anzeige integrierter View der zeitlichen Verteilung von SAR-Aufnahmen einer bestimmten Szene. Das automatische Kopieren von Koordinaten in die Zwischenablage bei einem Klick auf die Heatmap zählt ebenso dazu wie auch das automatisierte Zuschneiden der schwarzen Ränder, die bei der Koregistrierung den Aufnahmen hinzugefügt werden. Auch eine Verknüpfung mit der im Institut genutzten Software Google Earth ist vorstellbar, in der Form, dass bei einem Klick auf die Heatmap die entsprechenden Koordinaten bereits in einen Link zu Google Earth eingebettet werden und dieser in die Zwischenablage kopiert wird. Dadurch könnten Nutzende im Browser direkt zu der den Koordinaten entsprechenden Position in Google Earth gelangen.

In erster Linie ist jedoch für die Weiterentwicklung von StackSAR die Integration der Geokodierung unerlässlich, insbesondere um genau diejenigen dunklen und/oder kontrastarmen Aufnahmen, an denen die ORB-Koregistrierung zu versagen scheint, anhand ihrer Aufnahmekoordinaten zuverlässig aneinander angleichen zu können. Zusätzlich würde es die Geokodierung ermöglichen, Radaraufnahmen mit sehr unterschiedlichen Dimensionen zu koregistrieren und letztendlich auch das angeforderte koordinatenbasierte Zuschneiden in den Koregistrierungsvorgang zu integrieren. Im Ergebnis kämen dadurch neue Kombinationsmöglichkeiten von Radaraufnahmen für die Erstellung von Präsentationsmaterial infrage, auch dann, wenn die Verfügbarkeit ähnlicher Aufnahmen einer bestimmten Szene geringer ausfällt.

Daran anknüpfend sind für die Zukunft auch neue Formen von Präsentationsmaterial als die in der vorliegenden Arbeit betrachteten vorstellbar. Anstatt lediglich die Veränderungen über die Zeit an einem festen Ort darzustellen, bieten sich noch weitere Möglichkeiten für die Kombination der in den Radaraufnahmen enthaltenen Informationen an. Beispielsweise könnten Videos erstellt werden, die zu einem festen Zeitpunkt eine kontinuierliche Abfolge von koregistrierten, benachbarten Szenen darstellen, sodass ein Überflug der Erdoberfläche simuliert wird. Zusammenfassend kann für die Erstellung von Präsentationsmaterial aus Radaraufnahmen durch die bisherigen und zukünftigen Funktionalitäten von StackSAR eine unterstützende Wirkung erwartet werden.

Literatur

[1] „Synthetic Aperture Radar (SAR) | NASA Earthdata“. Zugegriffen: 3. Juni 2025. [Online]. Verfügbar unter: https://www.earthdata.nasa.gov/learn/earth-observation-data-basics/sar

[2] „ARSET - An Introduction to Synthetic Aperture Radar (SAR) and Its Applications | NASA Applied Sciences“. Zugegriffen: 4. Juni 2025. [Online]. Verfügbar unter: https://appliedsciences.nasa.gov/get-involved/training/english/arset-introduction-synthetic-aperture-radar-sar-and-its-applications

[3] M. Eineder, W. Abdel Jaber, D. Floricioiu, H. Rott, und N. Yague-Martinez, „Glacier Flow and Topography Measurements with TerraSar-X and TanDEM-X“, in *2011 IEEE International Geoscience and Remote Sensing Symposium*, Juli 2011, S. 3835–3838. doi: 10.1109/IGARSS.2011.6050067.

[4] Airbus Defence and Space, „TerraSAR-X Image Product Guide“. Zugegriffen: 8. August 2025. [Online]. Verfügbar unter: https://airbusus.com/wp-content/uploads/2020/06/r459_9_20171004_tsxx-airbusds-ma-0009_tsx-productguide_i2.01.pdf

[5] V. Ablavsky, „2-View Alignment and RANSAC“. Zugegriffen: 21. August 2025. [Online]. Verfügbar unter: https://courses.cs.washington.edu/courses/csep576/21au/lectures/03_2of2_RANSAC.pdf

[6] G. Saur und W. Krüger, „Fine-Geocoding of SAR Using Robust Map-to-Image Registration“, [Online]. Verfügbar unter: https://www.researchgate.net/publication/228895003_Fine-geocoding_of_SAR_using_robust_map-to-image_registration

[7] W. Schramm, „Anforderungsanalyse Und -Spezifikation“. Zugegriffen: 28. Juni 2025. [Online]. Verfügbar unter: https://services.informatik.hs-mannheim.de/~schramm/see/files/Kapitel03.pdf

[8] K. Pohl und C. Rupp, *Requirements Engineering Fundamentals: A Study Guide for the Certified Professional for Requirements Engineering Exam, Foundation Level, IREB Compliant*, Second edition. Santa Barbara, CA: Rocky Nook, 2015. [Online]. Verfügbar unter: https://dl.acm.org/doi/book/10.5555/2018734

[9] K. Pohl und C. Rupp, *Basiswissen Requirements Engineering: Aus- und Weiterbildung nach IREB-Standard zum Certified Professional for Requirements Engineering Foundation Level*, 5., überarbeitete und aktualisierte Auflage. Heidelberg: dpunkt.verlag, 2021. [Online]. Verfügbar unter: https://www.semanticscholar.org/paper/Basiswissen-Requirements-Engineering-Aus-und-zum-Pohl-Rupp/1e5d393ed241fef7c446619e67c55ca91d80cbf4

[10] M. Grande, *100 Minuten Für Anforderungsmanagement: Kompaktes Wissen Nicht Nur Für Projektleiter Und Entwickler*. Wiesbaden, GERMANY: Springer Fachmedien Wiesbaden GmbH, 2014. Zugegriffen: 1. Juli 2025. [Online]. Verfügbar unter: http://ebookcentral.proquest.com/lib/dlr-ebooks/detail.action?docID=1802936

[11] „TerraSAR-X“. Zugegriffen: 4. Juni 2025. [Online]. Verfügbar unter: https://www.dlr.de/de/forschung-und-transfer/projekte-und-missionen/terrasar-x

[12] „TanDEM-X“. Zugegriffen: 22. Mai 2025. [Online]. Verfügbar unter: https://www.dlr.de/de/forschung-und-transfer/projekte-und-missionen/tandem-x

[13] „TanDEM-X Mission“. Zugegriffen: 20. Mai 2025. [Online]. Verfügbar unter: https://www.dlr.de/de/hr/forschung-transfer/projekte/tandem-x

[14] A. Tsokas, M. Rysz, P. M. Pardalos, und K. Dipple, „SAR Data Applications in Earth Observation: An Overview“, *Expert Systems with Applications*, Bd. 205, S. 117342, Nov. 2022, doi: 10.1016/j.eswa.2022.117342.

[15] A. van Zyl, „Geschlechtersensible Sprache - Ein Leitfaden“. Zugegriffen: 17. November 2024. [Online]. Verfügbar unter: https://www.dhbw.de/fileadmin/user_upload/Dokumente/Gleichstellung_Chancengleichheit/LeitfadenGeschlechtersensibleSprache.pdf

[16] „Overview | Get to Know SAR“. Zugegriffen: 11. Juni 2025. [Online]. Verfügbar unter: https://nisar.jpl.nasa.gov/mission/get-to-know-sar/overview

[17] A. Flores, K. Herndon, R. Thapa, und E. Cherrington, „Synthetic Aperture Radar (SAR) Handbook: Comprehensive Methodologies for Forest Monitoring and Biomass Estimation“, 2019, doi: 10.25966/NR2C-S697.

[18] Dr. Niklas Reinke und Rolf Wernighaus, „TerraSAR-X - Das Deutsche Radar-Auge Im All“, Juli 2009. Zugegriffen: 9. Januar 2023. [Online]. Verfügbar unter: https://www.dlr.de/de/medien/publikationen/broschueren/terrasar-x-das-deutsche-radar-auge-im-all/@@download/file

[19] „Satelliten und ihre Bahnen um die Erde“. Zugegriffen: 18. Juli 2025. [Online]. Verfügbar unter: https://www.dlr.de/de/next/schule-und-ausbildung/lernmodule/die-erde-von-oben/satelliten-und-ihre-bahnen-um-die-erde

[20] B. Fischer und J. Modersitzki, „Ill-Posed Medicine—an Introduction to Image Registration“, *Inverse Problems*, Bd. 24, Nr. 3, S. 34008, Mai 2008, doi: 10.1088/0266-5611/24/3/034008.

[21] S. Saleem, A. Bais, und R. Sablatnig, „Towards Feature Points Based Image Matching between Satellite Imagery and Aerial Photographs of Agriculture Land“, *Computers and Electronics in Agriculture*, Bd. 126, S. 12–20, Aug. 2016, doi: 10.1016/j.compag.2016.05.005.

[22] C. Liu, J. Xu, und F. Wang, „A Review of Keypoints' Detection and Feature Description in Image Registration“, *Scientific Programming*, Bd. 2021, Nr. 1, S. 8509164, 2021, doi: 10.1155/2021/8509164.

[23] S. Baker, S. K. Nayar, und H. Murase, „Parametric Feature Detection“, *International Journal of Computer Vision*, Bd. 27, Nr. 1, S. 27–50, März 1998, doi: 10.1023/A:1007901712605.

[24] E. Sansosti, P. Berardino, M. Manunta, F. Serafino, und G. Fornaro, „Geometrical SAR Image Registration“, *IEEE Transactions on Geoscience and Remote Sensing*, Bd. 44, Nr. 10, S. 2861–2870, Okt. 2006, doi: 10.1109/TGRS.2006.875787.

[25] P. A. Zandbergen, „Geocoding Quality and Implications for Spatial Analysis“, *Geography Compass*, Bd. 3, Nr. 2, S. 647–680, 2009, doi: 10.1111/j.1749-8198.2008.00205.x.

[26] „IEEE Standard Glossary of Software Engineering Terminology“, *IEEE Std 610.12-1990*, S. 1–84, Dez. 1990, doi: 10.1109/IEEESTD.1990.101064.

[27] M. Broy und M. Kuhrmann, „Vorgehen in der Anforderungserhebung“, *Einführung in die Softwaretechnik*. Springer Vieweg, Berlin, Heidelberg, 2021. doi: 10.1007/978-3-662-50263-1_7.

[28] „Appendix C: How to Write a Good Requirement - NASA“. Zugegriffen: 16. Juli 2025. [Online]. Verfügbar unter: https://www.nasa.gov/reference/appendix-c-how-to-write-a-good-requirement/

[29] „ISO/IEC/IEEE 29148:2018“. Zugegriffen: 9. Juli 2025. [Online]. Verfügbar unter: https://www.iso.org/standard/72089.html

[30] „ISO/IEC/IEEE International Standard - Systems and Software Engineering – Life Cycle Processes –Requirements Engineering“, *ISO/IEC/IEEE 29148:2011(E)*, S. 1–94, Dez. 2011, doi: 10.1109/IEEESTD.2011.6146379.

[31] D. Sale, *Testing Python: Applying Unit Testing, TDD, BDD and Acceptance Testing*. Newark, UNITED KINGDOM: John Wiley & Sons, Incorporated, 2014. Zugegriffen: 4. Juli 2025. [Online]. Verfügbar unter: http://ebookcentral.proquest.com/lib/dlr-ebooks/detail.action?docID=1729555

[32] K. Beck, *Test-Driven Development : By Example*. Boston : Addison-Wesley, 2002. Zugegriffen: 9. Juli 2025. [Online]. Verfügbar unter: http://archive.org/details/est-driven-development-by-example

[33] L. Copeland, *A Practitioner's Guide to Software Test Design*. Norwood, MA, UNITED STATES: Artech House, 2003. Zugegriffen: 4. Juli 2025. [Online]. Verfügbar unter: http://ebookcentral.proquest.com/lib/dlr-ebooks/detail.action?docID=227688

[34] P. Liggesmeyer, *Software-Qualität: Testen, Analysieren und Verifizieren von Software*. Springer Science & Business Media, 2009. [Online]. Verfügbar unter: https://publica.fraunhofer.de/entities/publication/d0e06424-e786-4624-aa11-310459e95e5b

[35] R. C. Martin, „Professionalism and Test-Driven Development“, *IEEE Software*, Bd. 24, Nr. 3, S. 32–36, Mai 2007, doi: 10.1109/MS.2007.85.

[36] „Missions Engineering“. Zugegriffen: 1. Juli 2025. [Online]. Verfügbar unter: https://www.dlr.de/de/hr/ueber-uns/abteilungen/satelliten-sar-systeme/missions-engineering

[37] „SatDSiG - Nichtamtliches Inhaltsverzeichnis“. Zugegriffen: 14. Juli 2025. [Online]. Verfügbar unter: https://www.gesetze-im-internet.de/satdsig/

[38] „BAFA - Satellitendatensicherheit“. Zugegriffen: 14. Juli 2025. [Online]. Verfügbar unter: https://www.bafa.de/DE/Aussenwirtschaft/Satellitendatensicherheit/satellitendatensicherheit_node.html

[39] E. Rosten, R. Porter, und T. Drummond, „Faster and Better: A Machine Learning Approach to Corner Detection“, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Bd. 32, Nr. 1, S. 105–119, Jan. 2010, doi: 10.1109/TPAMI.2008.275.

[40] D. Hutchison *u. a.*, „BRIEF: Binary Robust Independent Elementary Features“, *Computer Vision – ECCV 2010*, Bd. 6314. Springer Berlin Heidelberg, Berlin, Heidelberg, S. 778–792, 2010. doi: 10.1007/978-3-642-15561-1_56.

[41] E. Rublee, V. Rabaud, K. Konolige, und G. Bradski, „ORB: An Efficient Alternative to SIFT or SURF“, in *2011 International Conference on Computer Vision*, Nov. 2011, S. 2564–2571. doi: 10.1109/ICCV.2011.6126544.

[42] D. Lowe, „Object Recognition from Local Scale-Invariant Features“, in *Proceedings of the Seventh IEEE International Conference on Computer Vision*, Sep. 1999, S. 1150–1157. doi: 10.1109/ICCV.1999.790410.

[43] P. F. Alcantarilla, A. Bartoli, und A. J. Davison, „KAZE Features“, in *Computer Vision – ECCV 2012*, A. Fitzgibbon, S. Lazebnik, P. Perona, Y. Sato, und C. Schmid, Hrsg., Berlin, Heidelberg: Springer, 2012, S. 214–227. doi: 10.1007/978-3-642-33783-3_16.

[44] M. Muja und D. Lowe, „FLANN - Fast Library for Approximate Nearest Neighbors User Manual“. Zugegriffen: 13. Mai 2025. [Online]. Verfügbar unter: https://www.cs.ubc.ca/research/flann/uploads/FLANN/flann_manual-1.8.4.pdf

[45] „Biomass Completes a Relay Race of a LEOP“. Zugegriffen: 12. Mai 2025. [Online]. Verfügbar unter: https://www.esa.int/Enabling_Support/Operations/Biomass_completes_a_relay_race_of_a_LEOP

[46] „OpenCV: Introduction“. Zugegriffen: 14. Mai 2025. [Online]. Verfügbar unter: https://docs.opencv.org/4.5.5/d1/dfb/intro.html

[47] „OpenCV: Feature Matching“. Zugegriffen: 14. Mai 2025. [Online]. Verfügbar unter: https://docs.opencv.org/4.5.5/dc/dc3/tutorial_py_matcher.html

[48] C. Stachniss, „Visual Features: Descriptors (SIFT, BRIEF, and ORB)“. [Online]. Verfügbar unter: https://www.ipb.uni-bonn.de/html/teaching/photo12-2021/2021-pho1-11-features-descriptors.pptx.pdf

[49] „OpenCV: OpenCV-Python Tutorials“. Zugegriffen: 15. Mai 2025. [Online]. Verfügbar unter: https://docs.opencv.org/4.5.5/d6/d00/tutorial_py_root.html

[50] A. S. Facility, „Introduction to SAR“. Zugegriffen: 8. August 2025. [Online]. Verfügbar unter: https://hyp3-docs.asf.alaska.edu/guides/introduction_to_sar/

[51] M. Ester, H.-P. Kriegel, J. Sander, und X. Xu, „A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise“, in *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, in KDD'96. Portland, Oregon: AAAI Press, Aug. 1996, S. 226–231. doi: 10.5555/3001460.3001507.

[52] „OpenCV: Morphological Transformations“. Zugegriffen: 12. August 2025. [Online]. Verfügbar unter: https://docs.opencv.org/4.x/d9/d61/tutorial_py_morphological_ops.html

[53] J. Brooke, *SUS - A Quick and Dirty Usability Scale*. Digital Equipment Co Ltd., 1986. Zugegriffen: 17. August 2025. [Online]. Verfügbar unter: https://hell.meiert.org/core/pdf/sus.pdf?/

[54] S. Schmidt, A. Minow, S. Schmidt, A. Minow, und I. Böckelmann, „Einsatz und Aussagekraft etablierter quantitativer Usability-Fragebögen in einem User-Test“, *Zentralblatt für Arbeitsmedizin, Arbeitsschutz und Ergonomie*, Bd. 70, Nr. 6, S. 256–263, Nov. 2020, doi: 10.1007/s40664-020-00399-2.

[55] S. Bradner, „Request for Comments: 2119“. Zugegriffen: 18. August 2025. [Online]. Verfügbar unter: https://www.ietf.org/rfc/rfc2119.txt

[56] N. Kano, N. Seraku, F. Takahashi, und S.-i. Tsuji, „Attractive Quality and Must-Be Quality“, *Journal of The Japanese Society for Quality Control*, Bd. 14, Nr. 2, S. 147–156, 1984, doi: 10.20684/quality.14.2_147.

[57] D. Clegg und R. Barker, *Case Method Fast-Track: A Rad Approach*. USA: Addison-Wesley Longman Publishing Co., Inc., 1994. [Online]. Verfügbar unter: https://dl.acm.org/doi/book/10.5555/561543