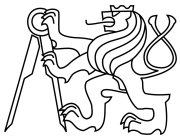
CZECH TECHNICAL UNIVERSITY IN PRAGUE
FACULTY OF INFORMATION TECHNOLOGY



# ASSIGNMENT OF BACHELOR'S THESIS

| | |
|---|---|
| **Title:** | Graphical display of radiation data of spacecraft payload SATRAM/Timepix |
| **Student:** | Ondřej Bílek |
| **Supervisor:** | doc. Ing. Carlos Humberto Granja, Ph.D. |
| **Study Programme:** | Informatics |
| **Study Branch:** | Computer Science |
| **Department:** | Department of Theoretical Computer Science |
| **Validity:** | Until the end of summer semester 2016/17 |

## Instructions

Design and implement a desktop application to generate a graphical display of evaluated data of the spacecraft SATRAM/Timepix payload on board of the ESA Prova-V satellite in the LEO orbit since 2013. Data should be displayed as 1D and 2D plots in the form of spatial- and time-correlated Earth maps. Display layout such as color scale must take into account the wide range of plotted data, e.g., log scale up to 10 orders of magnitude. Explore and implement creative graphical concepts including multi-parameter display (e.g., via parametric symbols or 3D volume graphics) as well as suitable viewing styles (projection, perspective). Input data should also be pre-processed, e.g., correction for empty cells in the resulting plot. Test and explore graphics plotting in a vector and bitmap format considering both low and high spatial resolution. For the desktop application, provision of a GUI tool is desirable.

## References

Will be provided by the supervisor.

L.S.

doc. Ing. Jan Janoušek, Ph.D.
Head of Department

prof. Ing. Pavel Tvrdík, CSc.
Dean

Prague February 4, 2016

Czech Technical University in Prague
Faculty of Information Technology
Department of Theoretical Computer Science

Bachelor's thesis

# Graphical display of radiation data of the spacecraft payload SATRAM/Timepix

***Ondřej Bílek***

Supervisor: doc. Ing. Carlos Granja, Ph.D.

11th May 2016

# Acknowledgements

I would like to thank my bachelor's thesis supervisor doc. Ing. Carlos Granja, Ph.D. for his guidance and knowledge that helped me during the writing of this thesis and I am grateful that I could be part of this exciting scientific project. I also want to thank my parents Miroslav Bílek, Magdaléna Bílková and my brother Tomáš Bílek and my girlfriend Kačenka for supporting me and cheering me up especially during these months.

# Declaration

I hereby declare that the presented thesis is my own work and that I have cited all sources of information in accordance with the Guideline for adhering to ethical principles when elaborating an academic final thesis.

I acknowledge that my thesis is subject to the rights and obligations stipulated by the Act No. 121/2000 Coll., the Copyright Act, as amended. In accordance with Article 46(6) of the Act, I hereby grant a nonexclusive authorization (license) to utilize this thesis, including any and all computer programs incorporated therein or attached thereto and all corresponding documentation (hereinafter collectively referred to as the "Work"), to any and all persons that wish to utilize the Work. Such persons are entitled to use the Work in any way (including for-profit purposes) that does not detract from its value. This authorization is not limited in terms of time, location and quantity. However, all persons that makes use of the above license shall be obliged to grant a license at least in the same scope as defined above with respect to each and every work that is created (wholly or in part) based on the Work, by modifying the Work, by combining the Work with another work, by including the Work in a collection of works or by adapting the Work (including translation), and at the same time make available the source code of such work at least in a way and scope that are comparable to the way and scope in which the source code of the Work is made available.

In Prague on 11th May 2016 .....

Czech Technical University in Prague
Faculty of Information Technology


### Citation of this thesis

Bílek, Ondřej. *Graphical display of radiation data of the spacecraft payload SATRAM/Timepix*. Bachelor's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2016.

# Abstrakt

Cílem této práce je vytvořit desktopovou aplikaci, která generuje grafické vizualizace radiačních dat. Data jsou poskytnuta detektorem Timepix zabudovaným v přístroji SATRAM, který je umístěn na nízké oběžné dráze Země od roku 2013 na palubě družice Evropské kosmické agentury Proba-V. Tato práce popisuje zpracování a úpravu dat, mapové projekce a implementační metodiky použité při tvorbě těchto grafických vizualizací. Dále také popisuje proces vývoje desktopové aplikace ve frameworku Electron za použití průběžné integrace.

**Klíčová slova** zobrazení dat, zpracování dat, grafická vizualizace, mapová projekce, desktopová aplikace, průběžná integrace, Electron framework

# Abstract

The subject and aim of this thesis is to develop desktop application for generating the graphical visualizations of space radiation data. Data is provided by the Timepix detector embedded in the spacecraft payload SATRAM on board the ESA's Proba-V satellite. The thesis describes the data processing, map projection and implementation methodology used to create these visualizations. This work also describes the desktop application development using the Electron framework with usage of continuous integration.

**Keywords** data display, data preprocessing, map projection, graphical visualization, desktop application, continuous integration, Electron framework

# Contents

# List of Figures

# Introduction

*"One picture is worth ten thousand words."*

— *Chinese proverb*

In todays world we are able to collect and store large amount of data, more than was possible ever before. To make use of this collected data we need a new range of powerful tools to assist us in extraction of valuable information.

Even when we manage to successfully mine our data we need to be able to visualize it for us to properly understand it. The quote on the top of this page applies very well in data science. Proper graphical visualization can reveal surprising detail that would be never visible when looking at the plain numbers.

Scientific instruments operating in space now has access to a higher data-rate downlinks and therefore can produce and transmit even more valuable data.

Space Application of Timepix based Radiation Monitor (SATRAM) is a technology demonstrator spacecraft payload successfully operating in open space since 2013 in Low Earth Orbit (LEO) at the altitude of 820km on board European Space Agency's (ESA) Proba-V satellite.

The instrument is equipped with the position-sensitive hybrid semiconductor pixel detector Timepix serving as a comprehensive and wide-dynamic range radiation monitor of the radiation environment. In particular the X-ray and charged particle field are studied and characterized in detail. The continuous and high frame rate operation allows for complete and systematic mapping of the space radiation environment along the satellite orbit. The SATRAM/-Timepix instrument provides the particle type flux and dose deposited, energy loss spectra and directional information. [1]

Downlinked data via the ESA REDU station are relayed to Prague where they are collected and processed by The Institute of Experimental and Applied Physics (IEAP) of the Czech Technical University in Prague (CTU).

This thesis describes the design and implementation of the desktop application that uses evaluated data to generate appropriate visualizations in the form of spatial- and time-correlated Earth maps. This data is collected by the

state of the art science instrument and the application would help experts at IEAP and other institutes with their research.

In the first chapter we will describe methods and theory that are referenced in later chapters. The second chapter introduces input data, how it is processed and analysis of visualizations followed by chapter about application design. In the fourth chapter we will discuss the implementation and finish with chapter showing the resulting graphical visualizations.

For more information about space radiation please refer to Appendix A that contains a brief description of space radiation environment. Appendix B summarizes radiation effects on human and electronic instruments and finally Appendix C links to the user documentation for this application.

Chapter 1

# Methods and Theory

In this chapter we will describe important methods and theory that were used in creating the desktop application. We will begin with general overview of data pre-processing as an important step of data mining.

## 1.1 Data pre-processing

Data pre-processing describes any type of transformation of raw data for another processing procedure [2]. It is commonly used in data mining because real world data are generally:

- **Incomplete:** missing attribute values or containing only aggregate data
- **Noisy:** corrupt data, containing errors or values that deviate from the expected
- **Inconsistent:** containing inconsistencies in the codes or names, different formats

Inaccurate, incomplete, and inconsistent data are commonplace properties of large real-world measurements. Elimination of poor quality data is essential for any further processing to yield meaningful output because the quality of the output is determined by the quality of the input.

Data cleaning (or data cleansing) focus on attempts to fill missing values (imputation), smooth out noise while identifying outliers and correct inconsistencies in data.

### 1.1.1 Missing values

Missing values can mean that data was not recorded or none of the possible values were appropriate. It is important to note that in some cases a missing value may not imply an error but is intentionally left blank. We can deal with missing data using following procedures.

- **Ignore the tuple:** This method deletes tuples that contain missing attributes. It is not very useful especially when there is a high percentage of missing values per attribute. By deleting tuple we also make no use of the remaining attributes.
- **Manual fill-in:** Time consuming approach that is not applicable for large datasets.
- **Use global constant:** Replace all missing values with a constant such as 0 or "Unknown". Some data mining algorithms can mistakenly interpret those constants as interesting similarity and generate misguided models. Simple, but not foolproof technique.
- **Use a measure of central tendency:** Missing values are replaced by mean or median of the parameter. In normal distribution a mean can be used while skewed data distribution should utilize the median because it better represents the central tendency of the distribution of the dataset. Improvement can be made by using the attribute mean or median for all samples that belongs to the same class. Another approach takes to the account only $k$ nearest instances using an appropriate distance function.
- **Use the most probable value:** Prediction can be made with regression (e.g., K-NN Regression) or classification model (e.g., Decision trees, Neural networks)

### 1.1.2 Data smoothing

Noise is a random error that occurs during the data measurement process. Methods of data visualization (e.g., Box plot, Scatter plot) can be used to identify outliers which may represent noise. No matter the data gathering process, noise inevitably exist. Following techniques are used for data smoothing.

#### Binning

Binning methods smooth data by taking to account its vicinity [3]. Data is sorted and distributed into bins. **Equal-width binning** is an unsupervised binning method that divides data into $k$ intervals of equal size. The width $w$ of the interval where $min$ and $max$ are minimal and maximal values is:

$$w = \frac{(max - min)}{2} \tag{1.1}$$

And the interval boundaries are:

$$min + w, min + 2w, \ldots, min + (k-1)w \tag{1.2}$$

This is a simple approach but empty bins may occur and outliers can cause uneven distribution. This method is not suitable for skewed data distributions.

**Equal-height binning** chooses intervals so that every bin has approximately the same number of values. This type of binning usually yields better results. There are also supervised binning methods that determine interval boundaries so that the best class separation is achieved.

With values distributed into bins we can use **smoothing by bin means** to replace each value by the mean of the values in the bin. Similarly, **smoothing by bin medians** replaces each value by the median of the bin. In **smoothing by bin boundaries** the minimum and the maximum values are labeled as bin boundaries. Each bin value is then replaced by its closest boundary.

### Clustering

Clustering is a process of organizing similar objects into groups. A cluster is therefore a collection of objects which are similar between themselves and dissimilar to the other objects. Typical is distance-based clustering which uses metric function to determine if objects are "close" to each other.

Objects that does not fall into any cluster can be considered outliers and dealt with accordingly. Two basic approaches are truncation and Winsorizing. Truncating excludes outliers from the dataset and is a controversial practice. Winsorizing replaces outliers with nearest valid observations or some other reasonable values [4].

### Regression

Most commonly used is linear regression where you can use one variable to predict the other. In linear regression a function in a form of a line that best approximates the data is used. Linear regression can be extended to make use of more variables in form of multilinear regression.

## 1.2 K-NN regression

K-nearest neighbors is a data mining algorithm used to predict property value for the object based on the other objects in its vicinity. It uses a similarity measure in form of distance functions. It is common to use distance weighting so that closer objects have greater influence [5]. The K in the name describes the number of closest objects that are considered.

### 1.2.1 Distance function

A metric on a set $S$ i.e. distance is a function:

$$d : S \times S \to [0, \infty) \tag{1.3}$$

### 1.2.2 Euclidean distance

Commonly used is an Euclidean distance. It is a length of a line between two points described by Cartesian coordinates. Let there be two points $\overrightarrow{p} = (p_1, p_2, \ldots, p_n)$ and $\overrightarrow{q} = (q_1, q_2, \ldots, q_n)$ in Euclidean $n$-space, then the distance $d$ from $\overrightarrow{p}$ to $\overrightarrow{q}$ or $\overrightarrow{q}$ to $\overrightarrow{p}$ is:

$$d(\overrightarrow{p}, \overrightarrow{q}) = d(\overrightarrow{q}, \overrightarrow{p}) = \sqrt{\sum_{i=1}^{n} (\vec{q_i} - \vec{p_i})^2} \tag{1.4}$$

### 1.2.3 K-nearest neighbors

Using this distance function we can determine $k$-nearest neighbors of point $\overrightarrow{p}$ from set $S$ using the following algorithm:

- **Step 1:** Compute the distance between point $\overrightarrow{p}$ and every other point in the set $S$. Choose the one with the smallest distance, remove it from the set $S$ and insert in the set $N$.
- **Step 2:** Repeat step 1 until the set $N$ contains $k$ points or the set $S$ is empty.
- **Step 3:** Set $N$ contains $k$-nearest neighbors of point $\overrightarrow{p}$.

### 1.2.4 Regression

Using a set $N$ of $k$-nearest neighbors we can calculate an approximation. Simple regression algorithms use an average of the set $N$. Another technique is to use the inverse distance weighted average of $k$-nearest neighbors.

Given a set $N$ containing $k$-nearest neighbors and a set $D$ containing distances and a positive real number $p$ we can approximate using following calculation:

$$x = \frac{\sum_{i=1}^{k} n_i d_i^{-p}}{\sum_{i=1}^{k} d_i^{-p}} \tag{1.5}$$

Constant $p$ is called power parameter. It is used to control the weight of distant points. When $p = 1$ the influence is linear. By increasing the power parameter, closer points have greater influence.

## 1.3 Vector graphics

Vector graphics use polygons and other geometric primitives to represent computer graphics. Vector graphics are built on vectors that have definite position on the $x$ and $y$ axis which determines direction or path. Additionally, each object may have assigned shape, curve, thickness, stroke color and fill.

The main advantage over raster graphics where images are represented by an array of pixels of various color is scalability. Vector images do not lose detail and quality when resized. Image size is reduced as well because the image is rendered by the viewer according to vectors.

### 1.3.1 Simple objects

Vector graphics consist of geometric primitives. Transformation can be applied on these shapes to produce more sophisticated images. Vector graphics editors usually support set operations on closed shapes (e.g. union, intersection, difference, etc.) or rotation, movement, affine transformations, etc. Most common primitive objects are:

#### Lines, polylines and polygons

In vector graphics line is simply defined by two vectors. Polyline is a sequence of vectors. Curve itself consist of the line segments connecting the vectors in order. Polygon is formed when the first and the last vector of polyline is the same.

#### Circles and ellipses

Circle and ellipse are also a common shapes used in vector graphics. A circle is defined by its center $x$ and $y$ values and radius. Ellipse needs apart from center vector a horizontal and vertical radius.

#### Bézier curves and bezigons

Bézier curves are used to produce smooth curves that can be scaled indefinitely [6]. They are defined by set of control points $P_0, P_1, ..., P_n$ where $n$ is the order. First point is the beggining of the line and last point is where the line ends. Intermediate control points are used to determine the path. Quadratic ($n = 2$) and cubic ($n = 3$) Bézier curves are the most common.

A quadratic Bézier curve in time $t$ can be traced by given points $P_0$, $P_1$ and $P_2$ using the following function.

$$B(t) = (1-t)^2 P_0 + 2(1-t)\,tP_1 + t^2 P_2, t \in [0,1] \tag{1.6}$$

Four points $P_0$, $P_1$, $P_2$ and $P_3$ define a cubic Bézier curve. $P_1$ indicates a direction in which the curve is heading after departing from $P_0$. $P_2$ indicates from which direction the curve is heading to the final point $P_3$. Curve can be traced by the following function.

$$B(t) = (1-t)^3 P_0 + 3(1-t)^2 tP_1 + 3(1-t)\,t^2 P_2 + t^3 P_3, t \in [0,1] \tag{1.7}$$

Polybezier is similar to polyline as it is a series of Bézier curves that are joined end to end. When the start of the polybezier coincides with the end it creates a bezigon. Similarly to polygon it is a set of vertices connected by Bézier curves.

### 1.3.2 Scalable Vector Graphics

Scalable Vector Graphics (SVG) is a markup language that describes two-dimensional vector graphics. It is an open standard managed by the World Wide Web Consortium (W3C) and supported on all major modern browsers. SVG 1.1 (Second edition) in the current W3C recomendation with SVG 2 currently under development.

## 1.4 Logarithmic scale

A logarithmic scale is a method to scale data that span large range of quantities. It is a non linear scale meaning that each increment does not step up an equal amount.

Instead logarithmic scale is based on orders of magnitude where every mark on the scale is the previous multiplied by a value (base of the logarithm). Transformation to log scale with base $a$ is defined using the following function:

$$f(x) = \log_a x \tag{1.8}$$

### 1.4.1 Graphical representation

Visualization of data using logarithmic scale can be useful when the data are spread across wide range of values since logarithm reduces this range to more compact and manageable size. It is also useful for the visualization of power laws because they show up as a straight line.

## 1.5 Voronoi diagram

Voronoi diagram is used to partition a plane with $n$ generating points into convex polygons [7]. Every point in that polygon is closest to its generating point than to any other. These regions are called Voronoi cells. The boundary between two adjacent cells is a line which is constructed as perpendicular bisector of the line segment connecting two generating points. Interesting property of Voronoi diagram is that its dual graph is Delaunay triangulation. Dual graph of graph $G$ is a graph that has a vertex for each face of $G$. For example of voronoi diagram see Figure 1.1.

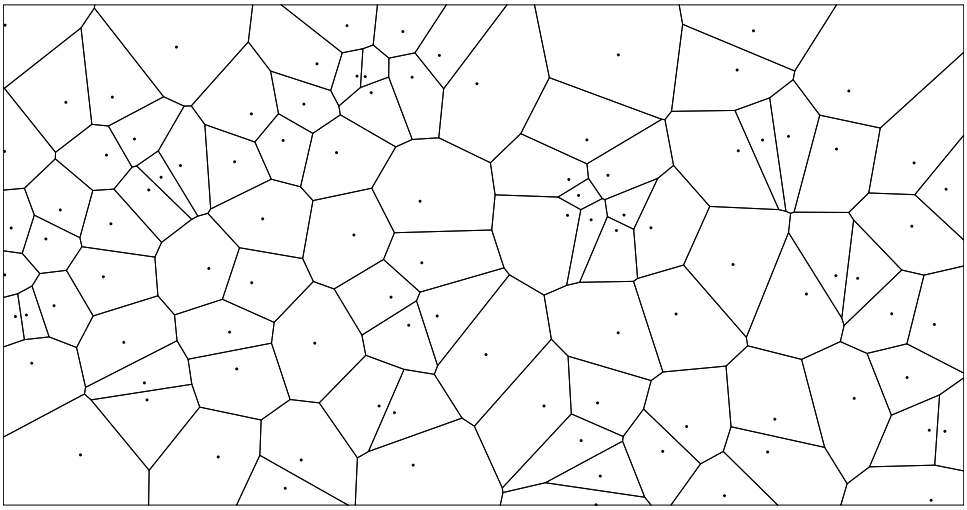


Figure 1.1: Voronoi diagram

An example of Voronoi diagram and generating points visualized using D3.js library.

### 1.5.1 Formal definition

Let there be a metric space $X$ with distance function $d$. Let $P$ be a set of generating points. The Voronoi cell $C_k$ associated to generating point $P_k$ is:

$$C_k = \{x \in X | d(x, P_k) \leq d(x, P_i) \, \forall i \neq k\} \tag{1.9}$$

### 1.5.2 Algorithms

The naive approach to the construction of Voronoi cells is to determine a region for each generating point one at a time. We can take all of the perpendicular bisectors of the segments connecting generating point to each other. We can use these rays to create half-planes and then use an $\mathcal{O}(n \log n)$ half-plane intersection algorithm. This gives us a $\mathcal{O}(n^2 \log n)$ naive algorithm.

More advanced Fortune's algorithm can generate a Voronoi diagram from a set of points using $\mathcal{O}(n \log n)$ time and $\mathcal{O}(n)$ space [8]. It is a sweep line algorithm that uses a sweep line and a beach line which both move through the plane and keep track of was seen along the way.

Horizontal sweep line moves through the plane encountering interesting points i.e. events. When sweep line encounters a new generating point it creates a parabola. Every point is the focus of its parabola and the directrix of the parabola is the sweep line itself. The set of parabolic arcs form a beach-line. Break points between adjacent arcs trace out Voronoi edges. An arc disappears whenever an empty circle touches three or more generating points and is tangent to the sweep line. Center of this circle identifies a Voronoi vertex. After sweeping the entire plane a Voronoi diagram is created.

Implementations pull events from the priority queue until empty. Current state of the Voronoi diagram such as list of the half-edges, vertices and cell records can be stored in a doubly linked list. Current state of the beach line can be stored in a binary search tree where inner nodes keep track of the break points and leaf nodes keep track of the arcs currently on the beach line.

## 1.6 Map projections

A map projection is a formal process that converts latitudes and longitudes of locations on the surface of the sphere or ellipse to a projection surface [9]. This is necessary for creating maps. All map projections create distortions in a form of false presentation of angles, shapes, distances or areas. That is because the original surface's features can never be flawlessly converted to a flat map. Formally a map projection is a mathematical function that transforms coordinates from the curved surface to the plane.

There are number of projection types each preserving one or more metric properties of the map. There can never be a perfect projection, something will always be distorted. It is important to understand these properties and choose the right one for the job. Figure 1.2 illustrates four common map projections used in this application. The most common map projection categories are:

- **Conformal:** preserves angles, projected circles are not distorted into ellipses
- **Equal-area:** preserves area, shapes are distorted
- **Equidistant:** preserves distance from some standard point or line

### 1.6.1 Geographic coordinate system

This system is used to describe any position on the Earth by a set of numbers with defined letters and symbols. Commonly used is latitude and longitude measured in degrees, minutes and seconds or decimal degrees.

Latitude ($\varphi$) is an angle that specifies the north-south position on the Earth's surface. It ranges from 0° to 90° (North or South). Lines with the same latitude are called parallels. The 0° parallel of latitude is the equator.

Longitude ($\lambda$) is an angle that specifies the east-west position on the Earth's surface. It ranges from 0° to 180° (East or West). Lines with the same longitude are called meridians. The 0° meridian which passes through the Royal Observatory in Greenwich, England is called the prime meridian.

Using the combination of latitude and longitude we can describe any position on the Earth. The grid of parallels and meridians is called the graticule.

### 1.6.2 Equirectangular projection

The Equirectangular projection is a simple equidistant projection that preserves distances along given standard parallel. The projection is neither equal area nor conformal. It is defined using the following formulas.

$$x = \lambda \cos \varphi_1 \tag{1.10}$$

$$y = \varphi \tag{1.11}$$

In these formulas $\varphi_1$ is the standard parallel (north and south of equator) where the scale projection is true. There is a special case where standard parallel is the equator ($\varphi_1 = 0°$ ) called plate carrée.

### 1.6.3 Orthographic projection

The Orthographic projection is a perspective projection where the sphere is projected onto a tangent plane. It shows a hemisphere of the Earth as it appears from the space. Shapes and areas are distorted more and more as we approach the edges. The formula for the orthographic projection is below.

$$x = R \cos \varphi \sin (\lambda - \lambda_0), \tag{1.12}$$

$$y = R \left[ \cos \varphi_0 \sin \varphi - \sin \varphi_0 \cos \varphi \cos (\lambda - \lambda_0) \right], \tag{1.13}$$

Radius of the sphere is $R$ and $(\lambda_0, \varphi_0)$ is the center point of the projection. Points on the the other side of the hemisphere should not be visible and therefore clipped. That is done by calculating the distance from the center of the orthographic projection.

### 1.6.4 Mollweide projection

The Mollweide projection is equal-area projection that preserves the proportion of the area of the ellipse between any given parallel and the equator. That is useful in maps of global distributions. The downside is that the shapes and angles are distorted. The transformation is described using the following equations:

$$x = R \frac{2\sqrt{2}}{\pi} (\lambda - \lambda_0) \cos (\theta), \tag{1.14}$$

$$y = R\sqrt{2} \sin (\theta), \tag{1.15}$$

where $R$ is the radius of the globe to be projected and $\theta$ an auxiliary angle defined by:

$$2\theta + \sin (2\theta) = \pi \sin (\varphi), \tag{1.16}$$

and $\lambda_0$ is the central meridian.

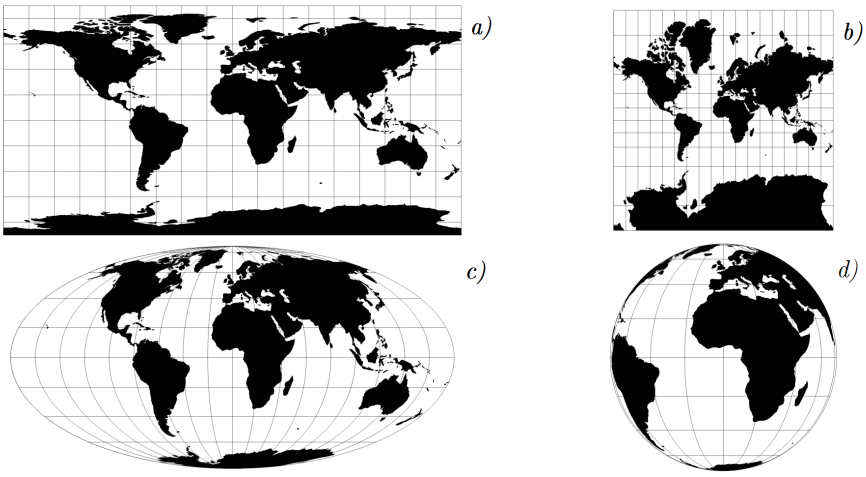



Figure 1.2: Chosen map projections

Earth map projected using Equirectangular (a), Mercator (b), Mollweide (c) and Ortographic (d) projection. The illustration was generated by this application.

### 1.6.5 Mercator projection

The Mercator projection is a conformal projection that is useful in navigation because it preserves angles. Lines of constant heading are straight lines on the map. However it distorts the size of the objects as the latitude increases. At poles the scale becomes infinite.

Variation of Mercator projection called Web Mercator is used by many major online mapping services for their map images.

Mercator projection is defined as:

$$x = R\,(\lambda - \lambda_0)\,, \tag{1.17}$$

$$y = R \ln\left[\tan\left(\frac{\pi}{4} + \frac{\varphi}{2}\right)\right], \tag{1.18}$$

where $\lambda_0$ is the central meridian and $R$ is the radius of the Earth.

## 1.7 Interpolation

Interpolation is used to estimate new data points on a function within a range of known points.

### 1.7.1 Linear interpolation

Linear interpolation (LERP) is the simplest method of interpolation. It is a process of fitting a curve using a linear polynomials to construct new data points.

Given the coordinates $(x_0, y_0)$ and $(x_1, y_1)$ and value $x \in (x_0, x_1)$ the value $y$ along the straight line is calculated by solving the following equation.

$$x = R\,(\lambda - \lambda_0) \tag{1.19}$$

The linear interpolation is used in the computer science and is easy to calculate. Interpolation between two inputs $(a, b)$ with interpolant $t \in (0, 1)$ can be easily calculated using the following method:

```
float lerp(float a, float b, float t) {
        return (1-t)*a + t*b;
}
```

Listing 1.1: Linear interpolation function

### 1.7.2 Logarithmic interpolation

Logarithmic interpolation is very similar to the linear version. Because the values are scaled using the logarithm we need to reverse it using an inverse logarithm. Then we can apply the linear interpolation as usual and scale the values back.

## 1.8 Contour plot

Contour plot is used to visually represent three-dimensional surface on a two-dimensional format. That is done by plotting contour lines using a third ($z$) variable. A common example is a contour map of geographical terrain but contours may be used to represent constant lines of other properties.

### 1.8.1 Contour line

Contour line or isoline is a curve along which a function has a constant value. In other words it is a cross-section of the three-dimensional parallel to the $x$ and $y$ plane. A contour interval is a difference of the third-dimension between adjoining contour lines.

A group of contour lines next to each other means a steep variation because the gradient of the function is always perpendicular to the contour lines.

### 1.8.2 Marching squares algorithm

Marching squares is a computer algorithm for generating contours for a contour plot [10]. It takes a rectangular array of numerical values as an input and produces the coordinates of a contour. Here are the steps:

- **Step 1:** Using a $z$ value of contour line transform an input array into the binary array where 0 represents the value below the contour line and 1 value above the contour line.

- **Step 2:** Construct a 2x2 blocks of pixels. Each block contain four binary digits. Go in a clockwise direction from upper left corner using a bitwise OR and shifting left. The result is a 4 bit binary number that corresponds to a decimal index ranging from 0 to 15.
- **Step 3:** Use that index to read from lookup table containing edges that represent the block.
- **Step 4:** Look at the original data field and use linear interpolation as described in section 1.7 to compute the exact location of the contour line.

This algorithm can be heavily parallelized because every block of 2x2 pixels can be processed independently.

Chapter 2

# Input data processing and analysis

In this chapter we will be introduced to input data and its format. We will discuss steps that were used in processing of said data and then decide what visualizations to use.

## 2.1 Input data

Each day SATRAM payload takes an approximate of 5000 measurements that are later downlinked to the Earth. A single measurement called the frame is an event when Timepix detector stores data about energetic particles that are passing through the detector at that time.

After IEAP receives relayed data from ESA's ground station network a complicated chain of processing begins. Each frame needs to be processed and evaluated. One of the last steps in this chain is when the data are in a table format where each row is a single frame and columns represent their numeric attributes. Attributes contain data such as:

- **Frame number:** Unique identification number of the frame in a single day. Some frames can be discarded in the previous data processing steps. One of the reasons can be a oversaturation of the detector.
- **Timestamp:** Timestamp in UNIX time.
- **Frame acquire time:** A time in milliseconds of how long the detector was measuring.
- **Satellite position:** A latitude, longitude and altitude of the satellite.
- **Fluxes:** Counts of the different particle types captured by the detector and measured flux.

- **Dosimetry:** Dose and dose rates for different particle types.
- **Deposited energy**
- **Angular distribution:** Angular distribution of charged particles.

From these parameters we can get more data such as ratios of a particular particle type to other or all particle types in a single frame. Particle types include Heavy charged particles (HCPs), Light charged particles (LCPs), HZP, Medium energy HCPs and SINGLE PX.

## 2.2 Table data processing

Some of the parameters like flux or dosimetry have a large range of values. That poses a problem when creating visualization because the fine details would get lost in the scale. A logarithm as described in section 1.4 is used to scale the data to the orders of magnitude. Resulting dataset is easier to read and process. When using logarithm transformation we set a special value of $-1$ when the measured value is zero to avoid computing $\log(0)$ which is undefined.

Measurements are taken on a regular basis along the path of the satellite. To fully map the entire Earth we need to combine reading from multiple days and create aggregate table data. This can be done by simply appending the data from multiple days together to a single file. In the Figure 2.1 you can see the exact locations of measurements collected in the month of August 2015. Notice the area around South Atlantic Anomaly (SAA - described in Appendix A) where the dots are sparse. That is because a lot af frames from that area were discarded due to the oversaturation of the detector.

You can also see that the points are scattered all over the map. This is not suitable for graphical visualizations so another processing procedure needs to happen. We will use a geographic binning.

### 2.2.1 Binning

With data in table format we can apply geographic binning by choosing a size of the bin in decimal degrees of latitude and longitude and then move each frame into its corresponding bin. When there is more than one row in a single bin we can apply a mean or a median to the parameters to produce a single aggregate frame for each bin. This naive approach yields a fixed number of bins with parameters and known geographic center of the bins.

The size of the bin is an important factor. When using a large bin size there is a low change of empty bins and less number of objects to draw. However the image resolution is poor and the resulting visualizations less accurate because they group larger area potentially containing an important gradient

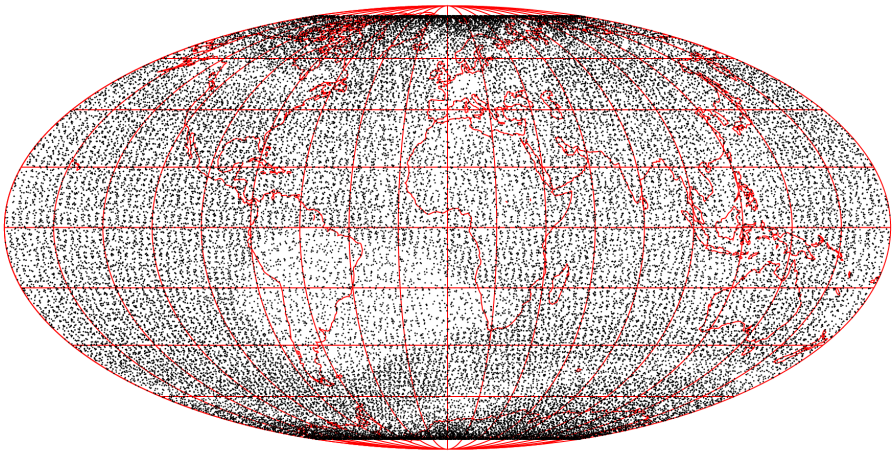


Figure 2.1: Map of the measured data frames

Map of the measured data frames collected by the SATRAM/Timepix payload during the 31 day period (August 2015). The expanded region above the SAA is shown in Figure 2.2. Projection used is Mollweide. The illustration was generated by this application.

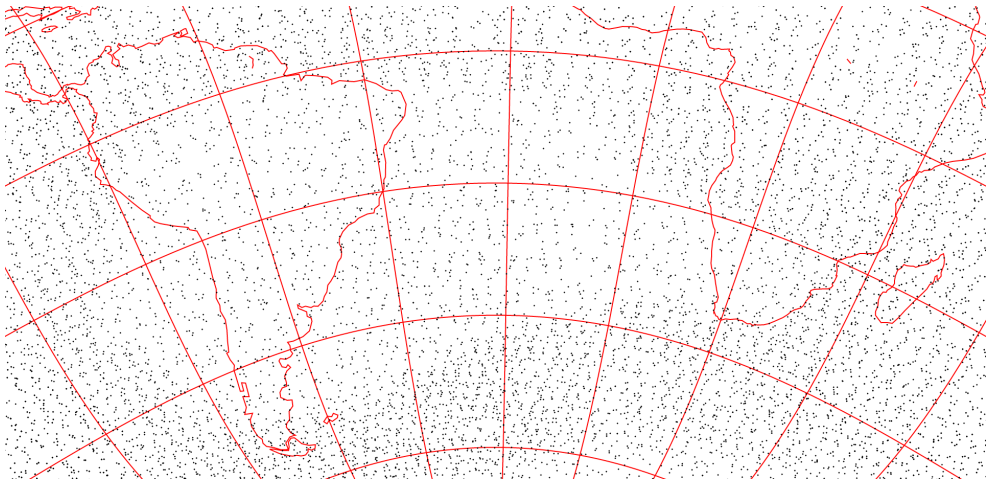


Figure 2.2: Map of the measured data frames - SAA region

that becomes less apparent. With smaller bins the image resolution improves showing more detail but the chance of a empty bins is greater and number of bins to draw significantly increases.

Typical size of a bin can range from $1° \times 1°$ to $0.1° \times 0.1°$. Binning with the size of $1° \times 1°$ produces 64 800 bins. A seemingly slight change of binning size to $0.25° \times 0.25°$ has great effect on number of output bins, in this case 1 036 800.

An easy and compact way to store the binned data is by using a text matrix composed of $x$ rows and $y$ columns where $x$ and $y$ is determined by how many bins can fit on the world map. Binning with the size of $1° \times 1°$ has matrix with 360 columns and 180 rows. Higher resolution version with the bin size of $0.25° \times 0.25°$ has matrix with 1440 columns and 720 rows.

The packing of bins into a matrix format is done by choosing a frame parameter and using the following steps:

- **Step 1:** Start at $-90°$ latitude and $-180°$ longitude as the current position.
- **Step 2:** Write the chosen parameter of the bin on the current position and the space character. Add the length of the bin to the longitude of the current position. Repeat *Step 2* until the current longitude is 180°.
- **Step 3:** Write the new line character. Add the lenght of the bin to the latitude of the current position and set the longitude of the current position to $-180°$. If the current position is 90° latitude and 180° longitude go to *Step 4* otherwise go to *Step 2*.
- **Step 4:** The matrix is generated.

The downside of the matrix format is that it only allows to store a single frame parameter. On the other hand it is easier for scientist at the IEAP to generate and work with. Modification to the matrix format can be made to store multiple parameters but we will assume from now on only matrix with single parameter. An empty bin contains a special value of $-2$.

### 2.2.2 Advanced binning

One of the flaws of the basic binning is its fixed width. Even though the decimal width is the same the actual width differs with latitude. For example the distance between two points 1° apart on the equator is 111.32 km. On the 45° N/S parallel it is 78.71 km and on the 45° N/S parallel it is 43.496 km.

This is very apparent on some map projections in higher latitude where bins become very small. A solution in form of a adaptive binning can be applied but that would compromise the matrix format we established. Alternative is to spread bins in the higher latitudes into adjacent bins in result

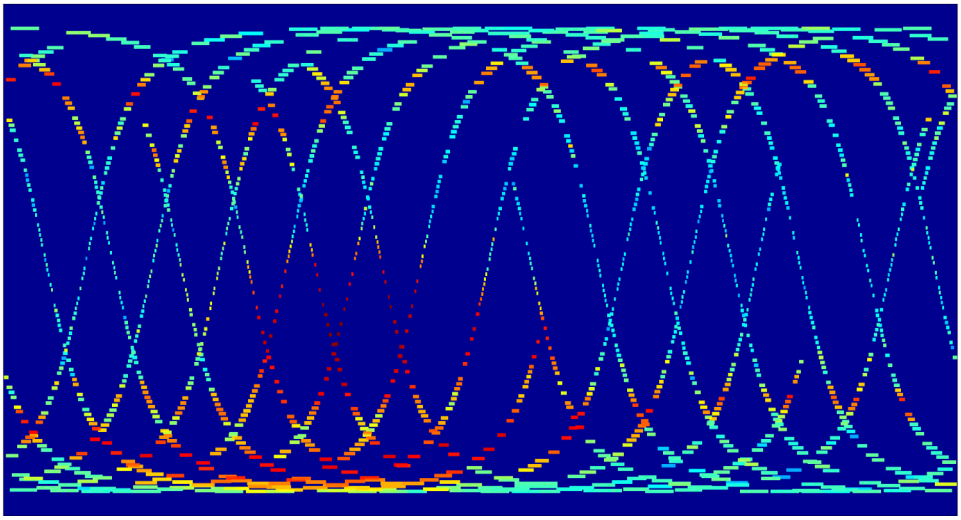


Figure 2.3: Adaptive binning

Adaptive binning of the measured data frames collected by SATRAM/Timepix payload in a single day. The illustration was generated with Matlab by the IEAP.

simulating the adaptive binning. Figure 2.3 shows the adaptive binning applied on small dataset.

## 2.3 Matrix data processing

Now, when we have a dataset to visualize, we need to focus on the next step and that is matrix data processing.

Data in matrix format are not smooth and can contain empty cells. To generate aesthetically pleasing graphical visualizations we need to eliminate the flaws in the data. If left unchanged, the resulting maps could contain gaps of undefined data and noisy values. However, there are a lot of cases where we don't need to pre-process the data, such as when we need to be as close to the true measurements as possible. Nevertheless, let's use methods described in section 1.1 to deal with the imperfections in the data.

### 2.3.1 Missing values

Empty matrix cell marked with special value of $-2$ occurs because no frames were measured in the specific area. It is to be expected because the Proba-V satellite, that is carrying the instrument, is operating in sun-synchronous orbit. That means that the satellite will not pass above the Geographic Poles. The satellite also has a 101.21 minute period [11] and can only cover small percentage of the Earth surface in a day.

That means we have to be careful when pre-processing a large number of missing values or avoid pre-processing such data because the result would be inaccurate. We also have to take in mind an area above the Geographic Poles where no data can be recorded. It would be best to discard this area from pre-processing.

After analyzing monthly data, we can safely discard the area above 81° N/S because it does not contain any readings.

Let's take a look at methods to deal with missing data as described in subsection 1.1.1 and determine what to use for our data:

- **Ignore the tuple:** We cannot ignore missing matrix cells because they contain a value of $-2$. Values are transformed from linear to logarithmic scale and a value of $-2$ would mean that the measured data was 0.01 which is not the case.

- **Manual fill-in:** What was not measured is not known and cannot be computed by hand or guessed.

- **Use global constant:** If we were to ignore the missing values and generate visualizations without pre-processing, we would have to configure the pre-processing and visualization program to recognize the special value of an empty cell.

- **Use a measure of central tendency:** Every cell in the input data matrix has a number of other cells with the same attributes. That attributes being latitude and longitude i.e. the whole row and the whole column of the matrix. These cells with common values can be used to compute mean and median to fill the missing value. This approach is not scientifically accurate because we cannot use values of distant cells. If we only want to fill the missing value for pure aesthetic purpose a median of the whole dataset would be the most appropriate because the dataset distribution is skewed.

- **Use the most probable value:** It seems that the best approach is to utilize the most probable value. Common supervised learning algorithms, such as Decision trees or Neural networks, would be for no use in this dataset because we lack suitable parameters. Therefore we should focus on the locality of the dataset and employ k-nearest neighbors algorithm. With it we can make prediction based on the neighborhood of the missing cell and make the most accurate decision. A modification in form of K-NN regression as described in section 1.2 would be the best choice, so that the points closer to the missing cell would have more influence in determining the missing value.

### 2.3.2 Data smoothing

Noise in this dataset can be attributed to errors in the data measuring or evaluation process. Errors can be easily identified while looking at the suspicious cell and comparing it with its neighborhood. When the values in the vicinity differ by the orders of magnitude or even less, we can confirm that the cell is suspected to be an error.

Let's take a look at methods to deal with noisy data as described in subsection 1.1.2 and determine what to use for our data:

- **Binning:** Data are already binned so using additional binning would reduce the resolution of the resulting visualization. It would be better to use a larger bin size in binning from table data rather than use binning on the matrix data.

- **Clustering:** Errors can occur anywhere in the data matrix and erroneous values can fall into another cluster so clustering would be unable to identify any outliers.

- **Regression:** Use of regression as described in subsection 2.3.1 can be utilized for smoothing. A prediction in a form of K-NN regression as described in section 1.2 can be made for each cell. Then using given threshold we can decide if the value is an error and replace it with the predicted value.

## 2.4 Graphical visualizations

Now that we have defined our input data, we can choose and analyze a suitable graphical visualizations. Our goal is to represent binned data on a geographic map. We have to keep in mind that the input data can be binned with different bin sizes so we have to have an universal approach.

### 2.4.1 Symbols

One option is to use a proportional symbols. This approach takes the input matrix data and draws symbols which size reflects the actual data value. There would be symbols evenly spread around the map and it would be easy to read. The downside is when there would be a large number of bins (small bin size) because the the symbols would overlap and become indistinguishable. Additionally, the symbols representing very high values could obscure other symbols.

This approach seems more suitable for showing other parameters such as ratios instead of fluxes or dosimetry, in combinations with other map types.

### 2.4.2 Graduated colors

We can represent quantities on a map using varying colors. Colors are easy to read when using shades of a single color, but color legend needs to be provided when combining multiple colors. This type of graphical visualization needs an area where to fill in the color. In our case that area is a bin.

This poses a challenge because we only have the center of a bin. To solve this we generate the Voronoi diagram as described in section 1.5 from the set of the bin centers. Using this algorithm, we have a perfectly even areas to fill.

This type of graphical visualization is ideal for the dosimetry, fluxes and deposited energy. In combinations with symbols for ratios we can generate multi-parameter maps very easily.

Chapter 3

# Application design

In this chapter we will discuss the application design and choose the right technologies for this project.

## 3.1 Visualization technologies

One of the most important step was to choose the right visualization library. The requirements were that the library has to be easily integrated into the desktop application. Additionally, it has to have a lot of built-in functionality and could generate scalable outputs. Among the candidates were:

- **HTML5 canvas:** Canvas can be used to draw draw graphics using Javascript (JS). Native HTML5 canvas has an advantage in speed but has many drawbacks as well. It would be needed to write everything from scratch. More suitable would be to use a library that is built on top of HTML5 canvas.
- **Paper.js:** Paper.js is a vector graphics scripting framework built on top of the HTML5 canvas. It is a powerful tool for scripting vector images, allowing a highly interactive visualizations. The downside is a missing support for cartography.
- **Leaflet:** Leaflet, on the other hand, is a Javascript library that specializes in interactive maps. However. it relies on online maps and cannot be used for anything other than maps.
- **D3.js:** D3.js is another Javascript library. It is used to manipulate Document Object Model (DOM) of the website to create a data-driven visualizations, mainly using the SVG element. It heavily supports cartography using custom maps and projections. The downside can be a performance limitation because drawing a large number of DOM object in a browser can be expensive.

- **Unity3D:** Unity is a game engine that uses a 3D graphics API to draw vector graphics using the power of the GPU. It is a incredibly complex but powerful tool.

I chose to work with D3.js because it is a mature library with huge support for geographic visualizations. It has a large number of tools and plugins that made the development much easier.

## 3.2 Desktop frameworks

After selecting D3.js as my visualization library of choice the path was clear. I needed to find a desktop framework that could build an application using the web technologies such as HTML, JS and CSS. At the time of the writing there were two main competitors.

### 3.2.1 Electron

Electron is a framework that is used for the development of desktop GUI applications. It is open source, cross platform, developed by GitHub and uses web technologies such as HTML, CSS and Javascript with Chromium and Node.js to build the applications. One of its features is support for automatic updates, crash reporting, Windows installers, modern debugging tools and advanced integration with target platform allowing native menus, notifications and more. It also supports native Node modules.

Although it is a relatively young framework (2014), it was used to build large scale projects such as Slack, Atom or Microsoft's Visual Studio Code.

### 3.2.2 NW.js

NW.js is a framework for building desktop GUI applications using web technologies on top of Chromium and Node.js. Like Electron, it is open source and cross platform framework. It is a older, more mature project sponsored by Intel with many features of Electron.

One of the advantage of NW.js is its support for Windows XP or more recent Node.js version.

### 3.2.3 Decision

These two frameworks have very similar features and the decision comes to the personal preference. With Electron having more momentum and community support, I have decided to use it for this application.

## 3.3 Front-end technologies

Web development and design is focused on tools and libraries that assist the developer. It is because writing a modern responsive website from scratch became very complicated these days. Electron uses HTML and CSS just like any other website so we have a great choice of front-end libraries to use. I was choosing from the following libraries.

- **React:** React is an open-source Javascript library for building user interfaces from Facebook. It is used as the View in Model-View-Controller (MVC) architecture. It abstracts the Document Object Model (DOM) enabling simpler programming model and better performance.
- **Angular:** Angular is an open-source web application framework from Google. It provides framework for the client-side MVC architecture. One of its goals is to separate the DOM manipulation from the application logic and simplify the development and testing of single-page applications.
- **Bootstrap:** Bootstrap is an open-source front-end library. It is a collection of tools, CSS templates and Javascript extensions that assist in development of dynamic websites and web applications. One of the main benefits is the ease of use. Developers can easily create nice dynamic websites just by applying styles from the library to the elements.

This application did not need any advanced libraries or frameworks. It just needed a simple tool to wrap the visualizations in nice UI that would be responsive on multiple screen sizes and a set of components like buttons or input fields that are ready to use. Bootstrap managed that and that is why I chose it for this application.

## 3.4 Source code hosting services

The most logical step for this application was to open-source it. This way anyone can use and edit the application or look at the source code. I was already using Git as a version control system on my machine. This meant uploading the source code to some public repository. I was deciding between the following hosting services.

- **Bitbucket:** Bitbucket is a web-based hosting service for Mercurial or Git. It is mainly team-oriented with its private repositories restricted to the team members. However, it can also host public repositories with powerful collaboration tools and many integrations. Bitbucket can be used in the cloud or hosted on a server and is priced by the number of team members with public repositories being free.

- **Gitlab:** Gitlab is another web-based hosting service for Git with wiki and issue tracking capabilities. Users can create repositories in the cloud or host the open-source Community Edition on their own server for free. Faculty of Information Technology hosts their own Gitlab hosting service available for their students.

- **Github:** GitHub is the most popular web-based hosting service for Git with many features including access control, bug tracking, feature request, wikis, release hosting and more. Hosting public repositories is free.

For hosting of this application I wanted a trusted and tested solution. That is why I went with Github because it allowed me to have a full-featured public repository with powerful integration to the existing Continuous Integration (CI) services. I was also able to host the releases of the application and manage a landing page for the project.

## 3.5 Continuous integration software

With this application I wanted to incorporate a continuous integration techniques. That would allow me to push a code revision which would start a build process that would produce the application executable. Because of that, I needed a CI software that could integrate with the hosting service and do the application build in the cloud. I was choosing between the following software.

- **Jenkins:** Jenkins is an open-source continuous integration server that provides many plugins to support building, deploying and automating any project. It is written in Java and runs in a servlet container. The build can be started automatically when a code revision is published to the hosting service or manually by HTTP request. The build process is in the hands of the developer who has to script the entire build pipeline. FIT CTU hosts their own Jenkins server available for their students.

- **Travis CI:** Travis CI is a hosted continuous integration service used to build and test projects hosted on GitHub. When the code is pushed to GitHub it is automatically built in the cloud and the application can be deployed anywhere. Additionally, it is free to open-source projects but only supports builds on Linux.

- **AppVeyor:** AppVeyor is another hosted continuous integration service for projects on GitHub. It has the similar features of Travis CI but is used to build on Windows platform. Just like Travis it is free to use for open-source projects.

I needed a simple CI that would integrate with my source code hosting service of choice GitHub and could build on Windows and Linux. I also did not want to operate any servers so building in cloud was mandatory. Beacuse of that, I have decided to use both Travis CI and AppVeyor beacuse they are both free, easy to set up and integrate with GitHub.

## 3.6 User documentation software

One of the tasks was to create a user documentation for the application. Because it is a technical project, I did not want to use text editors like Word but something advanced in the combination with version control system that could produce a web page or PDF document. There were a couple of options.

- **LaTeX:** LaTeX is a document preparation system used for communication and publication of scientific documents. It uses plain text and markup tagging conventions to produce an output file such as PDF. LaTeX source files can be version controlled.
- **DocBook:** DocBook is a schema (avaliable in laguages such as XML) that is suited for books and papers about computer hardware and software but can be used for anything else. It enables to create document content in neutral (plain text) form that defines the logical structure of the document. Using a stylesheet, this content can be then published in variety of formats such as HTML, EPUB, PDF, etc. without making any changes to the source.
- **Gitbook:** Gitbook is an open-source book format, toolchain and online platform for writing and hosting books. It supports writing books in Markdown and AsciiDoc and fully embraces Git version control. Books can be created using their desktop editor and hosted on their site or generated into PDF, ePub or Mobi.

LaTeX and Gitbooks proved to be a viable solutions but Gitbooks won with its build-in hosting and easier editing in Markdown in their desktop application. DocBook is a professional and powerful solution but too complicated for a few pages of this documentation.

Chapter 4

# Application implementation

This chapter follows an implementation of this application. We will begin by introducing the application architecture and then describe technical details of this implementation.

This application is called SatGraph because it generates graphical visualizations and is available for Linux and Windows platform. The project is located on GitHub at `https://github.com/OndrejBilek/satgraph`.

## 4.1 Application architecture

This desktop application wraps around the entire process of the graphical visualization generation. It consist of three main modules that are invisible to the user because they cooperate seamlessly. Processing module takes input files from the user, converts them and applies pre-processing. Visualization module generates vector representation of the graphical visualizations and the resulting SVG image. Display module takes care of the user interface. Please refer to Figure 4.1 that shows the simple view of the application architecture. We will now closely introduce the modules.

## 4.2 Display module

This application begins and ends with the display module. It is used to operate the user interface and then to display the visualization before exporting them. Because we are using Electron framework, which is the desktop wrapper for the Chromium browser, the entire application user interface is written in web technologies such as HTML, CSS and Javascript.

### 4.2.1 Electron

The entry point for the Electron is a Javascript file, in our case main.js, which spawns the main process. The script that runs in the main process can

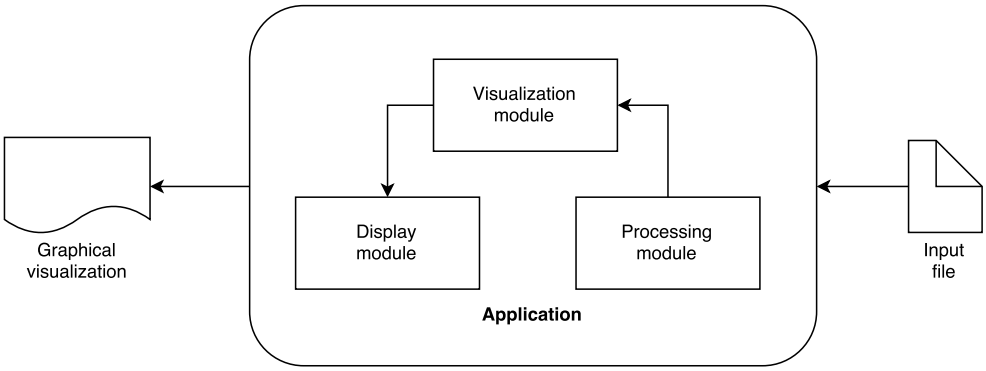



Figure 4.1: Application architecture

display GUI by creating web pages or dialogs. Since Electron uses Chromium for displaying web pages, it also uses Chromium's multi-process architecture. That means that each web page in Electron runs in its own process called the renderer process. Renderer processes are isolated and only cares about the web page running in it. When the web page is closed the renderer process is destroyed. Calling native GUI related APIs is not allowed in the renderer process because it is dangerous and can leak resources and needs to be done in the main process [12].

If the renderer processes wish to communicate with the main process and vice versa an inter-process communication needs to be used. Electron's IPC module handles asynchronous and synchronous messages while Remote module enables remote procedure calls.

Our main module first handles the launch parameters mentioned in subsection 4.5.2 and then spawns a renderer process with the main HTML file.

### 4.2.2 User interface

The main HTML5 file is a common website served with local resources and does not connect to the Internet. It first loads the Bootstrap CSS library that is used to for the user interface and then my own CSS file. Javascript files with the main source are loaded right after that and then finally the HTML file is rendered.

The application interface consist of two main panels and a viewing area. These panels are contained in Bootstrap's fluid containers and adapt to the current resolution. Both panels are used for operating the application with its input components that are styled by the Bootstrap library and bound to the Javascript file. The viewing area is styled using a flex property and fills the rest of the viewport. The contents of the viewing area is empty at first.

When the HTML file finishes loading a Javascript function is called. It is used to initialize the application before it can process any inputs. The function

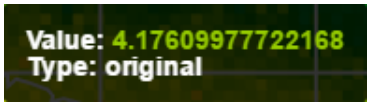



Figure 4.2: Live cursor

first initializes validation listeners that react on input and check whether the input values are valid. When everything is valid and input files entered the generate button is enabled. After validation listeners an SVG element that fills the entire viewing area is created with the D3 library. Visualization module then loads a JSON file containing the Earth map and displays it with a graticule using the chosen map projection. Users can adjust the Earth map, map projection and pre-processing parameters then load the input data and generate the visualization.

### 4.2.3 Exporting

Resulting visualization can be exported to SVG and PNG. Since the SVG source is generated by the visualization module the export process just consist of retrieving outer HTML of the SVG element. However, since the elements are styled using an external CSS file and SVG does not allow external resources, the style sheet need to be embedded inside the SVG file.

PNG export is a bit more complicated because it involves making use of browser's rasterisation and is done using a save-svg-as-png library.

### 4.2.4 Map adjustment

The viewing area is responsive to mouse events like drag or zoom. That is used to adjust the map by panning it around or zooming to a specific location. It is implemented using the D3.js events library that return a scale value for zooming event and rotation vector for drag events. These values are passed to the visualization module to re-render the image.

### 4.2.5 Live cursor

This application also has a live cursor feature on the resulting visualization. When hovering over a data point it displays a small overlay with information like value or origin of the data. That is done by retrieving the on-screen position and the index of the point and drawing a fixed position semi-transparent division (DIV). Example of the live cursor can be seen in Figure 4.2.

## 4.3 Processing module

Processing module handles input files that are converted, processed and passed to the visualization module. The biggest concern with implementing this module in Javascript was that the Javascript is lacking performance and would slow down the application. Because of that I decided to implement the processing module with Node.js Addons.

### 4.3.1 Node.js Addons

Node.js Addons are dynamically-linked shared objects written in C or C++ that can be loaded into Node.js and used as a regular modules [13]. Node.js uses a Chrome V8 engine to provide the Javascript implementation so all Addons use V8 API for mechanism like creating objects, calling functions, etc.

When the source code has been written it needs to be compiled into binary file that can be included as a module. This is done by creating a Node.js specific configuration file and using a node-gyp tool written to specifically compile Node.js Addons. This tools is cross-platform enabling to build on Windows or Linux.

### 4.3.2 Processing Addon

Node.js Addon that implements the processing for this application is a C++ class. It is run with a Javascript function call with the first parameter being the path to the input file and the second a Javascript object with configuration.

When the function is called a V8 API is used to extract the parameters of the function call and convert them to C++ data types. An instance of processing class is created and parameters passed to the constructor and saved. The processing is then controlled by the parameters passed in the function call.

First the Addon locates the input file in the matrix format and converts it into a 2D array. Pre-processing and data smoothing via a K-NN regression as described in section 1.2 is done by traversing the the neighborhood of the cell until the specified number of neighbors is found. The predicted value is then calculated and information about the origin of the cell saved.

After finishing processing the 2D array every cell of that matrix is converted into a Javascript object using a V8 API that contains following parameters.

- **Latitude and longitude:** position of the cell in decimal degrees in the geographic coordinate system
- **Value:** floating point number representing the value of the cell
- **Origin:** number representing whether the cell value is original or was smoothed or imputed

Figure 4.3: Chosen color scale

Assigning location to each cell is easy because we know the dimensions of the matrix and that it starts at the $-180°$ of longitude and $-90°$ of latitude. Each step in latitude and longitude is calculated by dividing the the number of degrees in one direction with the size of the matrix in that dimension.

Each Javascript objects is inserted into a Javascript array and passed back by the Addon as a return value of the function call.

## 4.4 Visualization module

With data returned by the processing module we can begin creating the visualizations. The data are in an array of objects that contain precise coordinates. First we have to convert these data points to polygons representing the area of the bin.

### 4.4.1 Voronoi diagram and color scale

That is done using a Voronoi algorithm described in subsection 1.5.2. We use an implementation of Fortune's algorithm provided by the D3.js library that returns an array of objects containing coordinates of the polygons.

Next step is to fill the areas with appropriate colors based on the value. That is done using a D3.js scale function. It has to be configured by specifying the range of the scale in color values and their corresponding domain. The function can be called by passing a value in the range and it returns a linearly interpolated color. This application uses a color scale consisting of cyan, green, yellow, red and purple as seen in Figure 4.3.

### 4.4.2 Projecting and drawing

Before drawing the map we have to convert these polygons to the target map projection. First we retrieve the scale and the origin of the map projection from the display module and apply the transformation as described in section 1.6 and implemented in D3.js library.

Drawing the visualization means converting the polygons into the SVG DOM format. It is done by appending to the DOM of the SVG element assisted by D3.js library. This is the slowest part of the application since the browser must parse and display thousands of elements. On small datasets running time of generating procedure is in seconds and can take up to minutes

with large resolution matrix. In my testing the application became unstable and ran out of 8GB of memory approaching two million elements. This meant that the largest stable matrix resolution is 1440x720.

Visualization module also supports drawing multi-parameter symbols. The data are in the same format as those used as an input to the Voronoi algorithm. We take them a draw an empty circle with with the center at the position of the bin and radius linearly interpolated from the value. Positions where the frame parameter was measured to be zero is indicated by the smallest black filled circle.

Contour lines can also be enabled in the user interface and are drawn on top of the map with isoband width of one magnitude using the isoband library that uses the Marching squares algorithm described in subsection 1.8.2. It uses the input in matrix format to produce the polygons of individual isobands that are later projected onto the Earth map.

### 4.4.3 Histogram

This application can also draw histograms from another dataset. This dataset contains time-value pairs that can be visualized on a simple graph. Histogram is implemented by defining $x$ and $y$ axis and creating a line connecting the points where the value is represented on the graph and filling the area under the graph. Zooming and panning on the graph is implemented as well.

### 4.4.4 Map legend

Map legend is essential to illustrate the meaning of the symbols and colors appearing on the map. It is implemented using the d3-svg-legend library that extends D3.js scale function. We can define the position, main label, format of the sub-labels and what symbols to show. The SVG layer with legend is generated and corresponds to the map.

## 4.5 Continuous integration

For this application I have implemented a continuous integration process. This meant remote building of the application to verify it works on both platforms and distribution of the executable.

### 4.5.1 Build

For building the application I have used Travis CI for Linux and AppVeyor for Windows. This meant setting up hooks and configuration files. For both continuous integration services the process was the same. When new revision was pushed to the repository a build was triggered. Both services first cloned the source from the GitHub repository and then installed required packages

specified in the configuration file. Next they executed scripts defined in the configuration file which ran the build process. If every process exited with 0 the build was a success otherwise it failed and the developer was notified.

Build status of both continuous integration services is available from the project GitHub page.

### 4.5.2 Distribution

When the build process is successful the built application called artifact is left on the file system of the virtual machine. This artifact can be optionally deployed anywhere.

For distribution of application executables I used GitHub releases. Developer can create releases from the current state of the source code. Releasing new version triggers build in the continuous integration services and the application is built and selected artifact attached to the release page. This process is fully automatic and saves a lot of time.

The produced executable contains a build-in installer that unpacks the application and runs it with special parameter indicating that the application was just installed. It is up to the developer to react to these events. This application registers the start menu shortcut and creates a desktop shortcut on installation a removes it when the application is uninstalled.

## 4.6 User documentation

User documentation is written using Markdown and hosted on GitBook. Their desktop editor was easy to use with version control implemented as well. When the new revision of the user documentation was published it triggered a GitBook build that produced the documentation in HTML, PDF, ePub and Mobi. User documentation is linked in Appendix C

Chapter 5

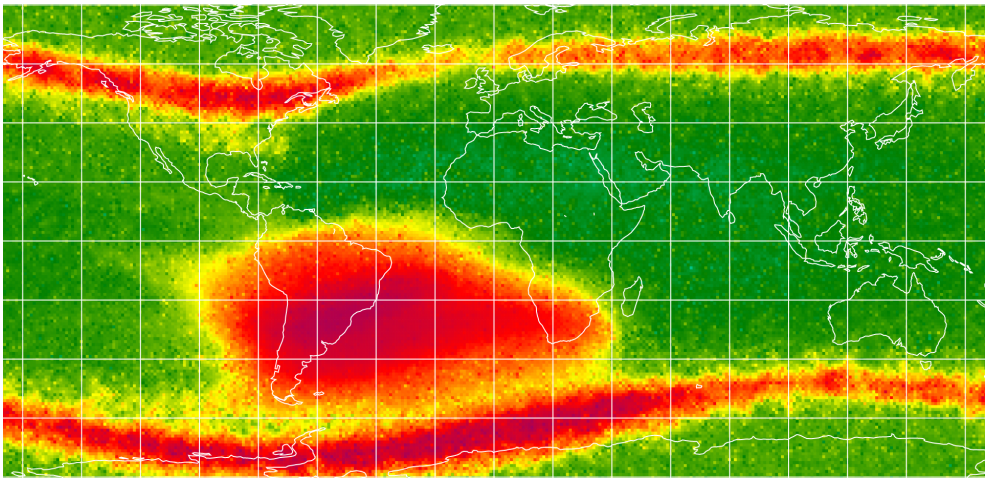
# Graphical visualization results

In this chapter we will present the resulting visualizations from this application. There are many variants of generated images but we will try to include only the most important ones. In case of pre-processing if not stated otherwise a 8-NN regression with power parameter $p = 1$ and smoothing threshold $t = 1$ was used. The default bin size in binning is $1^\circ \times 1^\circ$.

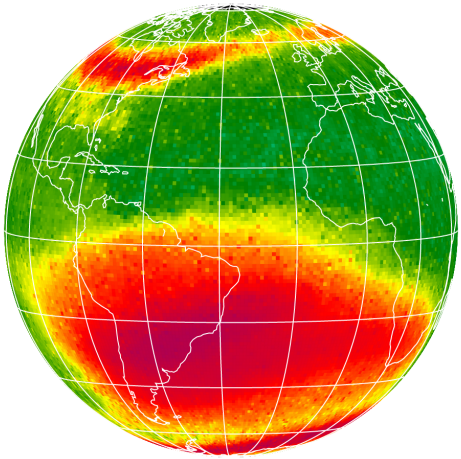
All of the images presented in this chapter and more are included on the enclosed CD.

## 5.1 Map projections

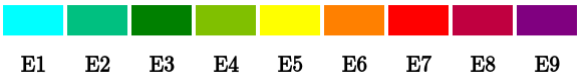
We have decided to use Mollweide projection due to its accuracy of proportions in area for the majority of visualizations. Because of that the following Figure 5.1 shows the other projection types.



Equirectangular projection



Orthographic projection




Dose rate [log nGy/h]

Figure 5.1: Dose rate map – projection types

Map of the dose rate (all particles) measured by the SATRAM/Timepix payload during the 30 day period (November 2015) with imputed and smoothed data. Map is generated using the Equirectangular and Orthographic projection by this application.

## 5.2 Imputation and smoothing

In the following Figure 5.2 we have the original binned dataset and pre-processed variant to correct for missing values.

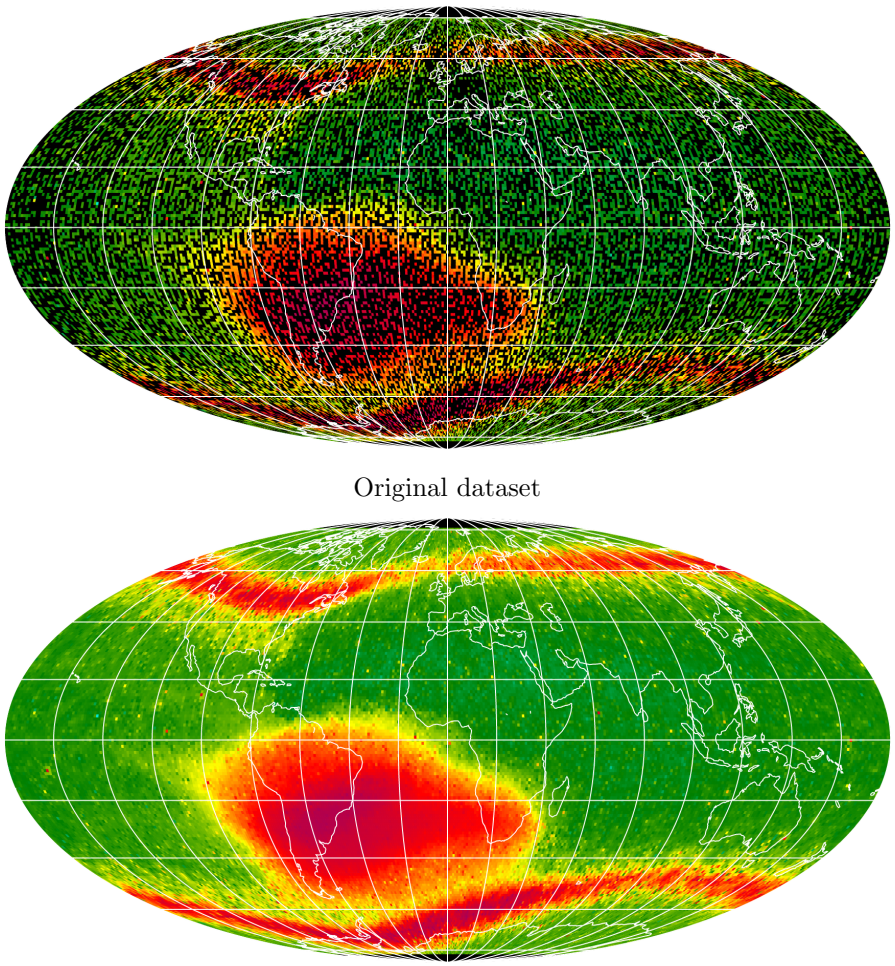
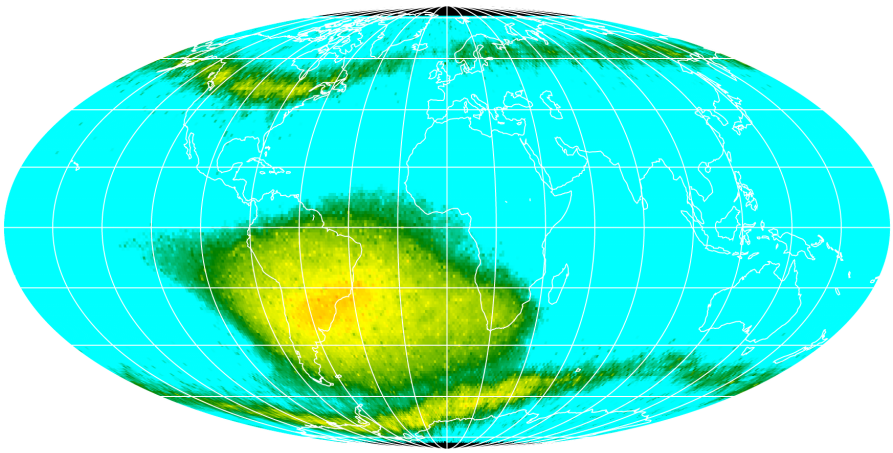



Figure 5.2: Dose rate map – pre-processing

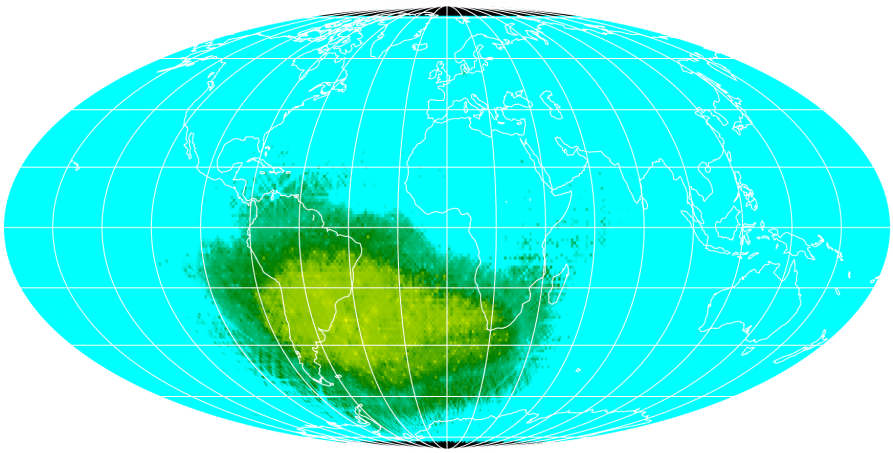
Map of the dose rate (all particles) measured by the SATRAM/Timepix payload during the 30 day period (November 2015). Map is generated using the Mollweide projection by this application.

## 5.3 Particle flux

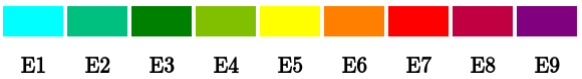
In the following Figure 5.3 we have the maps of particle flux count with different particle types.



Heavy charged particles (HCPs)



Medium energy HCPs




Flux [log count/$cm^2$/min]

Figure 5.3: Particle flux map – particle types

Map of the flux count measured by the SATRAM/Timepix payload during the 30 day period (November 2015) with imputed and smoothed data. Map is generated using the Mollweide projection by this application.

## 5.4 Multi-parameter map

In the following Figure 5.4 we have the multi-parameter map that displays particle count ratios on top of the particle flux map. We have chosen a region of the South Atlantic Anomaly (SAA) to demonstrate varying ratios.

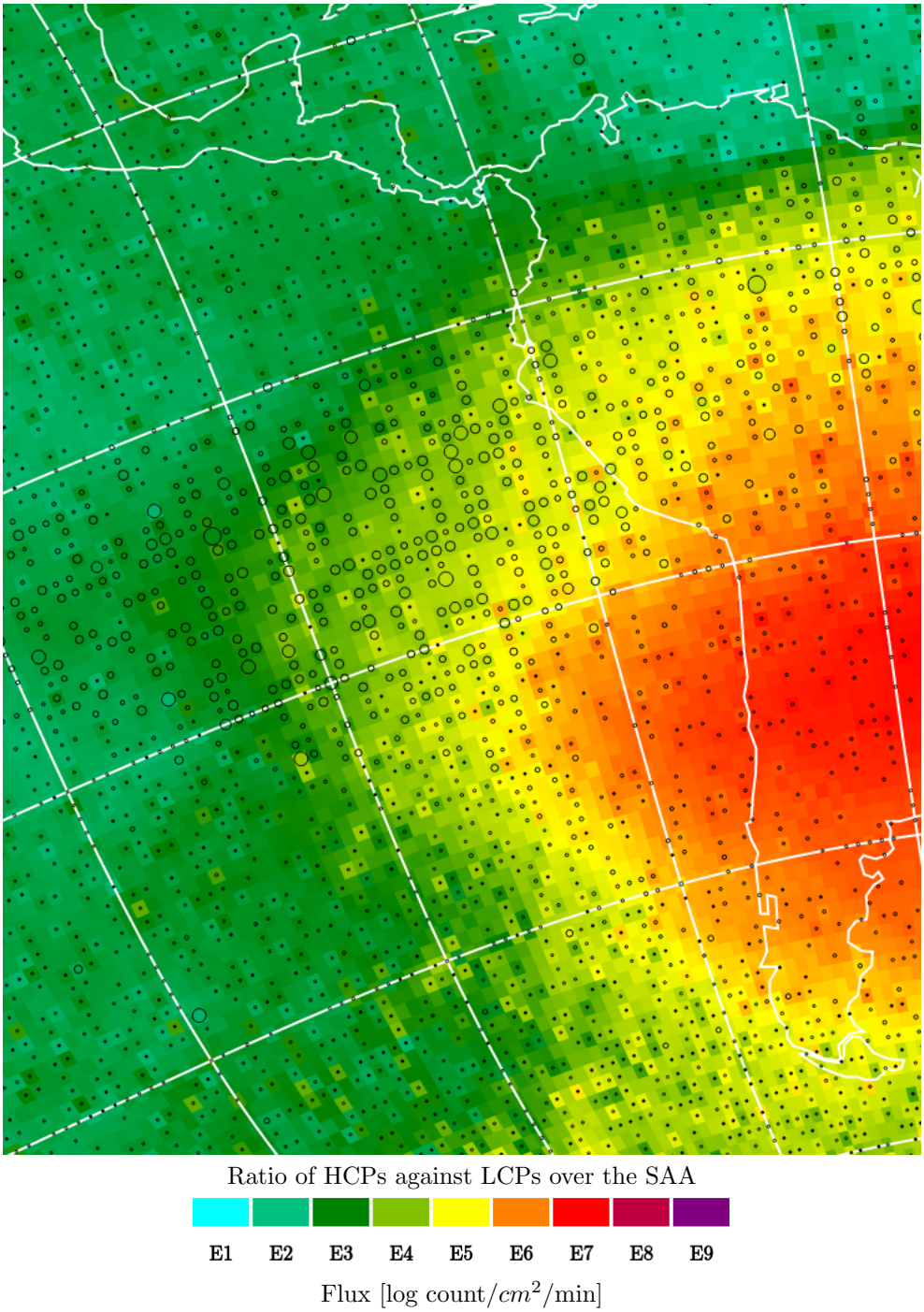



Figure 5.4: Multi-parameter particle flux map

Multi-parameter map of the fluxes (all particles) and particle count ratios measured by the SATRAM/Timepix payload during the 30 day period (November 2015) with imputed and smoothed data. Size of the circle represents ratio of the particle type. Map is centered above the South Atlantic Anomaly (SAA) and generated using the Mollweide projection by this application.

## 5.5 Contour map

In the following Figure 5.5 we have generated a layer of contour lines above the particle flux map near the SAA region. The contour interval is an order of magnitude.

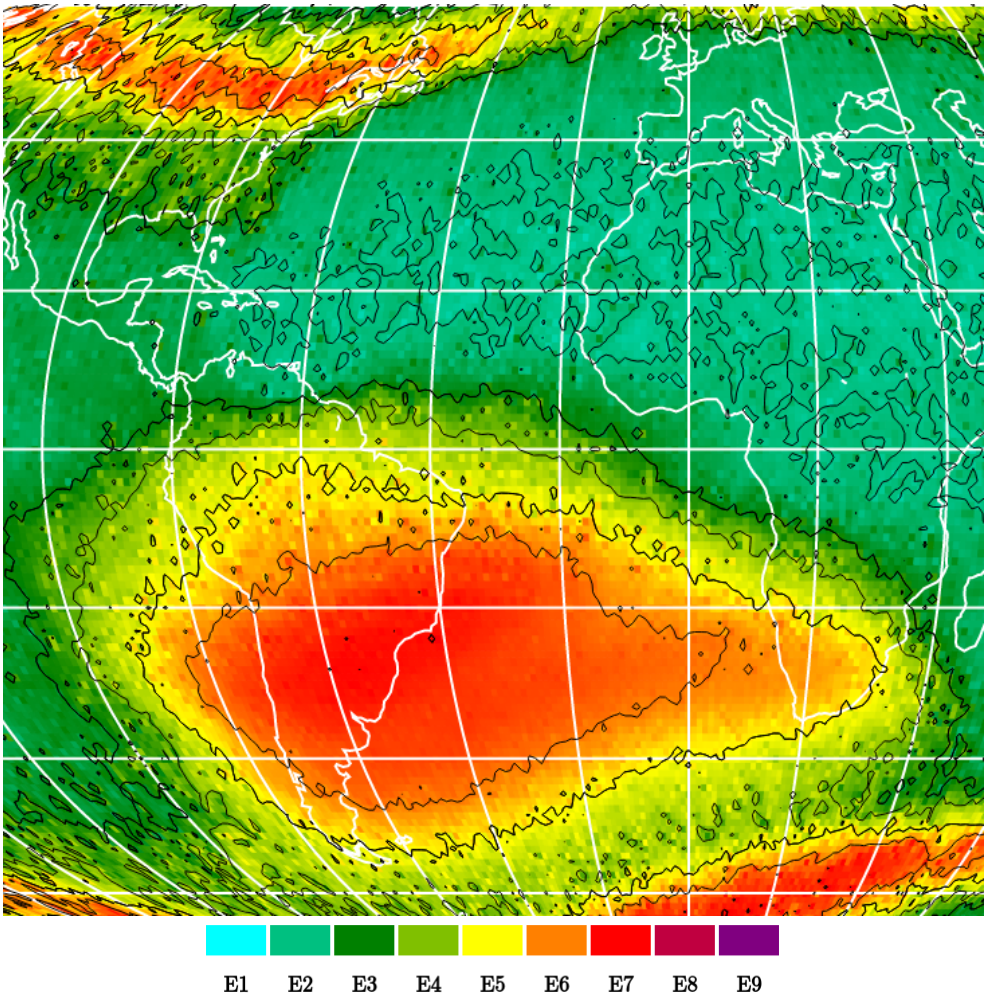



Figure 5.5: Contour particle flux map

Contour lines on top of the flux map (all particles) measured by the SATRAM/Timepix payload during the 30 day period (November 2015) with imputed and smoothed data. Map is generated using the Mollweide by this application.

## 5.6 Adaptive binning

In the following Figure 5.6 we have a detail of particle flux map that was processed using adaptive binning with a bin size of $0.5° \times 0.5°$ without additional pre-processing. We can see that bins in latitudes approaching the Geographic poles appears to have the same width.

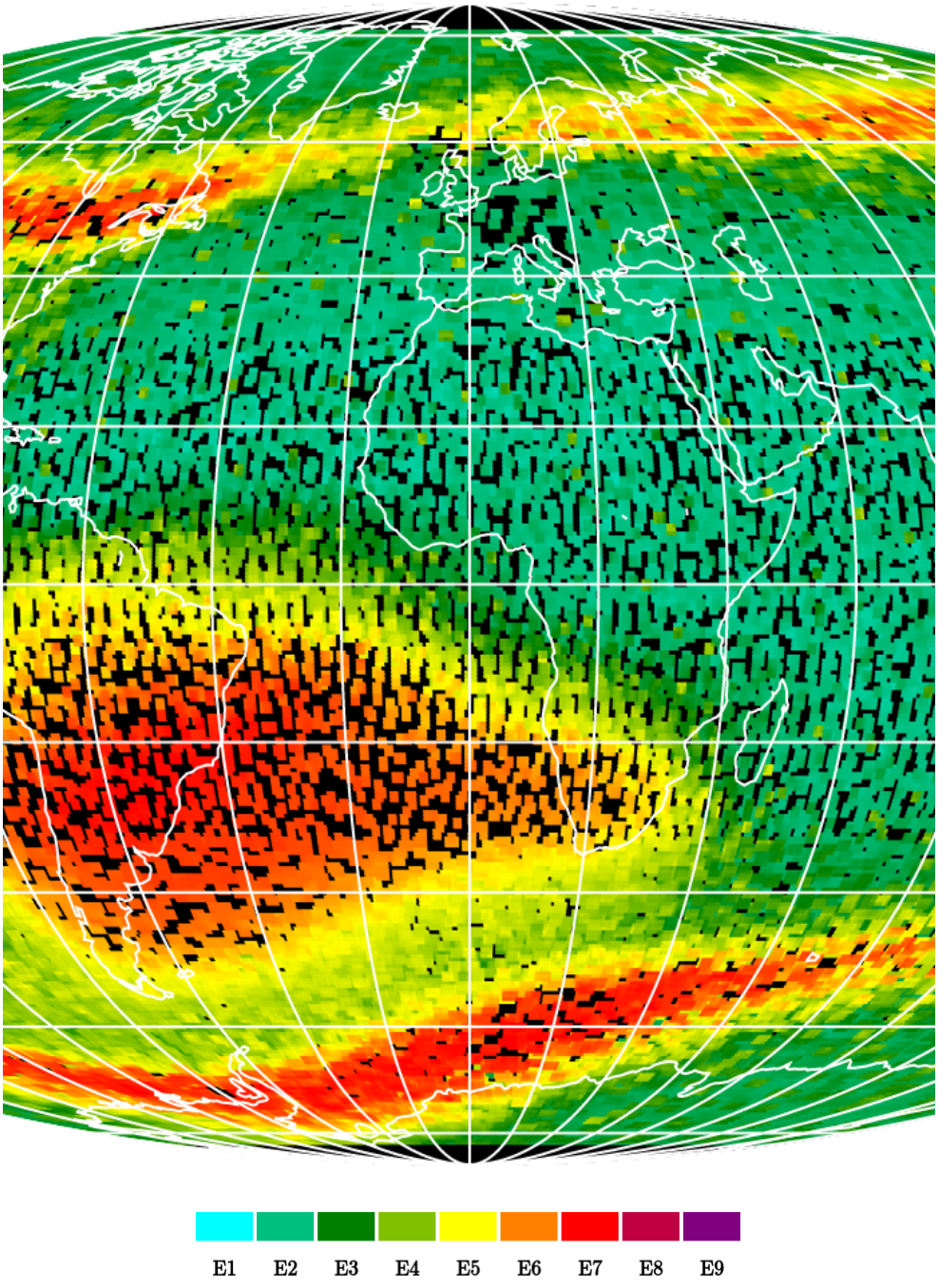



Figure 5.6: Particle flux map – adaptive binning

Particle flux map utilizing adaptive binning and bin size of $0.5° \times 0.5°$ with data measured by the SATRAM/Timepix payload during the 30 day period (November 2015). Map is generated using the Mollweide projection by this application.

## 5.7 Histogram

In the following Figure 5.7 we have the histogram of time-value dataset. The histogram spans 24 days so we show two close-ups of a single day.

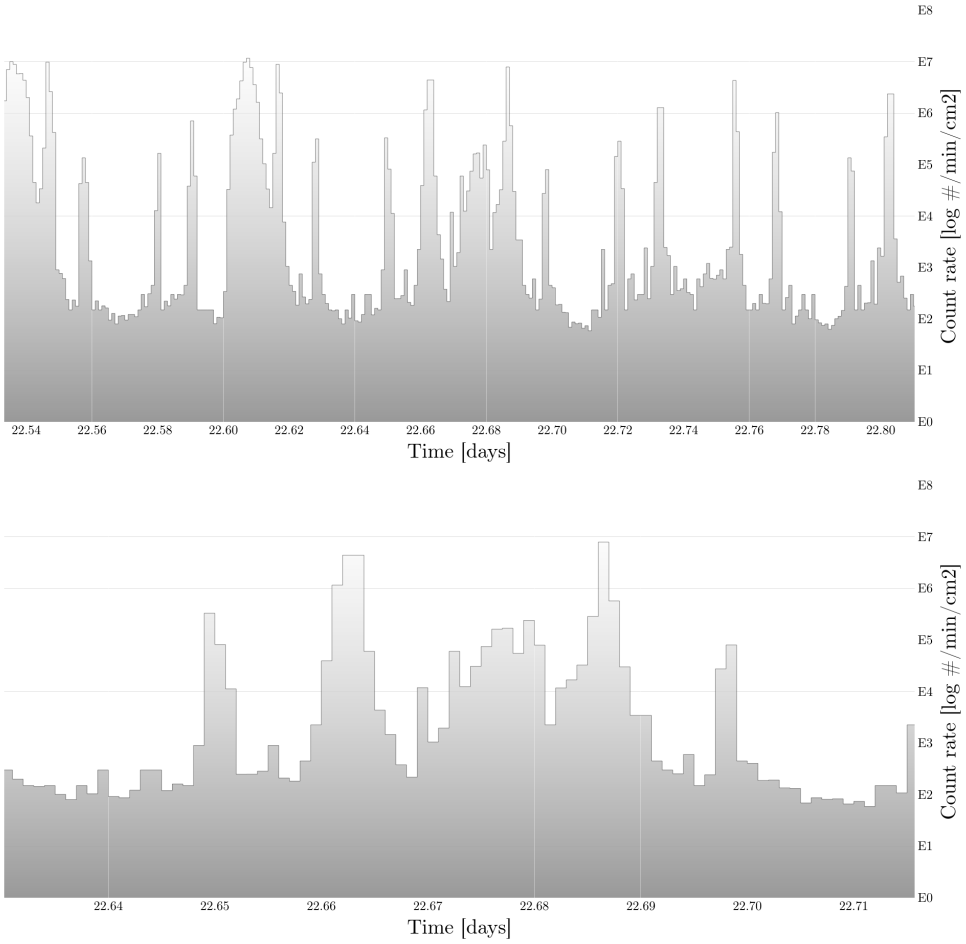



Figure 5.7: Histogram

Histogram of the count rate measured by the SATRAM/Timepix payload during the 24 day period (September 2015) generated by this application.

# Conclusion

In conclusion the assignment was fulfilled because all of the goals were met. We managed to implement multi-platform desktop application that was able to generate graphical display of evaluated data of the spacecraft payload SATRAM/Timepix.

In the beginning we have researched and defined important methods and theory such as data pre-processing or vector graphics that helped us to tackle the visualization goal. We also learned about map projections and their role in the construction of the Earth maps.

Next we were introduced to the input data format and many physical properties that it contains and used binning with multiple bin sizes to define the new data format in the matrix form. In this procedure we have applied logarithmic scaling and two types of binning including the adaptive bin sizing to offset the diminishing size of the bin with the increased latitude.

After evaluating different types of graphical visualizations we have designed and implemented desktop application that uses the dataset in the matrix format to generate graphical visualization in a form of single and multiple parameter Earth maps with the appropriate color scale on various map projections. That is done utilizing voronoi diagrams and D3.js library running inside the Electron framework. The application is also able to apply missing value imputation and smoothing using the K-NN regression and produce more types of graphical visualizations like contour maps or histogram of time-value dataset. Resulting graphical visualizations can be exported to vector SVG format or bitmap PNG format.

The application is open-source and extensible incorporating continuous integration and providing a user documentation. It will assist experts at the IEAP with their research on space radiation and can be expanded with more visualization types, input data or other tools.

# Bibliography


[1] Granja, C.; Polansky, S.; Vykydal, Z.; et al. The SATRAM Timepix spacecraft payload in open space on board the Proba-V satellite for wide range radiation monitoring in LEO orbit. *Planetary and Space Science*, 2016: pp. –, ISSN 0032-0633, doi:http://dx.doi.org/10.1016/j.pss.2016.03.009.

[2] Han, J.; Pei, J.; Kamber, M. *Data Mining: Concepts and Techniques*. The Morgan Kaufmann Series in Data Management Systems, Elsevier Science, 2011, ISBN 9780123814807.

[3] García, S.; Luengo, J.; Herrera, F. *Data Preprocessing in Data Mining*. Intelligent Systems Reference Library, Springer International Publishing, 2014, ISBN 9783319102474.

[4] Eriksson, L.; Byrne, T.; Johansson, E.; et al. *Multi- and Megavariate Data Analysis Basic Principles and Applications:*. MKS Umetrics, 2013, ISBN 9789197373050.

[5] James, G.; Witten, D.; Hastie, T.; et al. *An Introduction to Statistical Learning: with Applications in R*. Springer Texts in Statistics, Springer New York, 2013, ISBN 9781461471387.

[6] Buss, S. *3D Computer Graphics: A Mathematical Introduction with OpenGL*. 3-D Computer Graphics: A Mathematical Introduction with OpenGL, Cambridge University Press, 2003, ISBN 9780521821032.

[7] Okabe, A.; Boots, B.; Sugihara, K.; et al. *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*. Wiley Series in Probability and Statistics, Wiley, 2009, ISBN 9780470317853.

[8] Fortune, S. A sweepline algorithm for Voronoi diagrams. *Algorithmica*, volume 2, no. 1, 1986: pp. 153–174, ISSN 1432-0541, doi:10.1007/BF01840357.

[9] Snyder, J. *Flattening the Earth: Two Thousand Years of Map Projections*. University of Chicago Press, 1997, ISBN 9780226767475.

[10] Maple, C. Geometric design and space planning using the marching squares and marching cube algorithms. In *Geometric Modeling and Graphics, 2003. Proceedings. 2003 International Conference on*, July 2003, pp. 90–95, doi:10.1109/GMAG.2003.1219671.

[11] N2YO. PROBA V Satellite details 2013-021A NORAD 39159. 2013, [online]. [Cited 2016-04-20]. Available from: `http://www.n2yo.com/satellite/?s=39159`

[12] GitHub. Electron documentation: Renderer process. 2016, [online]. [Cited 2016-05-10]. Available from: `http://electron.atom.io/docs/latest/tutorial/quick-start`

[13] Node.js. Node.js documentation: Addons. 2016, [online]. [Cited 2016-05-10]. Available from: `https://nodejs.org/api/addons.html`

[14] Bourdarie, S.; Xapsos, M. The Near-Earth Space Radiation Environment. *IEEE Transactions on Nuclear Science*, volume 55, no. 4, Aug 2008: pp. 1810–1832, ISSN 0018-9499, doi:10.1109/TNS.2008.2001409.

[15] Reames, D. V. The Two Sources of Solar Energetic Particles. *Space Science Reviews*, volume 175, no. 1, 2013: pp. 53–92, ISSN 1572-9672, doi:10.1007/s11214-013-9958-9.

[16] Tripathi, R. K. Radiation Effects In Space. *AIP Conference Proceedings*, volume 1336, no. 1, 2011: pp. 649–654, doi:http://dx.doi.org/10.1063/1.3586182.

[17] Adams, J.; Falconer, D.; Fry, D. The ionizing radiation environment in space and its effects. *AIP Conference Proceedings*, volume 1500, no. 1, 2012: pp. 198–203, doi:http://dx.doi.org/10.1063/1.4768766.

Appendix A

# The near Earth space radiation environment

Radiation is energy emitted as particles or waves. It is divided into two categories - ionizing and non-ionizing. Space radiation consists mainly of ionizing radiation which has enough energy to knock electrons off atoms and create ions, positively charged particles. Space radiation takes form of sub-atomic, high-energy, charged particles. Their natural sources are: trapped radiation, galactic cosmic rays (GCR) and solar energetic particles (SEP). [14]

The Sun constantly produces a stream of charged particles called Solar wind which are deflected by Earth's magnetic field. However not all of the particles are deflected and some become trapped in one of two magnetic rings surrounding the Earth called the Van Allen radiation belts. A part of the inner Van Allen belt drops down to an altitude of 200 km into the upper region of the atmosphere over the southern Atlantic Ocean. This area is known as the South Atlantic Anomaly. The drop in altitude results from the fact that the magnetic axis of the Earth is tilted from the spin axis and the center of the magnetic field is offset from the geographical center of the Earth.

The Sun is one of the main source of the space radiation with its SEP. They are high-energy particles of plasma that are accelerated from the energy that is released in the solar flare. These particles can be accelerated up to almost 80% of the speed of light. SEPs can also originate from the shock waves associated with the coronal mass ejections (CMEs). They most often form in the active regions on the Sun's surface and release huge quantities of matter and electromagnetic radiation into space. [15]

GCR is another type of extremely energetic particles that originate outside of our solar system. They are likely formed by explosive events such as supernova.

Near Earth space radiation monitoring is valuable to many space related fields. Some particle radiation is so powerful it can penetrate the outside of the

spacecraft and damage electronic circuits. Understanding of radiation effects is important for design and operation of spacecrafts and their components. Other fields that benefit from radiation monitoring include: solar activity, space radiation, space weather.

Appendix B

# Radiation effects

The ionizing radiation is a big concern that poses risk for spacecraft and space crews and needs to be studied to assist in spacecraft design and crew protection. If humans were to live and work safely in space a system to manage the radiation exposure has to implemented. Models that could predict the radiation conditions few days into the future would be highly desirable.

Radiation can penetrate spacecrafts and cause damage to the electronic components. Electron and protons deposited by the radiation can accumulate inside the dielectric layers of MOSFET transistors and after some time shift the operating voltage of the components using MOSFETs so much that the circuit stops working [16]. These total dose effects can be mitigated using a proper shielding and radiation-hardened electronics so they stop posing a threat.

Single event effect are much harder to overcome for space systems designers. There can be both destructive and non-destructive effects. Destructive effects include hard errors also referred as "stuck bit" in memory or gate ruptures. In case of non-destructive events the functionality of the circuit can be recovered. Most common are single event upsets that spontaneously flip bits of the memory and can be corrected by re-writing the bit. Single event latchups happen when bit is stuck in one power state until the chip is power cycled. Designers predict the space radiation environment for the lifetime of the mission and account for the most extreme radiation environment that can occur. However the solar energetic particle events cannot be predicted so engineers need to use previous measurements of extreme events.

Space radiation has effects on space crews as well. The effects of an acute dose can be malaise, nausea or vomiting which could be very dangerous during extravehicular activity (EVA). Chronic doses can cause carcinogenesis, neurological damage and degenerate tissues causing hearth diseases, respiratory and digestive diseases [17]. NASA sets radiation exposure limits to keep them as low as reasonably possible. Crewed mission vehicles are designed to withstand the expected radiation but long term missions such as those to Mars are in

doubt because of uncertainties in risks of chronic effects.

NASA is currently improving their space weather models so they can provide more detailed analysis of the radiation environment. Radiation exposure management for the crewed mission can be improved if the models are accurate and fast enough for use in real-time radiation exposure management.

Appendix C

# User documentation

The user documentation for this application is available on the attached CD in the PDF format or online as HTML page at `https://www.gitbook.com/read/book/ondrejbilek/satgraph`.

Appendix D

# Acronyms

**SATRAM** Space Application of Timepix based Radiation Monitor

**GCR** Galactic Cosmic Rays

**SEP** Solar Energetic Particles

**CME** Coronal Mass Ejection

**SAA** South Atlantic Anomaly

**ESA** European Space Agency

**IEAP** Institute of Experimental and Applied Physics

**CTU** Czech Technical University in Prague

**LEO** Low Earth Orbit

**K-NN** K-nearest neighbors

**DR** Dose rate

**LERP** Linear interpolation

**DOM** Document Object Model

**W3C** World Wide Web Consortium

**DIV** Division

**JS** Javascript

**SVG** Scalable Vector Graphics

**GUI** Graphical User Interface

**MVC** Model-View-Controller

**CI** Continuous Integration

**EVA** Extravehicular Activity

**NASA** National Aeronautics and Space Administration

**HCP** Heavy charged particle

**LCP** Light charged particle

Appendix E

# Contents of enclosed CD

- `readme.txt` ..... the file with CD contents description
- `src` ..... the directory of the source codes
  - `thesis` ..... the directory of the LaTeX source codes of the thesis
  - `satgraph` ..... the directory of the application source code
  - `doc` ..... the directory of the user documentation source code
- `thesis` ..... the directory of the thesis text and visualizations
  - `BP_Bilek_Ondrej_2016.pdf` ..... the thesis text in PDF format
- `satgraph` ..... the directory of the application installers
- `doc` ..... the directory of the user documentation
- `data` ..... the directory with input data and output images