BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF INFORMATION TECHNOLOGY
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

AUTOMATIC EVENT IDENTIFICATION FROM FLIGHT DATA RECORDS

AUTOMATICKÁ IDENTIFIKACE UDÁLOSTÍ ZE ZÁZNAMŮ LETOVÝCH DAT

BACHELOR'S THESIS
BAKALÁŘSKÁ PRÁCE

AUTHOR **ADRIÁN KIRÁLY**
AUTOR PRÁCE

SUPERVISOR **doc. Ing. PETER CHUDÝ, Ph.D., MBA**
VEDOUCÍ PRÁCE

BRNO 2018

Brno University of Technology - Faculty of Information Technology

Department of Computer Graphics and Multimedia Academic year 2017/2018

Bachelor's Thesis Specification

For: **Király Adrián**
Branch of study: Information Technology
Title: **Automatic Event Identification from Flight Data Records**
Category: Signal Processing

Instructions for project work:

1. Get familiar with the aircraft equations of motion.
2. Research state-of-the-art flight data recording technologies.
3. Design and implement an application for flight event identification.
4. Evaluate achieved results and discuss potential further improvements.

Basic references:

- According to supervisor's recommendations.

Requirements for the first semester:
Tasks No. 1, 2 and partially task No. 3.

Detailed formal specifications can be found at http://www.fit.vutbr.cz/info/szz/

The Bachelor's Thesis must define its purpose, describe a current state of the art, introduce the theoretical and technical background relevant to the problems solved, and specify what parts have been used from earlier projects or have been taken over from other sources.

Each student will hand-in printed as well as electronic versions of the technical report, an electronic version of the complete program documentation, program source files, and a functional hardware prototype sample if desired. The information in electronic form will be stored on a standard non-rewritable medium (CD-R, DVD-R, etc.) in formats common at the FIT. In order to allow regular handling, the medium will be securely attached to the printed report.

Supervisor: **Chudý Peter, doc. Ing., Ph.D. MBA**, DCGM FIT BUT
Beginning of work: November 1, 2017
Date of delivery: May 16, 2018

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav počítačové grafiky a multimédií
612 66 Brno, Božetěchova 2

L.S.

Jan Černocký
Associate Professor and Head of Department

Abstract

This bachelor's thesis is focused on automatic processing of flight data records, namely on identification of flight phases and turn maneuvers. A set of multiple methods for identification of these events has been developed and evaluated on a publicly available dataset containing data collected from a single type of regional jet. In order to demonstrate these methods, an interactive 3D application has been developed that visualizes both flight data and detected events in an intuitive way.

Abstrakt

Táto bakalárska práca je zameraná na automatické spracovanie záznamov letových dát, konkrétne na identifikáciu fáz letu a zátačiek. S pomocou verejne dostupného datasetu obsahujúceho záznamy zozbierané z jedného druhu regionálneho prúdového lietadla bola vyvinutá sada niekoľkých metód pre identifikáciu týchto udalostí. Pre demonštráciu týchto metód bola vyvinutá interaktívna 3D aplikácia, ktorá intuitívne vizualizuje letové dáta a identifikované udalosti.

Keywords

flight data processing, automatic event identification, phase of flight, aviation, flight data visualization, Unity 3D

Kľúčové slová

spracovanie letových dát, automatická identifikácia udalostí, fáza letu, letectvo, vizualizácia letových dát, Unity 3D

Reference

KIRÁLY, Adrián. *Automatic Event Identification from Flight Data Records.* Brno, 2018. Bachelor's thesis. Brno University of Technology, Faculty of Information Technology. Supervisor doc. Ing. Peter Chudý, Ph.D., MBA

Rozšírený abstrakt

Táto bakalárska práca sa zaoberá problematikou automatického spracovania záznamov letových dát, konkrétne na identifikáciu fáz letu a zátačiek. Fáza letu je bežne používaná ako jednoduchý spôsob špecifikácie konkrétnej časti letu. V spojení s informáciami o zátačkach alebo iných udalostiach je napríklad možné automaticky extrahovať segmenty letu obsahujúce želané udalosti pre ďalšie manuálne spracovanie.

Vzhľadom na to, že dáta získané zo senzorov často obsahujú rôzne neželané artefakty ktoré komplikujú spracovanie dát, je nutné dáta pred ďalším použitím vhodne predspracovať. Jedným z významných artefaktov je napríklad viditeľné „krokovanie" v GPS dráhe lietadla, ktoré bolo odstránené s pomocou algoritmu založeného na konvolúcii.

Pre identifikáciu spomínaných udalostí bola vyvinutá sada metód, ktoré na základe vybraných parametrov letových dát identifikujú jednotlivé udalosti v niekoľkých krokoch, pričom každý krok využíva informácie získané v prechádzajúcom kroku pre zjednodušenie alebo skvalitnenie identifikácie. V prvom kroku sa identifikuje vzlet a pristátie, na základe ktorých je možné zvyšok letu rozdeliť na segmenty „na zemi" a „vo vzduchu". S využitím tohto rozdelenia sú následne identifikované ďalšie fázy, napríklad vyrovnaný let, stúpanie a klesanie. Podobným spôsobom je vyriešená aj identifikácia zátačiek, ktorá využíva zmeny v kurze lietadla.

Nakoľko lietadlá majú šesť stupňov voľnosti, ich pohyb je mnohokrát pomerne komplikovaný. I keď je pre vizualizáciu jednotlivých parametrov stále možné využiť aj tradičné dvojdimenzionálne grafy, takáto vizualizácia nie je vzhľadom na povahu pohybu lietadla optimálna a jej interpretácia si vyžaduje znateľne viac úsilia. Táto skutočnosť viedla k rozhodnutiu implementovať interaktívnu 3D vizualizáciu, ktorá v ľahko zrozumiteľnom formáte zobrazuje letové dáta aj identifikované udalosti, čím je možné vykonať jednoduchú vizuálnu kontrolu identifikácie a skúmať letové dáta.

Samotná výsledná aplikácia obsahuje implementácie spomínaných metód pre predspracovanie dát, detekciu udalostí a vizualizáciu. Pre jej vývoj bolo využité herné jadro Unity, ktoré bolo pre tento účel vhodnou platformou vzhľadom na kvalitnú podporu práce s 3D grafikou, animáciami, osvetlením, tvorbu grafického používateľského rozhrania a ďalšiu funkcionalitu.

Kľúčovou časťou vizualizácie je terén, nakoľko tvorí väčšinu viditeľného prostredia okolo lietadla a slúži ako vizuálna referencia aktuálnej polohy. V aplikácií sú implementované dve rôzne riešenia pre zobrazovanie terénu — riešenie využívajúce výškové dáta a satelitné snímky zo služby Mapbox, a riešenie využívajúce voľne dostupné výškové dáta získané projektom Shuttle Radar Topography Mission (SRTM) v spojení s procedurálnymi textúrami. Vzhľadom na to, že letové dáta obsahujú aj časové informácie, je možné na základe času, dátumu a aktuálnej pozície lietadla vypočítať polohu Slnka v scéne a vhodne nastaviť osvetlenie vizualizácie. To umožnuje ľahko vizuálne určiť približný lokálny čas v aktuálnej pozícii bez potreby znalosti časových pásiem.

Najvýznamnejším prvkom scény je samozrejme model lietadla, ktorý slúži pre vizualizáciu väčšiny letových dát. V prvom rade zobrazuje pozíciu lietadla vo svete a jeho rotáciu okolo ťažiska. Ďalšími významými parametrami sú uhly hlavných riadiacich plôch, vďaka čomu je možné sledovať ich vplyv na pohyb lietadla. Vizualizované sú taktiež rýchlosti jednotlivých motorov a stav podvozku.

Vďaka tomu, že dáta použité počas vývoja a evaluácie metód identifikácie obsahujú aj parameter označujúci fázy letu určené ACMS systémom, overiť úspešnosť identifikácie fáz bolo možné priamočiarym porovnaním s týmto parametrom. Počas testov nad dátami

z 3 lietadiel (približne 600 letov na lietadlo) sa priemerná úspešnosť po vylúčení súborov s poškodenými dátami pohybovala na úrovni 92 %.

Bohužiaľ, pre overenie úspešnosti identifikácie zátačiek nie je v dátach dostupný žiadny podobný parameter. Z tohto dôvodu nie je možné určiť úspešnosť a množstvo neidentifikovaných manévrov. Počas manuálnej vizuálnej verifikácie pomocou implementovanej aplikácie bolo ale zistené, že skoro všetky nájdené manévre boli identifikované správne.

Automatic Event Identification from Flight Data Records

Declaration

Hereby I declare that this bachelor's thesis was prepared as an original author's work under the supervision of doc. Ing. Peter Chudý, Ph.D., MBA. All the relevant information sources, which were used during preparation of this thesis, are properly cited and included in the list of references.

.....
Adrián Király
May 15, 2018

Acknowledgements

I would like to thank my supervisor doc. Ing. Peter Chudý, Ph.D., MBA for his excellent guidance and invaluable feedback and advice during my work on this bachelor's thesis.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

While most people are familiar with the use of flight data records during accident investigation, this has not been the only use of these data for a quite long time. In fact, quite the opposite is true and these data have proven to be extremely useful during normal aircraft operations. By analyzing these data, it is possible to determine extent of required maintenance, anticipate failures of components before they happen, optimize aircraft operations and more, ultimately making air travel even safer, more reliable, and available.

However, given the large number of flights performed every day and amount of collected data, manual analysis of each flight has no longer been an option for a long time. Fortunately, thanks to the rising availability of computing power and advances in data processing, automated processing of flight data has quickly become a standard industry practice.

The focus of this work is on detection of flight phases and turn maneuvers. Flight phases are commonly used as a simple way to specify a part of flight. Combined with turns or other events, they can be, for example, used for automatic extraction of segments containing the event of interest for further manual analysis. This is particularly useful when working with large number of data.

Development of methods for detection of this events required constant visualization of both flight data and results of detection. As the aircraft movement is inherently more complex when compared to most of other modes of transport, interpretation of flight data is also often more difficult. This has led to development of an interactive 3D visualization which makes it possible to easily interpret flight data and compare them with detected events even from a short glance.

The beginning of this document is dedicated to theoretical basics, with chapter 2 being focused on explanations of aircraft coordinate systems and equations of motion, providing a necessary foundation for any work with flight data. Chapter 3 describes technologies and systems involved in collection of flight data on aircraft, starting with sensors, data buses used (not only) for transfer of measured data on aircraft, and ending with an overview of systems designed to safely store the data. This chapter also contains a basic overview of a publicly available dataset of flight data used in this bachelor's thesis.

Chapter 4 first deals with preprocessing of the flight data before use, then describes details about the detected events, and methods how they are detected in this work. Finally, chapter 5 combines knowledge from all previous chapters into the final product of this bachelor's thesis—an interactive 3D application for automatic detection of flight events. The evaluation of implemented methods and disussion of possible improvements then follows in chapter 6.

Chapter 2

Aircraft Coordinate Systems and Equations of Motion

Before doing any real work with the flight data itself, it is necessary to establish a solid theoretical base that describes aircraft position and motion. Knowing these basics is not only required during their creation, whether it is by direct measurements on a real flight or using a flight simulator, but also provides deeper understanding while interpreting recorded flight data.

This chapter will start off with descriptions of coordinate systems that can be used to describe position of an aircraft in relation to various frames. As it is often necessary to work with multiple coordinate systems, the focus will then shift to transformations between them. Last part of the chapter will focus on equations of motions—a set of equations that are the core of aircraft movement.

2.1 Coordinate Systems and Notation

In order to describe the basic relations and models of aircraft motion in space, it is first necessary to define the various frames and coordinate systems that can describe the position of the aircraft. These descriptions can be relatively complex, since aircraft have six degrees of freedom. While there are many different frames and coordinate systems commonly in use, this section will focus only on some of the most used and well-known ones.

2.1.1 Earth-Centered, Earth-Fixed Frame

The Earth-Centered, Earth-Fixed frame (ECEF) (see fig. 2.1) is a global frame that allows to locate any given point on the surface of the Earth using a fixed set of coordinates. This is possible because the axes of the ECEF rotate with the Earth around its spin axis [15].

The origin of the frame is located at the center of the Earth and the axes are defined as follows [15]:

- the z_e axis points towards the true North Pole,
- the x_e axis is perpendicular to the z_e axis and intersects the prime meridian,
- the right-hand coordinate system is completed with the y_e axis orthogonal to the axes z_e and x_e.

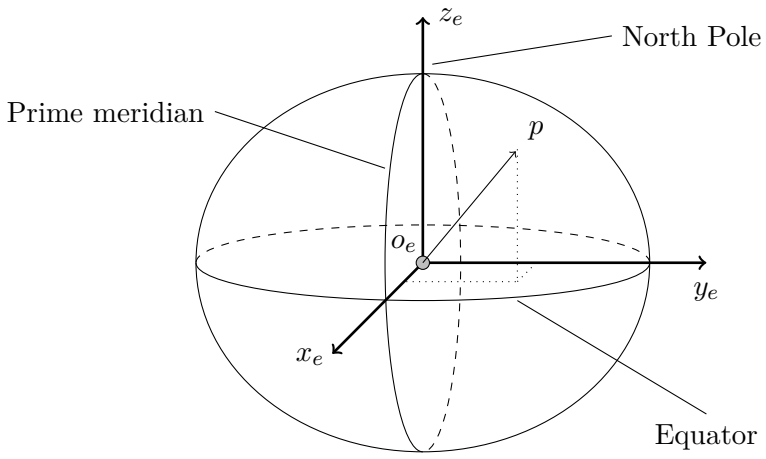


Figure 2.1: The Earth-Centered, Earth-Fixed frame is defined by the origin o_e and the x_e, y_e, and z_e axes.

While it is possible to determine a position in ECEF using the x_e, y_e, and z_e axes as shown in fig. 2.1, it is also common to use a polar-type Latitude, Longitude, Altitude (LLA) coordinates, denoted by Φ, λ, and h, respectively. This requires use of a reference ellipsoid, such as the one defined by World Geodetic System 1984 (WGS-84).

Note that the latitude angle Φ of a given point is defined with reference to the local normal and generally does not cross the origin o_e. Correspondingly, the altitude h of the point is measured along the local normal as the distance above (or below) the reference ellipsoid, *not* the distance between the Earth's center and the point itself.

2.1.2 Local Geographic Frame

In many scenarios a simpler definition of Earth axes is preferable. In such cases, a local geographic frame can be used. This frame assumes that the aircraft flies above a flat ground, tangent to the Earth's surface at a certain, specified point. Such assumption is perfectly adequate in many scenarios concerning only a short term motion or with total extent smaller than some tens of kilometers [10, 15].

The coordinate system of this frame, the North-East-Down (NED) coordinate system (see fig. 2.2), is defined as follows: The origin o_0 is an arbitrary reference point on or above the surface of the Earth. The x_0, y_0, and z_0 axes point, as the name of this frame suggests, to the north, east, and vertically down, respectively. The x_0 and y_0 axes together define a local horizontal plane tangential to the surface of the Earth [10].

An alternative choice of cartesian axes results in a similar, East-North-Up (ENU) coordinate system, also commonly used with the local geographic frame. The origin of the local geographic frame may also be placed at the center of gravity of a flying aircraft. This may be especially useful in combination with the NED coordinate system, which is then sometimes referred to as *Vehicle-Carried North-East-Down coordinate system* [6].

2.1.3 Body Fixed Frame

The Body Fixed Frame (BFF) (see fig. 2.3) is a frame described by a set of axes fixed to the aircraft frame. Conventionally, the origin of the coordinate system is located at the center of gravity (CG) of the aircraft. The x_b axis then points forward along the nose

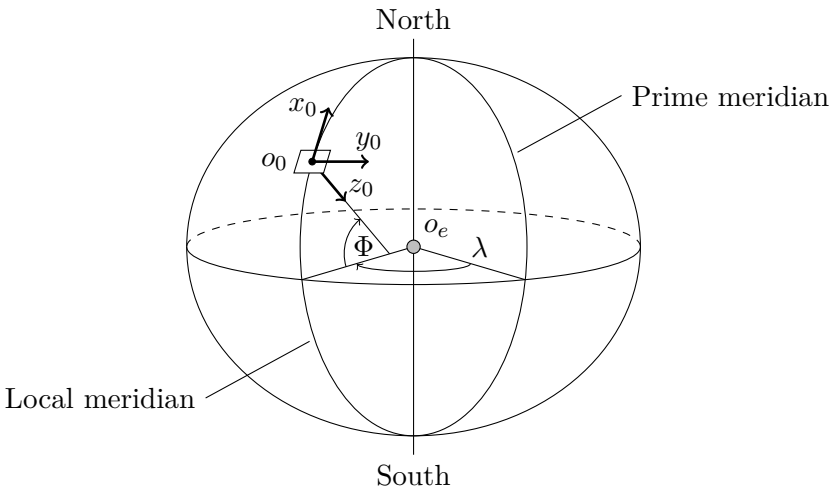


Figure 2.2: The Local Geographic Frame and the North-East-Down (NED) coordinate system. Also pictured is the relation between the local geographic frame and the Earth-Centered, Earth-Fixed frame through the latitude Φ and longitude λ.

of the aircraft, the y_b axis points to the starboard and the remaining z_b axis is directed downwards. The plane defined by the x_b and z_b axes also usually coincides with the plane of the symmetry of the aircraft [10].

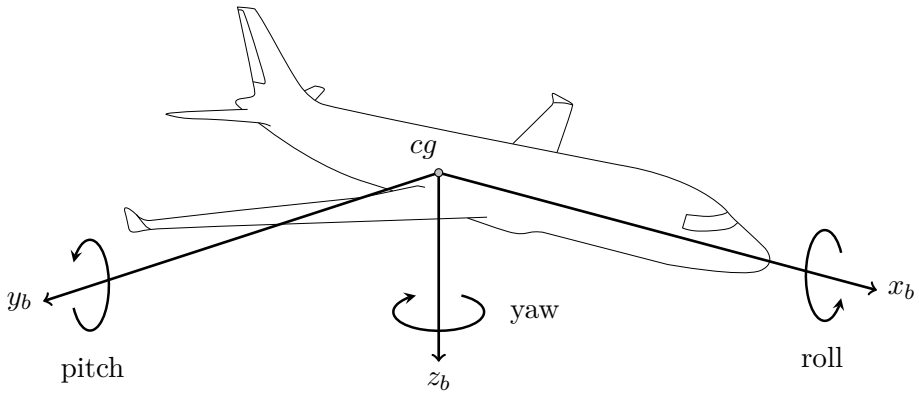


Figure 2.3: The Body Fixed Frame, defined by origin in the CG and the fixed axes x_b, y_b, and z_b.

2.1.4 Stability and Wind Axes

Both stability and wind coordinate systems (see fig. 2.4) are reference frames with origin in the CG. However, unlike the Body Fixed Frame, their axes are not fixed to the airframe, but related to the current flight path vector V.

The stability axis y_s is always aligned with the y_b body axis, and the x_s axis is parallel with the projection of the flight path vector V into the plane formed by axes x_b and y_b. These two axes, x_s and y_s, together form the plane of motion of the aircraft. The stability

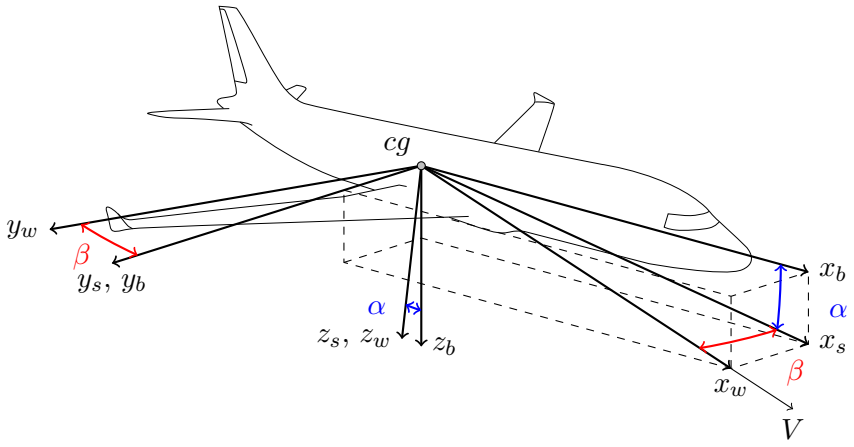


Figure 2.4: Relations between the flight path vector V, Body Fixed Frame, Stability, and Wind axes.

axes are related to BFF through a single rotation about the y_s axis. The angle α of this rotation is commonly known as the angle of attack (AoA) [1].

In wind coordinate system, the x_w axis is aligned with the flight path vector V itself instead. Similarly to the stability axes, the plane defined by axes x_w and y_w is the instantaneous plane of motion mentioned earlier. The wind axes are related to the stability axes through a single rotation around the z_s axis. The angle β of this rotation is defined as the angle of sideslip (AoS) [1].

2.2 Transformations Between Coordinate Systems

As each of the frames and coordinate systems described in section 2.1 is better suited for set of certain tasks, it is often more convenient or even necessary to use multiple coordinate systems and to perform the appropriate transformations between them. This section will describe some of the methods that can be used for these transformations.

2.2.1 Local Geographic Frame to Body Fixed Frame

Let's assume that both local geographic frame and the Body Fixed Frame are originated at the same point o. Then, we can describe the transformation to BFF as a series of three consecutive rotations through the angles ψ, θ, ϕ, called the *Euler angles*. It's important to note that these rotations are not commutative and must be performed in a specified order to achieve correct aircraft orientation [1].

With reference to fig. 2.5, $ox_0y_0z_0$ are the local geographic frame reference axes and $ox_3y_3z_3$ are the axes of BFF, the orientation of aircraft with respect to local frame is established by considering the following rotations required to bring $ox_0y_0z_0$ into coincidence with $ox_3y_3z_3$ [1]:

1. Rotation about the oz_0 axis through the heading angle ψ, defined as the angle between north and the horizontal projection of the vehicle roll axis [13]. This results in an

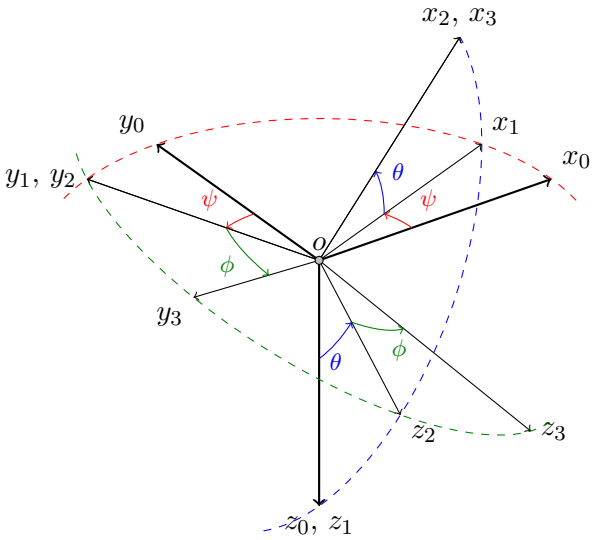


Figure 2.5: The Euler angles.

intermediate coordinate system $ox_1y_1z_1$

$$\boldsymbol{R_3}\left(\psi\right) = \begin{bmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.1}$$

2. Rotation about the oy_1 axis through the pitch angle ψ, resulting in an intermediate coordinate system $ox_2y_2z_2$

$$\boldsymbol{R_2}\left(\theta\right) = \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix} \tag{2.2}$$

3. Finally, rotation about the ox_2 axis through the roll angle ϕ

$$\boldsymbol{R_1}\left(\phi\right) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & -\sin\phi & \cos\phi \end{bmatrix} \tag{2.3}$$

The final transformation matrix can be obtained by multiplying the rotation matrices in eqs. (2.1) to (2.3), therefore

$$\begin{aligned} \boldsymbol{T_{BL}} &= \boldsymbol{R_1}\left(\phi\right)\boldsymbol{R_2}\left(\theta\right)\boldsymbol{R_3}\left(\psi\right) \\ &= \begin{bmatrix} \cos\theta\cos\psi & \cos\theta\sin\psi & -\sin\theta \\ \sin\phi\sin\theta\cos\psi - \cos\phi\sin\psi & \sin\phi\sin\theta\sin\psi + \cos\phi\cos\psi & \sin\phi\cos\theta \\ \cos\phi\sin\theta\cos\psi + \sin\phi\sin\psi & \cos\phi\sin\theta\sin\psi - \sin\phi\cos\psi & \cos\phi\cos\theta \end{bmatrix} \end{aligned} \tag{2.4}$$

The resulting matrix eq. (2.4) can be then used to transform components x_0, y_0, z_0 of a vector in local geographic frame to components x_3, y_3, z_3 of a vector in the Body Fixed Frame using the following relation

$$\begin{bmatrix} x_3 \\ y_3 \\ z_3 \end{bmatrix} = \boldsymbol{T_{BL}} \begin{bmatrix} x_0 \\ y_0 \\ z_0 \end{bmatrix} \tag{2.5}$$

By inverting the transformation matrix $\boldsymbol{T_{BL}}$, the reverse transformation is obtained

$$\begin{bmatrix} x_0 \\ y_0 \\ z_0 \end{bmatrix} = \boldsymbol{T_{LB}} \begin{bmatrix} x_3 \\ y_3 \\ z_3 \end{bmatrix} = \boldsymbol{T_{BL}}^{-1} \begin{bmatrix} x_3 \\ y_3 \\ z_3 \end{bmatrix} \tag{2.6}$$

2.2.2 Body Fixed Frame to Stability and Wind Axes

Transformation from the Body Fixed Frame to stability and wind axes is based on same principles as transformations in previous section. Whereas the BFF is related to the local geographic frame through angles ψ, θ, ϕ, the wind axes are related to the BFF through angles α and β, as mentioned earlier in the section 2.1.4.

First, let's consider transformation from stability axes to BFF. As we established earlier in section 2.1.4, the BFF and stability axes are related through a single rotation about the oy_b axis by an angle α, thus, the following transformation matrix applies [1]

$$\boldsymbol{T_{BS}} = \boldsymbol{R_y}(\alpha) = \begin{bmatrix} \cos\alpha & 0 & -\sin\alpha \\ 0 & 1 & 0 \\ \sin\alpha & 0 & \cos\alpha \end{bmatrix} \tag{2.7}$$

Then, a single rotation about the oz_s axis by an angle β performs the transformation from stability to wind coordinates. This operation can be described by the following rotation matrix [1]

$$\boldsymbol{T_{WS}} = \boldsymbol{R_z}(\beta) = \begin{bmatrix} \cos\beta & \sin\beta & 0 \\ -\sin\beta & \cos\beta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.8}$$

By appropriately combining the matrices in eqs. (2.7) and (2.8), we obtain the following matrix for transformation from BFF to wind coordinates

$$\boldsymbol{T_{WB}} = \boldsymbol{T_{WS}}\,\boldsymbol{T_{BS}}^{-1} = \begin{bmatrix} \cos\alpha\cos\beta & \sin\beta & \sin\alpha\cos\beta \\ -\cos\alpha\sin\beta & \cos\beta & -\sin\alpha\sin\beta \\ -\sin\alpha & 0 & \cos\alpha \end{bmatrix} \tag{2.9}$$

2.3 Equations of Motion

The focus of this section will be several differential equations that together make up the model of the motion of aircraft in space. In all of these equations, we will consider the body of aircraft to be perfectly rigid.

First two sets of equations are realizations of the Newton's second law of motion and describe the changes in aircraft's translation and rotation depending on the forces acting on it. The next equations then describe these changes further in different frames.

2.3.1 Differential Equations of Force

The force equations of motion for rigid body can be obtained by first realizing the second Newton's law for each differential element of mass of the body, and integrating over the whole vehicle, resulting in the following set of differential equations for components of the

velocity u, v, w in each of the coordinate directions [10]

$$\dot{u} = rv - qw + f_x \tag{2.10}$$
$$\dot{v} = pw - ru + f_y \tag{2.11}$$
$$\dot{w} = qu - pv + f_z \tag{2.12}$$

The resulting equations contain not only contributions from forces in each of the directions in form of specific forces f_x, f_y, f_z but also terms describing the contributions due to the rotation rates p, q, r.

Furthermore, we can also add the effects of the gravitational force acting on the vehicle by transformation of the gravity vector from inertial (local geographic) system to body axis system using the methods described in section 2.2.1, obtaining the final set of equations [9]

$$\dot{u} = rv - qw - g\sin\theta + f_x \tag{2.13}$$
$$\dot{v} = pw - ru + g\cos\theta\sin\phi + f_y \tag{2.14}$$
$$\dot{w} = qu - pv + g\cos\theta\cos\phi + f_z \tag{2.15}$$

2.3.2 Differential Equations of Moments

Similarly to the force equations, the moment equations are realization of the rotational form of Newton's second law of motion. In this case, the incremental moment components are summed over the whole body, giving the total moments L, M, N about the coordinate axes.

Assuming that the origin of the body axis system is in the CG and the aircraft is symmetric about the plane defined by x_b and z_b axes, the following equations describing the rolling, yawing and pitching motion apply [10]

$$\dot{p} = \frac{I_y - I_z}{I_x} qr + \frac{I_{xz}}{I_x}\left(pq + \dot{r}\right) + \frac{L}{I_x} \tag{2.16}$$

$$\dot{q} = \frac{I_z - I_x}{I_y} pr + \frac{I_{xz}}{I_y}\left(p^2 r^2\right) + \frac{M}{I_y} \tag{2.17}$$

$$\dot{r} = \frac{I_x - I_y}{I_z} pq + \frac{I_{xz}}{I_z}\left(qr - \dot{p}\right) + \frac{N}{I_z} \tag{2.18}$$

The I_x, I_y, I_z symbols denote the moments of inertia about the corresponding axes, while I_{xz} is the product of inertia about the ox and oz axes. Thanks to the symmetricity of the aircraft, products of inertia about axes perpendicular to the plane of symmetry, I_{xy} and I_{yz}, are zero, and terms containing them are left out of the equations. Furthermore, since the origin of the body axis system is in the CG, there is no net moment caused by gravitational forces [9].

2.3.3 Differential Equations of Angular Velocity

Another set of differential equations relates the angular velocities p, q, r of the body fixed axes to the attitude rates $\dot{\phi}, \dot{\theta}, \dot{\psi}$ in the inertial frame [10]

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi\sec\theta & \cos\phi\sec\theta \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{2.19}$$

The inverse equation to eq. (2.19) is following

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} 1 & 0 & -\sin\theta \\ 0 & \cos\phi & \sin\phi\cos\theta \\ 0 & -\sin\phi & \cos\phi\cos\theta \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \tag{2.20}$$

2.3.4 Differential Equations of Position

The last set of differential equations defines relationship between the vehicle carried NED velocity and Latitude, Longitude, Altitude coordinates in ECEF frame using the WGS-84 reference ellipsoid [6]

$$\dot{\Phi} = \frac{V_N}{M_\Phi + h} \tag{2.21}$$

$$\dot{\lambda} = \frac{V_E}{(N_\Phi + h)\cos\Phi} \tag{2.22}$$

$$\dot{h} = -V_D \tag{2.23}$$

Velocity components V_N, V_E, V_D can be obtained by transformation of the velocity components u, v, w measured in the BFF to the local geographic frame using eq. (2.6), as described in section 2.2.1.

The meridian radius of curvature M_Φ and the prime vertical radius of curvature N_Φ for a given longitude Φ can be found using the following equations [6]

$$M_\Phi = \frac{a\left(1-e^2\right)}{\left(1-e^2\sin^2\Phi\right)^{\frac{3}{2}}} \tag{2.24}$$

$$N_\Phi = \frac{a}{\sqrt{1-e^2\sin^2\Phi}} \tag{2.25}$$

The semi-major axis a of the ellipsoid is one of the defining parameters for WGS-84, while the first eccentricity e is one of the constants derived from the defining parameters. Both constants are defined by [2] as follows

$$a = 6\,378\,137.0\text{m} \tag{2.26}$$

$$e = 8.181\,919\,084\,262\,2 \times 10^{-2} \tag{2.27}$$

2.4 Numerical Methods for Ordinary Differential Equations

Solving the differential equations in section 2.3 analytically is generally not practicable and for many purposes not even required. Instead, the equations can be solved effectively using a wide range of numerical methods. Detailed explanation of these numerical methods would be far out of scope of this work—thus, this section will only briefly focus on the simplest and best known one: the *Euler's method*.

2.4.1 Euler's Method

The Euler's method is a simple numerical method for solving ordinary differential equations in the following form with a given initial value Y_0

$$\begin{aligned} Y'(t) &= f(t, Y(t)), \qquad t_0 \leq t \leq b \\ Y(t_0) &= Y_0 \end{aligned} \tag{2.28}$$

The result of Euler's method is an approximate solution $y(t)$ at a discrete set of nodes $t_0 < t_1 < \cdots < t_N \leq b$, given by the following equation [5]

$$y_{n+1} = y_n + hf(t_n, y_n) \tag{2.29}$$

The principle behind the Euler's method is visualized in fig. 2.6: In order to calculate the value y_{n+1}, the slope of the line tangent to $Y(t)$ at t_n is obtained using the function $f(t_n, y_n)$ and multiplied by the distance of step h.

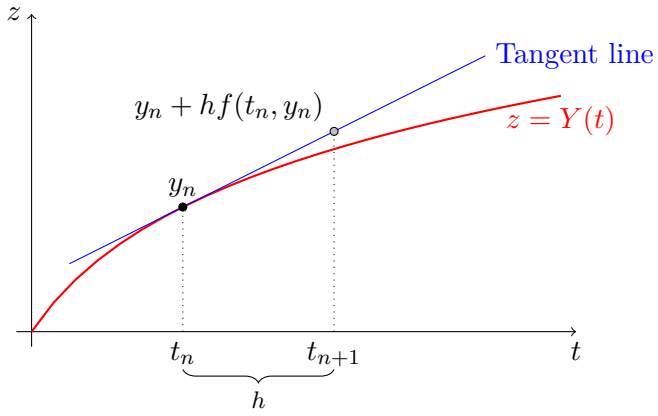


Figure 2.6: An illustration of a single step of the Euler's method.

As can be seen, the approximate value y_{n+1} obtained using the Euler's method may have a quite noticeable error when compared to the real value of $Y(t)$ at t_{n+1}. While this error decreases with the size of the step h, it is often still too large compared to the more advanced numerical methods [5]. As such, the Euler's method is considered ineffective and rarely used in practice, where more effective methods, such as the families of Runge–Kutta methods, are being preferred.

Chapter 3

State-of-the-Art Flight Data Technologies

Ever since the beginnings of modern aviation there has been a constant need to measure aircraft flight parameters for various purposes. With the ever increasing number of aircraft, air travel and increasing safety requirements, both quality and quantity of these measurements must grow. Today's aircraft are not only sophisticated mechanical machines, but also complex computer networks interconnecting many various devices, with strict requirements on reliability [16].

These networks fulfill large number of tasks and purposes, and covering them all would be far out of scope of this work. Therefore, this section will only focus on small part of this topic, covering it from perspective of measurement and collection of flight data. First, sources of these data—sensors—will be briefly discussed. Then, we will focus on one of the protocols designed for use in aircraft networks, as these networks transport measured data from sensors to various other devices. Next, an overview of devices and architecture used to safely store measured data is presented.

The last part of this chapter is dedicated to a short overview of publicly available dataset of flight data obtained using similar technologies to the ones described in this chapter. Furthermore, these data have been used for development and evaluation of event detection methods described in chapter 4.

3.1 Sensors

In order to measure various physical quantities and other parameters, aircraft contain a large collection of different sensors that utilize various physical phenomena. The amount of different sensor types used in aviation is simply too large to be covered here. Therefore, this section will only provide a brief look at principles behind inertial sensors—one of the key components of aircraft avionics.

3.1.1 Gyroscopes

A traditional gyroscope utilizes the principle of conservation of angular momentum—a spinning mass or wheel will try to hold its position in space. If the spinning wheel is mounted in a gimballing system comprised of three gimbals, the gimballing system will allow the wheel to stay fixed in the space and the roll, pitch, and yaw angles may be measured. These gyroscopes are commonly known as *position gyroscopes* [16].

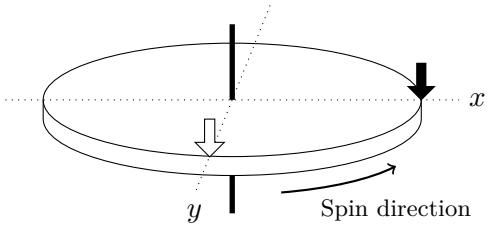


Figure 3.1: Gyroscopic precession.

If a force is applied to an outward edge of the spinning wheel as shown in fig. 3.1 by white arrow, the resulting force will act by rotating along the y axis as shown by the black arrow. This property of gyroscopic precession allows to measure the body roll, pitch, and yaw rates using the *rate gyroscopes* [16].

While mechanical gyroscopes using these principles were commonly used in older airplanes, the mechanical complexity, power consumption, and cost were major limitations. This has led to development of various kinds of solid-state gyroscopes. Nowadays, most modern aircraft systems use optical gyroscopes, such as the Ring Laser Gyroscope (RLG) [16].

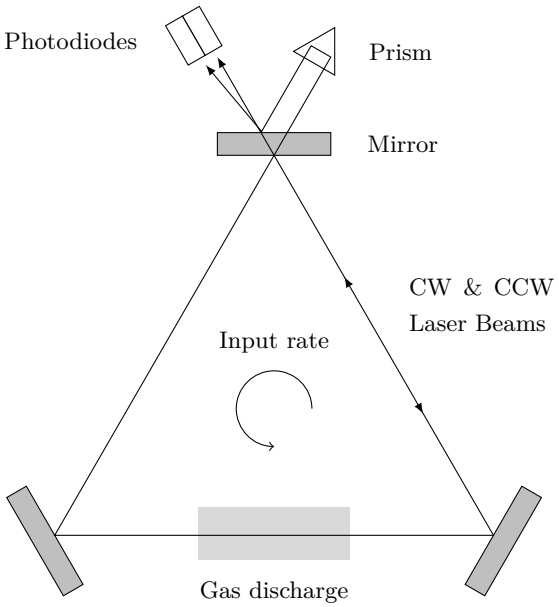


Figure 3.2: Ring Laser Gyroscope.

In Ring Laser Gyroscope (RLG), two counter rotating laser beams form a continuous light path using a series of mirrors, as shown in fig. 3.2. When the sensor is stationary, the frequency of the laser beams detected by the photodiodes is the same. If the sensor is rotating around the axis perpendicular to the plane of the laser beam (as indicated in the figure), the *Sagnac effect* will cause that the frequencies of the laser beams differ, making it possible to measure the rotation rate [16].

Other kinds of gyroscopes are also being used. A Fibre Optic Gyroscope (FOG) also uses similar principles as the RLG, however it is less complex and thus cheaper to make. Particularly noteworthy is also Microelectromechanical Systems (MEMS) technology, that allows manufacturing of much smaller and cheaper gyroscopes and other sensors.

3.1.2 Accelerometers

Similarly to gyroscopes that enable measurement of rotation about a particular axis, accelerometers enable measurement of acceleration along a given axis, which can be then integrated to obtain position. Generally, accelerometers use a small proof mass of known weight to sense forces acting on it [16].

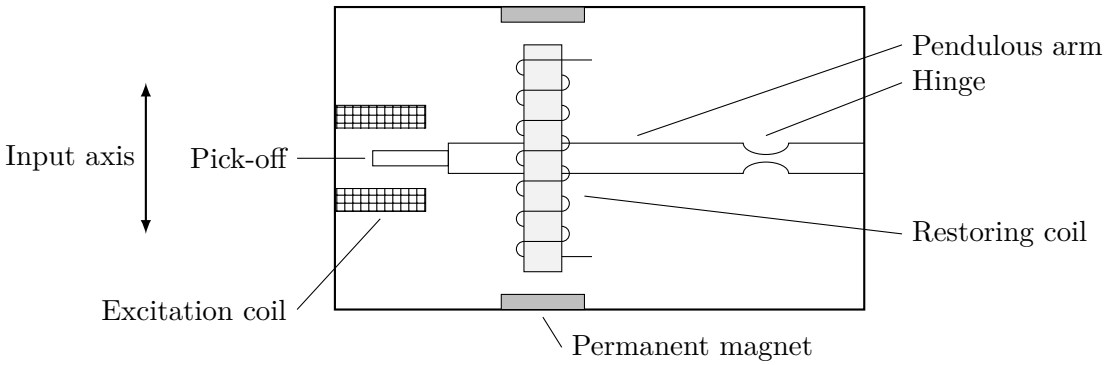


Figure 3.3: Closed loop pendulous force feedback accelerometer.

One of the commonly used types of accelerometers is a pendulous force feedback accelerometer, pictured in fig. 3.3. In this case, the proof mass is represented by a pendulous arm attached to case by a hinge that allows movement in the direction of input axis. The pick-off mechanism at the end of the arm is then used to determine displacement of the arm relative to the null position, which relates to the acceleration in the input axis. Furthermore, in case of a closed loop accelerometer, a corrective current is applied to the restoring coil attached to the arm, immediately returning it to the null position [16].

3.2 Data Buses

During the flight, data measured using the sensors described in section 3.1 and other data need to be distributed to various systems through the aircraft. In the past, this was achieved by direct connections between the necessary systems, resulting in a rather large amount of wiring. With the advent of digital computers in avionics, these direct connections were replaced by digital data buses, gradually reducing the amount of wiring, yet increasing the available bandwidth and enabling development of new applications [16].

Today's aircraft contain large amount of various avionics systems, often interconnected using a combination of multiple kinds of data buses and protocols. Many of these technologies, such as ARINC 429 or ARINC 629, are proprietary and their specifications are not publicly available. However, there are also open, free to use standards such as *CANaerospace*.

CANaerospace is a lightweight protocol designed for highly reliable communication of computer systems in airborne applications via Controller Area Network (CAN) [22]. The Controller Area Network is a serial communications protocol originally designed mainly for automotive electronics, but later found its use in many different applications. The relation between CAN and CANaerospace can be seen in fig. 3.4—CAN defines functions corresponding to the lower layers of Open Systems Interconnection (OSI) model, such as bit/frame encoding, timing, synchronization and error detection, while CANaerospace completes the network stack with definitions of logical channels, message structure, data representation and more [20, 22]. Both protocols will be explored in the following sections.

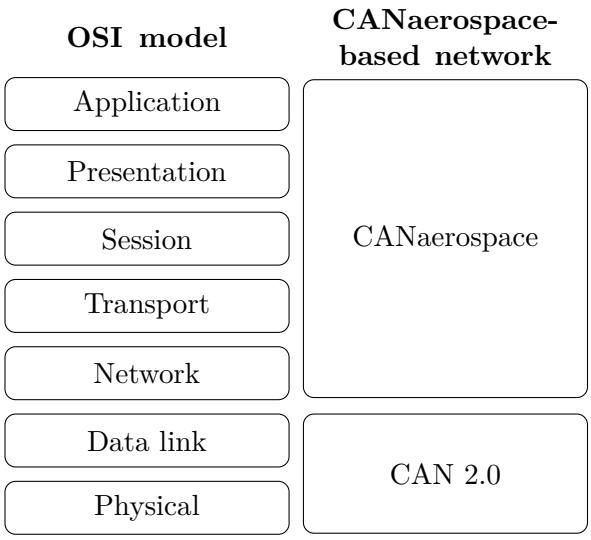


Figure 3.4: Relation between CAN and CANaerospace and correspondence of their features to the OSI model.

3.2.1 Controller Area Network 2.0

The Controller Area Network (CAN) is a serial communications protocol designed for distributed real-time control with high level of reliability. Among the main features of CAN are prioritization of messages, guarantee of latency times, configuration flexibility and error detection and signaling [20].

In CAN networks, nodes (at least two) are connected to a bus consisting of a high and low signal lines (see fig. 3.5). At any moment, the bus can be in one of two states: dominant or recessive. In an absence of any inputs, the bus is in recessive state by default. The recessive state of the bus is always overwritten by any single node that writes a dominant bit to the bus.

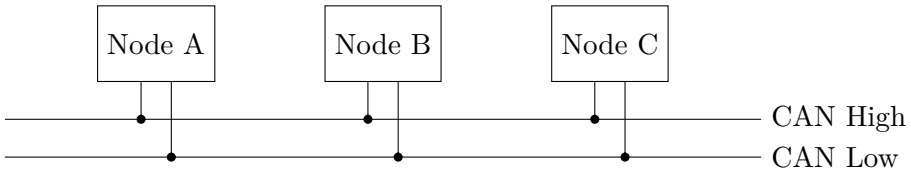
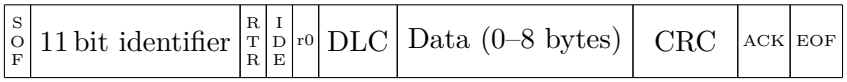


Figure 3.5: Simplified scheme of CAN bus.

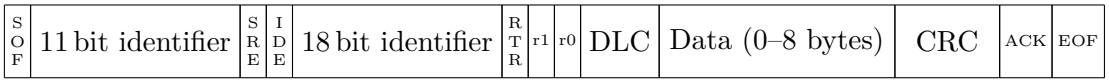
Any node can start transmitting messages at any time, provided that the bus is currently free. Unlike most of the other data buses, CAN message does not contain any kind of identification of sending or receiving node. Instead, it identifies type of the message and nodes then recognize and filter received messages according to this identification. This also ensures data consistency across the system, as every message is delivered to each node.

CAN 2.0, Part B specifies two kinds of message frames: a standard one with 11 bit message identifier, and an extended with 29 bit identifier. The format of both frames is depicted in fig. 3.6 and the meanings of the fields are following [20]:

- **Start of Frame (SOF)** Marks the beginning of a message, consists of a single dominant bit that enables synchronization of all nodes.



(a) Standard format



(b) Extended format

Figure 3.6: CAN 2.0 data frame format.

- **Identifier** Either 11 bit (standard format) or 29 bit (extended format) long identifier of message. Identifies the kind of transmitted message and determines the priority of message (will be described later).
- **Remote Transmission Request (RTR)** A node can request an information from another node (determined by the message identifier) by sending a message with dominant value of this bit.
- **Substitute Remote Request (SRR)** Placeholder bit used in place of the RTR in the extended message format.
- **Identifier Extension Bit (IDE)** Determines whether a standard (IDE set to dominant) or extended (IDE set to recessive) message format is used.
- **r1, r0** Reserved for future use.
- **Data Length Code (DLC)** Indicates number of bytes in data field.
- **Data field** Contains 0–8 bytes of data to be transmitted.
- **Cyclic Redundancy Code (CRC)** Contains 15 bit long checksum of the preceding data field for error detection, followed by single delimiter bit.
- **ACK field** Contains a 1 bit long slot used for acknowledgement of successful message transfer. Each node that successfully received the message overwrites the transmitted recessive bit by dominant bit.
- **End of Frame (EOF)** Each frame is delimited by a sequence of 7 recessive bits.

As was mentioned earlier, a node can start transmitting on bus only if it detects that no other node is currently transmitting. However, multiple nodes may start transmitting at the same moment, which would result in a conflict. This conflict is resolved using a non-destructive bitwise arbitration in the first phase of transmission.

During this phase, nodes transmit the message identifier bit by bit (see fig. 3.7), while they simultaneously read the value on bus and verify it against the transmitted value. If at least one node writes a dominant bit to the bus, every node that wrote a recessive bit will read back a dominant value. As the transmitted and received values do not match, the node loses the arbitration, stops the transmission and waits. At the end of the message identifier field, only one node with the lowest message identifier (highest priority) will remain and continue with transmission of the data [20].

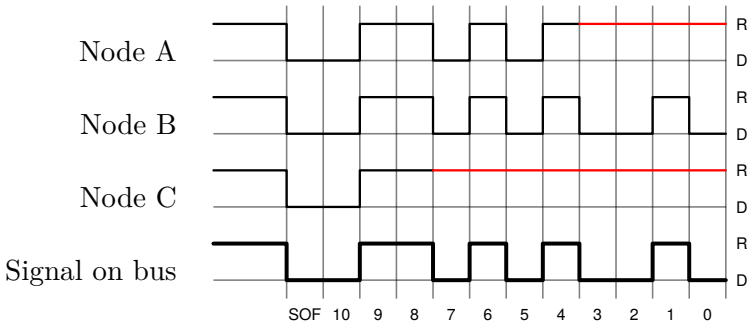


Figure 3.7: Signals transmitted by nodes and resulting signal on bus during CAN bitwise arbitration.

3.2.2 CANaerospace

The CANaerospace standard specifies additional layers and conventions around the basic CAN protocol that together provide a complete data communication system for aerospace applications comparable to commercial solutions [22].

CANaerospace communication is realized through messages, that are then encapsulated in the data field of the CAN frame. Messages in CANaerospace are categorized into 6 basic types, defined by CAN message identifier ranges allocated for specific purposes, as shown in table 3.1. Notice that CANaerospace also enables creation of user-defined message types with custom format and transmission intervals. Furthermore, the node service protocol provides a connection-oriented communication, either as High Priority or Low Priority Node Service Data.

Table 3.1: CANaerospace basic message types [22].

CAN ID	Message Type
0–127	Emergency Event Data (EED)
128–199	High Priority Node Service Data (NSH)
200–299	High Priority User-Defined Data (UDH)
300–1799	Normal Operation Data (NOD)
1800–1899	Low Priority User-Defined Data (UDL)
1900–1999	Debug Service Data (DSD)
2000–2031	Low Priority Node Service Data (NSL)

The standard defines a general message format (shown in fig. 3.8) consisting of a 4 B long header, leaving the remaining 4 B of CAN data field for payload. While the exact meaning of header fields may slightly vary in some contexts, the general meaning is following [22]:

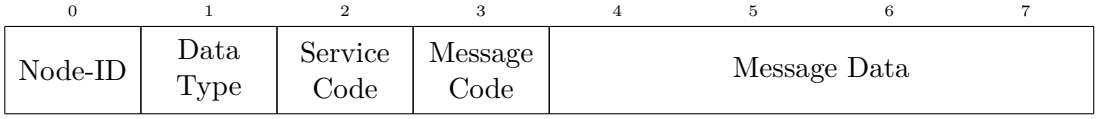


Figure 3.8: CANaerospace general message format.

- **Node-ID** Numerical node identifier in range of 0–255, with 0 being the broadcast identifier addressing all nodes on bus. In normal messages, it identifies the transmitting station, while in node service data messages, it identifies the addressed node.
- **Data Type** Specifies the type and coding of data. Available basic data types are defined by the CANaerospace standard, while use of user defined types is also possible.
- **Service Code** For node service messages, the service code contains the node service code for the current operation. In normal messages, this field may be used as required for the specific data.
- **Message Code** In normal messages the value is incremented by one for each message, allowing any node to determine the age of a signal. Node service data messages use this field for extended specification of the service.

The CANaerospace standard also includes a default identifier distribution that specifies identifiers, data types, sign conventions and units to ensure interoperability between nodes. For this purpose, the Normal Operation Data (NOD) identifiers in range of 300–1499 are reserved. While use of the default distribution is recommended, the standard also allows user-defined distribution schemes. As an example, an extract from the default identifier distribution list is shown in table 3.2.

Table 3.2: Extract from the default identifier distribution list included in the CANaerospace specification [22].

CAN ID	Parameter name	Data types	Units	Notes
300	Body longitudinal acceleration	`FLOAT` `SHORT2`	g	forward: + aft: -
304	Body roll rate	`FLOAT` `SHORT2`	°/s	roll right: + roll left: -
321	Heading angle	`FLOAT` `SHORT2`	°	±180°

1036	GPS aircraft latitude	`DOUBLEL/`
		`DOUBLEH`
		`FLOAT`
		`SHORT2`
			°	

3.3 Data Acquisition and Recording

For safety reasons, recording of flight data is mandated by for most of aircraft involved in passenger operations, as these data can and often are invaluable source of information during aircraft incident investigation [16]. This is done using the Flight Data Recorder (FDR), colloquially also often referred to as "black box". However, these data are often useful during normal operation of aircraft, whether for service, quality assurance, or research purposes. For this reason, many current aircraft also contain a Quick Access Recorder (QAR) that provides faster and easier access to data [7].

Older FDRs were responsible both for processing and collection of data. The ever increasing number of recorded parameters necessitated development of the Flight Data

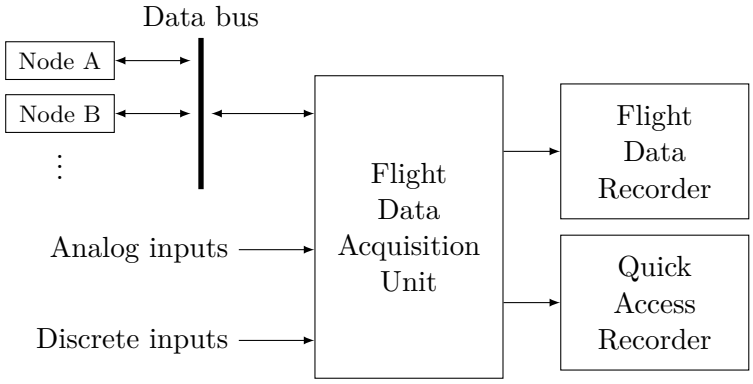


Figure 3.9: General architecture of flight data recording system.

Acquisition Unit (FDAU) responsible for collection and appropriate processing of measured values, which are then sent to FDR. Modern FDAUs are capable of collection of data from multiple sources, such as the aircraft data buses and various additional analog and discrete inputs [7]. A simplified, general architecture of the flight data recording system can be seen in fig. 3.9.

3.3.1 Flight Data Recorder

The primary task of the Flight Data Recorder (FDR) is to record and store flight data and to ensure their survival in case of an accident. For this reason, they are ruggedized to withstand shock, fire, and immersion in seawater. Furthermore, the casing of FDRs is painted bright orange to aid during recovery [16].

There have been numerous changes of used medium through the history of FDRs. Older recorders recorded data in analog form by engraving traces onto a metal foil. Later, they were replaced by digital FDRs that stored data on magnetic tapes. Nowadays, most aircraft use solid state FDRs that store data using solid state memory chips. This has enabled collection of more parameters during larger periods and significantly reduced need for maintenance [7].

3.3.2 Quick Access Recorder

As the FDR is usually not very accessible due to the high requirements on durability, many large airplanes carry an Quick Access Recorder (QAR). QAR receives and stores the same values as the FDR, but they are not designed to survive an accident.

On the other hand, they are easily accessible and data can be obtained by replacing the memory cards, or downloaded. Modern QARs also support wireless transfer of data, enabling even easier and faster access to data [7].

3.4 Flight Data Records

In order to develop, evaluate and demonstrate methods described in chapter 4, a sufficient dataset of flight data was required. While such dataset could be created using a flight simulator, it would not capture all intricacies of the real world. Different pilots, technical status of the aircraft, weather conditions, sensor imperfections—these are only some of the factors that differentiate real world data from simulated.

While collection of such data is nowadays common, for example in commercial aviation as part of Flight Data Monitoring (FDM) or Flight Operational Quality Assurance (FOQA) programs, they are practically never available publicly. Fortunately, National Aeronautics and Space Administration (NASA) has published such dataset to general public as a part of its Aviation Safety Program [17].

This section will provide an overview of this dataset and data contained in it—their origin, format, available parameters, some of the artifacts that can be found in these data, and how they are preprocessed before use.

3.4.1 Overview

The sample flight data have been published by NASA to provide researchers and general public access to data which can be used to evaluate and advance data mining capabilities that can be used to promote aviation safety [17].

These data have been obtained from a single type of regional jet over the period of three years and contain detailed aircraft dynamic, system performance and other engineering data [17]. They are, however, de-identified and do not provide any information that can be traced to a particular airline or manufacturer.

3.4.2 Format

The data can by downloaded from the project page on the NASA DASHlink portal[1]. They are split into 35 datasets, each containing 5–10 ZIP archives, with each of the archives containing several hundreds of flights.

Flight data of each flight are stored in MATLAB 5.0 MAT-file and store 186 parameters[2]. Each of the parameters is stored in a structure that contains the parameter name, description, units, sampling rate and the recordings itself.

[1] Sample Flight Data project page on DASHlink portal: https://c3.nasa.gov/dashlink/projects/85/

[2] Full list of parameters is available in appendix A

Chapter 4

Design of Flight Phases and Events Identification

A phase of flight refers to a period within a flight, such as taxi, takeoff, or landing. Each of these phases has its own characteristics and can be related to a set of operations usually performed at such phase. For this reason, phases are commonly used as a simple way to refer to a part of flight that is of interest.

Furthermore, we will assume that phases of flight do not overlap, and that each part of flight should be assigned a phase, if it is possible from the available information. Phases of flight used in this work are based on the taxonomy created by CAST/ICAO Common Taxonomy Team [8].

Another area of interest is detection of other events that occur during the flight. This project focuses on detection of turns—maneuvers that result in change of direction the aircraft is flying towards.

This chapter will focus on methods developed and used for identification of flight phases and events. The detection is done in multiple steps, where each step uses information generated by previous steps in order to simplify or improve detection. Thus, the used methods itself are relatively simple, but they have proven to be sufficiently powerful on tested data.

However, real world sensor data often contain various unwanted artifacts which complicate further processing. Therefore, first section of this chapter shorty focuses on these artifacts and how they were removed before further processing.

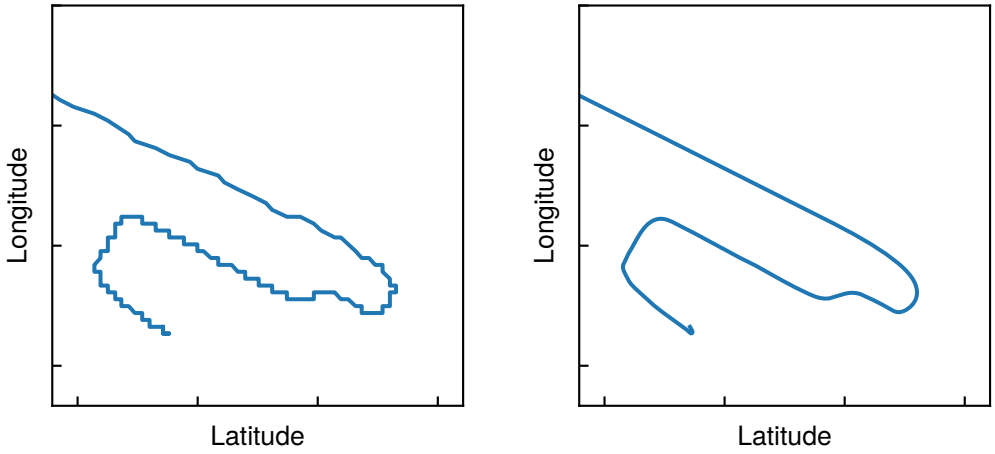
4.1 Preprocessing of Flight Data

Whether due to hardware limitations of sensors, choice of format used for storage of data in FDR or other reasons, flight data often contain artifacts and other elements that may not always be suitable for analysis or visualization. This section will describe methods that were applied before analysis and visualization of the flight data.

4.1.1 GPS Track Smoothing

The resolution of stored data is limited and depends on the format used by FDR. The accuracy is usually sufficient for most parameters such as roll and pitch angles where 1° change results only in a very small movement. However, a 1° change in latitude or longitude may represent as much as 111 km on the surface of Earth, and as such, much higher resolution is

needed. Results of insufficient resolution can be seen as a visible stepping in track plotted in fig. 4.1a. This stepping is especially noticeable during the taxi phase, when the aircraft slowly moves to or from the runway.



(a) Before (b) After

Figure 4.1: Demonstration of the track smoothing algorithm.

Such stepping could cause issues during analysis and is definitely not suitable for visualization. A relatively simple, but effective method of smoothing that can be used to remove this stepping has been described on the Flight Data Community blog [14].

The core of this method is to repeatedly use discrete convolution to smoothen the track by averaging out each of the samples with two preceding and succeeding samples. In order to simplify the process, first and last two samples are left untouched. The rest of the samples are obtained from results of convolution[1] $d * k_s$, where d is a vector of unprocessed samples (or samples from previous iteration), and k_s is the smoothing kernel. An example of a good smoothing kernel that has been used in this project can be seen in eq. (4.1).

$$k_s = \left[\tfrac{7}{40} \quad \tfrac{7}{40} \quad \tfrac{3}{10} \quad \tfrac{7}{40} \quad \tfrac{7}{40}\right] \tag{4.1}$$

A cost function c (eq. (4.2)) is introduced to stop the smoothing at an optimal point—when the track is as close to a straight line as possible, but not too far away from original data. This cost function consists of two components: First component (eq. (4.3)) is a simple total sum of squares between original data d and smoothened data d', which represents error caused by smoothing. The straightness of the smoothened line is represented by second component (eq. (4.4)), calculated by summing the squares of results of the convolution of

[1] This corresponds with the `'valid'` mode of MATLAB's `conv()` function

smoothened data d' and error kernel k_e.

$$c = c_o + 1000c_s \tag{4.2}$$

$$c_o = \sum_i \left(d'_i - d_i\right)^2 \tag{4.3}$$

$$c_s = \sum_i \left(d' * k_e\right)[i]^2 \tag{4.4}$$

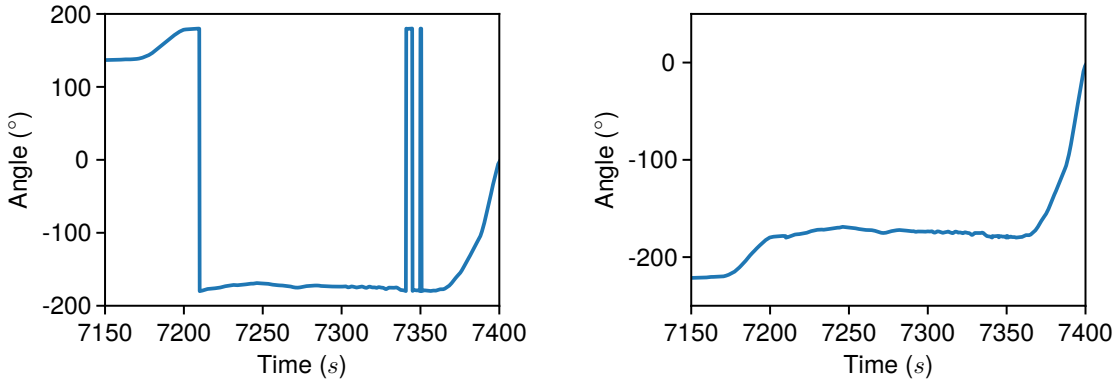
$$k_e = \begin{bmatrix} -1 & 2 & -1 \end{bmatrix} \tag{4.5}$$

The result of smoothing can be seen in fig. 4.1b. Both latitude and longitude are smoothened at the same time. The cost function is calculated for both latitude and longitude, its results are added together and used to stop the smoothing when the sum of cost functions is at its minimum.

In the final application (described in chapter 5) this algorithm has been extended to preprocess not only latitude and longitude, but also altitude of the aircraft. This results in a smooth movement in all three axes.

4.1.2 Phase Unwrapping

Parameters representing angles are stored phase wrapped—normalized in range from $-180°$ to $180°$. While this may ease storage of these data in FDR, phase wrapping would significantly complicate use of the data during analysis and results in unwanted behavior during visualization, visible for example in fig. 4.2a.



(a) Before

(b) After

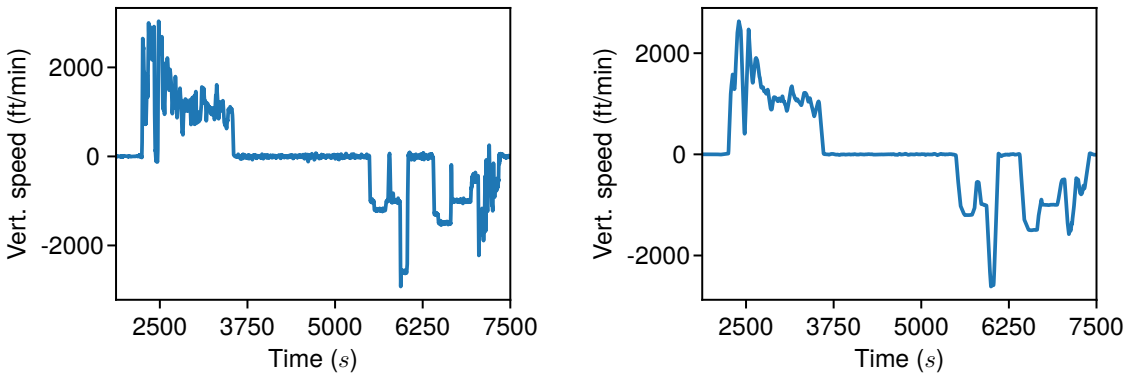
Figure 4.2: Values of true heading before and after phase unwrapping.

Fortunately, the principle of algorithm for phase unwrapping is very simple [12]: Iterate through all values and for each pair of succeeding values, calculate their difference. If the difference is larger than π rad, then subtract 2π rad from this and all values to right. Conversely, if the difference is smaller than $-\pi$ rad, add 2π rad to this and all values to right.

The result of phase unwrapping is a smooth, continuous curve, such as the one seen in fig. 4.2b.

4.1.3 Smoothing Other Parameters

In some cases, the used sensors may be too sensitive and produce noise that may cause issues with some detection methods. A good example is the *inertial vertical speed* parameter shown in fig. 4.3a, which is used for detection of cruise phases (described in section 4.3).



(a) Before

(b) After

Figure 4.3: Values of inertial vertical speed before and after application of moving average filter.

One of the easiest ways to solve this issue is filtering the signal using a moving average filter. While very simple to understand and implement, the moving average filter is often the optimal choice for reducing random noise in time domain encoded signals [21]. As the eq. (4.6) shows, each filtered sample $y[i]$ is obtained by averaging values of m samples around the original value $x[i]$.

$$y[i] = \frac{1}{m} \sum_{j=-\frac{m-1}{2}}^{\frac{m-1}{2}} x[i+j] \tag{4.6}$$

Result of filtering the inertial vertical speed parameter using a moving average filter can be seen in fig. 4.3b. Notice that the noise visible around zero values has been significantly reduced.

4.2 Takeoff and Landing Detection

Takeoff is defined as a phase starting with the application of takeoff power through rotation and to an altitude of 35 feet above runway elevation [8]. Landing phase starts with the beginning of the landing flare[2] and ends when the aircraft exits the landing runway or comes to a stop on the runway [8].

Detection of takeoffs and landings is performed in two steps. First, the *weight on wheels* parameter is used to approximate the takeoff and landing as single points in time of flight. These approximations are then used as a starting point in next step, in which time intervals of takeoff and landing phases are determined using additional parameters.

[2] A maneuver during which aircraft transitions from nose-low to nose-up attitude before landing

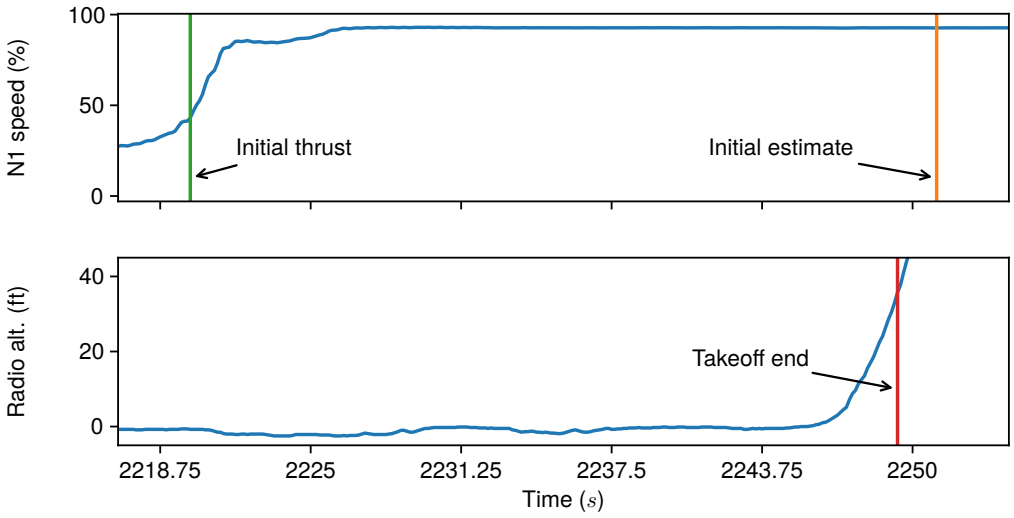


Figure 4.4: Parameters used for takeoff detection and detected start and end times.

Intervals between detected takeoff and landing phases are then classified as intervals "on ground" and "in air". This simplifies detection of other phases (described in next sections), as only appropriate detections are performed (e.g., it doesn't make sense to attempt to detect cruise phase when the aircraft is on ground).

4.2.1 Takeoff Detection

Detection of takeoff (see fig. 4.4) starts from the initial approximation and searches left for application of takeoff power, identified from the raise of engines N1 speed above 50 % RPM. As the search always starts from the initial approximation, only the closest application of takeoff power is considered, while any other instances where the N1 speed crosses 50 % RPM are ignored.

The end of takeoff phase is then detected using the *radio altitude* parameter—first moment when aircraft radio altitude crosses 35 ft marks the end of takeoff.

4.2.2 Landing Detection

Landing detection (see fig. 4.5) uses similar methods as the takeoff detection. The search starts from the initial approximation and searches left for a landing flare. The landing flare is approximated as a moment when the aircraft pitch angle raises above 0°.

During landing, the true heading of aircraft will remain constant and equal to the runway heading. This fact is utilized for the detection of the phase end, as any significant deviation from the initial heading indicates that the aircraft left the runway. As it is part of airport operations procedures to immediately exit the runway without delay after landing, this method should successfully detect the end of landing in virtually all flights.

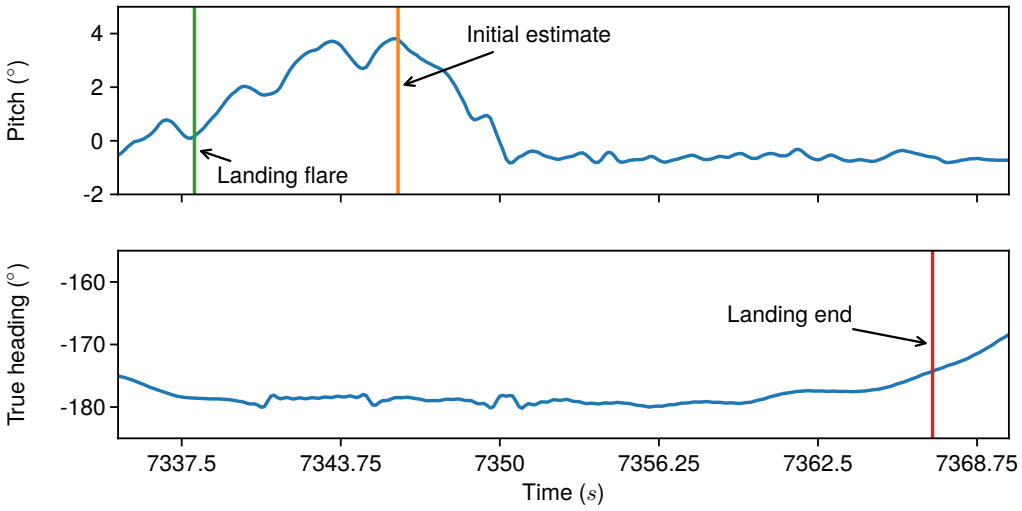


Figure 4.5: Parameters used for landing detection and detected start and end times.

4.3 Detection of Cruise Phase and Changes of Cruise Level

In this case, used definitions differ a little from [8] in order to better match phases detected by Aircraft Condition Monitoring System (ACMS). Any level flight segment will be considered as *cruise* phase, while changes in cruise level will be marked either as *climb* or *descent*.

First, as can be seen in top part of fig. 4.6, cruise phases are detected using the *inertial vertical speed* parameter by searching for any continuous intervals during which the speed remains constantly below $\pm 100\,\mathrm{ft/min}$.

The climb and descent phases then fill the undetected intervals between takeoff, cruise, and landing phases, as shown in bottom part of fig. 4.6. To determine whether the phase is an climb or descent, it's sufficient to consider the change of altitude between the two phases it is connecting.

4.4 Turn Detection

As the main objective of performing turns in aircraft is change of its direction, they can be easily detected from these changes. It is however necessary to differentiate between aircraft heading and course, as changes in their values may differ. While the heading represents direction the aircraft is pointing towards, course is the direction it's flying towards. Therefore, it is more appropriate to use course for turn detection.

The method itself is very similar to method used for detection of cruise phase (section 4.3). Here, the *track angle* parameter is first searched for sufficiently long parts of flight where the course does not change more than 2° since start of the segment. Other parts, where the course changes, are then classified as left or right turn depending whether the change in course was positive or negative, as can be seen in fig. 4.7.

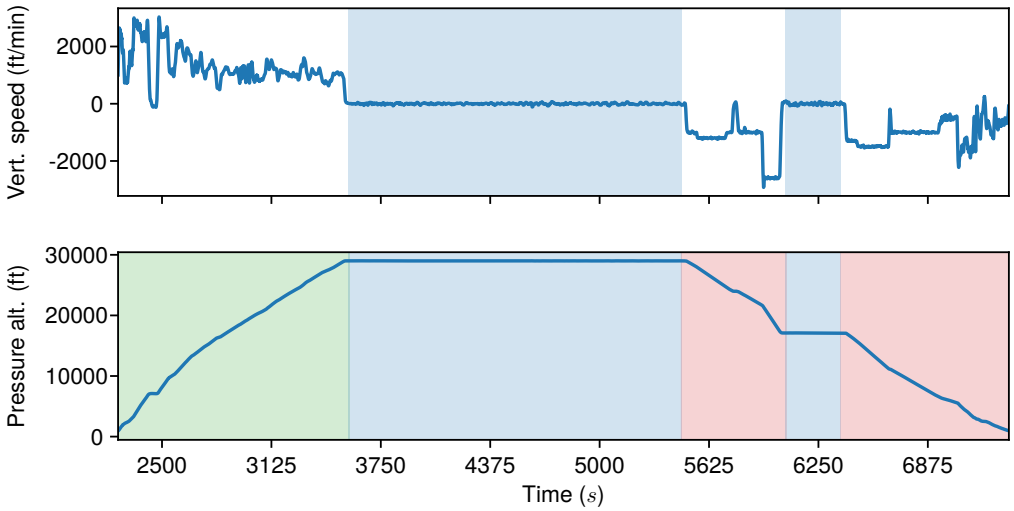


Figure 4.6: Cruise phases are detected from inertial vertical speed, remaining phases are classified from changes in altitude.

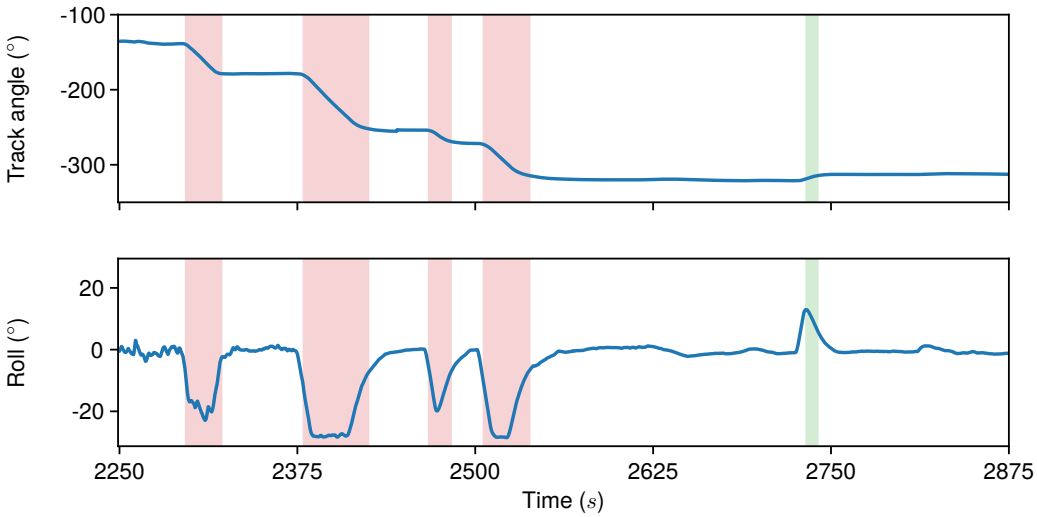


Figure 4.7: Detected left (red) and right (green) turns.

4.5 Overview of Detection Conditions

A short overview of detection conditions used for phase and event detection is provided by table 4.1. Where applicable, both start and end conditions are given.

Table 4.1: Phase and event detection conditions.

Ord.	Phase or event	Start condition	End condition	Notes
1	Takeoff	`N1_1` $> 50\,\%$	`RALT` $> 35\,\mathrm{ft}$	Start cond. searched from approximation
2	Landing	`PTCH` $> 0.5°$	`TH` change from initial value $> 5°$	Start cond. searched from approximation
3	Taxi	Must be on ground		
4	Cruise	Must be in air, `IVV` change does not exceed $100\,\mathrm{ft}/\,\mathrm{min}$		`IVV` filtered using moving average
5	Climb	Previous phase Takeoff or Cruise with lower FL	Next phase Cruise with higher FL	
6	Descent	Previous phase Cruise with higher FL	Next phase Landing or Cruise with lower FL	
7	Turn	Must be in air, change in `TRK` $> 2°$, at least $2\,\mathrm{s}$ long		

Chapter 5

Implementation

Since aircraft have six degrees of freedom, their movement is inherently more complex. While visualization using traditional two-dimensional charts is still possible, it is considerably harder to read and understand. This has led to decision to develop an interactive 3D visualization as a way to present information obtained using methods in chapter 4. Such form of visualization is not only easier to comprehend, but also allows inclusion of other context information about status of various aircraft systems and its surroundings.

The final application can be thought of as an analogue to a video player—which instead of video files replays flights from recorded flight data. The main view (see fig. 5.1) contains a simplified 3D model of an aircraft surrounded by terrain generated from real world height data and satellite view of a given location.

Furthermore, the file containing flight data is also analyzed using methods described in chapter 4. Detected flight phases and events are then shown in the User Interface (UI) and single click rewinds the player to the relevant part of flight.

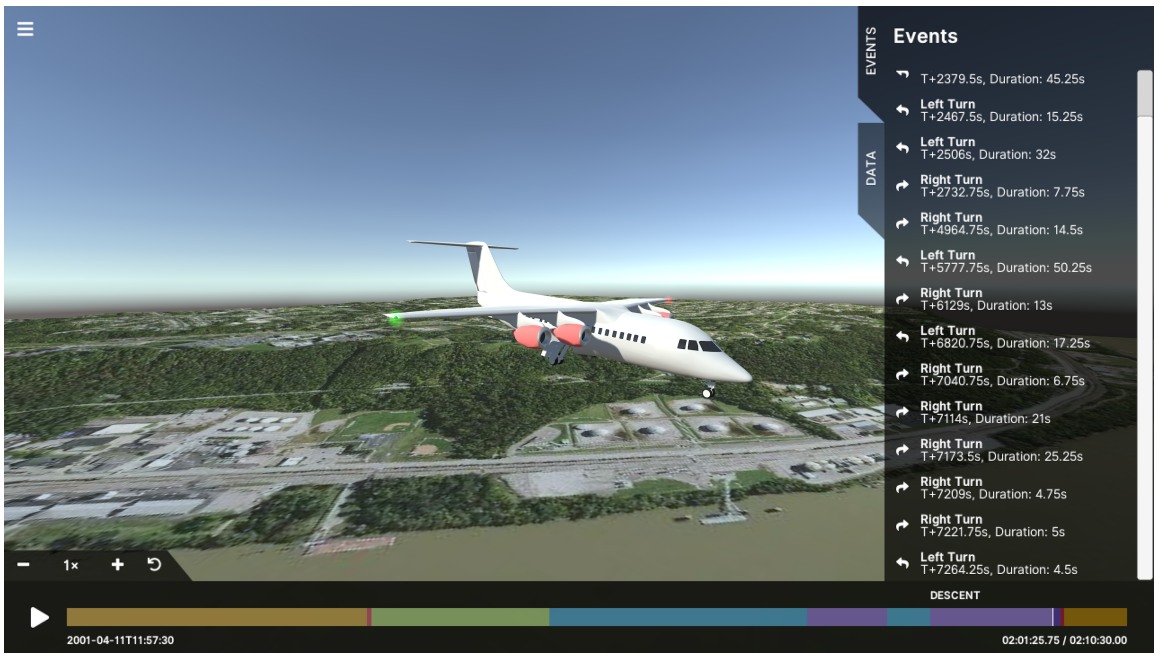


Figure 5.1: Main view of the application.

5.1 Used Technologies

The application was developed using the cross-platform game engine Unity. As a game engine, Unity is well equipped for 2D and 3D graphics, animation, lightning, UI creation, and more, making it a good choice for this purpose. This has, in turn, enabled to spend more time and focus on the process of identification and features of the final application. Furthermore, use of game engines outside the gaming industry is not unprecedented and has been spreading to more and more industries [3].

Unity uses Mono/.NET Framework as its scripting runtime and C# as a scripting language [24]. This, however, also enables use of any managed DLL assembly inside the application, which makes it possible to use a large number of third party libraries in case of need.

5.2 Flight Data Input and Processing

Given the fact that the methods used for detection of phases and events are relatively simple, they have been implemented as a direct part of the Unity application. This streamlines the user experience, as it is possible to open flight data files directly without any external preprocessing.

As the files containing flight data are distributed in a binary `.mat` format, an open-source library *CSMatIO* is used for reading. The data are then copied into more efficient data structures to improve performance during playback at a cost of longer loading and processing. At this step, parameters that require pre-processing are also pre-processed using methods described in section 4.1.

As the last step, the data are analyzed using the methods in chapter 4—first, the flight phases are detected, which are then used during detection of turns. The main scene and UI are then supplied with required data and prepared for use.

5.3 Terrain and Environment

A key part of the visualization is the terrain, as it forms most of the environment around the aircraft and serves as a visual reference of current position. Two different solutions for visualization of terrain were implemented—one utilizing height data and satellite imagery from the service *Mapbox*, the other using freely available data from Shuttle Radar Topography Mission (SRTM) and procedural, heightmap based texture splatting. By default, the application uses the Mapbox terrain, but it is possible to easily switch between providers at runtime. Comparison of both terrains can be seen in fig. 5.2.

The geographic area covered by flights in the dataset is too large to be loaded into memory and rendered at once. Furthermore, the limited precision of the `float` type used by Unity to specify position within the game world would soon start causing major issues [18]. Therefore, the terrain is split into multiple smaller chunks or tiles, and only the nearest few of them are loaded and rendered. To prevent issues with floating point numbers, the aircraft model is moved only within a limited area near the game world coordinate system origin.

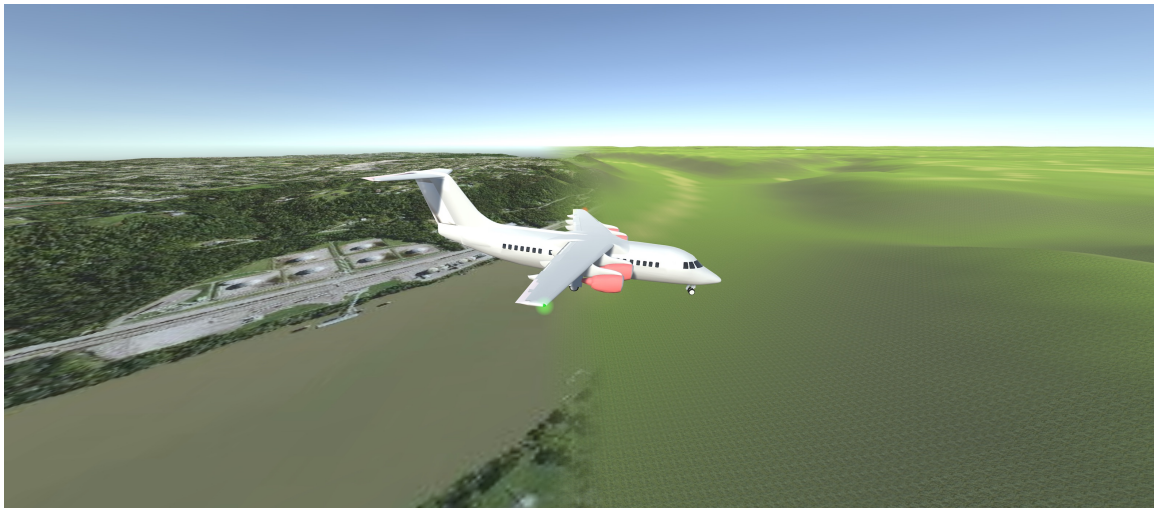


Figure 5.2: Comparison of both terrain solutions—Mapbox on the left, SRTM on right.

5.3.1 Mapbox SDK

Mapbox is a location data platform providing a range of tools and services for creating and publishing custom maps on web, embedded devices, and mobile platforms [4]. While Mapbox publishes large amount of their source code and Software Development Kits (SDKs) as open source, the service itself is commercial. However, a free starter plan is also available, with sufficiently large usage quotas for this project.

The main advantage of Mapbox was availability of both satellite imagery and terrain elevation data with same projection, and an open source SDK for Unity, which enabled faster integration into the application.

Mapbox serves its map data in form of tilesets—a collection of raster or vector data split into uniform grid of square tiles at 22 preset zoom levels (see fig. 5.3) in Web Mercator projection [4]. This principle has been popularized by Google Maps and has been heavily used in most of the web map applications. Data provided by Mapbox are collected from a large number of sources, both open and proprietary.

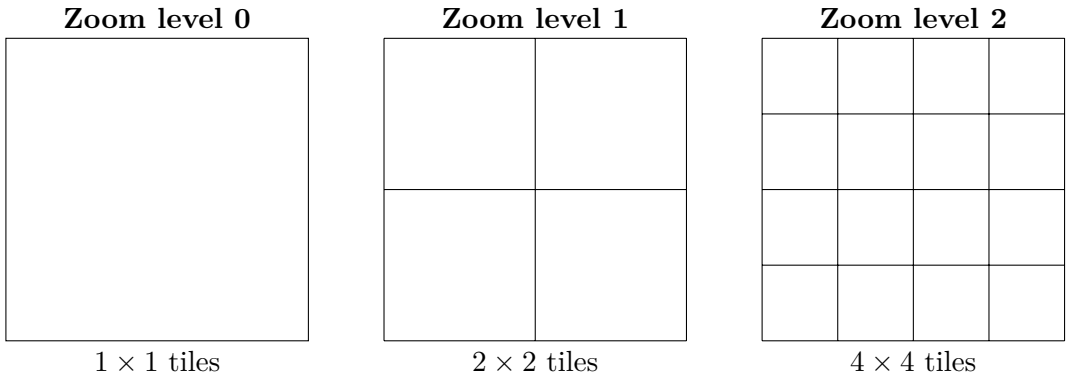


Figure 5.3: Illustration of tiles used by Mapbox. The same area is represented by different number of tiles on each zoom level.

All the background work related to loading of tiles from servers, creating terrain meshes and applying textures is handled by the Mapbox SDK, which is already optimized for regular movement of the map center and automatically loads and renders missing tiles. Thus, when the Mapbox maps are used in the visualization, the aircraft model is floating above the game world origin, while the terrain tiles shift horizontally as it moves on its path.

As the altitude of the aircraft increases, less detailed tiles covering larger area are needed. An ideal implementation would also consider the camera bounds and decrease the zoom level of more distant tiles. However, the Mapbox SDK is still in active development and doesn't seem to provide this exact functionality at this time. To achieve similar functionality, a minor modifications of the SDK code have been done and an additional custom code changes map zoom level depending on the aircraft altitude above ground level.

5.3.2 SRTM Heightmaps

Provided as an alternative to Mapbox, this solution is independent of third party services and utilizes freely available elevation data from the Shuttle Radar Topography Mission (SRTM) with the Unity's built-in terrain system.

The Shuttle Radar Topography Mission was an international project that created the first ever near global digital elevation model of the Earth. Data were collected using interferometric synthetic aperture radars, carried as a payload of the Space Shuttle Endeavour's mission STS-99 in February 2000 [11].

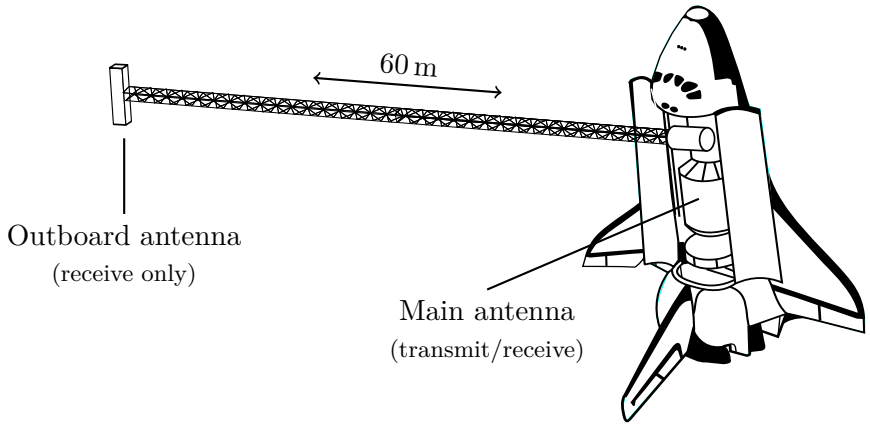


Figure 5.4: Illustration of the radar system configuration during the SRTM mission.

The SRTM radar system (see fig. 5.4) consisted of two antennas—a main antenna placed in the payload bay, and an outboard antenna attached to an end of a 60 m long extendable mast. During its 11 day mission, Endeavour completed 176 orbits around the Earth and collected elevation data for most of the land surface between 60 °N and 56 °S, covering about 80 % of Earth's land mass [11].

While the raw SRTM data contain voids and inaccuracies, they have been reprocessed numerous times by various organizations. Processed SRTM data are currently available in 1, 3, and 30 arc-second resolutions in various file formats.

For purposes of this project, data with 3 arc-second resolution distributed in `.hgt` files were used. These files contain 1201×1201 16 bit signed integer values in big endian byte

order, representing height of a given point on Earth, referenced to the EGM96 geoid [23]. Each of these files cover an area of $1° \times 1°$ of the Earth's surface.

Due to optimizations in Unity's terrain system, the resolution of a heightmap should[1] be a power of two plus one [24]. For this reason a Python script for bulk preprocessing of height data was created. This script resamples the entire heightmap into resolution supported by Unity, and optionally splits the heightmap into multiple smaller tiles. The resulting processed files are then saved in a format similar to `.hgt`, but with little endian byte order. Furthermore, the script also generates a file containing metadata used during loading of height data in Unity.

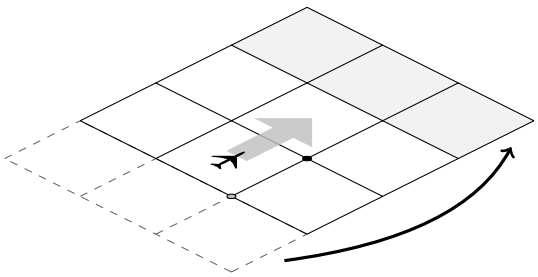


Figure 5.5: Loading of new tiles in front of the aircraft after crossing the edge of tile.

When SRTM elevation data are used by the application, it prepares an array of Unity's Terrain objects, which are reused during the entire run of application. The aircraft model is actually moved in both horizontal and vertical axes, but only within an area of a single Terrain object tile. This keeps the game world coordinates of the aircraft model relatively close to the origin and prevents issues caused by limited precision of floating point numbers. When the aircraft crosses the edge of the tile, all of the tiles and aircraft model are shifted back, and new terrain data are loaded in to appropriate tiles (see fig. 5.5). This whole solution is largely based on a method described by Robert Oates in his Unite 2013 talk [19].

In order to better highlight terrain features, a splatmap is procedurally generated at runtime from the terrain heightmap and used to apply textures on the terrain.

5.3.3 Sun Position

The recorded flight data also contain timestamps, enabling to establish time and date of the flight, which can be then visualized by properly setting lightning of scene to match current time and season. This also makes it easy to visually determine approximate time of day at the current location of aircraft without any confusion caused by changes of time zones during flight.

Unity has support for realtime global illumination and a sophisticated procedural sky system, that can be linked to a directional light (representing Sun) in scene [24]. In this case, Unity automatically matches the skybox according to current direction of light to create a realistically looking scene.

Therefore, it is only necessary to calculate altitude and azimuth angles of the Sun at given time, date, and location. These angles can be calculated by many methods with various accuracy. For purposes of this project, a relatively simple algorithm described by the U.S. Naval Observatory has been used. Accuracy of this algorithm is sufficient for

[1] Even though use of other resolutions is technically possible, the engine changes the resolution to nearest larger power of two and pads the height data with zeros to match the set resolution.

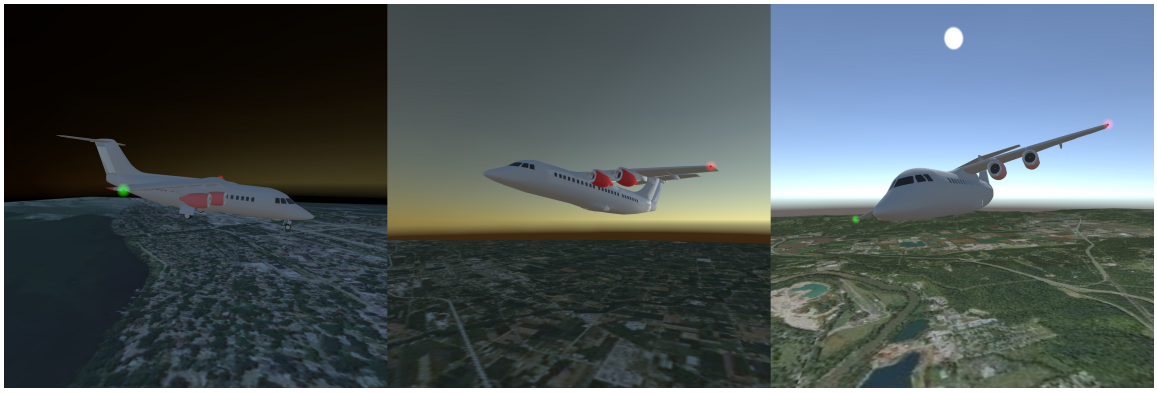


Figure 5.6: Position of the Sun at various times and places.

purposes of visualization—about 1 arcminute within two centuries of 2000 and gradually degrades [25].

5.4 Flight Playback

The main and most important part of the scene is the aircraft model, as it is used for visualization of most of the flight data. Due to the fact that the used flight data were de-identified, it was not possible to find and use the exact aircraft model in visualization. Instead, a freely available 3D model of an aircraft with similar characteristics was used as a substitute for visualization purposes.

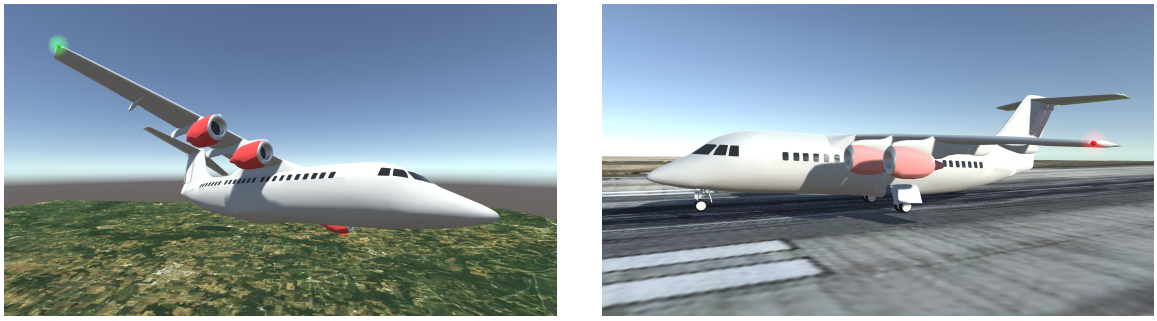
Currently visualized values of flight data are determined using an internal, frame independent time variable that represents the current time since the beginning of flight. This also enables user to speed up or down the visualization, play the flight in reverse or skip to any moment in the flight. To ensure smooth animation of the visualization, transition between the data samples is linearly interpolated.

The most important parameters that are visualized are, of course, position in the world and rotation around the CG. The position in the world is determined from recorded geographic coordinates and appropriately recalculated into game world coordinates depending on the currently used terrain provider. The rotation is obtained directly from the aircraft's true heading, roll, and pitch angles.

To see effects of aircraft controls on its movement, deflection angles of the main control surfaces—ailerons, elevator, and rudder—are visualized. These angles may be, mainly in case of ailerons, relatively small and hard to see. Therefore, the models of control surfaces are also colored according to their current deflection.

Similar color coding is used to visualize engine thrust. In this case, the engine nacelle is colored according to the engine's current N1 speed. While the N1 speed itself is not a measure of engine thrust (which is not measured), it is closely related to it and commonly used for this purpose.

Lastly, the aircraft landing gear deployment and retraction is visualized (see fig. 5.7). This parameter has obviously only two possible values—retracted, and deployed. In order to achieve smooth transition between these two states, it was necessary to animate it manually using Unity's animation system.



(a) Landing gear retracted (b) Landing gear deployed

Figure 5.7: Visualization of landing gear status.

5.5 User Interface

The user interface of the main view (see fig. 5.1) is partially inspired by interface of video players, and thus, it's relatively simple. The main focus is on visualization of the aircraft status and intuitive representation of results obtained using methods from chapter 4.

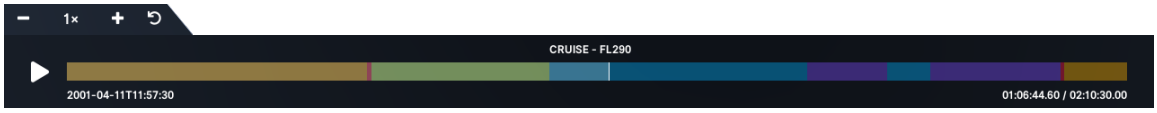
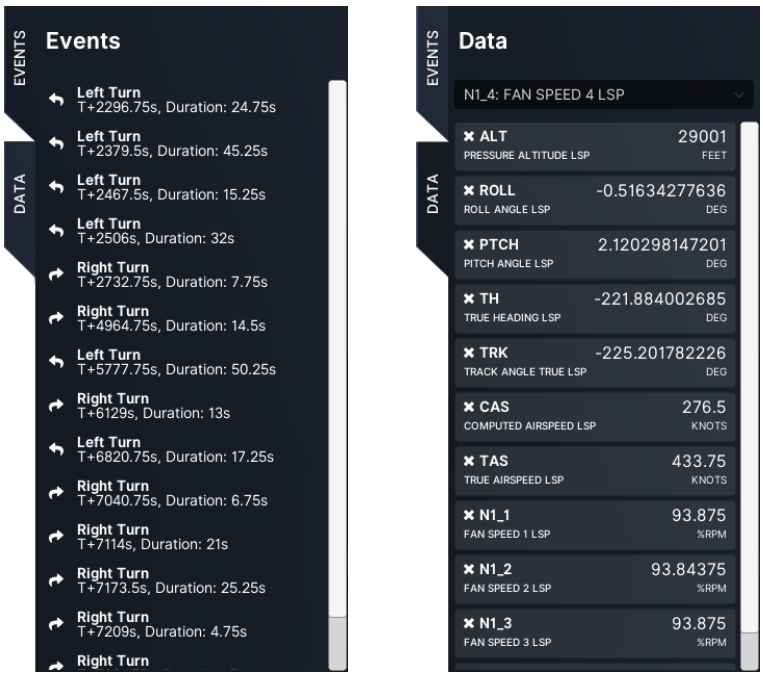


Figure 5.8: Progressbar and playback controls.

Bottom part of the main view, shown in fig. 5.8, contains controls related to playback. Besides play/pause button, there are also controls for controlling playback speed and player progressbar. This progressbar serves multiple purposes—it indicates both current time in the flight and detected flight phases, visualized as colorized segments of the progressbar. Naturally, it can be also used to skip to any part of the flight by clicking or dragging.

The right part contains a hideable sidebar. By default it contains list of all detected events (turns), shown in fig. 5.9a. The sidebar can be switched to data view (see fig. 5.9b), where it is possible to add and watch any of the parameters available in the flight data file.



(a) Events view (b) Data view

Figure 5.9: Different views of the sidebar.

Chapter 6

Evaluation

As one of the parameters contained in the used datasets is flight phase obtained from ACMS, it is possible to easily evaluate performance of phase detection algorithm by directly comparing its results to this parameter. When tested on randomly selected collection of flight record files from three aircraft (around 600 files for each aircraft), the average success rate has been around 92 % after excluding malformed or invalid records. It is however important to note that the actual success rate may be slightly higher, as definitions of flight phases used by these methods and ACMS seem to be different.

Unfortunately, such evaluation of the turn detection performance is not possible, as there is no parameter that could be used to compare the results. Furthermore, manual analysis and labeling of the data to would be extremely time-consuming and unfeasible. Therefore, it is not possible to approximate number of undetected turn maneuvers, but visual verification using the application implemented in chapter 5 reveals that almost all detected turns have been correctly identified.

6.1 Potential Future Improvements

While the final application implements all of the core functionality and the detection algorithms work sufficiently well, there is, of course, still a lot of room for improvements in all areas. For example, the detection algorithms could be further tweaked for better performance and new detection capabilities could be added—such as detection of holding maneuvers, go-arounds, and other less common events. With appropriate additional data sources, information such as departure and arrival airport, used runway or historical weather records could be also obtained.

Data pre-processing also has some shortcomings that could be solved, namely issues with aircraft altitude inaccuracies and correction of displacements in GPS data. A lot of interesting upgrades could be done on the visualization side, such as general graphics improvements, increased terrain render distance or 3D visualizations of buildings. There are also many parameters that could possibly be visualized on the aircraft model, such as deflections of secondary control surfaces, or more detailed visualization of engine status.

Chapter 7

Conclusion

The main goal of this bachelor's thesis was to design and implement an application for detection of flight phases and events in data obtained from flight data recorder. In order to achieve this goal, a set of multiple methods for detection of various phases and turn maneuvers was developed and successfully tested. Publicly available dataset of flight data obtained from a regional jet has been used to develop and evaluate these methods.

The slightly experimental nature of this work and need to easily and intuitively evaluate results of the detection has eventually caused a much stronger focus on visualization of both flight data and results of detection. The result is an application that couples the detection algorithms with an interactive 3D visualization. Use of such visualization is not only more natural, given the complex nature of aircraft motion, but it makes possible to present vastly larger amount of contextual information on screen.

Bibliography

[1] Volume 1. Performance Flight Testing. Chapter 13. Equations of Motion I. Technical report. U.S. Air Force Test Pilot School. November 1993.
Retrieved from: http://www.dtic.mil/docs/citations/ADA320205

[2] Department of Defense World Geodetic System 1984, Its Definition and Relationships With Local Geodetic Systems. NIMA Technical Report TR8350.2. July 1997.
Retrieved from:
http://earth-info.nga.mil/GandG/publications/tr8350.2/tr8350_2.html

[3] Engines of creation. *The Economist*. June 2016. ISSN 0013-0613.
Retrieved from:
https://www.economist.com/news/science-and-technology/21700618-slick-graphics-modern-video-games-are-spreading-ever-further-outside-their

[4] Mapbox. 2018.
Retrieved from: https://www.mapbox.com/

[5] Atkinson, K.; Han, W.; Stewart, D. E.: *Numerical Solution of Ordinary Differential Equations*. New York, NY: John Wiley & Sons. 2011. ISBN 978-1-118-16452-5. OCLC: 897576807.

[6] Cai, G.; Chen, B. M.; Lee, T. H.: Coordinate Systems and Transformations. In *Unmanned Rotorcraft Systems*. London: Springer London. 2011. ISBN 978-0-85729-634-4 978-0-85729-635-1. pp. 23–34. DOI: 10.1007/978-0-85729-635-1_2.
Retrieved from: http://link.springer.com/10.1007/978-0-85729-635-1_2

[7] Campbell, N. A. H.: The Evolution of Flight Data Analysis. 2007.
Retrieved from: http://www.asasi.org/papers/2007/The_Evolution_of_Flight_Data_Analysis_Neil_Campbell.pdf

[8] CAST/ICAO Common Taxonomy Team: Phase of Flight: Definitions and Usage Notes. April 2013.
Retrieved from: http://www.intlaviationstandards.org/Documents/PhaseofFlightDefinitions.pdf

[9] Caughey, D. A.: *Introduction to Aircraft Stability and Control Course Notes for M&AE 5070*. 2011.
Retrieved from: https://courses.cit.cornell.edu/mae5070/Caughey_2011_04.pdf

[10] Cook, M. V.: *Flight dynamics principles: a linear systems approach to aircraft stability and control*. Elsevier aerospace engineering series. Oxford [UK] ; Burlington,

MA: Butterworth-Heinemann/Elsevier. second edition. 2007. ISBN 978-0-7506-6927-6.

[11] Farr, T. G.; Rosen, P. A.; Caro, E.; et al.: The Shuttle Radar Topography Mission. *Reviews of Geophysics*. vol. 45, no. 2. May 2007. ISSN 8755-1209. doi:10.1029/2005RG000183.
Retrieved from: http://doi.wiley.com/10.1029/2005RG000183

[12] Gdeisat, M.; Lilley, F.: One-Dimensional Phase Unwrapping Problem.
Retrieved from: https://www.ljmu.ac.uk/~/media/files/ljmu/about-us/faculties-and-schools/tae/geri/onedimensionalphaseunwrapping_finalpdf

[13] Institute of Electrical and Electronics Engineers: *IEEE standard for inertial systems terminology*. New York: Institute of Electrical and Electronics Engineers. 2009. ISBN 978-0-7381-5996-6. OCLC: 468786271.
Retrieved from: http://ieeexplore.ieee.org/servlet/opac?punumber=5226533

[14] Jesse, D.: Track Smoothing. October 2013.
Retrieved from: http://www.flightdatacommunity.com/track-smoothing/

[15] Koks, D.: Using Rotations to Build Aerospace Coordinate Systems. Technical Report DSTO-TN-0640. DSTO Systems Sciences Laboratory. August 2008.
Retrieved from: http://www.dtic.mil/docs/citations/ADA484864

[16] Moir, I.; Seabridge, A.; Jukes, M.: *Civil avionics systems*. Aerospace series. Chichester: Wiley. second edition. 2013. ISBN 978-1-118-34180-3 978-1-118-53672-8 978-1-118-53673-5 978-1-118-53674-2. OCLC: 931242505.

[17] National Aeronautics and Space Administration: DASHlink - Sample Flight Data.
Retrieved from: https://c3.nasa.gov/dashlink/projects/85/

[18] Newson, D.: Unity: Coordinates and scales. April 2013.
Retrieved from:
http://davenewson.com/posts/2013/unity-coordinates-and-scales.html

[19] Oates, R.: GIS Terrain & Unity. January 2014.
Retrieved from:
https://unity3d.com/learn/resources/talks/gis-terrain-unity

[20] Robert Bosch GmbH: CAN Specification Version 2.0. September 1991.
Retrieved from: https://web.archive.org/web/20100922201217/http://www.semiconductors.bosch.de/pdf/can2spec.pdf

[21] Smith, S. W.: *The Scientist and Engineer's Guide to Digital Signal Processing*.
Retrieved from: http://www.dspguide.com

[22] Stock Flight Systems: CANaerospace Specification V 1.7. January 2006.
Retrieved from: http://www.stockflightsystems.com/canaerospace.html

[23] United States Geological Survey: SRTM Topography.
Retrieved from:
https://dds.cr.usgs.gov/srtm/version2_1/Documentation/SRTM_Topo.pdf

[24] Unity Technologies: Unity User Manual (2017.4).
Retrieved from: https://docs.unity3d.com/2017.4/Documentation/Manual/

[25] USNO Astronomical Applications Department: Approximate Solar Coordinates. November 2012.
Retrieved from: http://aa.usno.navy.mil/faq/docs/SunApprox.php

List of Acronyms

ACMS	Aircraft Condition Monitoring System
AoA	Angle of Attack
AoS	Angle of Sideslip
BFF	Body Fixed Frame
CAN	Controller Area Network
CAST	Commercial Aviation Safety Team
CG	Center of Gravity
ECEF	Earth-Centered, Earth-Fixed frame
EGM96	Earth Gravitational Model 1996
ENU	East-North-Up
FDAU	Flight Data Acquisition Unit
FDM	Flight Data Monitoring
FDR	Flight Data Recorder
FL	Flight Level
FOG	Fibre Optic Gyroscope
FOQA	Flight Operational Quality Assurance
GPS	Global Positioning System
ICAO	International Civil Aviation Organization
LLA	Latitude, Longitude, Altitude
MEMS	Microelectromechanical Systems
N1	Fan Rotation Speed on Jet Engine
NASA	National Aeronautics and Space Administration
NED	North-East-Down
OSI	Open Systems Interconnection model
QAR	Quick Access Recorder

RLG	Ring Laser Gyroscope
SDK	Software Development Kit
SRTM	Shuttle Radar Topography Mission
UI	User Interface
WGS-84	World Geodetic System 1984

List of Symbols

α	Angle of Attack
β	Angle of Sideslip
f_x, f_y, f_z	Components of the specific force
g	Gravitational acceleration
I_x, I_y, I_z	Moments of inertia
I_{xy}, I_{xz}, I_{yz}	Products of inertia
L, M, N	Components of the moments
M_Φ	Meridian radius of curvature
N_Φ	Prime vertical radius of curvature
Φ, λ, h	Latitude, longitude, and altitude
ϕ, θ, ψ	Roll, pitch, and yaw angles
p, q, r	Rotation rates around the axes
u, v, w	Components of the velocity
V	Flight path vector

Appendix A

List of Available Flight Parameters

The following table is a complete listing of flight parameters available in the NASA Sample Flight Data dataset described in section 3.4.

Parameter	Description	Units	Rate
ABRK	Airbrake Position	°	1 Hz
ACID	Aircraft Number	—	0.25 Hz
ACMT	ACMS Timing Used T1HZ	—	1 Hz
AIL_1	Aileron Position LH	°	1 Hz
AIL_2	Aileron Position RH	°	1 Hz
ALT	Pressure Altitude LSP	ft	4 Hz
ALTR	Altitude Rate	ft/ min	4 Hz
ALTS	Selected Altitude LSP	ft	1 Hz
AOA1	Angle of Attack 1	°	4 Hz
AOA2	Angle of Attack 2	°	4 Hz
AOAC	Corrected Angle of Attack	°	4 Hz
AOAI	Indicated Angle of Attack	°	4 Hz
APFD	AP FD Status	—	1 Hz
APUF	APU Fire Warning	—	2 Hz
ATEN	A/T Engage Status	—	1 Hz
A_T	Thrust Automatic On	—	1 Hz
BAL1	Baro Correct Altitude LSP	ft	4 Hz
BAL2	Baro Correct Altitude LSP	ft	4 Hz
BLAC	Body Longitudinal Acceleration	G	16 Hz
BLV	Bleed Air All Valves	—	1 Hz
BPGR_1	Brake Pressure LH Green	psi	1 Hz
BPGR_2	Brake Pressure RH Green	psi	1 Hz
BPYR_1	Brake Pressure LH Yellow	psi	1 Hz
BPYR_2	Brake Pressure RH Yellow	psi	1 Hz
CALT	Cabin High Altitude	—	1 Hz
CAS	Computed Airspeed LSP	kn	4 Hz
CASM	Max Allowable Airspeed	kn	4 Hz
CASS	Selected Airspeed	kn	1 Hz
CCPC	Control Column Position Capt	Counts	2 Hz
CCPF	Control Column Position F/O	Counts	2 Hz
CRSS	Selected Course	°	1 Hz
CTAC	Cross Track Acceleration	G	16 Hz
CWPC	Control Wheel Position Capt	Counts	2 Hz
CWPF	Control Wheel Position F/O	Counts	2 Hz
DA	Drift Angle	°	4 Hz
DATE_DAY	Date (Day)	Day	0.25 Hz
DATE_MONTH	Date (Month)	Month	0.25 Hz
DATE_YEAR	Date (Year)	Year	0.25 Hz

Parameter	Description	Units	Rate
DFGS	DFGS 1&2 Master	—	1 Hz
DVER_1	Database ID Version Char 1	—	0.25 Hz
DVER_2	Database ID Version Char 2	—	0.25 Hz
DWPT	Distance to Waypoint LSP	—	1 Hz
EAI	Engine Antice All Positions	—	1 Hz
ECYC_1	Engine Cycle 1 LSP	h	0.25 Hz
ECYC_2	Engine Cycle 2 LSP	h	0.25 Hz
ECYC_3	Engine Cycle 3 LSP	h	0.25 Hz
ECYC_4	Engine Cycle 4 LSP	h	0.25 Hz
EGT_1	Exhaust Gas Temperature 1	°	4 Hz
EGT_2	Exhaust Gas Temperature 2	°	4 Hz
EGT_3	Exhaust Gas Temperature 3	°	4 Hz
EGT_4	Exhaust Gas Temperature 4	°	4 Hz
EHRS_1	Engine Hours 1 LSP	h	0.25 Hz
EHRS_2	Engine Hours 2 LSP	h	0.25 Hz
EHRS_3	Engine Hours 3 LSP	h	0.25 Hz
EHRS_4	Engine Hours 4 LSP	h	0.25 Hz
ELEV_1	Elevator Position Left	°	1 Hz
ELEV_2	Elevator Position Right	°	1 Hz
ESN_1	Engine Serial Number 1 LSP	—	0.25 Hz
ESN_2	Engine Serial Number 2 LSP	—	0.25 Hz
ESN_3	Engine Serial Number 3 LSP	—	0.25 Hz
ESN_4	Engine Serial Number 4 LSP	—	0.25 Hz
EVNT	Event Marker	—	1 Hz
FADF	FADEC Fail All Engines	—	1 Hz
FADS	FADEC Status All Engines	—	1 Hz
FF_1	Fuel Flow 1	lb/h	4 Hz
FF_2	Fuel Flow 2	lb/h	4 Hz
FF_3	Fuel Flow 3	lb/h	4 Hz
FF_4	Fuel Flow 4	lb/h	4 Hz
FGC3	DFGS Status 3	—	1 Hz
FIRE_1	Engine Fire #1	—	1 Hz
FIRE_2	Engine Fire #2	—	1 Hz
FIRE_3	Engine Fire #3	—	1 Hz
FIRE_4	Engine Fire #4	—	1 Hz
FLAP	T.E. Flap Position	Counts	1 Hz
FPAC	Flight Path Acceleration	G	16 Hz
FQTY_1	Fuel Quantity Tank 1 LSB	lb	1 Hz
FQTY_2	Fuel Quantity Tank 2 LSB	lb	1 Hz
FQTY_3	Fuel Quantity Tank 3 LSB	lb	1 Hz
FQTY_4	Fuel Quantity Tank 4 LSB	lb	1 Hz
FRMC	Frame Counter	—	0.25 Hz
GLS	Glideslope Deviation	DDM	1 Hz
GMT_HOUR	Greenwich Mean Time (Hour)	Hour	2 Hz
GMT_MINUTE	Greenwich Mean Time (Minute)	Minute	2 Hz
GMT_SEC	Greenwich Mean Time (Second)	Second	2 Hz
GPWS	GPWS 1-5	—	1 Hz
GS	Ground Speed LSP	kn	4 Hz
HDGS	Selected Heading	°	1 Hz
HF1	HF Keying #1	—	1 Hz
HF2	HF Keying #2	—	1 Hz
HYDG	Low Hydraulic Pressure Green	—	1 Hz
HYDY	Low Hydraulic Pressure Yellow	—	1 Hz
ILSF	Ils Frequency LSP	—	1 Hz
IVV	Inertial Vertical Speed LSP	ft/ min	16 Hz
LATG	Lateral Acceleration	G	4 Hz
LATP	Latitude Position LSP	°	1 Hz

Parameter	Description	Units	Rate
LGDN	Gears L&R Down Locked	—	1 Hz
LGUP	Gears L&R Up Locked	—	1 Hz
LMOD	Lateral Engage Modes	—	1 Hz
LOC	Localizer Deviation	DDM	1 Hz
LONG	Longitudinal Acceleration	G	4 Hz
LONP	Longitude Position LSP	°	1 Hz
MACH	Mach LSP	MACH	4 Hz
MH	Magnetic Heading LSP	°	4 Hz
MNS	Selected Mach	MMACH	1 Hz
MRK	Markers- Inner, Middle, Outer	—	1 Hz
MSQT_1	Squat Switch Left Main Gear	—	2 Hz
MSQT_2	Squat Switch Right Main Gear	—	2 Hz
MW	Master Warning	—	1 Hz
N1C	N1 Command LSP	% RPM	4 Hz
N1CO	N1 Compensation	—	1 Hz
N1T	N1 Target LSP	% RPM	4 Hz
N1_1	Fan Speed 1 LSP	% RPM	4 Hz
N1_2	Fan Speed 2 LSP	% RPM	4 Hz
N1_3	Fan Speed 3 LSP	% RPM	4 Hz
N1_4	Fan Speed 4 LSP	% RPM	4 Hz
N2_1	Core Speed 1 LSP	% RPM	4 Hz
N2_2	Core Speed 2 LSP	% RPM	4 Hz
N2_3	Core Speed 3 LSP	% RPM	4 Hz
N2_4	Core Speed 4 LSP	% RPM	4 Hz
NSQT	Squat Switch Nose Main Gear	—	4 Hz
OIPL	Low Oil Pressure All Engines	—	1 Hz
OIP_1	Oil Pressure 1	psi	1 Hz
OIP_2	Oil Pressure 2	psi	1 Hz
OIP_3	Oil Pressure 3	psi	1 Hz
OIP_4	Oil Pressure 4	psi	1 Hz
OIT_1	Oil Temperature 1	°	1 Hz
OIT_2	Oil Temperature 2	°	1 Hz
OIT_3	Oil Temperature 3	°	1 Hz
OIT_4	Oil Temperature 4	°	1 Hz
PACK	Pack Air Conditioning All	—	1 Hz
PH	Flight Phase From ACMS	—	1 Hz
PI	Impact Pressure LSP	mbar	2 Hz
PLA_1	Power Lever Angle 1	°	4 Hz
PLA_2	Power Lever Angle 2	°	4 Hz
PLA_3	Power Lever Angle 3	°	4 Hz
PLA_4	Power Lever Angle 4	°	4 Hz
POVT	Pylon Overheat All Engines	—	1 Hz
PS	Static Pressure LSP	IN	2 Hz
PSA	Avarage Static Pressure LSP	mbar	2 Hz
PT	Total Pressure LSP	mbar	2 Hz
PTCH	Pitch Angle LSP	°	8 Hz
PTRM	Pitch Trim Position	°	1 Hz
PUSH	Stick Pusher	—	1 Hz
RALT	Radio Altitude LSP	ft	8 Hz
ROLL	Roll Angle LSP	°	8 Hz
RUDD	Rudder Position	°	2 Hz
RUDP	Rudder Pedal Position	Counts	2 Hz
SAT	Static Air Temperature	°	1 Hz
SHKR	Stick Shaker	—	2 Hz
SMKB	Animal Bay Smoke	—	1 Hz
SMOK	Smoke Warning	—	1 Hz
SNAP	Manual Snapshot Switch	—	1 Hz

Parameter	Description	Units	Rate
SPLG	Spoiler Deploy Green	—	1 Hz
SPLY	Spoiler Deploy Yellow	—	1 Hz
SPL_1	Roll Spoiler Left	°	1 Hz
SPL_2	Roll Spoiler Right	°	1 Hz
TAI	Tail Antice On	—	1 Hz
TAS	True Airspeed LSP	kn	4 Hz
TAT	Total Air Temperature	°	1 Hz
TCAS	TCAS LSP	—	1 Hz
TH	True Heading LSP	°	4 Hz
TMAG	True/Mag Heading Select	—	1 Hz
TMODE	Thrust Mode	—	1 Hz
TOCW	Takeoff Conf Warning	—	2 Hz
TRK	Track Angle True LSP	°	4 Hz
TRKM	Track Angle Mag LSP	°	4 Hz
VAR_1107	Sync Word for Subframe 1	—	0.25 Hz
VAR_2670	Sync Word for Subframe 2	—	0.25 Hz
VAR_5107	Sync Word for Subframe 3	—	0.25 Hz
VAR_6670	Sync Word for Subframe 4	—	0.25 Hz
VHF1	VHF Keying #1	—	1 Hz
VHF2	VHF Keying #2	—	1 Hz
VHF3	VHF Keying #3	—	1 Hz
VIB_1	Engine Vibration 1	IN/SEC	4 Hz
VIB_2	Engine Vibration 2	IN/SEC	4 Hz
VIB_3	Engine Vibration 3	IN/SEC	4 Hz
VIB_4	Engine Vibration 4	IN/SEC	4 Hz
VMODE	Vertical Engage Modes	—	1 Hz
VRTG	Vertical Acceleration	G	8 Hz
VSPS	Selected Vertical Speed	ft/ min	1 Hz
WAI_1	Inner Wing Deice	—	1 Hz
WAI_2	Outer Wing Antice	—	1 Hz
WD	Wind Direction True	°	4 Hz
WOW	Weight on Wheels	—	1 Hz
WS	Wind Speed	kn	4 Hz
WSHR	Windshear Warning	—	1 Hz