DHBW
Duale Hochschule
Baden-Württemberg
Mannheim

Studiengang Informationstechnik

Bericht zum Modul Praxis I

1. Praxisphase: 1.1.2015 - 29.3.2015
2. Praxisphase: 20.6.2015 - 30.9.2015

Schnittstellen Implementierung für ein SAR-Daten Analysewerkzeug in Python

-

Auswahl von Interessengebieten in F-SAR Produkten: Konzeption und Implementierung in Python

von

Michel van Kempen

Matrikelnummer: 5036467

Kurs: TINF14ITIN

DLR

Deutsches Zentrum für
Luft- und Raumfahrt e.V.
in der Helmholtz-Gemeinschaft

Standort: Oberpfaffenhofen

Microwaves and Radar Institute
Abteilung: SAR-Technologie

Betreuer: Dr. Rolf Scheiber

Eidesstattliche Erklärung

gemäß § 5 (3) der „Studien- und Prüfungsordnung DHBW Technik“ vom 22. September 2011.

Ich habe die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet.

Oberpfaffenhofen, den 11. September 2015

Inhaltsverzeichnis

Abbildungsverzeichnis

Abkürzungsverzeichnis

DLR	**D**eutsches Zentrum für **L**uft- und **R**aumfahrt
SAR	**S**ynthetic **A**perture **R**adar
GUI	**G**raphical **U**ser **I**nterface
XML	**E**xtensible **M**arkup **L**anguage
PyRAT	**Py**thon **RA**DAR **T**ools

Teil I

Schnittstellen Implementierung für ein SAR-Daten Analysewerkzeug in Python

1 Einleitung

1.1 Das Deutsche Zentrum für Luft- und Raumfahrt

Das DLR beschäftigt sich als Forschungsorganisation mit den Themen Luft und Raumfahrt, Verkehr, Umwelttechnologien, Sicherheit und auf vielen weiteren Feldern. Es ist aufgeteilt in 16 Standorte, die über Deutschland verteilt sind. Der Hauptsitz befindet sich in Köln.

Das Institut für Hochfrequenztechnik und Radarsysteme befindet sich am Standort Oberpfaffenhofen. Es beschäftigt sich in erster Linie mit der Entwicklung neuer Radarsysteme, für flugzeuggestützte- und satellitengestützte Plattformen, sowie mit der Auswertung schon aufgezeichneter Datensätzen.

Die Abteilung SAR-Technologie des Instituts entwickelt unter anderem neue Methoden zu Verarbeitung von Radardaten, ist für den Einsatz des flugzeuggestützten Radarsystems F-SAR verantwortlich und erstellt mithilfe der Datensätze Höhenmodelle und andere Produkte aus den Rohdaten. Einerseits werden Technologien weiterentwickelt und andererseits werden externe Aufträge bearbeitet, wie zum Beispiel die Erstellung von Höhenmodellen für Auftraggeber aus der Wirtschaft.

1.2 Motivation

Mithilfe von Radar-Sensoren, die auf Satelliten oder Flugzeugen montiert sind, können schnell und einfach in einer riesigen Dimension Flächen bis zu der kompletten Erdoberfläche in Datensätzen festgehalten werden. Diese Daten können eine Quelle für ein besseres Verständnis unserer Erde bilden. Es ist möglich in umfangreiche Erhebungen große Teile des Planeten mit einzubeziehen, indem die Daten von Satelliten ausgewertet werden. Dies ermöglicht es ein sehr genaues und für große Teile der Erde gültige Bilder und Modelle zu erzeugen und wiederum auf neue Erkenntnisse für verschiedene Bereiche, wie zum Beispiel die Umweltforschung oder die Bodengeografie hoffen lässt. Doch damit es möglich ist aus den Datensätzen von verschiedenen Sensoren die gesuchten Informationen, wie die Bodenbeschaffenheit oder Merkmale der Vegetation, zu extrahieren, müssen die Datensätze bearbeitet werden, um sie anschließend im Zusammenhang interpretieren zu können.

Mit den Python Radar Tools (PyRAT) können diese Bearbeitungsschritte an Datensätzen durchgeführt werden. Es ist des Weiteren wichtig möglichst viele Datenformate bearbeiten zu können, sowie die eigentliche Bedienung der Software vielseitig und geeignet zu gestalten. Sonst können Engpässe bei den Schnittstellen zwischen den verschiedenen Sensoren und der Software für das Einlesen der Daten oder zwischen Mensch und Software für das Verarbeiten von Daten entstehen, welche sich hemmend auf den ganzen Prozess auswirken können. Um eine solche Behinderung zu vermeiden ist es elementar, dass die Software die richtige Unterstützung bei dem genannten Prozess darstellt.

1.3 Aufgabenstellung

Da es meine erste Praxisphase war und erst einmal die Grundlagen geschaffen werden mussten, bekam ich noch keine von Anfang an fest formulierte Aufgabe, sondern nur ein übergeordnetes Thema, die Schnittstellenimplementierung in Python zugewiesen. Deswegen konnten sich die einzelnen Aufgabenstellungen erst mit der Zeit herauskristallisieren. Eine Gemeinsamkeit aller Aufgaben ist der Bezug auf PyRAT. Des Weiteren lassen sich meine Aufgaben aufteilen in Schnittstellenimplementierungen für Importroutinen um externe Datensätze einzulesen und in Implementierung bzw. Modifizierung der grafischen Benutzerschnittstelle. Bei meinen Aufgaben war es meistens gefordert schon vorhandene Strukturen zu verstehen um sie anschließend zu ergänzen oder mithilfe von Beispielen und Vorlagen neue Abschnitte zu schreiben.

2 Grundlagen der Fernerkundung mit Radar

Der erste Teil fasst die notwendige Theorie zusammen, welche für die Bearbeitung der Aufgabenstellungen wichtig war. Da ich mich das erste Mal mit diesem Themengebiet auseinander gesetzt habe, habe ich auch die Grundlagen für Radarsysteme aufgezeichnet. Zudem sind die Themen aufgrund der Übersicht nicht nach der chronologischen Reihenfolge geordnet, sondern wie es für die Aufgaben benötigt wurde. Des Weiteren sind zum Beispiel spezielle Sachverhalte, die zwar theoretischer Natur sind, aber nur für das Verständnis meiner Begründungen des Vorgehens von Bedeutung sind, in den zweiten Teil ausgelagert, da sonst zu viele unterschiedliche Themen angeschnitten werden müssten. Im zweiten Teil beschreibe ich meine Aufgabenstellung genauer, da diese sich oft erst mit dem Fortschritt der Aufgabe weiterentwickelt hat. Im Anschluss erläutere ich meinen Ansatz zur Lösung des gebotenen Problems und erkläre mein Vorgehen. Dabei gehen ich jedoch nicht bis auf die Erklärung des eigentlichen Programmcodes, welchen ich geschrieben habe, ein, da meine Aufgaben meistens in Modifizierungen der schon gegebenen Strukturen bestand und deshalb der Code im Gesamtzusammenhang gesehen werden müsste, sodass dies in einer Beschreibung der gesamten Struktur enden würde. Daher werde ich nur das zu Grunde liegende Konzept erläutern.

Des Weiteren ist zu beachten, dass ich hier nur die fertiggestellten Aufgaben dokumentiert habe. Ich konnte nicht alle Aufgaben bearbeiten oder habe sie letztendlich übersprungen, wenn es wichtigere Funktionen zu implementieren gab. Dies ist aufgrund des Umstandes, dass ich viele kleine Aufgaben bekommen habe, welche nicht unbedingt auf meine Zeit abgestimmt waren, zu Stande gekommen.

2.1 Grundlagen

In diesem Kapitel werde ich auf die benötigten Grundlagen für meine Aufgabe eingehen, mein Schwerpunkt liegt hierbei auf den essentiellen Themen, wie zum Beispiel die Theorie hinter der SAR-Technologie, der Programmiersprache Python sowie dem Programm PyRAT. Diese Grundlagen sollen den theoretischen Hintergrund für meinen Bericht bilden, worauf der praktische Teil aufbaut. In der praktischen Anwendung erwies sich von mitgebrachtem Wissen vor allem die Vorlesung Praktische Datenverarbeitung als hilfreich, da sie ein grund-

legendes Verständnis über in der Praxis notwendige und alltägliche Dinge wie zum Beispiel Versionskontrollsysteme vermittelt hatte.

2.1.1 Radartheorie

Der Begriff Radar kommt aus dem Englischen und ist die Abkürzung für Radio Detection and Ranging, also das Aufspüren und Messen mithilfe von Radiowellen. Die ersten Radarsysteme entstanden in England und wurden für die Überwachung des Luftraumes eingesetzt.

Das grundlegende Konzept eines Radarsystems besteht darin durch die Reflexion ausgesendeter elektromagnetischen Wellen Informationen über das Objekt zu erhalten, dies ist vornehmlich ein Bereich der erkundet werden soll. Damit gibt es Parallelen zwischen den Sehorganen von Lebewesen und optischen Systemen, wie zum Beispiel eine Fotokamera, da diese ebenfalls durch empfangenes Licht ihre Umgebung abbilden kann. Jedoch senden Radargeräte aktiv die elektromagnetischen Wellen aus, während eine Kamera die Sonne oder Lampen als Quelle verwenden. Ein weiterer großer Unterschied besteht in der verwendeten Wellenlänge. Das für den Mensch sichtbare Licht besitzt eine deutlich kleinere Wellenlänge im Vergleich zu den üblichen Frequenzen, die in Radarsystemen Verwendung finden. Daraus resultieren die Eigenschaften von Radarsystemen, dass sie auch bei Nacht uneingeschränkt verwendet werden können, da sie ihre eigene Quelle für elektromagnetische Wellen darstellen. Des Weiteren durchdringen Wellen mit großen Wellenlängen viele Materialien, an denen sichtbares Licht schon reflektiert wird, problemlos. Zum Beispiel stellen Wolken oder Nebel keine Hindernisse für Radargeräte dar, sodass ihre Reichweite deutlich höher sein kann als Sensoren, welche mit sichtbarem Licht arbeiten. Somit werden sie deutlich weniger von Umwelteinflüssen beeinflusst und eigenen sich daher besser für Aufgaben die ständige Funktionsfähigkeit voraussetzen. Ein Nachteil aufgrund der verwendeten Wellenlängen ist die, im Vergleich zu optischen Systemen, geringe Auflösung.

Es gibt verschiedenste Arten und Konfigurationen von Radarsystemen, daher werde ich mich im folgenden auf die im DLR verwendeten Systeme beschränken und daran die grundlegenden Züge erklären. Um Informationen zu erhalten wird zuerst eine elektromagnetische Wellen als kurzer Puls ausgesendet, um anschließend auf die zurück reflektierten Signale zu warten. Bei einem reflektierten Signal kann über die vergangene Zeit zwischen senden und empfangen die Entfernung des Objektes ermittelt werden. Wobei hier R_Z die Entfernung zum Ziel bezeichnet und t_L für die vergangene Zeit steht.

$$R_Z = \frac{t_L}{2} \cdot c_0$$

Je kürzer der ausgesandte Impuls ist, desto kleiner darf der Abstand zwischen Objekten, die hintereinander platziert sind und beide einen Teil der Wellen reflektieren, sein. Wenn der Abstand zu klein ist wird aus den beiden reflektierten Signalen der beiden Objekte ein verschwommenes Signal, welches suggeriert, dass in dieser Richtung nur ein größeres Objekt steht. Bei kurzen Signalen tritt jedoch das Problem auf, dass das Signal mit weniger Energie ausgesandt wird, sodass dies die Reichweite einschränkt. Daher werden die Impulse durch Impulsmodulation mit verschiedenen Frequenzen auf-moduliert, sodass trotz längerer Signale die Auflösung nicht mehr nur von der Länge des Impulses abhängig ist, sondern auch von der Bandbreite, die bei der Aufmodulierung des Impulses verwendet wird.

Die Auflösung zwischen Objekten, die von der Antenne aus gesehen nebeneinander stehen, ist abhängig von der Winkelauflösung des Radargerätes bzw. wie gut der Impuls fokussiert werden kann, diese Eigenschaft ist von der verwendeten Antenne abhängig. Je größer die Apertur der Antenne ist, umso genauer lässt sich der Radarimpuls in der Breite bzw. in Azimut fokussieren. Daher erreicht ein Radarsystem mit steigender Größe der Antenne auch eine bessere Winkelauflösung.

Die Achse in Ausbreitungsrichtung der Radarwellen wird Entfernung (range) genannt, die dazu senkrechte Achse wird als Azimut (cross-range) bezeichnet, bei SAR-Systemen ist sie parallel zur Bewegungsrichtung der Trägerplattform. Da Radarsysteme nur in zwei Dimensionen aufzeichnen, existiert keine weitere Achse.[6, S.1-103]

Auf diese Art und Weise lassen sich Gebiete mithilfe von gepulsten Radarsystemen erkunden. Es gibt natürlich noch viele andere Anwendungen, wie zum Beispiel die Luftüberwachung, welche in der Geschichte das erste Anwendungsgebiet für das Radar darstellt.

2.1.2 SAR-Theorie

Der Unterschied zwischen einem SAR-System und einem normalem Radar besteht in der Art und Weise, wie die Radarbilder mithilfe der Apertur zustande kommen. Im Gegensatz zu einem normalen Radarsystem ist eine Geschwindigkeit relativ zu dem Objekt bei einem Radar mit synthetischer Apertur zwingend notwendig, sodass der stationäre Betrieb für ein solches System in fast allen Fällen nicht in Frage kommt. Der Umstand, dass jedes aufgenommene Bild zu seinem Vorgänger versetzt ist, ermöglicht eine höhere Auflösung

als mit konventionellen Geräten, da durch die Prozessierung mehrerer Bilder eine bessere Auflösung berechnet werden kann. Auf diese Weise kommt auch der Name Synthetic Apertur Radar, also Radar mit synthetischer Apertur, zustande, da die Apertur durch die zeit verzögerte Aufnahme mehrerer Bilder künstlich verlängert wird. Die Größe der Apertur ist für abbildende Radargeräte entscheidend, da mit steigender Größe der Impuls besser gebündelt werden kann und somit eine höhere Winkelauflösung erreicht wird. SAR-Systeme sind immer senkrecht zur Bewegungsrichtung ausgerichtet und somit wird diese Auflösung in Bewegungsrichtung, auch Azimut genannt, verbessert. Die Auflösung in Entfernung (range) ist abhängig von der Länge des ausgesandten Pulses und kann nur durch Aufmodulierung verbessert werden. Je größer die Bandbreite des Impulses ist, desto größer ist der Gewinn an Auflösung in der Entfernung. Ein großer Vorteil für Flugzeuge sind die geringeren Ausmaße der Antenne, da diese Antennen ohne zu hohen Lustwiderstand und Gewicht realisiert werden können, die trotzdem eine hohe Auflösung erreichen.

Eine Eingenschaft der elektromagnetischen Welle, die Polarisierung, kann von Radarsystemen genutzt werden um weitere Informationen über das Objekt zu gewinnen. Die Polarisierung einer elektromagnetischen Welle bezeichnet in welche Richtung die Auslenkungen entstehen bzw. bildlich gesprochen wie sich die Welle um die Achse in Ausbreitungsrichtung gedreht hat. Da die Materialien, auf welche die Radarwellen treffen, sich in ihrer Auswirkung auf deren Polarisation unterscheiden, kann mithilfe der Polarisation auf weitere Eigenschaften geschlossen werden. Die Messung erfolgt sozusagen durch einen Filter, welcher entweder nur die senkrechten oder nur die waagerechten Anteile der reflektierten Welle durchlässt. So entstehen Datensätze mit unterschiedlichen Polarisationen vom gleichen Objekt. Mithilfe dieser Datensätze kann bei unterschiedlicher Farbgebung der verschiedenen Kanälen ein Radarbild in Farben visualisiert werden.[6, S.26]

Die Interferometrie ist ein weiterer Modus für Radargeräte und eignet sich für die Erstellung von Höhenmodellen und die Erfassung von Fließgeschwindigkeiten am Boden. Im einfachsten Fall wird sie mit zwei Antennen die in einem Abstand voneinander montiert sind realisiert. Mithilfe der Interferometrie ist es zum Beispiel möglich genauere Abstandsmessungen als im normalen Betriebsmodus mit nur einer Antenne ohne Interferometrie durchzuführen. Im Normalfall kann die Entfernung zwischen Antenne und einer reflektierenden Fläche durch die Messung der Zeit unter der Voraussetzung, dass sich die Welle mit Lichtgeschwindigkeit fortbewegt, berechnet werden. Anschließend kann mithilfe der Koordinaten der Antenne die absolute Höhe des Objektes ermittelt werden. Jedoch ist die Genauigkeit von der verwendeten Wellenlänge

abhängig und die Winkelmessung des Einfallswinkels der Welle zu ungenau. Um eine höhere Genauigkeit zu erreichen können zwei Antennen, die eine unterschiedliche Entfernung zum Ziel haben, Messungen des Abstandes vornehmen, indem sie die Phasendifferenz der empfangenen Welle an den beiden Antennen messen. Womit die auf normalen Weg gemessene Entfernung korrigiert werden kann.[6, S.272]

Für das Arbeiten mit Daten von Radarsystemen ist das Verständnis von elektromagnetischen Wellen elementar, da dies die Grundlage darstellt. Deshalb konnte das gewonnene Wissen aus den Physikvorlesungen des letzten Semesters in die Erarbeitung mit einfließen.

3 Radardatenanalyse mit Python

3.1 Python

Ein weiterer wichtiger Bestandteil für die notwendigen theoretischen Kenntnisse war das Erlernen der Programmiersprache Python. Python ist eine plattformunabhängige Scriptsprache. Sie wird in Zukunft immer mehr die im Institut vorherrschende proprietäre Sprache ablösen. Deshalb werden alle neuen Projekte in Python geschrieben. Python steht für Quelloffenheit und sehr gut verständlichen Syntax, sodass sie Vorteile wie Unabhängigkeit von kommerziellen Unternehmen sowie einen erleichterter Umstieg von anderen Sprachen, durch die gute Lesbarkeit des Codes, bietet. Des Weiteren kann Python als objektorientierte oder funktionale Sprache eingesetzt werden. Ein weiterer Vorteil ist die einfache Integration von C-Code, sodass die sonst verhältnismäßig unoptimierte und somit langsame Scriptsprache beschleunigt werden kann.

Der grundsätzliche Unterschied der Philosophien hinter Java, welches ich schon aus der Vorlesung kannte, und Python machte es schwierig das Gelernte auf die neue Programmiersprache zu beziehen, da das Konzept von Java sehr auf Sicherheit beruht und der Syntax eher mit C und C++ vergleichbar ist. Ganz im Gegensatz dazu Python, dem Programmierer werden viele Freiheiten gewährt und der Syntax ist an andere Scriptsprachen angelehnt. Jedoch konnte ich die Grundsätze von Java auch in Python wieder erkennen, da die Programmierung an sich trotz unterschiedlicher Konzepte immer wieder auf die gleichen Prinzipien zurückgreift, wie beispielsweise die Kontrollstrukturen für die Steuerung des Programmflusses. Das auffälligste an Python-Code ist, dass im Unterschied zu Java die Verwendung von Zeilen und Absätzen eine Bedeutung hat und deshalb viele Klammern, die den Code sonst unübersichtlich machen, weg fallen. Dagegen dienen Zeilen usw. in Java und anderen Sprachen nur der Übersicht für menschliche Leser, werden jedoch vom Compiler ignoriert. In Python in Gegensatz dazu wird der Programmierer gezwungen übersichtlichen Code zu schreiben. Des Weiteren gibt es in Python die Möglichkeit einen interaktiven Interpreter aufzurufen mit dessen Hilfe Zwischentests schneller durchgeführt werden können und die Möglichkeit bietet unbekannte Funktionen erstmal in diesem Umfeld zu testen, um sie anschließend in den Programmcode einzubauen.

3.2 Radar-Sensoren

Um die Importroutinen für die jeweiligen Sensoren zu schreiben ist ein grundlegendes Wissen über ihre Eigenschaften notwendig, da diese sich auch in den Metadaten widerspiegeln und so das Verständnis für die jeweiligen speziellen Anforderungen geschaffen wird.

Das experimentelle Synthetik-Apertur-Radarsystem, kurz E-SAR, gehört zum DLR und ist an ein Flugzeug montiert. Es besteht aus mehreren Antennen, für die Frequenzen der Bänder X, C, L und P, und wird zur Erprobung neuer Radartechniken eingesetzt. In diesem Zusammenhang bezeichnet ein Band, was die Abkürzung von Frequenzband ist, einen Teil des elektromagnetischen Spektrums. Mithilfe der unterschiedlichen Frequenzbänder können die Objekte unterschiedlich betrachtet werden. Je kleiner die Wellenlänge ist, desto höher ist die Auflösung und gleichzeitig ist aber die Eindringtiefe geringer, sodass hohe Frequenzen mit kleineren Wellenlängen schneller wieder reflektiert werden. Durch dieses Verhalten unterschiedlicher Schwingungen elektromagnetischer Wellen können mehr Informationen über die Ziele empfangen werden. Des Weiteren ist der E-SAR-Sensor in der Lage horizontale und vertikale Polarisationen voneinander getrennt zu senden und zu empfangen, sodass durch die unterschiedliche Auswirkung der Objekte auf die Polarisierung weitere Informationen gewonnen werden können. Der E-SAR-Sensor wurde 2006 durch F-SAR abgelöst.[2]

TerraSAR-X bezeichnet einen Erdbeobachtungssatelliten des DLR der im Jahr 2007 gestartet wurde. Er ist mit einem SAR-Sensor, der im X-Band arbeitet und somit eine hohe Auflösung erlaubt, ausgestattet. Das System wird für das Kartografieren der Erde und Aufdecken von Veränderungen verwendet. Es erreicht eine Auflösung von bis zu einem Meter.[4]

Die Bezeichnung KOMPSAT-5 steht für Korea Multi-Purpose Satellite-5 und ist ein koreanischer Satellit mit diversen Aufgaben im Bereich der Erstellung von Karten und Überwachung von geografischen Zielen. Er erreicht eine Auflösung von einem Meter und ist seit 2013 im Einsatz.[3]

Damit bestimmte Funktionen, wie das Einlesen eines bestimmten Datenformates, nicht immer wieder aufwendig neu geschrieben werden müssen und somit wiederverwendbar sind, werden wichtige Funktionen in Softwarebibliotheken abgelegt, sodass die Funktionen immer bereit stehen. Qt ist eine in C++ geschriebene Bibliothek für die Programmierung von Benutzeroberflächen und da es auch eine Schnittstelle für Python gibt wird sie auch für die Programmierung der GUI verwendet.

3.3 PyRAT

Das institutsinterne Programm Python Radar Tools, kurz PyRAT, ist ein in Python geschriebenes Programm für die Bearbeitung von Radarbildern. Es ist anwendbar auf Radarprodukte die schon im Falle von SAR-Daten fertig prozessiert wurden, dass bedeutet, dass die Radardaten schon aufbereitet sind und im Gegensatz zu den Rohdaten die abgebildeten Objekte für den Menschen erkennbar dargestellt werden. PyRAT dient dazu diese Bilder zum Beispiel mithilfe von Filtern und anderen Tools zu bearbeiten. Dabei ist es darauf ausgerichtet, dass es möglichst einfach ist neue Werkzeuge zu programmieren bzw. sollen vorgefertigte Strukturen des Frameworks dem Programmierer viel Arbeit abnehmen, wie das Einbinden oder die Optimierung des Einlesens von Daten, sodass sich Programmierung von neuen Werkzeugen nur auf den Kern, nämlich den Code der das Werkzeug ausmacht, beschränkt. PyRAT wird in erster Linie über die Kommandozeile verwendet, besitzt jedoch auch eine grafische Benutzerschnittstelle, sodass die eingelesenen und bearbeiteten Bilder auch direkt angezeigt werden können. Eine große Herausforderung bei der Realisierung eines solchen Programms sind die deutlich größeren Datenmengen, die bei Radarbildern anfallen, im Vergleich zu normalen Bildbearbeitungsprogrammen. Die Struktur an sich ist geordnet nach den verschiedene Aufgaben, welche die jeweiligen Module besitzen. Wenn in der Kommandozeile gearbeitet wird müssen die einzelnen Module und deren Methoden, um einen Bearbeitungsschritt durchzuführen, selbst angesprochen werden. Dies wird mit dem interaktiven Interpreter von Python durchgeführt. Er bildet die Grundstruktur für die Ausführung des Programms. Wenn die GUI verwendet werden soll, wird zuerst mithilfe der Bibliothek Qt eine Schnittstelle generiert, welche dann die einzelnen Module mit ihren jeweiligen Funktionen aufruft.

3.4 Umsetzung der Aufgaben

Nachdem jetzt die notwendige Theorie zusammengefasst und beschrieben wurde, wird im Folgenden in Schwerpunkten das Vorgehen bei meinen Aufgaben in der Praxis beschrieben. Einerseits habe ich verschiedene Importe für Radarsensoren verbessert oder selber geschrieben. Auf der anderen Seite habe ich zusätzliche Funktionen in der grafischen Benutzerschnittstelle implementiert. Das Vorhandensein der Importroutine ist elementar für das Arbeiten mit Radardaten, da diese sonst nicht in das Programm eingelesen werden könnten, sodass PyRAT für die Bearbeitung von Radardaten, von Sensoren mit Datenformaten für die es noch keine

Importroutine gibt, nutzlos ist. Da es sich bei allen Radarsensoren um Neuentwicklungen handelt, ist das Datenformat von Sensor zu Sensor unterschiedlich, sodass letztendlich für jeden Sensor ein individueller Importmechanismus programmiert werden muss. Dabei ist zu beachten, dass sich nicht nur die Formate, in denen die Bild- und Metadaten gespeichert werden, unterscheiden, sondern dass sich Sensoren in weiteren Eigenschaften unterscheiden, die für die Verarbeitung der Daten wichtig sind und beachtet werden sollten. Zum Beispiel ist es unterschiedlich in vielen verschiedenen Frequenzen der jeweiligen Sensoren Radardaten aufnehmen kann. Zuweilen gibt es sogar bei demselben Sensor unterschiedliche Formate, wie beispielsweise beim E-SAR. Es folgt eine Übersicht über die Aufgaben.

- E-SAR Import Widget (GUI)
- Importroutinen zusammenführen
- TerraSAR-X Importroutine
- Farbauswahl (GUI)
- KOMPSAT-5 Importroutine
- Blockimport

3.4.1 E-SAR Import Widget (GUI)

Meine erste größere Aufgabe bestand darin den bestehenden Import für den E-SAR-Sensor zu modifizieren und den GUI Dialog zu verbessern. Da das E-SAR mit verschiedenen Datenformaten arbeitete, gab es auch dementsprechend mehrere Importe. Der erste Teil der zu bewältigenden Aufgabe bestand darin den E-SAR Import, der später mal alleine die beiden Formate des Sensors beherrschen sollte, so zu modifizieren, sodass er zumindest eines der beiden Formate einlesen konnte, um im Anschluss daran den Eingabedialog in der GUI zu verbessern.

Der erste Schritt erwies sich als eine Tätigkeit die ausschließlich aus Fehlersuchen bestand, da der Import in den früheren Versionen schon funktionsfähig, jedoch durch Aktualisierungen von PyRAT nicht mehr vollständig kompatibel mit dessen aktueller Version war. Der nächste Schritt war die Verbesserung des Dialoges, dessen Aufgabe es ist dem Benutzer die Auswahl der richtigen Datei, mit den Radardaten, zu erleichtern. Das Ziel war es, dass der Anwender nur den Dateipfad mithilfe eines grafischen Eingabefensters anzugeben braucht, damit anschließend das Programm anzeigt um was für E-SAR Daten es sich dabei handelt. So kann der Benutzer

feststellen ob es sich bei der angegebenen Datei um die gewünschte für den Import handelt. Da der E-SAR Sensor auf mehreren Frequenzbändern und in unterschiedlichen Polarisationen Daten aufnehmen kann und diese dann wieder in zwei verschiedenen Formaten abspeichert, ergeben sich so die benötigten Attribute, die bei einem E-SAR Datensatz angezeigt werden müssen. Des Weiteren gibt es noch den Fall, dass mehrere Bilder gleichzeitig angezeigt werden sollen. Dies führt zu einem bunten Bild, im Gegensatz zu Bildern mit nur einem Kanal, die nur schwarz-weiß gefärbt sind. Ein Kanal bezeichnet hierbei einen Datensatz mit einer bestimmten Polarisation und einem bestimmten Band. Mit mehreren Kanälen des gleichen Objektes können Bilder in Falschfarben angezeigt werden. Für diese Funktionalität muss statt einem Pfad zu einer Datei nur der Pfad zu dem Ordner, der die verschiedenen Datensätze von den unterschiedlichen Kanälen enthält, eingegeben werden. Durch die Auswahl im Dialog, welche Attribute die Kanäle besitzen sollen, kann so die Menge eingegrenzt werden, sodass nicht alle Dateien aus dem benannten Ordner in PyRAT geladen werden. Im Dialogfenster sollen die verschiedenen Attribute, die in dem Ordner zur Verfügung stehen, angezeigt werden, damit der Benutzer nur noch die gewünschte Kombination von Attributen auswählen muss. Zudem wird in dem Dialog noch eine Möglichkeit benötigt nicht das gesamte Bild zu importieren, sondern nur einen Ausschnitt daraus herauszuschneiden. Das Layout für den grafischen Dialog war schon mit Qt programmiert und vorgegeben, jedoch mit nur eingeschränkter Funktionalität, sodass es meine Aufgabe war die benötigten Informationen, wie zum Beispiel um welche Datensätze es sich handelt und welche Eigenschaften diese besitzen, dem Dialogfenster bereitzustellen.

Um diese Funktionen zu realisieren mussten zuerst die benötigten Informationen aus den zusätzlichen Dateien mit Metainformationen ausgelesen werden. Dazu wurde eine neue Methode implementiert, welche aus dem Dateinamen der Datei, welche die Bilddaten enthält, als Parameter den Namen der Datei, wo die Metainformationen abgespeichert sind, errechnet, um anschließend die gewünschten Informationen unter Berücksichtigung der Syntax des Formates, in welchem die Metainformationen abgespeichert wurden, zu extrahieren. Die Metainformationen enthalten zusätzliche Parameter, abgesehen von den Eigenschaften wie die Polarisation und der Frequenz sind auch Parameter enthalten, die benötigt werden um die Daten anschließend unter bestimmten Aspekten mit Filtern korrekt bearbeiten zu können. Da leider bei der Strukturierung der Metadatendatei, nicht die Struktur von XML oder eine vergleichbare verwendet wurde, ist die Funktion darauf angewiesen, dass die eigene Struktur immer sehr genau eingehalten wird. Um die mit derartigen Fehlern verbundene, Abstürze zu verhindern

habe ich eine Fehlerbehandlung implementiert, sodass aufgrund einer falschen Formatierung der Metadaten das ganze Programm nicht seine Funktionsfähigkeit verliert, sondern nur diese zusätzlichen Features, welche nur die Bedienung erleichtern sollen, beeinträchtigt sind. Im Anschluss wurde die Funktion in den Import integriert. Eine Herausforderung war die neuen Funktionen so in den Code zu schreiben, dass die Kommandozeilen Schnittstelle, welche nicht auf Features der grafischen Funktionen zugreifen kann, immer noch ihre Funktionsfähigkeit beibehält und nicht beeinträchtigt wird.

Es folgten kleinere Verbesserungen für die erleichterte Bedienung der GUI, wie zum Beispiel die Implementierung der Funktion, welche bestimmte Schaltflächen durch einen Doppelklick aktiviert, um nicht recht umständlich erst ein Menü zu öffnen und dort die Option zu wählen, welche die betreffende Schaltfläche aktiviert. In Python existieren viele erweiterten Funktionen, genauso wie für Qt in der Programmierung für grafische Schnittstellen. Das hat zur Folge, dass bei der Implementierung solcher Zusatzfunktionen, aber auch beim allgemeinen Programmierung, der größte der Teil der benötigten Zeit darauf entfällt, nach den geeigneten Funktionen in den Dokumentationen der diversen Softwarebibliotheken zu suchen. Um letztendlich die gewünschten Funktionen mit nur sehr wenig neuem Code zu implementieren. Dies fällt natürlich deutlich stärker aus, je unbekannter das betreffende Thema ist und umso weniger Dokumentation zur Verfügung steht.

Eine weitere auffallende Problemstellung, die bei der Programmierung der Benutzerschnittstelle deutlich wurde, ist, dass es nicht so leicht ist die genauen Bedürfnisse des Benutzers und Notwendigkeiten für die Anwendung zu erfassen, da ich letztendlich nicht der Anwender bin und damit verbunden nicht die genauen Arbeitsvorgänge kenne. Deshalb erwies dich ein Austausch immer als sehr hilfreich.

3.4.2 Importroutinen zusammenführen

Der nächste Schritt in der Bearbeitung des E-SAR Imports war die Zusammenführung der einzelnen Importroutinen, für die beiden unterschiedlichen Datenformate, zu einer Importroutine, welche alle Formate beherrschte. Der Unterschied zwischen den beiden Datenformaten des E-SAR Sensors ist die unterschiedliche Speicherung der Bilddaten. Da bei dem flugzeuggestützten E-SAR in Realität der Winkel zwischen Flugrichtung und der Ausrichtung der Radarantenne nicht genau 90 Grad beträgt, wird nicht ein Rechteck des Bodens aufgezeichnet, sondern ein Parallelogramm. Bei dem einen Format wird das Parallelogramm wieder begradigt,

wie ein Rechteck, bei dem anderen nicht. Durch diese Ursache unterschieden sich die beiden Import in der Art auf welche Weise die Daten eingelesen, da letztendlich das Gleiche in PyRAT importiert werden sollte. Des Weiteren unterschieden sich die beiden bestehenden Importroutinen auch im Aufbau, sodass die erste Schwierigkeit darin bestand die wichtigen Unterschiede des Einlesevorgangs heraus zu filtern.

Um das Schreiben einer Importroutine in PyRAT zum Einen einfacher, zum Anderen standardisiert zu machen, bedient man sich deshalb bei einem Prinzip der objektorientierten Programmierung. Es existiert eine sogenannte Elternklasse, welche alle benötigten und somit auch standardisierten Eigenschaften und Funktionen besitzt, die ein Import, hier gleichgesetzt mit einer Importroutine, benötigt, damit dieser ohne weiteres in PyRAT funktioniert. Diese Elternklasse bildet die Grundlage für alle folgenden Importroutinen und in dem die Importroutinen von ihre Eigenschaften übernehmen oder auch erben, sind diese als Standard gesetzt und können deshalb als gegeben angenommen werden. In diesem Fall muss sich der Programmierer eines Imports nur auf eine Methode konzentrieren, die den spezifischen Code für das jeweilige Format enthält. Dabei ist noch zu beachten, dass auch die Metadaten des Datensatzes an das Programm weiter gegeben werden müssen. Nach der Vereinigung der beiden Importe in einen neuen Import, mithilfe von Fallunterscheidungen an den Stellen, wo sich die beiden Einleseroutinen unterschieden, war es wichtig möglichst viele Fehler abzufangen und für nicht korrekt gesetzten Parameter eine Lösung zu programmieren. Es stellte sich vor allem in der anschließenden Test-Phase als schwierig heraus auf alles vorbereitet zu sein und die Stabilität sicherzustellen. Das über viele Jahre eingesetzte Format des E-SARs enthält ein paar Variationen, dies erwies sich als insbesondere kritisch bei der Syntax der Dateinamen, da nur durch ein festgelegtes Muster gewährleistet werden kann, dass die Pfade der Metadaten richtig erkannt werden. Letztendlich konnte die Routine, nach mehreren Tests mit den verschiedenen Formaten und Fehlerbehebung, in die aktuelle Version mit aufgenommen werden.

3.4.3 TerraSAR-X Importroutine

Die nächste Aufgabe bestand darin die Importroutine für den Satelliten TerraSAR-X zu bearbeiten. Den Vorgang des Lesens des binären Dateiformates ist auf eine externe Bibliothek ausgelagert worden, deren Schwerpunkt auf dem Lesen und Bearbeiten von geografischen Daten liegt, sodass das Problem darin lag die Daten richtig in die bereitgestellten Funktionen der Bibliothek fließen zu lassen. Des Weiteren sind die Datensätze des Satelliten aufgrund ihrer Größe problematisch, da ein einzelner Datensatz schon mehrere Gigabyte groß ist. Daher

ist das Laden solcher Dateien mit dem Import nur mithilfe von Serverhardware, die über ausreichend große Kapazitäten verfügt, zu bewerkstelligen. Der Arbeitsspeicher muss groß genug sein um den die gesamten Bilddaten auf einmal zu fassen.

3.4.4 Farbauswahl (GUI)

Die Aufgabe bestand darin zusätzliche Features für die grafische Benutzerschnittstelle zu programmieren. Die erste benötigte Zusatzfunktion bestand in der Möglichkeit die Belegung der RGB-Farben der verschiedenen Kanälen über die GUI zu verändern.

Ein Kanal repräsentiert eine Ansicht auf eine bestimmte Szene oder Objekt. Hier sind mit mehrere Kanäle immer verschiedene Datensätze zu genau dem gleichen Objekt gemeint, also zum Beispiel bei einem Ausschnitt von einem Gelände kann es dann Datensätze mit unterschiedlicher Polarisierung geben, die aber alle dasselbe abbilden.

Im Normalfall wird ein Radarbild mit nur einem Kanal in Grautönen dargestellt. Die Bilddaten an sich bestehen aus einem Zahlenwert pro Pixel. Dieser kommt zu Stande indem von dem komplexen Ausgangswert der Betrag gebildet wird. Die Ausgangswerte von Radargeräten befinden sich immer in der Gaußschen-Zahlenebene, da die Anteile der empfangenen Wellen zueinander stehen. Dem Betrag wird eine Farbe zugewiesen. Ist der Wert zum Beispiel der höchste in Vergleich zu den Übrigen, wird ihm der hellste Ton zugewiesen. Da es eine begrenzte Anzahl von Abstufungen bei der Helligkeit gibt, bekommen gleich große Werte die gleiche Stufe zugewiesen. Auf diese Art und Weise werden die Werte bei einem einkanaligen Bild dargestellt.

Wenn jedoch mehrere Kanäle in einem Bild dargestellt werden sollen, werden Farben verwendet. Alle Farben können durch die drei Farbkanäle Rot, Grün und Blau, auch kurz als RGB bekannt, repräsentiert werden. Je nachdem mit welcher Intensität die jeweilige Farbe in das Ergebnis mit einfließt, ändert sich der Farbton. Mit drei Datensätzen kann ein Gesamtbild in Farben dargestellt werden indem ein Kanal durch eine Farben dargestellt wird.

Der wichtige Punkt ist, dass die Farbe die ein Pixel letztendlich repräsentiert nicht nur durch die drei Werte der Kanäle bestimmt ist, sondern auch deren Aufteilung auf die drei Farbkanäle Rot, Grün und Blau. Meine Aufgabe war dir Implementierung einer Funktion mit deren Hilfe sich diese Zuordnung zwischen den Kanälen und den Farben variabel einstellen lässt und nicht mehr ausschließlich statisch durch das Programm erfolgt.

Die zu realisierende Funktion kann in zwei Teile aufgeteilt werden. Der erste Teil muss über die

GUI von dem Benutzer abfragen wie die Kanäle den Farben zugeordnet werden sollen und die bestehende Ordnung entsprechend visualisieren. Der andere Teil ist dafür verantwortlich die Einstellungen umzusetzen. Bei der Implementierung habe ich darauf geachtet mit möglichst wenigen Änderungen im Code die Funktion umzusetzen, sodass er seine alte Struktur beibehält. Für die Implementierung des ersten Teils musste das Objekt, was in der in Qt geschriebenen grafischen Schnittstelle die Übersichtsstruktur beschreibt, verändert werden. In dieses Objekt wurde ebenfalls die notwendige Datenstruktur gesetzt, welche die derzeitige Zuweisung von Farben und Kanälen speichert. Diese war vorher nicht notwendig, da die ersten drei Kanäle der Reihenfolge nach Rot, Grün und Blau zugewiesen wurden. Für die Speicherung verwendete ich ein Dictionary. Dies ist eine Struktur in Python, welche mit Key/Value-Paaren arbeitet. Indem man den Key angibt, erhält man den darunter gespeicherten Wert. In diesem Fall wurden die Nummern der Kanäle als Schlüssel verwendet und darunter jeweils die zugehörige Farbe als Value abgelegt. Der Vorteil dieser Datenstruktur ist die Übersichtlichkeit und in Relation zu anderen Datenstrukturen, wie zum Beispiel Arrays, vergleichsweise hohe Overhead kann durch den geringen Umfang, der zu speichernden Daten, vernachlässigt werden. Die Übersicht in der grafischen Darstellung kann am Einfachsten als Liste, deren Einträge die jeweiligen Kanäle darstellen, beschrieben werden. Damit bei einem Rechtsklick auf einen Listeneintrag ein Menü erscheint, in welchem die gewünschte Farbe ausgewählt werden kann, musste ich dessen Signal mit einer Methode verbinden. Dies wurde durch vorgefertigte Strukturen in Qt realisiert. In dieser Methode wird mithilfe der beiden Parameter, welcher Kanal angeklickt und welche Farbauswahl getroffen wurde, die Zuweisungen im Dictionary so verändert, dass der angegebene Kanal die gewünschte Farbe zugewiesen bekommt, sowie der Kanal, dem die Farbe ursprünglich zugewiesen war, die alte Farbe erhält. Die Farben werden letztendlich getauscht. Die Methode ruft anschießend die Aktualisierungs-Methode für das gesamte Fenster auf, damit die Änderungen im Dictionary wirksam werden.

Für die Ausführung der Änderungen musste entsprechender Code in der Methode, welche für die Berechnung des Radarbildes der GUI zuständig ist, geschrieben werden. Da die Zuweisung in einem der Objekte, worauf das Hauptfenster eine Referenz besitzt, gespeichert sind, kann von der Methode auf das zuvor beschriebene Dictionary zugegriffen werden. Um die Zuweisung zwischen Farben und Datensätze zu verändern muss die Reihenfolge verändert werden, in welcher die Datensätze im Arbeitsspeicher aufgelistet sind. Die ersten drei Datensätze werden von PyRAT für die Darstellung des Bildes verwendet. Die Datensätze werden in speziellen Arrays der Python-Bibliothek NumPy, welche in C geschrieben sind und sich daher durch eine

deutlich höhere Performance, gerade bei großen Datenmengen, als bei normale Python Arrays auszeichnen, gespeichert. Letztendlich sind alle Daten für das Bild in einem dreidimensionalen NumPy-Array geordnet, jeweils auf einer Achse die einzelnen Kanäle, die X- und Y-Koordinaten. In jedem Feld dieses Arrays ist ein Wert gespeichert, welcher im Bild anschließend dafür verantwortlich ist wie intensiv, der seinem Kanal zugewiesene Farbton, in der Farbe des Pixel auftritt. Eine Änderung der Zuordnung der Farben wird durch das Vertauschen der Einträge der Kanäle auf der Achse im Array bewerkstelligt. Diesem Vorgang kommt die starke Performance des NumPy-Arrays zu Gute, sodass keine merkliche Verzögerung entsteht. Durch das Tauschen der Einträge werden deren Datensätze in einer anderen Reihenfolge für das Bild eingelesen und deshalb anderen Farben zugeteilt und auf diese Art und Weise die Zuordnung der Farben verändert.

Ein weiterer Vorteil der Implementierung ist die Möglichkeit mehr als drei Kanäle zu importieren und anschließend zwischen den Kanälen zu wechseln indem jene, welche angezeigt werden sollen eine Farbe zugewiesen bekommen. Nur die Kanäle, welche eine Farbe zugewiesen bekommen haben, werden visualisiert. Diese Funktion ermöglicht eine nachträgliche Auswahl der Kanäle, welche sonst schon vor dem Laden getroffen werden musste.

Der nächste sinnvolle Schritt war die Implementierung einer ebenfalls über das GUI gesteuerten Funktion, welche es ermöglicht sich nur einen einzelnen Kanal anzeigen zu lassen. Dieser wird dann nur noch in schwarz-weiß angezeigt werden. Natürlich sollte man auch wieder anschließend in den Modus der Farben zurück springen können und sich somit wieder drei Kanäle gleichzeitig anzeigen lassen. Da die notwendigen Strukturen schon implementiert waren, wie das Menü, welches sich beim Rechtsklick öffnet oder die Speicherung der Zuordnung, konnte die neue Funktion nur durch Erweiterung des bestehenden Systems erreicht werden.

Die implementierten Funktionen in der Benutzerschnittstelle erhöhen die Flexibilität, wie zum Beispiel die Auswahl von verschiedenen Kanälen oder die Zuweisung der Farben, da diese Einstellungen jetzt in der GUI vorgenommen werden können und nicht mehr durch die Verschiebung der Datensätze im Dateisystem bewerkstelligt werden müssen. Zudem vereinfacht die Möglichkeit schnell die Belegung der Farbkanäle zu ändern die Interpretation der visualisierten Datensätze. Jedoch benötigt der bestehende Ansatz noch eine weitere Verbesserung, da in der derzeitigen Implementation zwar das Anzeigen eines einzelnen Kanals aus dem Pool von mehreren Kanälen ermöglicht, aber die Bearbeitung durch zum Beispiel einen Filter sich noch auf alle Kanäle auswirkt. Einer derartige Erweiterung wäre sicherlich sinnvoll, erfordert jedoch auch die Implementation an einer ganz anderen Stelle im Code.

3.4.5 KOMPSAT-5 Importroutine

KOMPSAT-5 ist ein südkoreanischer Satellit, welcher ebenfalls mit einem SAR-Sensor ausgestattet ist. Um auch dessen Daten in PyRAT einlesen zu können, war es meine Aufgabe eine Importroutine zu schreiben. Als Ausgangspunkt hatte ich ein Stück Quellcode als Beispiel für das Einlesen dieses Datenformates sowie mehrere Beispieldatensätze zur Verfügung. Die grundlegende Struktur der Routine ist vergleichbar mit der, die im Abschnitt E-SAR-Import beschrieben wurde, da sie ebenfalls auf der gleichen Elternklasse beruht. Des Weiteren bildeten die übrigen Importroutinen eine Quelle für die Programmierung des neuen Importes. Das Auslesen der Bilddaten wird wie im TerraSAR-X Import von einer externen Bibliothek übernommen. Weitere Probleme der Importroutine waren das Einlesen der Metadaten sowie die Implementierung einer Funktion um nur einen Teil des Gesamtbildes zu importieren.

Um die Metadaten aus der XML-Datei zu extrahieren, habe ich eine Bibliothek von Python verwendet, die für XML-Daten geeignet sind. Da für den Anfang nur ein paar Parameter aus der Datei ausgelesen werden brauchten, lag das Augenmerk ebenso darauf später schnell erweiterbar zu sein und zu ermöglichen ohne lange Einlesezeit weitere Parameter hinzufügen zu können. Es wurde eine Fehlerbehandlung für das Einlesen der Metadaten implementiert. Wenn es beim diesem Vorgang einen Fehler gibt, wird der Ablauf nicht unterbrochen, sondern die Metadaten werden übersprungen

Die Funktion für das Einlesen eines Ausschnittes konnte mit Abänderung von anderen Importroutinen übernommen werden. Des Weiteren liegt der Fokus zuerst auf der Funktionsfähigkeit des eigentlichen Imports, sodass es noch keinen komfortablen GUI-Dialog gibt, wie im Vergleich zum E-SAR-Import.

3.4.6 Blockverarbeitung

Eine Herausforderung beim Arbeiten mit Radardaten, vor allem im Unterschied zu Fotos bei normalen Bildbearbeitungsprogrammen, ist die Größenordnung der Bilddaten, da SAR-Sensoren besonders hohe Auflösungen erreichen und die abgebildeten Bereiche sehr groß sein können. Das hat zur Folge, dass die Bilddaten viel Speicherplatz benötigen, so ist zum Beispiel ein Datensatz des TerraSAR-X mit nur einem Kanal je nach Ausschnitt schnell mehrere Gigabyte groß. Dieser Umstand hat auch Auswirkungen auf den Importvorgang solcher Datensätze. In PyRAT werden beim Import die Datensätze mit ihrem unter Umständen speziellen Format eingelesen und anschließend in einem für alle Daten gleichen und einfach

zu bearbeitendem Format in einer temporären Datei auf der Festplatte wieder abgelegt. Von dort aus können sie dann auf dem Bildschirm ausgegeben oder bearbeitet werden. Beim einem einfachen Importvorgang müssen jedoch alle Daten zwischen dem Auslesen und dem Schreiben in den temporären Ordner im Arbeitsspeicher gespeichert werden. Dieser Umstand ist einerseits kein Problem, wenn es sich um kleine Datensätze handelt oder anderseits wenn PyRAT auf einer Hardware verwendet wird, die genügend Kapazitäten zur Verfügung stellt. Doch um es zu ermöglichen auch auf Desktopcomputern größere Datensätze, wie sie zum Beispiel häufig bei Satellitendaten auftreten, einlesen zu können, wurde die Möglichkeit der Blockverarbeitung eingesetzt. Ein Datensatz in Blöcken einzulesen bedeutet immer nur einen Ausschnitt aus der Ausgangsdatei einzulesen, in den Arbeitsspeicher zu laden und im Anschluss in der temporären Datei abzulegen. Aus mehreren Blöcken setzt sich dann der Datensatz in der temporären Datei auf der Festplatte wieder zusammen. Dieses Prinzip findet, mit einem anderen Ziel, in PyRAT schon in der Ausführung von Filtern Anwendung. Damit die Rechenlast auf mehrere Kerne aufgeteilt wird, wenn ein Filter eingesetzt wird, teilt PyRAT die Daten ebenfalls in Blöcke auf, sodass auf mehreren Kernen des Prozessors parallel gerechnet werden kann. Dies verfolgt zwar das Ziel der Minimierung des Rechenzeit durch Parallelisierung im Gegensatz zu Minimierung des benötigten Arbeitsspeichers natürlich ohne Parallelisierung, das Prinzip ist jedoch dasselbe. Der für die Blockverarbeitung verantwortliche Code steht in der Elternklasse der Importroutinen und auf diese Weise wird die Komplexität dieses Vorganges verschleiert. Bei der Programmierung neuer Importroutinen müssen, sofern die Blockverarbeitung aktiviert werden soll, nur ein paar andere Methoden der Elternklasse überschrieben werden. Meine Aufgabe bestand darin den bestehenden TerraSAR-X Import umzuschreiben, sodass die Daten mithilfe der Blockverarbeitung eingelesen werden. Um dies zu bewerkstelligen müssen drei Methoden der Elternklasse, der Importroutinen, überschrieben werden. Zum einen müssen zwei Methoden, die jeweils die Bilddaten und die Metadaten zurückgeben, überschrieben werden. Des Weiteren benötigt der Blockimport die Funktion, welche die Maße des Bildes angibt. Dies Information wird benötigt um zu berechnen wie der Datensatz in Böcke unterteilt werden kann. Diese Daten den Methoden bereitzustellen geht am besten die schon zuvor erwähnte Bibliothek, deshalb war die Suche nach der richtigen Dokumentation der Bibliothek am aufwendigsten. Im Unterschied zu der bestehenden Methode des TerraSAR-X, welche die Bilddaten zurück gibt, muss die neue Methode für den Blockimport die Parameter, welche angeben welcher 'Block' gerade ausgelesen werden soll, berücksichtigen und nur den spezifizierten Ausschnitt der Bilddaten zurückgeben. Diese Modifikation des Importes ermöglicht das Einlesen von TerraSAR-X Daten ohne hohe Beanspruchung des

Arbeitsspeichers.

4 Fazit

Meine erste Praxisphase im Institut für Hochfrequenztechnik und Radarsysteme in Oberpfaffenhofen kann ich mit vielen positiven und interessanten Eindrücken abschließen. Ich konnte Vieles, was ich im Verlauf der Praxisphase gelernt habe, wie zum Beispiel über die Radarsysteme und die Programmierung in Python in die Bearbeitung meiner Aufgaben mit einbringen. Zudem habe ich viele Einblicke in die Abläufe in einem wissenschaftlichen Institut erhalten, welche ich sehr interessant finde. Des Weiteren habe ich auch den Ablauf der Softwareentwicklung kennen gelernt und bin auf Probleme gestoßen, wie zum Beispiel die Kommunikation zwischen Anwender und Programmierer. Diese Erfahrungen waren lehrreich und diese Dinge kann ich in der nächsten Praxisphase versuchen gezielt zu berücksichtigen. Zu dem konnte ich meine Aufgabenstellungen umsetzen und feststellen, dass meine Modifizierungen an dem Programm PyRAT Verwendung gefunden haben.

Das Gelernte aus den Vorlesungen habe ich teilweise als nützlich empfunden, jedoch waren nicht alle Vorlesungen direkt auf die Praxis anwendbar. Dies durch den Umstand begründbar, dass ich auch erst ein Semester an Vorlesungen hatte. Andererseits haben viele theoretische Grundlagen, die in den Vorlesungen vermittelt wurden, als Grundlage zum Aufbauen von eigenem Wissen, welches anhand der speziellen Situation notwendig war, geholfen.

Auch abgesehen von neuem Wissen, ist diese Phase aufgrund der abgeschlossenen Aufgaben meiner Meinung nach als erfolgreich zu bewerten.

Teil II

Auswahl von Interessengebieten in F-SAR Produkten: Konzeption und Implementierung in Python

1 Einleitung

Der Bericht beschäftigt sich mit der Erstellung eines Zuschneidewerkzeugs für Datensätze des F-SAR Sensors des Deutschen Zentrums für Luft- und Raumfahrt. Der F-SAR Sensor ist ein flugzeuggestützter Radarsensor mit synthetischer Apertur. Er wird vom Institut für Hochfrequenztechnik und Radarsysteme eingesetzt um Erbeobachtungsdaten zu sammeln [8]. Mit dem Werkzeug sollen Ausschnitte aus Datensätzen ausgeschnitten und als eigenen Datensätze abgespeichert werden können.

1.1 Motivation

Information über die Erdoberfläche, die mit dem Radarsensor F-SAR des DLR gesammelten wurden, werden in großen Datensätzen zusammengefasst. Die Datensätze als Produkte des Sensors repräsentieren große Gebiete der Erdoberfläche. Wenn jedoch nur kleinere Abschnitte betrachtet werden sollen, ist die Größe der Datensätze unhandlich und erschwert die Weiterverarbeitung der Daten. Deshalb ist ein Werkzeug notwendig, welches es ermöglicht für kleine Interessengebiete eigenständige Datensätze zu erstellen. Dies vereinfacht die Informationsgewinnung aus den Erbeobachtungsdaten [8].

1.2 Aufgabenstellung

Die Aufgabe besteht in der Konzeption und Entwicklung eines Werkzeugs. Es soll in der Lage sein die Daten in den Datensätzen des F-SAR Sensors, ausgehend von einem gewählten Ausschnitt, zuzuschneiden und die Parameterdateien entsprechend anzupassen. Die ausgeschnittenen Daten sollen als neues Produkt abgespeichert werden. Zudem soll eine grafische Benutzeroberfläche bereitgestellt werden, um die Auswahl des Ausschnitts zu vereinfachen. Des Weiteren besteht die Aufgabe darin, die Ergebnisse des Werkzeugs zu prüfen. Die Ergebnisse sollen ohne Einschränkungen wie die Ausgangsdatensätze weiterverarbeitet werden können[8].

2 Theoretische Grundlagen

Im folgenden Kapitel werden für die Programmierung notwendigen Grundlagen zusammengefasst. Als erstes wird der Aufbau der F-SAR Daten beschrieben. Zudem wird auf die verschiedenen Dateitypen eingegangen. Dieses Wissen ist notwendig, da das Werkzeug Datensätze dieses Aufbaus bearbeiten soll. Des Weiteren werden die Grundlagen der Interferometrie erläutert um ein besseres Verständnis für den Aufbau der Datensätze zu bekommen.

2.1 F-SAR Daten

Der F-SAR Sensor wird seit 2007 im DLR eingesetzt und dient zu Erprobung neuer SAR-Techniken. Die daraus hervorgehenden Entwicklungen können zum Beispiel auf Satelliten zum Einsatz kommen [5]. Der F-SAR Sensor liefert die Rohdaten, welche nach der Prozessierung und Bearbeitung ein F-SAR Produkt ergeben. In dieser Arbeit werden die Begriffe Produkt und Datensatz äquivalent verwendet. Der Datensatz ist intern aus einer Ordnerstruktur, welche die verschiedenen Dateien enthält, aufgebaut. Die einzelnen Datensätze werden untereinander nach dem Jahr der Akquisition, dem Namen der Kampagne, der Flugnummer und der Passnummer geordnet. Ein Flug ist vom Start bis zur Landung und ein Pass ist vom

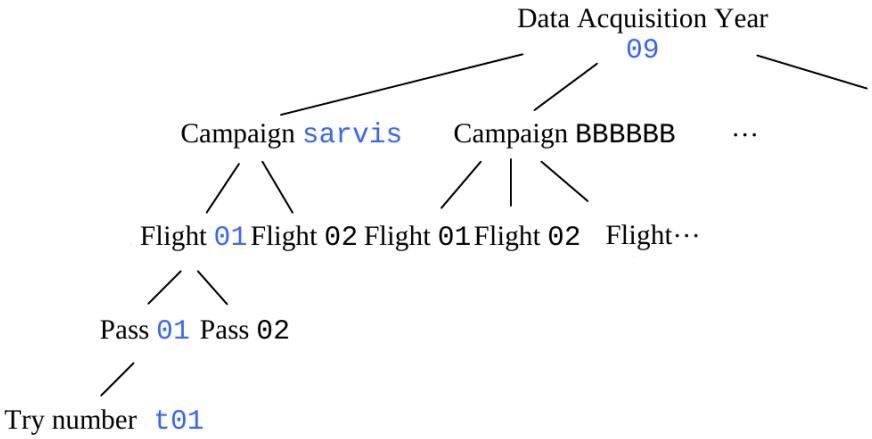

Abbildung 2.1: Struktur der Datensätze

Anschalten bis zum Abschalten des Radarsensors definiert. Also können mehrere Pässe bei einem Flug aufgezeichnet werden. Jeder dieser Pässe kann mehrmals prozessiert worden sein, sodass mehrere Datensätze einem Pass zugeordnet sind. Jeder dieser Versuche wird in einem *Try*-Ordner abgelegt. Jeder dieser Ordner stellt ein Produkt da. Wenn zum Beispiel von einem Produkt ein Ausschnitt ausgeschnitten bzw. kopiert wird, sollte das neue Produkt als neuer Versuch in einem neuen *Try*-Ordner neben dem bestehenden Produkt abgespeichert werden.

In der internen Struktur eines Produktes sind die Order bzw. Teilprodukte *RGI* und *INF* wichtig. Ein Datensatz enthält unter Umständen noch weitere Daten, jedoch bilden diese beiden Teilprodukte die Basis und müssen beim Ausschneiden beachtet werden.

Das Teilprodukt *radar geometry image product* im RGI-Ordner enthält alle Bilddaten sowie zusätzlich Parameterdateien, Quicklooks und weitere Hilfsdateien.

Die Bilddaten in den einzelnen Dateien eines RGI-Produktes beziehen sich alle auf die gleiche Szene. Das bedeutet, dass die Pixel gleicher Position in den jeweiligen Dateien die gleiche Stelle in der Szene darstellen. Die Dateien mit Bilddaten unterscheiden sich untereinander in der Art wie die Werte der Pixel gemessen wurden. Zum Beispiel gibt es Dateien mit absoluten oder komplexen Werten oder mit nur bestimmten Polarisationen. Beim Zuschneiden der verschiedenen Dateien müssen diese Unterschiede nicht beachtet werden, da die Werte nicht verändert werden müssen.

Die Prozessparameter werden im XML Format gespeichert. Sie stellen zusätzliche Informationen für die Prozessierung und weitere Verarbeitung dar.

Das INF-Produkt ist das Ergebnis der Interferometrie, die im nächsten Abschnitt beschrieben wird. Das Interferometrie Produkt ist in verschiedene Dateien, vergleichbar mit dem RGI-Produkt unterteilt.

2.2 Interferometrie

Mithilfe der SAR Interferometrie kann die Höhe der Ziele mit hoher Genauigkeit bestimmt werden. Deshalb werden Höhenmodelle mit dieser Methode erstellt. Mit normalen Radarmessungen lässt sich zwar der Abstand vom Sensor zum Ziel durch Messung der Signallaufzeit bestimmen, jedoch reicht dies nicht aus, um die Geländehöhe zu ermitteln. Bei der Interferometrie werden die Phasen von Messungen aus zwei geringfügig unterschiedlichen Positionen heraus betrachtet. Der Vergleich der Phasen führt zu noch besseren Auflösungen.[7]. Durch

die versetzte Sensorposition und deshalb geringfügig unterschiedliche Entfernung lässt sich die genaue Position des Ziels bestimmen. Die unterschiedlichen Positionen der beiden

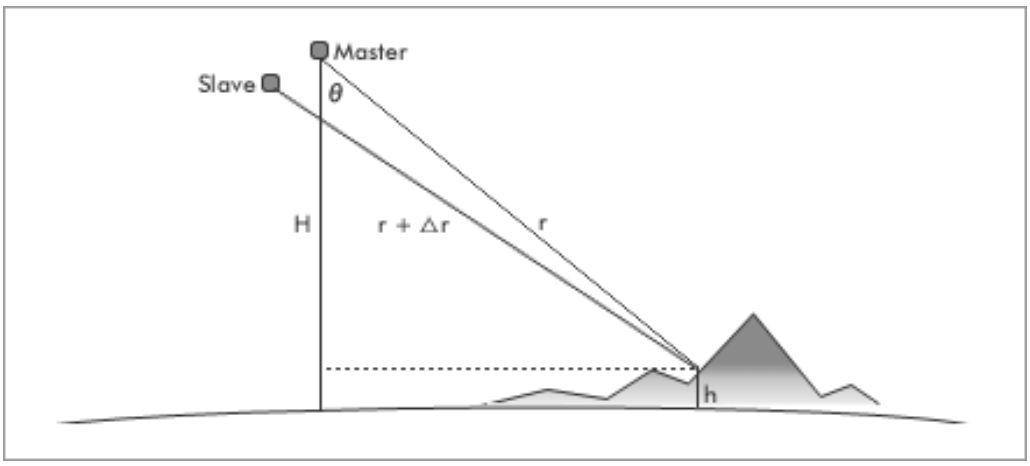

Abbildung 2.2: Interferometrische Messung mit Master und Slave [1]

Messungen können entweder durch zwei Sensoren, die zu der Szene räumlich versetzt sind, oder zeitlich versetztes Überfliegen der Szene mit einem Sensor erreicht werden.

Bei der letzten Variante werden zwei Bilddatensätze mit derselben abgebildeten Szene verwendet. Die empfangenen Werte in den einzelnen Pixeln sind in der komplexen Form abgelegt, sodass die Phasen in den einzelnen Pixeln verglichen und ein Höhenmodell erstellt werden kann [7].

Das Ergebnis dieser Methode ist das INF-Produkt. Es wird aus zwei RGI-Produkten erstellt. Die beiden Ausgangsdatensätze werden als Master und Slave bezeichnet. Der Master bestimmt die Größe und der Slave muss entsprechend angepasst werden, um die Voraussetzung zu erfüllen, dass beide Datensätze den gleichen Teil der Szene abbilden. Die INF-Produkte werden immer dem Slave-Datensatz hinzugefügt. Deshalb sollte die Möglichkeit bestehen das INF-Produkt (im Slave Verzeichnis lokalisiert) ebenfalls anzupassen, wenn dessen Master zugeschnitten wurde.

3 Ausgangssituation

Um bestimmte Ausschnitte aus den Daten des F-SAR Sensors zu erhalten, existieren schon verschiedene Möglichkeiten. Bei der Prozessierung kann auf der Basis der Rohdaten der gewünschte Ausschnitt prozessiert werden. Es wird also nicht auf ein schon vorhandenes und prozessiertes Produkt zurückgegriffen, sondern es werden nur die Rohdaten verwendet. Jedoch benötigt dieses Methode die Rohdaten des F-SAR Sensors und stellt zusätzlichen Rechenaufwand dar. Zudem setzt die Prozessierung spezielle Software voraus und ist deshalb nicht überall möglich.

Eine weitere Möglichkeit ist das manuelle Zuschneiden eines schon prozessierten Datensatzes. Für die verwendeten Datenformate existieren Methoden, welche zum Beispiel in Python implementiert sind und die einzelnen Dateien jeweils einlesen und bearbeiten können. Diese Methode eignet sich zum Bearbeiten von einzelnen Dateien. Ein F-SAR Produkt besteht jedoch aus vielen Dateien, daher ist der Aufwand zu groß, jede Datei des Datensatzes manuell mit den verfügbaren Methoden zuzuschneiden. Des Weiteren erfordert diese Möglichkeit eine genaue Kenntnis vom Aufbau des Produktes und der verschiedenen Eigenschaften der Dateitypen.

Bei dem Stand der Technik ist es schwierig aus einem bestehenden F-SAR Produkt einen Ausschnitt zu erhalten, weil die verfügbaren Bearbeitungsmethoden nicht automatisiert sind und die Parameterdateien manuell angepasst werden müssten.

4 Implementierung

Nachdem die notwendigen Grundlagen beschrieben wurden, beschäftigt sich dieses Kapitel mit der Umsetzung der Aufgabe. Im Anschluss an einen Überblick über das Vorgehen und die Programmstruktur werden die zu Grunde liegenden Konzepte erläutert. Die Problemstellung lässt sich in zwei Teilprobleme unterteilen. Der erste Punkt ist die Kernfunktionalität, welche für die Datenverarbeitung verantwortlich ist. Das andere Problem ist die Ansteuerung dieser Funktionalität durch den Benutzer bzw. die Bereitstellung einer möglichst benutzerfreundlichen Schnittstelle. Es werden jeweils teils mehrere Ansätze vorgestellt und die letztendliche Lösung begründet. Zudem werden weitere Ansätze für Teilprobleme innerhalb der Konzepte genauer dargestellt. Die Implementierung besteht aus der Automatisierung der vorhandenen Zuschneide Funktionen, aus zusätzlichen Zuschneide Funktionen für verschiedene Dateitypen und aus den Benutzerschnittstellen.

4.1 Vorgehen und Struktur

Das zu implementierende Tool wurde in Python geschrieben, da diese Programmiersprache im Institut standardmäßig verwendet wird und so schon bestehende Funktionalitäten übernommen werden konnten. Zur Entwicklung wurde die integrierte Entwicklungsumgebung PyCharm verwendet.

Das so genannte *crop tool* ist in mehrere Module aufgeteilt. Es gibt jeweils ein Modul für die Zuschneidefunktionalität, die Schnittstellen für die Benutzer und eine Sammlung von zusätzlichen Funktionen. Die Zusammenfassung der zusätzlichen Methoden enthält u.a. Ein- und Auslesemethoden, welche nur in den Softwarebibliotheken des Institutes vorhanden und schon in Python implementiert sind. Dies stellt sicher, dass das Tool auch außerhalb der Softwareumgebungen des Institutes verwendet werden kann. Zudem ist es sinnvoll, möglichst wenige Abhängigkeiten an zusätzliche Softwarebibliotheken einzubauen. Des Weiteren existiert eine *config*-Datei in welcher zusätzliche Parameter verändert werden können.

Die genannten Teilprobleme können unabhängig voneinander betrachtet werden. Die Schnittstelle für den Benutzer soll vor allem den gewünschten Ausschnitt abfragen. Also welcher Teil des Datensatzes ausgeschnitten werden soll. Für die Auswahl eignet sich eine grafische

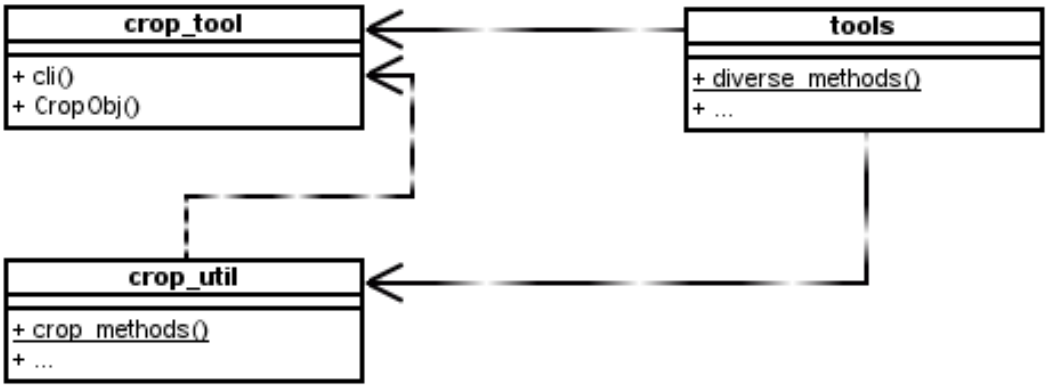

Abbildung 4.1: Klassendiagramm des Werkzeugs

Darstellung des Radarbildes, damit der Benutzer mit der Maus den gewünschten Ausschnitt markieren kann. Zudem werden weitere Angaben, welcher Datensatz zugeschnitten und wohin das neue Produkt anschließend gespeichert werden soll, benötigt. Diese Angaben werden dann an die Ausschneidefunktion weitergegeben. Diese schneidet alle Dateien zu und legt sie in einem neuen Verzeichnis ab.

4.2 Aufbau der Kernfunktionalität

Die Anforderung für diesen Abschnitt ist die Fähigkeit, einen Datensatz mithilfe von gegebenen Parametern zuzuschneiden. Das Konzept besteht darin, die Methoden, welche die verschiedenen Dateitypen eines F-SAR Produktes zuschneiden können, in einem gesonderten Modul bereit zu stellen. Dieses Modul enthält ausschließlich Zuschneidefunktionalitäten für F-SAR Produkte. Diese Methoden können unabhängig voneinander jeweils einen Dateityp bearbeiten. Die verschiedenen Typen von Dateien müssen jeweils unterschiedlich behandelt werden. Entweder sind sie direkt vom Ausschneiden betroffen, weil sie Bilddaten beinhalten, oder sie enthalten Parameter, die entsprechen angepasst werden müssen.

Des Weiteren enthält das Modul übergeordnete Methoden, welche diese Basis-Methoden zusammenfassen und dafür bestimmt sind, Teile eines Produktes auszuschneiden. Zum Beispiel wird eine Methode bereitgestellt, die den RGI Teil eines F-SAR Produktes entsprechend des ausgewählten Ausschnitts anpassen kann. In diesem Kontext bedeutet ausschneiden

einen Ausschnitt aus einem bestehenden Datensatz zu kopieren und anschließend als ein neues Produkt abzuspeichern. Diese Methoden werden wiederum zusammengefasst, um eine Methode bereitzustellen, welche einen kompletten Datensatz zuschneiden kann. Diese Strukturierung ermöglicht die Verwendung der Methoden in einem anderen Kontext.

Die Methoden dieses Moduls folgen alle einem gemeinsamen Schema. Zuerst wird eine Datei eingelesen. Entweder wird nur der benötigte Teil geladen oder im Falle von Parameterdateien die gesamte Datei geladen und anschließend verändert, um die Datei am Ende in ein neues Verzeichnis zu kopieren. Für das Einlesen und Schreiben der Dateien werden entweder interne Bibliotheken des Institutes verwendet oder es genügen die eingebauten Lese- und Schreibfunktionen von Python. Daher besteht die Funktionalität zum Einen darin, die Dateien entsprechend des Ausschnitts zu verändern. Zum Anderen ist die Aufgabe der übergeordneten Funktionen die richtigen Dateien aus dem Dateisystem ihren zugehörigen Funktionen zuzuweisen und das Ergebnis an der richtigen Stelle in der Produktstruktur zu speichern.

4.2.1 Sicherheitsmechanismus gegen das Überschreiben

Der Schutz der bestehenden Daten ist ein wichtiger Aspekt beim Programmieren einer Datenverarbeitung, sodass nur die vorgesehenen Daten verändert werden. Im vorliegenden Fall dürfen keine bestehenden Datensätze verändert werden. Sonst wären diese Datensätze nicht vollständig und damit unbrauchbar. Diese Vorgabe wird unter der Voraussetzung, dass der Code selbst fehlerfrei ist und der alle Fehler des Benutzers vom Code abgefangen werden, erfüllt. Jedoch können Programmierfehler nie ausgeschlossen werden. Dies gilt vor allem in diesem Fall, da die Hauptaufgabe des Werkzeugs im Ein- und Auslesen verschiedener Dateien liegt. Deshalb könnte jeder Fehler im Code dazu führen, dass eine Datei unbeabsichtigt überschrieben wird. Zudem sind F-SAR Produkte nicht ohne Umstände zu ersetzen und weitere Sicherungskopien mit hohem Aufwand, aufgrund der großen Datenmenge, verbunden. Deshalb empfiehlt es sich Sicherheitsvorkehrungen zu treffen, um zu verhindern, dass Dateien aus Versehen überschrieben oder verändert werden. Die umgesetzte Lösung verhindert grundsätzlich das Überschreiben von schon bestehenden Dateien. Für die Umsetzung wurde der Umstand ausgenutzt, dass alle Schreibroutinen, die in diesem Tool Verwendung finden, letztendlich auf eine eingebaute Standardfunktion von Python zurückgreifen.

Der Ansatz besteht darin, die eingebaute *open()-Funktion* zu überschreiben und mit zusätzli-

chem Code immer einen Fehler zu werfen, sofern eine schon bestehende Datei im Schreibmodus geöffnet wird. Somit wird verhindert, dass diese Datei eventuell überschrieben wird. Die Funktion wird mit einer zusätzlichen Abfrage, ob die Datei schon existiert und ob sie im Schreibmodus geöffnet wurde, erweitert. Sobald die neue *open()-Funktion* in ein Modul importiert wird, überdeckt sie die standardmäßige Funktion, da die Namen identisch sind. Auf diese Art und Weise wird im ganzen Modul verhindert, dass Dateien überschrieben werden können. Die Nachteile dieses Ansatzes sind, dass nicht vollständig verhindert werden kann, dass die ursprüngliche *open()-Funktion* aufgerufen wird. Zudem ist das Überschreiben von Standard-Funktionen, zumindest in den offiziellen Python *code style* Regeln, untersagt. Eine Alternative, zu dem aufgezeigten Weg, ist das Entziehen des Schreibrechts auf bestimmte Dateien durch das Betriebssystem. Jedoch würde diese Vorkehrung erst auf Ebene des Betriebssystem greifen und ist deshalb umständlicher, da sie zusätzliche Interaktion vom Benutzer erfordert oder das Werkzeug selbständig die Rechte verändern müsste.

4.2.2 Zusammenfassung gleicher Funktionalitäten

Der Ablauf beim Zuschneiden der verschiedenen Dateitypen weißt viele Parallelen auf. Es müssen immer zuerst die Dateien vom bestehenden Datensatz gesucht und anschließend die neuen Orte für die ausgeschnittenen Dateien bestimmt werden. Die einzigen Unterschiede in der Verarbeitung der verschiedenen Typen von Dateien ist die Art und Weise, wie zugeschnitten wird und wo die Dateien in der Dateistruktur gespeichert werden. Deshalb bietet es sich an, diese Vorgänge in einer zusätzlichen Funktion auszulagern. Diese Lösung führt zu weniger und damit auch übersichtlicherem Quellcode. Die Nachteile sind eine komplexere Programmstruktur, da die zusätzliche Funktion Vorgänge teilweise ausblendet. Die Zusammenhänge sind daher nicht so schnell ersichtlich wie im Vergleich zu der Alternative. Die Alternative ist für jeden Datentyp jeweils eine eigene Funktionalität zu implementieren, die dafür zuständig ist die richtigen Dateien für die Verarbeitung zu finden und für diese anschließend die richtige Stelle im Dateisystem festlegt. Leider wäre ein derartiger dezentraler Ansatz fehleranfälliger, da letztendlich mehr Code geschrieben werden müsste. Der gewählte Ansatz verwendet eine einzige Methode für die Aufgabe, welche daher zentral im Quellcode definiert ist. Die Anpassung an die jeweiligen Dateitypen wird über Parameter gesteuert. Veränderungen an der allgemeinen Verarbeitung der Dateien müssen so nur an der zentralen Funktion vorgenommen werden.

4.3 Aufbau der Schnittstellen für Benutzer

Damit im Falle der Anwendung eine einfache Benutzung möglich ist, muss eine Schnittstelle für den Benutzer bereitgestellt werden. Diese sollte einen einfachen Zugriff auf die zuvor vorgestellten Funktionalitäten gewährleisten. Die vom Benutzer abzufragenden Parameter können in zwei Gruppen eingeteilt werden. Einerseits werden Informationen benötigt welcher Datensatz als Vorlage genommen und wo der ausgeschnittene Datensatz abgespeichert werden soll. Diese Informationen kann der Benutzer am einfachsten in der Kommandozeile tätigen. Auf der anderen Seite werden die Koordinaten des gewünschten Ausschnitts benötigt. Bei dieser Auswahl ist es notwendig eine geeignete grafische Oberfläche bereit zu stellen. Diese sollte den Datensatz bzw. einen exemplarischen Teil des Datensatzes anzeigen können, damit der Benutzer nur noch den gewünschten Bereich mit der Maus angeben muss.

Die Anforderung an das Werkzeug ist vor allem eine leichte Bedienbarkeit, die kein Vorwissen über dessen Bedienung voraussetzt. Eine geringere Priorität haben zusätzliche Features in der Bedienung wie zum Beispiel eine einheitliche Abfrage aller Parameter. In der vorgenommenen Implementierung werden Parameter nur teilweise über eine grafische Schnittstelle abgefragt. Ein einheitliches Konzept würde zwar mehr Komfort bei der Bedienung bedeuten, jedoch ist nur von Anwendungsszenarien auszugehen in denen der Benutzer das Werkzeug nur in geringem Umfang verwenden, sodass eine aufwendigere Schnittstelle nicht begründbar ist. Die Verständlichkeit der Bedienung ist aber wichtig, damit sich Benutzer bei seltenem Gebrauch nicht jedes mal von neuem in der Anwendung zurechtfinden müssen.

4.3.1 Grafische Benutzerschnittstelle

Um den richtigen Ausschnitt auswählen zu können, muss zuerst das gesamte Radarbild angezeigt werden. Zur Visualisierung eines Datensatzes kann entweder ein schon fertiges Bild (ein Quicklook) angezeigt werden oder aus den Radardaten des Datensatzes selber ein Bild erzeugt werden. Es ist einfacher das schon fertige Quicklook zu laden, da so ein arbeitsaufwendiger Schritt entfällt. Weiterhin sollte der ausgewählte Ausschnitt nach der Auswahl neu angezeigt werden und eventuell wieder ausgeschnitten werden können.

Der erste Ansatz besteht in der Programmierung eines Anzeigeprogramms mithilfe der Bibliothek Qt. Diese Sammlung von Klassen für die Programmierung von grafischen Benutzeroberflächen stellt alle Grundbausteine zur Verfügung. Die Teile müssen aber noch miteinander

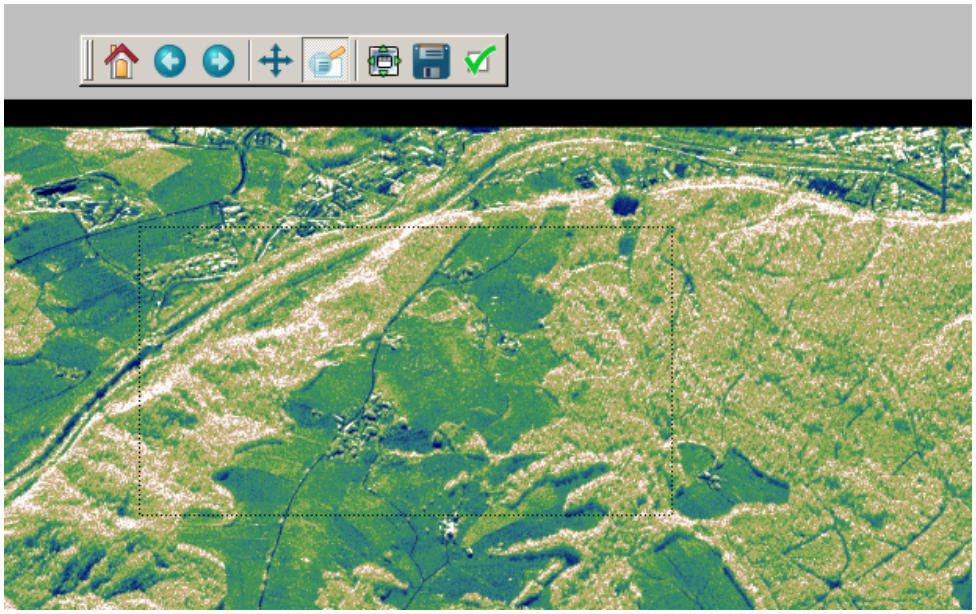

Abbildung 4.2: Screenshot: Auswahl eines Ausschnitts mit der GUI

verbunden und spezifischere Funktionen müssen selbst implementiert werden. Ein Beispiel für nicht vorhandene Funktionalitäten ist die Berechnung der Koordinaten bezogen auf das Ausgangsbild, wenn aus Ausschnitten wiederum ausgeschnitten wurde. Ein weiteres Beispiel ist, dass das Verhalten bei Veränderung der Fenstergröße usw. noch nicht programmiert ist. Im Gegenzug bietet diese Bibliothek alle Freiheiten um individuelle Oberflächen zu erstellen.

Der andere Ansatz besteht darin, das Anzeigeprogramm der Python Bibliothek *mathplotlib* zu verwenden. Dieses Programm bietet eine grafische Oberfläche um geplottete Graphen usw. anzuzeigen. Des Weiteren bietet es unter anderem schon Funktionen an, um Ausschnitte auszuschneiden. Zudem werden dem Nutzer erweiterte Funktionen bereitgestellt, wie zum Beispiel die Möglichkeit Bearbeitungsschritte zurückzunehmen usw. Um dieses Programm für das Auswählen in Radarbildern zu verwenden, müssen die Parameter aus dem Programm ausgelesen werden. Dies ist einfach möglich obwohl keine direkte Schnittstelle zu den internen Daten bereitgestellt wird, da dieses Anzeigeprogramm in Python geschrieben wurde und der daher der Quellcode zur Verfügung steht.

Aufgrund des geringeren Aufwands und den zusätzlichen Features ist die Verwendung von *mathplotlib* geeigneter für die Bereitstellung einer Benutzeroberfläche. Zumal die Vorteile von

Qt, wie zum Beispiel die größere Anzahl von Möglichkeiten, keine Verwendung finden, weil keine besonderen Funktionen gebraucht werden. Die Abhängigkeiten der beiden Lösungen an Bibliotheken sind als ungefähr gleich zu bewerten. Qt wird letztendlich in beiden Ansätzen verwendet (*mathplotlib* verwendet es intern), daher wird die größte Abhängigkeit in beiden Ansätzen benötigt.

4.3.2 Text-basierte Benutzerschnittstelle

Die übrigen Parameter werden über die Kommandozeile abgefragt. Zuerst wird die Angabe, welcher Datensatz zugeschnitten werden soll, benötigt. Anschließend wird die grafische Oberfläche für die Auswahl des Ausschnitts geöffnet oder alternativ die Möglichkeit angeboten die Koordinaten des gewünschten Ausschnitts manuell einzugeben.

Um die Verwendung der Zuschneidefunktionen einfacher und bequemer zu gestalten, existiert eine Schnittstelle, die den objektorientierten Ansatz verfolgt. Die Zuschneidefunktion kann auch ohne Zwischenfunktion aufgerufen werden. Dies erfordert jedoch die Eingabe aller Parameter, ohne eine Prüfung auf Richtigkeit, auf einmal. Des Weiteren ist es nicht möglich direkt bei dem Methodenaufruf die grafische Benutzeroberfläche zur Auswahl zu verwenden. Die Klasse, welche als erste Schnittstelle für die Zuschneidefunktionen dient, sammelt zuerst alle benötigten Parameter. Diese Informationen werden in den Instanzvariablen zwischengespeichert und durch zusätzliche Methoden auf Richtigkeit und Vollständigkeit überprüft. Zudem bietet sie die Möglichkeit die grafische Oberfläche durch einen Methodenaufruf zu öffnen. Wenn alle notwendigen Informationen an das Objekt weitergegeben wurden, kann der Zuschneide Vorgang gestartet werden und die Instanz gibt alle Werte an die Zuschneidefunktion weiter. Diese Schnittstelle ermöglicht eine einfachere und sicherere Bedienung und kann vom Benutzer über die interaktive Python Konsole verwendet werden. Jedoch ist die Voraussetzung für die Verwendung die Kenntnis der Bezeichnungen für die verschiedenen Parameter. Zudem liefert diese Schnittstelle dem Benutzer nicht genügend Informationen und lässt ihm zu viele Freiheiten, sodass eine Benutzung ohne Kenntnis der Schnittstelle nicht möglich ist. Daher entspricht sie noch nicht den Anforderungen. Der Ansatz eines Kommandozeilen Dialoges, welcher den Nutzer sequenziell alle nötigen Parameter abfragt und automatisch die grafische Oberfläche öffnet, entspricht den Anforderungen. Zudem werden alle benötigten Eingaben genau definiert, sodass kein besonderes Vorwissen zum Zuschneiden eines Datensatzes erforderlich ist. Die Methode, welche diesen Ansatz umsetzt, verwendet wiederum ein Objekt der zuvor beschriebenen Schnittstelle. Die Benutzerschnittstelle kann

von der Kommandozeile aus gestartet werden und fragt alle notwendigen Parameter ab. Wenn sich der Benutzer entscheidet den Ausschnitt mit grafischer Oberfläche auszuwählen, öffnet sich die grafische Schnittstelle. Sobald der Benutzer seine Eingaben bestätigt, werden diese zum Objekt weitergegeben, welches die Zuschneide Methode aufruft.

4.4 Evaluation der Ausgabe

Um sicherzustellen, dass mit den ausgeschnittenen Produkten gleichermaßen weitergearbeitet werden kann, müssen die Ergebnisse des Werkzeugs geprüft werden. Für die Weiterverarbeitung ist es einerseits wichtig, dass sich die ausgeschnittenen Bilddaten alle auf den gleichen Abschnitt beziehen. Auf der anderen Seite müssen die Parameter aus den Hilfsdateien dem Ausschnitt entsprechend angepasst werden. Aufgrund des großen Umfangs der Datensätze und abstrakter Parameter ist es am einfachsten die Datensätze weiterzuverarbeiten und anschließend die Ergebnisse zu betrachten.

4.4.1 Vergleich der INF-Produkte

Die RGI-Teilprodukte können auf Vollständigkeit und Korrektheit geprüft werden, indem aus zwei ausgeschnittenen RGI-Produkten ein neues INF Produkt erzeugt wird. Dieses INF Produkt kann dann mit einem schon bestehenden INF Produkt verglichen werden. Zudem wird auf diese Weise getestet ob sich die Zuschnitte überhaupt weiterverarbeiten lassen. Dies wäre nicht möglich, wenn z.B. bestimmte Parameter fehlen.

Zuerst wird das Interferometrie-Produkt, welches sich auf die komplette Szene bezieht, zugeschnitten. Aus den beiden RGI-Produkte, woraus das ursprüngliche INF-Produkt entstanden ist (also Master und Slave), wird ebenfalls der gleiche Ausschnitt mit dem Werkzeug heraus geschnitten. Aus den beiden neu erstellten Produkten wird anschließend ein neues INF-Produkt als Vergleich zu dem zuvor ausgeschnittenen INF-Produkt berechnet. Wenn die beiden INF-Produkte identisch sind, hat das *crop tool* keinen Fehler gemacht. Das ersten INF-Produkt wurde zuerst aus den RGI-Datensätzen erzeugt und dann zugeschnitten. Das zweite INF-Produkt wurde aus schon zugeschnittenen RGI-Produkten erstellt.

4.5 Einschränkungen des Werkzeugs

Dieser Abschnitt beschreibt die Einschränkungen des *crop tool*s in seiner Einsetzbarkeit zum Zuschneiden von Datensätzen. Eine wichtige Voraussetzung für das Zuschneiden der Dateien ist vor allem das Finden der jeweiligen Dateien. Deshalb besteht die Voraussetzung, dass die Produkte immer nach der Definition aufgebaut und dass die Dateinamen unverändert sind, da das Werkzeug ausschließlich die Dateien nach den Namen aussucht.

Des Weiteren kann es vorkommen, dass in einem Datensatz die Daten von mehreren Frequenzbändern gespeichert sind. Da sich die Größe der abgebildeten Szenen unter verschiedenen Bändern unterscheiden kann, wird das *crop tool* nicht in der Lage sein, alle Daten korrekt zuzuschneiden. Die Anwendung ist somit auf Datensätze beschränkt, deren Bilddaten alle die gleiche Größe besitzen.

5 Ergebnis

Das in Python implementierte *crop tool* ermöglicht das Ausschneiden von Ausschnitten aus F-SAR Produkten. Diese werden anschließend als vollständige Produkte abgespeichert. Das Werkzeug wird über die Kommandozeile gesteuert. Zudem besteht die Möglichkeit eine grafische Benutzeroberfläche für die Auswahl des gewünschten Ausschnitts zu verwenden. Im Anschluss an die Eingabe des Benutzers aller benötigten Parameter werden die notwendigen Dateien automatisch aus dem Datensatz kopiert und entsprechend angepasst, um anschließend ein neues Produkt zu ergeben. Die Ergebnisse des Tools wurden durch ihre Weiterverarbeitung auf Vollständigkeit und Richtigkeit geprüft. Letztendlich ist das *crop tool* in der Lage vollständige F-SAR Datensätze aus Bestehenden fehlerfrei auszuschneiden.

Literaturverzeichnis

[1] Interferometry Processing, PALSAR Project. http://gds.palsar.ersdac.jspacesystems.or.jp/e/data/img/photo_6_2_2.gif.

[2] Website: E-SAR - Das flugzeuggetragene SAR-System des DLR, 11 March 2015. http://www.dlr.de/hr/desktopdefault.aspx/tabid-2326/3776_read-5679/.

[3] Website: KOMPSAT-5 (Korea Multi-Purpose Satellite-5) Program, 11 March 2015. http://eng.kari.re.kr/sub01010204.

[4] Website: TerraSAR-X - Deutschlands Radar-Auge im All, 11 March 2015. http://www.dlr.de/eo/desktopdefault.aspx/tabid-5725/9296_read-15979/.

[5] Marc Jäger, Muriel Pinheiro, Octavio Ponce, Andreas Reigber, and Rolf Scheiber. A Survey of Novel Airborne SAR Signal Processing Techniques and Applications for DLR's F-SAR Sensor. International Radar Symposium, Dresden, 24 June 2015.

[6] Helmut Klausing and Wolfgang Holpp. *Radar mit realer und synthetischer Apertur*. Walter de Gruyter, Berlin, 1999.

[7] Alberto Moreira, Pau Prats-Iraola, Marwan Younis, Gerhard Krieger, Irena Hajnsek, and Konstantinos Papathanassiou. A Tutorial on Synthetic Aperture Radar. *IEEE Geoscience and Remote Sensing Magazine*, March 2013.

[8] Rolf Scheiber. Anmeldung Projektbericht, Modul T1000 (Praxis 1.+2. Semester), 17 July 2015.

A Anhang

User manual for the crop tool

18.08.2015

Contents

1 Introduction

The *crop tool* can crop products from the F-SAR sensor. It copies a chosen part from an existing product and saves this part as a new product. The result is a new product with the image data from the specified area. Moreover it never touches the exiting products - it only reads their data. The crop tool is able to crop the RGI product and it can also crop the INF products. It is written in python, therefore a python environment is necessary.

This manual is for the user and describes the functionality of the tool. It explains all important things for the usage, but it does not deliver any knowledge about F-SAR products. The first section explains the short setup. The next section is a step-by-step guide for the quick start. The last section gives a more detailed look at the tool and its usage.

2 Setup

The tool does not require a installation, because it is out of python scripts. It will function out of the box, if all libraries and the python interpreter are available. The tool needs a python interpreter version 3.x. It wont function with a lower version like 2.x. In addition there are several non standard libraries important. The tool was tested in an anaconda environment. Therefore the easiest way to get these libraries is to install the anaconda distribution. This should include all needed libraries. Some libraries are not necessary, but recommended for the graphical interface.

3 Quick start guide

This section should provide a quick guide how to use the tool. It shows step-by-step an example how to crop a product. If you searching for more detailed information, see the next section.

At first go in the directory from the crop tool. You should see several python scripts. You can start the tool by running your python interpreter with the *fsar_crop.py* file as argument as shown below. If you got an error, the tool properly did not found all necessary libraries.

```
1 python3 fsar_crop.py
```

If it works and the tool started, you will be asked to enter the first parameter. The crop tool reacts to non valid input by returning to the command line. Then you have to restart the tool. You can always leave the tool by typing exit or something obviously wrong.

The first parameter is the path to the data set from which you want to copy a part. This part will be saved later as a new product. You have to type the absolute path to the directory of the existing product. The path should point to a directory with at least an RGI folder. Properly the directory has also other products like the INF or GTC product. The crop tool will only look for the RGI folder in the specified directory.

```
Crop tool for data sets from the F-SAR sensor
Version 0.0, type 'exit' for exit

Please enter the path to the directory of the data set as input:
 e.g. /.../14TMPSAR/FL09/PS04/T02/
?> /absolute/path/to/the/fsar/product/directory
```

Next you can choose the area for the new product. For choosing the right part of the data are thee modes available. Just type '1', '2' or '2' and confirm it with with 'enter'

In mode 1 you can enter the coordinates manually with the command line. The unit from the coordinates is pixel. You have to define the area for the new product. The first two numbers define the start from the area in range and in azimuth. The third and the fourth define the size of the area in range and in azimuth.

```
 Next choose the crop area. You can use ...
   the keyboard, type             '1'
   the GUI with a RAT file, type   '2'
   the GUI with a Quick Look, type '3'
?> 1
  Please enter the coordinates from the crop area (in pixel)
   range_start, azimuth_start, range_size, azimuth_size (the commas are
 necessary)
?> 2000, 5000, 1000, 10000
```

If you want to choose the area with the graphical user interface type '2' or '3'. The difference is that mode 2 loads a **.rat* file for the viewer instead of a quicklook in mode 3. It is easier to work with quicklooks, therefore mode 3 is recommended. Press 'enter' again and the tool will open a window, which shows the image data from the existing product. Use the *zoom*

tool to choose the area. When you are satisfied with the region, just close the window.

```
 Next choose the crop area. You can use ...
   the keyboard, type              '1'
   the GUI with a RAT file, type   '2'
   the GUI with a Quick Look, type '3'
?> 3
  The window will open automatically. You can choose the right area wit
h the zoom tool and go back or forward with the arrows. Just close the
window to confirm your setting.
   press enter to start...
```

If the product inherits several band, you can specify a specific band. For Example type x or X to crop only the X-Band. For cropping all bands type instead *.

```
Now you can specify a frequency band.
Type '*' for all bands or a letter for one specific band, e.g. 'X' for X-Band
  *
```

Next you can decide if you want to change the directory for the new product. The new product will be written in the same directory by default. If you choose a different directory, the hole output will be saved in this one (e.g. RGI products, INF products from slaves). You have to type the path to an existing directory, where the tool can create a new product.

```
 Do you want to change the location for the new product ?
 If not the data set will be written in a separate folder in /absolute/
path/to/the/fsar
 Please type y/n :
?> y
  Please enter the new path for the new product:
?> /absolute/path/to/a/directory
```

In addition you can enter slaves (INF products). They will be cropped to the same size like the master (RGI product). If you did not set a directory in the previous step, the INF products will be saved next to the product from the slave. If you entered a alternative directory instead, the new INF products will be written into this directory. You have to enter the path to the slaves directory like the first parameter. The path should point to a existing directory with an INF product.

The last parameter is the name for the new product. The base folder from the hole product, where the RGI product is located, will get this name. Moreover the name will be used to

```
 Do you want to apply to crop slaves (INF product)? Please type y/n
?> y

 Please enter the path to the slave try directory.
?> /absolute/path/to/a/slave
```

extend all file names from the new product.

```
 Please enter the new name for this product. The name extends the filen
ames from all files and is is used as directory name.
  e.g. T01small ==> /.../T01small/RGI/RGI-SR/slc_14tmpsar0803_Lvh_t01sm
all.rat (an example for a file path)
?> newName
```

Eventually you have to check the parameters and start the crop process.

```
 Do you want this part [2000, 5000, 1000, 10000] from /absolute/path/to
/the/fsar/product written in /absolute/path/to/a/directory
 and in addition this slave(s):
  /absolute/path/to/a/slave

 Please type 'y' to start the crop tool or 'n' for exit
?> y
```

Now you should see that the tool lists the path from all files, which were created. If an error occurs, check the more detailed section. If the process was successful, the log file will be written in the directory, which is specified in the configuration file.

4 Usage of the tool

This section provides more information about the crop tool. It could be divided into issues about the input like the configuration file and the interfaces and into issues about the output like the output itself and the error handling.

4.1 Input

Check the tables 1 and 2 to understand how the tool recognizes files as input. If for example the name of a file in the RGI-SR directory of the product matches one pattern, the file will

be cropped and copied to the new product. Files with names, which do not match to any pattern, wont be add to the new product.

Furthermore please note that it maybe would cause problems if files from different sizes like those from different frequencies are in the same product.

There also exists a configuration file, which is written in python, but it is not any knowledge about python necessary to change it. The configuration file is located by the other python scripts from the crop tool. The first parameter sets the directory for the log files. If you want to get more information about an error, you can set the second parameter *True*. Then you'll get a more detailed error message.

4.2 Command Line Interface

There are two ways to start the crop tool from the command line. The first way is to run it from the shell this was also described in the quick start guide. You have to change the directory to the one with the fsar_crop.py script and then start your python interpreter with the script as argument as you see below.

```
1 python3 fsar_crop.py
```

The second possibility is to start the interactive python console. It is necessary that the files from the tool are in the python path. Therefore you can change the directory to the scripts and start the python interpreter from there.

```
1 python3
```

Now you can import the fsar_crop module.

```
1 import fsar_crop
```

If the import worked, you can start the crop tool as you see below.

```
1 fsar_crop.start()
```

4.3 Graphical User Interface

The graphical user interface should help to choose the right area. Use the *zoom tool* to cut an area. If you confirm by closing the window, you'll choose the area, which was chosen

with the zoom tool the last time. In the most cases it is the area, which is displayed in this moment. You could use the arrows to undo or redo your steps. Furthermore please avoid to use the buttons, which were not mentioned, because that could trigger some unwanted behavior.

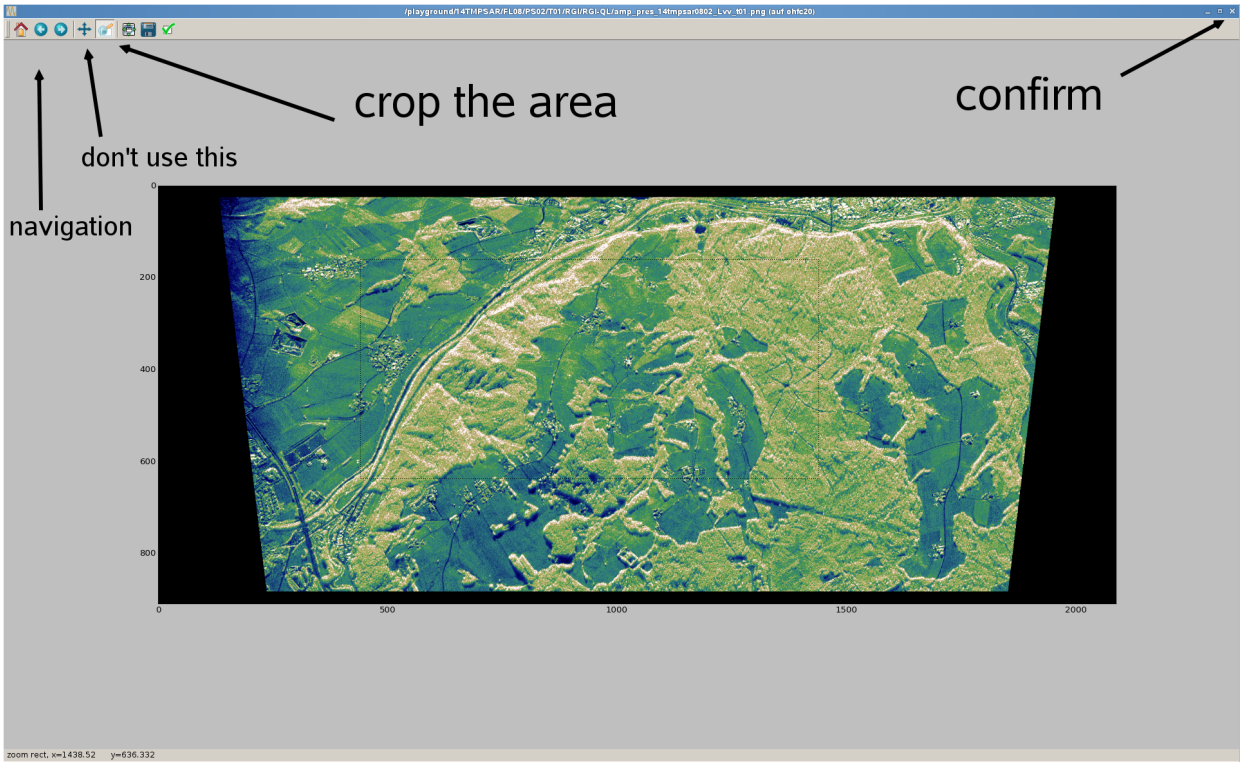

Figure 1: the graphical user interface

4.4 Output

The files for the new product are saved on the paths, which were displayed after the tool started. If you set an alternative directory for the output, all things will be written into this directory. If you did not set this path, each new product will be written in the directory from the originally product.

You find a list with all available file types in table 1 and table 2.

Table 1: list with all types of files from new RGI products

product directory	**file pattern**
RGI-SR	{mask,slc,amp,incidence}_*_*_*.rat
RGI-SR	{mask,slc,amp,incidence}_*_*_*.rat.hdr
RGI-RDP	pp_*_*_*.xml
RGI-TRACK	{reftr,attit,track}_*_*_*_.rat
RGI-TRACK	fixpoint_*_*_*_.rat
RGI-QL	{pol,amp,pauli}_pres_*_*_*.png
RGI-AUX	moco1_*_*_*.rat
RGI-AUX	offnadir_*_*_*.rat
RGI-AUX	slantdem_*_*_*.rat
RGI-AUX	tc_vector_*_*_*.rat

Table 2: list with all types of files from new INF products

product directory	**file pattern**
INF-SR	slc_coreg_*_*_*.rat
INF-SR	kz_*_*_*.rat
INF-SR	coh_*_*_*.rat
INF-SR	pha_fe_*_*_*.rat
INF-RDP	pp_*_*_*.xml
INF-TRACK	{reftr,attit,track}_*_*_*_.rat
INF-TRACK	fixpoint_*_*_*_.rat
INF-QL	{pha,coh}_*_*_*.png

4.5 Errors

The debug variable in the configuration file sets the verbosity in the case of an error. Furthermore don't try to override existing files. The tool will always throw an error.