UNIVERSITÄT OSNABRÜCK

Mobile Mapping
Visualisierung und Management georeferenzierter Bilddaten mit QuantumGIS

BACHELORARBEIT

Universität Osnabrück
Institut für Geoinformatik und Fernerkundung
Fachbereich 6 – Mathematik/Informatik

eingereicht von: Ines Schiller
Matrikelnummer: xxxxxx

Erstprüfer: Prof. Dr.-Ing Manfred Ehlers
Zweitprüfer: Dipl.-Geogr., Dipl.-Umweltwiss. Kai Behncke

IGF

Rheine, 30.11.2010

Danksagung

Während der dreimonatigen Arbeit an diesem Thema erhielt ich eine intensive Betreuung von Seiten der Universität Osnabrück und des Instituts für Geoinformatik und Fernerkundung durch Prof. Dr. Manfred Ehlers und Kai Behncke, Dipl.-Geogr., Dipl.-Umweltwiss., und von Seiten der Firma TopScan (Gesellschaft zur Erfassung topographischer Informationen mbH) durch Jana Gliet und Dr. Joachim Lindenberger. Hiermit bedanke ich mich für ihre fachliche Betreuung, Beratung und ihre Geduld.

Ein weiterer Dank gehört Andreas Huggle für den außerordentlichen und umfangreichen Support bei Fragen und Problemen mit Exiv2. Ich kann mit Überzeugung sagen, dass ich noch nie so einen guten Support erlebt habe.

Weiterhin konnte ich ohne Probleme und kurzfristig in schwierigen Situationen meiner Bachelorarbeit Urlaub nehmen. Daher möchte ich auch dem Zentrum für Informationsmanagement und virtuelle Lehre (virtUOS) der Universität Osnabrück meinen Dank aussprechen.

Ein besonderer Dank gilt meiner Mutti und meinem Vati, die immer ein offenes Ohr für mich hatten, auch über eine Distanz von mehr als 500 km hinweg. Auch mein Freund hat mich stets unterstützt, mich bei Problemen immer wieder aufgebaut und mir viel Halt gegeben.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

**Wer keine Ausdauer hat bei Kleinigkeiten,
dem misslingt der große Plan.**

(Chinesische Weisheit)

Quelle: Simon, H. (2000): Geistreiches für Manager.
Frankfurt/Main: Campus Verlag GmbH.

1 Einleitung

In einigen Praktika bei der Firma TopScan GmbH in Rheine hat die Autorin dieser Arbeit einen Einblick in das Thema Mobil Mapping bekommen. Hierbei wurde ihr der grobe Aufbau des Systems gezeigt und welche Daten erfasst werden. Dabei fiel ihr auf, dass bis dahin die Bilddaten nur selten verwendet wurden. Diese Erfahrungen veranlassten sie, sich in ihrer Bachelorarbeit intensiver mit diesem interessanten Bereich der Geoinformatik und Fernerkundung auseinander zu setzen und gemeinsam mit der Firma TopScan GmbH eine Verwendung für die Bilder zu finden.

Weiterhin beschäftigt sich die Autorin schon seit einiger Zeit mit der Verwaltung von Bildern. Sie ist eine begeisterte Hobby-Fotografin. Im Laufe der Zeit ist eine große Anzahl an Bildern entstanden, die verwaltet werden müssen. Hierbei sind Sortiermöglichkeiten nach dem Erstellungsort oder des Erstellungsdatums hilfreich. Diese Daten werden aus den Bildern ausgelesen.

Diese Erfahrungen werden in diese Arbeit eingebracht und liegen voll und ganz im Interessenbereich der Autorin. Daher wurde das Thema, „Mobil Mapping – Visualisierung und Management georeferenzierter Bilddaten mit QuantumGIS“, gewählt.

1.1 Problemstellung

Mobile Mapping ist zurzeit ein durchaus geläufiger Begriff, auch wenn viele Menschen noch nie davon gehört haben. Diese Methode der Datenaufnahme ist in Deutschland durch Google Street View ins Licht der Öffentlichkeit getreten (vgl. Street View – Weblink; Sieber 2010).

Mobile Mapping ist im Grunde nichts anderes, als das Aufnehmen (Abbilden/Kartieren) der Umgebung von einem sich bewegenden Fahrzeug aus, zum Beispiel die Aufnahme von Bild- und Positionsdaten (vgl. Zlatanova und Li 2008; Tao und Li 2007). Die von einigen Firmen verwendeten Systeme werden auf Autos oder andere fahrbare Untersätze montiert.

Systeme, wie das von TopScan verwendete, speichern Positionsdaten getrennt von den Bilddaten in einer Textdatei ab. Dabei geht der direkte Zusammenhang zwischen den Beiden verloren. Zu dem ist die Darstellung der Bilder mit Bezug auf die Positionsdaten schwierig oder gar nicht umsetzbar (vgl. TopScan - Weblink). Diese Problematik muss in der Arbeit gelöst werden.

Ein weiteres Problem ist die große Menge an Daten. Bei vorherigen Recherchen wurden verschiedene Programme zur Anzeige und Verwaltung von Bildern gefunden. Eine Anzeige der übrigen Daten, wie Positionsdaten, Ausrichtung etc., war mit diesen Programmen aber nicht möglich.

1.2 Zielsetzung

Wie zuvor erläutert, wird durch Fahrten mit Mobile Mapping-Systemen eine Vielzahl an Bild- und Positionsdaten erfasst. Für die Verwaltung und Verarbeitung müssen geeignete Datenfilter und Regeln zur Auswahl der zu bearbeitenden Daten vereinbart werden, dieses ist das primäre Ziel der Arbeit.

Ein weiterer Punkt ist die Visualisierung. Ziel hier ist eine Darstellung, die sowohl die Bilddaten, als auch gleichzeitig die Positionsdaten und den gefahrenen Fahrtweg darstellen kann. Es soll möglich sein, Kartenausschnitte mit der Anzeige der Bild- und Positionsdaten zu verbinden.

Die Visualisierung sollte, so weit möglich, mit Hilfe von Open Source-Produkten umgesetzt werden. Die Nutzung von proprietärer Software ist vorab dennoch nicht gänzlich ausgeschlossen. Daher sind Vergleiche verschiedener Systeme für diese Arbeit notwendig.

Zusammengefasst sind die Hauptziele dieser Arbeit die Zusammenführung der getrennt erfassten Bild- und Positionsdaten, die Verarbeitung und Verwaltung von großen Bildmengen mit Blickpunkt auf eine geeignete Darstellung, sowie die Umsetzung mit Open Source-Lösungen.

1.3 Aufbau und Nutzen

Die Arbeit beinhaltet grundlegend drei Bereiche (Abbildung 1). Zum einen geht es darum, die Bilddaten mit den Positionsdaten zu verbinden. Die verarbeiteten Bilder können in bestimmten Bildbetrachtungs- und Verwaltungsprogrammen angezeigt und zusätzlich die Positionsdaten sichtbar gemacht werden (Beispiel: Adobe Bridge und IrfanView).

Mit diesen, in den Bildern gespeicherten Daten, ist eine Verwaltung möglich. Durch Abfragen bestimmter Kriterien kann man diese selektieren und/oder sortieren, wie es auch bei Bildverwaltungsprogrammen angewandt wird (Beispiel: Adobe Bridge und Lightroom, vgl. Evening 2009).

Der zweite wichtige Bereich beschäftigt sich mit der Filterung der Daten. Diese sollen nach bestimmten Filterkriterien ausgedünnt werden können. Die Filterung ist auf Grund der großen Datenmenge für eine sinnvolle Darstellung wichtig. Weiterhin soll ermöglicht werden, dass nicht nur die Verknüpfung der Bilder mit ihrer Position, sondern auch die Beziehung zwischen den einzelnen Daten (Bilddaten + Positionsdaten) sichtbar wird.

Diese Art der Darstellung kann zum Nachvollziehen des Fahrtwegen verwendet werden. Weiterhin helfen die Bilddaten die Umgebung des gefahrenen Weges ins Gedächtnis zurück zu rufen oder für andere sichtbar zu machen.

Der letzte Punkt befasst sich schließlich mit der Darstellung an sich. Hier soll es um die genaue Verwendung der in Bereich II ausgegeben Daten gehen. Einen groben Überblick über alle drei Bereiche und deren Beziehung untereinander zeigt Abbildung 1.

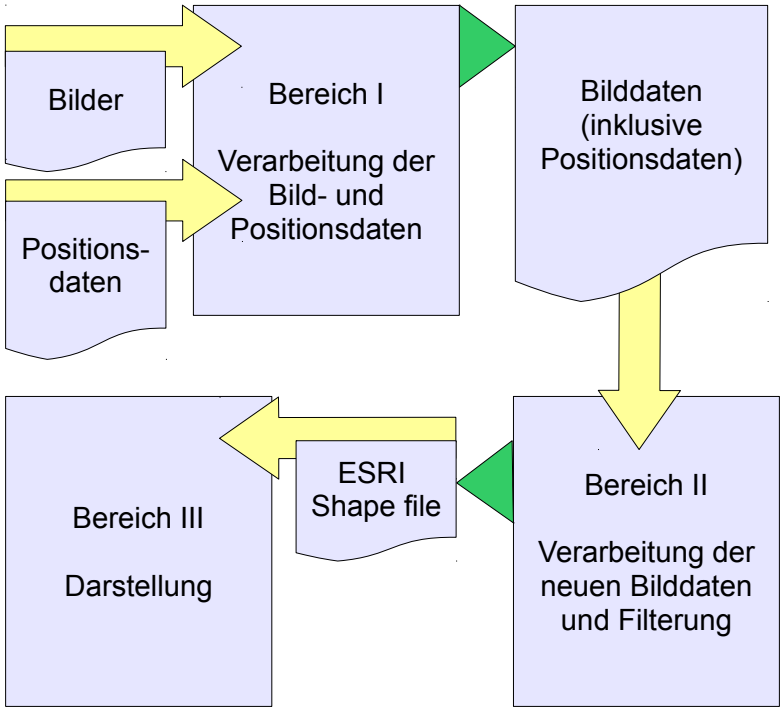


Abbildung 1: Grober Arbeitsablauf
(Quelle: Eigene Darstellung)

2 Hintergründe zur technischen Umsetzung

2.1 Datengrundlage

Die zu verwendenden Daten kommen aus Fahrten mit einem Mobile Mapping-System (TopScan 2010). Da es unterschiedliche Arten dieser Systeme gibt, können auch verschiedenste Daten vorliegen. Es gibt Systeme, die neben den Positionsdaten und Laserdaten, auch Bilder oder Videos der Umgebung aufnehmen (vgl. Ellum 2002, S. 15). Eine Abklärung, welches System verwendet wird und welche Daten erfasst werden, ist daher unerlässlich.

Mobile Mapping-Systeme bestehen, wie oben schon beschrieben, grundsätzlich aus einem Positionserfassungssystem und einem zur Aufnahme der Umgebung, zum Beispiel einer Kamera. Für diese Arbeit liegen Bilddaten im JPEG-Format vor (vgl. auch Kapitel 2.2). Die zugehörigen Positionsdaten sind in einer Textdatei, dem sogenannten Indexfile abgespeichert. Abbildung 2 zeigt das Erfassungssystem der Firma TopScan.

Abbildung 2: Mobiles Mapping-Messsystem
(Quelle: TopScan – Weblink)

In Abbildung 3 wird der grobe Aufbau eines Indexfiles gezeigt. Im Informationsblock der Datei werden die Kameradaten gespeichert. Diese Daten ergeben sich aus der Konfiguration des Kamerasystems. Hier befindet sich zum Beispiel der Wert des Winkel zwischen Kamera und der Mittelachse des Autos. Im Datenblock der Datei werden Positionsdaten, Bildverweise und weitere Werte zeilenweise pro Aufnahmestandpunkt gespeichert.

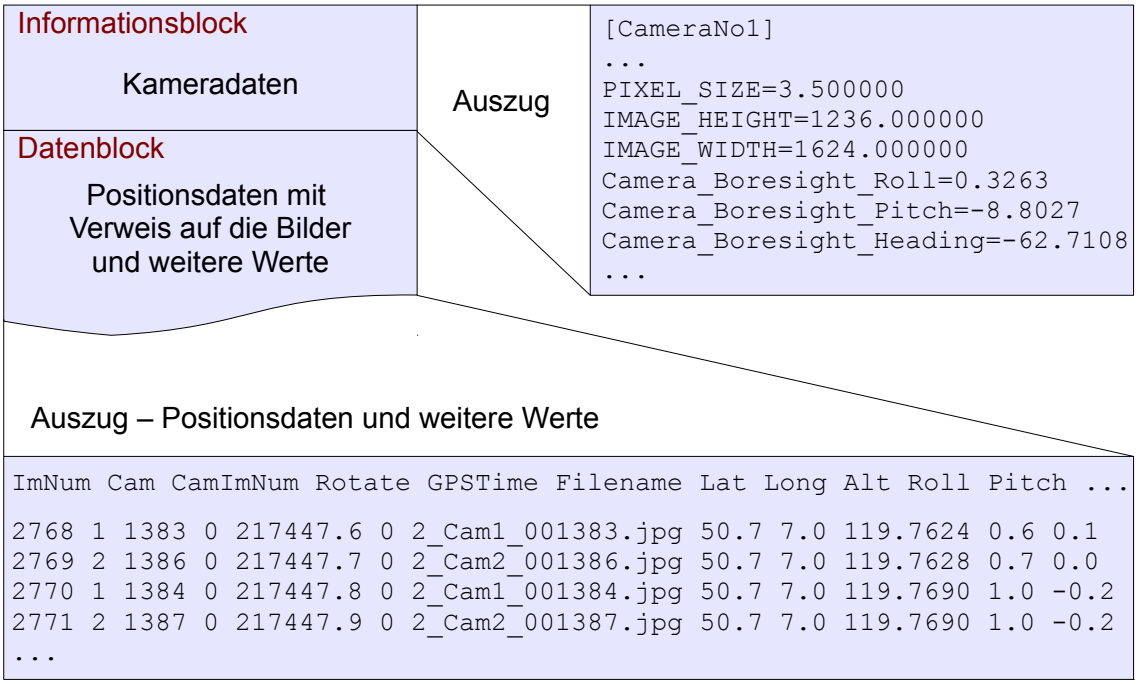


Abbildung 3: Grober Aufbau eines Indexfiles
(Quelle: Eigene Darstellung mit Beispielwerten)

Für diese Arbeit werden, wie oben bereits erwähnt, Bilder im JPEG-Format verwendet, da in diesem die Möglichkeit besteht, Daten in dem sogenannten „Header“ des Bildes abzulegen. Die Daten aus dem Indexfile werden später die Grundlage dafür bilden.

2.2 JPEG-Format und EXIF

JPEG (Joint Photographic Experts Group) ist neben dem GIF (Graphics Interchange Format), das am häufigsten verwendete Grafikformat. Es beschreibt eigentlich ein Komprimierungsverfahren von Bilddaten. Diese Komprimierung wird mit Hilfe der diskreten Kosinus-Transformation (DCT) (vgl. Bredies und Lorenz 2011, S. 133 f) und der Huffmann-Kodierung (vgl. Werner 2009, S. 216) erreicht. Diese komprimierten Bilddaten werden in dem Dateiformat JFIF (JPEG File Interchange Format) gespeichert und ermöglichen die Weitergabe der Bilder (vgl. Meinel und Sack 2009, S. 208 ff; Matzer und Lohse 2000, S. 229).

EXIF steht für Exchangeable Image File Format und ist ein Standard zur Speicherung von zusätzlichen Informationen in Bildern, speziell für Solche im JPEG-Format. Es ist zusätzlich ein Teil des DCF-Standards (Design rule for Camera File system) von JEIDA (Japan Electronics and Information Technologie Industries Association) (vgl. Walter 2005, S. 107 ff; JEITA 2002, S. 133; EXIF – Weblink; JEIDA 1998, S. 1 ff).

Die Exif Image File Spezifikation beschreibt die Methode, auf welche Art und Weise Informationen in die Bilder gespeichert werden können (JEITA 2002, S. 2). Abbildung 4 zeigt den Aufbau einer im JPEG-Format gespeicherten Bilddatei. Die Datei beginnt mit SOI (Start of Image) und endet mit EOI (End of Image). Für diese Arbeit wird nur das Segment APP1 verwendet, da dieses für die EXIF Daten vorgesehen ist. In dieser Abbildung wird die Struktur des APP1 Segments vorgestellt. Hierbei sind die Segmente „0th IFD“ und „0th IFD value“ wichtig – IFD bedeutet Image file direction (vgl. Salomon und Motta 2010, S. 539). In diesen Segmenten werden die einzelnen EXIF-Tags hinterlegt und mit Werten gefüllt. Im Abschnitt „0th IFD“ werden Zeiger zu den in 0th IFD Value hinterlegten Datenblöcken gespeichert. Diese Zeiger geben an, welche und wie viele Blöcke vorhanden sind. In 0th IFD Value kann ein EXIF IFD- und ein GPS IFD-Block mit Daten gefüllt werden. Welche Tags (zum Beispiel: Exif Version, in der Abbildung) mit Daten gefüllt werden können, ist im EXIF-Standard festgelegt (vgl. Tesic 2005, S. 86 ff; Walter 2005, S. 107 ff; JEITA 2002, S. 8 ff).

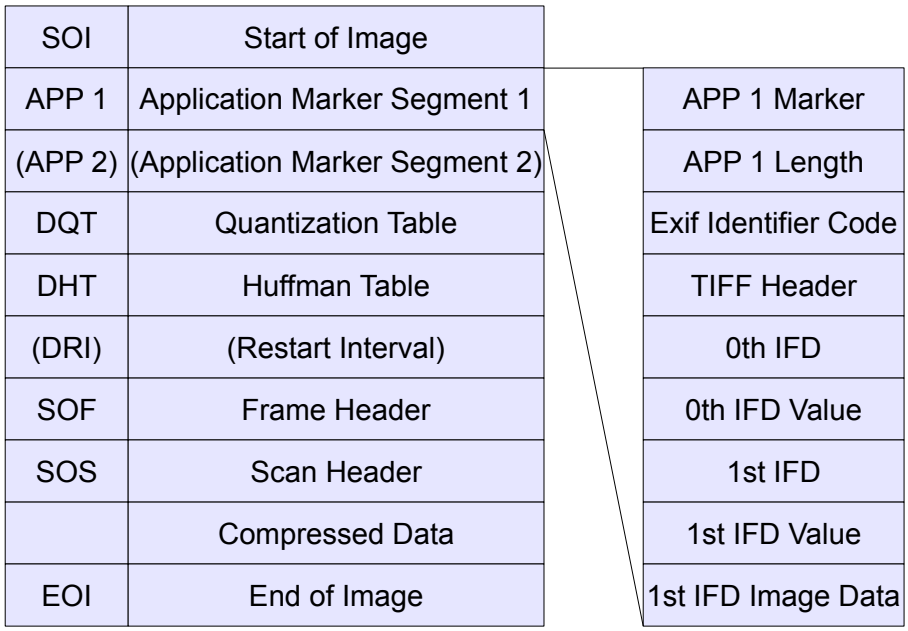


Abbildung 4: Basisstruktur von komprimierten EXIF-Bilddatenformat (Quelle: Eigene Darstellung nach JEITA 2002, S. 11)

3 Verwendete Software

In diesem Kapitel wird der Aufbau der gesamten Arbeits- und Entwicklungsumgebung beschrieben. Da auch das Betriebssystem für dieses Projekt eine wichtige Rolle spielt, wird hier ebenso darauf eingegangen, wie auch auf die verwendeten Programmiertools und Bibliotheken.

3.1 Vergleiche

Um den späteren Ablauf zu erleichtern und eine klare Gliederung zu erhalten, ist es zwingend erforderlich, dass die bereits vorhandenen Programme und Programmteile bereits vor der eigentlichen Entwicklung und Umsetzung miteinander verglichen werden.

Es gibt unterschiedlichste Software um Bilddaten zu visualisieren und zu verwalten. Meist können die Bilddaten und die in ihnen enthaltenen EXIF-Daten gar nicht oder nur teilweise ausgelesen und/oder zusammen angezeigt werden. Da ein Geoinformationssystem (kurz: GIS oder GI-System) vielfältige Fähigkeiten zur Visualisierung beinhaltet, wird in diesem Kapitel über Vergleiche der in Frage kommenden GI-Systeme und EXIF-Bibliotheken, die zur Umsetzung der Programmierung und späteren Visualisierung nötig sind, gesprochen.

3.1.1 Vergleich von GI-Systemen

Um verschiedene GI-Systeme zu vergleichen, müssen einige Kriterien, die für die Arbeit wichtig sind, festgelegt werden. Eine wesentliche Voraussetzung für die Verwendung eines GI-Systems ist die freie Nutzung. Daher sind Open Source-Produkte zu bevorzugen. Da es dort aber große Funktionsunterschiede gibt, müssen weitere Auswahlkriterien gefunden werden.

Viele Open Source-Produkte können ohne Probleme erweitert werden. Hier ist es wichtig zu wissen, welche Programmiersprache für Erweiterungen verwendet werden kann und ob es überhaupt eine Schnittstelle zur schnellen und einfachen Anbindung von neuen Programmen/Programmteilen gibt.

In der finalen Darstellung in einem GIS sollen auch Karten hinterlegt werden können. Somit müssen bestimmte Formate unterstützt werden. Hierbei sollte auf jeden Fall die Unterstützung von ESRI Shape files, im weiteren Verlauf Shape genannt, JPEG-Dateien und eine Möglichkeit der Datenbankanbindung vorhanden sein. Weiterhin ist das Einbinden von WMS (Web Map Service), WFS (Web Feature Service) und von OSM-Daten (OpenStreetMap-Daten) wünschenswert. Mit diesen Daten können Karten und Satellitenbilder angezeigt werden (vgl. Meng et al. 2005, S. 183 f). Sie helfen einen Bezug zum realen Aufnahmeort herzustellen.

Tabelle 1 zeigt die untersuchten Kriterien der einzelnen GI-Systeme und stellt diese einander gegenüber. Hiermit können die Systeme direkt miteinander verglichen werden. Vorab wird noch eine kurze Beschreibung, der in dieser Arbeit miteinander verglichene GIS, angeführt.

QuantumGIS

QuantumGIS, kurz QGIS, ist ein Open Source-Projekt, dass für verschiedene Betriebssysteme entwickelt wurde, zum Beispiel Linux und Windows. Dieses Projekt wurde 2002 von Gery Sherman ins Leben gerufen. Heute hat QGIS eine große Vielfalt an Funktionen. Es können eine Vielzahl an Raster- und Vektorformaten verarbeitet werden und auch Datenbankanbindungen sind möglich. Weiterhin werden immer wieder Erweiterungen dafür entwickelt (vgl. QuantumGIS – Weblink).

Die Arbeitsumgebung, auf der QGIS erstellt wurde, ist das Qt toolkit von Nokia. Dieses Programm bietet mit seinen Bibliotheken eine umfangreiche, aber einfach zu bedienende Umgebung für C++-Entwicklungen. Mit dem Qt toolkit können unter anderem GUI-Anwendungen (Graphical User Interface) erstellt werden (vgl. Blanchette und Summerfield 2009, S. 17 ff). Es dient im QGIS-Projekt zur Erstellung der graphischen Benutzeroberfläche. Um eine Erweiterung für QGIS zu schreiben, muss eine Qt-Umgebung auf dem Entwicklungsrechner installiert werden (vgl. QuantumGIS – Weblink).

Das QGIS-Projekt wird unter einer GNU Public License (GPL) entwickelt und herausgegeben (vgl. Mitchell et al. 2008, S. 272). Dadurch ist gewährleistet, dass die Entwickler den Quellcode komplett einsehen und nach den Regeln von GPL verändern und weiterentwickeln können. Weiterhin ist das Herunterladen dieses Programms für alle Nutzer kostenlos (vgl. Free Software Foundation 2007).

OpenEV

Ein weiteres Programm ist OpenEV – ein DesktopGIS. Es unterstützt durch die Einbindung von GDAL alle dort definierten Raster- und Vektorformate. OpenEV ermöglicht einfache Kartendarstellungen in 2- und 3-D und bringt Tools zur Bildanalyse und Erstellung von neuen Daten mit. Weiterhin ist OpenEV ein Teil des FWTool-Paketes (vgl. Mitchell et al. 2008, S. 28 f; OpenEV – Weblink).

Ein weiteres besonderes Feature dieses DesktopGIS ist die Erstellung von 3-D-Visualisierungen in Echtzeit. Die schnelle Darstellung wird mit Hilfe von OpenGL (Open Graphics Library) erreicht. Alle diese Funktionalitäten können mit der Programmiersprache Python oder C plattformunabhängig implementiert und entwickelt werden. Dadurch ist eine Erweiterung des Programms einfach möglich. Weiterhin ist zu erwähnen, dass OpenEV unter der GNU LGPL veröffentlicht wird (vgl. Mitchell et al. 2008, S. 28 f 272; OpenEV – Weblink).

MapWindow

MapWindow ist ein Open Source Desktop-GIS auf Basis der Mozilla Public License (MPL). Die Programmiergrundlage bildet hier das Microsoft .NET Framework. Alle Erweiterungen von MapWindow werden in den Programmiersprachen C# und Visual Basic geschrieben (vgl. Ames et al. 2007; Ames et. al. 2008, S. 633 f; MapWindow – Weblink).

OpenJUMP

OpenJUMP hat die Funktionalitäten von JUMP Unified Mapping Platform von Vivid Solutions übernommen und wird unter der GNU GPL veröffentlicht. Sie basiert auf einer GUI-Applikation und ermöglicht das Betrachten und Bearbeiten von räumlichen Daten. Eine Erweiterung von OpenJump ist auf Basis der Programmiersprache Java möglich. OpenJump unterstützt die OGC-Standards, wie zum Beispiel GML, WMS und WFS. Weiterhin kann OpenJump auf vielen verschiedenen Betriebssystemen verwendet werden (Windows, Linux etc.) (vgl. Mitchell et al. 2008, S. 122 272; OpenJump – Weblink).

ArcGIS

ArcGIS ist ein Produkt von ESRI (Environmental Systems Research Institute) und wird unter proprietärer Lizenz vertrieben. Es beinhaltet mehrere Teile, zum Beispiel ein Desktop GIS und ein Mobiles GIS, wie Abbildung 10 verdeutlicht. Der Funktionsumfang lässt keine Wünsche offen (vgl. ESRI – Weblink; Ormsby et al. 2004)

Ein kleiner Teil von ArcGIS ist das Desktop GIS, dass wiederum aus einzelnen Produkten besteht. Hierzu zählen die Programme ArcInfo, ArcEditor, ArcView, ArcReader und verschiedene andere Erweiterungen. ArcInfo beinhaltet Werkzeuge zur Automatisierung von Prozessen der räumlichen Modellierung, Analyse und der kartographischen Aufbereitung von Daten. Der ArcEditor kann für Analysen, Aufbereitung und Verwaltung von Daten verwendet werden. Im ArcView liegt das Augenmerk auf der Visualisierung. Es können Karten erfasst und ausgegeben werden. Dieses Produkt ist weltweit das bekannteste Produkt von ESRI. Bei ArcReader handelt es sich um einen kostenlosen Viewer. Dieser kann mit ArcInfo, ArcEditor und ArcView erstellte Karten anzeigen (ebda).

Abbildung 5: ArcGIS Diagramm
(Quelle: ESRI – Weblink)

Die folgende Tabelle zeigt die vorher beschriebenen Softwareprodukte und die Vergleichskriterien. Das erleichtert den Vergleich der verschiedenen Produkte.

	QGIS	*OpenEV*	*MapWindow GIS*	*OpenJUMP*	*ArcGIS*
Lizenz	GNU GPL	GNU LGPL	Mozilla Public License Version 1.1	GNU GPL	ESRI proprietäre Lizenz
Programmcode	C++	C Python	C# VB .NET C++	Java	VB C++ Delphi
PlugInSchnittstelle	+	?	+	+	+
PlugIn-Progammiersprache	C++ Python	Python	C# VB .NET	Java Python	VBA
Unterstützung bzw. Einbinden von ...					
• *Shape*	+	+	+	+	
• *Jpeg*	+ EXIF Header	?	+	?	+
• *Datenbankanbindung*	SpatialLite PostgreSQL/ PostGIS	?	?	MySQL Oracle	SQL
• *Online Geodaten (WMS/ WFS)*	+	?	+	+	+
• *OSM-Daten*	+	?	?	?	?
Legende: *„+“ vorhanden* *„?“ nicht bekannt*					

Tabelle 1: Vergleich verschiedener GIS nach bestimmten Kriterien (Quellen: ebda (aus Kurzbeschreibung der GIS))

Wie in der Tabelle 1 zu sehen, sind QuantumGIS und ArcGIS die umfangreichsten und scheinen für diese Arbeit am geeignetsten. Da der Einsatz von ArcGIS Lizenzkosten mit sich bringt und somit von den Vorgaben abweicht, wird dieses Programm für die weitere Umsetzung nicht in Betracht gezogen.

3.1.2 Vergleich von Bibliotheken

Durch Internetrecherchen wurden verschiedene Bibliotheken gefunden, die in einer Gegenüberstellung erkennen lassen, welche bei der späteren Umsetzung am ehesten zur erfolgreichen Bearbeitung der Programmieraufgabe geeignet sind. Die Tabelle 2 bezieht sich dabei auf folgende Bibliotheken:

- **Exiv2** - eine freie Open Source-Bibliothek und Commandline-Anwendung zum Verarbeiten von Metadaten in Bildern (vgl. Exiv2 – Weblink)
- **Libexif** - eine rein in C geschrieben Open Source-Bibliothek (vgl. libexif – Weblink)
- **Exiftool** – eine auf Perl basierende Bibliothek (vgl. ExifTool – Weblink)
- **Apache Sanselan** – eine reine Java Bibliothek (vgl. Apache Sanselan – Weblink)

Bei der späteren Verwendung der Bibliotheken ist ausschlaggebend, ob die EXIF-Header von JPEG-Bildern erfolgreich gelesen und geschrieben werden können. Ein entscheidendes Kriterium ist die verwendete Programmiersprache. Es sollten nur die Sprachen C++ und Java verwendet werden, da es für die Firma TopScan wichtig ist, die entstehenden Programme gegebenenfalls selbst erweitern zu können.

	Exiv2	*Libexif*	*Exiftool*	*Apache Sanselan*
Programmiersprache	*C++*	*C*	*Perl*	*Java*
Lesen von EXIF-Header	+	+	+	+
Schreiben von EXIF-Header	+	+	+	+
Lesen von EXIF-Header in Jpeg-Bilder	+	*?*	+	+
Schreiben von EXIF-Header in Jpeg-Bilder	+	*?*	+	*?*
Lizenz	*GNU GPL*	*GNU LGPL*	*GNU GPL*	*ASF (Apache)*
Legende: *„+“ vorhanden* *„?“ nicht bekannt*				

Tabelle 2: Vergleich von EXIF-Bibliotheken hinsichtlich ihrer Lese- und Schreibeigenschaften von JPEG-Bildern
(Quellen: Exiv2 – Weblink; Libexif – Weblink; Exiftool – Weblink; Apache Sanselan – Weblink)

Aus Tabelle 2 kann man ersehen, dass es die größte Übereinstimmung bei Exiv2 gibt. Der große Funktionsumfang und die verwendete Programmiersprache sind die maßgebenden Gründe, weshalb in dieser Arbeit Exiv2 zum Einsatz kommt.

3.2 Softwarearchitektur

3.2.1 Betriebssystem

„Historisch gesehen enthält ein Betriebssystem alle Programme und Programmteile, die nötig sind, einen Rechner für verschiedene Anwendungen zu betreiben. Die Meinungen, was alles in einem Betriebssystem enthalten sein sollte, gehen allerdings weit auseinander. […]“ (Brause 2004, S. 1). Ein Betriebssystem wird von Brause daher als: „Gesamtheit der Programmteile, die die Benutzung der Betriebsmitteln steuern und verwalten“ definiert – Betriebsmittel sind für ihn alle Einheiten, die zum Rechnerbetrieb nötig sind (ebda).

Auf dem Markt gibt es die unterschiedlichsten Betriebssysteme. Die am häufigsten verwendeten und bekanntesten sind Microsoft Windows und das Open Source-Produkt Linux, das in verschiedenen Distributionen (z.B. Ubuntu, Debian, SuSE, Redhat u.v.m.) erhältlich ist. Die meisten davon sind sogar kostenlos im Internet verfügbar (vgl. Distributionen Linux – Weblink; Tanenbaum 2009, S. 831 ff).

Welches Betriebssystem für diese Arbeit in Betracht kommt, muss durch Abwägung einiger Kriterien entschieden werden. Die Beurteilung der einzelnen Systeme muss selbstverständlich objektiv und vorurteilsfrei erfolgen.

Michael Kofler zeigt in seinem Buch, Linux, Vorurteile von Linux vs. Windows auf und betrachtet sie. Im Folgenden werden einige davon aufgeführt (2008, S. 34 ff):

- „Linux ist billiger als Windows“ (Kofler 2008, S. 36)
  Linux ist zwar kostenlos, aber dies gilt nicht für jede Linux-Distribution (ebda).
- „Linux ist komplizierter zu installieren“ und
  „[...] zu bedienen“ (Kofler 2008, S. 36)
  Dies stimmt bei den heutigen Versionen nicht mehr unbedingt, aber die Wartung des Systems ist um einiges aufwändiger. Die Umgewöhnungszeit auf Linux kann einige Zeit in Anspruch nehmen (ebda).
- „Linux ist stabiler als Windows“ (Kofler 2008, S. 35)
  Dafür muss die Art der Nutzung untersucht werden. Man kann nicht allgemein sagen, dass Linux stabiler läuft (ebda).

Da hieraus kein maßgeblicher Vorteil eines der beiden Betriebssysteme ersichtlich ist, müssen andere Aspekte zur Entscheidungsfindung beitragen. Installation und Einarbeitszeit sollten so gering wie möglich gehalten werden. Ein weiterer wichtiger Punkt ist die Analyse der Systeme auf denen die zu erstellenden Programme später zum Einsatz kommen sollen. Dazu wurde die Arbeitsumgebung der Firma TopScan GmbH näher betrachtet. Da alle verwendeten Rechner bereits mit dem Betriebssystem Microsoft Windows XP ausgestattet sind und eine Änderung der Systemsoftware nicht unerhebliche Kosten mit sich führen würde, führte dieses letztendlich zu der Entscheidung selbiges zu verwenden.

3.2.2 Entwicklungsumgebung

Es gibt viele verschiedene Entwicklertools im Bereich Open Source und auf dem proprietären Markt. Viele unterscheiden sich nur durch den Umfang und die Programmiersprachen, die diese unterstützen. Eine erste Entwicklungsumgebung (Qt) wurde zuvor schon angeführt (vgl. Kapitel 3.1.1).

Microsoft Visual Studio (MVS) bietet eine Entwicklungsumgebung für viele verschiedene Programmiersprachen. Es können .NET-Anwendungen oder auch native Programme erstellt werden. Seit der Version 2005 gibt es Visual Studio auch als kostenfreie und in ihrer Nutzung nicht eingeschränkte Version (Visual Studio Express) (vgl. Microsoft Visual Studio – Weblink).

MVS bietet ein umfangreiches und übersichtliches Programmierwerkzeug mit vielen Funktionen. Allerdings ist der Funktionsumfang bei den Express-Versionen eingeschränkt – ist aber für kleinere Softwareentwicklungen durchaus ausreichend (ebda).

Ein Grund für die Verwendung von MVS in dieser Arbeit sind verschiedene Probleme bei der Installation der Qt-Bibliotheken, die in Kapitel 5.1 näher erläutert werden. Weiterhin kann die Exiv2-Bibliothek schnell und einfach mit diesem Programm kompiliert werden. Alle zusätzlich Bibliotheken, die für Exiv2-Bibliothek benötigt werden, sind schon als MVS-Projekt vorhanden. Diese Vorkompilierungen stellen eine enorme Zeitersparnis dar und minimieren den im Vorfeld entstehenden Arbeitsaufwand (vgl. Exiv2 – Weblink).

Eine weitere Bibliothek, die in dieser Arbeit verwendet wird, ist die GDAL/OGR-Bibliothek. Für die Arbeit wird aber nur die darin enthaltene OGR Simple Freature Library benötigt (vgl. OGR – Weblink). GDAL wird als Library in das zu erstellende Programm des Bereichs II des Arbeitsablaufs (Abbildung 1) eingebunden um OGR nutzen zu können. OGR wird in dieser Arbeit für Speicherung der Positionsdaten in ein Shape ver-

wendet. Weiterhin kommt diese Bibliothek noch für die Koordinatentransformationen zum Einsatz. Nachfolgend werden daher GDAL und OGR näher erläutert.

GDAL, Geospatial Data Abstraction Library, ist eine Open Source-Bibliothek unter der X/MIT style Open Source-Lizenz der Open Source Geospatial Foundation zur Übersetzung von räumlichen Rasterdaten (vgl. Kropla 2005, S. 301 f; GDAL – Weblink).

Die OGR Simple Features Library ist in GDAL eingebunden und ist eine Bibliothek für die Verarbeitung (Zugriff, Konvertierung und Manipulation) von Vektordateiformaten, wie zum Beispiel das ESRI Shapefile. Sie ist in der Programmiersprache C++ geschrieben (vgl. OGR – Weblink).

Mit dem in OGR enthaltenen ogrinfo können Vektordaten untersucht und zugehörige Informationen über das Objekt angezeigt werden. ogr2ogr konvertiert Geodaten in alle von OGR unterstützte Formate, zum Beispiel ist eine Konvertierung von Shape zu GML (Geography Markup Language) möglich. Dazu gehört auch die Überführung von einem Ausgangskoordinatensystem in ein Zielkoordinatensystem (vgl. Mitchell et al. 2008, S. 38 f).

In dieser Arbeit werden ogrinfo und org2org nicht direkt verwendet. Es werden nur die Methoden dieser Programme durch Einbindung der OGR-Bibliothek genutzt.

3.2.3 Arbeitsumgebung

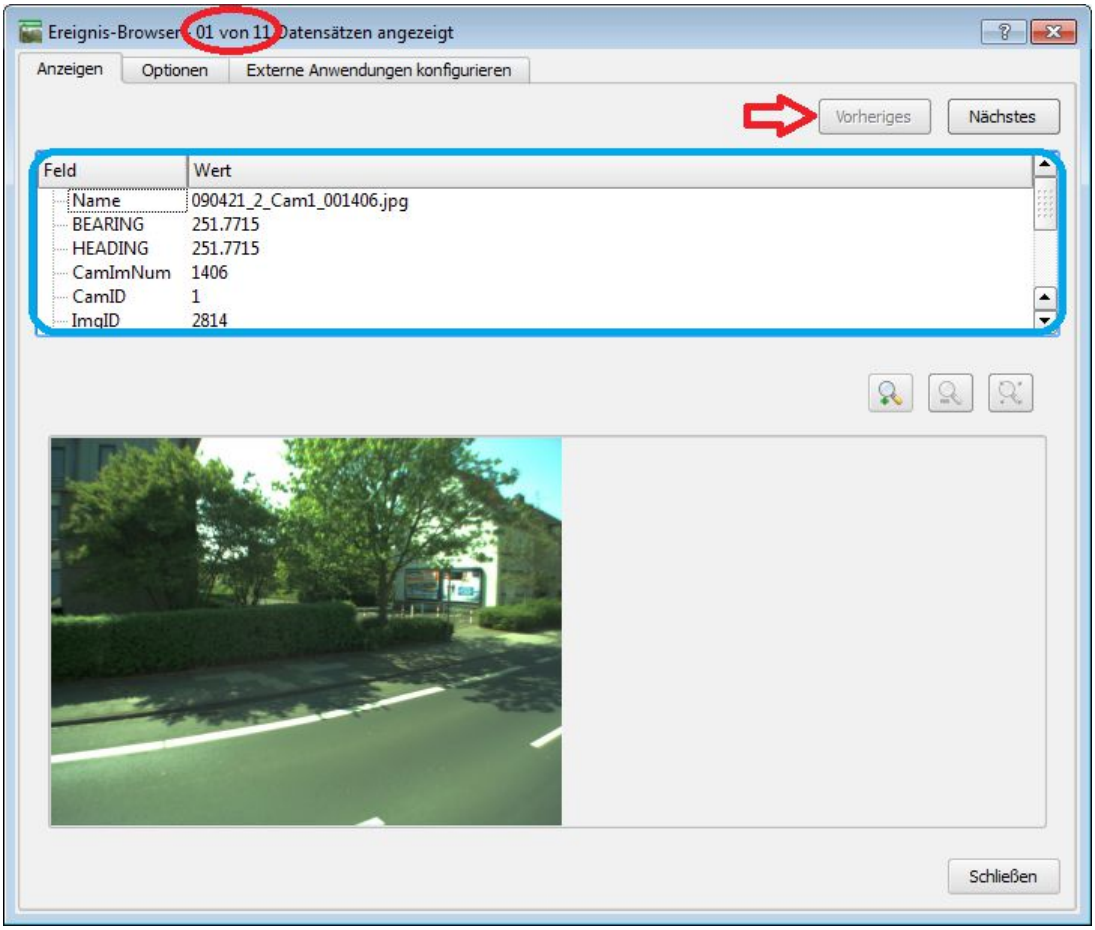
Wie zuvor im Vergleich der GIS zu erkennen, ist die Wahl auf QGIS gefallen. Diese Entscheidung wurde aus Gründen des Funktionsumfangs und der verschiedenen zusätzlichen PlugIns getroffen. Eines der PlugIns kann Bilder, die durch Angabe des Bilddateipfades in den Shape-Attributen verlinkt sind, anzeigen (vgl. QuantumGIS – Weblink).

Dieses PlugIn heißt Event Visualization Tool (eVis) und wurde von der Biodiversity Informatics Facility of the Center for Biodiversity and Conservation (CBC) im American Museum of Natural History (AMNH) als Tool für den Bereich Monitoring und Decision Support Tool für die Analyse geschützter Bereiche und die Landschaftsplanung entwickelt (vgl. Horning et al. 2009, S. 3).

Mit diesem Tool ist es möglich, Bilder mit Vektordaten und anderen von eVis unterstützten Formaten zu verknüpfen und anzuzeigen. Zusätzlich ist eVis seit Juli 2009 ein Standard QGIS-PlugIn. Es beinhaltet drei Funktionen (vgl. Horning et al. 2009, S. 3 f; QuantumGIS – Weblink):

- **Ereignis-Browser**

  Der Ereignis-Browser verknüpft die, zum Beispiel in einem Shape enthaltenen Daten (Pfad+Dateiname|URL und Richtung) mit einem Bild und kann diese anzeigen. Diese Funktionalität wird für das Abspeichern des Dateipfades/der URL in die Attribute des Shapes verwendet. Jetzt kann bei der Auswahl eines Punktes des Punkt-Shapes, das Bild im Ereignis-Browser angezeigt werden. Zusätzlich können Richtungspfeile durch eine Angabe von Winkelabweichungen bezogen auf die Nordposition dargestellt werden (vgl. Horning et al. 2009, S. 4 ff). Die folgende Abbildung zeigt und erklärt den wesentlichen Teil des Ereignisbrowsers.



Legende:

Nummer des Datensatzes im Shape — *Attributdaten des Shapes* — *Navigation durch die einzelnen Punkte im Shape*

Abbildung 6: eVis Ereignis-Browser
(Quelle: Eigene Darstellung)

- **Ereignis-ID-Werkzeug**
  Das Ereignis-ID-Werkzeug ermöglicht es durch Klicken auf einen Teil des eingeladenen Vektorlayers, das verknüpfte Bild anzuzeigen. Auch hier müssen die Daten des Vektorlayers auf den Ort des Bildes verweisen. Daher ist egal, ob eine URL oder ein lokaler Dateipfad angegeben ist (vgl. Horning et al. 2009, S. 9 f).

- **Datenbankverbindung**
  Dieses Datenbankfeature ermöglicht eine Verbindung mit ODBC (Open Database Connectivity) Datenquellen, wie zum Beispiel Excel-Tabellen oder auch Datenbanken verschiedenster Hersteller (PostGIS, SpatiaLite etc.) (vgl. Horning et al. 2009, S. 10 ff).

4 Technische Umsetzung

In dem zuvor beschriebenen Vergleich sind die grundlegenden Systeme und Bibliotheken, die für die weitere Programmierung benötigt werden, beschrieben worden. In diesem Kapitel geht es um die Umsetzung der einzelnen Teilbereiche.

Wie in der Einleitung erwähnt, gibt es zwei Bereiche (Abbildung 1), die für die Verarbeitung von Daten zuständig sind. Diese werden jeweils als separate Programme umgesetzt. Im ersten Teil dieses Kapitels wird daher die Umsetzung beschrieben, auf welche Art und Weise die Daten aus dem Indexfile in die EXIF-Header der Bilder geschrieben werden. Der danach folgende programmiertechnische Teil befasst sich mit der Erstellung eines Shapes und dem dazugehörigen Auslesen der vorher erstellten EXIF-Header aus den modifizierten Bildern. Zuletzt wird die Verwendung des Shapes erläutert. Hierzu wird das PlugIns eVis in QGIS verwendet. Dabei stehen die Anwendungen im Vordergrund, für die das Shape zum Einsatz kommen kann.

4.1 Programm I – IndexToEXIFData

Im ersten Teil des programmiertechnischen Teilbereichs der Arbeit müssen Positionsdaten und Bilddaten vereinigt werden. Dazu kommt die Exiv2-Bibliothek zum Einsatz. Diese Bibliothek erleichtert das Schreiben der in diesem Programm benötigten EXIF-Header, da sie Methoden dafür zur Verfügung stellt. Abbildung 7 zeigt den groben Programmablauf (Abbildung 1: Bereich I).

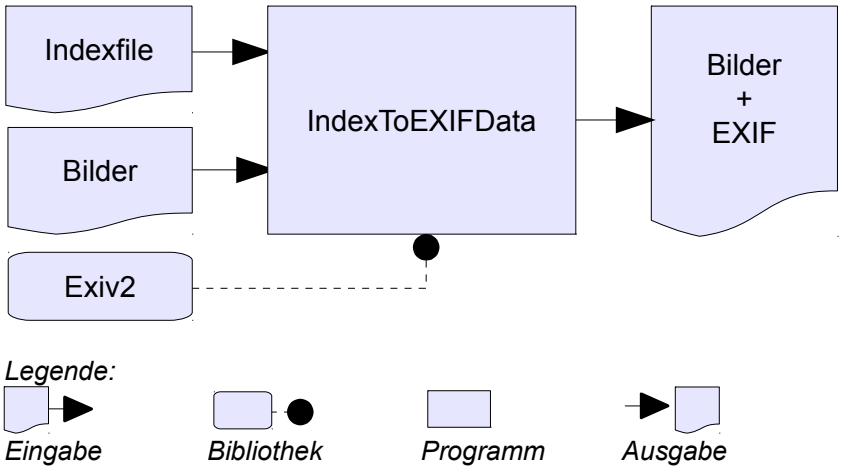


Abbildung 7: Programmablaufplan IndexToEXIFData (Quelle: Eigene Darstellung)

4.1.1 Exiv2-Bibliothek

Das Einbinden der Exiv2-Bibliothek erfordert vorab eine Kompilierung der Exiv2-Projektdaten. Dieses wird mit Microsoft Visual C++ (VC++) umgesetzt. Voraussetzung für das Kompilieren von Exiv2 ist das Vorhandensein folgender zusätzlicher Bibliotheken (vgl. Exiv2 – Weblink):

- **zlib** für PNG Support
- **gettext** für NLS (National Language Support)
- **libiconv** für Zeichensatzkonvertierungen
- **Expat** für XMP (Extensible Metadata Platform) Support

Diese liegen zum Teil als vorkompilierte Versionen vor. Die Pfade zu den Bibliotheken müssen nur an die richtige Stelle kopiert werden. Die Kompilierung von Exiv2 kann jetzt durchgeführt werden. Auftretende Fehler können durch falsche Pfadangaben zu den Include-Dateien und/oder den verlinkten Bibliotheken in der Projektdatei entstehen. Hierbei müssen entweder die Pfade verändert oder die Dateien an die vorgesehenen Stellen kopiert werden.

Nach erfolgreicher Kompilierung kann die Exiv2-Bibliothek in jedes VC++-Projekt eingebunden werden. Dazu müssen alle Link- und Include-Pfade gefüllt werden. Dies kann folgendermaßen aussehen:

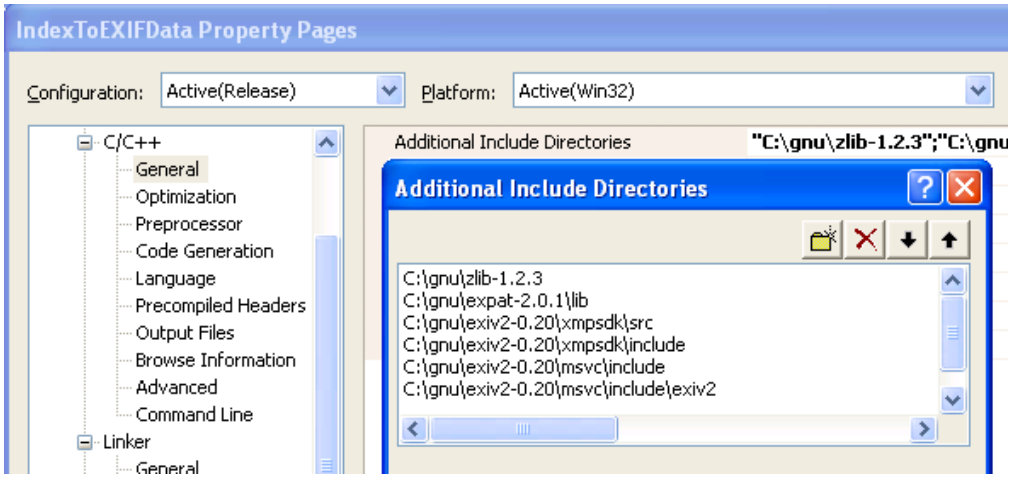


Abbildung 8: Include-Pfad Einstellungen für die Einbindung der Exiv2-Bibliothek (Quelle: Eigene Darstellung)

In der oberen Abbildung sieht man alle Link-Pfade. Diese müssen unter `C/C++ → General → Additional Include Directories` angegeben werden, damit auf die in den Ordnern enthaltenen Headerdateien zugegriffen werden kann. Dies geschieht mit `#include`, eine Direktive zur Einbindung von Bibliotheken (vgl. Definition: #Include

– Weblink; Erlenkötter 2008, S. 277).

Die nachfolgende Abbildung zeigt die unter `Linker → Input → Additional Dependencies` eingebundenen Bibliotheken. Diese werden dadurch bei der Erstellung direkt in die Exe-Datei (Win32-Datei) integriert (vgl. Toth und Louis 1999, S. 35 f). Somit müssen die externen Bibliotheken nicht zusätzlich mitgeliefert werden.

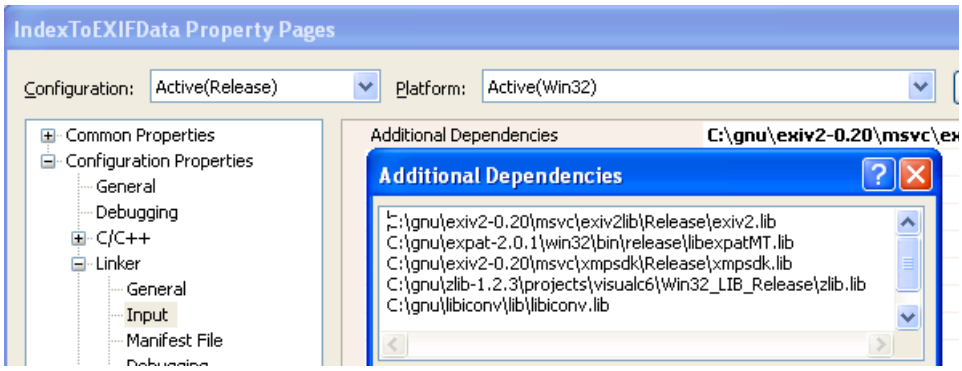


Abbildung 9: Link-Pfad Einstellung für die Einbindung der Exiv2-Bibliothek (Quelle: Eigene Darstellung)

4.1.2 IndexToEXIFData

Nach dem Einbinden der Exiv2-Bibliothek in die Projektdatei kann die Programmierung des ersten Tools beginnen.

Als Erstes muss eine geeignete Konsolenverarbeitung erstellt werden. Hier wird auf eine kontrolldateibasierende Anwendung zurückgegriffen. Dieses ermöglicht dem Benutzer später über Angaben von Variablen, das Einlesen und die Ausgabe zu steuern. Dazu muss von vornherein klar sein, welche Variablen benötigt werden. Abbildung 10 zeigt die zu der Anwendung gehörende Kontrolldatei mit Beispielpfaden.

Das Programm erfordert die folgenden Ein- und Ausgabeparameter:

- **ImageDir** (Bilderordner)
- **IndexFile** (Pfad + Indexfilename)
- **LogsDir** (Pfad zum Ausgabeordner der Logdatei)

Abbildung 10: Kontrolldatei für IndexToEXIFData (Quelle: Eigene Darstellung)

Die Variablen „ImageDir“ und „IndexFile“ bilden die im Programmablaufplan in Abbildung 7 gezeigten Eingabedateien. Die zusätzliche Variable „LogsDir“ wird später für die Ausgabe einer Log-Datei benötigt, die den geschriebenen EXIF-Header-Inhalt der Bilder wiedergibt. Dieses kann im Fehlerfall zur Analyse herangezogen werden. Aus dem Logfile ist ersichtlich, welche Daten aus dem Indexfile welchem Teil des Exif Headers zugeordnet wurden.

Der interne Verarbeitungsteil des Programms ist in drei wesentliche Teile aufgeteilt. Im ersten Teil wird die Kontrolldatei verarbeitet, die später über die Kommandozeile als Parameter an die exe-Datei übergeben wird. Dieser Abschnitt des Programmes steuert den genauen Ablauf. Das Fehlen von nicht optionalen Variablen kann zu Fehlern führen und einen Programmabbruch auslösen, daher werden nicht gesetzte nicht-optionale Variablen mit Standardwerten gefüllt oder es wird eine entsprechende Fehlermeldung generiert.

Der zweite Abschnitt des Programmes befasst sich mit dem Einlesen des Indexfiles. Die eingelesenen Daten werden in einer Map (Kontrollstruktur in C++) gespeichert. So können viele Daten zu einem bestimmten Wert innerhalb des Bildes gespeichert werden. Die folgende Abbildung zeigt den groben Verarbeitungsverlauf. Der Pfeil zeigt die Reihenfolge an, in der die Daten aus dem Indexfile ausgelesen werden.

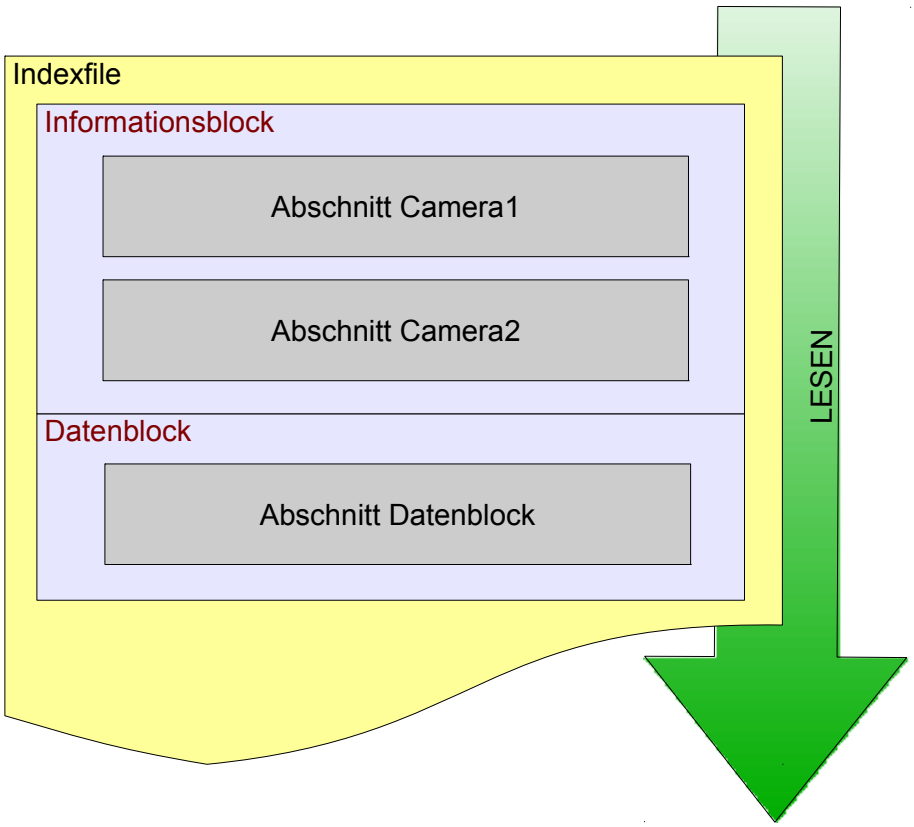


Abbildung 11: Programmablaufplan - Auslesen des Indexfile
(Quelle: Eigene Darstellung)

Der nächste Programmabschnitt befasst sich mit dem schrittweisen Auslesen der Daten, die im 2. Schritt in verschiedene Maps gespeichert wurden. Diese Daten werden in diesem Programmteil mit Hilfe von Exiv2 in die Bilder geschrieben.

Durch Setzen von #includes auf „exiv2/image.hpp“ und „exiv2/exif.hpp“ können nun die Methoden von Exiv2 genutzt werden. Als Grundlage für den Quellcode wird das Beispiel von Exiv2 verwendet (vgl. Exiv2 – Weblink).

EXIF-Header besitzen verschiedene Tags, die fest vorgegeben sind (vgl. JEITA 2002, S. 13 ff). Da nicht alle Daten aus dem Indexfile ohne weiteres in die vorgefertigten EXIF-Header-Tags eingefügt werden können, muss dafür an dieser Stelle eine Lösung gefunden werden. Durch die gute Unterstützung von Andreas Huggle (Support Exiv2) wurde eine, für diesen Zweck, gut funktionierende Lösung gefunden.

In Abbildung 12 ist ein Ausschnitt der Exif-Tags zu sehen. Jede Gruppe (Bsp.: Exif.I-

mage) der Exif-Keys kann von 0x0000 (Tag (hex)) bis 0xffff mit Tags belegt werden. Zum Beispiel steht 0x000b für ProcessingSoftware in der Gruppe „Exif.Image“. Daher können freie Tag-Felder mit zusätzlichen Werten gefüllt werden. Dies wurde zum Beispiel für die Roll- und Pitch-Werte aus dem Indexfile verwendet.

Tag (hex)	Tag (dec)	IFD	Key	Type	Tag description
0x000b	11	IFD0	Exif.Image.ProcessingSoftware	Ascii	
0x00fe	254	IFD0	Exif.Image.NewSubfileType	Long	
0x00ff	255	IFD0	Exif.Image.SubfileType	Short	
0x0100	256	IFD0	Exif.Image.ImageWidth	Long	
0x0101	257	IFD0	Exif.Image.ImageLength	Long	
0x0102	258	IFD0	Exif.Image.BitsPerSample	Short	
0x0103	259	IFD0	Exif.Image.Compression	Short	
0x0106	262	IFD0	Exif.Image.PhotometricInterpretation	Short	

Abbildung 12: Umsetzung der Exif-Tags in Exiv2 (Quelle: Exiv2 – Weblink)

Die Latitude- und Longitudekoordinaten liegen im Indexfile als Dezimalgrad vor. In der Exiv2-Bibliothek können die Koordinaten allerdings nur mit Grad Meter Sekunden abgespeichert werden. Daher muss eine Umrechnung erfolgen (vgl. Eder 2005, 8):

$$Grad = Vorkommaanteil\,(Dezimalgrad)$$

$$Minuten = Vorkommaanteil\,((Dezimalgrad - Grad)*60)$$

$$Sekunden = ((Dezimalgrad - Grad)*60 - Minuten)*60$$

Der Wert für die GPS-Image-Direction kann nicht einfach übernommen werden, da dieser sogenannte Headingwert für das Bild nur indirekt gilt. Der Wert bildet den Winkel zwischen Fahrtrichtung und der Position nach Norden ab. Dazu kommt, dass die Kameras in einem bestimmten Winkel zur Mittelachse des Fahrzeugs stehen. Dieser Wert wird unter dem jeweiligen Informationsblock der Kameras im Indexfile gespeichert. Für eine genauere Berechnung der Ausrichtung des Bildes von der Fahrzeugposition aus, kommen daher beide Winkelangaben in Betracht. In folgender Zeichnung wird der Sachverhalt dargestellt.

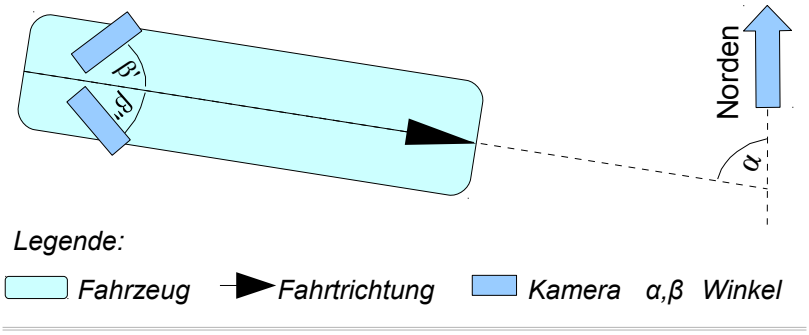


Abbildung 13: Darstellung des Verhältnissen von Nordrichtung, Fahrtweg und Kameraposition
(Quelle: Eigene Darstellung)

Dieser neu berechnete Heading-Wert ist eigentlich immer noch nicht ganz korrekt. Die Position von GPS-Empfänger und den Kameras ist nicht gleiche, somit müsste eine Transformation der Position des GPS-Empfängers zur jeweiligen Kamera durchgeführt werden. Diese wird hier nicht umgesetzt, da der auftretende Fehler für die spätere Anzeige irrelevant ist und der zusätzliche Aufwand dazu nicht im Verhältnis steht.

Die GPS-Zeit aus dem Indexfile muss für den EXIF-Header in UTC-Zeit (Coordinated Universal Time) umgerechnet werden. Hierbei muss beachtet werden, dass die GPS-Zeit am 6. Januar 1980 um Mitternacht eingeführt und die Zählung mit 0 Stunden, 0 Minuten und 0 Sekunden nach UTC begonnen wurde. Jede Woche beginnt die Zählung der Sekunden neu. Jeder Tag hat somit 86400 Sekunden, daraus ergibt sich, dass jede Woche 604800 Sekunden hat, genannt GPS Seconds of Week. Durch Verlangsamung der Erdrotation, muss die koordinierte Weltzeit auf die Universal Time angepasst werden. Dieser Abgleich wird durch das Setzen von Schaltsekunden (Leap Seconds) bewirkt (vgl. Dana 1997, S. 13 ff, Bennett et al. 2010, S. 134).

Diese Leap Seconds werden erst nach der Umrechnung von GPS-Zeit auf UTC-Zeit dazugerechnet. Folglich muss zur Berechnung des Tages in der Woche die folgende Formel verwendet werden $Tag = Vorkommaanteil\left(\frac{SecOfWeek}{(60*60*24)}\right)$.

Daraus kann man folgende Berechnungen für Stunden, Minuten und Sekunden ableiten:

$$Stunden = Vorkommaanteil\left(\frac{(SecOfWeek-(Tag*60*60*24))}{(60*60)}\right)$$

$$Minuten = Vorkommaanteil\left(\frac{\begin{array}{c}(SecOfWeek-(Tag*60*60*24))\\-(Stunden*60*60)\end{array}}{60}\right)$$

$$\begin{array}{l}Sekunden = (SecOfWeek-(Tag*60*60*24))-(Stunden*60*60)\\-(60*Minuten)+Nachkommaanteil(SecOfWeek)\end{array}$$

Die Millisekunden werden in dieser Formel einfach an den Sekundenwert angehängt. Für die Verarbeitung im Code ist es einfacher, die Millisekunden als einzelne Variable zu deklarieren und den Nachkommastellen der Seconds of Week mit 1.000.000 zu multiplizieren, damit man einen ganzzahligen Millisekunden-Wert erhält.

4.2 Programm II – EXIFToShape

Im zweiten Programm geht es darum, dass die in den EXIF-Headern gespeicherten Daten wieder ausgelesen werden und daraus ein Shape erstellt wird. Dieses Shape soll nicht alle Daten aus dem Bild beinhalten. Daher werden Einschränkungskriterien definiert, die es ermöglichen diese zu filtern.

Abbildung 14 zeigt den groben Programmablauf. Hier ist zu sehen, dass nur eine Eingabe benötigt wird. Diese wird, wie im ersten Programm, per Kontrolldatei übergeben. Diese Kontrolldatei beinhaltet ebenfalls die Einschränkungskriterien, die für eine individuelle Filterung benötigt werden.

Wie in der Abbildung 14 zu sehen, werden zwei Bibliotheken verwendet. GDAL/OGR dient zur Ausgabe und Erstellung des Shapes und wird bei der Koordinatentransformation und der Distanzberechnung zwischen zwei Punkten verwendet. Exiv2 wird für das Auslesen der EXIF-Header aus den Bildern benötigt.

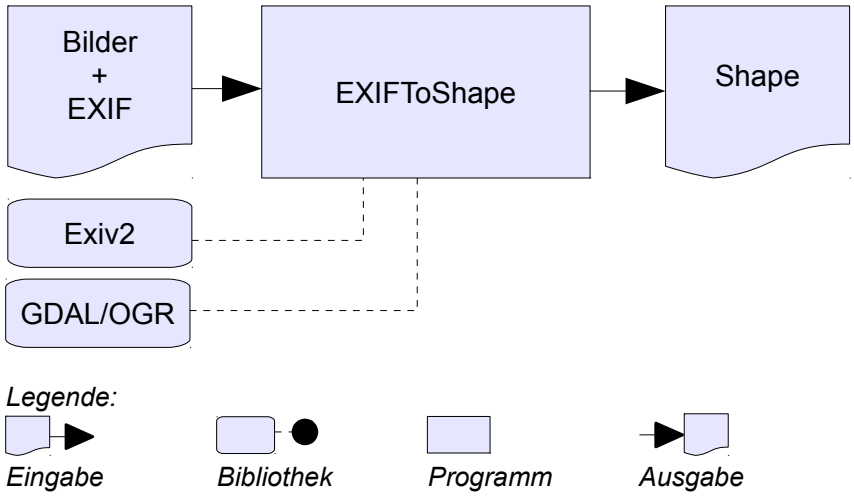


Abbildung 14: Programmablaufplan EXIFToShape (Quelle: Eigene Darstellung)

4.2.1 Datenmanagement

Datenmanagement soll in diesem Programm eine bessere Übersicht der vorhandenen Daten gewährleisten. Durch Einschränkungen, die die vorhandenen Daten nach bestimmten Kriterien ausdünnen, ist eine optimale Anzeige des auszugebenden Shapes möglich. Da bei den mobilen Mapping-Fahrten mehrere Bilder pro Sekunde erstellt und gespeichert werden, liegen große Datenmengen vor. Bei Haltepunkten und an Kreuzungen werden mehrere Bilder von der selben Stelle aufgenommen. Im Shape soll an solchen Stellen, wenn gewünscht, nur ein Bild dargestellt werden. Aus diesem Grund werden Einschränkungskriterien definiert, die, wenn nötig, nur bestimmte Bereiche zeigen bzw. nur Daten mit einem Abstand von x Sekunden oder x Metern zulassen.

Ein Einschränkungskriterium, das im letzten Beispiel erwähnt wurde, ist die Distanzberechnung. Dabei werden zwei unterschiedliche Wege verfolgt, die aus eigenen Überlegungen entstanden sind.

Im ersten Fall wird eine direkte Distanz zwischen dem ersten und dem zweiten Punkt berechnet. Ist die Distanz (x) kleiner der Angegebenen, wird die Distanz mit dem nächst folgenden Punkt verglichen. Sobald ein Distanzwert größer oder gleich dem Vergleichswert ist, wird der nächstfolgende Punkt zum 1. Referenzpunkt und die Schleife beginnt erneut bis alle Punkt abgearbeitet wurden. Dies ist in Tabelle 3 in Fall I verdeutlicht.

Fall II der folgenden Tabelle zeigt die Distanzberechnung über den exakten Fahrtweg. Hier wird geprüft, ob die Distanz (x) von Punkt 1 zu Punkt 2 größer oder gleich der Einschränkung ist. Falls dieser Fall eintritt, wird wie in Fall I, der 2. Punkt als nächster Ver-

gleichswert herangezogen. Ist dies aber nicht der Fall, so wird die Distanz vom nächsten Punkt (Punkt 3) zu Punkt 2 berechnet und zu dem vorherigen Abstand addiert. Ist dieses Ergebnis größer oder gleich, dann ist Punkt 3 der nächste Vergleichspunkt. Die Berechnung wird solange durchlaufen, bis die Distanz größer oder gleich ist oder keine Punkte mehr vorhanden sind.

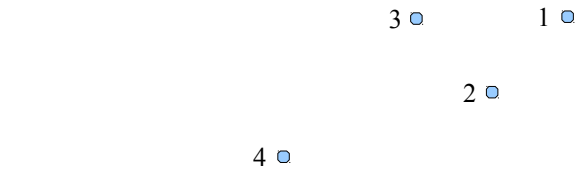
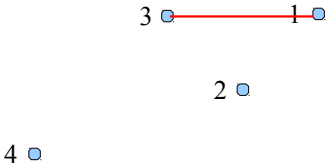
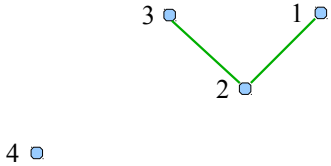
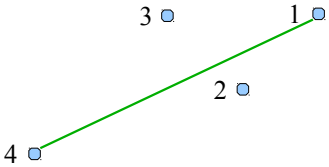
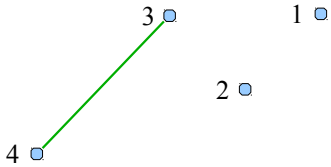
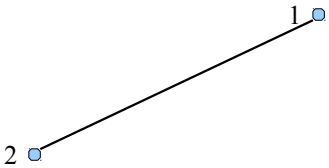
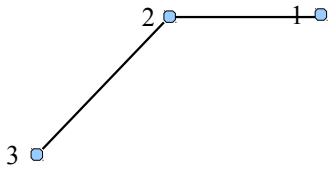
Durchgang	*Fall I*	*Fall II*
Punktfolge	 *Die Zahlen zeigen die Reihenfolge der Punkte an.*	
1		
2		 *Punkt 3 = neuer Referenzpunkt*
3	 *Punkt 4 = neuer Referenzpunkt*	
Übernommene Punkte		
Legende: — *Differenz größer gleich x* — *Differenz kleiner x* — *Pfad nach Filterung*		

Tabelle 3: Fallunterscheidung Wegberechnung
(Quelle: Eigene Darstellungen)

Weitere Einschränkungen können über die Zeit (Abstand in Sekunden oder von/bis), die Bildnummern (von/bis) und die Kameranummer (1 oder 2) gewählt werden. Alle Möglichkeiten sind einzeln, oder auch in Kombination auswählbar.

Eine andere Funktionalität, die über die Variablen in der Kontrolldatei eingestellt werden kann, ist die Transformation der Positionsdaten in ein anderes Koordinatensystem. Das Ausgangskoordinatensystem ist in diesem Fall immer Latitude/Longitude mit dem EPSG-Code: 4258, aus der Datensatzbeschreibung der European Petroleum Survey Group (EPSG) (vgl. Kresse und Fadaie 2004, 195). Wenn kein EPSG-Code für die Transformation angegeben ist, ist dies die Standardeinstellungen.

4.2.2 Shape-Erstellung

Sind alle Positionsdaten ermittelt, die im Shape abgespeichert werden sollen, wird für jeden Datensatz ein Punkt-Layer mit allen Attributen erstellt. Der Ablauf dieser Shape-Erstellung ist nachfolgend in der Abbildung 15 dargestellt (vgl. OGR – Weblink). Für die Erstellung des Shapes wird als Eingabe nur der EPSG-Code benötigt, falls die Koordinaten nicht in Latitude/Longitude Koordinaten ausgeben werden sollen.

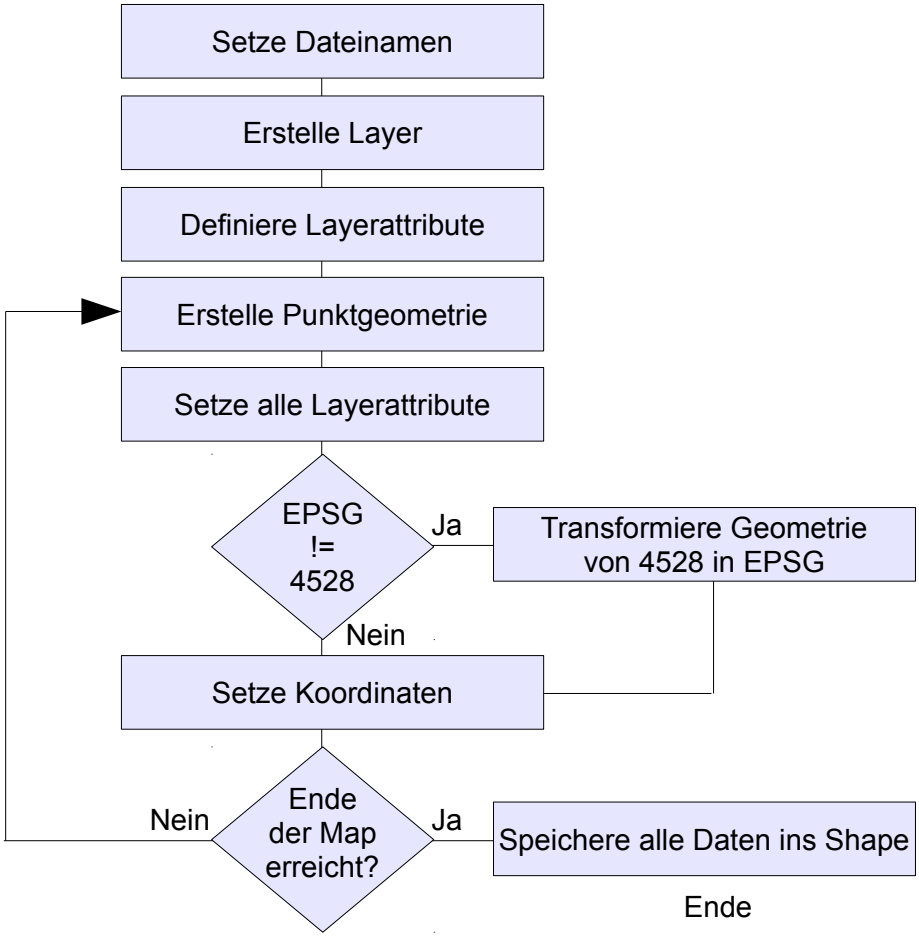


Abbildung 15: Programmablaufplan – Erstellung eines Shapes (Quelle: Eigene Darstellung nach OGR-Programmierbeispiel „Write Shape“ (OGR – Weblink))

Während der Erstellung des Shapes wird der übergebene EPSG-Code für die Transformation der Punktkoordinaten und der Definition der Projektion des gesamten Shapes

verwendet. Aufgrund der Angabe der Projektion im Shape wird zu den Standarddateien .shp (Speicherung der Geometrien), .dbf (Sachdaten/Attribute im dBase-Format) und .shx (Index der Geometrie zur Verknüpfung mit den Sachdaten) noch eine .prj-Datei angelegt, welche die Projektion beinhaltet. (vgl. Shapefile – Weblink).

Alle Punkte sind als Geometrie von OGR angelegt und können somit durch Verwendung einer Transformationsmethode in ein neues Koordinatensystem überführt werden (vgl. OGR – Weblink).

4.3 Verwendung des erstellten Shapes

Es gibt verschiedene Einsatzmöglichkeiten das Shape in QuantumGIS zu nutzen. Eine Möglichkeit ist das Anzeigen des Shapes zur Ermittlung des Fahrtverlaufs. Dies kann durch Hinterlegung von Karten (WMS etc.) geschehen. Dabei kann es nötig sein, dass das Shape in das Koordinatensystem der Karte transformiert werden muss. Diese Transformation ist aus QGIS ohne Probleme möglich. Abbildung 16 zeigt ein mit dem Programm ExifToShape erstelltes Shape mit einem Kartenausschnitt von Google (vgl. QuantumGIS PlugIns – Weblink).

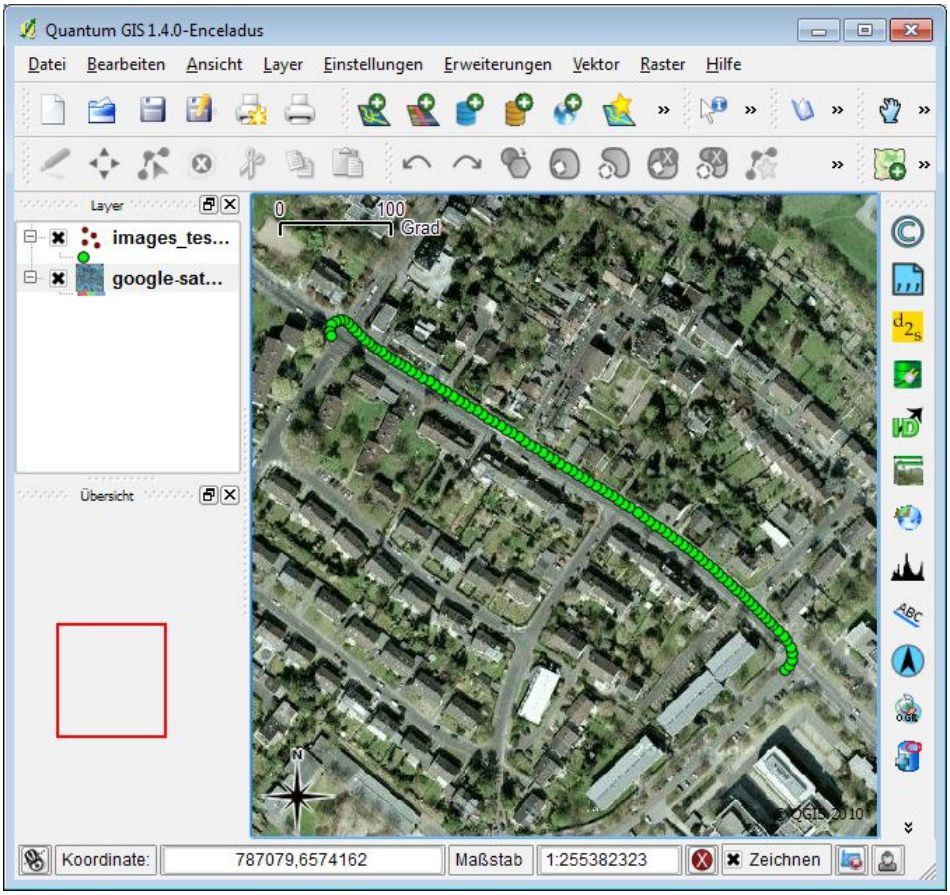


Abbildung 16: Kamera Shape in Google-Mercator-Projektion und Google-Satelitenbild
(Quelle: Eigene Darstellung)

Eine weitere Verwendung der Daten ist die Betrachtung des Fahrtweges mit Anzeige der Kamerarichtung. Dies wird mit dem Tool „Ereignis-Browser“ von eVis dargestellt. Abbildung 17 zeigt das Shape mit einer Google Straßenkarte. Der Pfeil zeigt die Ausrichtung der Kamera. Im Ereignis-Browser sind alle im Shape abgespeicherten Daten (links oberhalb des Kamerabildes) zu sehen. Mit Hilfe der Navigation (Vorheriges und Nächstes) kann man den Fahrtweg Bild für Bild nachvollziehen.

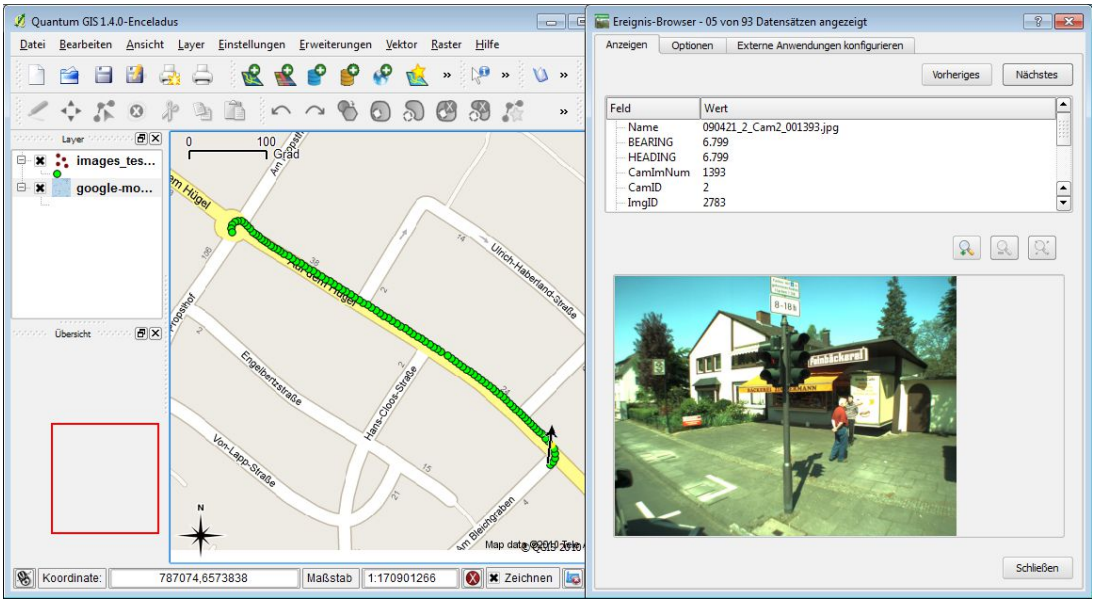


Abbildung 17: Kamera Shape und eVis Ereignis-Browser
(Quelle: Eigene Darstellung) – vergrößerte Darstellung siehe Anhang A

5 Diskussion

5.1 Programmierumgebung

Es gibt viele verschiedene Programmierumgebungen, die in dieser Arbeit zur Anwendung kommen könnten. Die Entscheidung für das Microsoft Produkt Visual Studio C++ (VC++) wurde durch verschiedene Faktoren und aufgetretende Probleme beeinflusst.

Da QuantumGIS auf dem Qt toolkit basiert, wurde zunächst die Umsetzung auf Basis dieser Entwicklungsumgebung erstellt. Sie ist leicht zu handhaben und ermöglicht es auf einfache Weise eine GUI-Anwendung in C++ zu schreiben. Das Problem hierbei war die Installation und Verwendung der Qt-Bibliotheken. In vielen verschiedenen Forenbeiträgen (vgl. Qt-Recherche – Weblink) ist beschrieben, welche Schwierigkeiten bei der Verwendung von Qt unter Windows auftreten können. Trotz langer Recherchen konnten die Probleme mit der Qt-Umgebung nicht zur Zufriedenheit gelöst werden. Auch der Wechsel von Windows 7 auf Windows XP half wenig.

Daher wurde schnell klar, dass sich nicht nur die Programmierung der beiden Programme schwierig gestalten sollte, sondern auch die Einbindung des 2. Programms (ExifToShape) in QuantumGIS. In Folge dessen wurde ExifToShape nur als seperates kommandozeilenorientiertes Programm konzipiert.

Der Einsatz von Visual Studio war für die Arbeit durchaus positiv zu sehen. Das leichte Einbinden von Bibliotheken und die unkomplizierte Handhabung waren von entscheidendem Vorteil.

Abschließend war der Wechsel auf das Microsoft Produkt durchweg ein Erfolg. Die zeitlichen Verzögerungen konnten damit schnell ausgeglichen werden.

5.2 EXIF-Bibliotheken

Die erste Umsetzung des Programms, IndexToEXIFData, wurde in Java erstellt. Nach der Einbindung, der in Java programmierten Bibliothek Apache Sanselan, stellten sich diverse Probleme ein. Laut Dokumentation können mit dieser Bibliothek „EXIF-Header“ eingelesen und auch bearbeiten werden (vgl. Apache Sanselan – Weblink). Im weiteren Verlauf stellte sich jedoch heraus, dass diese Bearbeitung nur dann funktioniert, wenn bestimmte Tags, die benötigt werden, schon in den Bildern vorhanden sind. Da dieses hier nicht der Fall war, musste auf eine andere Bibliothek zurückgegriffen werden.

Aus diesem Grund wurde der Vergleich der vorhandenen Bibliotheken nochmals überarbeitet. Die vorherigen Erfahrungen flossen in diesen Vergleich ein und führten zu dem Ergebnis, wie es schon in Kapitel 3.1.2 erläutert wurde.

Die neue Umsetzung des Programms „IndexToEXIFData“ in der Programmiersprache C++ unter der Verwendung der Exiv2-Bibliothek führte schnell zum gewünschten Ergebnis. Die Positionsdaten aus dem Indexfile konnten leicht in den „EXIF-Header“ der Bilder eingefügt werden. Eine Erweiterung dieser Bibliothek um einen MakerNote, ein Tag für herstellerspezifische Angaben (vgl. Exiv2 – Weblink), für die Firma TopScan GmbH, gestaltete sich allerdings deutlich schwieriger.

Durch schnellen und detaillierten Support von Andreas Huggle stellte sich schnell heraus, dass die Erweiterung nicht wirklich effektiv für die Arbeit sein würde. Daher wurde eine neue Lösung gesucht und gefunden. In Kapitel 4.1.1 ist nun die letztendliche Lösung beschrieben. In dieser werden in der Exif Version 2.2 freie/ungenutzte Tagfelder für die zusätzlichen Angaben „missbraucht“. Es ist zwingend notwendig, bei Erweiterungen oder Änderungen des Exif-Standards, diesen Umstand zu berücksichtigen.

6 Fazit und Ausblick

Das Thema der Arbeit befasst sich mit der Verknüpfung der getrennt aufgenommenen Bild- und Positionsdaten während Mobile Mapping-Fahrten. Eine Zusammenführung der Daten ist notwendig, um diese nach geeigneten Kriterien zu filtern und anzuzeigen. Diese Anzeige soll mit einem Open Source-Produkt durchführbar sein.

Für die Zusammenführung der Bild- und Positionsdaten wurde das Programm IndexToEXIFData erstellt. Dieses Programm verarbeitet das Indexfile, das die Positionsdaten vorhält, und speichert diese Datensätze in die zugehörigen Bilder.

Die modifizierten Bilder werden als Grundlage für das Programm ExifToShape verwendet. Die in den Bildern gespeicherten Daten werden ausgelesen und nach Filterung in ein Shape gespeichert. Zusätzlich wird noch der Pfad zu den jeweiligen Bildern in den Attributen des Shapes hinterlegt. Dieses Shape beinhaltet jetzt alle für eine geeignete Visualisierung benötigten Daten.

Nun kann die Visualisierung mit dem in QGIS enthaltenen PlugIn eVis durchgeführt werden. Hierbei werden die Daten aus dem zuvor erstellten Shape ausgelesen und dargestellt. Damit ist nicht nur die Anzeige des Fahrtweges möglich, sondern auch die Anzeige und Darstellung des zugehörigen Bildes.

Die zuvor gesetzten Ziele konnten im Rahmen dieser Arbeit vollständig umgesetzt werden. Dadurch ist der Einsatz der Programme bei der Firma TopScan GmbH zur Analyse und Visualisierung ihrer Mobile Mapping-Fahrten möglich.

Die Programme sollen veröffentlicht und weiterentwickelt werden. Ein großer Schritt hierfür wäre die Einbindung des ExifToShape-Programms in QuantumGIS. Dazu muss das Programm weniger restriktiv mit dem Einlesen der Bilddaten umgehen. Dies könnte durch kleinere Änderungen gelöst werden. Eine mögliche Erweiterung wäre die Ausgabe von KML-Dateien (Keyhole Markup Language), somit könnten die Bilder auch direkt Google Earth dargestellt werden.

Diese Änderungen würde auf jeden Fall den Anwendungsbereich vergrößern und dafür sorgen, dass die Programme auch für Nutzer von Digitalkameras mit GPS-Empfänger interessant sind. Daher wird die Autorin die Programme über diese Arbeit hinaus weiterentwickeln und auf ihrer Webseite (www.inesschiller.de) veröffentlichen.

Literaturverzeichnis

Publikationen

Ames, D. P.; C. Michaelis, T. Dunsford (2007): *Introducing the MapWindow GIS Project. In: OSGeo Journal - The Journal of the Open Source Geospatial Foundation 2007, ISSN 1994-1897, 1-3.*

Ames, D. P.; C. Michaelis, A. Anselmo, L. ChenH. und Dunsford (2008): *MapWindow GIS. In: Shekhar, S. und H. Xiong. (Hg.): Encyclopedia of GIS. New York USA: Springer Science+Buisiness Media, LLC, 633f.*

Bennett, J.; M. Donahue, N. Schneider, und M. Voit (2010): *Astronomie – Die komische Perspektive. München: Pearson Studium. 5. aktualisierte Auflage.*

Bredies, K. und D. Lorenz (2011): *Mathematische Bildverarbeitung – Einführung in Grundlagen und moderne Theorie. Wiesbaden: Vieweg+Teubner.*

Blanchette, J. und M. Summerfield (2009): *C++ GUI Programmierung mit Qt 4 – Die offizielle Einführung. München: Addison-Wesley Verlag. 2., aktualisierte Auflage*

Dana, P. H. (1997): *Global Positioning System (GPS) Time Dissemination for Real-Time Applications. Boston: RealTime systems - Kluwer Academic Publishers, 13-15.*

Eder, H. (2005): *Voronoi Diagramme für die Planung von GSM und 3G-Mobilfunknetzwerken. Wien: Technische Universität - Institut für Computergraphik und Algorithmen.*

Ellum, C. und N. El-Sheimy (2002): *Land-Based Mobile Mapping Systems. In: Photogrammetric Engineering & Remote Sensing, Januar 2002, S. 13-28.*

Erlenkötter, H. (2008): *C++ - Objektorientiertes Programmieren von Anfang an. Reinbek: Rowohlt Taschenbuch Verlag, 12. Auflage.*

Evening, M. (2009): *Photoshop CS4 Für Fotografen – Handbuch für professionelle Bildgestalter. München: Addison-Wesley.*

Free Software Foundation (2007): *GNU General Public Licence – Version 3, 29. June 2007. USA: Free Software Foundation, Inc. Available from http://www.gnu.org/licenses/-gpl.txt.*

Horning, N.; K. Koy und P. Ersts (2009): *eVis (v1.1.0) User's Guide. New York: American Museum of Natural History, Center for Biodiversity and Conservation. Available from http://biodiversityinformatics.amnh.org.*

JEIDA (1998): *JEIDA-49-2-1998 – Japan Electronic Industry Development Association Standard – Design Rule for Camera File System: Version 1.0. Japan: Japan Electronic Industry Development Association. Available from http://www.exif.org/dcf.PDF.*

JEITA (2002): *JEITA CP-3452 – Exchangeable image file format for digital still cameras: Exif Version 2.2. Japan : Japan Electronics and Information Technology Industries Association. Available from http://www.exif.org/Exif2-2.PDF.*

Kofler, M. (2008): *Linux – Installation, Konfiguration, Anwendung. München: Addison-Wesley. 8., überarbeitete und erweiterte Auflage*

Kropla, B (2005): *Beginning MapServer – Open Source GIS Development. Berkley, CA: Apress.*

Kresse, W. und K. Fadaie (2004): *ISO standards for geographic information. Berlin Heidelberg: Springer-Verlag.*

Matzer, M. und H. Lohse (2000): *Dateiformate – RTF, DOC, TIF, EPS, WAV etc. - Bedeutung, Einsatz und Konvertierung. München: Deutscher Taschenbuch Verlag GmbH & Co. KG.*

Meinel, C. und H. Sack (2009): *Digitale Kommunikation: Vernetzung, Multimedia, Sicherheit. Berlin Heidelberg: Springer-Verlag. 2. Auflage.*

Meng, L.; A. Zipf, T. Reichenbach (2005): *Map-based mobile service: Theories, Methods and Implementations. Berlin Heidelberg: Springer-Verlag. Band 1.*

Mitchell, T.; A. Emde und A. Christl (2008): *Webmapping mit Open Source-GIS-Tools. Köln: O'Reilly Verlag. 1. Auflage.*

Ormsby, T.; E. Napoleon, R. Burke, C. Groessl und L. Feaster (2004): *Getting to know ArcGIS desktop – Basis of ArcView, ArcEditor, and ArcInfo. Redlands: ESRI Press. 2. Auflage.*

Salomon, D. und G. Motta (2010): *Handbook of Data Compression. London: Springer-Verlag.*

Sieber, S. (2010): *Google Street View – Laser scannen Häuserfronten. In: News.de, 18.05.2010.*

Tanenbaum, A. S. (2009): *Moderne Betriebssysteme. München: Pearson Studium. 3., aktualisierte Auflage.*

Tao, C. V. und J. Li (2007) (Hg): *Advances in Mobile Mapping Technology. London UK: Taylor & Francis Group.*

Tesic, J. (2005): *Metadata Practices for Consumer Photos. In: IEEE MultiMedia, vol. 12 no. 3, 86-92.*

Toth, V. und D. Louis: *Visual C++ 6 – Profi-Handbuch inklusive Autorenversion. München: Markt und Technik Verlag.*

Walter, T. (2005): *MediaFotografie analog & digital – Begriffe, Techniken, Web. Berlin Heidelberg: Springer-Verlag.*

Werner, M. (2009): *Nachrichtentechnik – Einführung für alle Studiengänge. Wiesbaden: Vieweg+Teubner. 6., Auflage.*

Zlatanova, S. und J. Li (2008) (Hg.): *Geospatial Information Technology for Emergency Response. London UK: Taylor & Francis Group.*

Internetlinks

Angegeben ist jeweils das Datum des letzten Aufrufs.

Apache Sanselan – Weblink: *Apache Software Foundation*

http://commons.apache.org/sanselan/index.html (30.11.2010)

Definition: #Include – Weblink:

http://library.thinkquest.org/3285/glossindex.html (30.11.2010)

ESRI – Weblink:

http://www.esri-germany.de/products/arcgis/index.html (30.11.2010)
http://www.esri-germany.de/products/arcgis/about/desktop.html (30.11.2010)

EXIF – Weblink (2002):

http://www.exif.org/ (30.11.2010)
http://www.exif.org/makernotes/SanyoMakerNote.html (30.11.2010)

ExifTool – Weblink: *ExifTool by Phil Harvey.*

http://www.sno.phy.queensu.ca/~phil/exiftool/ (30.11.2010)

Exiv2 – Weblink: *Exiv2-Documentation by Andreas Huggel*

http://www.exiv2.org/ (30.11.2010)
http://exiv2.org/download.html (30.11.2010)
http://exiv2.org/doc/addmoddel_8cpp-example.html (30.11.2010)
http://www.exiv2.org/tags.html (30.11.2010)
http://www.exiv2.org/makernote.html (30.11.2010)
http://dev.exiv2.org/boards/3/topics/show/44 (30.11.2010)

GDAL – Weblink:

http://www.gdal.org (30.11.2010)

libexif – Weblink:

http://libexif.sourceforge.net/ (30.11.2010)

Distributionen Linux – Weblink: *Liste verschiedener Distributionen von Linux*

http://de.wikipedia.org/wiki/Liste_von_Linux-Distributionen (30.11.2010)

MapWindow Open Source Team – Weblink:

http://www.mapwindow.org (30.11.2010)
http://www.mapwindow.org/mapwinapp.php (30.11.2010)
http://www.mapwindow.org/plugins.php (30.11.2010)
http://www.mapwindow.org/wiki/index.php/Main_Page (30.11.2010)
http://www.mapwindow.org/wiki/index.php/Tutorials (30.11.2010)
http://www.mapwindow.org/wiki/index.php/MapWindow_User%27s_Guide (30.11.2010)
http://www.mapwindow.org/wiki/index.php/GISTools_User%27s_Guide (30.11.2010)

Microsoft Visual Studio – Weblink:

http://de.wikipedia.org/wiki/Microsoft_Visual_Studio (30.11.2010)

OGR – Weblink:

http://www.gdal.org/ogr/ (30.11.2010)
http://www.gdal.org/ogr/ogr_apitut.html (30.11.2010)
http://www.gdal.org/ogr/classOGRGeometry.html (30.11.2010)
http://www.gdal.org/ogr/ogr_arch.html (30.11.2010)
http://www.gdal.org/ogr/osr_tutorial.html (30.11.2010)

OpenEV – Weblink:

http://openev.sourceforge.net (30.11.2010)
http://openev.sourceforge.net/index.php?page=features (30.11.2010)
http://openev.sourceforge.net/index.php?page=developers-documentation (30.11.2010)

OpenJump – Weblink:

http://sourceforge.net/apps/mediawiki/jump-pilot/index.php (30.11.2010)
http://openjump.org/index.html (30.11.2010)

Qt-Recherche – Weblink: *Auszug aus Recherche-Ergebnissen*

http://www.qtforum.de/forum/viewtopic.php?t=11901&sid=4be0937cd2541083c65e0c2331408f8 (30.11.2010)
http://www.qtcentre.org/archive/index.php/t-25821.html (30.11.2010)
http://www.qtforum.de/forum/viewtopic.php?t=11227&sid=f6dee26536bec577ce0e6aaddf370edc (30.11.2010)

QuantumGIS – Weblink:

http://www.qgis.org/de/ueber-qgis.html (30.11.2010)
http://download.osgeo.org/qgis/doc/manual/qgis-1.5.0_user_guide_de.pdf (30.11.2010)
http://forum.qgis.org/viewtopic.php?f=2&t=6559 (30.11.2010)

QuantumGIS PlugIns – Weblink:

http://pyqgis.org/repo/contributed (30.11.2010)

Shapefile – Weblink:

http://gis.hsr.ch/wiki/Shapefile (30.11.2010)
http://www.esri.com/library/whitepapers/pdfs/shapefile.pdf (30.11.2010)

Street View – Weblink:

http://de.wikipedia.org/wiki/Google_Street_View (30.11.2010)
http://maps.google.de/intl/de/help/maps/streetview/ (30.11.2010)

TopScan – Weblink: *Internetauftriff der Firma TopScan GmbH*

http://topscan.de/front_content.php (30.11.2010)
http://topscan.de/deutsch/ueberuns/ (30.11.2010)
http://topscan.de/deutsch/mobile-laser-scanning/ (30.11.2010)

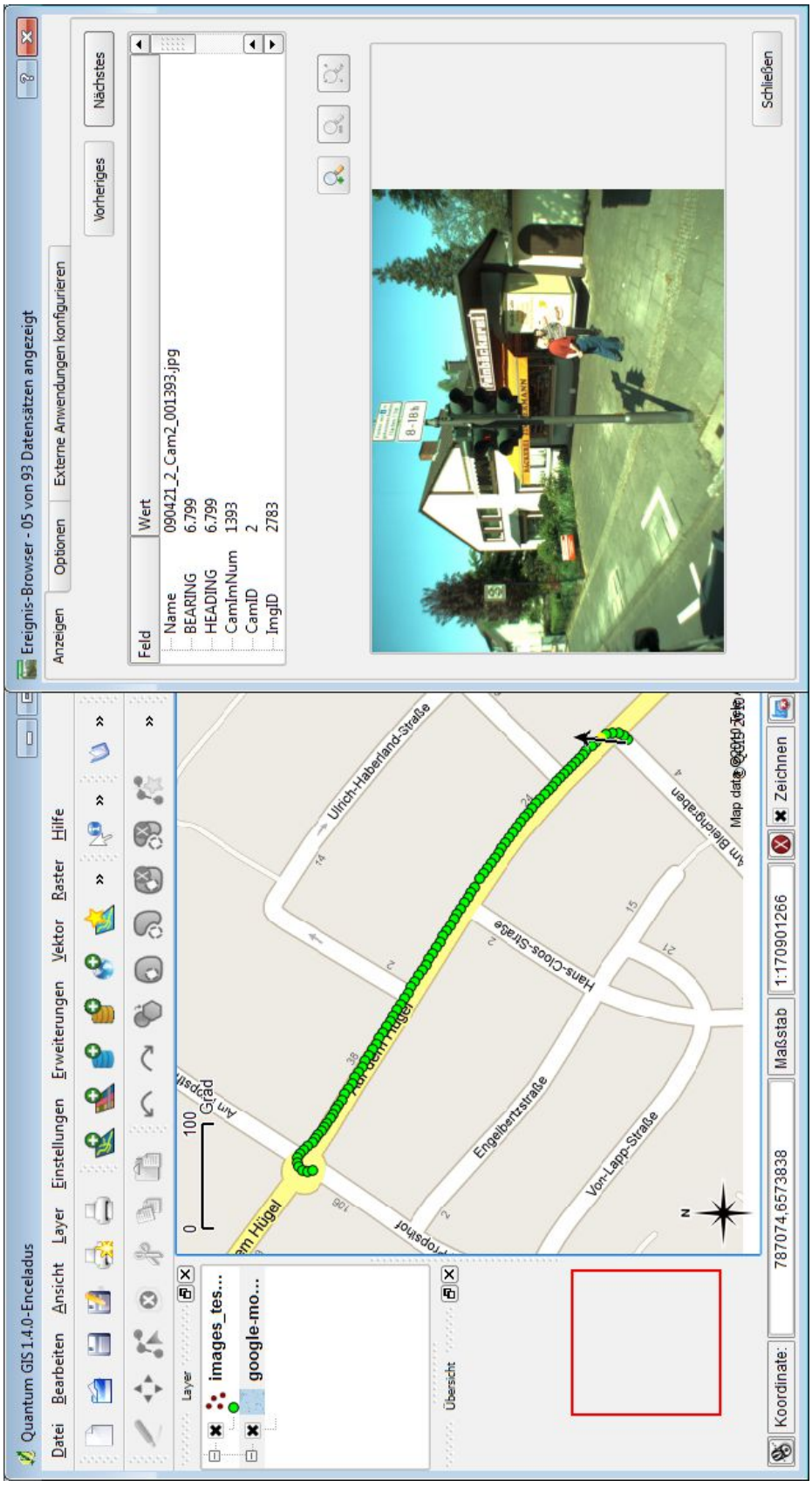
Weitere Literatur:

TopScan (2010): Bilddaten und Indexfile – bereitgestellt am 21.05.2010 per externe Festplatte

Die ausgehändigten Daten der Firma TopScan GmbH wurde nur für die Erstellung dieser Bachelorarbeit ausgehändigt. Eine Verwendung der Daten nach Abschluss dieser Arbeit ist ausgeschlossen.

Anhang

A


Quantum GIS 1.4.0-Enceladus
Datei Bearbeiten Ansicht Layer Einstellungen Erweiterungen Vektor Raster Hilfe
Ereignis-Browser - 05 von 93 Datensätzen angezeigt
Anzeigen Optionen Externe Anwendungen konfigurieren
Vorheriges Nächstes
Schließen
Feld Wert
Name 090421_2_Cam2_001393.jpg
BEARING 6.799
HEADING 6.799
CamImNum 1393
CamID 2
ImgID 2783
Layer
images_tes...
google-mo...
Übersicht
Ulrich-Haberland-Straße
Hans-Cloos-Straße
Engelbertzstraße
Von-Lapp-Straße
Am Bleichgraben
Am Hügel
0
100 Grad
N
Koordinate:
787074,6573838
Maßstab
1:170901266
Zeichnen

B

CD-ROM mit Quellcode und ausführbare Projektdateien

Diese CD enthält beide vollständige Visual Studio-Projekte der Programme IndexToEXIF-Data und ExifToShape. Weiterhin ist dieses Dokument als PDF enthalten. Zusätzlich sind noch für diese Arbeit wichtige Spezifikationen und Handbücher beigefügt.

Erklärung zur selbstständigen Abfassung der Bachelorarbeit

Ich versichere hiermit an Eides statt, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die im Literaturverzeichnis angegebenen Quellen benutzt habe. Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten oder noch nicht veröffentlichen Quellen entnommen sind, sind als solche kenntlich gemacht. Die Zeichnungen oder Abbildungen in dieser Arbeit sind von mir selbst erstellt worden oder mit einem entsprechenden Quellennachweis versehen. Diese Arbeit ist in gleicher oder ähnlicher Form noch bei keiner anderen Prüfungsbehörde eingereicht worden.

Rheine, den 30.11.2010

_____

Ines Schiller