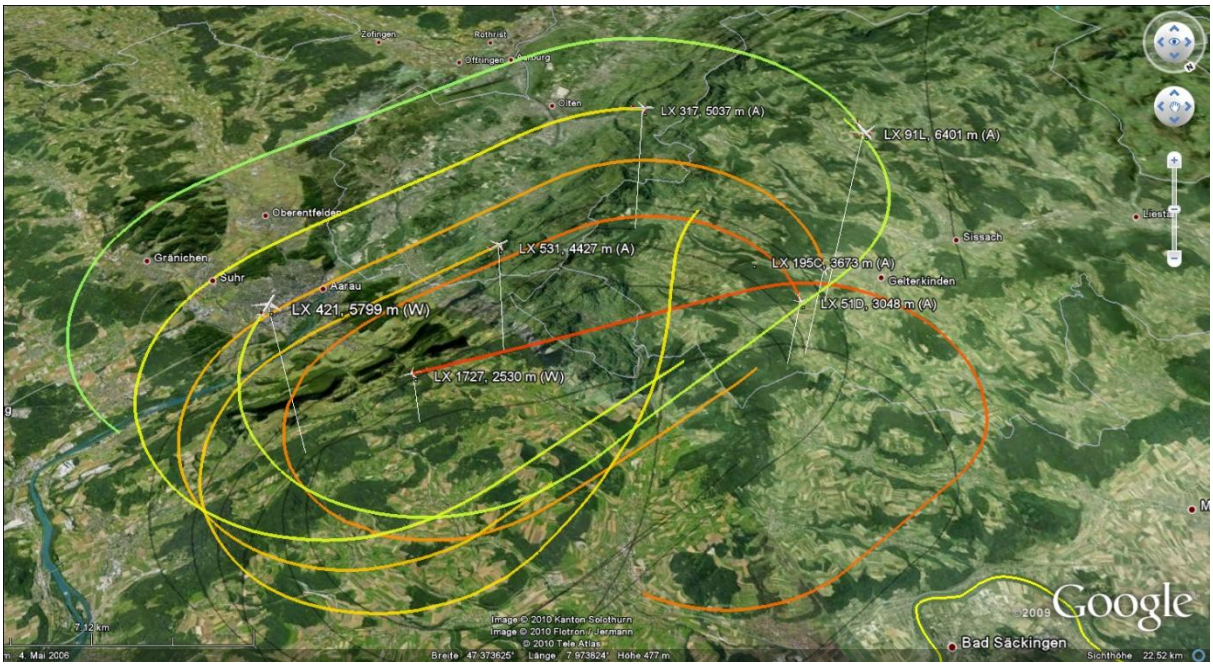


Visualizing traffic data with Google Earth™: Real-time Air Traffic in 3D

Albert Steiner, IDP, ZHAW

Conference paper STRC 2010

STRC | 10th Swiss Transport Research Conference
Monte Verità / Ascona, September 1 - 3, 2010

Visualizing traffic data with Google Earth™: Real-time Air Traffic in 3D

Albert Steiner
IDP
ZHAW

Rosenstrasse 3
8401 Winterthur
Switzerland

Phone: +41 (0)58 934 78 01
Fax: +41 (0)58 935 78 01
Email: albert.steiner@zhaw.ch

September 2010

Abstract

During the last decades, traffic demand increased continuously for all transport modes, while supply, i.e. infrastructures and their management, could often not keep pace with this growth. Some of the consequences are severe congestions, delays, emissions (green-house gases, particles), all leading to enormous costs. One important input to optimally manage traffic systems is accurate and reliable information on the system's current and predicted state. Depending on the transport mode under consideration, a variety of traffic data sources is available, capturing either local conditions or the movement of objects through time and space. From these data, appropriate traffic information can be extracted and based on this, management decisions can be made and/or services provided. Due to the widespread use of intelligent communication technologies (ICT) and the arising convergence of communication, navigation, and positioning technologies/devices, many new services have emerged, often belonging to the group of so called location based services (LBS). Regarding the field of transport, these services include, amongst others, information on traffic state, recommendations of routes to travel together with corresponding travel time information, availability of taxis or public transport connections. With mobile or computer based applications and a variety of tools specialized in visualizing traffic and geographic information as well as other contents (e.g., Google Earth™), traffic participants are able to get appropriate and useful information while travelling through transport networks. In this paper, we present a framework to compute and visualize traffic information in real-time and in two or three dimensional space. To demonstrate its practical use, we apply the approach to real-time data from air traffic over parts of Switzerland and Central Europe, where trajectories of a number of airplanes equipped with ADS-B are visualized. After a system's overview we provide an introduction to the algorithm, discuss computational aspects and present some examples using real data.

Keywords

Traffic data – Air traffic – Real-time visualization – Traffic state – Google Earth – Geo-information – ADS-B – Interpolation

1. Introduction

During the last decades, traffic demand increased more or less continuously, holding for all transport modes, and for passengers as well as for goods. On the other hand, supply, i.e. infrastructures and their management, could often not keep pace with this growth. Some of the consequences are severe congestions, delays, emissions (green-house gases, particles), all leading to enormous current and future costs.

One important input to both, optimally manage traffic systems at an operational level as well as to navigate efficiently through these systems, is accurate and reliable information on the system's current and predicted state. As the systems become increasingly complex, it is essential, that the information available is presented in a clear and easy to understand way to all parties involved, e.g. traffic management and travellers. Depending on the transport mode under consideration, a variety of traffic data sources is available, capturing either local conditions or the movement of objects (passengers, vehicles, goods, etc.) through time and space (two- or three-dimensional). From these data, information on traffic conditions can be extracted and based on this, management decisions can be made and pre-trip or en-route information can be provided to travellers to improve their trip-planning.

Due to the widespread use of intelligent communication technologies (ICT) and the arising convergence of communication, navigation, and positioning technologies/devices, many new services have emerged, often subsumed by the term location based services (LBS). Regarding the field of transport, these services include, amongst many others, information on traffic state, recommendations on routes to travel together with corresponding travel time information, time tables, or on the availability of taxis or public transport connections near the current location.

With the recent availability of computer based and mobile ICT-devices/tools (e.g., Google Maps, Google Earth™, Microsoft Bing™ Maps (Virtual Earth 3D), Yahoo!® Maps, or Smartphone applications (e.g., iPhone® Apps, BlackBerry®'s App World); for details see the corresponding web links in the reference section at the end of this document), many dedicated applications became available, like for example (i) online railway time tables and flight schedules, (ii) weather and climate information, (iii) tourist, leisure and shopping information, or (iv) full navigation systems (e.g., iPhone App «TomTom Europe» from TomTom™). The number of devices/tools and the range of the content provided is huge and continuously growing. Thus, we cannot cover the many aspects in this paper, but refer instead on literature available on this topic, e.g., Brimicombe and Li (2009), covering commercial as well as technical aspects and Küpper (2005), which has a more technical focus, and on the references therein.

Besides commercial interests, regarding mobility and sustainability, one major goal of using these tools shall be to provide traffic (and related) information in an appropriate way (i.e. easy handling, clear visualization, etc.) to facilitate fast, reliable, safe and secure transport at low energy consumption (per passenger- or ton-kilometre) and low emissions. Although these are very general and ambitious goals, we feel confident that on the long run, tools and technologies developed should be assessed with regard to these criteria, even if many of these tools and devices are often only a small component within a larger system or concept.

In this paper, we present a framework to compute and visualize data in real-time and three dimensions. As interesting three-dimensional data from a related project (see Kramarz and Loeber (2007) and the Radar project (2007)) were kindly provided by the project investigators, we demonstrate the practical use of the framework by applying it to real-time data from air traffic over parts of Switzerland and surrounding Central Europe. The data used stem from a number of airplanes equipped with ADS-B[1] devices (for technical details see section see 3.1 and the references mentioned), which allows for sending information on position, velocity, etc. to a ground station or other airplanes.

The framework can be adapted and extended to other dynamic or static data from transport or other fields, with reasonable effort. The focus of this project was on processing and visualizing data, in particular on presenting information in three dimensions, hence making full use of the three-dimensional capabilities of Google Earth.

The rest of the document is organised as follows: In section 2 we start by explaining the general framework, including the steps from ADS-B data sent by airplanes up to the real-time visualization. In section 3 we discuss the overall procedure, provide a brief introduction to the data transmitted and its processing, and explain some of the key steps in a bit more detail. In section 4 we present some results, including screen shots of the application. (We refer here also to the project website, where the application files and various supplementary information can be downloaded.) Finally, section 5 provides a summary and a brief outlook to future research and on other potential applications.

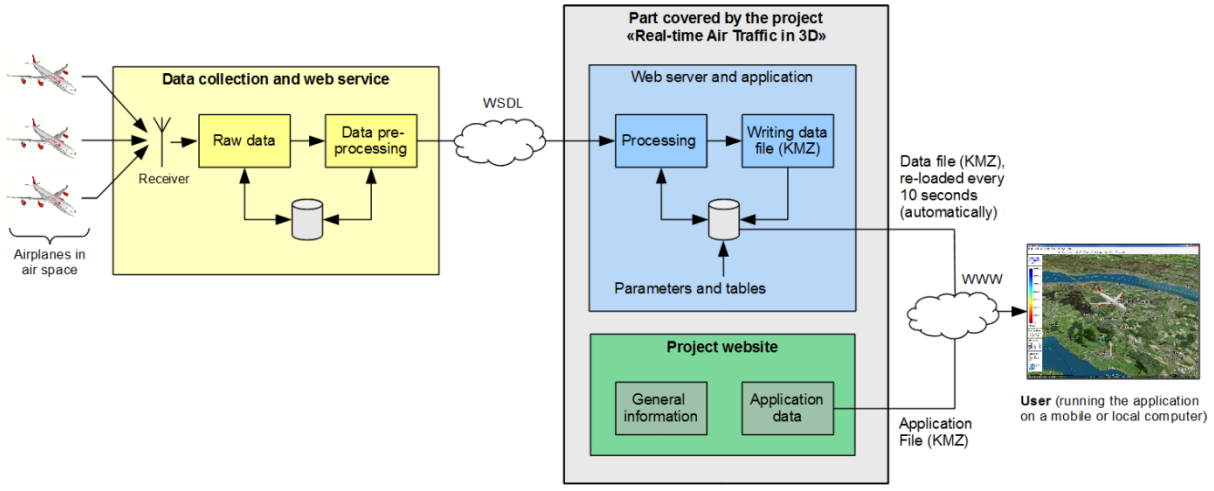
[1] ADS-B: Automatic Dependent Surveillance-Broadcast. For references see those mentioned in section 3.1 and explanations, references and links found on Wikipedia (by searching for ADS-B).

2. System overview

In this section we provide an overview on the components of the system under consideration and how these elements are interrelated.

The overall system consists of four main parts: (i) The air space and the airplanes sending data, (ii) An ADS-B device, which receives data sent by the airplanes, a subsequent pre-processing unit that extracts the variables from the raw data and a web service that provides the airplane specific information for subsequent use, (iii) The main components of the project presented here: a web server on which the application is running, where the main tasks of this application are the computation of the trajectories and the conversion of their description into KML[2]-format, such that these and additional information (e.g., colour bar with colours representing altitude levels, update time) can be watched with Google Earth (GE) (for technical details on software and hardware used, we refer again to the project website as listed in the references below), and finally (iv) The user equipped with a standard computer device with GE to watch air traffic in real-time.

Figure 1 Components of the overall system. The project presented here covers the part in the light gray box.



[2] KML: Keyhole Markup Language. An XML (Extensible Markup Language)-based language schema for expressing geographic annotation and visualization of two- and three-dimensional information (for details we refer to KML Reference (2010)).

The receiver used in this project and the implementation of the software for data collection, pre-processing and providing the web service was developed as part of the Radar project (see Radar project (2010) and Kramarz and Loeber (2007)). Thanks to their previous work, we have access to these data.

To run the application on a local machine, two files (in KMZ[3]-format) are required, but only one of them needs to be downloaded by the user from the project website (only once):

- The "application file" contains the path of the web server, i.e. where the "data file" is located, and the update interval (e.g., 10 seconds). This file needs to be downloaded and stored locally by the user only once. By double-clicking the stored file, both GE and the visualization application start automatically. Detailed instructions can be found on the project website.
- The "data file" comprehends the actual data. This file is updated by the programme running on the web server about every 5 seconds and contains all trajectory data, paths for screen overlays (date and time, number of airplanes detected, etc.), the 3D-model of the airplanes, etc.

[3] KMZ: Files with extension KMZ are simply zipped KML files, i.e. the KML-file is zipped and the file extension is changed for example from ZIP to KMZ. As for any other compressed file, zipping reduces the file size to some extent (in our case by around 85%), which speeds up the file download significantly.

3. Airplane data, overall process and data processing

Starting from the system overview illustrated in Figure 2, we now have a closer look on the airplane data requested via WSDL[4] and the major computational steps performed to finally get the data file required by the application file (see previous section).

3.1 Airplane data

The data transmitted by the airplanes consist of dynamically changing variables determined by appropriate measurement devices in the airplane (e.g., position via GPS) and some (static) parameters (e.g., call sign, transponder address). As for the visualization of the trajectories only a subset of these variables and parameters is actually required, we restrict the list to the values shown in Table 1, together with a short explanation.

Table 1 Some of the data received from airplanes equipped with ADS-B devices.

#	Name	Short description
1	baro_altitude	Altitude (h) in meters above sea level, in ft (converted to metres later on)
2	icao	24-bit mode S transponder address
3	longitude	Longitude (λ) in degrees, with $\lambda = 0$ at the prime meridian and $-\pi < \lambda \leq \pi$
4	latitude	Latitude (φ) in degrees, with $\varphi = 0$ at the equator and $-\pi/2 < \varphi \leq \pi/2$
5	timestamp	Time stamp t in UTC (Universal Time Coordinated), i.e. in milliseconds since January 1, 1970
6	identification	Call sign of flight (see Libhomeradar.org website for details)
7	ew_velocity	Velocity in east-west direction, in knots (converted to m/s later on)
8	ns_velocity	Velocity in north-south direction, in knots (converted to m/s later on)

We will not explain the technical details regarding ADS-B here, but instead refer to EUROCONTROL (undated), EUROCONTROL (2010), Honorjeff et al. (2010), EUROCONTROL (2006), and other technical references. Additional information can be found also in Kramarz and Loeber (2007).

[4] WSDL: Web Services Description Language. WSDL is a description language based on XML for web services to exchange information, i.e. it provides a model to describe web services. It is independent of the platform, programming language and protocol used. A detailed technical introduction can be found on http://www.w3.org/TR/wsdl20-primer/.

Regarding the transmission rate of the different variables/parameters with ADS-B, it is important to note, that not all variables are transmitted together. There are specified message types, each of which includes defined variables and parameters. According to Garrity (2006), ADS-B permits high update rates, or, to be more concrete: (i) 2 position updates per second, (ii) 2 velocity updates per second, (iii) 2 'on event' updates per second, (iv) 0.2 flight identification updates per second (i.e., one update every five seconds). Further information on the different ADS-B messages can be found also in Kramarz and Loeber (2007).

3.2 Overall process

The overall process, illustrated in Figure 2 as pseudo code, is pretty much straightforward. However, various computational steps require special attention and careful implementation to make sure that trajectories visualized are correct and that the application runs efficiently and stable. To keep explanations compact, we will discuss in detail only the important task of performing interpolations in case of missing data.

Figure 2 Pseudo code of the main elements of the algorithm. We only show the version for running the application on a web server, as the version running on a local computer and the one collecting and processing data only are not discussed in this paper.

```
Read parameters (update time intervall ΔT, etc.)
Read kml keyword and format set
Read airline codes and flight codes
Initialize connection to web service
Initialize variables and sets (set with all available airplanes I_all = Ø (empty set), t = ΔT)
While application_running is true
    Set activity flag for cron job on server (to detect application crash)
    Read data from ring buffer (provided by web service) for last time interval [t-ΔT, t) → matrix S
    If all required fields (icao, timestamp, lon, lat, alt) are available in S
        Search and remove invalid entries
            → Sorted set of current valid identification numbers I_curr = {id_p, ..., id_q} (with p < q)
        Convert valid variables from all fields to the required format
        Determine airplane identification numbers (ids) from airplanes that have sent new data
            Determine set of ids (incl. properties) of new airplanes I_new = I_curr \ I_all
            Update set of all ids (incl. properties): I_all = I_all ∪ I_curr
        For i = 1 to |I_new|
            Determine airplane identification number from set I_new → id_i
            Create new airplane object with id_i
        For j = 1 to |I_curr|
            Determine airplane identification number from set I_curr → id_j
            Assign new trajectory data to airplane with id_j
            Remove invalid or duplicated entries earlier assigned to airplane
            Perform interpolation (if required)
            Remove data older than some altitude dependent time window
        Update overlays for Google Earth and write them to the web server
        Generate new kmz data file and write it to the web server
    Else
        Update overlays for Google Earth and write them to the web server
        Generate new kmz data file and write it to the web server
            → Only overlays are updated as no new data were received
    Wait until the end of the current time interval t + ΔT is reached, update time: t ← t + ΔT
```

3.3 Data processing

From the pseudo code in Figure 2 we see that there are several key steps required to finally get correct airplane trajectories visible in Google Earth. However, in this section we focus on the handling of missing data, i.e. on data interpolation. Hence, we omit computational details, e.g., the computation of heading, tilt, distances on orthodromes, etc. as they can be found in textbooks on aviation (e.g., Stengel (2004)), on geodesy (e.g., Torse (2001)), or in standard mathematical references like Bronstein et al. (2000).

As mentioned in section 3.1, the time between correctly received subsequent data points, sent by the airplane's ADS-B device, depends mainly on:

- Update rate and type of the message (variables) sent,
- Atmospherical interferences, which can lead to signal distortions, and
- The topographical environment within the transmission space that can lead to dead spots.

Provided that the ADS-B devices, both in the airplane as well as the one at the ground station, work correctly, atmospherical and topological influences are the main sources for missing data points. Errors are mainly caused by the measurement devices, but are not discussed here.

In the following we explain the idea of data interpolation as applied here by means of a concrete example of a typical trajectory, illustrated in Figure 3: The black dots represent data points (resulting from the airplane's measurements) in three-dimensional space and the blue line connecting these points represents the original trajectory. As we can see, there are two gaps, where the time difference is significantly longer than between the other points: the gap between points $\boldsymbol{x}_6$ and $\boldsymbol{x}_7$ as well as the gap between points $\boldsymbol{x}_8$ and $\boldsymbol{x}_9$ (details on notation see below). The overall framework requires a multitude of parameters. However, in Table 2 we list only those relevant for the selected interpolation of missing data and provide some brief explanations.

Table 2 Some specific parameters required for interpolation of missing airplane data.

#	Name	Short description (in brackets: values chosen in this application).
1	Δt_{gap}	Maximum time difference allowed between two successive data points, in seconds. If the difference (i.e. the gap) is larger than Δt_{gap} we perform an interpolation. (5 seconds)
2	Δt_{max}	Maximum time difference between points within interpolation interval, in seconds. (2 seconds)
3	s_{min}	Minimum length of a sequence of data points required before and after a gap to allow for an interpolation. (3)

The trajectory under consideration is a sequence of n data points (measurements) belonging to flight i, written as:

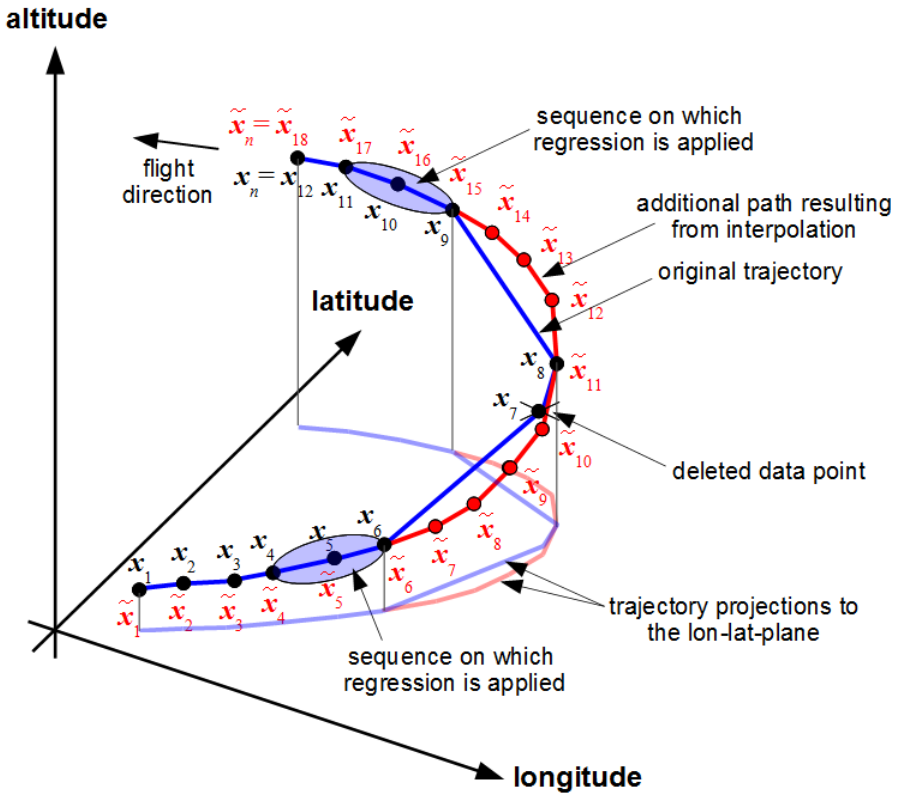
$$\boldsymbol{X}_i = \{\boldsymbol{x}_{i1}, \dots, \boldsymbol{x}_{ij}, \dots, \boldsymbol{x}_{in}\}, \quad (1)$$

where $\boldsymbol{x}_{ij} = [t_{ij}, \lambda_{ij}, \varphi_{ij}, h_{ij}]$, with t_{ij} denoting the time stamp of data point j of flight i, and $\lambda_{ij}, \varphi_{ij}$ and h_{ij}, representing the corresponding longitude, latitude and altitude, respectively.

For the sake of simplicity, we omit index i in the remainder, as we will consider only one flight at a time. Hence, we can write:

$$\boldsymbol{X} = \{\boldsymbol{x}_1, \dots, \boldsymbol{x}_j, \dots, \boldsymbol{x}_n\}, \text{ with } \boldsymbol{x}_j = [t_j, \lambda_j, \varphi_j, h_j]. \quad (2)$$

Figure 3 Graphical representation of (i) the original (artificial) trajectory considered in this example (black dots and blue lines connecting the sequence of dots), with two gaps and (ii) the additional points after performing the interpolation (red and black dots, and red lines connecting the dots).



To detect a gap within sequence $\boldsymbol{X}$, we check for all $j \in \{s_{\min} + 1, \dots, n - s_{\min} + 1\}$ if $\Delta t_j = t_j - t_{j-1} > \Delta t_{\text{gap}}$, where $s_{\min}$ is the minimum number of subsequent points without gap, required to determine the gradient by robust linear regression.

For each gap found, we capture the starting point by $j' = j - 1$, and determine the number of additional points required (in the gap) after point j' as follows:

$$m_{j'} = \left\lfloor \frac{t_j - t_{j-1}}{\Delta t_{\max}} \right\rfloor, \tag{3}$$

where $\Delta t_{\max}$ is the maximum time span allowed between points inserted in the gap and $\lfloor a \rfloor$ returns the largest integer $\leq a$. Before performing the interpolation, we need to increase the index of all data points $j' < j \leq n$, i.e. $j \leftarrow j + m_{j'}$.

Next we compute the additional time stamps within the gap according to:

$$t_k = t_{j'} + (k - j') \cdot \Delta t'_j, \tag{4}$$

where $\Delta t'_j = (t_j - t_{j-1}) / (m_{j'} + 1)$ is the actual time interval between the additional points and k is the corresponding index defined in the range $k = j' + 1, \ldots, j' + m_{j'}$.

As we can see from Figure 3, we removed point $\boldsymbol{x}_7$ before performing the interpolation. This is due to the requirement defined, that subsequent gaps need a minimum of $s_{\min}$ (in our case = 3) valid points in between. Thus, if the number of points s is smaller than $s_{\min}$, we remove all points except point $\lceil (s+1)/2 \rceil$, where $\lceil a \rceil$ returns the smallest integer $\geq a$. As here $s = 2$ holds, we removed point $\boldsymbol{x}_7$ but kept point $\boldsymbol{x}_8$.

After testing different interpolation methods, we decided to use cubic splines (individually for longitude, latitude and altitude), as it outperformed other approaches. A good introduction to splines can be found for example in de Boer (1978).

A useful property of (cubic) splines is, that the they guarantee a smooth transition between the points adjacent to the interpolated points. However, longitude, latitude, altitude, and other variables determined by measurement devices on board of the airplanes are always distorted to some extent due to various effects (inaccuracy of devices, bias due to drift, thermal effects, etc.). Hence, determining the true gradients $\mathrm{d}\lambda/\mathrm{d}t$, $\mathrm{d}\varphi/\mathrm{d}t$ or $\mathrm{d}h/\mathrm{d}t$ by simple approximations from subsequent measurements, e.g. $\mathrm{d}\lambda_j/\mathrm{d}t \approx (\lambda_j - \lambda_{j-1})/(t_j - t_{j-1})$ at point j, leads often to large errors and, as a consequence, to non-realistic trajectories. To avoid this, we perform a robust linear regression[5] leading to approximations of the gradients (using the $s_{\min}$ values), both before and after the gap.

[5] Fundamentals on robust regression can be found for example in Huber (1981), Holland and Welsch (1977) or Street et al. (1988).

Let us assume, $s_{\min}$ = 3 holds and we want to compute the gradients from the three points right before the gap. The data sequence $\boldsymbol{X}_{j-3:j-1} = \{\boldsymbol{x}_{j-3}, \boldsymbol{x}_{j-2}, \boldsymbol{x}_{j-1}\}$ is available and comprehends all information required.

It is important to note, that the computations were performed independently for each of the three variables. Various tests were conducted and the results have proven, that this a reasonable assumption. However, we will not further elucidate this here.

Based on the three time stamps t_{j-3}, t_{j-2} and t_{j-1} and the associated longitude values λ_{j-3}, λ_{j-2} and λ_{j-1} we perform a robust linear regression (using a bisquare weight function), which leads to $\tilde{\lambda}_{j-3} = \tilde{\lambda}_{j-3}(t_{j-3})$, $\tilde{\lambda}_{j-2} = \tilde{\lambda}_{j-2}(t_{j-2})$ and $\tilde{\lambda}_{j-1} = \tilde{\lambda}_{j-1}(t_{j-1})$. The required gradient then results from $\Delta\tilde{\lambda}_{j-1} = (\tilde{\lambda}_{j-1} - \tilde{\lambda}_{j-2})/(t_{j-1} - t_{j-2})$. Next we adjust λ_{j-1} so that the gradient from point $j-2$ to $j-1$ is identical to the desired $\Delta\tilde{\lambda}_{j-1}$, while λ_{j-3} and λ_{j-2} remain unchanged and finally get $\lambda'_{j-1} = \lambda_{j-2} + \Delta\tilde{\lambda}_{j-1} \cdot (t_{j-1} - t_{j-2})$.

This procedure is identical for longitude, latitude and altitude, leading to λ'_{j-1}, φ'_{j-1}, and h'_{j-1}, respectively and thus to $\boldsymbol{x}'_{j-1} = [t_{j-1}, \lambda'_{j-1}, \varphi'_{j-1}, h'_{j-1}]$.

The procedure for determining the gradients from the three points right after the gap, i.e. from $\boldsymbol{X}_{j+m_{j'}:j+m_{j'}+2}$, is very similar to the one described above and will thus not be explained in detail here. From these computations we get $\boldsymbol{x}'_{j+m_{j'}} = [t_{j+m_{j'}}, \lambda'_{j+m_{j'}}, \varphi'_{j+m_{j'}}, h'_{j+m_{j'}}]$, containing the adjusted values at point $j' + m_{j'}$, where $m_{j'}$ is again the number of interpolation points inserted into the gap before original point j (increased later by $m_{j'}$; see equation 3).

Given the additional time stamps t_k (with $k = j' + 1, \ldots, j' + m_{j'}$) from equation (4), we can now perform the interpolation.

As discussed, we need to take special care on points between two gaps, if their number $s < s_{\min}$. In case that $s \geq s_{\min}$ holds, we perform the regression to determine the gradients as described above. In the former case, we remove all entries except point $\lceil (s+1)/2 \rceil$, which we call an intermediate point and hence consider the range as one gap. If available, this value needs to be included into the procedure. However, as applying the spline function itself is straightforward, we do not explain the mathematics behind it, but rather show its use for the example trajectory in Figure 3. From there we see, that the original trajectory consists of two gaps, with one valid intermediate point ($\boldsymbol{x}_8$). Hence, the available data points used by the spline function are $\boldsymbol{x}_5, \boldsymbol{x}'_6, \boldsymbol{x}_8, \boldsymbol{x}'_9, \boldsymbol{x}_{10}$, where $\boldsymbol{x}_5$ and the adjusted value $\boldsymbol{x}'_6$ determine the transition (gradients) before the gap, $\boldsymbol{x}_8$ denotes the intermediate point, and the adjusted value $\boldsymbol{x}'_9$ and $\boldsymbol{x}_{10}$ determine the transition (gradients) after the gap.

By applying the spline function to these data points, using the time stamps t_k already computed, and adding the unchanged values before and after the adjusted values, we finally get vector $\widetilde{\boldsymbol{X}} = \{\widetilde{\boldsymbol{x}}_1, .., \widetilde{\boldsymbol{x}}_{18}\}$ (in accordance with Figure 3), where

- $\widetilde{\boldsymbol{x}}_p = \boldsymbol{x}_p$ for $p \in \{1, \dots, 5\}$, are unchanged values before adjusted value $\boldsymbol{x}'_6$,
- $\widetilde{\boldsymbol{x}}_6 = \boldsymbol{x}'_6$ and $\widetilde{\boldsymbol{x}}_{15} = \boldsymbol{x}'_9$ allocate the adjusted values at the beginning and at the end, respectively,
- $\widetilde{\boldsymbol{x}}_p$ for $p \in \{7, \dots, 10\}$ and $\widetilde{\boldsymbol{x}}_p$ for $p \in \{12, \dots, 14\}$ represent values resulting from the interpolation,
- $\widetilde{\boldsymbol{x}}_{11} = \boldsymbol{x}_8$ allocates the intermediate point, and finally
- $\widetilde{\boldsymbol{x}}_p = \boldsymbol{x}_p$ for $p \in \{16, \dots, 18\}$, are unchanged values after adjusted value $\boldsymbol{x}'_9$.

Again, the spline function was applied to longitude, latitude and altitude separately, thus we finally write the resulting trajectory as $\widetilde{\boldsymbol{X}} = \{\widetilde{\boldsymbol{x}}_1, \dots, \widetilde{\boldsymbol{x}}_j, \dots, \widetilde{\boldsymbol{x}}_n\}$, with $\widetilde{\boldsymbol{x}}_j = [\tilde{t}_j, \tilde{\lambda}_j, \tilde{\varphi}_j, \tilde{h}_j]$ and $n = 18$.

Some remarks

Applying the interpolation procedure to data, which become available in real-time, requires some decisions regarding delay and interpolation/extrapolation:

- As described above, we only perform an interpolation if a sequence of at least $s_{\min}$ valid data points (without any gap within each sequence) is available before and after the actual gap(s). A big advantage of that procedure is that no data revisions, i.e. changes of trajectories at a later point in time, are required. On the contrary, this requirement, in particular when considering the data after the gap, leads in some cases to a larger delay, as the algorithm needs to wait for $s_{\min}$ data points without any gap. Nonetheless, we weighted the accuracy and reliability from a user's point of view higher than the update speed. In the next point, we will see that incorporating trajectory extrapolation could allow for higher update rates.
- The issue of extrapolating trajectories is closely related to the previous point. So far, we did not implement the extrapolation of trajectories. However, in particular if the gradient of a trajectory in any of the three variables is large (where changes of longitude and latitude are always interrelated), the time intervals between successive messages should be within a reasonable limit. In any case, applying Kalman or related filters (see Grewal and Andrews (2008) or Simon (2006), and references therein) could allow for extrapolation and thus increased update rates.

4. Results

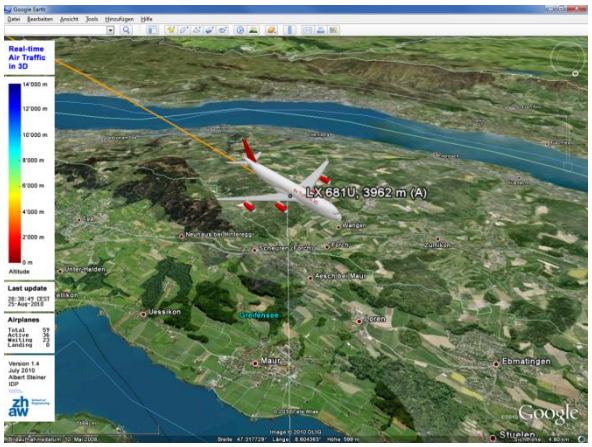
To demonstrate how the application described in the preceding sections works in practice, we present some examples here, including snap-shots as well as the illustration of trajectories captured over a longer time interval. However, as the main goal of the framework is to visualize dynamically changing data, we refer here in particular to the project website, where the required KMZ-file is available for download, and various links with further information (flight plans, movies (some in time lapse mode), etc.) are provided. Nonetheless, in section 4.2 we show how the data extracted can be used for further analyses. This can include, for example, the analysis of flight routes, e.g. during starting and landing, or at different weather/wind conditions.

4.1 Trajectory visualization

In Figure 4 we see some snap-shots (taken directly from the application screen) of different situations (traffic states), e.g. airplanes landing, airplanes in waiting loops, etc. These snap-shots do, of course, not capture the actual dynamics of the air traffic. Thus, to watch the application in real-time, i.e. with a delay of only around 10 seconds, please check out the project website at: http://www.idp.zhaw.ch/airtraffic3d (with various supplementary material).

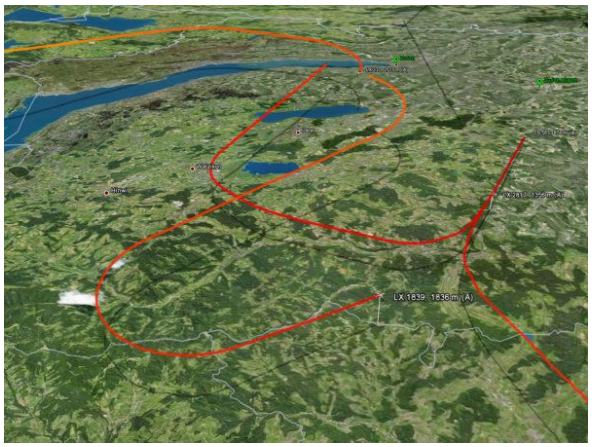
Figure 4 Screen shots of the application for different scenes: (a) close look to the 3D-airplane model used (labelled with flight number, last known altitude in metres, and transmission state (in brackets; A = active, W = waiting, L = landing)), (b) and (c) show airplanes approaching Zurich airport from east (i.e., towards runway 28), (d) and (e) show airplanes in waiting loops. The colours of the trajectories represent the corresponding altitudes, ranging from dark blue at 14'000 metres above sea level to dark red at sea level (0 metres).

(a)

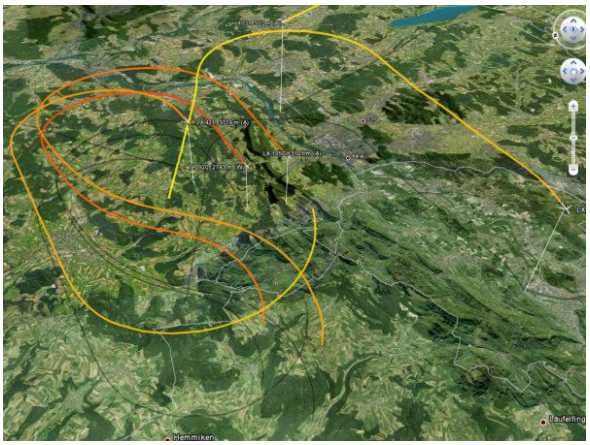


(b)

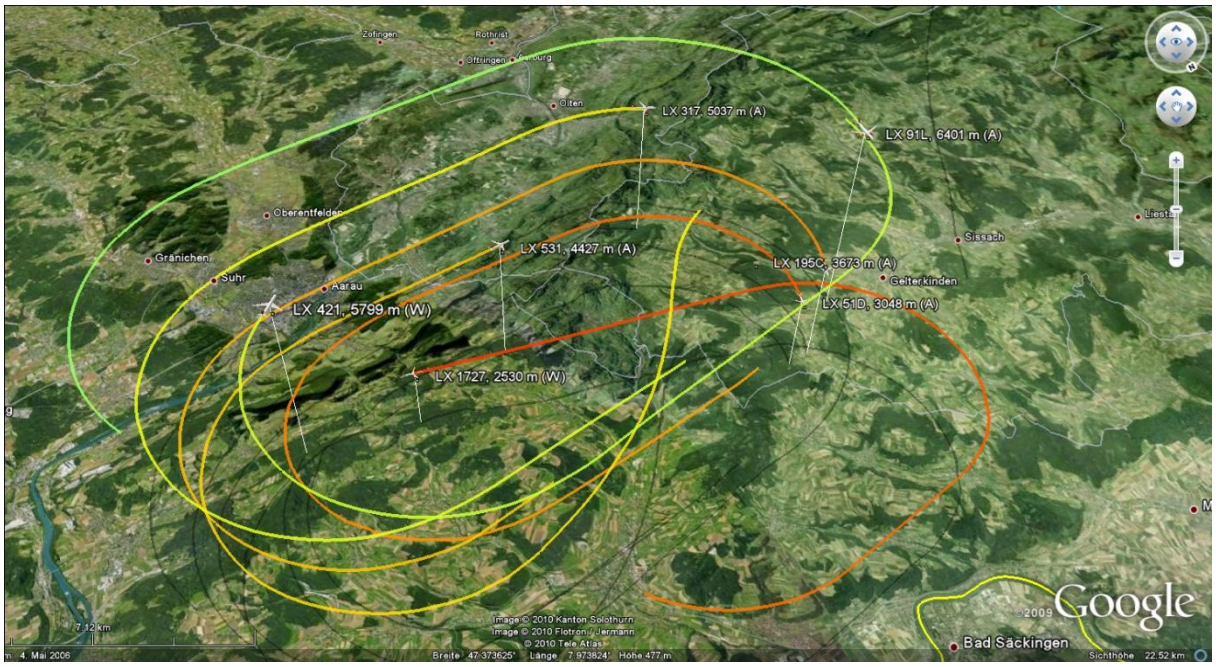
(c)



(d)



(e)

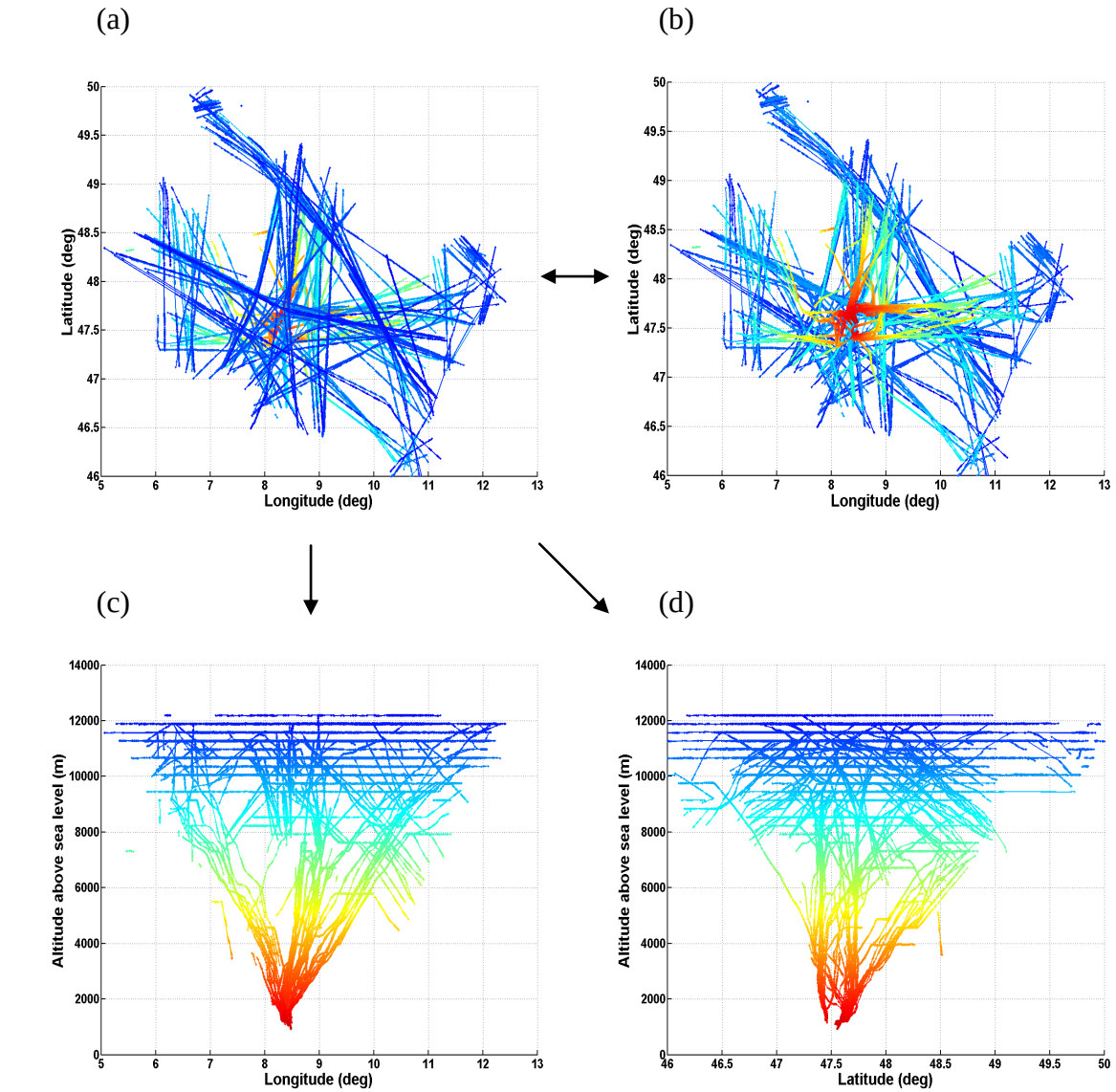


4.2 Further analysis

Although not described in detail in this paper, the application can also be run in 'data collection' mode. This means, that the airplane data are collected for later offline analysis. Such analyses can include starting and landing routes for different weather conditions (wind, rain, etc.) or the analysis of the accuracy of different systems determining the airplanes' position (longitude, latitude, altitude) and speed, to name only a few.

An example of such data, i.e. airplane trajectories (without interpolation of missing data) is presented in Figure 5. This shall demonstrate how such data typically look like.

Figure 5 Airplane trajectories computed from original data points (i.e., without interpolations), collected between 11:00 and 12:40 on February 26, 2010. The colour of the trajectories represent the corresponding altitude, ranging from dark blue at 14'000 metres above sea level to dark red at sea level (0 metres). The area with red coloured trajectories near ground is around Zurich airport. (a) View of the longitude-latitude-plane, (b) View of the longitude-latitude-plane as in (a), but with reversed altitude axis, which allows for better visibility of the trajectories near ground, (c) View of the longitude-altitude-plane, and (d) View of the latitude-altitude-plane. From (a) the most frequently used overflight routes at altitudes ≥ around 9'000 metres become obvious, whereas in (b) the trajectories from departures and landings are clearly visible (yellow to red colours). From (c) and (d) the vertical separation minimum (VSM) distance of 300 metres (see ICAO (1996) and Mensen (2004) for details) is visible at altitudes in the range between around 7'000 and 12'400 metres. Taking a closer look, we see various gaps within the trajectories, i.e. long time intervals between subsequent points, due to missing data.



5. Conclusions and outlook

We have presented a framework to visualize dynamically changing information in Google Earth. The framework was applied to real-time data received from airplanes over parts of Switzerland and Central Europe and results were presented. The application together with the accompanying project website is online since March 2010, with more than 50'000 visits between May and August 2010.

Based on the current development status, we see several ways to extend the air traffic application. Some of them are (whereat the list is not exhaustive):

- Including data extrapolation by using appropriate state estimation approaches,
- Including additional air traffic data sources,
- Adding aviation specific information on weather (wind, fog, etc.) and climate,
- Combining the application with other existing applications (e.g. view of air space sectors)
- Increasing performance (if required), and
- Further reducing the model and file sizes.

Currently, one of the limiting factors for large scale applications is the bandwidth of the internet connections. This holds in particular if various three-dimensional models and/or large data sets need to be downloaded at high update rates (e.g., every second). However, (i) the expected increasing speed of the internet connections, (i) new developments in data compression (images, data files, etc.), (iii) the continuous extension of the KML standard due to additional functions, and (iv) general technological developments, will all contribute to further facilitate the number of new applications and to enhance the speed and capabilities of these applications.

Besides this application, we are extending the framework by adding new functionalities. Furthermore we are working on applications to visualize road traffic and dynamics in pedestrian facilities, with displaying information dynamically, i.e. in real-time, as well as statically.

As already mentioned, there is a vast number of fascinating applications using data, which are changing over time and space, both from transport and mobility, but also from many other fields (e.g., ecology, economy, health (e.g. epidemiology), politics, travelling, etc.). As the list of applications would go beyond the scope of this paper, we refer to the numerous websites on these topics, e.g. Google Earth Blog, Google Earth Hacks and the examples on the official Google Earth website (all links can be found in the reference section at the end of this document).

Acknowledgements

I would like to thank Marcel Rupf (ZSN, Centre for Signal Processing and Communications Engineering) and Karl Rege (InIT, Institute of Applied Information Technology), both with the School of Engineering at ZHAW, for initializing and designing the "Radar" project and for providing online access to air traffic data. Furthermore, I want to thank Daniel Kramarz (InIT) for helping with access to the web service, for fruitful discussions, and for providing the thesis Kramarz and Loeber (2007). I also want to thank my colleagues at IDP for valuable discussions and suggestions during the development of the framework. Finally, I am grateful to Remo Maurer, head of IT, School of Engineering at ZHAW, for invaluable support with setting up and running the Linux server hosting the application.

References

de Boor, C. (1978) *A Practical Guide to Splines*, Springer, New York, USA.

Brimicombe, A. and Li, C. (2009) *Location-Based Services and Geo-Information Engineering*, , John Wiley & Sons Ltd., Chichester, UK.

Bronstein, I.N, Semendjajew, K.A., Musiol, G. and Mühlig, H. (2000) *Taschenbuch der Mathematik*. Verlag Harri Deutsch.

EUROCONTROL (2010) The CASCADE programme (http://www.eurocontrol.int/cascade/public/subsite_homepage/homepage.html), Accessed August 3, 2010.

EUROCONTROL (undated) ADS-B for Dummies – 1090 MHz Extended Squitter (available at: http://www.ans.dhmi.gov.tr/TR/Sistem/Dok/ADS-B%20for%20Dummies-1090ES%20v04.pdf), Accessed August 3, 2010.

EUROCONTROL (2006) CASCADE programme: 1090 MHz Capacity Study – Final Report, edition 2.7, July 2006.

Garrity, R. (2006) Automatic Dependent Surveillance – Broadcast. ICAO CNS/ATM Technology Seminar, Lima, Peru, 26.29 September 2006.

Google Earth (http://earth.google.co.uk/intl/en_uk/), Accessed July 10, 2010.

Google Earth Blog. Website with various examples of applications, most of them available for download (http://www.googleearthblog.com), Accessed August 3, 2010.

Google Earth Hacks website (http://www.gearthhacks.com/), Accessed July 10, 2010.

Grewal, M.S. and Andrews, A.P. (2008) *Kalman Filtering: Theory and Practice Using MATLAB*, 3rd edition, John Wiley & Sons, Inc., Hoboken, NJ, USA.

Holland, P.W. and Welsch, R.E. (1977) Robust Regression Using Iteratively Reweighted Least-Squares, *Communications in Statistics: Theory and Methods*, **A6**, 813-827.

Honorjeff, R., McKelvey, F.X., Sproule, W.J. and Young, S.B. (2010) Planning & Design of Airports, 5th edition, McGraw-Hill.

Huber, P.J. and Ronchetti, E.M. (2009) *Robust Statistics*, 2nd edition, John Wiley & Sons, Inc., Hoboken, NJ, USA.

ICAO (1996) *Procedures for Air Navigation Services – Rules of the Air and Air Traffic Services*. Doc 4444-RAC/501, 13th edition, ICAO.

Libhomeradar. (http://www.libhomeradar.org/), Accessed July 10, 2010.

MATLAB. Product website of The MathWorks, Inc. (http://www.mathworks.com/), Accessed July 10, 2010.

KML Reference (http://code.google.com/intl/en/apis/kml/documentation/kmlreference.html), Accessed July 10, 2010.

Kramarz, D. and Loeber, A. (2007) Visualisierung von Transponder-Daten mittels Mashup, Diploma thesis, ZHAW, 5. Oktober 2007.

Küpper, A. (2005) *Location-based Services – Fundamentals and Operation*, John Wiley & Sons Ltd., Chichester, UK.

Mensen, H. (2004) *Moderne Flugsicherung*, Springer-Verlag, Berlin Heidelberg.

Microsoft Bing Maps. (http://www.bing.com/maps/), Accessed July 10, 2010.

Radar project (http://radar.zhaw.ch/), Accessed August 3, 2010.

Real-time Air Traffic in 3D. Project website (http://www.idp.zhaw.ch/airtraffic3d), last update April 29, 2010.

Simon, D. (2006) *Optimal State Estimation*, John Wiley & Sons, Inc., Hoboken, NJ, USA.

Stengel, R.F. (2004) *Flight Dynamics*, Princeton University Press.

Street, J.O., Carroll, R.J. and Ruppert, D. (1988) A Note on Computing Robust Regression Estimates via Iteratively Reweighted Least Squares, *The American Statistician*, **42**, 152-154.

Torse, W. (2001) *Geodesy*, 3rd edition, de Gruyter, Berlin and New York.

Yahoo! Maps website (http://maps.yahoo.com/), Accessed August 3, 2010.