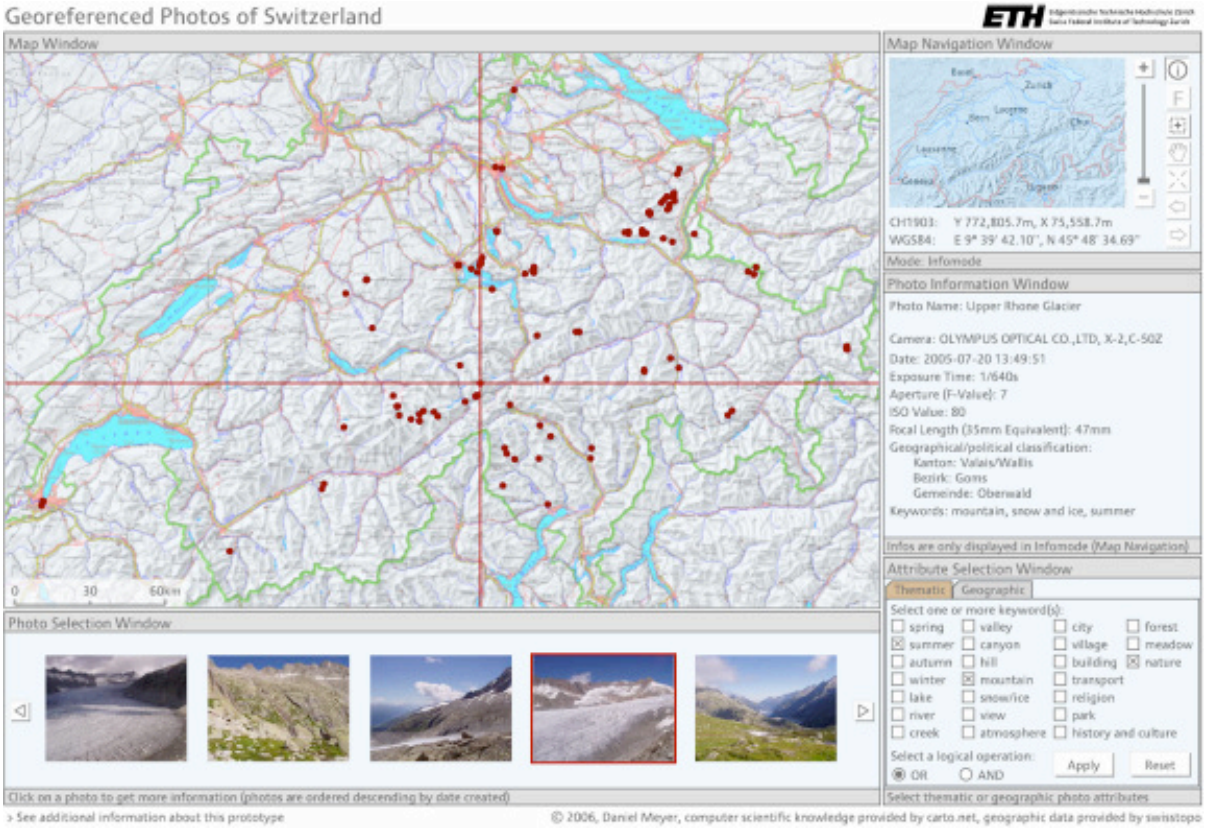
Eidgenössische Technische Hochschule (ETH) Zürich
Departement Bau, Umwelt und Geomatik
Studienrichtung Geomatikingenieurwissenschaften

Vertiefungsblockarbeit am Institut für Kartographie (IKA)

Webpublikation von georeferenzierten Fotos

Entwicklung eines auf die Schweiz begrenzten Prototyps



Autor: Daniel Meyer
Leitung: Prof. Dr. Lorenz Hurni, IKA
Betreuung: Andreas Neumann, IKA

Zürich, 9. Februar 2006

Vorwort

Die vorliegende Arbeit erstellte ich in meinem neunten Studiensemester der Geomatikingenieurwissenschaften an der ETH Zürich im Rahmen eines Vertiefungsblocks am Institut für Kartographie.
Der Themenkreis <georeferenzierte Fotos und interaktive Webkarten>, welcher dieser Arbeit zugrunde liegt, weckte mein Interesse aus zwei Gründen. Erstens fotografiere ich gerne und verwende häufig eine Digitalkamera. Zweitens sagten mir erste Erfahrungen bei der Entwicklung von interaktiven Webkarten im Rahmen der Vorlesung <Multimedia- und Webkartografie>, welche ebenfalls vom Institut für Kartographie angeboten wird, sehr zu.
Die Arbeit bereitete mir grosse Freude, da ich sehr viel Neues lernte. Mein eher bescheidenes technisches Vorwissen im Hinblick auf die Zielsetzung des Projekts bot eine motivierende Ausgangslage. So konnte ich im Verlauf der Arbeit Wissen und Erfahrungen sammeln in einigen Skriptsprachen und Datenbankanwendungen.
Dass dieser Lernprozess für mich stets angenehm verlief, ist zu einem grossen Teil auf meinen Betreuer vom Institut für Kartographie, Andreas Neumann, zurückzuführen. Er unterstützte mich während des gesamten Projekts mit viel Interesse an der Arbeit, grosser Hilfsbereitschaft und überragender fachlicher Kompetenz. Für diese reiche Unterstützung möchte ich ihm an dieser Stelle ganz herzlich danken.
Ebenfalls einen grossen Dank möchte ich an Prof. Dr. Lorenz Hurni richten für die Möglichkeit, dieses Thema im Rahmen eines Vertiefungsblocks überhaupt bearbeitet haben zu können.
Des weiteren sei allen Personen gedankt, welche dieses Projekt in irgendeiner Form, insbesondere durch die Spende von Fotos, unterstützt haben.

Daniel Meyer
Zürich, 9. Feb. 2006

Zusammenfassung

Die Nutzung von Digitalkameras hat sich in den letzten Jahren rasant verbreitet und die Geräte werden technisch immer professioneller. Ein wahrscheinlicher Entwicklungstrend liegt in der Aufwertung der Digitalkamera mit einem integrierten GPS-Empfänger. Mit dieser Kombination ist es möglich, die aufgenommenen Bilder gleich mit Koordinaten zu versehen und somit georeferenzierte Fotoaufnahmen zu machen. Die Koordinaten werden dabei als Metainformationen in die Bilddateien geschrieben.
Es existiert heute bereits ein Standard für die Speicherung von Metainformationen in Bilddateien. Dieser Standard heisst EXIF und steht für Exchangeable Image File Format. Die entsprechenden Metainformationen werden daher EXIF-Daten genannt. Abgelegt werden die Daten im Header der Bilddatei. Die Palette der EXIF-Daten ist beeindruckend. So werden von Digitalkameras automatisch sämtliche Aufnahmeparameter und Angaben zur Beschreibung der erstellten Bilddatei abgespeichert. Darüber hinaus kann ein Fotograf die aufgenommenen Bilder auch nachbearbeiten und ihnen beschreibende Texte, wie z. B. den Bildtitel, den Namen des Fotografen mit Urheberrechtsnotiz oder thematische Schlagwörter zuweisen. Ebenfalls kann neben dem Aufnahmestandort ein weiterer Punkt, der Zielpunkt des aufgenommenen Fotos, mit Koordinaten definiert werden.
Ziel dieser Arbeit ist es nun, ausgehend von diesen EXIF-Daten eine ansprechende und benutzerfreundliche Webseite mit einer interaktiven Karte zu erstellen, auf welcher georeferenzierte Fotos visualisiert werden. Es sollen auch Suchfunktionalitäten für die angebotenen Fotos zur Verfügung gestellt werden.
Der erste Arbeitsschritt zur Erreichung dieser Zielsetzung ist das Sammeln von geeigneten Fotografien und deren Nachbearbeitung. Dann wird eine Datenbank erstellt, in welcher ausgewählte Metainformationen aus den EXIF-Daten der Fotos abgelegt werden können. Anschliessend wird ein Perl-Skript geschrieben, das dem automatisierten Auslesen der EXIF-Daten aus den Bilddateien und dem gleichzeitigen Schreiben dieser Daten in die Datenbank dient.
Der nächste Arbeitsschritt ist die Erstellung der eigentlichen Webseite. Dabei wird als Grundtechnik SVG (Scalable Vector Graphics, XML-Derivat zur Beschreibung zweidimensionaler Vektorgrafiken) angewandt. Für die Aufwertung der Webseite mit dynamischen Visualisierungen und Interaktionsmöglichkeiten wird ECMAScript (standardisierte clientseitige Skriptsprache) eingesetzt. Die Anbindung der erstellten Datenbank mit den Metainformationen zu den angebotenen Fotos wird mithilfe von PHP (Hypertext Preprocessor, serverseitige Skriptsprache) und der Anwendung von SQL-Abfragen (Structured Query Language, deklarative Datenbanksprache) realisiert.
Die erstellte Webseite bietet eine interaktive Karte der Schweiz mit komfortablem Navigationsinstrumentarium, eine Galerie der angebotenen Fotos, die Anzeige von Metadaten zu den Fotos und Auswahlmöglichkeiten zur thematischen und geografischen Suche nach Fotos an. Die Fotos werden dabei auf der Karte durch Kamerasymbole an den Aufnahmestandorten und einem Blickfächer, welcher vom Zielpunkt und von dem aus der Brennweite berechneten Blickwinkel des Fotos abgeleitet wird.

Abstract

The use of digital cameras has spread rapidly in recent years and the devices have become more and more technically professional. One probable development trend lies in the upgrade of digital cameras with a GPS-receiver. This combination enables digital cameras to enhance the taken pictures with coordinates and to provide thus georeferenced photos. Thereby the coordinates are written as metadata into the image files.
Today, there's already a standard for the storage of metadata in image files. This standard is called EXIF and stands for Exchangeable Image File Format. Therefore the metadata in this context are called EXIF-Data. The data are stored in the header of the image file. It's impressing, what diversity of information one can define by these EXIF-Data. Thus all sorts of camera parameters and information to the provided image are stored automatically by digital cameras. Beyond that, a photographer can also rework taken pictures and describe them with texts, such as the title of the photo, the name of the photographer with a copyright notice or thematic keywords. Furthermore it's possible to define another point besides the standpoint of the photographer with coordinates which is considered the viewpoint of the photo.
The goal of this thesis is it to prepare a pleasing and user-friendly web page with an interactive map on which the georeferenced photos are visualized. Furthermore some search functionalities should be offered for the exploration of the provided photos.
The first step in the procedure is to collect suitable photos and to rework them. Then a database has to be established, into which selected information from the EXIF-Data of the photos can be stored. Subsequently, a perl-script has to be written for the automated selection of the EXIF-Data from the image files and the simultaneous writing of these data into the database.
The next step is the actual preparation of the web page. The web page is described in SVG (Scalable Vector Graphics, XML markup language for describing two-dimensional vector graphics). For the amelioration of the web page with dynamic visualizations and interaction possibilities the standard of ECMAScript (client-side scripting programming language) is used. The connection to the established database with the metadata of the provided photos is realized with PHP (Hypertext Preprocessor, server-side programming language) and the use of SQL queries (Structured Query Language, declarative database language).
The prepared web page offers an interactive map of Switzerland with comfortable navigation equipment, a gallery with the provided photos, the display of metadata of the photos and some options to search for photos in a thematic or geographic approach. The photos are visualized on the map with camera symbols located where the pictures have been taken and a sector of view, which is determined by the coordinates of the view point and the angle of view that is calculated from the used focal length of the camera.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Inhaltsverzeichnis Anhang

1 Definition des Auftrags

1.1 Rekapitulation der Ausschreibung

In der Ausschreibung des Vertiefungsblocks wird das Projekt folgendermassen umschrieben:

Projekt geophotos.net
Profi-Digitalkameras sind heute bereits mit einem GPS Modul ausgerüstet oder nachrüstbar und speichern mit den normalen Kameraparametern auch die Geokoordinaten, an denen das Foto aufgenommen wurde, in den **EXIF-Daten** mit ab. Es ist abzusehen, dass dieser Trend auch in Richtung Consumer-Kameras geht und in Zukunft eine Bilderflut von **georeferenzierten Fotos** auf uns zukommen wird. In diesem Projekt soll eine Datenbank georeferenzierter Fotos sowie derer Metadaten erstellt werden. Ein Karteninterface und eine Suchmaske sollen die Bilderrecherche erleichtern. Als Kartengrundlage könnte die Digital Chart of the World dienen, oder, falls vorhanden, auch existierende WMS Services genutzt werden. Es können auch die Fotos von http://www.confluence.org genutzt werden.

1.2 Konkretisierung des Auftrags

An der **Startsitzung** zum Vertiefungsblock vom 27.10.2005 wurde der Auftrag besprochen und einige Modifikationen definiert. Die wichtigste Änderung betrifft den geografischen Bezugsrahmen. Aus Gründen der Datenverfügbarkeit soll die Basis keine Weltkarte sein, sondern eine Karte, welche sich auf das Gebiet der **Schweiz** beschränkt. Diese Basiskarte soll interaktiv sein und in verschiedenen Detaillierungsgraden zur Verfügung gestellt werden.
Des weiteren soll sich das Projekt primär auf die **Visualisierung der Fotos auf der Karte** und auf ein **Abfragesystem für die Foto-Metadatenbank** beschränken. Auf eine Daten- und Userverwaltung über das Web soll aus Zeitgründen grundsätzlich verzichtet werden.
Bezüglich der zu verwendenden Fotos einigt man sich darauf, vorwiegend auf eigene, private Sammlungen zurückzugreifen.

Als **Hauptbetreuer** des Projekts wird Andreas Neumann bestimmt. Weitere Nebenbetreuer sind Peter Sykora und André Winter.

Der **Arbeitstitel** wird aufgrund der Modifikationen folgendermassen geändert:
Webpublikation von georeferenzierten Fotos
Entwicklung eines auf die Schweiz begrenzten Prototyps

2 Konzept

2.1 Definition von Ziel und Zweck

Das Ziel dieses Projekts ist die Entwicklung eines Prototyps für eine **Webseite**, welche Fotos von Landschaften oder Bauwerken zusammen mit ihrem auf einer Karte visualisierten Aufnahmestandort präsentiert. Der Prototyp beschränkt sich dabei geografisch auf das Gebiet der Schweiz.
Die Metadaten der präsentierten Fotos werden in einer **Datenbank** abgelegt, für welche ein Abfragesystem entwickelt werden soll.

2.2 Zielpublikum und Nutzungssituation

Die angebotene Webseite als Resultat des Projekts wendet sich grundsätzlich an jeden, der Interesse an georeferenzierten Fotos (der Schweiz) hat. Die Webseite soll in englischer Sprache gehalten sein und keine spezifischen Kenntnisse voraussetzen. Die Berechtigung zu einer Veröffentlichung der kartografischen Grundlagen im Web wurde abgeklärt und bestätigt.
Die Nutzungssituation des Prototyps ist auf die Betrachtung der angebotenen Fotos und eine **explorative Analyse des Datenbestands** beschränkt. Auf ein Angebot, das fremden Usern erlaubt, eigene Fotos in die Webseite zu integrieren, wird verzichtet.

2.3 Daten

2.3.1 Kartografische Grundlagedaten

Die kartografischen Grundlagedaten werden von Andreas Neumann über WMS zur Verfügung gestellt. WMS steht für **Web Map Service** und ist ein Standard für die internetgestützte Erstellung von Karten innerhalb eines verteilten Geoinformations-systems. Dieser Standard beruht auf Spezifikationen des Open Geospatial Consortium [OGC]. Bei den Karten handelt es sich um die offiziellen **digitalen Landeskarten der Schweiz** vom Bundesamt für Landestopografie [STOPO] in Rasterformat in den Massstäben 1:25'000, 1:50'000, 1:100'000, 1:200'000, 1:500'000 und 1:1'000'000.

2.3.2 Fotografische Daten

Der fotografische Datenbestand wird aus **privaten Sammlungen** der Beteiligten akquiriert. Es soll sich dabei um digital vorliegende Fotografien mit einer ansprechenden Qualität und einem sinnvollen Sujet (Landschaft, Bauwerke) handeln.
Zentral bei diesem Projekt sind die **EXIF-Metadaten** (siehe Kap. 3), mit welchen man die Fotos beschreiben kann. Die nächstliegenden Metadaten sind dabei in den IPTC-Tags

gespeichert. So können dort der Bildname, der Aufnahmezeitpunkt und Schlagwörter bestimmt werden. Wichtig sind auch die GPS-Tags, mit welchen die Position des Aufnahmestandortes, die Position eines Zielpunktes und die Blickrichtung des Fotos definiert werden können. Es gibt sogar IPTC-Tags, mit welchen eine gestufte geografische Einordnung des Aufnahmestandortes durchgeführt werden kann (Country, Province-State, Sub-location und City). Ebenfalls ist es natürlich möglich, die Aufnahmeparameter der Digitalkamera anzugeben.

2.4 Konzept zu Inhalt und Layout

Die Webseite besteht aus verschiedenen Bereichen, welche durch fest platzierte **Fenster** voneinander abgegrenzt sind. Diese Fenster sollen aber nicht verschiebbar sein, da für das Grundlayout nicht viele sinnvolle Alternativen bestehen. Frei beweglich sollen lediglich die separat aufrufbaren Foto-Fenster sein, welche ausgewählte Fotos in besserer Auflösung präsentieren. Dominiert wird das Layout der Webseite vom Kartenfenster. Rechts daran angrenzend sollen im oberen Bereich die Kartennavigation, im mittleren Bereich Informationen für das selektierte Foto und im unteren Bereich die Attributauswahl zur Foto-Suche platziert sein. Unterhalb der Karte wird eine Foto-Galerie präsentiert, welche Kleinansichten der angebotenen Fotos enthält. Natürlich gehören zum Seitenlayout auch ein Kopfbereich mit dem Titel der Seite und ein Fussbereich mit ergänzenden Angaben.
Auf dem **Kartenausschnitt** sind die Aufnahmepositionen der in diesem Bereich angebotenen Fotos punkthaft dargestellt. Wird auf einen dieser Punkte geklickt, wird das entsprechende Foto in der Galerie hervorgehoben und der Aufnahmestandort mit Blickrichtung und Blickwinkel zum Zielpunkt des Fotos in Form eines Fächers dargestellt. Basierend auf der Grösse dieses Fächers wird der Kartenausschnitt wenn nötig angepasst, sodass der ganze Fächer sichtbar ist.
Die Instrumente zur **Kartennavigation** bestehen einerseits aus einer kleinen Referenzkarte mit einem beweglichen Rechteck, welches den ausgewählten Kartenausschnitt repräsentiert. Andererseits werden die klassischen Navigationsinstrumente wie Plus-Minus (Vergrössern-Verkleinern), Aufziehrahmen zur Vergrösserung, Pan-Funktion, Rezentrierungs-Funktion und Vollansicht-Funktion zur Verfügung gestellt. Ebenfalls soll es möglich sein, in der vorangegangenen Navigation einen Schritt zurück und vorwärts zu gehen (History-Funktionen). Zudem werden die Koordinaten des Mauspfeiles angezeigt.
Bei der **Selektion von Attributen** werden zwei getrennte Vorgehensweisen angeboten. Einerseits kann thematisch nach Fotos gesucht werden, indem verschiedene die Fotos beschreibende Schlagwörter ausgewählt werden. Andererseits kann der Kartenausschnitt aufgrund der Auswahl von politischen Einheiten bestimmt werden. Die Auswahl der Schlagwörter wird durch einzelne Checkboxes und zwei Radio-Buttons für eine AND-, OR-Verknüpfung realisiert. Die Auswahl der politischen Einheiten wird mit drei Selektionslisten für Kanton, Bezirk und Gemeinde bewerkstelligt.

Als Informationen zu dem ausgewählten Foto sollen folgende Attribute aus den **EXIF-Metadaten** präsentiert werden:

- Name des Fotos
- Kameramodell
- Aufnahmezeitpunkt
- Aufnahmeparameter der Kamera (Brennweite, Blende, Belichtungszeit, ISO-Wert)
- Thematisch beschreibende Schlagwörter
- Koordinaten des Aufnahme- und Zielpunktes des Fotos
- Distanz und Blickwinkel (berechnet aus Brennweite)
- Politisch gestufte Angabe der Aufnahmeposition (Kanton, Bezirk, Gemeinde)
- Eigentümer des Fotos mit Verweis auf Urheberrechte

Das **Farbkonzept** besteht aus einer Kombination von unauffälligen Farben für die Fenster und starken Farben für die Hervorhebung der Foto-Visualisierungen. Für die Fenster wird jeweils ein dunkelgrauer Farbton für Titel- und Statusleiste sowie ein helles Blau für den Fensterhintergrund gewählt. Für die Darstellung der Kamerasymbole auf dem Kartenhintergrund hat sich ein dunkles Rot bewährt. Diese Darstellung muss sich von allen Karten (verschiedene Massstäbe) gut abheben, was aufgrund der etwas unterschiedlichen Farbgebung der Karten ein wenig diffizil ist.
Eine **schematische Darstellung des Layouts** (ohne gestalterische Details und Farben) ist in Abb. 1 gegeben. Anzumerken bleibt, dass sich die effektive Ausgestaltung des Layouts wie auch die Farbkombinationen nicht zu Beginn des Projekts strikt festlegen lässt, sondern vielmehr in einem iterativen Prozess während der ganzen Bearbeitung eruiert wird.

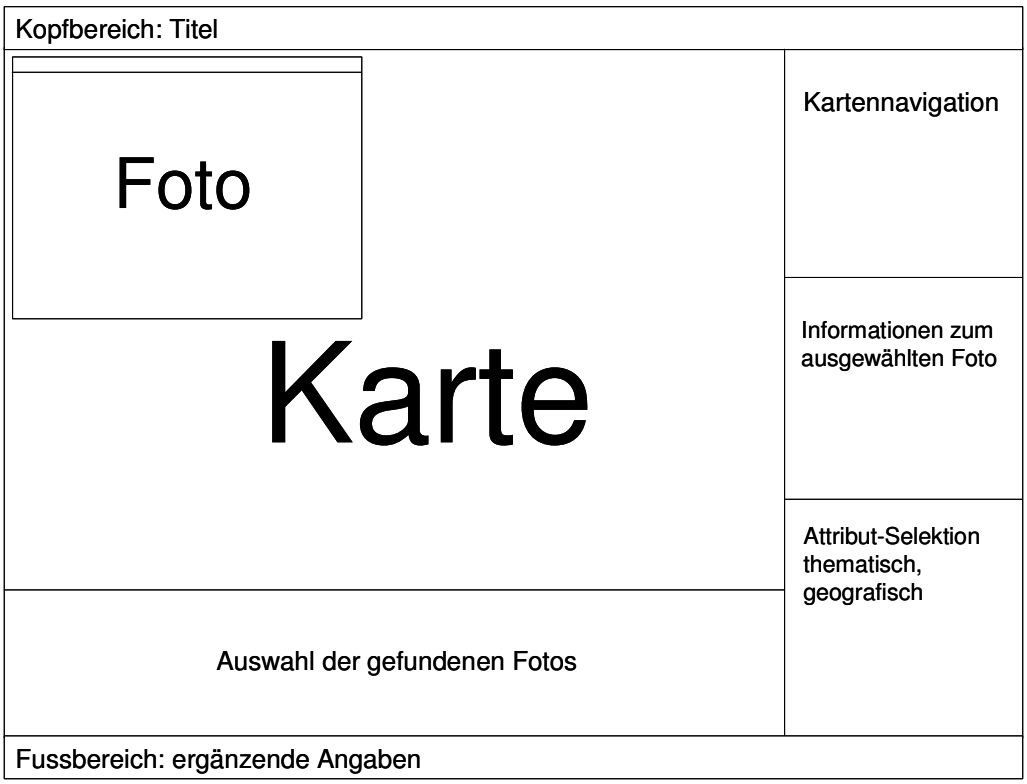


Abb. 1 Schematische Darstellung des Layoutkonzepts

2.5 Konzept zur Technik

Für die Verwaltung der Foto-Metadaten wird eine **PostgreSQL Datenbank** genutzt (siehe Kap. 4.1). Das Design der Datenbank enthält dabei neben einer Auswahl der EXIF-Daten auch Tabellen und Funktionen, welche Koordinatenumrechnungen und geografische Abfragen erlauben. Die einzelnen Bilder sollen über einen fixen Pfad, welcher sich an den IPTC-Tags zur gestuften geografischen Einordnung orientiert, abrufbar sein.
Zum Einlesen der EXIF-Metadaten aus den Bilddateien in die Datenbank wird ein **Perl-Skript** erstellt (siehe Kap. 4.2).
Die ganze Webseite wird in **XML und SVG** (siehe Kap. 5.1.1) beschrieben. Die Formatierung einzelner Elemente wird mit **CSS-Eigenschaften** definiert. Die Interaktionsmöglichkeiten werden mit **ECMAScript** (siehe Kap. 5.1.2) realisiert. Die Anbindung an die Datenbank mit den Foto-Metadaten erfolgt über **PHP und SQL** (siehe Kap. 5.1.3).

2.6 Arbeitsablauf

Ein schematischer Zeitablauf mit den einzelnen **Bearbeitungsschritten** des Projekts ist in Abb. 2 dargestellt. Die Eckpunkte des Projekts sind:

- Start: 27.10.2005
- Schlusspräsentation: 02.02.2006
- Abgabe: 10.02.2006

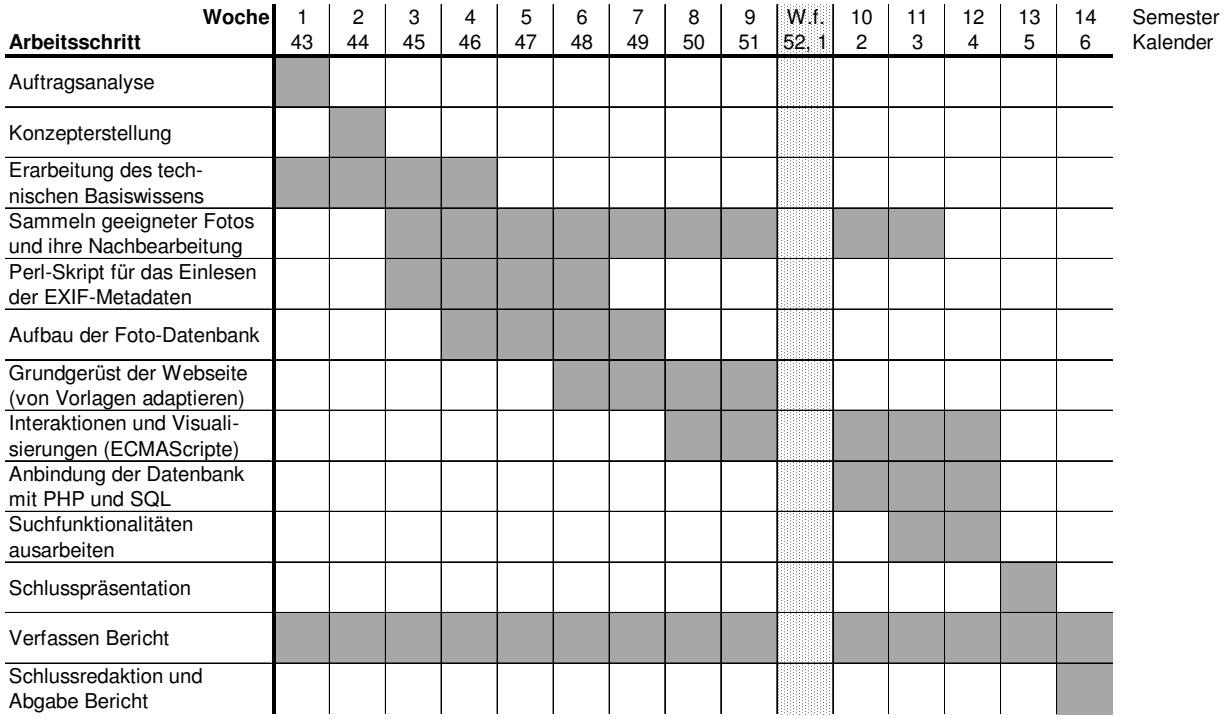


Abb. 2 Schematische Darstellung des Arbeitsablaufs

3 EXIF-Metadaten

3.1 Allgemeines

EXIF steht für **Exchangeable Image File Format** und ist ein Standard der Japan Electronic Industry Development Association (JEIDA). Dieses Dateiformat wird dazu verwendet, Metadaten der Bilder von Digitalkameras zu speichern. Die EXIF-Metadaten sind dabei im Header der Bilddateien (etwa JPEG oder TIFF) abgelegt. Weitere Informationen sind unter [EXIF1] zu finden.
Die zentralen EXIF-Metadaten sind dabei in den **EXIF-Tags** abgelegt. Hier werden alle Aufnahmeparameter der Digitalkamera abgespeichert.
Einen anderen Teil der EXIF-Spezifikation bilden die **IPTC-Tags**. IPTC steht für International Press Telecommunications Council und ist ein älterer Standard für die Speicherung von Textinformationen in Bilddateien. Für weitere Informationen beachte man [IPTC].
Ein weiterer, für dieses Projekt wichtiger Teil der EXIF-Metadaten sind die **GPS-Tags**. GPS steht für Global Positioning System. Diese Tags dienen der Definition von Koordinaten im WGS84-Projektionssystem (World Geodetic System 1984).
Ein Index aller EXIF-Metadaten ist unter [EXIF2] zu finden. Die für dieses Projekt besonders relevanten Tags (EXIF-, IPTC- und GPS-Tags) sind mit ihren Definitionen im Anhang Kap. 2 aufgeführt.

3.2 Bearbeitung der EXIF-Metadaten

Es existieren diverse Programme, welche einen Ausschnitt aus den EXIF-Metadaten zusammen mit den Bilddateien anzeigen können. Am meisten verbreitet ist die Unterstützung der IPTC-Tags, welche auch mit vielen Programmen editierbar sind. Einige EXIF-Tags, wie der Hersteller und das Model der Kamera werden ebenfalls von diversen Programmen angezeigt.
Viele weitere EXIF-Tags und insbesondere die GPS-Tags werden jedoch nur von wenigen Programmen unterstützt. Die meisten von ihnen bieten dabei keine Editiermöglichkeit für diese Tags an.
Ein einfaches und populäres Programm, das die Anzeige von EXIF- und GPS-Tags sowie die Bearbeitung von IPTC-Tags anbietet, ist IrfanView [IRFAN]. Nachfolgend aufgelistet sind einige **Freeware-Programme**, welche auf EXIF-Metadaten spezialisiert sind:

- ExifTool [EXIF3]
- Opanda IExif und PowerExif [OPANDA]
- Exifer for Windows [EXIFER]
- Exif-Viewer [EXIFV]
- RoboGeo [ROBO]

Freeware-Programme, die eine umfangreiche Bearbeitung der EXIF-Metadaten erlauben, sind leider noch äusserst rar. Mit dem oben genannten Opanda PowerExif ist eine

komfortable Bearbeitung möglich, allerdings muss das Programm für die Speicherung einiger editierter Werte registriert werden, was mit einem Kostenaufwand verbunden ist. Das einzige Freeware-Programm, mit welchem alle EXIF-Metadaten editiert werden können, ist das in diesem Projekt verwendete Perl-Modul ExifTool von Phil Harvey. Die Bilddateien werden dabei über eine command-line Applikation bearbeitet.

3.3 Praktische Ausführungen

3.3.1 Auswahl der relevanten Tags

Aus der Fülle der EXIF-Metadaten sind für dieses Projekt nur einige wenige relevant. Die in Tab. 1 aufgeführten Tags werden für die weitere Bearbeitung ausgewählt. Wichtig ist dabei die Unterscheidung in Tags, welche von normalen Digitalkameras automatisch Werte zugewiesen erhalten und Tags, welche manuell zu bearbeiten sind.

Tag-Name	**Beschreibung**	**a (automatisch) / m (manuell)**
EXIF-Tags		
Make	Kamerahersteller	a
Model	Kameramodell	a
ExifImageLength	Fotohöhe in Pixel	a
ExifImageWidth	Fotobreite in Pixel	a
DateTimeOriginal	Aufnahmedatum und -zeit	a
ExposureTime	Belichtungszeit in sec	a
FNumber	Blende	a
ISO	ISO-Wert	a
FocalLength	Brennweite in mm	a
FocalLengthIn35mmFormat	Brennweite in 35mm-Äquivalent	a / m
FocalPlaneXResolution	Auflösung des CCD-Chips in X-Richtung	a
FocalPlaneResolutionUnit	Einheit der Grössenangabe des CCD-Chips (mm, inches)	a
ScaleFactor35efl	Umrechnungsfaktor der Brennweite in 35mm	a
IPTC-Tags		
ObjectName	Name des Fotos	m
Keywords	Schlagwörter zur Fotobeschreibung	m
City	Hierarchisch gestufte Angaben zur geografisch- politischen Einordnung des Aufnahmestandortes	m
Sub-location		m
Province-State		m
Country-PrimaryLocationCode		m
Credit	Name des Fotografen	m
CopyrightNotice	Notiz zu Urheberrechten	m
GPS-Tags		
GPSLatitudeRef	Norden oder Süden	a / m
GPSLatitude	geografische Breite des Aufnahmestandortes (WGS84)	a / m
GPSLongitudeRef	Osten oder Westen	a / m
GPSLongitude	geografische Länge des Aufnahmestandortes (WGS84)	a / m
GPSDestLatitudeRef	Norden oder Süden	m
GPSDestLatitude	geografische Breite des Fotozielpunktes (WGS84)	m
GPSDestLongitudeRef	Osten oder Westen	m
GPSDestLongitude	geografische Länge des Fotozielpunktes (WGS84)	m

Tab. 1 Ausgewählte EXIF-Metadaten

Die manuelle Ergänzung der Werte für diejenigen Tags, welche nicht von der Kamera automatisch ausgefüllt werden, wird mithilfe des bereits genannten Perl Moduls ExifTool [EXIF3] bewerkstelligt.

3.3.2 IPTC-Tags

Bei der Ergänzung der IPTC-Tags handelt es sich ausschliesslich um **Texteingaben**. Hierbei ist besonders auf die Einheitlichkeit der Angaben zu achten. Alle Beschreibungen sollen (soweit sinnvoll) in englischer Sprache ausgeführt werden.
Das Tag Keywords soll **Schlagwörter** enthalten, die das zu beschreibende Foto klassifizieren. Dieses Tag wird später dazu genutzt, um thematisch nach Fotos zu suchen. Für mögliche Wertbesetzungen wird ein **Auswahlkatalog** erstellt, welcher in Tab. 2 zu sehen ist. Es handelt sich dabei um 24 Schlagwörter, welchen jeweils auch alternative Begriffe zugewiesen sind (Aliases). Ein solcher Katalog von Schlagwörtern bietet natürlich grosses Diskussionspotential, freilich hätte man auch eine andere Auswahl treffen können. Eine Evaluation, ob die getroffene Auswahl einerseits alle relevanten Bereiche abdeckt und andererseits diese auch gleichmässig unterteilt, hätte den Rahmen dieser Arbeit überschritten.

Id	Keyword	Aliases						
1	**spring**	Frühling	springtime					
2	**summer**	Sommer	summertime					
3	**autumn**	Herbst	fall					
4	**winter**	Winter	wintertime					
5	**building**	Gebäude	buildings	house	houses	structure	structures	cottage
		cottages	castle	fort				
6	**village**	Dorf	villages	hamlet				
7	**city**	Stadt	cities	town	towns	urban		
8	**transport**	Transport	transportation	road	roads	street	streets	alley
		alleys	railroad	railroads	railway	railways	square	bridge
9	**forest**	Wald	wood	woods	trees			
10	**meadow**	Wiese	meadows	grassland	hayfield			
11	**lake**	See	lakes	pond	ponds			
12	**river**	Fluss	rivers	stream	streams			
13	**creek**	Bach	creeks	brook	brooks			
14	**valley**	Tal	valleys	dale	dales	vale	vales	glen
15	**mountain**	Berg	mountains	mount	pass	summit	peak	
16	**hill**	Hügel	hills	knoll				
17	**view**	Aussicht	outlook	look-out	vista	vistapoint	viewpoint	
18	**nature**	Natur	natural	animal	plant	vegetation	flora	fauna
19	**atmosphere**	Stimmung	ambience	sunset	sunrise	clouds	Wolken	
20	**history and culture**	history	culture	Kultur	Geschichte	past	heritage	
21	**snow and ice**	snow	ice	Schnee	Eis	glacier	Gletscher	
22	**park**	Park	municipal park	garden				
23	**religion**	Religion	church	churches	monastery	convent	cathedral	chapel
		cemetary	graveyard	temple	synagoge	mosque	minaret	shrine
24	**canyon**	Schlucht	canyons	gorge	ravine			

Tab. 2 Auswahlkatalog für Keywords mit Aliases

Die **Tags zur geografischen Einordnung** erhalten in diesem Projekt klar definierte Bedeutungen. Da sich der Prototyp auf die Schweiz begrenzt, ist dem Tag CountryPrimaryLocationCode stets der Wert Switzerland zugeordnet. Für das Tag Province-State wird der Name eines Kantons gewählt. Das Tag Sub-location ist mit einem Bezirksnamen definiert und dem Tag City wird ein Gemeindename zugeordnet. Diese Tags werden im vorliegenden Projekt jedoch nicht manuell bearbeitet, da die Geometriedefinitionen aller Gemeinden der Schweiz verfügbar sind. So können die Fotos mithilfe der Koordinaten des Aufnahmestandpunkts und Definitionen aus der erstellten Datenbank (siehe Kap. 4) automatisch den richtigen politischen Einheiten zugewiesen werden.

3.3.3 GPS-Tags

Bei den GPS-Tags können diejenigen Tags, welche den **Aufnahmestandort** betreffen, unter Umständen automatisch generiert werden. Dies ist der Fall, wenn man über ein Gerät verfügt, das sowohl als Digitalkamera als auch als GPS-Empfänger genutzt werden kann (bisher gibt es aber erst wenige derartig ausgerüstete Kameras oder Mobiltelefone, ein Beispiel ist das Motorola i860). Man kann aber auch seine Digitalkamera mit einem separaten GPS-Empfänger synchronisieren und mit diesem einen entsprechenden Tracklog aufzeichnen. Am PC kann man dann mithilfe des Perl-Moduls gpsPhoto von Andreas Neumann und Peter Sykora [GPSP] die GPS-Koordinaten in die EXIF-Metadaten schreiben lassen.
Diejenigen GPS-Tags, welche den **Zielpunkt** des Fotos betreffen, müssen in jedem Fall manuell bearbeitet werden. Dazu müssen erst die entsprechenden Koordinaten ermittelt (herauslesen aus einer Karte, umrechnen in WGS84) und dann in die Foto-Metadaten geschrieben werden.

3.3.4 Brennweite und Blickwinkel

Die **Brennweite f** ist definiert als der Abstand des Brennpunkts von dem ihm zugeordneten Hauptpunkt der Bildebene. Das Zoomen der Kamera verändert die Brennweite. Eng mit der Brennweite verknüpft ist der **Blickwinkel w**, der von einem gemachten Bild wiedergegeben wird. Ebenfalls wichtig in diesem Zusammenhang ist die **Grösse d des Negativs** (belichteter Teil des Filmstreifens bei Analogkameras) bzw. des CCD-Chips (Charge-Coupled Device, lichtempfindlicher Chip bei Digitalkameras). Die drei Grössen Brennweite, Blickwinkel und Negativgrösse sind in Abb. 3 dargestellt.

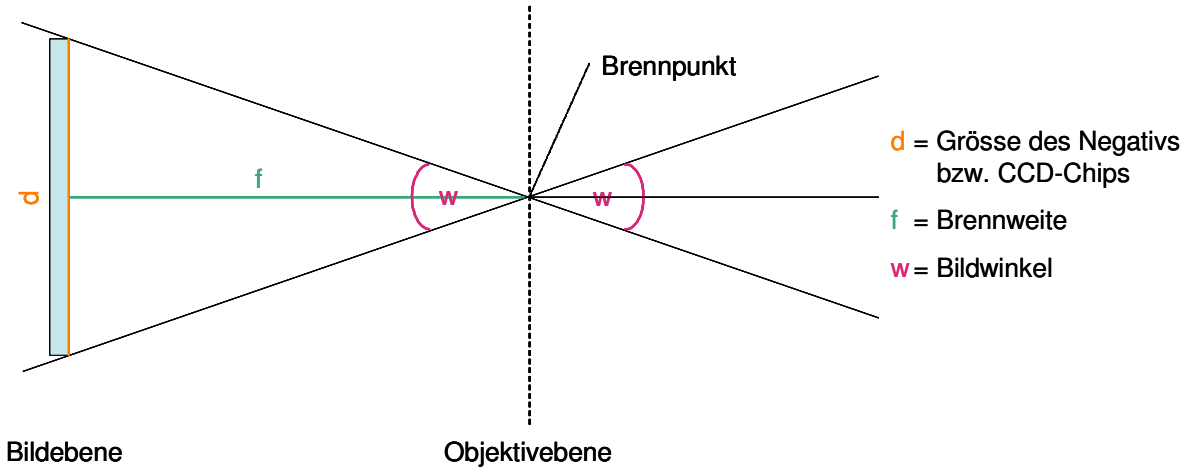


Abb. 3 Brennweite, Blickwinkel und Negativgrösse einer Kamera

Die Beziehung der drei Grössen zueinander ist mit folgender **trigonometrischer Gleichung** definiert:

$$\tan(\frac{w}{2}) = \frac{d/2}{f} \qquad \text{oder:} \qquad w = 2 \bullet \arctan(\frac{d/2}{f}) \quad .$$

Die Grösse d kann hierbei verschieden interpretiert werden, nämlich als Diagonale, Höhe oder Breite des Negativs bzw. CCD-Chips, was natürlich bei der Umrechnung zu einem entsprechend gelagerten Blickwinkel führt. In dieser Arbeit ist der horizontale Blickwinkel relevant. Es ist also darauf zu achten, dass je nach Format (Hoch- oder Quer-) des Fotos, der Blickwinkel mit der Höhe oder Breite des Negativs bzw. CCD-Chips ausgerechnet wird.
Will man nun die Brennweite einer Digitalkamera mit derjenigen einer analogen Kamera vergleichen, muss man erst festlegen, welches Filmformat als Referenz angenommen werden soll. Hier wird von einem **35 mm Normalfilm** ausgegangen, der ein Bildformat von 24 x 36 mm (Höhe x Breite) hat. Ist die Grösse des CCD-Chips bekannt, kann ihr Verhältnis zur Brennweite der Digitalkamera (f) gleichgesetzt werden zum Verhältnis der Grösse des Normalfilmbildes zur 35 mm Brennweite (f35). Die Grössen des CCD-Chips und des Normalfilmbildes werden dabei durch ihre Breite angegeben, B für die Breite des CCD-Chips und 36 für die Breite des Normalfilmbildes. Die entsprechende Gleichung lautet:

$$\frac{f}{B} = \frac{f35}{36} \qquad \text{oder:} \qquad f35 = \frac{36 \bullet f}{B} \quad .$$

Nun ist aber die Breite des CCD-Chips in den EXIF-Metadaten nicht direkt definiert, sondern muss mit dem Tag FocalPlaneXResolution, welches die Auflösung des Chips in X-Richtung in Pixel pro Längeneinheit angibt (siehe Kap. 3.3.1), berechnet werden. Die Längeneinheit ist mit dem Tag FocalPlaneResolutionUnit definiert, dem entweder der Wert mm oder inches zugewiesen ist. Die Breite des Chips ist dann gegeben als die

Anzahl Pixel der längeren Seite des Fotos (ExifImageLength oder ExifImageWidth) geteilt durch Auflösung des CCD-Chips (FocalPlaneXResolution).
Leider variiert die Wertbesetzung der einzelnen EXIF-Tags abhängig von Modell und Hersteller der Kamera sehr stark. So unterstützen einige Kameramodelle eine direkte Umrechnung der Brennweite in ein 35mm-Äquivalent (FocalLength35mmFormat) und andere nicht. Ebenso bieten nur einige Kameramodelle einen Umrechnungsfaktor (ScaleFactor35efl) an. Dieser ist aber in entsprechenden Tests als nicht verlässlich eruiert worden. Viele Kameramodelle unterstützen selbst so simple Attribute wie die oben beschriebene Auflösung des CCD-Chips (FocalPlaneXResolution) nicht.
Diese zurzeit noch **fehlende standardisierte Wertbesetzung bzw. mangelnde Unterstützung von vielen EXIF-Tags** ist ein grosses Problem, aufgrund dessen beim Perl-Skript zum Einlesen der Metadaten in eine Datenbank (siehe Kap. 4.2) viele alternative Berechnungsmethoden für das 35mm-Äquivalent der Brennweite zur Verfügung gestellt werden müssen.

3.4 Beispiel-Foto mit EXIF-Metadaten

Als Illustration der EXIF-Metadaten ist in Abb. 4 beispielhaft ein Foto mit den meisten dazu automatisch definierten EXIF-Tags und einigen selbst definierten GPS- und IPTC-Tags zu sehen.

```
ExifTool Version Number         : 5.73
File Name                       : 009.jpg
File Size                       : 1045 kB
File Modification Date/Time     : 2005:12:23 17:37:44
File Type                       : JPEG
MIME Type                       : image/jpeg
Image Description               : OLYMPUS DIGITAL CAMERA
Make                            : OLYMPUS OPTICAL CO.,LTD
Camera Model Name               : X-2,C-50Z
Orientation                     : Horizontal (normal)
X Resolution                    : 144
Y Resolution                    : 144
Resolution Unit                 : inches
Software                        : 28-1100
Date/Time Of Last Modification  : 2005:09:09 10:17:51
Y Cb Cr Positioning             : Co-sited
Shutter Speed                   : 1/200
Aperture                        : 5.6
Exposure Program                : Creative (Slow speed)
ISO Speed                       : 80
Exif Version                    : 0220
Shooting Date/Time              : 2005:09:09 10:17:51
Date/Time Of Digitization       : 2005:09:09 10:17:51
Components Configuration        : YCbCr
Max Aperture Value              : 2.8
Metering Mode                   : Multi-segment
Flash                           : Auto, Did not fire
Focal Length                    : 13.5mm
Image Quality                   : SHQ
Macro                           : Off
Black & White Mode              : Off
Digital Zoom                    : 1.0
Focal Plane Diagonal            : 8.86 mm
Firmware Version                : D4028
Camera ID                       : OLYMPUS DIGITAL CAMERA
Zoom                            : 15
Macro Focus                     : 319
Color Matrix                    : 356 65418 2 65486 374 65464 65528 65408 388
Sharpness Factor                : 640
Color Control                   : 24 1024 768 1024 16 256
Valid Bits                      : 10 0
Coring Filter                   : 2048
Olympus Image Width             : 2560
Olympus Image Height            : 1920
Flashpix Version                : 0100
Color Space                     : sRGB
Exif Image Width                : 2560
Exif Image Length               : 1920
File Source                     : Digital Camera
Custom Rendered                 : Normal
Exposure Mode                   : Auto
Digital Zoom Ratio              : 1
Scene Capture Type              : Standard
Contrast                        : Normal
Saturation                      : Normal
Sharpness                       : Normal
GPS Version ID                  : 2.2.0.0
GPS Latitude Ref                : North
GPS Latitude                    : 47 deg 17' 36.50"
GPS Longitude Ref               : East
GPS Longitude                   : 9 deg 26' 47.70"
GPS Dest Latitude Ref           : North
GPS Dest Latitude               : 47 deg 15' 12.10"
GPS Dest Longitude Ref          : East
GPS Dest Longitude              : 9 deg 20' 48.60"
Thumbnail Offset                : 5392
Thumbnail Length                : 6105
Credit                          : Daniel Meyer
Source                          : http://www.carto.net
Object Name                     : Cows in front of Ebenalp and Säntis
Keywords                        : mountain, hill, nature, summer
Copyright Notice                : Commercial use requires license. Contact: dameyer@student.ethz.ch
Image Width                     : 2560
Image Height                    : 1920
Aperture                        : 5.6
Scale Factor To 35mm Equivalent : 4.9
Circle Of Confusion             : 0.006 mm
Depth of Field                  : inf (5.26 m - inf)
Focal Length                    : 13.5mm (35mm equivalent: 65.7mm)
Hyperfocal Distance             : 5.26 m
```

Abb. 4 Beispiel-Foto mit EXIF-Metadaten

4 Datenbank

4.1 Aufbau der Datenbank

4.1.1 Allgemeines

Für die Verwaltung der Fotos und ihrer Metadaten wird eine **PostgreSQL Datenbank** aufgebaut. Das objektrelationale Datenbanksystem PostgreSQL entstand an der University of California am Berkeley Computer Science Department, ist seit 1994 ein Open-Source-Programm und wird ständig weiterentwickelt. Weitere Informationen zu PostgreSQL findet man unter [PGSQL].
Für die Datenbankanbindung von geografischen Objekten wurde im Jahr 2005 die PostgreSQL Erweiterung **PostGIS** entwickelt. PostGIS ist ebenfalls open-source und enthält alle nötigen Funktionen, um PostgreSQL als einen Datenserver für Geoinformationssysteme einzusetzen. Weitere Informationen zu PostGIS sind unter [PGIS] zu finden.
Eine grafische Benutzeroberfläche für die Arbeit mit einer PostgreSQL Datenbank ist das frei verfügbare Programm **pgAdmin** [PGADM]. Mit diesem Programm kann man komplexe Datenbanken entwickeln und SQL-Abfragen ausführen.

4.1.2 Erstellte Tabellen

In der Datenbank werden folgende Tabellen kreiert:

- photos
- users
- user_aliases
- keywords
- keyword_aliases
- keywords_photos
- kantone
- bezirke
- gemeinden
- geometry_columns
- spatial_ref_sys

Die Spalten der **Tabelle photos** entsprechen ungefähr den ausgewählten EXIF-Metadaten (siehe Kap. 3.3.1). Zusätzlich werden die Spalten PhotoID, UserID, FileName und AngleOfView definiert.
Die UserID ergibt sich aus den beiden **Tabellen users**, in welcher alle user mit Adresse aufgeführt sind und **user_aliases**, in welcher verschiedene Schreibweisen des user-Namens zusammengefasst sind.
Die geografische Einordnung der Fotos erfolgt über die Koordinaten des Aufnahmestandpunktes. Diese werden vom GPS-Projektionssystem WGS84 zum schweizerischen Projektionssystem CH1903+ umgerechnet, wofür die **Tabelle spatial_ref_sys** die nötigen Definitionen liefern. Die Zuordnung des Aufnahmestandpunktes zu einem Kanton, Bezirk

und Gemeinde wird über die Tabelle geometry_columns und gemeinden realisiert. Die **Tabelle gemeinden** enthält neben geometrischen Definitionen auch die beiden Spalten bezirksnr und kantonsnr, über welche eine Verbindung zu den **Tabellen bezirke und kantone** hergestellt wird.
Das EXIF-Tag Keywords kann nicht wie die anderen einfach einer Spalte der Tabelle photos zugewiesen werden, da hier mehrere Einträge möglich sind. Es muss also eine separate **Tabelle keywords** mit den beiden Spalten keyword und id erstellt werden und eine **Verbindungstabelle keywords_photos**, welche die Spalten photo_id und keyword_id enthält. Zusätzlich wird die **Tabelle keyword_aliases** definiert, welche den einzelnen Keywords alternative Begriffe zuordnet (siehe Kap. 3.3.2). Alle erstellten Tabellen und ihre Beziehungen zueinander sind in Abb. 5 dargestellt.

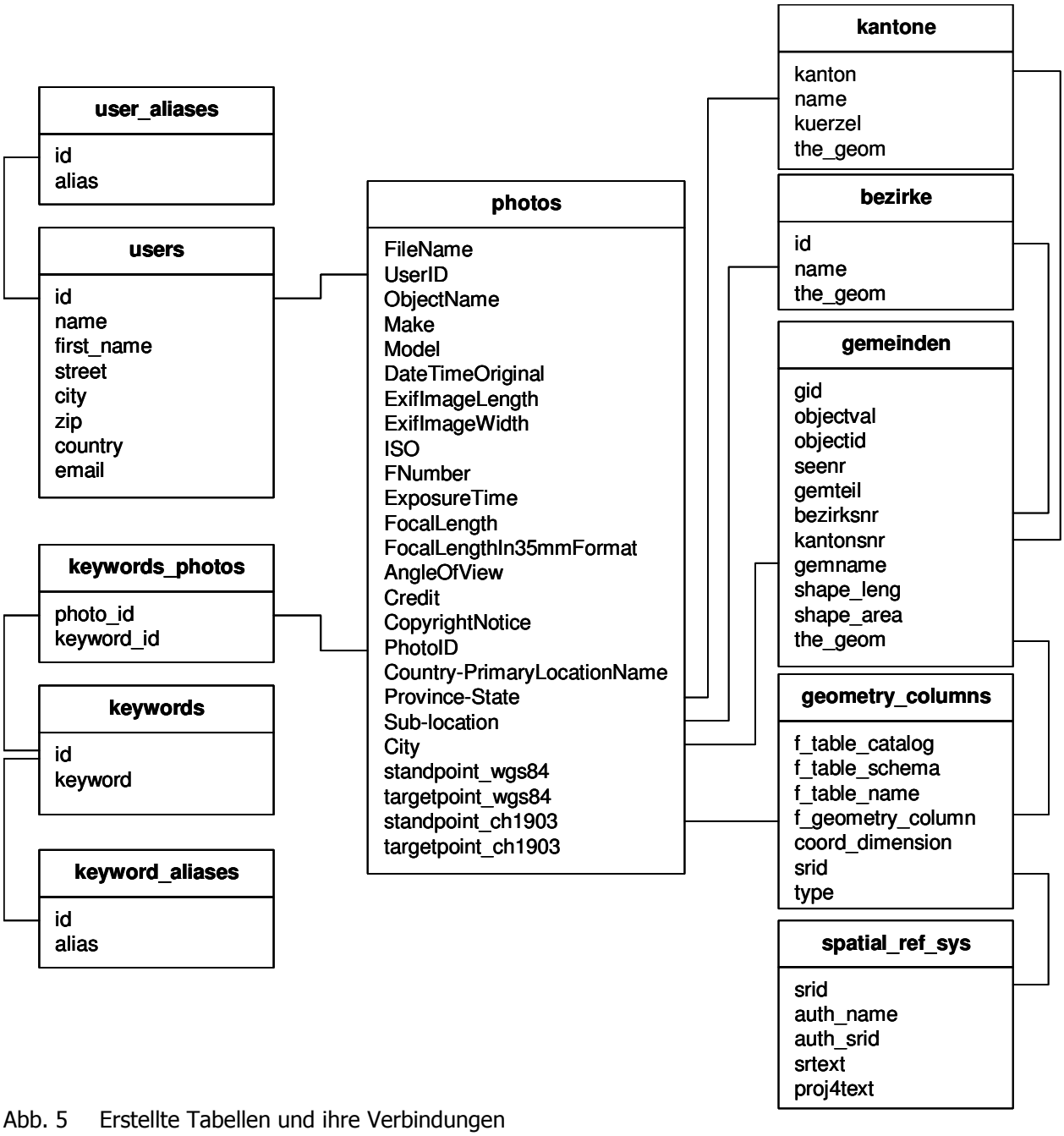


Abb. 5 Erstellte Tabellen und ihre Verbindungen

4.2 Perl-Skript zum Einlesen der EXIF-Metadaten

4.2.1 Allgemeine Informationen zu Perl

Für das automatische Einlesen der EXIF-Metadaten aus den Bilddateien in die Datenbank wird ein Perl-Skript erstellt. **Perl** ist eine freie und plattformunabhängige Programmiersprache, die ursprünglich zur System- und Netzwerkadministration entwickelt wurde. Eine Dokumentation zu Perl ist unter [PERL] zu finden. Eine frei benutzbare und umfassende Sammlung von Perl-Modulen (wie z. B. das ExifTool) findet man unter [CPAN].

4.2.2 exif.pl

Der gesamte Inhalt des selbst erstellten **Perl-Skripts exif.pl** ist in Anhang Kap. 1.2 aufgeführt. Im Folgenden werden das Skript zusammengefasst und einige Elemente erläutert.

Zuerst muss eine Verbindung zu der PostgreSQL Datenbank hergestellt werden.

```
my $dbh = DBI->connect($db, $db_user, $db_pass) || die "Error connecting to
the database: $DBI::errstr\n";
```

Als nächster Schritt wird das spezifizierte Verzeichnis mit den zu übergebenden Fotos geöffnet.

```
GetOptions("dir=s" => \$dir);
opendir(DIR, $dir) or die "can't open directory $dir: $!";
```

Die einzelnen Dateien werden als erstes auf ihre Endung geprüft, womit sichergestellt wird, dass es sich wirklich um Bilder handelt.

```
my ($base, $dir, $ext) = fileparse($file,'\..*');
if ($ext eq ".jpg" || $ext eq ".JPG" || $ext eq ".jpeg" || $ext eq ".JPEG")
```

Dann werden mithilfe des ExifTool-Moduls einzelne EXIF-Tags eingelesen und den entsprechenden Variablen zugewiesen.

```
my $exifTool = new Image::ExifTool;
my $imgInfo = $exifTool->ImageInfo($file,"Make",...)
my @tags = $exifTool->GetRequestedTags();
$make = Encode::encode("utf8",$exifTool->GetValue($tags[0]));
```

Einige Tags bedürfen für die reibungslose Weiterverwendung noch einer Bearbeitung. Zum Beispiel hat das Tag DateTimeOriginal einen Zeitzonen-Zusatz, der weggeschnitten werden soll.

```
my @dateTime = split(/\./,$dateTimeOriginal);
$dateTimeOriginal = $dateTime[0];
```

Als nächstes werden die Koordinaten mithilfe von Definitionen aus der Datenbank vom Projektionssystem WGS84 in CH1903+ umgerechnet.

```
$mySQL = "SELECT AsText(TRANSFORM(SetSRID(GeometryFromText('POINT($lon
$lat)'),$inCoorSystem),$outCoorSystem)) AS coord;";
$sth = $dbh->prepare($mySQL);
$sth->execute or db_err("Unable to execute query", $dbh->errstr);
while (@data = $sth->fetchrow_array()) {
   $chcoord = $data[0];
}
```

Nun erfolgt die geografisch-politische Einordnung der Fotos. Dies geschieht über die Koordinaten des Aufnahmestandortes und der Geometrie-Definitionen der Gemeinden sowie den Verknüpfungen zu Bezirken und Kantonen aus der Datenbank.

```
$mySQL = "SELECT gemeinden.gemname,bezirke.name,kantone.name from
gemeinden,bezirke,kantone WHERE
Contains(the_geom,setSRID(GeometryFromText('$chcoord'),$outCoorSystem)) AND
gemeinden.bezirksnr = bezirke.id AND gemeinden.kantonsnr =
kantone.kanton;";
$sth = $dbh->prepare($mySQL);
$sth->execute or db_err("Unable to execute query", $dbh->errstr);
while (@data = $sth->fetchrow_array()) {
   $city = replaceChars($data[0]);
   $subLocation = replaceChars($data[1]);
   $provinceState = replaceChars($data[2]);
}
```

Dann werden Unterordner für die Ablage der Fotos erstellt, sofern sie nicht bereits bestehen. Jedes Foto wird dabei folgendermassen abgelegt: Switzerland\Kanton\Bezirk\ Gemeinde\Foto, wobei der Name des Fotos zusammengesetzt wird aus den Tags DateTimeOriginal und ObjetName.

```
my $kantondir = $photodir.$provinceState."/";
unless (-d $kantondir) { #see if directory exists
   mkdir($kantondir);
}
...
$newfile =
$gemeindedir.replaceChars($dateTimeOriginal).'_'.replaceChars($objectName).
$ext;
copy($file,$newfile) or die "could not copy $file";
```

Zusätzlich werden in das gleiche Verzeichnis auch zwei Thumbnails (Kleinansichten) abgelegt, welche später in der Webseite für die Anzeige der Bilder genutzt werden.

```
$status = system("convert '".$file."' -resize ".$thumbwidth."x".
$thumbheight." ".$thumbfile);
```

In einem nächsten Schritt wird, falls nicht in den EXIF-Metadaten bereits definiert, das 35mm-Äquivalent der Brennweite berechnet und daraus der Blickwinkel abgeleitet. Erklärungen zu diesem Schritt sind in Kap. 3.3.4 zu finden.

Anschliessend wird über das Tag Credit, in welchem der Name des Fotografen definiert ist, versucht die entsprechende User-ID aus der Datenbank zu finden.

```
$mySQL = "SELECT id from user_aliases WHERE alias LIKE '%$credit%';";
```

Dann werden alle EXIF-Daten, die das Programm bearbeitet hat ausgegeben.und in die Datenbank geschrieben.

```
$mySQL = qq(INSERT INTO photos ("Make",...) VALUES ('$make',... ););
$sth = $dbh->prepare($mySQL);
$sth->execute or db_err("Unable to insert foto data", $dbh->errstr);
```

Anschliessend müssen noch die Schlagwörter, die im Tag Keywords definiert sind, in die Datenbank geschrieben werden. Die Keywords werden in einer separaten Tabelle abgelegt und durch ID's mit den Fotos verbunden. Das Skript stellt sicher, dass keinem Foto mehrmals identische Schlagwörter zugewiesen sind.

```
#get current photo id
$mySQL = qq(SELECT currval('create_photo_id'));
...
#split keywords
my @Keywords = split(/\s*,\s*/,$keywords);
...
foreach $value (@Keywords) {
   ...
   $mySQL = "SELECT id FROM keyword_aliases WHERE alias = '$value';";
   ...
   if ($keywordId != -99 && $unique == 1) {
      $mySQL = qq(INSERT INTO keywords_photos VALUES ($photoId,$keywordId));
   ...
   }
   push(@KeywordIDs,$keywordId);
}
```

Zum Schluss wird noch ausgegeben, dass die Informationen nun in die Datenbank geschrieben wurden mit dem Hinweis, wie viele Fehler dabei aufgetreten sind.

```
print "All informations have been written into database. $error errors
occured.\n";
print "Processing photo ".($imgCounter + 1)." completed.\n\n\n";
```

5 Erstellen der Webseite

5.1 Eingesetze Techniken

5.1.1 XML und SVG

XML steht für Extensible Markup Language und ist ein vom World Wide Web Consortium (W3C) definierter Standard für die Erstellung von maschinen- und menschenlesbaren Dokumenten in Form einer Baumstruktur. XML gibt dabei die Regeln für den Aufbau solcher Dokumente vor, definiert also die Strukturelemente und ihre Anordnung innerhalb des Dokumentenbaums. Auf der Grundlage dieser Regeln existieren verschiedene Auszeichnungssprachen für unterschiedliche Anwendungsbereiche. Weitere Informationen zu XML sind unter [XML] zu finden.
Eine der auf XML-Syntax basierenden Auszeichnungssprachen ist SVG. SVG steht für Scalable Vector Graphics und ist ein ebenfalls vom W3C definierter Standard zur Beschreibung von zweidimensionalen Vektorgrafiken. Die Vektorgrafiken werden dabei durch Beschreibungstexte definiert und können mit einem gewöhnlichen Texteditor bearbeitet werden. Weitere Informationen zu SVG erhält man unter [SVG] und [CARTO].
XML und SVG werden als Grundgerüst für die erstellte Webseite eingesetzt.

5.1.2 ECMAScript

ECMA steht für European Computer Manufacturers' Association und bezeichnet ein Gremium zur Standardisierung von Sprachen, Formaten oder Schnittstellen im IT-Bereich. Der Begriff ECMAScript wird für den standardisierten Sprachkern von JavaScript verwendet. JavaScript ist die am weitesten verbreitete clientseitige und objektbasierte Skriptsprache zur dynamischen Gestaltung von Webseiten. Die ECMAScript Spezifikation ist unter [ECMA] zu finden.
ECMAScript wird für die Ermöglichung von dynamischen Visualiserungen und Benutzerinteraktionen der Webseite eingesetzt.

5.1.3 PHP und SQL

PHP steht für Hypertext Preprocessor und benennt eine weit verbreitete serverseitige Skriptsprache, welche generell zur Erstellung von dynamischen Webseiten angewandt wird und sich insbesondere für Datenbankzugriffe eignet. Weitere Informationen zu PHP sind unter [PHP] zu finden.
SQL steht für Structured Query Language und ist eine deklarative Datenbanksprache. Mit SQL werden Datenstrukturen definiert, Datenbestände manipuliert und abgefragt. Der SQL-Sprachstandard wird von den meisten Datenbanksystemen genutzt.
PHP und SQL werden für die Anbindung der erstellten Datenbank an die Webseite eingesetzt.

5.2 Benutzte Vorlagen

Als Grundlage für die Webseite wird der von Andreas Neumann erstellte **Geovite Data Browser** [GEOVITE] übernommen. Mit dieser Grundlage sind die interaktive Karte der Schweiz und die Kartennavigationsinstrumente realisiert. Für weitere Funktionalitäten wie Selektionslisten und Checkboxen existieren unter [CARTO] auch entsprechende Vorlagen, welche eingesetzt werden.

5.3 Dateiüberblick

Die Webseite nutzt insgesamt 28 Dateien, welche nachfolgend aufgelistet sind:

- HTML-Datei: index.html
- SVG-Dateien: index.svg, referenceMap.svg
- XML-Datei: geophotos_map.xml
- CSS-Datei: styles.css
- ECMAScripte: button.js, checkbox.js, flowText.js, geographic_projections.js, helper_functions.js, initMap.js, loadData.js, mapApp.js, navigation,js, photos.js, scalebar.js, search.js, selectionList.js, slider.js, timer.js, window.js
- PHP-Dateien: sendPhotoData.php, extractAdminUnits.php, sendSingleAdminData.php, connect_vars.php
- Weitere Dateien: relief.jpg, ETH_Logo.png, Bericht.pdf

Die zentrale Datei ist **index.svg**, in der alle ECMAScripte eingebettet sind. Sie wird beim Start der Applikation vom Browser aufgerufen. Die Datei **geophotos_map.xml** wurde erstellt, um Definitionen des Layouts von anderen Inhalten zu separieren.
In der Datei **styles.css** sind alle Formatierungsdefinitionen aufgeführt, welche von verschiedenen Text- und Geometrieelementen genutzt werden. Die meisten ECMAScripte wurden von Vorlagen übernommen und entsprechend den Funktionalitäten des Projekts angepasst. Vollständig neu erstellt und zentral für diese Arbeit sind die Dateien **photos.js** und **search.js**. Ebenfalls neu erstellt sind die PHP-Dateien. Im Folgenden wird beispielhaft je ein ECMAScript (photos.js) und eine PHP-Datei (**sendPhotoData.php**) erläutert (siehe Kap. 5.4.2 und 5.4.3). Für weitere Erklärungen wird auf [CARTO] verwiesen.

5.4 Visualisierung der Fotos

5.4.1 Funktionsüberblick

Beim Aufrufen der Webseite sowie bei jeder Kartennavigation wird neben anderen Funktionen die **Funktion loadPhotoData()** der Datei loadData.js aufgerufen. Diese Funktion stellt einen URL (Uniform Resource Locator) zusammen, welcher die Datei sendPhotoData.php mit einigen Parametern aufruft. Die Parameter sind ein timestamp,

die Eckkoordinaten des aktuellen Kartenausschnitts sowie die selektierten Schlagwörter und die logische Verknüpfungsoperation der thematischen Suche. Zudem wird die Funktion receivePhotoData() der Datei photos.js aufgerufen.

```
function loadPhotos() {
   //check keywords
   var myKeywords = new Array();
   for (var checkBox in myMapApp.KWCheckBoxes) {
      if (myMapApp.KWCheckBoxes[checkBox].checkedStatus) {
         myKeywords.push(checkBox.replace(/KW/,""));
      }
   }
   var myPhotoDataUrl =
"sendPhotoData.php?xmin="+myMainMap.curxOrig+"&ymin="+((myMainMap.curyOrig
+ myMainMap.curHeight)* -1)+"&xmax="+(myMainMap.curxOrig+myMainMap.
curWidth)+"&ymax="+(myMainMap.curyOrig * -1)+"&keywords="+myKeywords.
join(",")+"&logicalAndOr="+myMapApp.logicalAndOr+"&timestamp="+myMainMap.
timestamp;
   var getPhotoDataObj = new getXMLData(myPhotoDataUrl,receivePhotoData);
   getPhotoDataObj.getData();
}
```

Die Datei **sendPhotoData.php** schreibt basierend auf den Parametern des aufgerufenen URL eine xml-Datei mit allen Informationen zu den gewünschten Fotos. Als Erläuterung wird nachfolgend beispielhaft ein URL und die dazu erstellte xml-Datei angegeben.

```
URL: http://www.carto.net/geofotos/sendPhotoData.php?xmin=486000
&xmax=833000&ymin=75000&ymax=295000&timestamp=123&logicalAndOr=AND
&keywords=lake,atmosphere
```

```
- <photos timestamp="123">
   - <photo id="1384"
      FileName="photos/Switzerland/Zug/Zug/Zugersee__ZG_/
      20041018_093834_Landsgemeindeplatz_Zug_and_view_to_the_Alps.jpg"
      ObjectName="Landsgemeindeplatz Zug and view to the Alps" Make="OLYMPUS
      OPTICAL CO.,LTD" Model="X-2,C-50Z" DateTime="2004-10-18 09:38:34"
      Length="1536" Width="2048" Exposure="1/320" Aperture="5.6" ISO="80"
      FocalLength35="47" UserFirstName="Daniel" UserName="Meyer"
      Copyright="Commercial use requires license. Contact
      dameyer@student.ethz.ch" StandPointX="681529" StandPointY="224785"
      TargetPointX="677193" TargetPointY="218065" AngleOfView="41.9116"
      ProvinceState="Zug" SubLocation="Zug" City="Zugersee (ZG)">
      - <keywords>
            <keyword>autumn</keyword>
            <keyword>lake</keyword>
            <keyword>mountain</keyword>
            <keyword>atmosphere</keyword>
         </keywords>
      </photo>
   </photos>
```

Schliesslich liest die **Funktion receivePhotoData()** der Datei photos.js die Informationen dieser xml-Datei ein und weist sie entsprechenden Variablen zu.

5.4.2 photos.js

Der gesamte Quelltext dieser Skript-Datei ist im Anhang Kap. 1.3 aufgeführt. Hier werden nur die Hauptfunktionen kurz erläutert und einige Auszüge aus dem Quelltext wiedergegeben.

Am Anfang der Datei wird die **Funktion receivePhotoData()** definiert, welche die Metadaten der im aktuellen Kartenausschnitt sichtbaren Fotos einliest. Zudem wird die Funktion drawCameraSymbols() aufgerufen. Falls ein Foto selektiert ist, wird auch die Funktion drawSelectedPhotoDetail() aufgerufen, ansonsten die Funktion displayPhotosSmall().

```
function receivePhotoData(node) {
   ...
   var child = node.firstChild;
   myMapApp.photoData = new Array();
   while(child) {
      if (child.nodeType == "1") {
         if (child.nodeName == "photo") {
            var id = "id_"+child.getAttributeNS(null,"id");
            var FileName = child.getAttributeNS(null,"FileName");
            ...
         }
         child = child.nextSibling;
      }
      drawCameraSymbols();
      if (myMapApp.photoSelectStatus) {
         drawSelectedPhotoDetail();
      }
      else {
         displayPhotosSmall("start",undefined);
      }
   }
}
```

Die **Funktion drawCameraSymbols()** zeichnet abhängig vom aktuellen Kartenmassstab an den Aufnahmekoordinaten der Fotos ein Punkt- oder Kamerasymbol. Die Symbole werden beim Zoomen der Karte jeweils zurückskaliert, sodass sie absolut gesehen stets gleich gross erscheinen. Zusätzlich werden den Symbolen Mausinteraktionen zugewiesen.

```
function drawCameraSymbols() {
   ...
   //create temporary group for photo symbols
   var photoSymbolsTempGroup = document.createElementNS(svgNS,"g");
   photoSymbolsTempGroup.setAttributeNS(null,"id","photoSymbolsTemp");
   photoSymbolGroup.appendChild(photoSymbolsTempGroup);

   //update symbol sizes
   document.getElementById("cameraSymbGeom").setAttributeNS(null,
"transform","scale("+(myMainMap.curWidth * 0.002)+")");

   for (var i=0;i<myMapApp.photoData.length;i++) {
      var photoSymbol = document.createElementNS(svgNS,"use");
      if (myMainMap.curWidth > 10000) {
         photoSymbol.setAttributeNS(xlinkNS,"href","#cameraSmall");
```

```
      }
      else {
        photoSymbol.setAttributeNS(xlinkNS,"href","#camera");
        var direction = ...
      }
      photoSymbol.setAttributeNS(null,"x",myMapApp.photoData[i].value.
StandPointX);
      photoSymbol.setAttributeNS(null,"y",myMapApp.photoData[i].value.
StandPointY * -1);
      photoSymbol.setAttributeNS(null,"id","photoSymbol_"+
myMapApp.photoData[i].value.id);
      photoSymbol.addEventListener("mouseover",showPhotoData,false);
      photoSymbol.addEventListener("mouseout",showPhotoData,false);
      photoSymbol.addEventListener("click",setPhotoExtentCheck,false);
      photoSymbol.addEventListener("click",showPhotoData,false);
      photoSymbolsTempGroup.appendChild(photoSymbol);
    }
}
```

Die **Funktion displayPhotosSmall(groupId,evt)** bewirkt, dass in der Foto-Galerie jeweils fünf Kleinansichten von Fotos des aktuellen Kartenausschnitts angezeigt werden. Zudem werden zwei Buttons generiert, welche zum Blättern in der Fotogalerie dienen. Die Kleinansichten müssen abhängig von ihrem Format (hoch oder quer) platziert und mit Mausinteraktionen versehen werden.

```
function displayPhotosSmall(groupId,evt) {
   ...
   if ((myMapApp.photoGalleryOffset + myMapApp.nrPerPage) <
myMapApp.photoData.length) {
      photoGalleryEnd = myMapApp.photoGalleryOffset + myMapApp.nrPerPage;
      myMapApp.buttons["photoGalForward"].activate();
   }
   else {
      myMapApp.buttons["photoGalForward"].deactivate();
   }
   if (myMapApp.photoGalleryOffset == 0) {
      myMapApp.buttons["photoGalBackward"].deactivate();
   }
   else {
      myMapApp.buttons["photoGalBackward"].activate();
   }

   //arrange thumbnails
   var x = 35;
   var photoGap = 18;
   var bigLength = 120;
   for (var i = myMapApp.photoGalleryOffset;i<photoGalleryEnd;i++) {
      ...
      photosSmall.setAttributeNS(null,"id","photoSmall_"+curPhoto.id);
      photosSmall.setAttributeNS(xlinkNS,"href",fileSmall);
      photosSmall.addEventListener("mouseover",showPhotoData,false);
      ...
      if (curPhoto.Width > curPhoto.Height) {
        //case landscape
        photosSmall.setAttributeNS(null,"width",bigLength);
        photosSmall.setAttributeNS(null,"height",bigLength *
(curPhoto.Height / curPhoto.Width));
```

```
        y += (bigLength - bigLength * (curPhoto.Height / curPhoto.Width))/2;
        photosSmall.setAttributeNS(null,"x",x);
        photosSmall.setAttributeNS(null,"y",y);
        x += bigLength + photoGap;
      }
      else {
        //case portrait
        ...
      }
      photosSmallTempGroup.appendChild(photosSmall);
    }
}
```

Wenn ein einzelnes Foto ausgewählt wird, was entweder durch ein Klick auf ein Kamerasymbol auf der Karte oder auf eine Kleinansicht in der Foto-Galerie geschieht, ruft das unter anderem die **Funktion displaySinglePhoto(id)** auf. Diese Funktion blendet die Fotogalerie aus und präsentiert an ihrer Stelle eine etwas grössere Ansicht des selektierten Fotos mit zusätzlichen Metainformationen und drei Buttons („Zoom to this photo“, „See photo in higher resolution“ und „Return to photo gallery“). Ein Klick auf diese Buttons löst dann wiederum entsprechende Funktionen aus.

```
function displaySinglePhoto(id) {
   ...
   var curPhoto = myMapApp.photoData[id].value;
   file = curPhoto.FileName.split(".");
   var ext = file[1];
   var fileSmall = file[0] + "_thumb200." + ext;
   var singlePhotoSmall = document.createElementNS(svgNS,"image");
   ...
   //create temporary group for photo text
   var photoTextTempGroup = document.createElementNS(svgNS,"g");
   photoTextTempGroup.setAttributeNS(null,"id","photoTextTemp");
   photoTextTempGroup.setAttributeNS(null,"pointer-events","none");
   myMapApp.photoGalleryGroup.appendChild(photoTextTempGroup);
   var standPoint = document.createElementNS(svgNS,"text");
   standPoint.setAttributeNS(null,"x",230);
   standPoint.setAttributeNS(null,"y",30);
   standPoint.setAttributeNS(null,"class","allText smallText");
   var textNode = document.createTextNode("Standpoint Coordinates: Y " +
   curPhoto.StandPointX + "m, X " + curPhoto.StandPointY + "m");
   standPoint.appendChild(textNode);
   photoTextTempGroup.appendChild(standPoint);
   ...
}
```

Wenn ein einzelnes Foto ausgewählt ist, wird auch sein Blickfächer auf dem Kartenausschnitt gezeichnet, was durch die **Funktion drawSelectedPhotoDetail()** realisiert wird. Der Blickfächer wird dabei in SVG als elliptischer Bogen definiert und als Pfad-Element aufgerufen. Die dabei anzuwendenden Parameter sind in Abb. 6 erläutert. Dabei stehen die beiden Zahlen nach M (move to) für die Startkoordinaten des Bogens, die beiden Zahlen nach a für die Halbachsen der vom Bogen beschriebenen Ellipse und die letzten beiden Zahlen für die Endkoordinaten des Bogens.

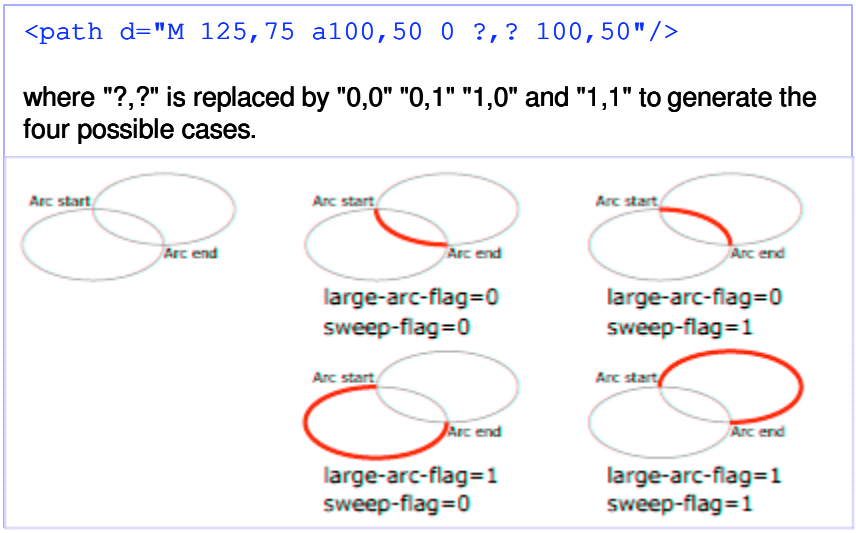


Abb. 6 Definition eines elliptischen Bogens in SVG
Quelle: http://www.w3.org/TR/SVG/paths.html

Die Start- und Endkoordinaten des Bogens werden trigonometrisch aus Blickdistanz und Blickrichtung des Fotos berechnet, welche wiederum aus den Koordinaten des Stand- und Zielpunkts des Fotos bestimmt werden.

```
function drawSelectedPhotoDetail() {
   ...
   //only draw if the data is available
   if (myMapApp.photoData[id]) {
      ...
      var viewAngle = document.createElementNS(svgNS,"path");
      viewAngle.setAttributeNS(null,"stroke","darkred");
      viewAngle.setAttributeNS(null,"stroke-width",(myMainMap.curWidth *
0.001));
      viewAngle.setAttributeNS(null,"fill","darkred");
      viewAngle.setAttributeNS(null,"fill-opacity","0.2");
      var d = "M"+curPhoto.StandPointX+","+(curPhoto.StandPointY*-1)+"L"+
myMapApp.selPhotoPointX1+","+myMapApp.selPhotoPointY1+"A"+myMapApp.
selPhotoDistance+","+myMapApp.selPhotoDistance+" 0 0,1 "+
myMapApp.selPhotoPointX2+" "+myMapApp.selPhotoPointY2+"z";
      viewAngle.setAttributeNS(null,"d",d);
      photoSymbolDetailTempGroup.appendChild(viewAngle);
      var connectLine = document.createElementNS(svgNS,"line");
      ...
      var photoSymbol = document.createElementNS(svgNS,"use");
      ...
      photoSymbol.setAttributeNS(xlinkNS,"href","#camera");
      photoSymbol.setAttributeNS(null,"transform","rotate("+(myMapApp.
selPhotoDirection*-1)+","+curPhoto.StandPointX+","+(curPhoto.StandPointY *
-1)+")");
      ...
   }
}
```

5.4.3 sendPhotoData.php

Der gesamte Inhalt dieser PHP-Datei ist im Anhang Kap. 1.4 aufgeführt. Im Folgenden werden die Hauptbestandteile des Skripts erläutert.
Beim Aufruf der Datei durch den entsprechenden URL (siehe Kap. 5.4.1) werden die dort definierten Parameter eingelesen. Die Koordinaten müssen noch umgerechnet werden, da die Landeskarten im bisherigen schweizerischen **Projektionssystem** CH1903 (LV03, Fundametalpunkt in alter Sternwarte Bern) und die Koordinaten in der Datenbank (standpoint_ch1903, targetpoint_ch1903 und Gemeindegeometrien) im neuen schweizerischen Projektionssystem CH1903+ (LV95, Fundamentalpunkt in Geostation Zimmerwald) definiert sind. Anschliessend wird eine Verbindung zu der Datenbank mit den Foto-Metadaten hergestellt.

```
//get parameters
$xmin = intval($_GET['xmin'])+2000000;
$xmax = intval($_GET['xmax'])+2000000;
$ymin = intval($_GET['ymin'])+1000000;
$ymax = intval($_GET['ymax'])+1000000;
$andOr = $_GET['logicalAndOr'];
$keywordsString = $_GET['keywords'];
$keywords = explode(",",$keywordsString);
$timestamp = $_GET['timestamp'];
...
//connect to db
$my_pg_connect = pg_Connect('host='.$hostname.' dbname='.$dbName.'
user='.$username.' password='.$password) or die ('Cannot connect to
database "'.$dbGeomName.'"');
```

Dann wird, abhängig davon, ob Keywords im URL definiert sind, eine **SQL-Abfrage an die Datenbank** zusammengestellt für die benötigten Metadaten der gewünschten Bilder. Sind Keywords definiert, wird zusätzlich nach logischer Verknüpfungsoperation (OR, AND) unterschieden. Die SQL-Abfragen sind lange und komplizierte Gebilde und sollen nun erklärt werden. Für die Tabellen der Datenbank und ihre Spalten beachte man die Ausführungen in Kap. 4.1.2.

Komplette SQL-Abfrage ohne Keywords:

```
SELECT DISTINCT "PhotoID", "FileName", "ObjectName", "Make", "Model",
"DateTimeOriginal" AS datetime, "ExifImageLength", "ExifImageWidth",
"ExposureTime", "FNumber", "ISO", "FocalLengthIn35mmFormat", "UserID",
"CopyrightNotice", astext("standpoint_ch1903") AS StandPoint,
astext("targetpoint_ch1903") AS TargetPoint, "AngleOfView", "Province-
State", "Sub-location", "City" FROM photos WHERE standpoint_ch1903 &&
setSRID(\'BOX3D('.$xmin.' '.$ymin.', '.$xmax.'
'.$ymax.')\'::box3d,'.$srid.') ORDER BY "DateTimeOriginal" DESC
```

Hier führt die einzige **Bedingung** (WHERE standpoint_ch1903 && …) dazu, dass nur die Metadaten der Fotos im aktuellen Kartenausschnitt eingelesen werden. Die Fotos werden dabei nach absteigendem Aufnahmezeitpunkt **sortiert** (ORDER BY „DateTimeOriginal“ DESC). SELECT DISTINCT verhindert mehrfache Selektion der gleichen Informationen.

Müssen nun zusätzlich die ausgewählten **Schlagwörter** als Selektionskriterien berücksichtigt werden, kompliziert dies die Abfrage. Bei den nachfolgenden Ausführungen wird nur noch das Attribut „PhotoID“ stellvertretend für alle anderen Metadaten der Tabelle photos selektiert. Die ausgewählten Keywords werden vereinfachend durch A, B und C repräsentiert. Für einen besseren Überblick ist in Abb. 7 eine **schematische Darstellung der relevanten Tabellen** der Datenbank gegeben.

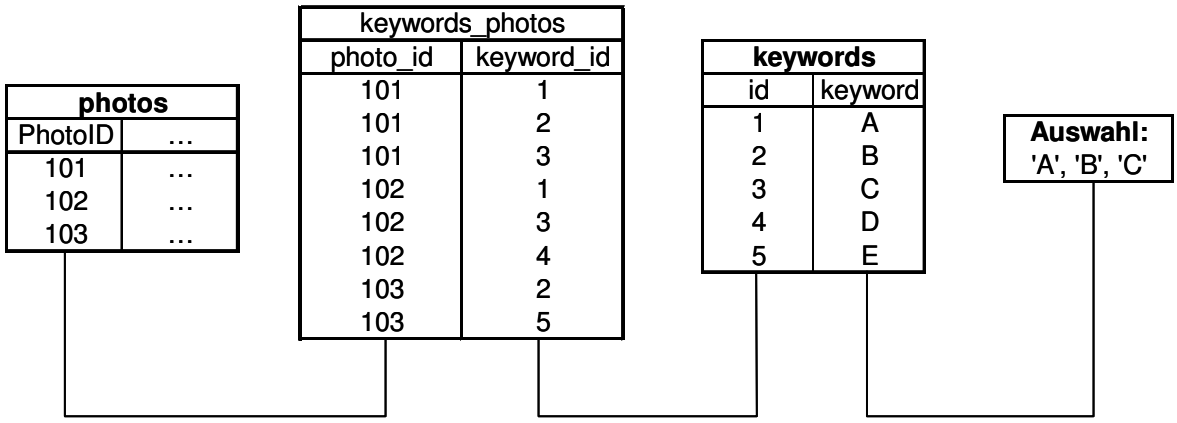


Abb. 7 Schematische Darstellung zur Keyword-Abfrage

Verkürzte SQL-Abfrage mit OR-Verknüpfung der Keywords:

```
SELECT DISTINCT photos."PhotoID" FROM photos, keywords_photos, keywords
WHERE photos.standpoint_ch1903 && setSRID(\'BOX3D('.$xmin.' '.$ymin.',
'.$xmax.' '.$ymax.')\'::box3d,'.$srid.') AND photos."PhotoID" =
keywords_photos.photo_id AND keywords_photos.keyword_id = keywords.id AND
keywords.keyword IN ('A','B','C') ORDER BY photos."DateTimeOriginal" DESC'
```

Die zusätzlichen Bedingungen verknüpfen die relevanten Tabellen der Datenbank und wählen die gewünschten Keywords aus der Tabelle keywords aus, indem die Keyword-Texte verglichen werden. Der Ausdruck <keyword IN ('A','B','C')> ist dabei äquivalent zu <keyword = 'A' OR keyword = 'B' OR keyword = 'C'>.

Verkürzte SQL-Abfrage mit AND-Verknüpfung der Keywords:

```
SELECT DISTINCT photos."PhotoID" FROM photos, keywords_photos WHERE
photos.standpoint_ch1903 && setSRID(\'BOX3D('.$xmin.' '.$ymin.', '.$xmax.'
'.$ymax.')\'::box3d,'.$srid.') AND photos."PhotoID" = keywords_photos.
photo_id AND keywords_photos.photo_id IN (SELECT keywords_photos.photo_id
FROM keywords_photos,keywords WHERE keywords_photos.keyword_id =
keywords.id AND keywords.keyword IN ('A','B','C') GROUP BY
keywords_photos.photo_id HAVING count(keywords_photos.keyword_id) = 3)
ORDER BY photos."DateTimeOriginal" DESC';
```

Die Realisierung der AND-Verknüpfung ist ein wenig diffizil, da mit Abfragen stets einzelne Tabellenzeilen angesprochen werden und keine Zeile mehrere Keywords enthält. So muss zuerst eine **Vorselektion** durchgeführt werden, welche das Attribut

photo_id der Tabelle keywords_photos gruppiert und dann durch den Befehl count dessen Vorkommen zählt. Entspricht die Häufigkeit dabei der Anzahl ausgewählter Keywords, ist garantiert, dass dem entsprechenden Foto alle ausgewählten Keywords zugewiesen sind. Dies bedingt jedoch, dass kein Keyword mehrmals dem gleichen Foto zugewiesen ist, was im Perl-Skript zum Einlesen der EXIF-Metadaten (siehe Kap. 4.2.2) kontrolliert wird. Das Resultat der Vorselektion angewandt auf das schematische Beispiel von Abb. 7 ist in Abb. 8 dargestellt.

```
SELECT keywords_photos.photo_id FROM
keywords_photos, keywords WHERE
keywords_photos.keyword_id = keywords.id
AND keywords.keyword IN ('A', 'B', 'C,)
GROUP BY keywords_photos.photo_id
```

keywords_photos	
photo_id	Anzahl
101	3
102	2
103	1

Abb. 8 SQL-Abfrage für AND-Verknüpfung der Keywords

Anschliessend an die Zusammenstellung der SQL-Hauptabfrage wird eine zusätzliche Abrage für den **Namen des Fotografen** definiert basierend auf dem Attribut UserID der Tabelle photos.

```
$mySQLuser = 'SELECT users."first_name", users."name" FROM users WHERE
users."id" = '.$resultArray['UserID'];
```

Dann werden die bisher von der Datenbank abgerufenen Metainformationen als **xml-Code** ausgegeben.

```
print '<photo id="'.$resultArray['PhotoID'].'" FileName="'.str_replace("/
home/meyer/www/geofotos/","",$resultArray['FileName']).'"
ObjectName="'.htmlspecialchars($resultArray['ObjectName'],ENT_QUOTES).'"
Make="'.$resultArray['Make'].'" Model="'.$resultArray['Model'].'" ...
```

Zum Schluss werden noch die **Keywords** abgefragt und ebenfalls ausgegeben.

```
$mySQLkeywords = 'SELECT DISTINCT keywords.keyword FROM keywords_photos,
keywords WHERE keywords_photos.photo_id = '.$resultArray['PhotoID'].' AND
keywords_photos.keyword_id = keywords.id';
...
print "\t".'<keywords>'."\n";
while ($j < $numRecs_key) {
   $resultKeywordArray = pg_Fetch_Array($my_result_keywords, $j);
   print "\t\t".'<keyword>'.$resultKeywordArray['keyword'].'</keyword>'.
"\n";
   $j++;
}
print "\t".'</keywords>'."\n";
```

5.5 Geografisch-politische Suche

Die geografisch-politische Suche wird mithilfe von drei **Selektionslisten** für Kanton, Bezirk und Gemeinde realisiert. Zu Beginn sind die Listen mit allen politischen Einheiten der Schweiz gefüllt. Man kann direkt die gewünschte Einheit in der entsprechenden Hierarchie auswählen. Man kann sich aber auch schrittweise annähern, indem man zuerst einen Kanton auswählt. Dann werden die Listen Bezirk und Gemeinde entsprechend angepasst und bieten nur noch Einträge des selektierten Kantons an. Analog ändert sich die Gemeindeliste, wenn man auch noch einen Bezirk selektiert. Bei jeder Auswahl einer politischen Einheit, wird diese in der Karte mit gelber, halbtransparenter Farbe hervorgehoben und der Kartenausschnitt entsprechend angepasst.
Den obersten Eintrag der Selektionslisten bildet jeweils der Ausdruck <--->, welcher eine Deselektion der entsprechenden Liste auslöst. Wird ein vorher ausgewählter Eintrag in der Gemeindeliste deselektiert, springt der Kartenausschnitt zum ausgewählten Bezirk zurück. Wird alles deselektiert, kehrt man in die Vollansicht der ganzen Schweiz zurück.
Diese Funktionalitäten werden mit den drei **Dateien search.js, extractAdminUnits.php und sendSingleAdminData.php** realisiert. Zuerst werden durch den Aufruf von extractAdminUnits.php alle ID's und Namen der in der Datenbank abgelegten politischen Einheiten eingelesen. Dann werden in search.js durch entsprechende Funktionen die einzelnen Selektionslisten mit Einträgen gefüllt. Hier werden die oben genannten Unterscheidungen definiert, sodass die Auswahl eingeschränkt wird, falls eine übergeordnete Einheit selektiert ist. Wenn man nun einen Listeneintrag auswählt, wird automatisch ein entsprechender URL zusammengestellt, welcher die Datei sendSingleAdminData.php mit dazugehörenden Parametern aufruft. Mit diesem Aufruf wird die Geometrie der ausgewählten politischen Einheit von der Datenbank eingelesen und diese dann über die Funktion addAdminPolygon(node) auf dem Kartenausschnitt dargestellt, welcher auch gleich entsprechend gezoomt wird.

search.js:

```
//get kanton data (SVG polygon)
var url = "sendSingleAdminData.php?adminType=kantone&name="+
myMapApp.SelectionLists["kantone"].getCurrentSelectionElement();
var getAdminDataObj = new getXMLData(url,addAdminPolygon);
getAdminDataObj.getData();

function addAdminPolygon(node) {
  if (myMainMap.adminGroup.hasChildNodes()) {
    myMainMap.adminGroup.removeChild(myMainMap.adminGroup.firstChild);
  }
  myMainMap.adminGroup.appendChild(node);
  var bbox = node.getBBox();
  myMainMap.showAdmin = true;
  myMainMap.setNewViewBox(bbox.x - bbox.width*0.01,(bbox.y + bbox.height
*1.02)*-1,(bbox.x + bbox.width*1.02),(bbox.y - bbox.height*0.01)*-1,true);
}
```

sendSingleAdminData.php:

```
if ($adminType == "kantone") {
   $mySQL = 'SELECT AsSVG(Simplify(the_geom,100),1,0) AS the_geom from
kantone WHERE name = '."'$name'";
}

print '<path xmlns="http://www.w3.org/2000/svg" fill="yellow" stroke="none"
fill-opacity="0.5" pointer-events="none"
d="'.$resultArray['the_geom'].'"/>'."\n";
```

6 Präsentation des Produkts

6.1 Navigation ohne Suchfunktionalitäten

Die Navigation ohne Suchfunktionalitäten beinhaltet einerseits die Kartennavigation und andererseits das Blättern in der Foto-Galerie. Dabei werden zur Visualisierung der Fotos und ihrer Metadaten **mouse-over-Effekte** genutzt. Eine beispielhafte Illustration ist in Abb. 9 gegeben.

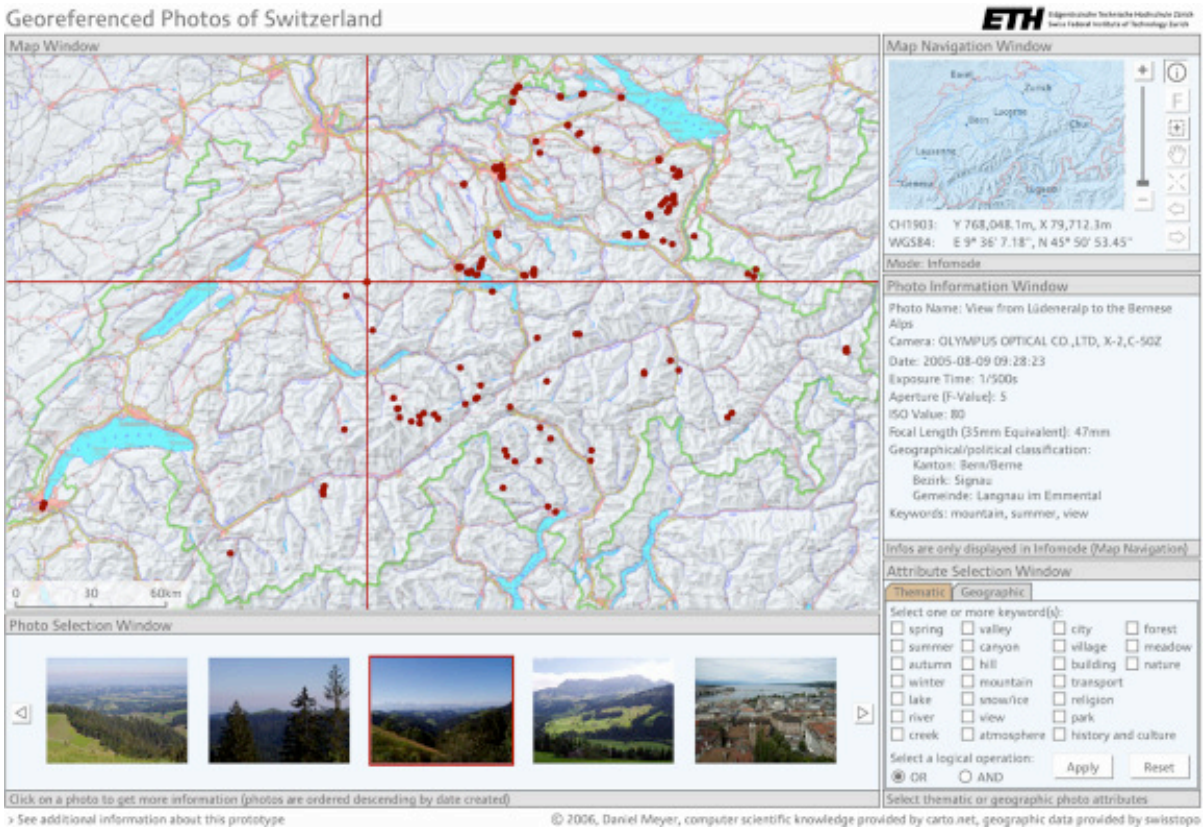


Abb. 9 Allgemeine Navigation in der Webseite mit mouse-over-Effekten

Die Auswahl eines einzelnen Fotos erfolgt durch Mausklick entweder auf ein Symbol in der Karte oder auf eine Kleinansicht in der Galerie. Ist ein Foto selektiert, ändert sich die Anzeige in der Galerie. Es wird nun nur noch das ausgewählte Foto mit zusätzlichen Metadaten und drei Buttons dargestellt. Zudem wird auf der Karte der ganze **Blickfächer** des ausgewählten Fotos angezeigt. Eine beispielhafte Illustration mit einem ausgewählten Photo ist in Abb. 10 gegeben.

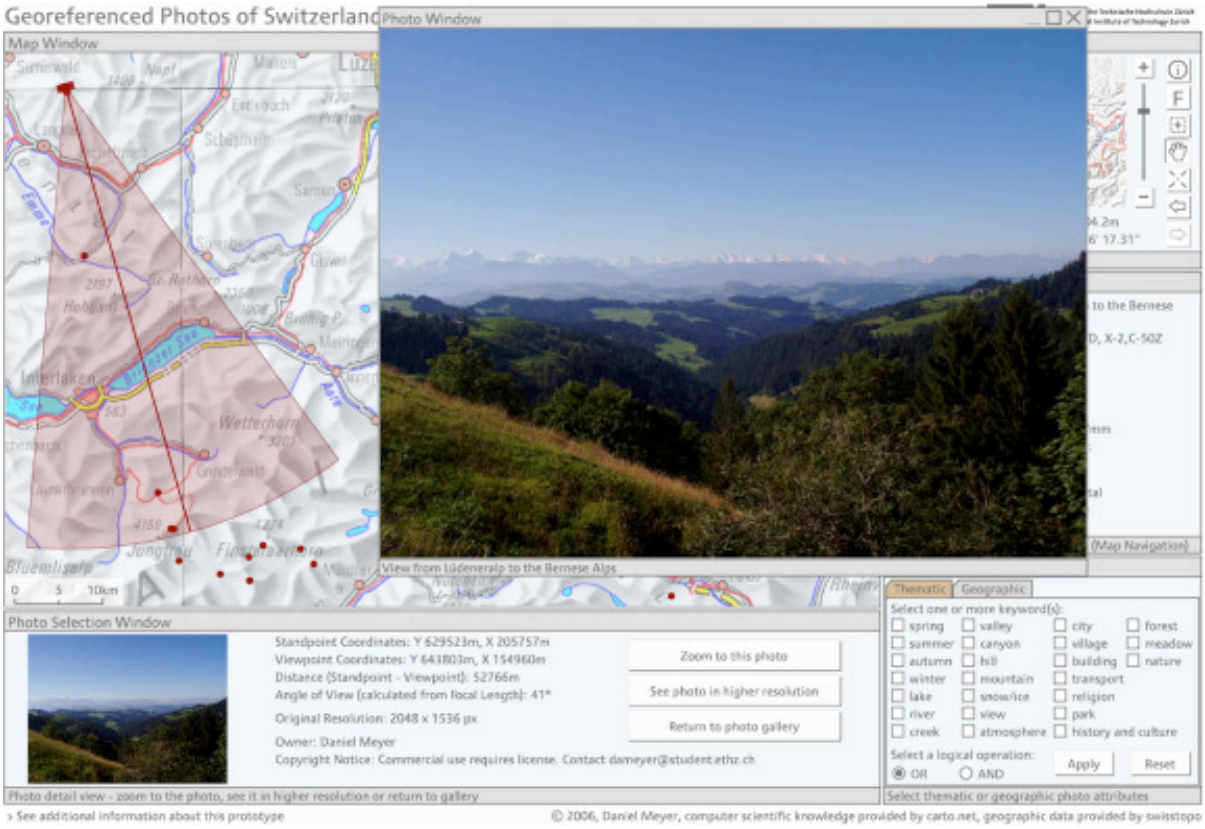


Abb. 10 Anzeige bei einem ausgewählten Foto

6.2 Auswahl thematischer Attribute

Die Anzeige der angebotenen Fotos kann direkt mithilfe der Auswahl von beschreibenden **Schlagwörtern** gesteuert werden. Für Informationen zum Auswahlkatalog der Schlagwörter beachte man Kap. 3.3.2. Es können beliebig viele Schlagwörter anhand von **Checkboxes** ausgewählt und dann mit einer logischen Verknüpfung (OR, AND) zu einer Abfrage verbunden werden. Alle Fotos, die den Suchkriterien nicht entsprechen, werden nach dem Ausführen der Suche nicht mehr angezeigt. Eine beispielhafte Illustration zur Auswahl der thematischen Schlagwörter ist in Abb. 11 gegeben.

Abb. 11 Auswahl von beschreibenden Schlagwörtern mithilfe von Checkboxes

6.3 Auswahl geografisch-politischer Attribute

Unter den geografisch-politischen Attributen sind **politische Einheiten**, also Kanton, Bezirk und Gemeinde zu verstehen. Diese können anhand von drei **Selektionslisten** ausgewählt werden. Mit der Auswahl einer politischen Einheit sucht man nicht direkt nach Fotos, sondern legt vielmehr den Kartenausschnitt fest. Es werden also nach einer Auswahl nicht nur noch jene Fotos angezeigt, welche sich effektiv innerhalb der selektierten Einheit befinden, sondern einfach alle, die im aktuellen Kartenausschnitt noch sichtbar sind. Eine beispielhafte Illustration zur Selektion von politischen Einheiten ist in Abb. 12 gegeben.

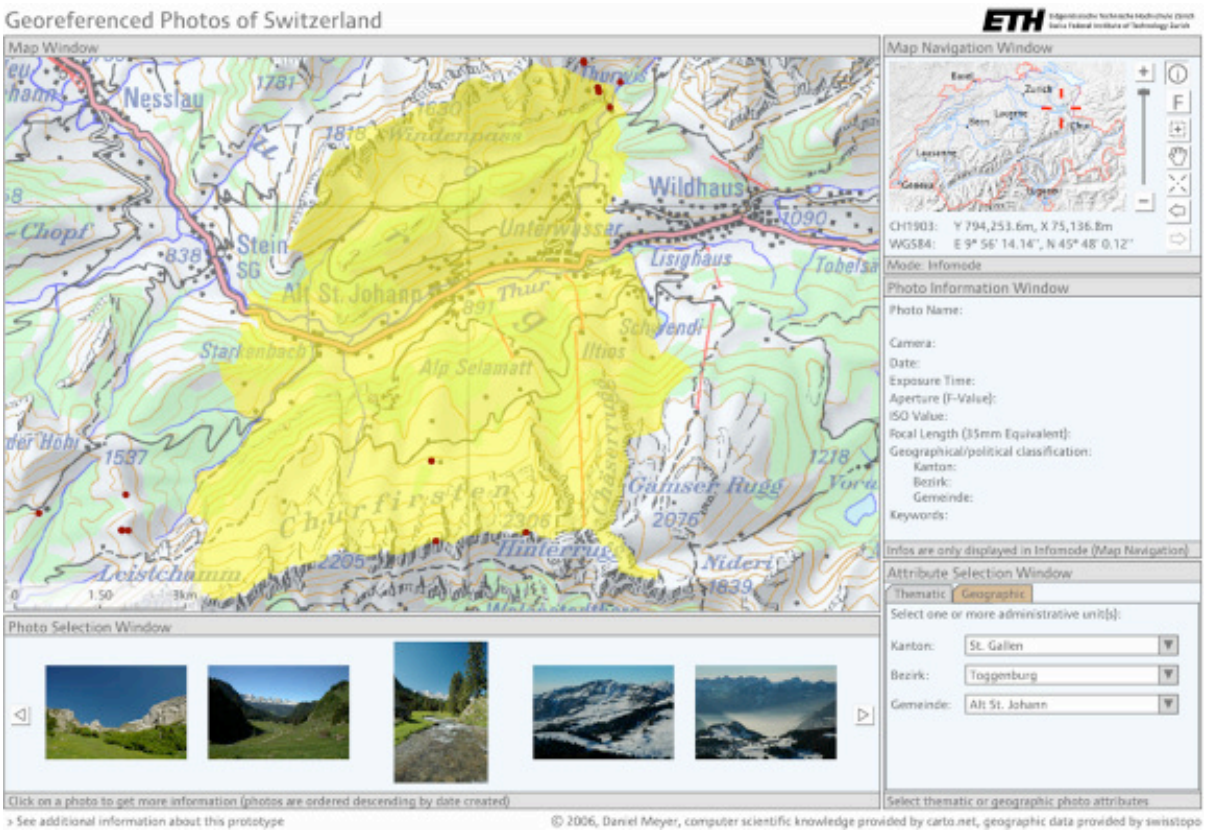


Abb. 12 Auswahl von politischen Einheiten mithilfe von Selektionslisten

7 Verbesserungspotential und Weiterentwicklung

7.1 Allgemeines

Bei der Diskussion um die Beurteilung des erstellten Produkts muss erst einmal grundlegend festgehalten werden, dass es sich bei der erarbeiteten Webseite um einen **Prototyp** handelt. Die Zeit, die im Rahmen eines Vertiefungsblocks zur Verfügung steht, ist sehr begrenzt, was zur Folge hat, dass kein vollends ausgereiftes Produkt hergestellt werden kann.
Insbesondere kann der eigentliche Feinschliff in der Bedienung der Applikation lediglich in Grundzügen erfolgen. Auf eine Evaluation der **Benutzerfreundlichkeit** mit anschliessender Verbesserung wird wegen Zeitmangels verzichtet. Es ist also evident, dass noch einige Schwächen im Prototyp vorliegen. Diese sollen nachfolgend kurz angesprochen werden. Ebenfalls sind einige Ideen für mögliche Weiterentwicklungen aufgeführt.

7.2 Schwächen des Prototyps

Eine grundlegende Schwäche besteht in der mangelnden **Generalisierung** der auf der Karte dargestellten Kamerasymbole. So erscheinen bei kleinen Kartenmassstäben mehrere Symbole schlicht übereinander oder verdecken sich teilweise. Sehr nahe beieinander liegende Symbole sollten zu einer gemeinsamen Signatur zusammengefasst werden. Beim vorliegenden Prototyp wird lediglich die rudimentäre Unterscheidung definiert, dass ab einem gewissen Kartenmassstab anstelle der Kamerasymbole nur noch Punktsymbole dargestellt werden.
Eine weitere Schwäche, die in die gleiche Richtung geht, besteht in der **Anzeige von Kamerasymbolen** des (nahezu) gleichen Aufnahmestandortes. Es kommt häufig vor, dass ein Fotograf vom gleichen Standort, beispielsweise einer Aussichtsplattform, mehrere Aufnahmen macht. Auf der Karte werden diese Aufnahmen dann durch verschiedene, sich überlagernde Kamerasymbole dargestellt, welche sich nicht mehr einzeln identifizieren lassen. Auch kann durch Mausinteraktion jeweils nur das zuoberst liegende Symbol, per Definition jenes Foto mit dem neuesten Aufnahmezeitpunkt, angesprochen werden. Dieser Umstand ist zwar nicht sehr problematisch, da die anderen Fotos in der Galerie angesprochen werden können, beeinträchtigt jedoch stark die Attraktivität der Visualisierung.
Ebenfalls leicht störend wirkt der **Zeilenabstand** bei der Anzeige der Metainformationen. Hier ist das Problem, dass die **Texteinträge**, wie z. B. der Name des Fotos, unterschiedlich lange sind. Das Problem wird angegangen, indem für potentiell lange Texteinträge grundsätzlich zwei Zeilen reserviert werden und ein Texteintrag, falls er diese zwei Zeilen überschreitet mit drei Punkten abgeschnitten wird. Das Erscheinungsbild präsentiert sich aber nicht ideal, wenn ein entsprechender Texteintrag sehr kurz ist und darunter noch eine leere Zeile steht.

Des weiteren fällt die etwas schlechte **Performance** des Kartenaufbaus auf. Bei langsamen Verbindungen bedingt das Laden der Seite jeweils merkliche Wartezeiten.
Auch die Kartennavigation könnte noch optimiert werden, obschon sie sicherlich bereits sehr komfortabel ist. Beim **Aufziehrahmen zur Vergrösserung** des Kartenausschnitts wäre wohl die Definition einer Minimalfläche sinnvoll, welche überschritten werden muss, damit ein Zoomen ausgelöst wird. Häufig vergisst der Anwender nämlich, dass dieser Modus noch aktiv ist, klickt in die Karte und löst so ein ungewolltes Zoomen aus. Eine andere Herangehensweise an diese Problematik wäre, dass der Navigationsmodus automatisch in den Info-Modus zurückwechselt, wenn eine gewisse Zeit lang keine Kartennavigation mehr ausgeführt wurde.
Ein letzter negativ auffallender Punkt ist die **Einteilung der Zoomskala** bei der Kartennavigation. Im vorliegenden Prototyp ist die Differenz zwischen minimaler und maximaler Ausdehnung des Kartenausschnitts sehr gross. Dies führt dazu, dass man beim Hereinzoomen (Plus-Button) relativ schnell ans Ende der Zoomskala gelangt, obschon man noch viel weiter hineinzoomen kann. Man könnte entweder die Skala des Zoomsliders anpassen oder einen zweiten Zoombereich definieren, welcher eingesetzt wird, wenn die erste Zoomskala ihr Ende erreicht hat. Der Zoomslider könnte dann wieder an den Anfang springen und die Referenzkarte entsprechend vergrössert werden.

7.3 Ideen zur Weiterentwicklung

Der erste Schritt zur Weiterentwicklung des Prototyps bildet natürlich die Behebung der vorgenannten Schwächen. Darüber hinaus bietet die erarbeitete Basis grosses Potential für zusätzliche Funktionalitäten. Der Phantasie sind dabei keine Grenzen gesetzt. Es sollen nun fünf Richtungen aufgezeigt werden, die bei einer allfälligen Weiterentwicklung im Vordergrund stehen könnten:

- **Erweiterung der Suchfunktionen durch Texteinträge**
  Bisher entspricht die Suche nach Fotos eher einer Auswahl. Es kann nach Fotos gesucht werden, indem man thematische Schlagwörter oder politische Einheiten auswählt. Lohnenswert wäre die Realisierung einer textbasierten Suche, die es erlaubt, Texteinträge in den Metainformationen der Fotos, beispielsweise des Bildnamens, zu finden.
- **User-Verwaltung mit Möglichkeit zur selbstständigen Integration eigener Fotos**
  Für eine breite Anwendung der Applikation ist es von zentraler Bedeutung, dass aus den Benutzern Teilhaber werden. Dieser generelle Entwicklungstrend des Web sollte aufgenommen werden. Es sollte die Möglichkeit angeboten werden, dass ein Benutzer sich einloggen und dann selbstständig eigene Bilder integrieren kann. Zu berücksichtigende Probleme dabei sind die Stabilität der Applikation und die Wahrung der Qualität des Angebots.

- **Möglichkeit zur Gruppierung von Fotos und zur Darstellung von Animationen**
  Durchaus reizvoll wäre es, wenn ein Benutzer einige seiner Bilder zu einer Reise zusammenfassen könnte und diese dann beispielsweise in Form einer Tageswanderung als Animation präsentieren könnte.
- **Anreicherung der Karte mit Tourismusinformationen**
  Auf der Karte könnten touristisch interessante Objekte, wie z. B. Stationen von Bergbahnen, punkthaft mit interaktiven Signaturen dargestellt werden. Zu diesen Signaturen könnten, etwa mithilfe von tooltips, Zusatzinformationen, wie z. B. die Fahrzeiten der Bergbahn, angezeigt werden. Die Signaturen könnten auch mit externen Webseiten verlinkt werden.
- **Integration eines Höhenmodells zur Verbesserung des Blickfächers**
  Der Prototyp benützt in keiner Form geografische Höhen. Eine wohl etwas utopische Idee zur Weiterentwicklung ist die Integration eines digitalen Höhenmodells. Dieses könnte für die Berechnung des auf der Karte dargestellten Blickfächers einer Aufnahme verwendet werden. So wäre es möglich, allfällige Verdeckung durch die Topografie im Blickfächer darzustellen. Damit wäre eine bessere Übereinstimmung des Blickfächers mit dem eigentlichen Foto erreicht.

8 Quellen und Dokumentationen

[CARTO]: carto:net: cartographers on the net. http://www.carto.net

[CPAN]: CPAN: Comprehensive Perl Archive Network. http://www.cpan.org

[ECMA]: Ecma International: Standard ECMA-262. ECMAScript Language Specification. http://www.ecma-international.org/publications/standards/Ecma-262.htm

[EXIF1]: EXIF.org: EXIF and related resources. http://www.exif.org

[EXIF2]: Phil Harvey: ExifTool Tag Names. http://www.sno.phy.queensu.ca/~phil/exiftool/TagNames/index.html

[EXIF3]: Phil Harvey: ExifTool. http://www.sno.phy.queensu.ca/~phil/exiftool

[EXIFER]: Friedemann Schmitt: Exifer for Windows. http://www.friedemann-schmidt.com/software/exifer

[EXIFV]: Ralf Bibinger: Exif Viewer. http://www.amarra.de/exif.htm

[GEOVITE]: Andreas Neumann: carto.net – Geovite Data Browser. http://www.carto.net/neumann/geovite

[GPSP]: Andreas Neumann, Peter Sykora: carto:net – gpsPhoto.pl. http://www.carto.net/projects/photoTools/gpsPhoto

[HTML]: Stefan Münz: SELFHTML 8.1 (HTML-Dateien selbst erstellen). http://de.selfhtml.org

[IPTC]: International Press Telecommunications Council: http://www.iptc.org/pages/index.php

[IRFAN]: IrfanView: one of the most popular viewers worldwide. http://www.irfanview.com

[OGC]: Open Geospatial Consortium (OGC): http://www.opengeospatial.org

[OPANDA]: Opanda.com: Professional Photography Software (DigitalFilm, PowerExif, IExif, PhotoFilter). http://www.opanda.com/en/index.html

[PERL]: perl.org: Perl version 5.8.7 documentation. http://perldoc.perl.org

[PGADM]: pgAdmin: Presentation of pgAdmin III. http://www.pgadmin.org

[PGIS]: PostGIS: PostGIS Documentation. http://postgis.refractions.net/documentation

[PGSQL]: PostgreSQL: PostgreSQL 8.0.4 Documentation. http://www.postgresql.org/docs/8.0/interactive/index.html

[PHP]: PHP.net: PHP Manual. http://www.php.net/manual/en

[ROBO]: RoboGeo: Geoference Photos. http://www.robogeo.com

[SVG]: World Wide Web Consortium (W3C): Scalable Vector Graphics (SVG). http://www.w3.org/Graphics/SVG

[STOPO]: Bundesamt für Landestopografie (swisstopo): http://www.swisstopo.ch

[XML]: World Wide Web Consortium (W3C): Extensible Markup Language (XML). http://www.w3.org/XML

Eidgenössische Technische Hochschule (ETH) Zürich
Departement Bau, Umwelt und Geomatik
Studienrichtung Geomatikingenieurwissenschaften

Vertiefungsblockarbeit am Institut für Kartographie (IKA)

Webpublikation von georeferenzierten Fotos

Entwicklung eines auf die Schweiz begrenzten Prototyps

Anhang

Autor: Daniel Meyer
Leitung: Prof. Dr. Lorenz Hurni, IKA
Betreuung: Andreas Neumann, IKA

Zürich, 9. Februar 2006

Inhaltsverzeichnis Anhang

1 Quelltexte

1.1 Einleitung

Nachfolgend sind die vollständigen Quelltexte einiger für dieses Projekt zentraler, selbstständig erarbeiteter Skripts aufgeführt. Es handelt sich dabei um je ein Perl-, ECMA- und PHP-Skript. Das Perl-Skript exif.pl ermöglicht ein automatisiertes Einlesen der EXIF-Metadaten aus den Bilddateien und schreibt diese Daten in die erstellte Datenbank. Das ECMA-Script photos.js regelt die Visualisierung und Interaktion der Fotos in der erstellten Webseite. Das PHP-Skript sendPhotoData.php macht die Metadaten aus der Datenbank mithilfe von SQL-Abfragen für die Webseite verfügbar.

1.2 exif.pl

```
#!/usr/bin/perl
#embed modules
use strict;
use File::Find;
use File::Basename;
use File::Copy;
use File::Spec;
use Math::Round;
use Getopt::Long;
use Image::ExifTool;
use Encode;
use DBI;

#define variables
my ($dir,$photodir,$origdir,$file,$imgCounter,$userid,$mySQL,$SQL_sp_wgs,$SQL_sp_ch,$SQL_tp_wgs,
$SQL_tp_ch,$sth,$chcoord,$status,$newfile,$thumbfile,$thumblongside,$factor,$thumbwidth,$thumbheight,$imagelength,$ccd,
$w,$pi);
my ($make,$model,$dateTimeOriginal,$objectName,$city,$cityReplaced,$subLocation,$subLocationReplaced,$provinceState,
$provinceStateReplaced,$credit,$copyrightNotice,$exifImageHeight,$exifImageWidth,$iso,$focalLength,$focalLength35,$fNum
ber,$exposureTime,$lat,$latRef,$lon,$lonRef,$destLat,$destLatRef,$destLon,$destLonRef,$imageDir,$imageDirRef,
$imageDist,$imageDistRef,$focalPlaneXResolution,$focalPlaneResolutionUnit,$keywords,$keywordId,$value,$scaleFactor);
my (@data,@imgTitles,@imgFilenames,@imgWidths,@imgHeights);
my $CoordFormat = Encode::decode_utf8("%.6f");
my $dateTimeFormat = Encode::decode_utf8("%Y:%m:%d %H:%M:%S");
my $inCoorSystem = 4326;
my $outCoorSystem = 2056;
my $version = "0.1";
my $date = "2005-10-30";
my $program = $0;
$program =~ s/(\.\/)//;
my $usage = "$program (version $version, $date)\nUsage: $program --dir myphotos/dir/";

#get parameters
GetOptions("dir=s" => \$dir);
unless ($dir) {
    die "$usage you have to specify a directory containing the images (--dir)!\n";
}

#treat strings to be correctly utf8
if ($dir) {
    $dir = Encode::encode("utf8",$dir);
}
$origdir = $dir;

#connect to a PostgreSQL database.
my $db_host = 'xxx.xxx.xxx.xxx';
my $db_user = 'xxx';
my $db_pass = 'xxx';
my $db_name = 'xxx';
my $db = "dbi:Pg:dbname=${db_name};host=${db_host}";
```

```
my $dbh = DBI->connect($db, $db_user, $db_pass) || die "Error connecting to the database: $DBI::errstr\n";

#read directory and count image files
opendir(DIR, $dir) or die "can't open directory $dir: $!";
$imgCounter = 0;

while (defined($file = readdir(DIR))) {
    $file = File::Spec->rel2abs($dir.$file);
    #first check if it is a jpeg file
    my ($base, $dir, $ext) = fileparse($file,'\..*');
    if ($ext eq ".jpg" || $ext eq ".JPG" || $ext eq ".jpeg" || $ext eq ".JPEG") {
        print "Infos to image ".($imgCounter + 1)."($file):\n";
        my $exifTool = new Image::ExifTool;
        #set a few parameters
        $exifTool->Options(Charset => 'UTF8', CoordFormat => $CoordFormat, DateFormat => $dateTimeFormat);
        my $imgInfo = $exifTool->ImageInfo($file,"Make","Model","DateTimeOriginal","ObjectName","Credit","CopyrightNotice",
"ExifImageLength","ExifImageWidth","ISO","FocalLength","FocalLengthIn35mmFormat","FNumber","ExposureTime",
"GPSLatitude","GPSLatitudeRef","GPSLongitude","GPSLongitudeRef","GPSDestLatitude","GPSDestLatitudeRef",
"GPSDestLongitude","GPSDestLongitudeRef","FocalPlaneXResolution","FocalPlaneResolutionUnit","Keywords",
"ScaleFactor35efl");
        #get individual parameters
        my @tags = $exifTool->GetRequestedTags();
        $make = Encode::encode("utf8",$exifTool->GetValue($tags[0]));
        $model = Encode::encode("utf8",$exifTool->GetValue($tags[1]));
        $dateTimeOriginal =  Encode::encode("utf8",$exifTool->GetValue($tags[2]));
        $objectName = Encode::encode("utf8",$exifTool->GetValue($tags[3]));
        $credit = Encode::encode("utf8",$exifTool->GetValue($tags[4]));
        $copyrightNotice = Encode::encode("utf8",$exifTool->GetValue($tags[5]));
        $exifImageHeight = Encode::encode("utf8",$exifTool->GetValue($tags[6]));
        $exifImageWidth = Encode::encode("utf8",$exifTool->GetValue($tags[7]));
        $iso = Encode::encode("utf8",$exifTool->GetValue($tags[8]));
        $focalLength = Encode::encode("utf8",$exifTool->GetValue($tags[9]));
        $focalLength35 = Encode::encode("utf8",$exifTool->GetValue($tags[10]));
        $fNumber = Encode::encode("utf8",$exifTool->GetValue($tags[11]));
        $exposureTime = Encode::encode("utf8",$exifTool->GetValue($tags[12]));
        $lat = Encode::encode("utf8",$exifTool->GetValue($tags[13]));
        $latRef = Encode::encode("utf8",$exifTool->GetValue($tags[14]));
        $lon = Encode::encode("utf8",$exifTool->GetValue($tags[15]));
        $lonRef = Encode::encode("utf8",$exifTool->GetValue($tags[16]));
        $destLat = Encode::encode("utf8",$exifTool->GetValue($tags[17]));
        $destLatRef = Encode::encode("utf8",$exifTool->GetValue($tags[18]));
        $destLon = Encode::encode("utf8",$exifTool->GetValue($tags[19]));
        $destLonRef = Encode::encode("utf8",$exifTool->GetValue($tags[20]));
        $focalPlaneXResolution = Encode::encode("utf8",$exifTool->GetValue($tags[21]));
        $focalPlaneResolutionUnit = Encode::encode("utf8",$exifTool->GetValue($tags[22]));
        $keywords = Encode::encode("utf8",$exifTool->GetValue($tags[23]));
        $scaleFactor = Encode::encode("utf8",$exifTool->GetValue($tags[24]));

        $error = 0;
        if (!$make) {$make = "NULL"; print "Make not defined\n"; $error += 1;}
        if (!$model) {$model = "NULL"; print "Model not defined\n"; $error += 1;}
        if (!$dateTimeOriginal) {$dateTimeOriginal = "NULL"; print "DateTimeOriginal not defined\n"; $error += 1;}
        if (!$objectName) {$objectName = "NULL"; print "ObjectName not defined\n"; $error += 1;}
        if (!$credit) {$credit = "NULL"; print "Credit not defined\n"; $error += 1;}
        if (!$copyrightNotice) {$copyrightNotice = "NULL"; print "CopyrightNotice not defined\n"; $error += 1;}
        if (!$exifImageHeight) {$exifImageHeight = "NULL"; print "ExifImageLength not defined\n"; $error += 1;}
        if (!$exifImageWidth) {$exifImageWidth = "NULL"; print "ExifImageWidth not defined\n"; $error += 1;}
        if (!$iso) {$iso = "NULL"; print "ISO not defined\n"; $error += 1;}
        if (!$focalLength) {$focalLength = "NULL"; print "FocalLength not defined\n"; $error += 1;}
        if (!$fNumber) {$fNumber = "NULL"; print "FNumber not defined\n"; error += 1;}
        if (!$exposureTime) {$exposureTime = "NULL"; print "ExposureTime not defined\n"; $error += 1;}
        if (!$keywords) {$keywords = "NULL"; print "Keywords not defined\n"; $error += 1;}
        if (!$lat || !$latRef || !$lon || !$lonRef || !$destLat || !$destLatRef || !$destLon || !$destLonRef) {print "Coordinates not
correctly defined\n"; $error += 1;}

        #mask the ' for the sql-statement
        $objectName =~ s/\'/\\\'/gi;

        #extract the timezone appendix from dateTimeOriginal
        my @dateTime = split(/\./,$dateTimeOriginal);
        $dateTimeOriginal = $dateTime[0];

        #convert coordinate string to float values
        my @latCoords = split(/\s+/,$lat);
```

```
        $lat = $latCoords[0];
        my @lonCoords = split(/\s+/,$lon);
        $lon = $lonCoords[0];
        my @destLatCoords = split(/\s+/,$destLat);
        $destLat = $destLatCoords[0];
        my @destLonCoords = split(/\s+/,$destLon);
        $destLon = $destLonCoords[0];

        #get coords of standpoint in ch1903
        $mySQL = "SELECT AsText(TRANSFORM(SetSRID(GeometryFromText('POINT($lon $lat)'),$inCoorSystem),
$outCoorSystem)) AS coord;";
        $sth = $dbh->prepare($mySQL);
        $sth->execute or db_err("Unable to execute query", $dbh->errstr);
        while (@data = $sth->fetchrow_array()) {
            $chcoord = $data[0];
        }

        $city = "undefined";
        $cityReplaced = "undefined";
        $subLocation = "undefined";
        $subLocationReplaced = "undefined";
        $provinceState = "undefined";
        $provinceStateReplaced = "undefined";

        #get city, subLocation and provinceState according to coords
        $mySQL = "SELECT gemeinden.gemname,bezirke.name,kantone.name from gemeinden,bezirke,kantone WHERE
Contains(the_geom,setSRID(GeometryFromText('$chcoord'),$outCoorSystem)) AND gemeinden.bezirksnr = bezirke.id AND
gemeinden.kantonsnr = kantone.kanton;";
        $sth = $dbh->prepare($mySQL);
        $sth->execute or db_err("Unable to execute query", $dbh->errstr);
        while (@data = $sth->fetchrow_array()) {
            $city = $data[0];
            $cityReplaced = replaceChars($data[0]);
            $subLocation = $data[1];
            $subLocationReplaced = replaceChars($data[1]);
            $provinceState = $data[2];
            $provinceStateReplaced = replaceChars($data[2]);
        }

        #test if directories exist or create new ones
        $photodir = $dir;
        $photodir =~ s/$origdir/photos\/Switzerland\//;
        my $kantondir = $photodir.$provinceStateReplaced."/";
        unless (-d $kantondir) { #see if directory exists
            mkdir($kantondir);
        }
        my $bezirksdir = $kantondir.$subLocationReplaced."/";
        unless (-d $bezirksdir) { #see if directory exists
            mkdir($bezirksdir);
        }
        my $gemeindedir = $bezirksdir.$cityReplaced."/";
        unless (-d $gemeindedir) { #see if directory exists
            mkdir($gemeindedir);
        }

        if ($city eq "undefined") {
            print "Could not match the coordinates to a municipality !!!!!!!!!!!!!!!!!\n";
            $error += 1;
        }
        #copy file to specified directory
        $newfile = $gemeindedir.replaceChars($dateTimeOriginal).'_'.replaceChars($objectName).$ext;
        copy($file,$newfile) or die "could not copy $file";

        #create thumbnails
        $thumblongside = 200;
        while ($thumblongside != 1000) {
            if ($exifImageWidth > $exifImageHeight) {
                $factor = $exifImageHeight / $exifImageWidth;
                $thumbwidth = $thumblongside;
                $thumbheight = nearest(1,($thumbwidth * $factor));    #round to integer
            }
            else {
                $factor = $exifImageWidth / $exifImageHeight;
                $thumbheight = $thumblongside;
```

```
                $thumbwidth = nearest(1,($thumbheight * $factor));    #round to integer
            }
            $thumbfile = $gemeindedir.replaceChars($dateTimeOriginal).'_'.replaceChars($objectName)."
_thumb$thumblongside".$ext;
            $status = system("convert '".$file."' -resize ".$thumbwidth."x".$thumbheight." ".$thumbfile);
            die "could not create thumbnail $thumbfile" if $status == 1;
            $thumblongside += 400;
        }

        #extract the mm from focalLength
        $focalLength =~ s/mm//gi;

        #compute focalLength35 if not available
        #problem: other necessary attributes are maybe neither available!
        #problem: EXIF-Tag ScaleFactor35efl is not always reliable!
        if ($focalLength != "NULL" && !$focalLength35) {
            if ($make == "RICOH" && $model == "Caplio R2") {$focalLength35 = nearest(1,($focalLength*6));}
    #scalefactor for RICOH Caplio R2 = 6
            elsif ($make == "OLYMPUS OPTICAL CO.,LTD" && $model == "X-2,C50Z") {$focalLength35 =
nearest(1,($focalLength*4.9));}            #scalefactor for Olympus C-50Z = 4.9
            elsif ($scaleFactor) {$focalLength35 = nearest(1,($focalLength*$scaleFactor));}
            else {
                $imagelength = $exifImageWidth;
                if ($exifImageHeight > $exifImageWidth) {$imagelength = $exifImageHeight;}          #get the longer side of the
image according to focalPlaneXResoluiton
                if ($focalPlaneXResolution) {
                    $ccd = $imagelength/$focalPlaneXResolution;    #size of ccd = pixels/resolution (resolution = pixels/size)
                    if ($focalPlaneResolutionUnit eq "inches") {$ccd = $ccd*25.4;}          #convert inches to mm
                    $focalLength35 = nearest(1,($focalLength*36/$ccd));          # f : ccd = f35 : 36 (36 = length of a 35mm
negative)
                }
                else {
                    print "Could not detect or compute FocalLengthIn35mmFormat and therefore neither Angle of View\n";
                    $error += 1;
                }
            }
        }

        #compute angle of view
        if (!$focalLength35) {$w=0}
        else {
            if ($exifImageHeight > $exifImageWidth) {$w = 2*atan2(12,$focalLength35);}
            else {$w = 2*atan2(18,$focalLength35);}          #the values 12 and 18 are the half of the width or height of a 35mm
negative (24x36mm)
            $pi = atan2(1,1)*4;
            $w = $w*180/$pi;
        }

        #try to match user id
        $mySQL = "SELECT id from user_aliases WHERE alias LIKE '%$credit%';";
        $sth = $dbh->prepare($mySQL);
        $sth->execute or db_err("Unable to select user alias data", $dbh->errstr);
        $userid = -99;
        while (@data = $sth->fetchrow_array()) {
            $userid = $data[0];
        }

        #print out camera/photo info
        print "Make: $make\nModel: $model\nDateTimeOriginal: $dateTimeOriginal\nObjectName: $objectName\nProvince-
State: $provinceState\nSub-location: $subLocation\nCity: $city\nCredit: $credit\nCopyrightNotice:$copyrightNotice\n
ImageHeight: $exifImageHeight\nImageWidth: $exifImageWidth\nISO: $iso\nFNumber: $fNumber\nFocalLength: $focalLength\n
FocalLengthIn35mmFormat: $focalLength35\nExposureTime: $exposureTime\nStandpoint: lat: $lat, lon: $lon\n
Targetpoint: destLat: $destLat, destLon: $destLon\n";

        #define SQL-statements for coords
        $SQL_sp_wgs = qq(SetSRID(GeometryFromText('POINT($lon $lat)'),$inCoorSystem));
        $SQL_sp_ch = qq(Transform(SetSRID(GeometryFromText('POINT($lon $lat)'),$inCoorSystem),$outCoorSystem));
        $SQL_tp_wgs = qq(SetSRID(GeometryFromText('POINT($destLon $destLat)'),$inCoorSystem));
        $SQL_tp_ch = qq(Transform(SetSRID(GeometryFromText('POINT($destLon $destLat)'),$inCoorSystem),
$outCoorSystem));

        #write exif data into db
        $mySQL = qq(INSERT INTO photos ("Make","Model","DateTimeOriginal","ObjectName","City","Sub-location",
"Province-State","Credit","CopyrightNotice","ExifImageLength","ExifImageWidth","ISO","FocalLength","FNumber",
```

```
"FocalLengthIn35mmFormat","UserID","ExposureTime","AngleOfView","standpoint_wgs84","standpoint_ch1903",
"targetpoint_wgs84","targetpoint_ch1903","FileName") VALUES ('$make','$model',to_timestamp('$dateTimeOriginal',
'YYYY:MM:DD HH24:MI:SS'),'$objectName','$city','$subLocation','$provinceState','$credit','$copyrightNotice',$exifImageHeight,
$exifImageWidth,$iso,$focalLength,$fNumber,$focalLength35,$userid,'$exposureTime',$w,$SQL_sp_wgs,$SQL_sp_ch,
$SQL_tp_wgs,$SQL_tp_ch,'$newfile'););
        $sth = $dbh->prepare($mySQL);
        $sth->execute or db_err("Unable to insert foto data", $dbh->errstr);

        #get current photo id
        $mySQL = qq(SELECT currval('create_photo_id'));
        $sth = $dbh->prepare($mySQL);
        $sth->execute or db_err("Unable to query current foto id", $dbh->errstr);
        @data = $sth->fetchrow_array();
        my $photoId = @data[0];

        #split keywords
        my @Keywords = split(/\s*,\s*/,$keywords);
        my @KeywordIDs;
        my $KeywordID;
        my $unique;
        print "Keywords: @Keywords\n";
        foreach $value (@Keywords) {
            $keywordId = -99;
            $unique = 1;
            $mySQL = "SELECT id FROM keyword_aliases WHERE alias = '$value';";
            $sth = $dbh->prepare($mySQL);
            $sth->execute or db_err("Unable to select keyword alias data", $dbh->errstr);
            while (@data = $sth->fetchrow_array()) {
                $keywordId = $data[0];
            }
            foreach $KeywordID (@KeywordIDs) {
                if ($keywordId == $KeywordID) {
                    $unique = 0;
                }
            }
            if ($keywordId != -99 && $unique == 1) {
                $mySQL = qq(INSERT INTO keywords_photos VALUES ($photoId,$keywordId));
                $sth = $dbh->prepare($mySQL);
                $sth->execute or db_err("Unable to input keyword id", $dbh->errstr);
            }
            push(@KeywordIDs,$keywordId);
        }
        print "All informations have been written to database. $error errors occured.\n";
        print " Processing photo " ($imgCounter + 1)." completed.\n\n\n";

        $imgCounter++;
    }
}

close(DIR);
$sth->finish;
$dbh->disconnect;

#subroutine for special characters
sub replaceChars {
    my $string = shift;
    $string =~ s/\/|\(|\)|\s+/_/g;
    $string =~ s/ä/ae/g;
    $string =~ s/Ä/Ae/g;
    $string =~ s/ö/oe/g;
    $string =~ s/Ö/Oe/g;
    $string =~ s/ü/ue/g;
    $string =~ s/Ü/Ue/g;
    $string =~ s/é|è|ê/e/g;
    $string =~ s/à|â/a/g;
    $string =~ s/ô/o/g;
    $string =~ s/[^a-zA-Z0-9_]//g;
    return $string;
}
```

1.3 photos.js

```
//this function receives the photo metadata
function receivePhotoData(node) {
    var timestamp = parseInt(node.getAttributeNS(null,"timestamp"));
    var photoCounter = 0;
    if (timestamp == myMainMap.timestamp) {
        var child = node.firstChild;
        myMapApp.photoData = new Array();
        while(child) {
            if (child.nodeType == "1") {
                if (child.nodeName == "photo") {
                    var id = "id_"+child.getAttributeNS(null,"id");
                    var FileName = child.getAttributeNS(null,"FileName");
                    var ObjectName = child.getAttributeNS(null,"ObjectName");
                    var Model = child.getAttributeNS(null,"Make")+", "+child.getAttributeNS(null,"Model");
                    var DateTime = child.getAttributeNS(null,"DateTime");
                    var Height = parseInt(child.getAttributeNS(null,"Length"));
                    var Width = parseInt(child.getAttributeNS(null,"Width"));
                    var Exposure = child.getAttributeNS(null,"Exposure");
                    var Aperture = parseInt(child.getAttributeNS(null,"Aperture"));
                    var ISO = parseInt(child.getAttributeNS(null,"ISO"));
                    var FocalLength35 = parseInt(child.getAttributeNS(null,"FocalLength35"));
                    var Owner = child.getAttributeNS(null,"UserFirstName")+" "+child.getAttributeNS(null,"UserName");
                    var Copyright = child.getAttributeNS(null,"Copyright");
                    var StandPointX = parseInt(child.getAttributeNS(null,"StandPointX"));
                    var StandPointY = parseInt(child.getAttributeNS(null,"StandPointY"));
                    var TargetPointX = parseInt(child.getAttributeNS(null,"TargetPointX"));
                    var TargetPointY = parseInt(child.getAttributeNS(null,"TargetPointY"));
                    var AngleOfView = parseInt(child.getAttributeNS(null,"AngleOfView"));
                    var ProvinceState = child.getAttributeNS(null,"ProvinceState");
                    var SubLocation = child.getAttributeNS(null,"SubLocation");
                    var City = child.getAttributeNS(null,"City");
                    var keywordArray = new Array();
                    var grandChild = child.firstChild;
                    while(grandChild) {
                        if (grandChild.nodeType == "1") {
                            if (grandChild.nodeName == "keywords") {
                                var grandGrandChild = grandChild.firstChild;
                                while(grandGrandChild) {
                                    if (grandGrandChild.nodeType == "1") {
                                        if (grandGrandChild.nodeName == "keyword") {
                                    keywordArray.push(grandGrandChild.firstChild.nodeValue);
                                        }
                                    }
                                    grandGrandChild = grandGrandChild.nextSibling;
                                }
                            }
                        }
                        grandChild = grandChild.nextSibling;
                    }
                    myMapApp.photoData[id]= {key:id,value:new photo(id,FileName,ObjectName,Model,DateTime,Height,
Width,Exposure,Aperture,ISO,FocalLength35,Owner,Copyright,StandPointX,StandPointY,TargetPointX,TargetPointY,
AngleOfView,ProvinceState,SubLocation,City,keywordArray)};
                    myMapApp.photoData.push(myMapApp.photoData[id]);
                    photoCounter++;
                }
            }
            child = child.nextSibling;
        }
        drawCameraSymbols();
        if (myMapApp.photoSelectStatus) {
            drawSelectedPhotoDetail();
        }
        else {
            displayPhotosSmall("start",undefined);
        }
    }
}


//this function parses the metadata
```

```
function photo(id,FileName,ObjectName,Model,DateTime,Height,Width,Exposure,Aperture,ISO,FocalLength35,Owner,
Copyright,StandPointX,StandPointY,TargetPointX,TargetPointY,AngleOfView,ProvinceState,SubLocation,City,keywordArray) {
    this.id = id;
    this.FileName = FileName;
    this.ObjectName = ObjectName;
    this.Model = Model;
    this.DateTime = DateTime;
    this.Height = Height;
    this.Width = Width;
    this.Exposure = Exposure;
    this.Aperture = Aperture;
    this.ISO = ISO;
    this.FocalLength35 = FocalLength35;
    this.Owner = Owner;
    this.Copyright = Copyright;
    this.StandPointX = StandPointX;
    this.StandPointY = StandPointY;
    this.TargetPointX = TargetPointX;
    this.TargetPointY = TargetPointY;
    this.AngleOfView = AngleOfView;
    this.ProvinceState = ProvinceState;
    this.SubLocation = SubLocation;
    this.City = City;
    this.keywordArray = keywordArray.join(", ");
}


//this function draws the camera symbols according to the map scale
function drawCameraSymbols() {
    var photoSymbolGroup = document.getElementById("photoSymbols");
    if (photoSymbolGroup.hasChildNodes()) {
        photoSymbolGroup.removeChild(photoSymbolGroup.firstChild);
    }
    //create temporary group for photo symbols
    var photoSymbolsTempGroup = document.createElementNS(svgNS,"g");
    photoSymbolsTempGroup.setAttributeNS(null,"id","photoSymbolsTemp");
    photoSymbolGroup.appendChild(photoSymbolsTempGroup);

    //update symbol sizes
    document.getElementById("cameraSmallSymbGeom").setAttributeNS(null,"transform","scale("+(myMainMap.curWidth *
0.002)+")");
    document.getElementById("cameraSymbGeom").setAttributeNS(null,"transform","scale("+(myMainMap.curWidth *
0.002)+")");

    for (var i=0;i<myMapApp.photoData.length;i++) {
        var photoSymbol = document.createElementNS(svgNS,"use");
        if (myMainMap.curWidth > 10000) {
            photoSymbol.setAttributeNS(xlinkNS,"href","#cameraSmall");
        }
        else {
            photoSymbol.setAttributeNS(xlinkNS,"href","#camera");
            var direction = toPolarDir(myMapApp.photoData[i].value.TargetPointX -
myMapApp.photoData[i].value.StandPointX,myMapApp.photoData[i].value.TargetPointY -
myMapApp.photoData[i].value.StandPointY) / Math.PI * 180;
            photoSymbol.setAttributeNS(null,"transform","rotate("+(direction*-
1)+","+myMapApp.photoData[i].value.StandPointX+","+(myMapApp.photoData[i].value.StandPointY * -1)+")");
        }
        photoSymbol.setAttributeNS(null,"x",myMapApp.photoData[i].value.StandPointX);
        photoSymbol.setAttributeNS(null,"y",myMapApp.photoData[i].value.StandPointY * -1);
        photoSymbol.setAttributeNS(null,"id","photoSymbol_"+myMapApp.photoData[i].value.id);
        photoSymbol.addEventListener("mouseover",showPhotoData,false);
        photoSymbol.addEventListener("mouseout",showPhotoData,false);
        photoSymbol.addEventListener("click",setPhotoExtentCheck,false);
        photoSymbol.addEventListener("click",showPhotoData,false);
        photoSymbolsTempGroup.appendChild(photoSymbol);
    }
}


//this function writes information tho a photo in the photo informaiton window
function showInfos(id){
    var curPhoto = id;
    var metaObjectName = document.getElementById("metaObjectName");
    var dy = textFlow("Photo Name: " + curPhoto.ObjectName,metaObjectName,50,5,13,2);
```

```
    document.getElementById("metaModel").firstChild.nodeValue = "Camera: " + curPhoto.Model;
    document.getElementById("metaDate").firstChild.nodeValue = "Date: " + curPhoto.DateTime;
    if (curPhoto.Exposure == 'NULL') {
        document.getElementById("metaExposure").firstChild.nodeValue = "Exposure Time: undefined";
    }
    else {
        document.getElementById("metaExposure").firstChild.nodeValue = "Exposure Time: " + curPhoto.Exposure + "s";
    }
    if (isNaN(curPhoto.Aperture)) {
        document.getElementById("metaAperture").firstChild.nodeValue = "Aperture (F-Value): undefined";
    }
    else {
        document.getElementById("metaAperture").firstChild.nodeValue = "Aperture (F-Value): " + curPhoto.Aperture;
    }
    if (isNaN(curPhoto.ISO)) {
        document.getElementById("metaISO").firstChild.nodeValue = "ISO Value: undefined";
    }
    else {
        document.getElementById("metaISO").firstChild.nodeValue = "ISO Value: " + curPhoto.ISO;
    }
    document.getElementById("metaFocal").firstChild.nodeValue = "Focal Length (35mm Equivalent): " +
curPhoto.FocalLength35 + "mm";
    document.getElementById("metaClassification").firstChild.nodeValue = "Geographical/political classification:";
    document.getElementById("metaKanton").firstChild.nodeValue = "Kanton: " + curPhoto.ProvinceState;
    document.getElementById("metaBezirk").firstChild.nodeValue = "Bezirk: " + curPhoto.SubLocation;
    document.getElementById("metaGemeinde").firstChild.nodeValue = "Gemeinde: " + curPhoto.City;
    var metaKeywords = document.getElementById("metaKeywords");
    dy = textFlow("Keywords: " + curPhoto.keywordArray,metaKeywords,50,5,13,2);
}


//this function empties the photo information window
function showInfosEmpty(){
    var metaObjectName = document.getElementById("metaObjectName");
    var dy = textFlow("Photo Name: ",metaObjectName,50,5,13,undefined);
    document.getElementById("metaModel").firstChild.nodeValue = "Camera: ";
    document.getElementById("metaDate").firstChild.nodeValue = "Date: ";
    document.getElementById("metaExposure").firstChild.nodeValue = "Exposure Time: ";
    document.getElementById("metaAperture").firstChild.nodeValue = "Aperture (F-Value): ";
    document.getElementById("metaISO").firstChild.nodeValue = "ISO Value: ";
    document.getElementById("metaFocal").firstChild.nodeValue = "Focal Length (35mm Equivalent): ";
    document.getElementById("metaClassification").firstChild.nodeValue = "Geographical/political classification:";
    document.getElementById("metaKanton").firstChild.nodeValue = "Kanton: ";
    document.getElementById("metaBezirk").firstChild.nodeValue = "Bezirk: ";
    document.getElementById("metaGemeinde").firstChild.nodeValue = "Gemeinde: ";
    var metaKeywords = document.getElementById("metaKeywords");
    dy = textFlow("Keywords: ",metaKeywords,50,5,13,undefined);
}


//this function displays photo metadata according to mouse-events
function showPhotoData(evt) {
    var symbReference = evt.currentTarget;
    var origId = symbReference.getAttributeNS(null,"id");
    var id = origId.replace(/photoSymbol_|photoSmall_/,"");
    var curPhoto = myMapApp.photoData[id].value;
    if (evt.type == "mouseover") {
        showInfos(curPhoto);
        //show crosshair cursor in map
        if (origId.match(/photoSmall_/)) {
            myMainMap.highlightCrossHoriz.setAttributeNS(null,"x1",myMainMap.curxOrig);
            myMainMap.highlightCrossHoriz.setAttributeNS(null,"y1",curPhoto.StandPointY * -1);
            myMainMap.highlightCrossHoriz.setAttributeNS(null,"x2",myMainMap.curxOrig+myMainMap.curWidth);
            myMainMap.highlightCrossHoriz.setAttributeNS(null,"y2",curPhoto.StandPointY * -1);
            myMainMap.highlightCrossVert.setAttributeNS(null,"x1",curPhoto.StandPointX);
            myMainMap.highlightCrossVert.setAttributeNS(null,"y1",myMainMap.curyOrig);
            myMainMap.highlightCrossVert.setAttributeNS(null,"x2",curPhoto.StandPointX);
            myMainMap.highlightCrossVert.setAttributeNS(null,"y2",myMainMap.curyOrig+myMainMap.curHeight);
            myMainMap.highlightCross.setAttributeNS(null,"visibility","visible");
        }
        //adapt photo frame
        if (!myMapApp.photoSelectStatus) {
            try {
                var photoBBox = document.getElementById("photoSmall_"+id).getBBox();
```

```
                    myMapApp.photoFrame.setAttributeNS(null,"x",photoBBox.x-1);
                    myMapApp.photoFrame.setAttributeNS(null,"y",photoBBox.y-1);
                    myMapApp.photoFrame.setAttributeNS(null,"width",photoBBox.width+2);
                    myMapApp.photoFrame.setAttributeNS(null,"height",photoBBox.height+2);
                    myMapApp.photoFrame.setAttributeNS(null,"display","inherit");
                }
                catch(er) {}
            }
        }

        if (evt.type == "mouseout") {
            if (myMapApp.photoSelectStatus) {
                id = myMapApp.photoSelectedId;
                var curPhoto = myMapApp.photoData[id].value;
                showInfos(curPhoto);
            }
            else {
                showInfosEmpty();
            }
            myMapApp.photoFrame.setAttributeNS(null,"display","none");
            //hide crosshair cursor in map
            if (origId.match(/photoSmall_/)) {
                myMainMap.highlightCross.setAttributeNS(null,"visibility","hidden");
            }
        }

        if (evt.type == "click") {
            myMapApp.photoSelectStatus = true;
            myMapApp.photoSelectedId = id;
            showInfos(curPhoto);

            //calculate angle of view
            myMapApp.selPhotoDistance = toPolarDist(curPhoto.StandPointX - curPhoto.TargetPointX,curPhoto.StandPointY -
curPhoto.TargetPointY);
            myMapApp.selPhotoDirection = toPolarDir(curPhoto.TargetPointX - curPhoto.StandPointX,curPhoto.TargetPointY -
curPhoto.StandPointY) / Math.PI * 180;
            var diffX = toRectX((myMapApp.selPhotoDirection +
(curPhoto.AngleOfView*0.5))*Math.PI/180,myMapApp.selPhotoDistance);
            var diffY = toRectY((myMapApp.selPhotoDirection +
(curPhoto.AngleOfView*0.5))*Math.PI/180,myMapApp.selPhotoDistance) * -1;
            myMapApp.selPhotoPointX1 = curPhoto.StandPointX + diffX;
            myMapApp.selPhotoPointY1 = curPhoto.StandPointY * -1 + diffY;
            diffX = toRectX((myMapApp.selPhotoDirection -
(curPhoto.AngleOfView*0.5))*Math.PI/180,myMapApp.selPhotoDistance);
            diffY = toRectY((myMapApp.selPhotoDirection -
(curPhoto.AngleOfView*0.5))*Math.PI/180,myMapApp.selPhotoDistance) * -1;
            myMapApp.selPhotoPointX2 = curPhoto.StandPointX + diffX;
            myMapApp.selPhotoPointY2 = curPhoto.StandPointY * -1 + diffY;

            //first check if angle of view is within map extent
            if (myMainMap.checkPointWithinExtent(curPhoto.StandPointX,curPhoto.StandPointY) == true &&
myMainMap.checkPointWithinExtent(curPhoto.TargetPointX,curPhoto.TargetPointY) == true &&
myMainMap.checkPointWithinExtent(myMapApp.selPhotoPointX1,myMapApp.selPhotoPointY1*-1) == true &&
myMainMap.checkPointWithinExtent(myMapApp.selPhotoPointX2,myMapApp.selPhotoPointY2*-1) == true) {
                drawSelectedPhotoDetail();
            }
            else {
                zoomToPhoto();
            }
            //display a single photo
            displaySinglePhoto(id);
        }
}


//this function draws the angle of view to the selected photo
function drawSelectedPhotoDetail() {
    var id = myMapApp.photoSelectedId;
    var photoSymbolDetailGroup = document.getElementById("photoSymbolDetail");
    //check if we have to remove old geometry
    if (photoSymbolDetailGroup.hasChildNodes()) {
        photoSymbolDetailGroup.removeChild(photoSymbolDetailGroup.firstChild);
    }
    //only draw if the data is available
```

```
    if (myMapApp.photoData[id]) {
        //create temporary group for photo symbols
        var curPhoto = myMapApp.photoData[id].value;
        var photoSymbolDetailTempGroup = document.createElementNS(svgNS,"g");
        photoSymbolDetailTempGroup.setAttributeNS(null,"id","photoSymbolDetailTemp");
        photoSymbolDetailTempGroup.setAttributeNS(null,"pointer-events","none");
        photoSymbolDetailGroup.appendChild(photoSymbolDetailTempGroup);

        var viewAngle = document.createElementNS(svgNS,"path");
        viewAngle.setAttributeNS(null,"stroke","darkred");
        viewAngle.setAttributeNS(null,"stroke-width",(myMainMap.curWidth * 0.001));
        viewAngle.setAttributeNS(null,"fill","darkred");
        viewAngle.setAttributeNS(null,"fill-opacity","0.2");
        var d = "M"+curPhoto.StandPointX+","+(curPhoto.StandPointY*-
1)+"L"+myMapApp.selPhotoPointX1+","+myMapApp.selPhotoPointY1+"A"+myMapApp.selPhotoDistance+","+myMapApp.
selPhotoDistance+" 0 0,1 "+myMapApp.selPhotoPointX2+" "+myMapApp.selPhotoPointY2+"z";
        viewAngle.setAttributeNS(null,"d",d);
        photoSymbolDetailTempGroup.appendChild(viewAngle);

        var connectLine = document.createElementNS(svgNS,"line");
        connectLine.setAttributeNS(null,"stroke","darkred");
        connectLine.setAttributeNS(null,"id","connectLine");
        connectLine.setAttributeNS(null,"stroke-width",(myMainMap.curWidth * 0.002));
        connectLine.setAttributeNS(null,"x1",curPhoto.StandPointX);
        connectLine.setAttributeNS(null,"y1",curPhoto.StandPointY * -1);
        connectLine.setAttributeNS(null,"x2",curPhoto.TargetPointX);
        connectLine.setAttributeNS(null,"y2",curPhoto.TargetPointY * -1);
        photoSymbolDetailTempGroup.appendChild(connectLine);

        var photoSymbol = document.createElementNS(svgNS,"use");
        photoSymbol.setAttributeNS(xlinkNS,"href","#camera");
        photoSymbol.setAttributeNS(null,"x",curPhoto.StandPointX);
        photoSymbol.setAttributeNS(null,"y",curPhoto.StandPointY * -1);
        photoSymbol.setAttributeNS(null,"id","photoDetailSymbol_"+curPhoto.id);
        photoSymbol.setAttributeNS(null,"transform","rotate("+(myMapApp.selPhotoDirection*-1)+",
"+curPhoto.StandPointX+","+(curPhoto.StandPointY * -1)+")");
        photoSymbolDetailTempGroup.appendChild(photoSymbol);
    }
    else {
        myMapApp.photoSelectStatus = false;
        displayPhotosSmall("start",undefined);
        showInfosEmpty();
    }
}


//this function zooms to the selcted photo
function zoomToPhoto() {
    var curPhoto = myMapApp.photoData[myMapApp.photoSelectedId].value;
    xmin = curPhoto.StandPointX;
    if (curPhoto.TargetPointX < xmin) {
        xmin = curPhoto.TargetPointX;
    }
    if (myMapApp.selPhotoPointX1 < xmin) {
        xmin = myMapApp.selPhotoPointX1;
    }
    if (myMapApp.selPhotoPointX2 < xmin) {
        xmin = myMapApp.selPhotoPointX2;
    }
    xmax = curPhoto.StandPointX;
    if (curPhoto.TargetPointX > xmax) {
        xmax = curPhoto.TargetPointX;
    }
    if (myMapApp.selPhotoPointX1 > xmax) {
        xmax = myMapApp.selPhotoPointX1;
    }
    if (myMapApp.selPhotoPointX2 > xmax) {
        xmax = myMapApp.selPhotoPointX2;
    }
    ymin = curPhoto.StandPointY;
    if (curPhoto.TargetPointY < ymin) {
        ymin = curPhoto.TargetPointY;
    }
    if (myMapApp.selPhotoPointY1*-1 < ymin) {
```

```
        ymin = myMapApp.selPhotoPointY1*-1;
    }
    if (myMapApp.selPhotoPointY2*-1 < ymin) {
        ymin = myMapApp.selPhotoPointY2*-1;
    }
    ymax = curPhoto.StandPointY;
    if (curPhoto.TargetPointY > ymax) {
        ymax = curPhoto.TargetPointY;
    }
    if (myMapApp.selPhotoPointY1*-1 > ymax) {
        ymax = myMapApp.selPhotoPointY1*-1;
    }
    if (myMapApp.selPhotoPointY2*-1 > ymax) {
        ymax = myMapApp.selPhotoPointY2*-1;
    }
    var width = xmax - xmin;
    var height = ymax - ymin;
    myMainMap.setNewViewBox(xmin - width * 0.1,ymin - height * 0.1,xmax + width * 0.1,ymax + height * 0.1,true);
}


//this function displays thumbnails
function displayPhotosSmall(groupId,evt) {
    //alter window status bar
    myMapApp.Windows["photoSelectionWindow"].statusTextNode.nodeValue = "Click on a photo to get more information
(photos are ordered descending by date created)";

    //first remove old content, if any
    while (myMapApp.photoGalleryGroup.hasChildNodes()) {
        myMapApp.photoGalleryGroup.removeChild(myMapApp.photoGalleryGroup.firstChild);
    }
    //set selected status to false
    myMapApp.photoSelectStatus = false;
    var photoSymbolDetailGroup = document.getElementById("photoSymbolDetail");
    if (photoSymbolDetailGroup.hasChildNodes()) {
        photoSymbolDetailGroup.removeChild(photoSymbolDetailGroup.firstChild);
    }
    //create temporary group for photos small
    var photosSmallTempGroup = document.createElementNS(svgNS,"g");
    myMapApp.photoGalleryGroup.appendChild(photosSmallTempGroup);
    //display buttons
    myMapApp.buttons["photoGalForward"].showButton();
    myMapApp.buttons["photoGalBackward"].showButton();
    //hide three other buttons
    myMapApp.buttons["zoomToPhoto"].hideButton();
    myMapApp.buttons["higherResolution"].hideButton();
    myMapApp.buttons["returnToGallery"].hideButton();
    if (groupId == "start") {
        myMapApp.photoGalleryOffset = 0;
    }
    if (groupId == "photoGalBackward") {
        myMapApp.photoGalleryOffset -= myMapApp.nrPerPage;
    }
    if (groupId == "photoGalForward") {
        myMapApp.photoGalleryOffset += myMapApp.nrPerPage;
    }
    //need to control visibility of arrows
    var photoGalleryEnd = myMapApp.photoData.length;
    if ((myMapApp.photoGalleryOffset + myMapApp.nrPerPage) < myMapApp.photoData.length) {
        photoGalleryEnd = myMapApp.photoGalleryOffset + myMapApp.nrPerPage;
        myMapApp.buttons["photoGalForward"].activate();
    }
    else {
        myMapApp.buttons["photoGalForward"].deactivate();
    }
    if (myMapApp.photoGalleryOffset == 0) {
        myMapApp.buttons["photoGalBackward"].deactivate();
    }
    else {
        myMapApp.buttons["photoGalBackward"].activate();
    }

    //arrange thumbnails
    var x = 35;
```

```
    var photoGap = 18;
    var bigLength = 120;
    for (var i = myMapApp.photoGalleryOffset;i<photoGalleryEnd;i++) {
        var curPhoto = myMapApp.photoData[i].value;
        file = curPhoto.FileName.split(".");
        var ext = file[1];
        var fileSmall = file[0] + "_thumb200." + ext;
        var photosSmall = document.createElementNS(svgNS,"image");
        photosSmall.setAttributeNS(null,"id","photoSmall_"+curPhoto.id);
        photosSmall.setAttributeNS(xlinkNS,"href",fileSmall);
        photosSmall.addEventListener("mouseover",showPhotoData,false);
        photosSmall.addEventListener("mouseout",showPhotoData,false);
        photosSmall.addEventListener("click",setPhotoExtentCheck,false);
        photosSmall.addEventListener("click",showPhotoData,false);
        var y = 22;
        if (curPhoto.Width > curPhoto.Height) {
            //case landscape
            photosSmall.setAttributeNS(null,"width",bigLength);
            photosSmall.setAttributeNS(null,"height",bigLength * (curPhoto.Height / curPhoto.Width));
            y += (bigLength - bigLength * (curPhoto.Height / curPhoto.Width))/2;
            photosSmall.setAttributeNS(null,"x",x);
            photosSmall.setAttributeNS(null,"y",y);
            x += bigLength + photoGap;
        }
        else {
            //case portrait
            var widthTemp = bigLength * (curPhoto.Width / curPhoto.Height);
            photosSmall.setAttributeNS(null,"height",bigLength);
            photosSmall.setAttributeNS(null,"width",widthTemp);
            x += (bigLength - widthTemp)/2
            photosSmall.setAttributeNS(null,"x",x);
            photosSmall.setAttributeNS(null,"y",y);
            x += widthTemp + (bigLength - widthTemp)/2 + photoGap;

        }
        photosSmallTempGroup.appendChild(photosSmall);
    }
}


//this function displays a small photo with some metadata
function displaySinglePhoto(id) {
    //alter window status bar
    myMapApp.Windows["photoSelectionWindow"].statusTextNode.nodeValue = "Photo detail view - zoom to photo, see it in
higher resolution or return to gallery";
    //first remove old content, if any
    while (myMapApp.photoGalleryGroup.hasChildNodes()) {
        myMapApp.photoGalleryGroup.removeChild(myMapApp.photoGalleryGroup.firstChild);
    }
    //create temporary group for single photo
    var singlePhotoTempGroup = document.createElementNS(svgNS,"g");
    myMapApp.photoGalleryGroup.appendChild(singlePhotoTempGroup);
    //hide photo frame, crosshair and buttons
    myMapApp.photoFrame.setAttributeNS(null,"display","none");
    myMainMap.highlightCross.setAttributeNS(null,"visibility","hidden");
    myMapApp.buttons["photoGalForward"].hideButton();
    myMapApp.buttons["photoGalBackward"].hideButton();

    var curPhoto = myMapApp.photoData[id].value;
    file = curPhoto.FileName.split(".");
    var ext = file[1];
    var fileSmall = file[0] + "_thumb200." + ext;
    var singlePhotoSmall = document.createElementNS(svgNS,"image");
    singlePhotoSmall.setAttributeNS(null,"id","photoSmall_"+curPhoto.id);
    singlePhotoSmall.setAttributeNS(xlinkNS,"href",fileSmall);
    var x = 20;
    var y = 20;
    var bigHeight = 127;
    var widthTemp = bigHeight * (curPhoto.Width / curPhoto.Height);
    if (curPhoto.Width > curPhoto.Height) {
        //case landscape
        singlePhotoSmall.setAttributeNS(null,"width",widthTemp);
        singlePhotoSmall.setAttributeNS(null,"height",bigHeight);
        singlePhotoSmall.setAttributeNS(null,"x",x);
```

```
            singlePhotoSmall.setAttributeNS(null,"y",y);
        }
        else {
            //case portrait
            x += 40;
            singlePhotoSmall.setAttributeNS(null,"height",bigHeight);
            singlePhotoSmall.setAttributeNS(null,"width",widthTemp);
            singlePhotoSmall.setAttributeNS(null,"x",x);
            singlePhotoSmall.setAttributeNS(null,"y",y);
        }
        singlePhotoTempGroup.appendChild(singlePhotoSmall);

        //create temporary group for photo text
        var photoTextTempGroup = document.createElementNS(svgNS,"g");
        photoTextTempGroup.setAttributeNS(null,"id","photoTextTemp");
        photoTextTempGroup.setAttributeNS(null,"pointer-events","none");
        myMapApp.photoGalleryGroup.appendChild(photoTextTempGroup);

        var standPoint = document.createElementNS(svgNS,"text");
        standPoint.setAttributeNS(null,"x",230);
        standPoint.setAttributeNS(null,"y",30);
        standPoint.setAttributeNS(null,"class","allText smallText");
        var textNode = document.createTextNode("Standpoint Coordinates: Y " + curPhoto.StandPointX + "m, X " +
curPhoto.StandPointY + "m");
        standPoint.appendChild(textNode);
        photoTextTempGroup.appendChild(standPoint);

        var targetPoint = document.createElementNS(svgNS,"text");
        targetPoint.setAttributeNS(null,"x",230);
        targetPoint.setAttributeNS(null,"y",45);
        targetPoint.setAttributeNS(null,"class","allText smallText");
        var textNode = document.createTextNode("Viewpoint Coordinates: Y " + curPhoto.TargetPointX + "m, X " +
curPhoto.TargetPointY + "m");
        targetPoint.appendChild(textNode);
        photoTextTempGroup.appendChild(targetPoint);

        var distance = document.createElementNS(svgNS,"text");
        distance.setAttributeNS(null,"x",230);
        distance.setAttributeNS(null,"y",60);
        distance.setAttributeNS(null,"class","allText smallText");
        var textNode = document.createTextNode("Distance (Standpoint - Viewpoint): " +
parseInt(toPolarDist(curPhoto.StandPointX - curPhoto.TargetPointX,curPhoto.StandPointY - curPhoto.TargetPointY)) + "m");
        distance.appendChild(textNode);
        photoTextTempGroup.appendChild(distance);

        var angleOfView = document.createElementNS(svgNS,"text");
        angleOfView.setAttributeNS(null,"x",230);
        angleOfView.setAttributeNS(null,"y",75);
        angleOfView.setAttributeNS(null,"class","allText smallText");
        var textNode = document.createTextNode("Angle of View (calculated from Focal Length): " + curPhoto.AngleOfView +
String.fromCharCode(176));
        angleOfView.appendChild(textNode);
        photoTextTempGroup.appendChild(angleOfView);

        var resolution = document.createElementNS(svgNS,"text");
        resolution.setAttributeNS(null,"x",230);
        resolution.setAttributeNS(null,"y",95);
        resolution.setAttributeNS(null,"class","allText smallText");
        var textNode = document.createTextNode("Original Resolution: " + curPhoto.Width + " x " + curPhoto.Height + " px");
        resolution.appendChild(textNode);
        photoTextTempGroup.appendChild(resolution);

        var owner = document.createElementNS(svgNS,"text");
        owner.setAttributeNS(null,"x",230);
        owner.setAttributeNS(null,"y",115);
        owner.setAttributeNS(null,"class","allText smallText");
        var textNode = document.createTextNode("Owner: " + curPhoto.Owner);
        owner.appendChild(textNode);
        photoTextTempGroup.appendChild(owner);

        var copyright = document.createElementNS(svgNS,"text");
        copyright.setAttributeNS(null,"x",230);
        copyright.setAttributeNS(null,"y",130);
        copyright.setAttributeNS(null,"class","allText smallText");
```

```
    photoTextTempGroup.appendChild(copyright);
    var dy = textFlow("Copyright Notice: " + curPhoto.Copyright,copyright,100,230,13,2);

    //show buttons zoom to this photo, see higher resolution and return to gallery
    myMapApp.buttons["zoomToPhoto"].showButton();
    myMapApp.buttons["higherResolution"].showButton();
    myMapApp.buttons["returnToGallery"].showButton();

}


//this function shows the selected photo in a separate window
function showPhotoHigherResolution() {
    //first change status text of window
    myMapApp.Windows["photoSelectionWindow"].statusTextNode.value = " ";
    var bigLength = 600;
    var curPhoto = myMapApp.photoData[myMapApp.photoSelectedId].value;
    //create filename
    file = curPhoto.FileName.split(".");
    var ext = file[1];
    var fileBig = file[0] + "_thumb600." + ext;
    //create new photo
    var photoBig = document.createElementNS(svgNS,"image");
    photoBig.setAttributeNS(xlinkNS,"href",fileBig);
    photoBig.setAttributeNS(null,"x",1);
    photoBig.setAttributeNS(null,"y",18);
    if (curPhoto.Width > curPhoto.Height) {
        //case landscape
        var width = bigLength;
        var height = bigLength * (curPhoto.Height / curPhoto.Width);
        photoBig.setAttributeNS(null,"width",width);
        photoBig.setAttributeNS(null,"height",height);
    }
    else {
        //case portrait
        var width = bigLength * (curPhoto.Width / curPhoto.Height);
        var height = bigLength;
        photoBig.setAttributeNS(null,"width",width);
        photoBig.setAttributeNS(null,"height",height);
    }

    //create new window
    if (curPhoto.Width > curPhoto.Height) {
        //case landscape
        var windowX = 200;
        var windowY = 100;
    }
    else {
        //case portrait
        var windowX = 300;
        var windowY = 30;
    }

    var windowWidth = width+2;
    var windowHeight = height + 33;
    var freeWindow = false;
    var id = 0;
    while (!freeWindow) {
        if (myMapApp.Windows["photoWindow"+id]) {
            id++;
        }
        else {
            freeWindow = true;
        }
    }
    myMapApp.Windows["photoWindow"+id] = new
Window("photoWindow"+id,"Windows",windowWidth,windowHeight,windowX,windowY,true,0,0,1024,700,true,"nofill
normallinecol thinline","windowBackground","3",true,"Photo Window",curPhoto.ObjectName,true,true,true,"bars","bars","allText
middleText","allText smallText","lightgrayfill nostroke","normalLinestyle thinline",17,14,"destroyWindow");
    myMapApp.Windows["photoWindow"+id].appendContent(photoBig);
    //write objectName in statusbar
    var numberOfChar = parseInt(windowWidth * 0.2);
    var dy =
textFlow(curPhoto.ObjectName,myMapApp.Windows["photoWindow"+id].statusTextNode.parentNode,numberOfChar,3,13,1);
```

```
}


//this function destroys the closed photo windows
function destroyWindow(windowId,status) {
    if (status == "closed") {
        myMapApp.Windows[windowId]=null;
    }
}


//this function gets the right place in the photo gallery when returning from a selected photo
function backToPhotoGallery() {
    if (myMapApp.photoExtentCheck == true) {
        for (var i=0;i<myMapApp.photoData.length;i++) {
            if(myMapApp.photoData[i].value.id == myMapApp.photoSelectedId) {
                myMapApp.photoGalleryOffset = Math.floor(i/myMapApp.nrPerPage) * myMapApp.nrPerPage;
                break;
            }
        }
        displayPhotosSmall("unchanged",undefined);
    }
    else {
        displayPhotosSmall("start",undefined);
    }
}


function setPhotoExtentCheck() {
        //status to determine if extent was changed
        myMapApp.photoExtentCheck = true;
}
```

1.4 sendPhotoData.php

```
<?
header("Content-type: text/xml");
include 'connect_vars.php';
ini_set("display_errors",1);
//get parameters
$xmin = intval($_GET['xmin'])+2000000;
$xmax = intval($_GET['xmax'])+2000000;
$ymin = intval($_GET['ymin'])+1000000;
$ymax = intval($_GET['ymax'])+1000000;
$andOr = $_GET['logicalAndOr'];
$keywordsString = $_GET['keywords'];
$keywords = explode(",",$keywordsString);
$timestamp = $_GET['timestamp'];
$srid = 2056;

//connect to db
$my_pg_connect = pg_Connect('host='.$hostname.' dbname='.$dbName.' user='.$username.' password='.$password) or die
('Cannot connect to database "'.$dbGeomName.'"');

//write timestamp
print '<photos timestamp="'.$timestamp.'">'."\n";

//get number of keywords
$arrayLength = count($keywords);
if (strlen($keywordsString) > 0) {
    $addWhereClause = ' ';
    if ($andOr == 'OR') {
        //construct condition for case OR
        for ($i = 0; $i < $arrayLength; $i++) {
            $addWhereClause .= '\''.$keywords[$i].'\',';
        }
        $addWhereClause = substr($addWhereClause,0,strlen($addWhereClause)-1);
        //SQL statement for case OR
        $mySQL = 'SELECT DISTINCT photos."PhotoID", photos."FileName", photos."ObjectName", photos."Make",
photos."Model", photos."DateTimeOriginal" AS datetime, photos."ExifImageLength", photos."ExifImageWidth",
photos."ExposureTime", photos."FNumber", photos."ISO", photos."FocalLengthIn35mmFormat", photos."UserID",
photos."CopyrightNotice", astext(photos.standpoint_ch1903) AS StandPoint, astext(photos.targetpoint_ch1903) AS TargetPoint,
photos."AngleOfView", photos."Province-State", photos."Sub-location", photos."City" FROM photos, keywords_photos,
keywords WHERE photos.standpoint_ch1903 && setSRID(\'BOX3D('.$xmin.' '.$ymin.', '.$xmax.' '.$ymax.')\'::box3d,'.$srid.') AND
photos."PhotoID" = keywords_photos.photo_id AND keywords_photos.keyword_id = keywords.id AND keywords.keyword IN
('.$addWhereClause.') ORDER BY photos."DateTimeOriginal" DESC';
    }
    if ($andOr == 'AND') {
        //construct condition for case AND
        for ($i = 0; $i < $arrayLength; $i++) {
            $addWhereClause .= '\''.$keywords[$i].'\',';
        }
        $addWhereClause = substr($addWhereClause,0,strlen($addWhereClause)-1);
        //SQL statement for case AND
        $mySQL = 'SELECT DISTINCT photos."PhotoID", photos."FileName", photos."ObjectName", photos."Make",
photos."Model", photos."DateTimeOriginal" AS datetime, photos."ExifImageLength", photos."ExifImageWidth",
photos."ExposureTime", photos."FNumber", photos."ISO", photos."FocalLengthIn35mmFormat", photos."UserID",
photos."CopyrightNotice", astext(photos.standpoint_ch1903) AS StandPoint, astext(photos.targetpoint_ch1903) AS TargetPoint,
photos."AngleOfView", photos."Province-State", photos."Sub-location", photos."City" FROM photos, keywords_photos WHERE
photos.standpoint_ch1903 && setSRID(\'BOX3D('.$xmin.' '.$ymin.', '.$xmax.' '.$ymax.')\'::box3d,'.$srid.') AND photos."PhotoID"
= keywords_photos.photo_id AND keywords_photos.photo_id IN (SELECT keywords_photos.photo_id FROM
keywords_photos,keywords WHERE keywords_photos.keyword_id = keywords.id AND keywords.keyword IN
('.$addWhereClause.') GROUP BY keywords_photos.photo_id HAVING count(keywords_photos.keyword_id) = '.$arrayLength.')
ORDER BY photos."DateTimeOriginal" DESC';
    }
}
else {
    //SQL statement without keywords
    $mySQL = 'SELECT DISTINCT "PhotoID", "FileName", "ObjectName", "Make", "Model", "DateTimeOriginal" AS datetime,
"ExifImageLength", "ExifImageWidth", "ExposureTime", "FNumber", "ISO", "FocalLengthIn35mmFormat", "UserID",
"CopyrightNotice", astext("standpoint_ch1903") AS StandPoint, astext("targetpoint_ch1903") AS TargetPoint, "AngleOfView",
"Province-State", "Sub-location", "City" FROM photos WHERE standpoint_ch1903 && setSRID(\'BOX3D('.$xmin.' '.$ymin.',
'.$xmax.' '.$ymax.')\'::box3d,'.$srid.') ORDER BY "DateTimeOriginal" DESC';
}

//get metadata from database
```

```
$my_result_set = pg_query($my_pg_connect,$mySQL) or die (pg_ErrorMessage());
$numRecs = pg_NumRows($my_result_set);
$i = 0;
while ($i < $numRecs) {
    $resultArray = pg_Fetch_Array($my_result_set, $i);
    //SQL statement for user
    $mySQLuser = 'SELECT users."first_name", users."name" FROM users WHERE users."id" = '.$resultArray['UserID'];
    //get user information
    $my_result_user = pg_query($my_pg_connect,$mySQLuser) or die (pg_ErrorMessage());
    if (pg_NumRows($my_result_user) > 0) {
        $resultUserArray = pg_Fetch_Array($my_result_user, 0);
        $first_name = $resultUserArray['first_name'];
        $last_name = $resultUserArray['name'];
    }
    else {
        $first_name = "undefined";
        $last_name = "undefined";
    }
    //process coordinates
    $replaceArray = array('POINT(',')');
    $standPointCoordsString = str_replace($replaceArray,"",$resultArray['standpoint']);
    $targetPointCoordsString = str_replace($replaceArray,"",$resultArray['targetpoint']);
    $standPointCoords = explode(" ", $standPointCoordsString);
    $targetPointCoords = explode(" ", $targetPointCoordsString);

    //write metadata
    print '<photo id="'.$resultArray['PhotoID'].'"
FileName="'.str_replace("/home/meyer/www/geofotos/","",$resultArray['FileName']).'"
ObjectName="'.htmlspecialchars($resultArray['ObjectName'],ENT_QUOTES).'" Make="'.$resultArray['Make'].'"
Model="'.$resultArray['Model'].'" DateTime="'.$resultArray['datetime'].'" Length="'.$resultArray['ExifImageLength'].'"
Width="'.$resultArray['ExifImageWidth'].'" Exposure="'.$resultArray['ExposureTime'].'" Aperture="'.$resultArray['FNumber'].'"
ISO="'.$resultArray['ISO'].'" FocalLength35="'.$resultArray['FocalLengthIn35mmFormat'].'" UserFirstName="'.$first_name.'"
UserName="'.$last_name.'" Copyright="'.$resultArray['CopyrightNotice'].'" StandPointX="'.(round($standPointCoords[0])-
2000000).'" StandPointY="'.(round($standPointCoords[1])-1000000).'" TargetPointX="'.(round($targetPointCoords[0])-
2000000).'" TargetPointY="'.(round($targetPointCoords[1])-1000000).'" AngleOfView="'.$resultArray['AngleOfView'].'"
ProvinceState="'.$resultArray['Province-State'].'" SubLocation="'.$resultArray['Sub-location'].'" City="'.$resultArray['City'].'">'."\n";

    //SQL statement for keywords
    $mySQLkeywords = 'SELECT DISTINCT keywords."keyword" FROM keywords_photos, keywords WHERE
keywords_photos."photo_id" = '.$resultArray['PhotoID'].' AND keywords_photos."keyword_id" = keywords."id"';
    //get keywords
    $my_result_keywords = pg_query($my_pg_connect,$mySQLkeywords) or die (pg_ErrorMessage());
    $numRecs_key = pg_NumRows($my_result_keywords);
    $j = 0;
    //write keywords
    print "\t".'<keywords>'."\n";
    while ($j < $numRecs_key) {
        $resultKeywordArray = pg_Fetch_Array($my_result_keywords, $j);
        print "\t\t".'<keyword>'.$resultKeywordArray['keyword'].'</keyword>'."\n";
        $j++;
    }
    print "\t".'</keywords>'."\n";

    print '</photo>'."\n";
    $i++;
}

print "</photos>\n";
pg_Close($my_pg_connect);
?>
```

2 EXIF-Definition

2.1 Einleitung

Nachfolgend sind die EXIF-, IPTC- und GPS-Tags mit ihren Definitionen der EXIF-Spezifikation aufgeführt. Die Inhalte der Kap. 2.2, 2.3 und 2.4 sind direkt aus [EXIF2] übernommen. Für jedes Tag sind das definierte Datenformat und mögliche Wertbesetzungen angegeben.

2.2 EXIF-Tags

EXIF stands for 'Exchangeable Image File Format'. This type of information may be found in JPG, TIFF, PNG, MIFF and DNG images.
The EXIF meta information is organized into different Image File Directories (IFD's) within an image. The names of these IFD's correspond to the ExifTool family 1 group names. When writing EXIF information, the default **Group** listed below is used unless another group is specified.

Tag ID	Tag Name	Writable	Group	Values / Notes
0x0001	InteropIndex	string	InteropIFD	
0x0002	InteropVersion	undef	InteropIFD	
0x00fe	SubfileType	int32u!	IFD0	0 = Full-resolution Image 1 = Reduced-resolution image 2 = Single page of multi-page image 3 = Single page of multi-page reduced-resolution image 4 = Transparency mask 5 = Transparency mask of reduced-resolution image 6 = Transparency mask of multi-page image 7 = Transparency mask of reduced-resolution multi-page image
0x00ff	OldSubfileType	int16u!	IFD0	1 = Full-resolution image 2 = Reduced-resolution image 3 = Single page of multi-page image
0x0100	ImageWidth	int32u!	IFD0	
0x0101	ImageHeight	int32u!	IFD0	
0x0102	BitsPerSample	int16u[n]!	IFD0	
0x0103	Compression	int16u!	IFD0	1 = Uncompressed 2 = CCITT 1D 3 = T4/Group 3 Fax 4 = T6/Group 4 Fax 5 = LZW 6 = JPEG (old-style) 7 = JPEG 8 = Adobe Deflate 9 = JBIG B&W 10 = JBIG Color 32766 = Next 32771 = CCIRLEW 32773 = PackBits 32809 = Thunderscan 32895 = IT8CTPAD 32896 = IT8LW 32897 = IT8MP 32898 = IT8BL 32908 = PixarFilm 32909 = PixarLog

				32946 = Deflate 32947 = DCS 34661 = JBIG 34676 = SGILog 34677 = SGILog24 34712 = JPEG 2000 34713 = Nikon NEF Compressed
0x0106	PhotometricInterpretation	int16u!	IFD0	0 = WhiteIsZero 1 = BlackIsZero 2 = RGB 3 = RGB Palette 4 = Transparency Mask 5 = CMYK 6 = YCbCr 8 = CIELab 9 = ICCLab 10 = ITULab 32803 = Color Filter Array 32844 = Pixar LogL 32845 = Pixar LogLuv 34892 = Linear Raw
0x0107	Thresholding	int16u!	IFD0	1 = No dithering or halftoning 2 = Ordered dither or halftone 3 = Randomized dither
0x0108	CellWidth	int16u!	IFD0	
0x0109	CellLength	int16u!	IFD0	
0x010a	FillOrder	int16u!	IFD0	1 = Normal 2 = Reversed
0x010d	DocumentName	string	IFD0	
0x010e	ImageDescription	string	IFD0	
0x010f	Make	string	IFD0	
0x0110	Model	string	IFD0	
0x0111	StripOffsets	N	-	(PreviewImageStart in IFD0 of CR2 files)
	PreviewImageStart	int32u*	IFD0	
0x0112	Orientation	int16u	IFD0	1 = Horizontal (normal) 2 = Mirror horizontal 3 = Rotate 180 4 = Mirror vertical 5 = Mirror horizontal and rotate 270 CW 6 = Rotate 90 CW 7 = Mirror horizontal and rotate 90 CW 8 = Rotate 270 CW
0x0115	SamplesPerPixel	int16u!	IFD0	
0x0116	RowsPerStrip	int32u!	IFD0	
0x0117	StripByteCounts	N	-	(PreviewImageLength in IFD0 of CR2 files)
	PreviewImageLength	int32u*	IFD0	
0x0118	MinSampleValue	int16u	IFD0	
0x0119	MaxSampleValue	int16u	IFD0	
0x011a	XResolution	rational32u	IFD0	
0x011b	YResolution	rational32u	IFD0	
0x011c	PlanarConfiguration	int16u!	IFD0	1 = Chunky 2 = Planar
0x011d	PageName	string	IFD0	
0x011e	XPosition	rational32u	IFD0	
0x011f	YPosition	rational32u	IFD0	
0x0120	FreeOffsets	N	-	
0x0121	FreeByteCounts	N	-	
0x0122	GrayResponseUnit	int16u	IFD0	1 = 0.1 2 = 0.001 3 = 0.0001

				4 = 1e-05
				5 = 1e-06
0x0123	GrayResponseCurve	N	-	
0x0124	T4Options	N	-	
0x0125	T6Options	N	-	
0x0128	ResolutionUnit	int16u	IFD0	1 = None
				2 = inches
				3 = cm
0x0129	PageNumber	int16u[2]	IFD0	
0x012c	ColorResponseUnit	N	-	
0x012d	TransferFunction	N	-	
0x0131	Software	string	IFD0	
0x0132	ModifyDate	string	IFD0	
0x013b	Artist	string	IFD0	
0x013c	HostComputer	string	IFD0	
0x013d	Predictor	int16u!	IFD0	1 = None
				2 = Horizontal differencing
0x013e	WhitePoint	rational32u[2]	IFD0	
0x013f	PrimaryChromaticities	rational32u[6]	IFD0	
0x0140	ColorMap	N	-	
0x0141	HalftoneHints	int16u[2]	IFD0	
0x0142	TileWidth	int32u!	IFD0	
0x0143	TileLength	int32u!	IFD0	
0x0144	TileOffsets	N	-	
0x0145	TileByteCounts	N	-	
0x0146	BadFaxLines	N	-	
0x0147	CleanFaxData	N	-	0 = Clean
				1 = Regenerated
				2 = Unclean
0x0148	ConsecutiveBadFaxLines	N	-	
0x014a	SubIFD	-	-	--> EXIF Tags
0x014c	InkSet	int16u	IFD0	1 = CMYK
				2 = Not CMYK
0x014d	InkNames	N	-	
0x014e	NumberofInks	N	-	
0x0150	DotRange	string	IFD0	
0x0151	TargetPrinter	N	-	
0x0152	ExtraSamples	N	-	
0x0153	SampleFormat	N	-	1 = Unsigned integer
				2 = Two's complement signed integer
				3 = IEEE floating point
				4 = Undefined
				5 = Complex integer
				6 = IEEE floating point
0x0154	SMinSampleValue	N	-	
0x0155	SMaxSampleValue	N	-	
0x0156	TransferRange	N	-	
0x0157	ClipPath	N	-	
0x0158	XClipPathUnits	N	-	
0x0159	YClipPathUnits	N	-	
0x015a	Indexed	N	-	0 = Not indexed
				1 = Indexed
0x015b	JPEGTables	N	-	
0x015f	OPIProxy	N	-	0 = Higher resolution image does not exist
				1 = Higher resolution image exists
0x0190	GlobalParametersIFD	-	-	--> EXIF Tags
0x0191	ProfileType	N	-	0 = Unspecified
				1 = Group 3 FAX
0x0192	FaxProfile	N	-	0 = Unknown
				1 = Minimal B&W lossless, S

				2 = Extended B&W lossless, F
				3 = Lossless JBIG B&W, J
				4 = Lossy color and grayscale, C
				5 = Lossless color and grayscale, L
				6 = Mixed raster content, M
0x0193	CodingMethods	N	-	
0x0194	VersionYear	N	-	
0x0195	ModeNumber	N	-	
0x01b1	Decode	N	-	
0x01b2	DefaultImageColor	N	-	
0x0200	JPEGProc	N	-	1 = Baseline
				14 = Lossless
0x0201	ThumbnailOffset	int32u*	IFD1	
	PreviewImageStart	int32u*	MakerNotes	
	JpgFromRawStart	int32u*	SubIFD	
	JpgFromRawStart	int32u*	IFD2	
	OtherImageStart	N	-	
0x0202	ThumbnailLength	int32u*	IFD1	
	PreviewImageLength	int32u*	MakerNotes	
	JpgFromRawLength	int32u*	SubIFD	
	JpgFromRawLength	int32u*	IFD2	
	OtherImageLength	N	-	
0x0203	JPEGRestartInterval	N	-	
0x0205	JPEGLosslessPredictors	N	-	
0x0206	JPEGPointTransforms	N	-	
0x0207	JPEGQTables	N	-	
0x0208	JPEGDCTables	N	-	
0x0209	JPEGACTables	N	-	
0x0211	YCbCrCoefficients	rational32u[3]	IFD0	
0x0212	YCbCrSubSampling	int16u[2]!	IFD0	'1 1' = YCbCr4:4:4
				'1 2' = YCbCr4:4:0
				'2 1' = YCbCr4:2:2
				'2 2' = YCbCr4:2:0
				'4 1' = YCbCr4:1:1
				'4 2' = YCbCr4:1:0
0x0213	YCbCrPositioning	int16u	IFD0	1 = Centered
				2 = Co-sited
0x0214	ReferenceBlackWhite	rational32u[6]	IFD0	
0x022f	StripRowCounts	N	-	
0x02bc	ApplicationNotes	-	-	--> XMP Tags
0x1000	RelatedImageFileFormat	N	-	
0x1001	RelatedImageWidth	int16u	InteropIFD	
0x1002	RelatedImageLength	int16u	InteropIFD	
0x800d	ImageID	N	-	
0x80a4	WangAnnotation	N	-	
0x80e3	Matteing	N	-	
0x80e4	DataType	N	-	
0x80e5	ImageDepth	N	-	
0x80e6	TileDepth	N	-	
0x827d	Model2	N	-	
0x828d	CFARepeatPatternDim	N	-	
0x828e	CFAPattern2	N	-	
0x828f	BatteryLevel	N	-	
0x8298	Copyright	string	IFD0	
0x829a	ExposureTime	rational32u	ExifIFD	
0x829d	FNumber	rational32u	ExifIFD	
0x830e	PixelScale	N	-	
0x83bb	IPTC-NAA	undef	IFD0	--> IPTC Tags
0x8474	IntergraphPacketData	N	-	
0x847f	IntergraphFlagRegisters	N	-	

0x8480	IntergraphMatrix	N	-	
0x8482	ModelTiePoint	N	-	
0x84e0	Site	N	-	
0x84e1	ColorSequence	N	-	
0x84e2	IT8Header	N	-	
0x84e3	RasterPadding	N	-	
0x84e4	BitsPerRunLength	N	-	
0x84e5	BitsPerExtendedRunLength	N	-	
0x84e6	ColorTable	N	-	
0x84e7	ImageColorIndicator	N	-	
0x84e8	BackgroundColorIndicator	N	-	
0x84e9	ImageColorValue	N	-	
0x84ea	BackgroundColorValue	N	-	
0x84eb	PixelIntensityRange	N	-	
0x84ec	TransparencyIndicator	N	-	
0x84ed	ColorCharacterization	N	-	
0x84ee	HCUsage	N	-	
0x8568	IPTC-NAA2	-	-	--> IPTC Tags
0x85d8	ModelTransform	N	-	
0x8606	LeafData	-	-	--> Leaf Tags
0x8649	PhotoshopSettings	-	-	--> Photoshop Tags
0x8769	ExifOffset	-	-	--> EXIF Tags
0x8773	ICC_Profile	-	-	--> ICC_Profile Tags
0x87ac	ImageLayer	N	-	
0x87af	GeoTiffDirectory	N	-	
0x87b0	GeoTiffDoubleParams	N	-	
0x87b1	GeoTiffAsciiParams	N	-	
0x8822	ExposureProgram	int16u	ExifIFD	1 = Manual 2 = Program AE 3 = Aperture-priority AE 4 = Shutter speed priority AE 5 = Creative (Slow speed) 6 = Action (High speed) 7 = Portrait 8 = Landscape
0x8824	SpectralSensitivity	string	ExifIFD	
0x8825	GPSInfo	-	-	--> GPS Tags
0x8827	ISO	int16u	ExifIFD	
0x8828	Opto-ElectricConvFactor	N	-	
0x8829	Interlace	N	-	
0x882a	TimeZoneOffset	int16s	ExifIFD	
0x882b	SelfTimerMode	int16u	ExifIFD	
0x885c	FaxRecvParams	N	-	
0x885d	FaxSubAddress	N	-	
0x885e	FaxRecvTime	N	-	
0x888a	LeafSubIFD	-	-	--> Leaf SubIFD Tags
0x9000	ExifVersion	undef	ExifIFD	
0x9003	DateTimeOriginal	string	ExifIFD	
0x9004	CreateDate	string	ExifIFD	
0x9101	ComponentsConfiguration	undef	ExifIFD	
0x9102	CompressedBitsPerPixel	rational32u	ExifIFD	
0x9201	ShutterSpeedValue	rational32s	ExifIFD	
0x9202	ApertureValue	rational32u	ExifIFD	
0x9203	BrightnessValue	rational32s	ExifIFD	
0x9204	ExposureCompensation	rational32s	ExifIFD	
0x9205	MaxApertureValue	rational32u	ExifIFD	
0x9206	SubjectDistance	rational32u	ExifIFD	
0x9207	MeteringMode	int16u	ExifIFD	1 = Average 2 = Center-weighted average 3 = Spot

				4 = Multi-spot 5 = Multi-segment 6 = Partial 255 = Other
0x9208	LightSource	int16u	ExifIFD	1 = Daylight 2 = Fluorescent 3 = Tungsten 4 = Flash 9 = Fine Weather 10 = Cloudy 11 = Shade 12 = Daylight Fluorescent 13 = Day White Fluorescent 14 = Cool White Fluorescent 15 = White Fluorescent 17 = Standard Light A 18 = Standard Light B 19 = Standard Light C 20 = D55 21 = D65 22 = D75 23 = D50 24 = ISO Studio Tungsten 255 = Other
0x9209	Flash	int16u	ExifIFD	0x0 = No Flash 0x1 = Fired 0x5 = Fired, Return not detected 0x7 = Fired, Return detected 0x9 = On 0xd = On, Return not detected 0xf = On, Return detected 0x10 = Off 0x18 = Auto, Did not fire 0x19 = Auto, Fired 0x1d = Auto, Fired, Return not detected 0x1f = Auto, Fired, Return detected 0x20 = No flash function 0x41 = Fired, Red-eye reduction 0x45 = Fired, Red-eye reduction, Return not detected 0x47 = Fired, Red-eye reduction, Return detected 0x49 = On, Red-eye reduction 0x4d = On, Red-eye reduction, Return not detected 0x4f = On, Red-eye reduction, Return detected 0x59 = Auto, Fired, Red-eye reduction 0x5d = Auto, Fired, Red-eye reduction, Return not detected 0x5f = Auto, Fired, Red-eye reduction, Return detected
0x920a	FocalLength	rational32u	ExifIFD	
0x920b	FlashEnergy	N	-	
0x920c	SpatialFrequencyResponse	N	-	
0x920d	Noise	N	-	
0x920e	FocalPlaneXResolution	N	-	
0x920f	FocalPlaneYResolution	N	-	
0x9210	FocalPlaneResolutionUnit	N	-	1 = mm 10 = cm '0.001' = um '25.4' = inches
0x9211	ImageNumber	N	-	
0x9212	SecurityClassification	N	-	
0x9213	ImageHistory	N	-	

0x9214	SubjectLocation	int16u[4]	ExifIFD	
0x9215	ExposureIndex	N	-	
0x9216	TIFF-EPStandardID	N	-	
0x9217	SensingMethod	N	-	1 = Not defined 2 = One-chip color area 3 = Two-chip color area 4 = Three-chip color area 5 = Color sequential area 7 = Trilinear 8 = Color sequential linear
0x923f	StoNits	N	-	
0x927c	MakerNoteCanon	undef	ExifIFD	--> Canon Tags
	MakerNoteCasio	undef	ExifIFD	--> Casio Tags
	MakerNoteCasio2	undef	ExifIFD	--> Casio Type2 Tags
	MakerNoteFujiFilm	undef	ExifIFD	--> FujiFilm Tags
	MakerNoteKodak1a	undef	ExifIFD	--> Kodak Tags
	MakerNoteKodak1b	undef	ExifIFD	--> Kodak Tags
	MakerNoteKodak2	undef	ExifIFD	--> Kodak Type2 Tags
	MakerNoteKodak3	undef	ExifIFD	--> Kodak Type3 Tags
	MakerNoteKodak4	undef	ExifIFD	--> Kodak Type4 Tags
	MakerNoteKodak5	undef	ExifIFD	--> Kodak Type5 Tags
	MakerNoteKodak6a	undef	ExifIFD	--> Kodak Type6 Tags
	MakerNoteKodak6b	undef	ExifIFD	--> Kodak Type6 Tags
	MakerNoteKodakUnknown	undef	ExifIFD	--> Kodak Unknown Tags
	MakerNoteKyocera	undef	ExifIFD	--> Unknown Tags
	MakerNoteMinolta	undef	ExifIFD	--> Minolta Tags
	MakerNoteNikon	undef	ExifIFD	--> Nikon Tags
	MakerNoteNikon2	undef	ExifIFD	--> Nikon Type2 Tags
	MakerNoteNikon3	undef	ExifIFD	--> Nikon Tags
	MakerNoteOlympus	undef	ExifIFD	--> Olympus Tags
	MakerNoteLeica	undef	ExifIFD	--> Panasonic Tags
	MakerNotePanasonic	undef	ExifIFD	--> Panasonic Tags
	MakerNotePentax	undef	ExifIFD	--> Pentax Tags
	MakerNoteRicoh	undef	ExifIFD	--> Ricoh Tags
	MakerNoteRicohText	undef	ExifIFD	--> Ricoh Text Tags
	PreviewImage	undef	ExifIFD	(Samsung preview image)
	MakerNoteSanyo	undef	ExifIFD	--> Sanyo Tags
	MakerNoteSigma	undef	ExifIFD	--> Sigma Tags
	MakerNoteSony	undef	ExifIFD	--> Sony Tags
	MakerNoteSonySRF	undef	ExifIFD	--> Sony SRF Tags
	MakerNoteUnknown	undef	ExifIFD	--> Unknown Tags
0x9286	UserComment	undef	ExifIFD	
0x9290	SubSecTime	string	ExifIFD	
0x9291	SubSecTimeOriginal	string	ExifIFD	
0x9292	SubSecTimeDigitized	string	ExifIFD	
0x935c	ImageSourceData	N	-	
0x9c9b	XPTitle	undef	IFD0	(tags 0x9c9b-0x9c9f are used by Windows Explorer; special characters in these values are converted to UTF-8 by default, or Windows Latin1 with the -L option. XPTitle is ignored by Windows Explorer if ImageDescription exists)
0x9c9c	XPComment	undef	IFD0	
0x9c9d	XPAuthor	undef	IFD0	(ignored by Windows Explorer if Artist exists)
0x9c9e	XPKeywords	undef	IFD0	
0x9c9f	XPSubject	undef	IFD0	
0xa000	FlashpixVersion	undef	ExifIFD	
0xa001	ColorSpace	int16u	ExifIFD	1 = sRGB 2 = Adobe RGB 65535 = Uncalibrated
0xa002	ExifImageWidth	int16u	ExifIFD	
0xa003	ExifImageLength	int16u	ExifIFD	

0xa004	RelatedSoundFile	string	ExifIFD	
0xa005	InteropOffset	-	-	--> EXIF Tags
0xa20b	FlashEnergy	rational32u	ExifIFD	
0xa20c	SpatialFrequencyResponse	N	-	
0xa20d	Noise	N	-	
0xa20e	FocalPlaneXResolution	rational32u	ExifIFD	
0xa20f	FocalPlaneYResolution	rational32u	ExifIFD	
0xa210	FocalPlaneResolutionUnit	int16u	ExifIFD	1 = mm 10 = cm '0.001' = um '25.4' = inches
0xa211	ImageNumber	N	-	
0xa212	SecurityClassification	N	-	
0xa213	ImageHistory	N	-	
0xa214	SubjectLocation	int16u[2]	ExifIFD	
0xa215	ExposureIndex	rational32u	ExifIFD	
0xa216	TIFF-EPStandardID	N	-	
0xa217	SensingMethod	int16u	ExifIFD	1 = Not defined 2 = One-chip color area 3 = Two-chip color area 4 = Three-chip color area 5 = Color sequential area 7 = Trilinear 8 = Color sequential linear
0xa300	FileSource	undef	ExifIFD	3 = Digital Camera
0xa301	SceneType	undef	ExifIFD	1 = Directly photographed
0xa302	CFAPattern	undef	ExifIFD	
0xa401	CustomRendered	int16u	ExifIFD	0 = Normal 1 = Custom
0xa402	ExposureMode	int16u	ExifIFD	0 = Auto 1 = Manual 2 = Auto bracket
0xa403	WhiteBalance	int16u	ExifIFD	0 = Auto 1 = Manual
0xa404	DigitalZoomRatio	rational32u	ExifIFD	
0xa405	FocalLengthIn35mmFormat	int16u	ExifIFD	
0xa406	SceneCaptureType	int16u	ExifIFD	0 = Standard 1 = Landscape 2 = Portrait 3 = Night
0xa407	GainControl	int16u	ExifIFD	0 = None 1 = Low gain up 2 = High gain up 3 = Low gain down 4 = High gain down
0xa408	Contrast	int16u	ExifIFD	0 = Normal 1 = Low 2 = High
0xa409	Saturation	int16u	ExifIFD	0 = Normal 1 = Low 2 = High
0xa40a	Sharpness	int16u	ExifIFD	0 = Normal 1 = Soft 2 = Hard
0xa40b	DeviceSettingDescription	N	-	
0xa40c	SubjectDistanceRange	int16u	ExifIFD	1 = Macro 2 = Close 3 = Distant
0xa420	ImageUniqueID	string	ExifIFD	
0xa480	GDALMetadata	N	-	

0xa481	GDALNoData	N	-	
0xa500	Gamma	rational32u	ExifIFD	
0xc427	OceScanjobDesc	N	-	
0xc428	OceApplicationSelector	N	-	
0xc429	OceIDNumber	N	-	
0xc42a	OceImageLogic	N	-	
0xc44f	Annotations	N	-	
0xc4a5	PrintIM	undef	IFD0	--> PrintIM Tags
0xc612	DNGVersion	int8u[4]	IFD0	
0xc613	DNGBackwardVersion	N	-	
0xc614	UniqueCameraModel	string	IFD0	
0xc615	LocalizedCameraModel	string	IFD0	
0xc616	CFAPlaneColor	N	-	
0xc617	CFALayout	N	-	1 = Rectangular 2 = Even columns offset down 1/2 row 3 = Even columns offset up 1/2 row 4 = Even rows offset right 1/2 column 5 = Even rows offset left 1/2 column
0xc618	LinearizationTable	N	-	
0xc619	BlackLevelRepeatDim	N	-	
0xc61a	BlackLevel	N	-	
0xc61b	BlackLevelDeltaH	N	-	
0xc61c	BlackLevelDeltaV	N	-	
0xc61d	WhiteLevel	N	-	
0xc61e	DefaultScale	rational32u[2]	IFD0	
0xc61f	DefaultCropOrigin	int32u[2]	IFD0	
0xc620	DefaultCropSize	int32u[2]	IFD0	
0xc621	ColorMatrix1	N	-	
0xc622	ColorMatrix2	N	-	
0xc623	CameraCalibration1	N	-	
0xc624	CameraCalibration2	N	-	
0xc625	ReductionMatrix1	N	-	
0xc626	ReductionMatrix2	N	-	
0xc627	AnalogBalance	N	-	
0xc628	AsShotNeutral	N	-	
0xc629	AsShotWhiteXY	N	-	
0xc62a	BaselineExposure	rational32s	IFD0	
0xc62b	BaselineNoise	rational32u	IFD0	
0xc62c	BaselineSharpness	rational32u	IFD0	
0xc62d	BayerGreenSplit	int32u	IFD0	
0xc62e	LinearResponseLimit	rational32u	IFD0	
0xc62f	DNGCameraSerialNumber	string	IFD0	
0xc630	DNGLensInfo	rational32u[4]	IFD0	
0xc631	ChromaBlurRadius	rational32u	IFD0	
0xc632	AntiAliasStrength	rational32u	IFD0	
0xc633	ShadowScale	rational32u	IFD0	
0xc634	DNGPrivateData	N	-	
0xc635	MakerNoteSafety	int16u	IFD0	0 = Unsafe 1 = Safe
0xc65a	CalibrationIlluminant1	N	-	1 = Daylight 2 = Fluorescent 3 = Tungsten 4 = Flash 9 = Fine Weather 10 = Cloudy 11 = Shade 12 = Daylight Fluorescent 13 = Day White Fluorescent 14 = Cool White Fluorescent 15 = White Fluorescent

				17 = Standard Light A 18 = Standard Light B 19 = Standard Light C 20 = D55 21 = D65 22 = D75 23 = D50 24 = ISO Studio Tungsten 255 = Other
0xc65b	CalibrationIlluminant2	N	-	1 = Daylight 2 = Fluorescent 3 = Tungsten 4 = Flash 9 = Fine Weather 10 = Cloudy 11 = Shade 12 = Daylight Fluorescent 13 = Day White Fluorescent 14 = Cool White Fluorescent 15 = White Fluorescent 17 = Standard Light A 18 = Standard Light B 19 = Standard Light C 20 = D55 21 = D65 22 = D75 23 = D50 24 = ISO Studio Tungsten 255 = Other
0xc65c	BestQualityScale	rational32u	IFD0	
0xc660	AliasLayerMetadata	N	-	
0xfde8	OwnerName	string	ExifIFD	(tags 0xfde8-0xfe58 are generated by Photoshop Camera RAW -- some names are the same as other EXIF tags, but ExifTool will avoid writing these unless they already exist in the file)
0xfde9	SerialNumber	string	ExifIFD	
0xfdea	Lens	string	ExifIFD	
0xfe4c	RawFile	string	ExifIFD	
0xfe4d	Converter	string	ExifIFD	
0xfe4e	WhiteBalance	string	ExifIFD	
0xfe51	Exposure	string	ExifIFD	
0xfe52	Shadows	string	ExifIFD	
0xfe53	Brightness	string	ExifIFD	
0xfe54	Contrast	string	ExifIFD	
0xfe55	Saturation	string	ExifIFD	
0xfe56	Sharpness	string	ExifIFD	
0xfe57	Smoothness	string	ExifIFD	
0xfe58	MoireFilter	string	ExifIFD	

2.3 IPTC-Tags

IPTC stands for 'International Press Telecommunications Council'. This is an older meta information format that is slowly being phased out in favor of XMP. IPTC information may be embedded in JPG, TIFF, PNG, MIFF, PS, PDF, PSD and DNG images.
The IPTC specification dictates a length for ASCII (string or digits) values. These lengths are given in square brackets after the **Writable** format name. For tags where a range of lengths is allowed, the minimum and maximum lengths are separated by a comma within the brackets. IPTC strings are not null terminated.
IPTC information is separated into different records, each of which has its own set of tags.

Record	Tag Name	Writable	Values / Notes
1	IPTCEnvelope	-	--> IPTC EnvelopeRecord Tags
2	IPTCApplication	-	--> IPTC ApplicationRecord Tags
3	IPTCNewsPhoto	-	--> IPTC NewsPhoto Tags
7	IPTCPreObjectData	-	--> IPTC PreObjectData Tags
8	IPTCObjectData	-	--> IPTC ObjectData Tags
9	IPTCPostObjectData	-	--> IPTC PostObjectData Tags

IPTC EnvelopeRecord Tags

Tag ID	Tag Name	Writable	Values / Notes
0	EnvelopeRecordVersion	int16u	
5	Destination	string[0,1024]	
20	FileFormat	int16u	0 = No ObjectData 1 = IPTC-NAA Digital Newsphoto Parameter Record 2 = IPTC7901 Recommended Message Format 3 = Tagged Image File Format (Adobe/Aldus Image data) 4 = Illustrator (Adobe Graphics data) 5 = AppleSingle (Apple Computer Inc) 6 = NAA 89-3 (ANPA 1312) 7 = MacBinary II 8 = IPTC Unstructured Character Oriented File Format (UCOFF) 9 = United Press International ANPA 1312 variant 10 = United Press International Down-Load Message 11 = JPEG File Interchange (JFIF) 12 = Photo-CD Image-Pac (Eastman Kodak) 13 = Bit Mapped Graphics File [.BMP] (Microsoft) 14 = Digital Audio File [.WAV] (Microsoft & Creative Labs) 15 = Audio plus Moving Video [.AVI] (Microsoft) 16 = PC DOS/Windows Executable Files [.COM][.EXE] 17 = Compressed Binary File [.ZIP] (PKWare Inc) 18 = Audio Interchange File Format AIFF (Apple Computer Inc) 19 = RIFF Wave (Microsoft Corporation) 20 = Freehand (Macromedia/Aldus) 21 = Hypertext Markup Language [.HTML] (The Internet Society) 22 = MPEG 2 Audio Layer 2 (Musicom), ISO/IEC 23 = MPEG 2 Audio Layer 3, ISO/IEC 24 = Portable Document File [.PDF] Adobe 25 = News Industry Text Format (NITF) 26 = Tape Archive [.TAR] 27 = Tidningarnas Telegrambyra NITF version (TTNITF DTD) 28 = Ritzaus Bureau NITF version (RBNITF DTD) 29 = Corel Draw [.CDR]
22	FileVersion	int16u	
30	ServiceIdentifier	string[0,10]	
40	EnvelopeNumber	digits[8]	
50	ProductID	string[0,32]	
60	EnvelopePriority	digits[1]	

70	DateSent	digits[8]
80	TimeSent	string[11]
90	CodedCharacterSet	string[0,32]
100	UniqueObjectName	string[14,80]
120	ARMIdentifier	int16u
122	ARMVersion	int16u

IPTC ApplicationRecord Tags

Tag ID	Tag Name	Writable	Values / Notes
0	ApplicationRecordVersion	int16u	
3	ObjectTypeReference	string[3,67]	
4	ObjectAttributeReference	string[4,68]	
5	ObjectName	string[0,64]	
7	EditStatus	string[0,64]	
8	EditorialUpdate	digits[2]	
10	Urgency	digits[1]	
12	SubjectReference	string[13,236]	
15	Category	string[0,3]	
20	SupplementalCategories	string[0,32]	
22	FixtureIdentifier	string[0,32]	
25	Keywords	string[0,64]	
26	ContentLocationCode	string[3]	
27	ContentLocationName	string[0,64]	
30	ReleaseDate	digits[8]	
35	ReleaseTime	string[11]	
37	ExpirationDate	digits[8]	
38	ExpirationTime	string[11]	
40	SpecialInstructions	string[0,256]	
42	ActionAdvised	digits[2]	01 = Object Kill 02 = Object Replace 03 = Ojbect Append 04 = Object Reference
45	ReferenceService	string[0,10]	
47	ReferenceDate	digits[8]	
50	ReferenceNumber	digits[8]	
55	DateCreated	digits[8]	
60	TimeCreated	string[11]	
62	DigitalCreationDate	digits[8]	
63	DigitalCreationTime	string[11]	
65	OriginatingProgram	string[0,32]	
70	ProgramVersion	string[0,10]	
75	ObjectCycle	string[1]	'a' = Morning 'b' = Both Morning and Evening 'p' = Evening
80	By-line	string[0,32]	
85	By-lineTitle	string[0,32]	
90	City	string[0,32]	
92	Sub-location	string[0,32]	
95	Province-State	string[0,32]	
100	Country-PrimaryLocationCode	string[3]	
101	Country-PrimaryLocationName	string[0,64]	
103	OriginalTransmissionReference	string[0,32]	
105	Headline	string[0,256]	
110	Credit	string[0,32]	
115	Source	string[0,32]	
116	CopyrightNotice	string[0,128]	
118	Contact	string[0,128]	
120	Caption-Abstract	string[0,2000]	

121	LocalCaption	string[0,256]	(I haven't found a reference for the format of tags 121, 184-188 and 225-232, so I have just make them writable as strings with reasonable length. Beware that if this is wrong, other utilities won't be able to read these tags as written by ExifTool.)
122	Writer-Editor	string[0,32]	
125	RasterizedCaption	string[7360]	
130	ImageType	string[2]	
131	ImageOrientation	string[1]	'L' = Landscape 'P' = Portrait 'S' = Square
135	LanguageIdentifier	string[2,3]	
150	AudioType	string[2]	'0T' = Text Only '1A' = Mono Actuality '1C' = Mono Question and Answer Session '1M' = Mono Music '1Q' = Mono Response to a Question '1R' = Mono Raw Sound '1S' = Mono Scener '1V' = Mono Voicer '1W' = Mono Wrap '2A' = Stereo Actuality '2C' = Stereo Question and Answer Session '2M' = Stereo Music '2Q' = Stereo Response to a Question '2R' = Stereo Raw Sound '2S' = Stereo Scener '2V' = Stereo Voicer '2W' = Stereo Wrap
151	AudioSamplingRate	digits[6]	
152	AudioSamplingResolution	digits[2]	
153	AudioDuration	digits[6]	
154	AudioOutcue	string[0,64]	
184	JobID	string[0,64]	
185	MasterDocumentID	string[0,256]	
186	ShortDocumentID	string[0,64]	
187	UniqueDocumentID	string[0,128]	
188	OwnerID	string[0,128]	
200	ObjectPreviewFileFormat	int16u	0 = No ObjectData 1 = IPTC-NAA Digital Newsphoto Parameter Record 2 = IPTC7901 Recommended Message Format 3 = Tagged Image File Format (Adobe/Aldus Image data) 4 = Illustrator (Adobe Graphics data) 5 = AppleSingle (Apple Computer Inc) 6 = NAA 89-3 (ANPA 1312) 7 = MacBinary II 8 = IPTC Unstructured Character Oriented File Format (UCOFF) 9 = United Press International ANPA 1312 variant 10 = United Press International Down-Load Message 11 = JPEG File Interchange (JFIF) 12 = Photo-CD Image-Pac (Eastman Kodak) 13 = Bit Mapped Graphics File [.BMP] (Microsoft) 14 = Digital Audio File [.WAV] (Microsoft & Creative Labs) 15 = Audio plus Moving Video [.AVI] (Microsoft) 16 = PC DOS/Windows Executable Files [.COM][.EXE] 17 = Compressed Binary File [.ZIP] (PKWare Inc) 18 = Audio Interchange File Format AIFF (Apple Computer Inc) 19 = RIFF Wave (Microsoft Corporation) 20 = Freehand (Macromedia/Aldus) 21 = Hypertext Markup Language [.HTML] (The Internet

			Society) 22 = MPEG 2 Audio Layer 2 (Musicom), ISO/IEC 23 = MPEG 2 Audio Layer 3, ISO/IEC 24 = Portable Document File [.PDF] Adobe 25 = News Industry Text Format (NITF) 26 = Tape Archive [.TAR] 27 = Tidningarnas Telegrambyra NITF version (TTNITF DTD) 28 = Ritzaus Bureau NITF version (RBNITF DTD) 29 = Corel Draw [.CDR]
201	ObjectPreviewFileVersion	int16u	
202	ObjectPreviewData	string[0,256000]	
225	ClassifyState	string[0,64]	
228	SimilarityIndex	string[0,32]	
230	DocumentNotes	string[0,1024]	
231	DocumentHistory	string[0,256]	
232	ExifCameraInfo	string[0,4096]	

IPTC NewsPhoto Tags

Tag ID	Tag Name	Writable	Values / Notes
0	NewsPhotoVersion	int16u	
10	IPTCPictureNumber	string[16]	(4 numbers: 1-Manufacturer ID, 2-Equipment ID, 3-Date, 4-Sequence)
20	IPTCImageWidth	int16u	
30	IPTCImageHeight	int16u	
40	IPTCPixelWidth	int16u	
50	IPTCPixelHeight	int16u	
55	SupplementalType	int8u	0 = Main Image 1 = Reduced Resolution Image 2 = Logo 3 = Rasterized Caption
60	ColorRepresentation	int16u	0x0 = No Image, Single Frame 0x100 = Monochrome, Single Frame 0x300 = 3 Components, Single Frame 0x301 = 3 Components, Frame Sequential in Multiple Objects 0x302 = 3 Components, Frame Sequential in One Object 0x303 = 3 Components, Line Sequential 0x304 = 3 Components, Pixel Sequential 0x305 = 3 Components, Special Interleaving 0x400 = 4 Components, Single Frame 0x401 = 4 Components, Frame Sequential in Multiple Objects 0x402 = 4 Components, Frame Sequential in One Object 0x403 = 4 Components, Line Sequential 0x404 = 4 Components, Pixel Sequential 0x405 = 4 Components, Special Interleaving
64	InterchangeColorSpace	int8u	1 = X,Y,Z CIE 2 = RGB SMPTE 3 = Y,U,V (K) (D65) 4 = RGB Device Dependent 5 = CMY (K) Device Dependent 6 = Lab (K) CIE 7 = YCbCr 8 = sRGB
65	ColorSequence	int8u	
66	ICC_Profile	N	
70	ColorCalibrationMatrix	N	
80	LookupTable	N	
84	NumIndexEntries	int16u	

85	ColorPalette	N	
86	IPTCBitsPerSample	int8u	
90	SampleStructure	int8u	0 = OrthogonalConstangSampling 1 = Orthogonal4-2-2Sampling 2 = CompressionDependent
100	ScanningDirection	int8u	0 = L-R, Top-Bottom 1 = R-L, Top-Bottom 2 = L-R, Bottom-Top 3 = R-L, Bottom-Top 4 = Top-Bottom, L-R 5 = Bottom-Top, L-R 6 = Top-Bottom, R-L 7 = Bottom-Top, R-L
102	IPTCImageRotation	int8u	0 = 0 1 = 90 2 = 180 3 = 270
110	DataCompressionMethod	int32u	
120	QuantizationMethod	int8u	0 = Linear Reflectance/Transmittance 1 = Linear Density 2 = IPTC Ref B 3 = Linear Dot Percent 4 = AP Domestic Analogue 5 = Compression Method Specific 6 = Color Space Specific 7 = Gamma Compensated
125	EndPoints	N	
130	ExcursionTolerance	int8u	0 = Not Allowed 1 = Allowed
135	BitsPerComponent	int8u	
140	MaximumDensityRange	int16u	
145	GammaCompensatedValue	int16u	

IPTC PreObjectData Tags

Tag ID	Tag Name	Writable	Values / Notes
10	SizeMode	N	0 = Size Not Known 1 = Size Known
20	MaxSubfileSize	N	
90	ObjectSizeAnnounced	N	
95	MaximumObjectSize	N	

IPTC ObjectData Tags

Tag ID	Tag Name	Writable	Values / Notes
10	SubFile	N	

IPTC PostObjectData Tags

Tag ID	Tag Name	Writable	Values / Notes
10	ConfirmedObjectSize	N	

2.4 GPS-Tags

These GPS tags are part of the EXIF standard, and are stored in a separate IFD within the EXIF information.

ExifTool is very flexible about the input format for lat/long coordinates, and will accept 3 floating point numbers (for degrees, minutes and seconds) separated by just about anything. Many other GPS tags have values which are fixed-length strings. For these, the indicated string lengths include a null terminator which is added automatically by ExifTool.

When adding GPS information to an image, it is important to set all of the following tags: GPSLatitude, GPSLatitudeRef, GPSLongitude, GPSLongitudeRef, GPSAltitude and GPSAltitudeRef.

Tag ID	Tag Name	Writable	Values / Notes
0x0000	GPSVersionID	int8u[4]	
0x0001	GPSLatitudeRef	string[2]	'N' = North 'S' = South
0x0002	GPSLatitude	rational32u[3]	
0x0003	GPSLongitudeRef	string[2]	'E' = East 'W' = West
0x0004	GPSLongitude	rational32u[3]	
0x0005	GPSAltitudeRef	int8u	0 = Above Sea Level 1 = Below Sea Level
0x0006	GPSAltitude	rational32u	
0x0007	GPSTimeStamp	rational32u[3]	
0x0008	GPSSatellites	string	
0x0009	GPSStatus	string[2]	'A' = Measurement In Progress 'V' = Measurement Interoperability
0x000a	GPSMeasureMode	string[2]	2 = 2-Dimensional Measurement 3 = 3-Dimensional Measurement
0x000b	GPSDOP	rational32u	
0x000c	GPSSpeedRef	string[2]	'K' = km/h 'M' = mph 'N' = knots
0x000d	GPSSpeed	rational32u	
0x000e	GPSTrackRef	string[2]	'M' = Magnetic North 'T' = True North
0x000f	GPSTrack	rational32u	
0x0010	GPSImgDirectionRef	string[2]	'M' = Magnetic North 'T' = True North
0x0011	GPSImgDirection	rational32u	
0x0012	GPSMapDatum	string	
0x0013	GPSDestLatitudeRef	string[2]	'N' = North 'S' = South
0x0014	GPSDestLatitude	rational32u[3]	
0x0015	GPSDestLongitudeRef	string[2]	'E' = East 'W' = West
0x0016	GPSDestLongitude	rational32u[3]	
0x0017	GPSDestBearingRef	string[2]	'M' = Magnetic North 'T' = True North
0x0018	GPSDestBearing	rational32u	
0x0019	GPSDestDistanceRef	string[2]	'K' = Kilometers 'M' = Miles 'N' = Nautical Miles
0x001a	GPSDestDistance	rational32u	
0x001b	GPSProcessingMethod	undef	
0x001c	GPSAreaInformation	undef	
0x001d	GPSDateStamp	string[11]	
0x001e	GPSDifferential	int16u	0 = No Correction 1 = Differential Corrected