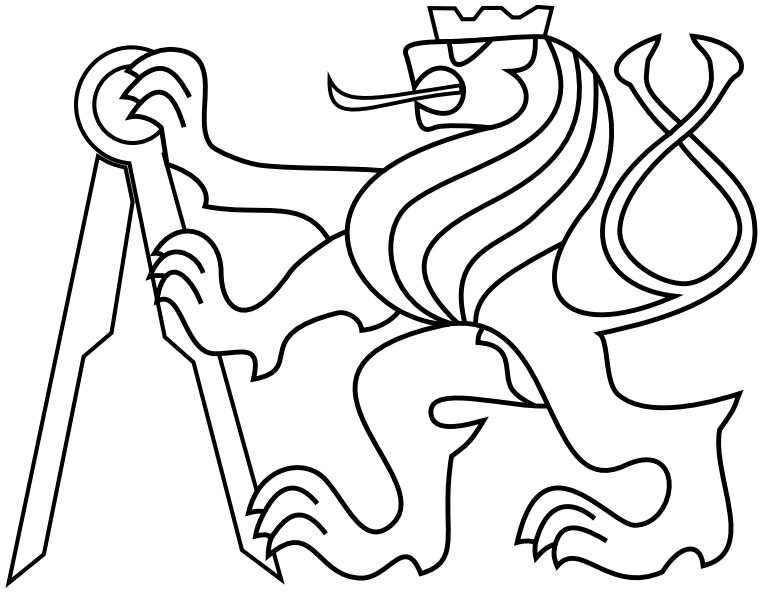
CZECH TECHNICAL UNIVERSITY IN PRAGUE

Faculty of Electrical Engineering

Department of Cybernetics

MASTER'S THESIS



Martin Jílek

Processing of Radiation Data from the Timepix Sensor on the VZLUSAT-1 Satellite

Thesis supervisor: **Ing. Tomáš Báča**

May 2018

Author statement

I declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university theses.

Prague, date..... .....

MASTER'S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Jílek Martin** Personal ID number: **419003**

Faculty / Institute: **Faculty of Electrical Engineering**

Department / Institute: **Department of Cybernetics**

Study program: **Cybernetics and Robotics**

Branch of study: **Robotics**

II. Master's thesis details

Master's thesis title in English:

Processing of Radiation Data from the Timepix Sensor on the VZLUSAT-1 Satellite

Master's thesis title in Czech:

Zpracování radiačních snímků ze sensoru Timepix na satelitu VZLUSAT-1

Guidelines:

The work aims to organize and process data from the X-Ray payload, which is onboard the VZLUSAT-1 satellite. The VZLUSAT-1 satellite was launched on July 23, 2017. Since the launch, the X-Ray payload has acquired more than 15 000 images (containing valuable information on the form of the ambient ionizing radiation in low-Earth orbit), most of which has been downloaded in a post-processed format [1]. The goal is to regress the information which was lost during the onboard processing of the data. The thesis will tackle the following points:Design and development of data manager, which will allow structuring, filtering, and annotation of measured data. The data manager should also serve as an interface to future machine learning algorithms.
1. Design of an image segmentation and classification method for particle tracks. Use segment features proposed in [2] and chooses appropriate classifier, e.g., Bayesian or artificial neural network (ANN).
2. Propose a regression technique to deduce the original radiation type in the post-processed orbital data.
3. Build a generator of learning datasets for the regression model based real-world data from the VZLUSAT-1 or other sources, e.g., SATRAM experiment onboard Probe-V satellite.
4. Apply the method to the measured data from VZLUSAT-1 and provide outputs in the form of labeled radiation maps.
5. If the proposed methods would not be capable of regressing the information, provide feedback for how the future missions should handle the onboard data processing to maximize the information gain.

Bibliography / sources:

[1] T Baca, M Platkevic, J Jakubek, A Inneman, V Stehlikova, M Urban, O Nentvich, M Blazek, R McEntaffer and V Daniel. Miniaturized X-ray telescope for VZLUSAT-1 nanosatellite with Timepix detector. Journal of Instrumentation 11(10):C10007, 2016.
[2] Holy, T., et al. "Pattern recognition of tracks induced by individual quanta of ionizing radiation in Medipix2 silicon detector." in Nuclear Instruments and Methods, Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment 591.1 (2008): 287-290.
[3] Hoang, Son M. A pattern recognition approach to learning tracks of heavy-ion particles in timepix detectors. Diss. 2013, University of Houston.
[4] ATLAS collaboration. "A neural network clustering algorithm for the ATLAS silicon pixel detector." Journal of Instrumentation 9.09 (2014): P09009.
[5] Bouchami, J., et al. "Study of the charge sharing in silicon pixel detector with heavy ionizing particles interacting with a Medipix2 and a Timepix devices." Nuclear Science Symposium Conference Record, 2008. NSS'08. IEEE. IEEE, 2008.

Name and workplace of master's thesis supervisor:

Ing. Tomáš Báča, Multi-robot Systems, FEE

Name and workplace of second master's thesis supervisor or consultant:

Date of master's thesis assignment: **12.01.2018** Deadline for master's thesis submission: **25.05.2018**

Assignment valid until: **30.09.2019**

Ing. Tomáš Báča	doc. Ing. Tomáš Svoboda, Ph.D.	prof. Ing. Pavel Ripka, CSc.
Supervisor's signature	Head of department's signature	Dean's signature

III. Assignment receipt

The student acknowledges that the master's thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the master's thesis, the author must state the names of consultants and include a list of references.

.	
Date of assignment receipt	Student's signature

Acknowledgements

I would like to thank my supervisor, Ing. Tomáš Báča, who was always ready for discussion of new ideas and who shared his knowledge from the field with deep interest, for his guidance during all stages of my work on this thesis.

Abstract

The cosmic environment puts space technology and astronauts in front of many challenges. One of these is ionising radiation that can cause failure of electronic systems, material degradation and radiation damage to living tissues. It is necessary to understand this environment to occupy the orbital space. One of the tools available for this purpose is a family of pixel radiation detectors. These sensors are lightweight, small and have low power requirements. The representative of this family of radiation detectors, Timepix, is part of a telescope in the nanosatellite VZLUSAT-1. Although the telescope is designed for observation in the X-ray spectra, traces of particles other than photons are visible on the images. However, the telescope sends most of the images to the Earth in a highly lossy compressed format optimised for X-ray observations, traces of particles other than photons are unidentifiable. In this work, a methodology has been proposed and successfully tested which can restore the lost information to such an extent that it allows the calculation of radiation maps for different types of particles on the low Earth orbit.

Keywords: semiconductor pixel radiation detector, space radiation monitoring, cubesat, LEO dosimetry, machine learning, computer vision, image processing

Abstrakt

Kosmické prostředí staví vesmírnou techniku i biologické materiály před řadu překážek. Jednou z nich je ionizující záření, které způsobuje poruchy elektronických systémů, degradaci materiálů i radiační poškození živých tkání. Pro využívání orbitálního prostoru je nutné tomuto prostředí porozumět. Jedním z nástrojů využitelných pro tento účel jsou pixelové radiační detektory, které jsou nenáročné na rozměry, hmotnost i napájení. Zástupce této rodiny radiačních detektorů, Timepix, je součástí teleskopu v nanosatelitu VZLUSAT-1. Přestože je teleskop určen pro pozorování v rentgenové oblast, na snímcích jsou patrné i stopy jiných částic než fotonů. Nicméně teleskop většinu snímků posílá na Zemi v silně komprimovaném formátu, optimalizovaném pro rentgenová pozorování. Stopy jiných částic než fotonů jsou v nich neidentifikovatelné. V této práci je navržena a úspěšně implementována metodika, která umožňuje z těchto snímků ztracenou informaci částečně obnovit a využít snímky satelitu obsahující jen záběry radiačního pozadí pro tvorbu map rozložení jednotlivých částic na nízké oběžné dráze Země.

Klíčová slova: polovodičový pixelový radiační detektor, sledování vesmírné radiace, cubesat, LEO dozimetrie, strojové učení, počítačové vidění, zpracování obrazu

Contents

List of Figures

List of Tables

List of Acronyms

FIPEX	Flux-Φ-Probe Experiment
ANN	Artificial Neural Network
ESA	European Space Agency
FPGA	Field-Programmable Gate Array
ISS	International Space Station
k-NN	k-Nearest Neighbors
LHC	Large Hadron Collider
LDA	Linear Discriminant Analysis
LEO	Low-Earth orbit
NASA	National Aeronautics and Space Administration
RHCH	Radiation Hardened Composite Housing
SATRAM	Space Application of Timepix based Radiation Monitor
SVM	Support Vector Machine
ToT	Time-over-Threshold
CRAND	Cosmic Ray Albedo Neutron Decay
IDW	Inverse Distance Weighting
MSE	Mean Squared Error
MST	Minimum Spanning Tree
NORAD	North American Defense Command
SVM	Support Vector Machine
TLE	Two-Line Elements
UART	Universal asynchronous Receiver/Transmitter
WMAP	Wilkinson Microwave Anisotropy Probe

Chapter 1

Introduction

This chapter explains the purpose of the mission of the VZLUSAT-1 satellite and also relationship of this satellite and this thesis. The entire satellite is introduced with all the scientific equipment it is equipped with. More attention is paid to the Timepix pixel detector as the central data source for this thesis. At the end of the chapter state-of-the-art analysis is presented.

1.1 The mission

Satellite VZLUSAT-1 is a Czech technological satellite launched on June 23, 2017. It is a part of the international project QB50 [1], which aims to demonstrate the low-cost space accessibility using small satellites. In total, 50 nanosatellites, which have an expected life span of months, are planned to be launched. These satellites serve as a technological demonstration while conducting research on the lower thermosphere. Each satellite carries at least one scientific apparatus related to this topic: Ion-Neutral Mass Spectrometer, Flux-Φ-Probe Experiment (FIPEX) and Multi-Needle Langmuir Probe. In addition, satellites can carry any other payload.

1.1.1 The satellite

VZLUSAT-1 (in figure 1.2) is a CubeSat with a mass of 1.83 kg and launch size of 2U ($10 \times 10 \times 20$ cm). After successful launch into the orbit (which was on June 23, 2017), the X-ray telescope was unfolded and the satellite extended to size of $10 \times 10 \times 30$ cm. The orbit (in figure 1.1) is circular polar with altitude 500 ± 20 km. Repeat cycle of the orbit is 12 hours with orbital period approximately 90 minutes. Except mission-critical components like power, communication and control systems, the satellite is equipped to perform three scientific experiments: FIPEX, RHCH and lobster-eye X-ray telescope sky monitoring. FIPEX experiment, which focuses on study of oxygen in lower atmosphere layers, is a part of the QB50 scientific program. RHCH (Radiation Hardened Composite Housing) is experiment which studies shielding capabilities and effects of orbital environment on degradation of a housing consisting of carbon fiber composite covered by thin metal layers. Last experiment is X-ray telescope for wide-sky imaging. The heart of this telescope – the Timepix pixel detector – is the source of all data processed in this thesis.

1.1.2 The X-ray telescope

X-ray telescope is, for this thesis, the most important part of scientific payload of the VZLUSAT – 1 satellite. It consists of two main parts: lobster eye X-ray optics and a Timepix sensor. Lobster eye optics is a reflection-based optical system which is capable of focusing high-energy electromagnetic radiation with index of refraction close to one for almost all materials. Lobster eye optics can also achieve very wide field of view. It consists of many channels (in 1D optics

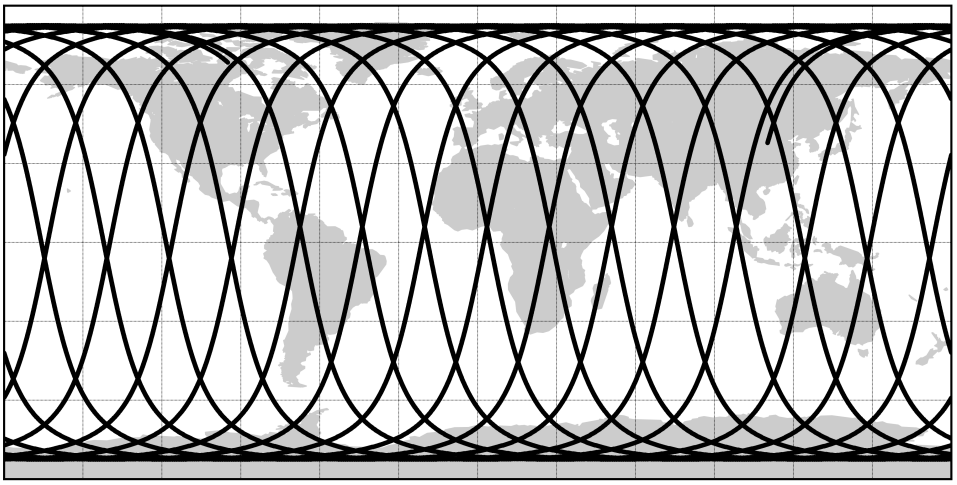


Figure 1.1: Orbit of the satellite during 24 hours in cylindrical projection.

case as in VZLUSAT-1), or tubes (in general 2D case) with reflective surface (e.g., gold plated), whose walls close nonzero angle between themselves. Rays passing through this optical system are focused onto the Timepix sensor, a hybrid semiconductor detector with resolution 256×256 pixels with pixel size 55×55 μm.

The primary objective of the X-ray telescope is an observation of the Sun in the X-ray spectrum. However, due to the problematic control of attitude of the satellite, it is difficult for the telescope to aim precisely at the target. As a result, most of the images contain only space radiation background comprising mainly of charged particles which originates from the Sun and were trapped in the magnetosphere of the Earth. Some of these (charged or uncharged) particles also come from the deep space environment. Trajectories of these particles mostly do not respect the X-ray optics and hit the Timepix sensor from various (and random) directions.

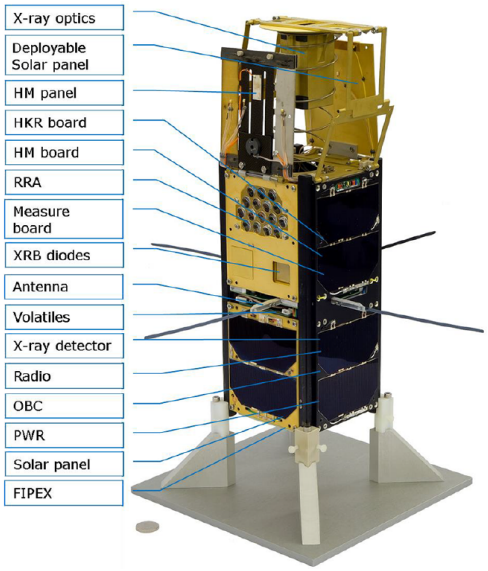


Figure 1.2: VZLUSAT-1 satellite, source [2].

Timepix, as a pixel detector, is capable of capturing the traces of the ionizing particles that pass through it and produces a bitmap-like image of them. Since these traces are characteristic for different particle types, it is possible to measure the flows of different types of ionizing radiation through the sensor at different locations of the satellite orbit. The problem is that the resulting full resolution images produced by the sensor have big data volume. The size limits number of images which can be sent to the Earth and makes dense scans hardly feasible. The data compression scheme is hardwired in the on-board computer of the satellite and is optimized for images produced by the lobster optics, which contains only photons, that leaves the simplest traces possible on the detector – most often only one or two active pixels. Because of that, the compression throws off almost all information about the shape of more complex particle trajectories and also the amplitude of a measured signal in pixels. However, it allows for collection and download of a

considerable amount of images (in order of tens of thousands during expected satellite life).

1.2 Problem statement

This thesis addresses the problem of prediction of ionizing particle fluxes from strongly compressed images created by the Timepix sensor in VZLUSAT-1 satellite. The solved problem is nontrivial, since, as stated before, most incoming data from the satellite are in a form which is not suitable for this task. Only a small subset (in order of hundreds of samples) of downloaded frames is available in a full resolution. Preprocessing pipeline with the problematic compression method is hard-wired in a computer of the satellite, so the only feasible way to count particles on downloaded data is to create regression model, trained on a few hundreds of available full resolution images, and then to apply it to all compressed data to obtain prediction.

The thesis deals with this problem in three steps. First one is an establishment of data organization tools since we are dealing with tens of thousands of images and this collection keeps growing as new data are being downloaded regularly (as of May, 2018).

During second phase it is necessary to train a classifier, which will label all particle traces in full resolution images. The labels will be based on a shape characteristics of particle tracks.

Then these labelled images will be compressed in the same way as they are processed in on-board computer of the satellite. Resulting low resolution images will be used in training of a regression model, which will be then applied to all images created by the satellite so far. Finally, counts of particles will be estimated and visualized in maps.

1.3 State-of-the-art

Hybrid pixel detectors like Timepix are used in laboratory experiments, but are also useful for real-world applications, e.g., in radiotherapy [3] or medical imaging [4]. It can be also used for industrial imaging and non-destructive testing [5].

These detectors are also used for aerospace purposes. In the year 2013 the probe PROBA-V with SATRAM (Space Application of Timepix based Radiation Monitor) was launched into space with Timepix detector operating in ToT[1] mode with the goal of sampling radiation environment along the 820 km LEO orbit. It is capable of sampling few full resolution images per second, which results in around 4×10^3 sampled images per day [6].

Another important space application is dosimetry on the ISS. In the year 2013, five Timepix detectors were placed on the ISS, together with software which processes the measurements and outputs dosimetry data. After failure of one unit two sensors were added. In the year 2015, these detectors were still working and proved that it should be possible to design personal dosimeters for the crew from them [7].

Postprocessing of data collected during particle physics and dosimetry experiments with pixel detectors is not much covered by the literature, even though there are projects dealing with large amounts of data (e.g., space dosimetry observations [6], [7] or particle physics laboratories like LHC).

LHC is one of the laboratories where automated processing of large volumes of data is helpful. Article [8] deals with clustering of images collected from multiple Timepix detector. Multiple feed-forward neural networks were used to separate tracks intersecting in one image segment on a pixel detector and also for estimation of a sensor entry point of every detected particle. Segmentation was performed as a search for connected components with 8-cell connectivity.

[1] ToT: time-over-threshold, a mode in which the sensor measures the energy rather then the number of incoming particles.

Classification of heavy ion traces (H, He, C, N, O, Ne, Si, Ar, Fe) on a pixel detector is treated in a dissertation thesis [9]. Hierarchical classifier is developed in this thesis. It consists of two parts: thresholding of selected features extracted from data and "black-box" machine learning methods (LDA, k-NN, ANN, SVM and gradient boosted decision tree). Features were derived with the use of skeletonization and distance transform. Moreover, a method for determination of azimuth angle of incoming particle was developed.

Similar approach was chosen in a more software-oriented thesis [10]. Here framework for recognition of traces of electrons, protons, alpha particles, muons and gamma-photons was designed and implemented. As a features linearity, standard deviation of cluster pixels from fitted line, tortuosity, convex hull area and circularity were used. Classification was performed with LDA, boosted decision trees and multi-layer perceptron.

Another possible approach is to use mathematical morphology as in [11]. Proposed method uses only dilation and erosion and is suitable for fast FPGA implementation. Particles are classified into three classes (blob, track, dot) in a residual approach.

1.4 Contributions

The use of the Timepix sensor for space dosimetric measurements is a current issue addressed by leading space organizations such as ESA and NASA. Nevertheless, results of these measurements as well as a more precise description of methods used in data processing are generally not available. Among the explored literature in available databases, this work is one of the the first contributions that use the Timepix detector in combination with machine learning methods to evaluate the spatial distribution of the amount and energy of various ionizing particles in the LEO environment.

The results of this thesis increase utility value of the VZLUSAT-1 satellite by designing and implementing a new data processing methodology for strongly lossy compressed Timepix data. Thanks to the developed method, it is possible to estimate numbers of detected both photonic and non-photonic ionizing particles in images produced by the satellite's X-ray telescope, which are damaged by lossy compression.

The resulting solution makes it possible to utilize thousands of images. These images would probably not be useful without the proposed reconstruction method, since the on-board compression algorithm is designed specifically for observation of X-ray sources like the Sun and all the analyzed images were taken at a time when the telescope was not aimed at it. A total of 29 scans of the whole Earth was scanned from September 2017 to May 2018. The proposed methodology was used to estimate spatial distributions of different types of ionizing particles in a low Earth orbit, along with their energetic spectra. In spite of the fact that the processed images were originally intended for a diametrically different purpose – observation of the Sun in X-ray spectrum – the proposed reconstruction methodology achieved meaningful results, at least from qualitative point-of-view. At the same time, it has been shown that even with the use of a low-cost technology, such as technological CubeSat, it is possible to achieve interesting results.

1.5 Outline

The solution is divided into several consecutive steps, which are described in the following chapters. In chapter 2, the physics of cosmic radiation environment is introduced along with a technical explanation of the Timepix detector and the compression method used by the on-board computer of the satellite. The chapter 3 describes the designed software infrastructure for data storage and processing. The data are more elaborately analyzed in chapter 4, which describes

the design of the classifier of particle traces. In chapter 5, the regression model is created which allows estimating counts of particles in compressed images. This chapter also describes the resulting data visualization. The obtained results are discussed in chapter 6. Chapter 7 concludes the output of this thesis.

Chapter 2

Preliminaries

This chapter describes the mechanisms involved in creation of the images produced by Timepix sensor on the VZLUSAT–1 satellite. At the most general level it is especially the behavior of charged particles in the Earth's magnetosphere. At a lower level, it is the interaction of these particles with the material of the detector. Also, the electronics of the sensor and the subsequent digital processing (in particular the data compression method), which have a considerable impact on the appearance of the resulting images, are explained.

2.1 Physics of cosmic radiation

Space around the Earth is a hostile environment in terms of a radiation. It is filled with galactic cosmic rays, heavy particles created during solar events, trapped radiation and secondary neutrons.

Spatial particle distribution can be explained by the geomagnetic field, generated mainly by convection of molten core of the Earth, a permanent magnetization of Earth's materials, by movements of atmospheric masses (which also causes movement of contained charged particles) and by an interaction of solar wind with atmosphere and magnetosphere. Geomagnetic field shape is similar to the field generated by a magnetic dipole located near the center of the Earth (which is a valid approximation for explanation), further deformed by the solar wind [12].

2.1.1 Movements of a charged particle

For explanation of Van Allen belts existence it is needed to examine motion of charged particle in electric and magnetic field. The force, acting on a particle traveling through electric field with intensity $\mathbf{E}$ depends in the intensity and charge of the particle as follows:

$$\mathbf{F_e} = q\mathbf{E}. \tag{2.1}$$

Presence of magnetic field causes that the charged particle is also influenced by the force

$$\mathbf{F_M} = q(\mathbf{v} \times \mathbf{B}), \tag{2.2}$$

where $\mathbf{v}$ is speed of a particle and $\mathbf{B}$ is magnetic field vector. Cross product in equation 2.2 causes that a particle in magnetic field can experience force in a direction perpendicular to the magnetic field line and also the velocity vector. In combination with centripetal force, a particle affected by nonzero $\mathbf{F_M}$ will move in a circle with a so-called Larmor radius R_c [12]:

$$R_c = \gamma \frac{m_0 v_\perp}{|q|B}, \tag{2.3}$$

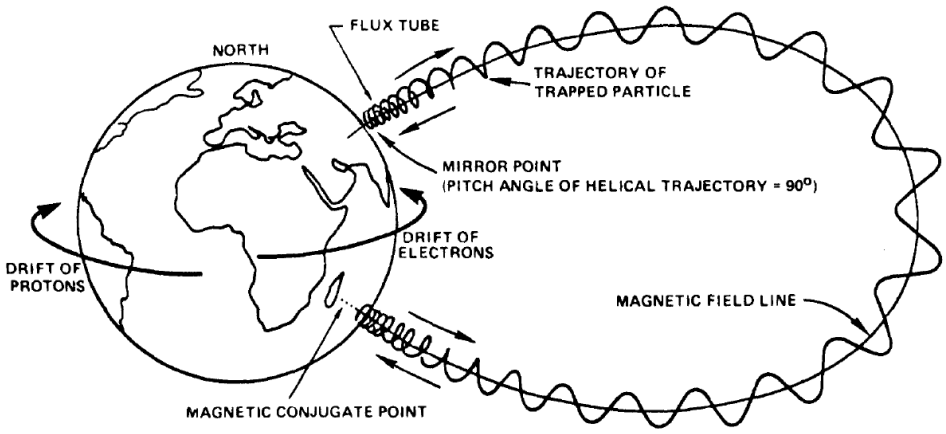


Figure 2.1: Most important movements of particles in magnetic field of the Earth, source [12].

where m_0 is mass of the particle, $v_\perp$ is size of speed perpendicular to the magnetic field $\mathbf{B}$ with size B and q is electric charge of the particle. γ is called relativistic factor, which can be written as

$$\gamma = \frac{1}{\sqrt{1-\beta^2}}. \tag{2.4}$$

Except this circular motion with radius R_c, charged particle in Earth-like magnetic field can bounce between mirror points of field and also drift. All these basic motions are depicted in figure 2.1. Mirror points are special points, where the field lines are converging. Here, due to the nonzero gradient of B, the particles can be slowed down and reflected. Each geomagnetic pole has its own set of mirror points, so there will be trapped particles, which are bouncing between mirror points of the North and the South pole [12].

Drift motion of charged particles is observable in azimuthal direction. This motion is caused mainly by inhomogeneity of magnetic field (in the size approximately of particle's gyroradius), which is stronger closer to the surface of the Earth. Due to the drift motion, charged particles spread in a ring around the earth, where direction of the drift depends on the charge of the particle [12].

2.1.2 Adiabatic invariants

Common way to model behavior of charged particles in magnetosphere is to use three adiabatic invariants [12], [13]. These invariants are quantities, which are (almost) constant during motion of a such charged particle.

The first adiabatic invariant is calculated from gyro motion of the particle and is called (relativistic) magnetic moment:

$$J_1 = \mu = \gamma^2 \left(\frac{\frac{1}{2} m_0 v_\perp^2}{B} \right). \tag{2.5}$$

This invariant can be assumed to be constant when spatial and temporal variations of magnetic field are much larger than gyroradius and gyroperiod of particle. The second adiabatic invariant is defined as:

$$J_2 = \frac{1}{2} \oint \mathbf{P} \mathrm{d}l = \int_{-l_m}^{+l_m} P_\parallel \mathrm{d}l, \tag{2.6}$$

where $P_\parallel$ is a size of particle momentum in direction of guiding center magnetic field line and

$\mathrm{d}l$ is an element of a guiding center magnetic field line. If the time variations of magnetic field are small in comparison with length of particle bounces between two mirrorpoints, this invariant will be constant. The third adiabatic invariant is given as:

$$J_3 = \frac{q}{c} \oint \mathbf{A} \cdot \mathrm{d}l = \int_S \mathbf{B} \cdot \mathrm{d}\mathbf{S}, \tag{2.7}$$

where A is magnetic vector potential. This invariant is approximately constant when the azimuthal drift around the Earth is much slower than rate of change of a magnetic field.

2.1.3 Structure of magnetosphere

The magnetosphere is a region of a magnetic field around the Earth, as depicted in figure 2.2. Without influence of the solar wind, the magnetosphere would be similar to the magnetic field generated by a magnetic dipole. However, this magnetic field is deformed by a solar wind, which travels at supersonic speed. On the day side of the Earth, the bow shock is formed by deceleration of solar wind to subsonic velocities. So, in comparison with the magnetic field of a dipole, day side magnetosphere is squeezed and a night side is extended to interplanetary space [14]. Under the bow shock there are regions called the magnetosheat and the magnetopause. The magnetopause is a border of magnetosphere. Its position is not exactly defined, since it depends on a solar activity [14].

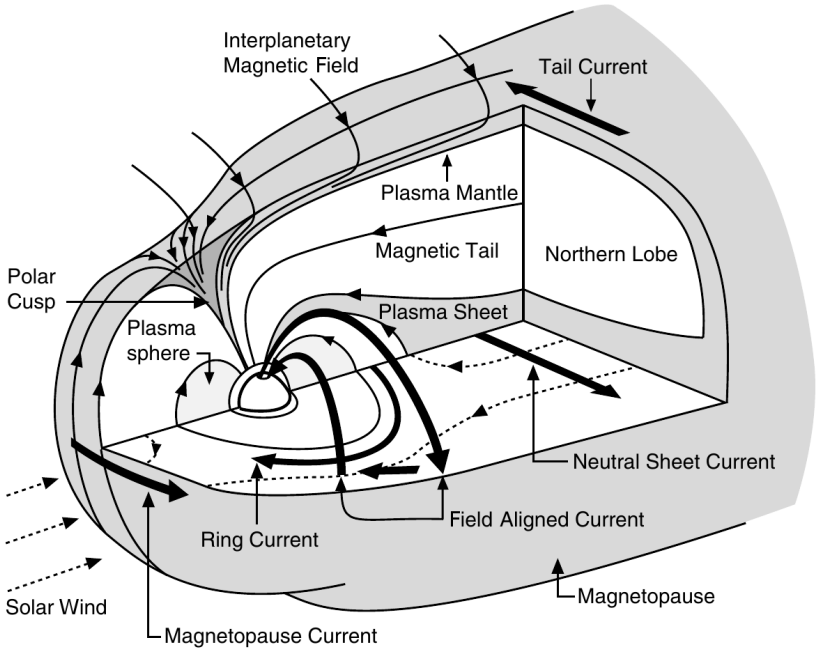


Figure 2.2: Topology of magnetosphere of the Earth, source [14].

On the night side of the Earth, the magnetosphere is elongated in two lobes, called the tail lobes. Each lobe is connected to south or north pole and extends up to the interplanetary space. This connection is caused by an interaction of magnetic field of plasma with the magnetic field of the Earth – the so-called magnetic reconnection. Due to the topology of field lines in the tail lobe, the density of particles in this area is not large. Between both tail lobes, there is a volume filled with higher density plasma, the plasma sheet. Magnetosphere contains two toroidal regions, the Van Allen belts, in proximity of the Earth. These regions are of a special importance in cosmic engineering because they are filled with charged particles and poses a threat to spacecraft and astronauts [14].

2.1.4 Van Allen belts

The ring-shaped regions, filled with charged particles, are known as the Van Allen belts (shown in figure 2.3). There are two known radiation belts, which lie in (magnetic) equatorial plane. The inner belt is comprised dominantly of protons with very high energy (up to multiple GeV), then there are also heavier ions and trapped electrons with energies lower than 1 MeV. The peak intensity of inner belt is detectable approximately 2 Earth radii (R_E) from the centre of the Earth, in the magnetic equatorial plane [15].

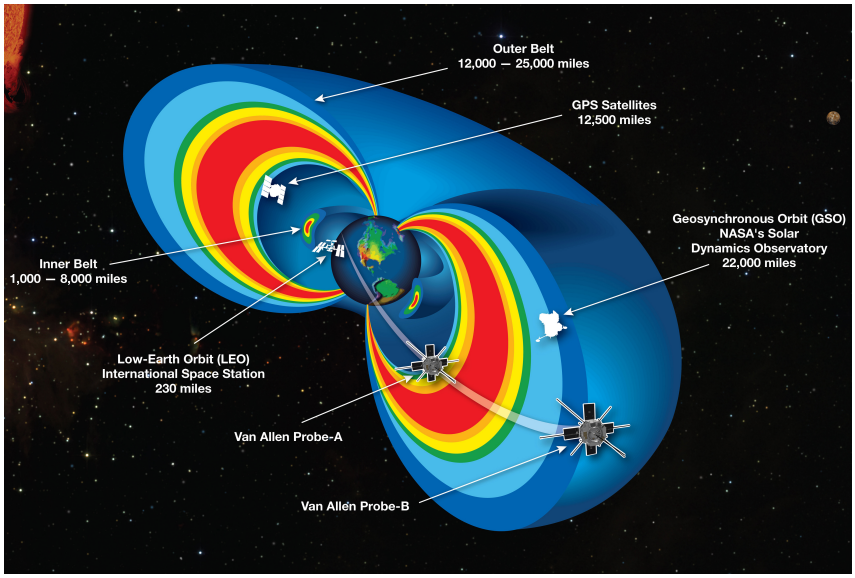


Figure 2.3: Radiation belts with Van Allen NASA probes, source [16].

As the first-order dipole approximation of magnetic field of the Earth is shifted from the geocentre and also tilted with respect to the geographic axis, part of the inner radiation belt is closer to the surface of the Earth. This results in higher proton fluxes in lower altitudes than anywhere else above the Earth. Because of the location of this effect, it is called the South Atlantic Anomaly [17]. Outer belt peaking intensities are approximately between 3 and 6 R_E. The belt contains high amount of electrons with energies between 100 keV to more than 10 MeV. There is a "slot" region between these two belts. Number of electrons in this region is lower, but it can be affected by a space weather [15].

According to [12], charged particles located in the radiation belts of the Earth originates mainly from:

- the solar wind,
- the ionosphere,
- cosmic ray albedo neutron decay (CRAND),
- particles accelerated in magnetospheres of other planets,
- low energy components of galactic cosmic rays.

2.2 Interaction of ionizing radiation with matter

Interactions of charged particles are of a different nature than interactions of particles with zero charge. All charged particles can interact by electromagnetic interaction. In the case of hadrons (particles made from quarks), nuclear interaction can also happen.

Electromagnetic interaction is a significant contributor to the energy loss along a track of a charged particle penetrating a medium. Penetrating charged particles leave track of excited and

possibly ionized atoms with free electrons. Most of these free electrons have microscopic trajectory, however, those with high energy travel macroscopic distances and also cause excitations or ionizations. These electrons are called δ-rays. Electromagnetic loses due to the interaction with electrons are given by the Bethe-Bloch equation [18]:

$$-\frac{\mathrm{d}E}{\mathrm{d}x} = Kz^2\frac{Z}{A\beta^2}[\frac{1}{2}\ln\frac{2m_ec^2\beta^2\gamma^2T_{max}}{I^2} - \beta^2 - \frac{\delta(\beta_\gamma)}{2}], \tag{2.8}$$

where $K = 0.31$ MeV·cm^2/g, z is charge of the particle, Z is atomic number of absorber, A is its atomic mass, $\beta = \frac{v}{c}$ (where v is speed of the particle and c speed of light), m_e is charge of electron, I is excitation energy and T_{max} is maximum energy, which can be transfered from traveling particle to electron in one collision. $\delta(\beta)$ is a correction factor and $\gamma = \frac{1}{(1-\beta^2)^2}$. This equation allows to calculate energy loss per unit length for a given particle traveling through given material, but its application have several constraints. For example, it is valid only in a statistical sense of view: the energy loss according to this equation is only average one, it is not possible to use it for exact prediction of behavior of a given one particular particle. Accuracy of calculated energy loss also depends on energy of the particle. Because of this, additional corrections must be applied to the equation when accuracy is a concern [18].

Energy losses of a particle traveling through particular medium are not constant along its trajectory. Dependence of energetic losses on depth of a particle in given material is called the Bragg curve. This curve has one global maximum, called the Bragg peak. This peak is in the depth where the particle looses maximum energy.

Important effect affecting particle trajectories is multiple scattering. Charged particle traveling through matter can be reflected due to the Coulombic forces, which results in change of its direction. This effect is not so important with heavy particles with energies around 1 MeV and higher, but it significantly affects trajectories of electrons, which are able to travel macroscopic distances. Change of speed of a charged particle is accompanied by release of electromagnetic radiation. This effect is called Bremsstrahlung in the case of interaction of charged particles with a nuclei of matter. Another effect is release of the Cherenkov radiation, visible radiation, which is caused by charged particles traveling at speeds higher than speed of light in a given medium.

High-energy photons (X-rays, gamma rays) exhibit three main effects: photoelectric effect, Compton scattering and pair production. Photoelectric effect is caused by excitation or ionization of atoms by incoming photons. If energy of the photon is sufficiently high, target atom can be ionized. This results in release of a free electron and rearrangement of electrons of the atom. Excess energy is released as X-ray radiation or in a form of an Auger electron. The photoelectric effect is a major interaction effect for photons with energy lower than 100 keV.

Compton scattering is interaction between a photon and an electron, where the photon collides with the electron and both of them are deflected. The interaction happens mostly for photons with energies between 100 keV and 1 MeV.

Pair production is interaction of photons with nuclei, where pair electron-positron is created. It is characteristic interaction for photons with energies higher than 1 MeV.

Neutrons, as well as protons, also undergo nuclear interactions. In comparison with neutrons, protons must have kinetic energy high enough to overcome repulsive coulombic forces from the nuclei. Interaction of a proton with nucleus is often followed by brake-up of the target nucleus into unstable products.

Neutrons can be divided into three energetic classes: high-energy, fast and slow, according to their decreasing kinetic energy. High-energy neutrons have energy around 1 GeV and interact with nuclei in a way similar to protons. Fast neutrons (energy between 100 keV and 1 GeV) exhibit elastic collisions and their energy is continually decreased due to the inelastic collisions.

Slow neutrons are neutrons which most often exhibit elastic scattering and neutron capture [19].

2.3 Timepix sensor

This section describes the Timepix sensor (in figure 2.4), which serves as a detector in a VZLUSAT-1 lobster-eye telescope. This sensor is responsible for generating all the data processed in this thesis. The sensor belongs to the family of semiconductor pixel detectors of ioniziong radiation. This class of sensors is a solid-state alternative to more classical particle detectors like bubble chambers. It works on a similar principle – it detects electric charge generated by the penetrating particle in the body of a detector. This generated charge is collected in a discrete rectangular grid of pixels and the signal is further processed.

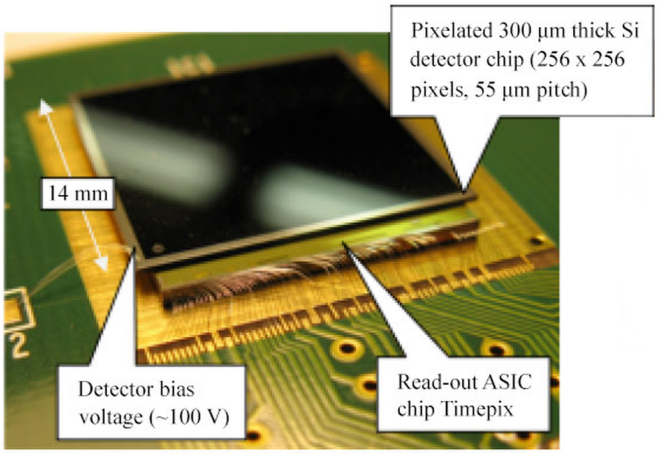


Figure 2.4: Timepix sensor, source [20].

Timepix detector mounted on VZLUSAT-1 has resolution of 256 × 256 pixels. Each pixel has square proportions with 55 μm of side length. The detecting material is a silicon plate, 300 μm thick and 14 mm long, coated with aluminium to mitigate exposure of visible light.

2.3.1 Electronics

Each pixel of Timepix sensor has its own analog and digital circuitry (figure 2.5). Analog circuitry contains preamplifier, which converts collected charge from pixel to voltage. Voltage is compared with threshold in comparator and if the threshold value is exceeded, positive logical signal is sent to the digital part of pixel circuitry. Digital part contains Timepix Synchronization Logic (a counter driven by a global clock signal), which fills shift register with data according to mode of operation. Each pixel counter has 14-bit depth, so the maximum number it can store is 11810.

The sensor can operate in three modes. In the first one, called the Medipix mode, each pixel counts threshold crossings during exposure. Second, the Timepix mode, measures time of particle arrival. Finally, the time-over-threshold mode uses counter as Wilkinson AD converter, thus allows to measure dissipated energy in pixel [22].

Our application uses time-over-threshold mode of measurement. In this mode measured raw sensor values have meaning of time. To assign energetic meaning to it, each pixel of the sensor must be calibrated. This procedure (which was performed in laboratory before the satellite's departure) consists of measuring radioactive sources with known energies. With the use of these data it is further possible to fit a function which assigns energy to each raw pixel value [23]. This kind of a sensor, operating in time-over-threshold mode, allows to observe and analyze both energetic and geometrical properties of particle track. Particle tracks are characteristic for each particle type, like in cloud or bubble chambers.

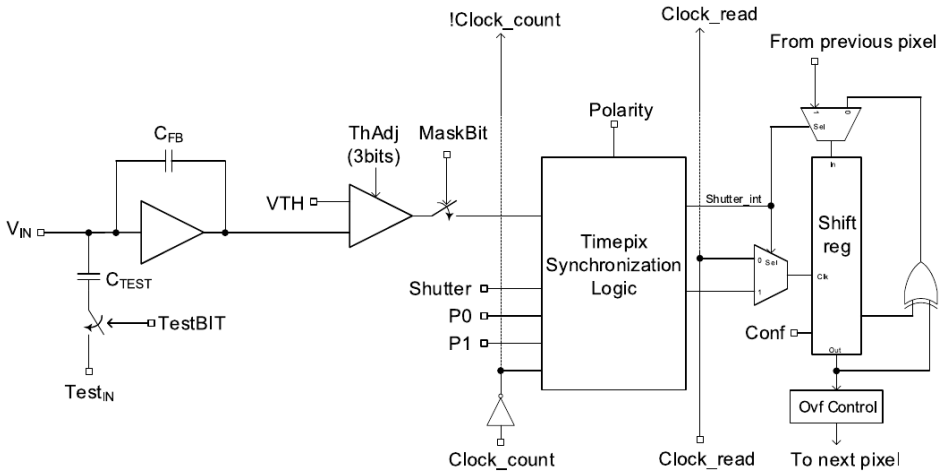


Figure 2.5: Timepix sensor pixel circuitry, source [21].

2.3.2 Interaction of a ionizing radiation with the detector

Particles interacting with the detector are both charged and neutral. For charged particles it is expected that main contribution to the dose on satellite's orbit will consist of protons, heavier particles (e.g., alpha particles), muons and electrons. Detected particles with zero charge are mainly X-ray and gamma photons.

Particles entering Timepix detector cause ionization of detector material along the track according to the processes described above. Generated charges drift along bias electric field and also diffuse laterally, which results in a so-called charge sharing effect [24]. It means that pixels in neighborhood of the track can capture some of the energy of the track, even if the track did not directly penetrate them. The impact of charge sharing effect on resulting images depends on multiple conditions, such as type of the penetrating particle, sensor threshold and bias voltage. This effect is illustrated in figure 2.6. Examples of full resolution images, captured in orbit, can be seen in figure 2.7.

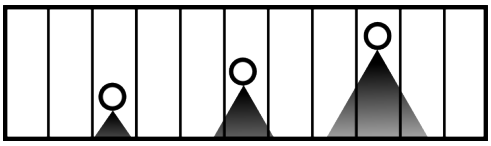


Figure 2.6: Illustration of charge sharing effect in Timepix detector viewed from the side. Circles represent point in pixels where the charge was generated by ionizing particle. The cones represent spreading of charge as it is travelling to the collection electrode in bias voltage.

2.4 On-board processing and image format

This section describes subsystems of the satellite which are responsible for control of the Timepix detector and processing of captured data. First, the control hardware is briefly introduced. It is followed by description of compression method applied on images and finally the format of downloaded data is described. All presented information is available in the article [2].

The Timepix detector, described in previous chapter, is connected to ATxmega128a4u microcontroller with 8 kB of internal RAM and 256 kB of external memory. External memory serves as a holder of settings and as a temporary storage for images.

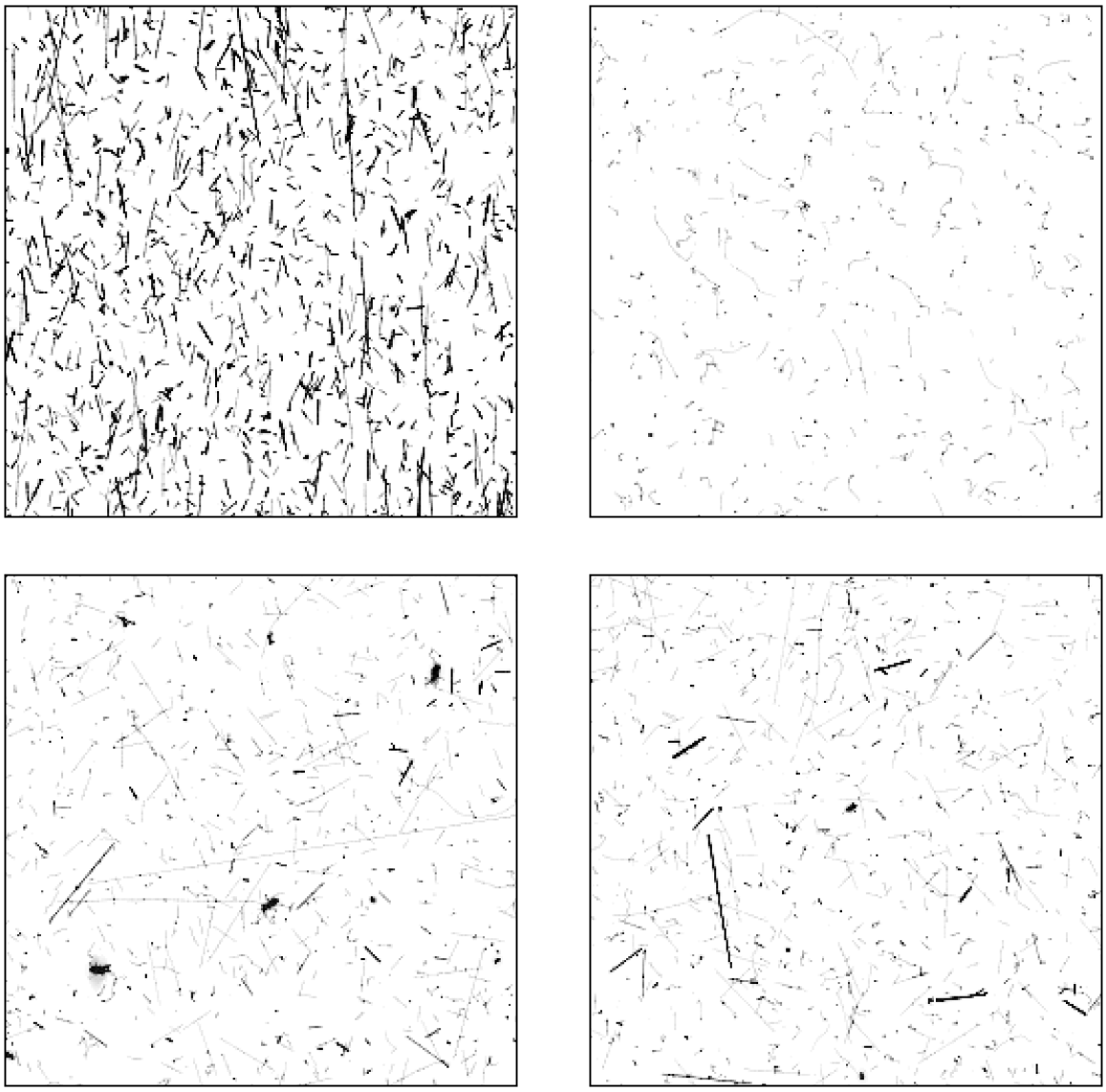


Figure 2.7: Selected examples of full resolution images produced by the Timepix detector of the VZLUSAT-1 satellite.

Infrared and wide-field UV sensors were installed on the board to improve estimation of attitude of the satellite. There is also narrow-field-of-view UV sensor, which serves as a trigger for the tellescope. Temperature is measured by a thermometer mounted on Timepix's aluminium heatsink. Safety shutdown is performed in a case of overheat.

Images are read from Timepix with USB Lite Interface, which is a miniaturized version of USB 1.0 [25]. It allows to communicate with sensor via UART and also provides power supply, clock signal and bias voltage for it.

Captured data are split into two parts: metadata and image data. Metadata contain detailed information about each captured frame:

- time of exposure,
- pixel hit count,
- minimum and maximum values,
- exposure parameters (threshold, bias, acquisition time),
- measurement mode,
- filtering (on, off),

- data format,
- data address,
- heatsink temperature.

Image data are represented not as a matrix of values (which would be ineffective, since the images contain mostly zero values), but rather as set of triplets, where each triplet contains two coordinates of one active pixel and also its value.

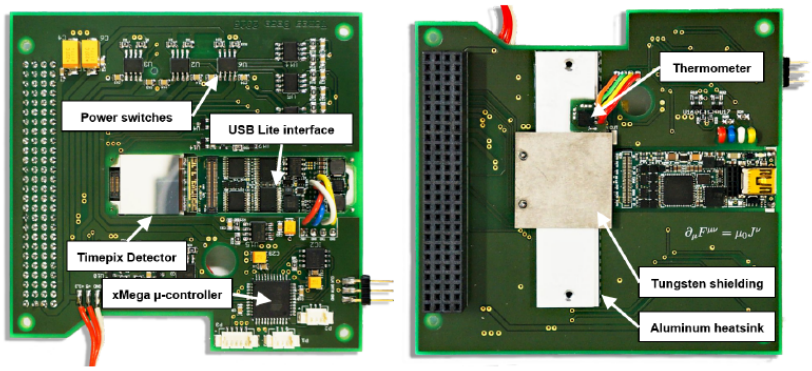


Figure 2.8: The X-ray payload board with the Timepix detector, source [2].

2.4.1 Image compression

Raw, full resolution image has a resolution of 256×256 px. Size of a transmitted raw image frame is variable and depends on a count of non-zero pixels. It can extend up to 3300 packets with 64 B payload. Each of these packets is an elementary data chunk, which can be downloaded from the satellite. The download speed can be as high as 1 kBit/s, but it must be balanced between multiple subsystems of the satellite. Moreover, connection with the satellite is possible at most six times per day for a period of 7-10 minutes. [26]

The captured image can be further processed by the on-board processor. It is possible to calculate and download energetic histogram (when the sensor is operated in time-over-threshold mode) and also projection of a full resolution image to x and y image axis. Since downloading all captured images in full resolution is not possible due to the restricted bandwidth, the on-board microcontroller supports image compression, which was designed specifically for the Sun observation with the X-ray telescope.

This compression is called binning, which (in image processing terminology) means that we merge information from neighboring pixels to create a new image with lower resolution. In our case, we perform the binning procedure in two steps. First one is division of the full resolution image into non-overlapping blocks (of size $n \times n$ pixels). In the second one we count nonzero pixels in each block and form new image from these counts. There are 3 binning modes. Mode called Binning 32 works with blocks of size 32×32 pixels and produces image with 8×8 pixels. Size of blocks in Binning 16 is 16×16 pixels, so the resulting image size is 16×16 pixels. Most detailed is Binning 8 mode with blocks of size 8×8 pixels, which produces images of size 32×32 pixels. Data produced by post-processing routines of the satellite are presented in figure 2.9.

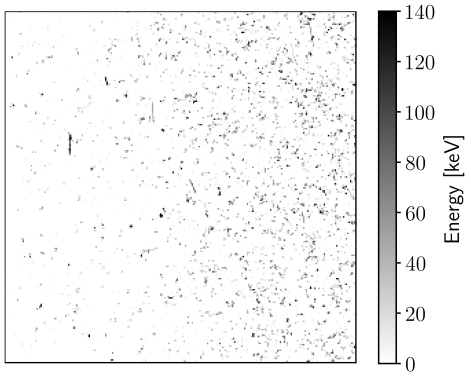
Image acquisition can be triggered manually or automatically. Manual configuration (the so-called Simple imaging mode) is possible via scripting. Another possibility is to use the Scanning mode, which is similar to manual configuration, but it differs in that it saves the frame only when pixel count exceeds some predefined threshold. Third, Adrenaline mode, is fully automatic. It uses narrow-field-of-view UV sensor to trigger exposure when the Sun is present in a frame. Adrenaline mode is accompanied by a method to determine trigger threshold automatically.

Moreover, simple on-board data filtering is possible. On-board processor is equipped with routine able to remove all clusters of active pixels with area larger than one pixel. Filtering algorithm is based on analysis of connected components and removes all 4-connected pixels. This is useful for use with the X-ray telescope, since X-ray photons often leave only 1-pixel long track. Since this work does not use the telescope, filtering was not enabled.

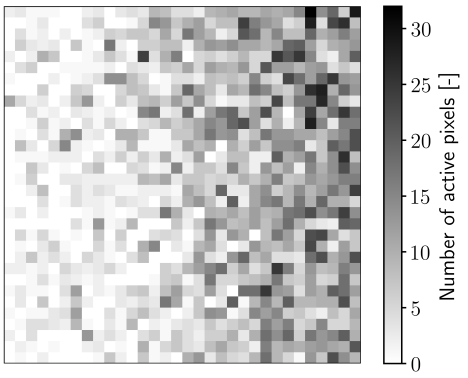
Exposed frames are then processed by the on-board computer of the satellite and send to the Earth as a stream of 64 B packets with download speed of up to 1 kBit/s [2]. Each day of operation, the satellite is able to transmit approximately 100–200 kB of data [27]. This transfer capacity is shared between multiple on-board instruments, so the Timepix gets only a part of it.

2.4.2 Downloaded data format

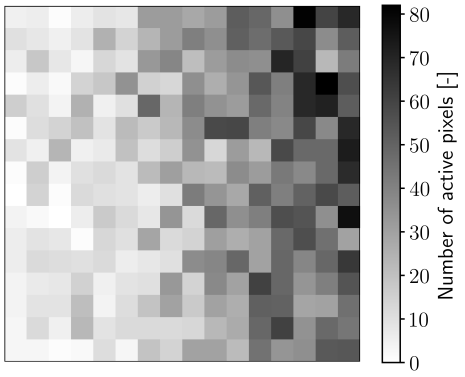
Downloaded data are available in binary serialized format in a form of a Python class. This serialized representation can be deserialized with the pickle Python module. Each downloaded frame is saved in single file. Each file has unique number identifier which allows identification of image among all other images captured during the mission.



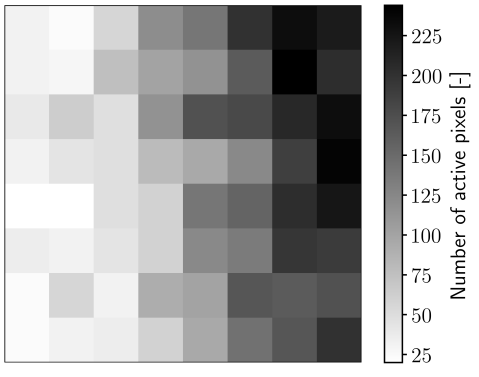
(a) Full resolution



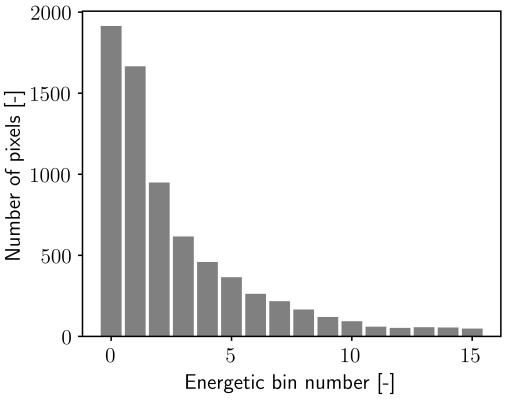
(b) Binnig 8



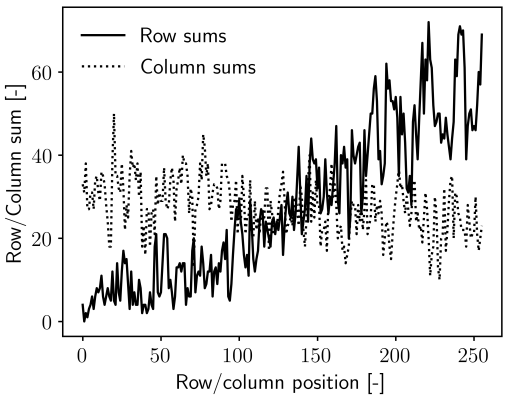
(c) Binnig 16



(d) Binnig 32



(e) Energetic histogram



(f) Row/Column sums

Figure 2.9: Examples of data which can be sent to the Earth from the on-board Timepix sensor, calculated from full resolution image (upper left).

Chapter 3

Data management

During work on this thesis, the Timepix sensor on the VZLUSAT-1 satellite has exposed 19492 frames in various compression levels, which were downloaded to the Earth and are available for processing. Every downloaded frame contains metadata. Almost all frames also carry image data, which can be of four types with different compression applied. The satellite can also append energetic histogram and row and column sums of original full resolution image to the frame. Moreover, significant amount of data is produced by the preprocessing and segmentation (described in the chapter 4) – segmentation of downloaded images resulted in more than 100000 segments.

Due to the amount of data, a set of tools was developed to allow user-friendly data access, filtering, and viewing. In particular, processing that this thesis deals with includes:

- **incremental storage of incoming data:** new data are downloaded from the satellite regularly and should be seamlessly incorporated into existing dataset,
- **handling of different data types:** data frames with various on-board preprocessing settings (such as different compression levels),
- **storage of intermediate data:** all the incoming data are preprocessed before they enter the reconstruction pipeline. This preprocessing is computationally expensive and it is not necessary to perform it every time the data are processed by the higher level functions developed in this thesis (such as classification),
- **visual inspection of results of preprocessing:** mainly for debugging purposes,
- **support for supervised classification:** the classifier designed in this thesis needs annotated training data as input. Since it is not possible to obtain relevant automatically labeled training dataset (e.g., from known laboratory radioactive sources or from simulations) in our case, the labeling of examples is performed by hand. At least few hundreds examples for each type of detected particles are needed for training of a classifier, so a fast and user-friendly interface is needed.

This chapter describes solution of all of these tasks, which is divided to three parts:

- Low-level data management (database backend)
- High-level database and preprocessing interface
- Graphical user interface

3.1 Database backend

The problem of ionizing particle classification is not much covered by literature. There is only a little information on how classificator for this special domain should be designed, so the proposed data manager should provide near real-time access to the data for exploration and

rapid prototyping. It must be reliable and allow data filtering. Since we are dealing with large amount of data, it is needed to effectively handle situations of RAM overflows.

Three possible solutions were examined:

1. list-based approach, where each item is stored as an instance of its own class and is accessible with iterations,
2. relational approach with Pandas dataframes, data structure from popular machine learning Python module,
3. relational approach with SQLite database.

All of these approaches were implemented and tested. The only one, which satisfied declared specifications, was SQLite database, which was developed further to the fully functional state. This solution uses two tables to organize the data (see figure 3.1). The *Images* table represents the data packages coming from the satellite (mainly containing image information), the *Segments* table represents the particles detected in the images. Each trace of a particle in the *Segments* table is linked to its parental image from the *Images* table via a foreign key, the image id.

Database tables contain only elementary metadata that are linked to data frames coming from a satellite or generated during processing of received data on the Earth. Larger amounts of data, such as images and histograms, are stored in an external folder and the database contains only relative reference to their location. As a result, the database file is small and can be shared between users. It is also possible to alter the referenced data without having to fundamentally change the structure of the database.

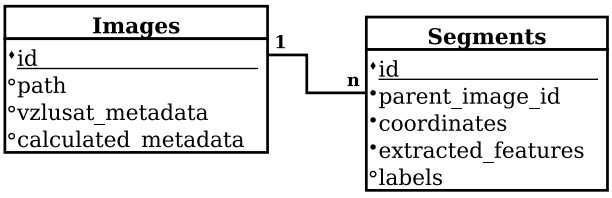


Figure 3.1: Simplified diagram of database designed for storage of satellite data. All attributes in the diagram actually do not represent attributes in implemented solution, which contains more (but for explanative purposes irrelevant) information.

The diagram in figure 3.1 shows a simplified version of the implemented tables. Both tables have attribute id, which serves as a primary key and unique identificator in the whole processing pipeline even outside the database. Images are stored in a folder relative to the database file (the "path" attribute). Each image downloaded from the satellite carries metadata, which were added by the on-board computer. These are stored as attributes in vzlusat_metadata group. Last type of metadata are metadata calculated during processing on the Earth, like latitude and longitude of the captured frame.

Each image is partitioned into segments. Each segment contains only one type of particle track. Information about these segments is stored in the *Segments* table. Again, bitmap data of the segments are not stored directly in the database, which contains only a reference to particular source images and segment coordinates in this image. Table with segments also contains information needed for classification: labels (if any) and extracted features (since feature extraction is in our case significantly computationally more expensive than classification).

3.2 Higher level interface

The database backend supports only low-level SQL control. This allows for very detailed control, but also brings a risk of database corruption when used incorrectly. It also increases the

complexity of often repetitive data manipulation tasks. An interface was created that allows the user to perform the usual data management tasks without having to use SQL. The interface is based on the Data Manager class that encapsulates the database and connects it to other functionalities implemented. It is capable of creating the database, retrieving the data from the serialized binary files 3.2 and converting them to a standard format in which these are represented in the whole pipeline. Data Manager can also alter the database to reflect changes which were performed during preprocessing or in graphical user interface. It also filters the data according to several criteria and calculates geographical data from time stamps with the use of TLE (detailed description available in the section 5.4.1). The Data Manager ensures that the database is not corrupted by some inappropriate query when being accessed by common operations provided by its interface. Nevertheless, the data can be accessed directly using SQL if necessary.

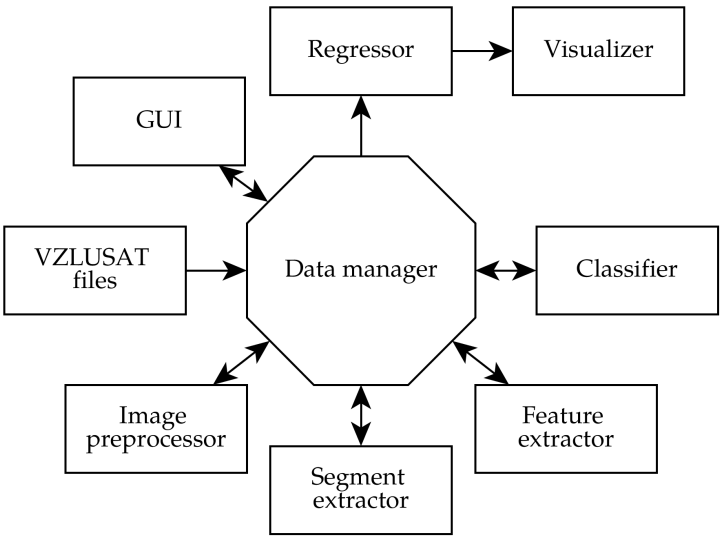


Figure 3.2: Diagram of the implemented solution.

The whole developed solution consists of functionally independent modules, which are centralized around the data manager which serves as a hub for data exchanges (which are performed in batches when processing large amounts of data). The core database functionality was implemented with the use of a SQLite DBMS, which is sufficiently fast for this type of work (it is able to perform thousands of insert operations per second on average desktop computer) and with its self-contained nature it can be used on every system where the important software libraries are present (e.g., large-scale computing cluster).

3.3 Graphical user interface

The highest level data management tool is the graphical user interface. Thanks to it, it is possible to carry out the usual repetitive tasks with the database and to interactively browse the database, which is essential, especially for the data analysis phase and the subsequent formation of the training set. Although the widest functionality is available through command line and scripting, the GUI provides the easiest access to the most commonly performed tasks such as inserting new images into the database and updating it when the pipeline that processes the images is changed.

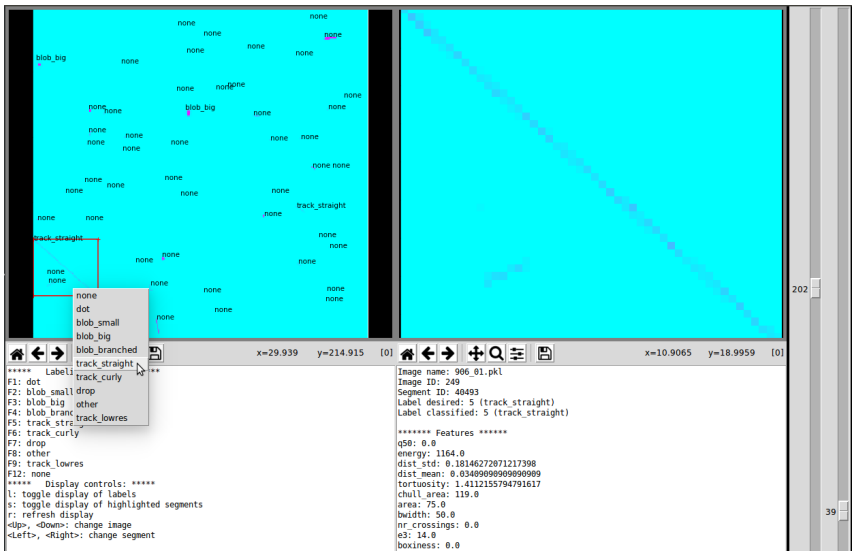


Figure 3.3: Tool for detailed inspection of data contained in the database.

However, the greatest use of the user interface is when selecting dataset for a training of the classifier – that would not even be possible without it. Two tools were created for this purpose. The first one (figure 3.3) allows the user to view the images in the database with detailed information, including segments detected in images. The segments can be annotated here. The second tool (figure 3.4) is intended purely for labeling of large amounts of particles and for qualitative evaluation of classification results.

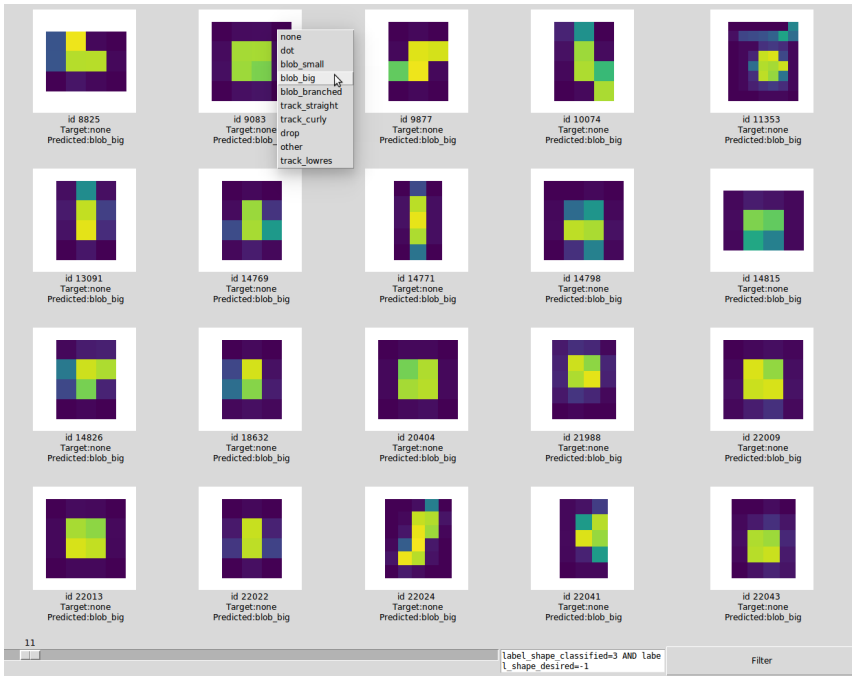


Figure 3.4: Graphical interface for fast labeling of training data.

Chapter 4

Data classification

The system for classification of particle traces was developed on top of the data manager. The purpose of this system is to find and classify all the particles present in full resolution images. Then these classified particles can be counted in each frame and used for training of regression model, as described in section 5.3.

Traditionally, image classification system is a pipeline of interconnected modules, which performs data preprocessing, feature extraction and, finally, classification. Preprocessing steps are commonly designed manually using some knowledge about the field of application. With carefully designed pipeline, amount of data needed for training can be smaller, because the designer can incorporate domain-specific heuristics about the particular problem into the solution.

Another approach that is popular in the recent years is deep learning, which uses neural networks with many hidden layers, which skips all the pre-classification steps and merges whole classification estimator into one black-box. It proved to be effective for learning from big amounts of data. The general architecture of a neural network is still created manually, but then it is automatically tuned for a particular problem during training. Unfortunately, this approach needs a big amount of data (an example is famous neural network ImageNet, which was trained on 1.2 million images [28]). So, since we have training dataset in a size in the order of hundreds of particle traces and we are dealing with a very specific problem, a feasible solution is to use a traditional approach with manually created pipeline as depicted in figure 4.1.

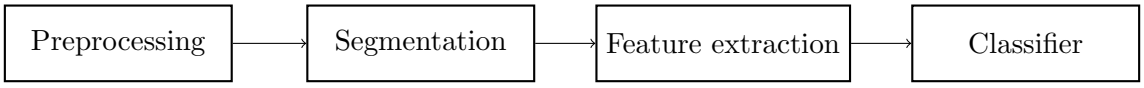


Figure 4.1: Schematics of the classification pipeline

Preprocessing steps involve e.g., pixel value calibration or filtering (such as noise removal and other techniques for image enhancement). Segmentation then divides the image into smaller regions, segments, where, in our case, each segment represents an indivisible entity with exact meaning, which can be described by a class label. Each segment is mathematically described during feature extraction. The goal of this step is to represent each segment as a point in n-dimensional feature space, optimally in a way which allows mathematically simple division of this space into regions occupied only by a set of segments belonging to one classification label. Finally, the classification algorithm is supposed to find boundaries between regions occupied by a single class instances. When found, these boundaries can be then used for prediction of labels of unknown samples.

4.1 Dataset analysis and classes proposal

Data for classification are available in form of single-channel bitmap images with size of 256×256 pixels (as shown in figure 4.3b). Each pixel carries information about energy dissipation in its volume during travel of ionizing particle through the sensor. Energy values from interval from 0 to 140 keV are uniformly quantized into 256 discrete levels by processor in the satellite. It is possible to identify large amount of particle traces of various sizes and shapes on available images. The amount is larger then in common laboratory experiments with defined radioactive sources and no source was found in the state-of-the art literature, which directly describes classification of such images from LEO experiments.

There is a visible inhomogeneity in the captured frames: different particle traces can be seen in frames captured in different places in orbit and in different times, some traces clearly stands out as a class on their own, but are very rare. Classification task also heavily relies on a correct exposure of captured frame. The overexposed frames contain overlapping traces, which are hard to classify correctly. Moreover, many traces in the dataset are very small, with absolute size maximally in the order of hundreds of micrometers. With 55 μm pixel pitch of the Timepix detector, this means size only a few pixels.

The relatively low resolution of the sensor can cause severe aliasing artifacts and kind of a polymorphism of particles appearance. One particle with defined energy and behavior can leave dramatically different traces when penetrating the sensor from different angles. This effect is illustrated in figure 4.2 for (idealized) particle which travels through the sensor material always in a straight direction (e.g., muons).

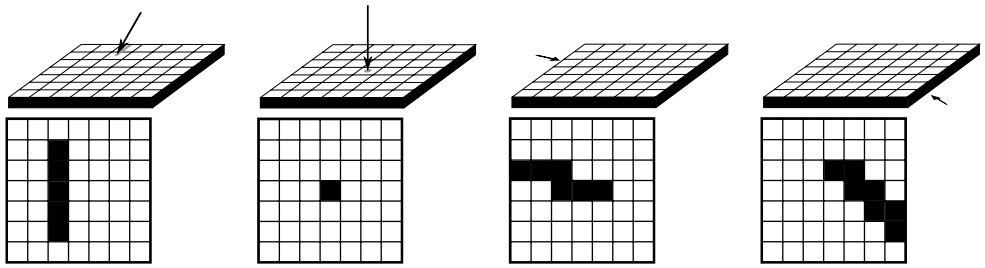
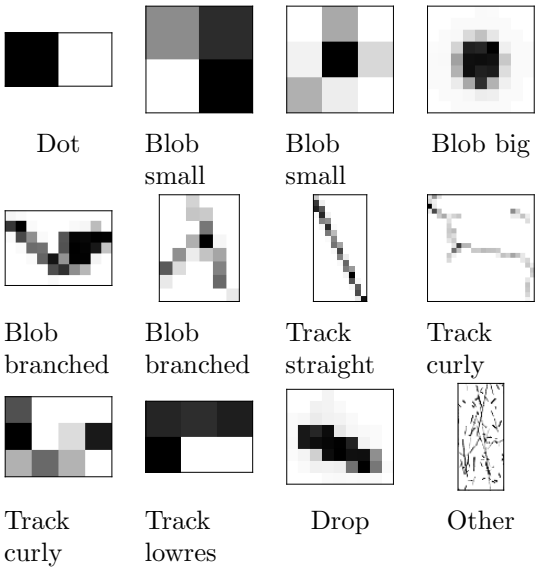


Figure 4.2: Idealized examples of traces of particles of the same type with different angles of arrival. Although all these particles represent particle of a single type, the one which is never deflected and always travels along the straight line through the sensor, resulting traces are very different. Charge sharing effect is neglected.

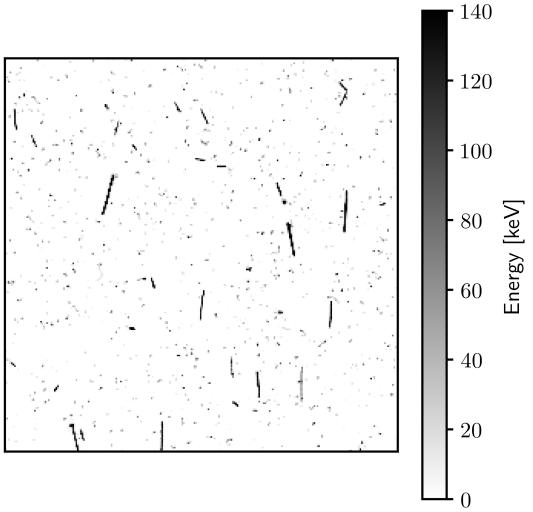
The state-of-the art survey about classification of particles from pixel detectors resulted in finding that this topic is scarcely covered by commonly accessible literature and the few results that addressed it are focused on laboratory experiments, where the particle sources (and often even angles of particle arrival) are known. Two main approaches are:

- classification into classes with exact physical meaning [10], [9],
- classification into classes based on geometrical properties of tracks [29], [11].

Since our data have large variability and we often do not know the origin of particular tracks from the point of view of particle physics, we loosely followed approach proposed in [29] with modifications to account for more detailed description of our data. Six classes were introduced in the original article (dot, small blob, heavy blob, straight thin track, heavy track and curly track). Examples for all of these classes can be found in our dataset, some particle traces share properties of more than one of these classes. For these particular traces, it is hard to recognize the correct class membership. Also, it was decided that each class should contain particles with



(a) Examples of identified most frequent types of particles



(b) Example of orbital image from Timepix sensor

Figure 4.3: Example of different types of particle trajectories observable on orbital pixel detector data (left) and a raw image from Timepix sensor (right).

similar geometrical structure and classified particles can be then (if necessary) filtered according to the attributes (e.g., length, energy) with exact geometric descriptors unique to the particular class. Following classes were proposed from these assumptions (examples can be seen in figure 4.3a):

- **dot:** one or two active pixels,
- **small blob:** small cluster, mostly contains between 4 and 9 active pixels, sometimes hard to describe parametrically because of the small size,
- **big blob:** larger than the small blob, always centered, higher energy pixels concentrated in the center of the cluster,
- **blob branched:** cluster of pixels from which one or more tracks originate,
- **track straight:** simple straight line, must be sufficiently long to be distinguishable from not being curly,
- **track curly:** similar to the track straight, but curly, it must be sufficiently long so the curliness can be distinguishable from aliasing,
- **track lowres:** all track-like particles, but so short that their shape (curly, straight) cannot be clearly distinguished due to the aliasing,
- **drop:** track with similar energy content, but short, shape similar to deltoid (on one end it is wider),
- **other:** rare event or error of measurement (e.g., overlap of particle traces on overexposed frame).

4.2 Preprocessing

As stated before, the goal of image preprocessing is to enhance an image in a way which makes classification more accurate. It is desirable to keep this step on a bare minimum in our case, because pixel values have exact physical (energetic) meaning, which is an important descriptor of a particle type.

Since it can be assumed that images are almost noiseless (which is guaranteed by setting a

sufficiently high threshold of the Timepix sensor operating in the ToT mode), noise filtration is not needed. The only necessary step is recovery of energy values. Since every pixel of the Timepix image contains 8-bit values and value 255 corresponds to energy 140 keV, denormalization can be performed as follows:

$$A_r(i,j) = \frac{140}{255} \cdot f_{ij}(A_{origl}(i,j)), \tag{4.1}$$

where A_r is denormalized image with pixel energies in the range of kiloelectronvolts and A_{orig} is image with pixel values in range $\langle 0; 255 \rangle$. Function f_{ij} is a calibration curve, specific for each individual pixel. This step is performed by the software which receives and converts the raw data to binary files, so the exact mechanism is not covered in this thesis – but it can be found in [23].

4.3 Segmentation

Segmentation is operation which divides an image into a set of regions with more abstract meaning than pixels alone. Methods for image segmentation can be divided into two main groups according to their core principle [30]:

- layer-based segmentation
- block-based segmentation
  - region-based segmentation
  - segmentation based on edges and boundaries

Layer-based segmentation uses a bank of object detectors to find segment shapes and classes. Block-based segmentation relies on methods like clustering (where segments are discriminated by clustering algorithm according to some chosen metric of similarity), region growing (starts with few initial pixels and adds its neighbors, again according to similarity), thresholding (gray-level threshold in grayscale image is selected and then used for conversion of this image to a binary one). There also exist region based techniques based on graph theory or segmentation algorithms which rely on edge detectors as their core principle.

The threshold of our Timepix sensor was set to a value which makes safe to assume that images produced by the sensor are free from electronic and thermal noise and all pixel activations are caused by incoming particles. Because of this assumption, the segmentation can be based on thresholding and is done in three consecutive steps:

1. thresholding,
2. connected component labeling,
3. segments extraction.

Thresholding

Conversion from image A to binary image A_b is performed as:

$$A_b(i,j) = \begin{cases} 0 & \text{for } A(i,j) = 0 \\ 1 & \text{for } A(i,j) > 0, \end{cases} \tag{4.2}$$

where $A(i,j)$ and $A_b(i,j)$ are pixels with coordinates i, j in real and binarized image.

Connected components labeling

Resulting binary image contains particles separated from the background, but all of them have the same pixel values. In this step it is not clear which particles form clusters. To extract each

cluster of connected pixels, connected component labeling is performed. This step is based on the idea of neighborhood, which is used for definition of connectivity. The 8-neighborhood is used in our application:

Definition 4.3.1. *Let A be a two–dimensional binary image and let A(i,j), A(k,l) be two (different) pixels. Then pixel A(k,l) lies in a 8-neighborhood of a pixel A(i,j) when k = i-1 or k=i or k=i+1, and l=j-1 or l=j or l=j+1.*

The concept of neighborhood is important for the definition of 8-connectivity:

Definition 4.3.2. *We say that pixels $A(i, j) = a_1$ and $A(k, l) = a_n$ are 8-connected, when there exists a path between them consisting of pixels $a_1, a_2, \ldots, a_n$ such that for $1 \leq m \leq n - 1$ pixels a_m, a_{m+1} are in 8-neighborhood.*

Task of finding these connected components in an image can be formulated as disjoint set union problem as described in [31]. In this problem we have disjoint sets of objects, which are initially spanned by one-element subsets. Each of these subsets is formed by one pixel. Moreover, two operations are defined: union and find. Union fuses two disjoint subsets into one and find returns type of the subset of which given element is a part of it. The goal is to apply sequences of union and find operations with best possible time complexity. An already implemented algorithm, which is a part of the scikit-image library, was used in this thesis.

Extraction of particle trajectories

Each labeled connected component is cropped from the image for further processing. Size of the crop is determined by the minimal rectangular hull (parallel to a pixel grid) of a given segment. These hulls are constructed and used for segment extraction as described in the algorithm 1. Result of the segmentation process can be seen in figure 4.4.

Algorithm 1 Rectangular hulls extraction

```
labels ←findUniqueLabels()
segments ←emptyList()
tempImage
for each label in labels do
    pixelCoords ←getAllPoints(label, image)
    topLeft ←[min(pixelCoords.rows), min(pixelCoords.cols)]
    lowLeft ←[max(pixelCoords.rows), min(pixelCoords.cols)]
    lowRight ←[max(pixelCoords.rows), max(pixelCoords.cols)]
    topRight ←[min(pixelCoords.rows), max(pixelCoords.cols)]
    tempImage ←crop(image, topLeft, lowLeft, lowRight, topRight)
    for each pixel in tempImage do
        if pixel ≠ label then pixel = 0
        end if
    end for
    append(segments, tempImage)
end for
return segments
```

4.4 Feature extraction

Each extracted particle trace has different dimensions, from 1×1 to, theoretically, 256×256 pixels. The goal of feature extraction is to describe each segment by n numbers, thus represent

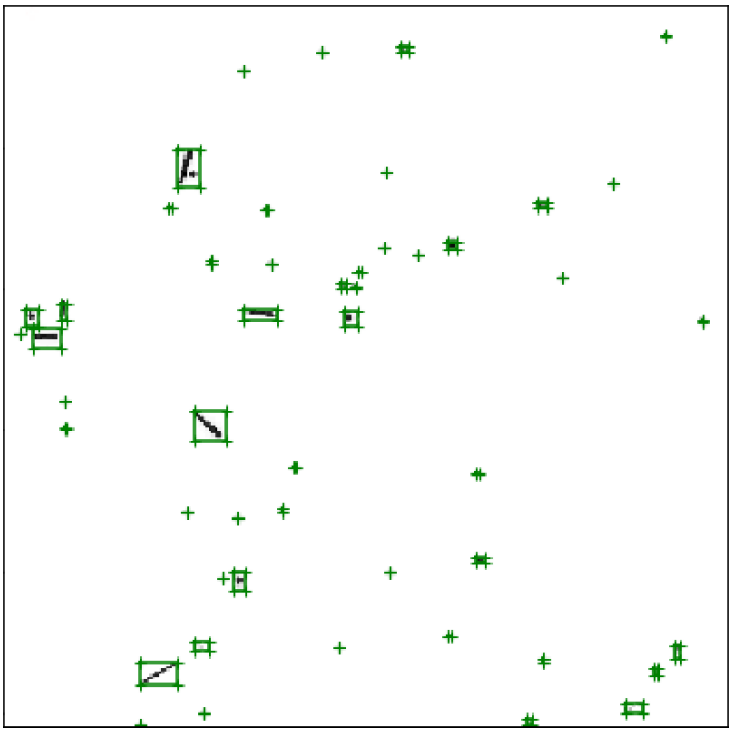


Figure 4.4: Example of segmented full resolution image. The green rectangles are identified rectangular hulls of detected segments.

it as a point in n-dimensional space. 25 features, which are further described, were proposed. These features can be divided into four groups:

- features based on pixel values,
- features based on description of geometry of tracks,
- features based on morphology,
- energetic histogram.

Before the description of each feature extractor, the concept of mathematical morphology, which is an important tool for some of the developed feature extractors, is introduced.

4.4.1 Mathematical morphology

Mathematical morphology is a tool used for analysis of spatial objects, discrete or continuous. It is often used in digital image processing [32], both for binary and integer-valued images. In our case, we restrict ourselves to binary mathematical morphology, where images are described as sets of ordered pairs of 2-D coordinates of active (that is, foreground) pixels. Between these two sets, A and B, morphological operations of dilation and erosion are defined. Let dilation between A and B be marked as $\oplus$. Then this operation can be defined as:

$$A \oplus B = \{a + b | (a \in A, b \in B)\} = \bigcup_{b \in B} A_b, \tag{4.3}$$

where A_b is set A translated by b. Erosion operation $\ominus$ for sets A and B is defined as

$$A \ominus B = \{a | (a + b) \in A, \forall (a \in A, b \in B)\} = \bigcap_{b \in \check{B}} A_b, \tag{4.4}$$

where $\tilde{B}$ is a reflection of a set B, that is

$$\tilde{B} = \{(-x, -y) | (x, y) \in B\}. \tag{4.5}$$

One of images of A and B, used for morphological image analysis, is usually relatively small compared to the another one and it is used as a probe, called the structuring element. Dilation is able to fill in holes between objects in image. Foreground objects grow under application of dilation. Erosion makes foreground objects thinner, small foreground objects are removed. Both of these operations changes shape of objects.

On top of operations of dilation and erosion, opening $\circ$ and closing $\bullet$ operations are defined. For the opening between A and B we can write:

$$A \circ B = (A \ominus B) \oplus B, \tag{4.6}$$

similarly for closing:

$$A \bullet B = (A \oplus B) \ominus B. \tag{4.7}$$

Opening separates objects and makes their shape simpler, but overall size is not changed. Closing merges nearby objects and fills in holes.

Important morphological operation is also hit-and-miss transform $\odot$, defined on binary image A with structuring element B as:

$$A \odot B = (A \ominus B_1) \cap (A^c \ominus B_2), \tag{4.8}$$

where A^c is a set complement to image A and B_1, B_2 are structuring elements from which element B is created. In comparison with the previous operations, B contains both foreground and background pixels and B_1 with B_2 are subsets of pixels of a same type.

Mathematical morphology can be also used for extraction of image topological skeletons. Skeleton is a simplified shape, which is created by throwing away unimportant information from an object on image, e.g., thickness and pixel values. Possible approach for finding skeleton is morphological thinning, where morphological operations are used for iterative removal of pixels on the borders of objects until these objects are 1-pixel thin.

4.4.2 Description of features

This section describes features extracted from the images. The feature extractors which share their core principles are grouped together where possible.

Elementary features

Five features are calculated directly from image sensor data:

- area: number on nonzero pixels in extracted rectangular segment,
- energy: sum of values of pixels of a track,
- energy per pixel: ratio of area and energy,
- energetic quantiles: lower decile, upper decile and median of energies found in cluster,
- width and height of a rectangular hull of segments.

Although these features are trivial, they are expected to be important for classification since they directly encode physical properties of detected particles.

Area and occupancy of a convex hull

According to the [33], convex hull of a set of vectors $S = \{x_1, ..., x_n\}$ is the smallest convex set in $\mathbb{R}$ that contains the vectors $x_1, ..., x_n$. In our case, S is a set of all active pixels of a given particle trace. Then the convex hull forms a convex 2D polygon. Since area of a convex hull is dependent on a scale of a trace, occupancy o_{hull} is also computed as

$$o_{hull} = \frac{n_{active}}{n_{hull}}, \tag{4.9}$$

where n_{active} is number of nonzero pixels of a given track and n_{hull} is number of pixels which form convex hull.

Linearity

Active pixels form a point cloud in a two-dimensional space. Since both axes are independent variables, linear regression alone cannot be used. Instead, a 2D ellipse is fitted on this point cloud and its semi-major and semi-minor axes are analyzed.

In particular, lets assume that X is a tuple of x-coordinates and Y tuple of y-coordinates of pixels, both are centered, i.e., their mean is zero. Let the covariance matrix between X and Y be Σ. Then principal component analysis can be performed, e.g., by extraction of eigenvalues of Σ. These eigenvalues correspond to variance of the point cloud explained by a corresponding eigenvector. The larger the eigenvalue, the larger the axis of a fitted ellipse, which can imply more linear shape in this direction.

Let the eigenvalues of a point cloud formed by active pixels be e_1, e_2. Then the linearity can be defined as:

$$linearity = \frac{\max(e_1, e_2)}{e_1 + e_2} \tag{4.10}$$

or:

$$eigRatio = \frac{\max(e_1, e_2)}{\min(e_1, e_2)}. \tag{4.11}$$

Both of these features are calculated.

Number of crossings

This feature contains information about number of pixels in which skeleton of the particle is branched. Calculation is carried out with the use of graph theory:

1. skeletonization of a binarized segment,
2. conversion of the skeleton to a graph, where each node represents a pixel, edges are created between 8-connected pixels,
3. removal of cycles: each edge is weighted, diagonal edges are penalized, minimum spanning tree (MST) is found,
4. degree of each node in MST is found: sum of degrees larger than 2 is the number of pixels where the trace is branched.

Skeleton to convex hull area ratio

This feature is calculated as

$$SCHR = \frac{n_{skeleton}}{n_{hull}}, \tag{4.12}$$

where $n_{skeleton}$ is number of pixels of which consists skeleton of a track, n_{hull} is number of pixels in its convex hull.

Tortuosity

Tortuosity is a measure of "curliness" of a curve. Possible way to define tortuosity t is by the ratio

$$t = \frac{l_{curve}}{l_{ends}}, \tag{4.13}$$

where l_{curve} is length of a curve and l_{ends} is a straight distance between its start and end. For calculation of the length of the curve, the track is skeletonized and connected pixels are represented as a graph. The length of a curve was then defined as a longest path in this graph.

Distance transform measures

The distance transform is a technique which labels all pixels of a foreground cluster in an image with their distance from background pixels. According to [32], the distance transform can be mathematically defined as a solution to the differential equation

$$||\nabla DT(x)|| = 1 \tag{4.14}$$

with boundary condition $DT(x) = 0$ for $x \in \delta R$, where δR is a boundary of the blob of active foreground pixels and x is a 2D vector of coordinates. According to [32], solution to this equation is given as:

$$DT(x) = \min_y \in \delta R||x - y||. \tag{4.15}$$

One of the ways to obtain distance transform is to perform repetitive morphological erosions of a given object, but in this case the results are dependent on the structuring element shape and the results are not interpretable as Euclidean distances to the borders. For overcoming this problem there are multiple definitions and algorithms for distance transform calculation. Algorithm which labels each pixel with Euclidean distance to its nearest trace border was used in this feature extractor. The algorithm was taken from the SciPy library[1] [34]. The image transformed with the distance transform was used for calculation of mean and standard deviation of distances. These statistics were used as features.

Boxiness

It was found that many particle traces of the class "Blob branched" often contain one or more clusters of pixels in a shape of a box. These boxes are connected by thin curly or straight lines of active pixels. To further discriminate class Blob branched, the feature which counts amount of these boxes in each segment is calculated with the use of morphological operators.

First step in this calculation is disconnecting of these box elements. This is achieved by erosion with structuring element s with a following structure:

$$s = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}. \tag{4.16}$$

After this step the segment contains separated islands of active pixels. On these islands the connected component analysis is performed once again and the number of separate connected components is returned.

[1] The particular algorithm implemented by this library is not known.

Foreground connectivity with background

The pixel at the foreground cluster boundary can be adjacent to the background with one, two, three or four of its edges. This set of features uses a series of morphological hit-and-miss transforms to detect and count these pixels. For each possible number of touching background-foreground edges the number of pixels is returned.

Diagonality and straightness

Diagonality and straightness are features designed as one of the possibilities for discrimination between curly and straight tracks. Both are based on morphological opening with 2×2 structuring element followed by subtraction of resulting image I_1 from the original image I_0. Finally, another opening is performed, this time with structuring element s_2:

$$I_1 = (I_0 - (I_0 \circ s_1)) \circ s_2. \tag{4.17}$$

Structuring element s_2 consists of two diagonally placed pixels (for diagonality) and two horizontally placed pixels (for straightness). Since diagonals and also straight lines can be oriented in two directions, these structuring elements are applied two times, each time with a different orientation. After the application of the morphological operations, the active pixels of the resulting image are counted and the value of straightness and diagonality (summed for structuring elements with both possible orientations) is returned as a result.

4.5 Feature processing

Extracted features are further processed before using them as input for the classification algorithm. This processing is carried out in order to make the classification less susceptible to possible outliers in a dataset and to emphasize characteristics of the data which are important for a given task. The most suitable processing steps depend on a particular classification problem. This section describes two processing steps which were experimentally identified as important for our problem.

4.5.1 Feature preprocessing

Feature preprocessing involves mathematical alterations of numerical feature values with the goal of enhancing ability of features to describe the underlying modeled process. Common operations are logarithm transformations of existing features to make skewed distributions more similar to the normal distribution, quantile transforms and scaling. It is also possible to calculate new features based on nonlinear transformation of original features. It was found that for the given task the only tested estimator which benefits from the nonlinear (in particular polynomial) transformation is the logistic regression.

All inputs to the classifier were also scaled. Feature scaling is operation, which maps values of different features to similar ranges. This operation has meaning only in a case when different features have different scale and these different scales are not important for solution of a given classification problem. Some classification algorithms also implicitly assume that their input is standardized to work properly. In general the scaling methods are based on properties and statistics of a particular feature X in a training dataset. Commonly used scaling method is a standardization to unity variance and unity mean with the use of a formula:

$$x_{new} = \frac{x_{old} - \bar{x}}{\sigma}, \tag{4.18}$$

where x_{old} is the original value of a feature for some sample, $\bar{x}$ is mean of all values of the feature X in a training dataset and σ is standard deviation of this feature in training data. When the feature X has a normal distribution, after scaling we have a guarantee that with the probability of 68% value of x_{new} is in a range $\langle -1, 1 \rangle$ [35].

4.5.2 Feature selection

Feature selection is a crucial step in design of classifiers. Methods for feature selection can be divided into three classes [36]: filter, wrapper and embedded methods. Filter methods assign scores to features and than select only those features, whose scores are higher then a predefined threshold. These scores are of a statistical nature, e.g., feature variance in training dataset, correlation or information criteria.

Wrapper methods assess usefulness of a feature subset for a given predictor. These methods are based on training of predictor for many subsets of features. A score of prediction is calculated every training step. Finally the feature subset with the best score is selected.

Embedded methods are methods, which are based on internal mechanism of a predictor. These methods are mostly specific to a given learning algorithm. These are similar to wrapper methods in that they try to return best features specifically for a given predictor.

Embedded methods were employed in the final version of two classifiers – the one based on random forest and the one based on logistic regression. In a random forest the feature selection is embedded in algorithm which constructs its basic unit – decision tree. Logistic regression was trained with the use of L1 regularization, which forces the regression coefficients to be sparse (which in turn behaves as a feature selector, omiting features multiplied by zero coefficients).

4.6 Classifier

This section describes design of a final classification stage. Since this thesis deals with the task of multiclass classification, several methods to deal with this issue are discussed. The considered classification algorithms are described and finally the whole model is assembled and verified.

4.6.1 Multiclass classification

The task of classification consists of finding decision function $f(x)$ which, for each sample x, returns class to which this sample belongs. Function $f(x)$ must return integer values in multiclass classification, as opposed to the more elementary binary classification problem where the range of this function is constrained to only two values. Problem of multiclass classification is greatly covered by literature, e.g., [37]. There are multiple approaches available for this problem. Some classification algorithms can be extended from binary to multiclass classification naturally (e.g., k-NN or neural networks). Unlike them, this thesis uses the fact that multiclass classification problem can be converted to a set of binary problems. To achieve this, more methods are possible, probably the most known ones are mainly:

- one-vs-rest,
- one-vs-one.

These methods can be used with binary classifiers which return confidence score for classified sample x, $g(x)$. The higher the score $g(x)$, the more certain the classifier is. One-vs-rest method relies on training n classifiers, for each class i one, which decides whether classified sample x belongs to class i or not. Individual classifiers vote (with their confidence scores) and the

resulting class $i^\star$ is chosen according to the equation:

$$i^\star = \arg\max_i g_i(x), \tag{4.19}$$

so that the sample x is assigned to the class whose classifier is most certain about its membership.

Unlike one-vs-rest, one-vs-one classification needs not only n, but n^2 classifiers. Each of these classifiers decides about sample membership in a pair of classes, i and j. Let the g_{ij} be confidence of classifier for which class i is positive detection and j negative. Then the estimated class $i^\star$ of sample x is decided as:

$$i^\star = \arg\max_i \left(\sum_j g(_{ij}(x)) \right). \tag{4.20}$$

4.6.2 Logistic regression

Logistic regression belongs to the group of linear classifiers. Its decision boundary is a linear function. The simplest variant of this model (used in this thesis) can be used for binary classification. Specifically, it models probability that particular object has label $Y = 1$ when we observe its attributes $\mathbf{x}$ with the model:

$$p(\mathbf{x}, \beta) = \frac{1}{1 + \mathrm{e}^{-(\beta_0 + \mathbf{x}\cdot\beta)}}, \tag{4.21}$$

where β and β_0 are parameters of the model [38]. When this probability is higher than 0.5, it predicts that object has label $Y = 1$. When it is lower, predicted label is 0. The decision boundary for logistic regression classifier is formed by all solutions of equation

$$\beta_0 + \beta \cdot \mathbf{x} = 0. \tag{4.22}$$

Since this is only a binary classifier, it was combined with the one-vs-one method to enable multiclass classification.

4.6.3 Random forest

Decision tree classifier

Decision tree is a nonparametric hierarchical model, which divides the feature space into regions. The tree consists of nodes, which represent decisions in a form of logical rules, outgoing branches of nodes represent different decision outcomes. These can be connected to another decision node, or to the leaf, which corresponds to the classification outcome. Important type of a decision tree is binary univariate tree, where each node makes decision based on univariate test $f(x)$ of some input variable x_j and the result is a binary value:

$$f(x) : x_j > w_{m0}. \tag{4.23}$$

According to result of this comparison the input space is split into subspaces L_m and R_m:

$$\begin{aligned} L_m &= \{\mathbf{x} | x_j > w_{m0}\} \\ R_m &= \{\mathbf{x} | x_j \leq w_{m0}\} \end{aligned} \tag{4.24}$$

When training this kind of tree classifier, the goal is to choose appropriate combination of splitting dimension j and weight w_{m0} for each node. Each leaf of a grown tree is associated with one hyperrectangle in input space. After training, each sample in this volume will by classified

as belonging to class associated with its leaf.

Problem of growing the tree from given training sample with restriction on maximally accurate classification of all training samples is called the tree induction. There are infinitely many trees with no classification error on training dataset, so constraint on minimum tree size (that is number of nodes and their complexity) is also introduced. Finding the optimal tree is a NP-complete problem, from which follows that in practice greedy methods are used, which looks for the best split for the actual node. Solution for the actual node splits the input (feature) space into subspaces, in which this approach is recursively repeated. There are more algorithms for tree growing, like ID3 [39] or CART.

Elementary pseudocode for tree construction for these algorithms can be written as:

1. start with the first variable in the root node and try to split the node at all values of this variable,
2. evaluate each split and remember the best one,
3. repeat this procedure for all remaining variables,
4. apply the best of all splits of all variables,
5. decide whether the node is terminal or should be split again,
6. repeat the steps 1–5 for all nonterminal nodes, until all nodes are marked as terminal.

From the decision tree to the random forest

The random forest algorithm is an extension of the decision tree algorithm which improves accuracy, stability and prevents overfitting [40]. Being ensemble method[2], multiple decision trees are generated during training of a random forest (each decision tree on its own training bootstrap sample D_i) and all of them make their predictions during prediction phase. All these individual predictions are then merged into the final one (e.g., by voting). The bootstrap training samples D_i of size n' are generated from the whole training dataset D of size n by uniform sampling with replacements. This whole method of training and combination of estimators is called bagging (or bootstrap aggregating).

4.6.4 Support vector machine

The support vector machine is, in its simplest form, a linear binary classifier [38]. The goal is to find hyperplane, defined as

$$\mathbf{w}^T\mathbf{x} + w_0 = 0, \tag{4.25}$$

which divides the input space with samples $\mathbf{x}$ into two parts, each occupied by one class of training examples. There is infinite number of such hyperplanes. Goal of the support vector machine is to find the one hyperplane that divides the space into two subspaces while passing exactly between two hyperplanes, which are as far as possible and still divide the space into two (from classification point of view correct) subspaces. Since distance of these two hyperplanes is given as:

$$d = \frac{2}{||\mathbf{w}||}, \tag{4.26}$$

the goal is to minimize $||\mathbf{w}||$ and still get correct classification results. This can be formulated as following optimization problem:

$$\begin{array}{rl} \text{minimize} & J(\mathbf{w}, w_0) \equiv \frac{1}{2}||\mathbf{w}||^2 \\ \text{subject to} & y_i(\mathbf{w}^T\mathbf{x_i} + w_0) \geq 1, \quad i = 1, 2, \ldots, N, \end{array} \tag{4.27}$$

[2]Methods which combine multiple estimators to improve performance of prediction.

where $\mathbf{x_i}, y_i$ are training pairs of feature vector and label. This formulation works only for perfectly linearly separable data. When this constraint is not fulfilled, the minimized function must be modified by inclusion of slack variables ξ_i, which expresses whether sample i located in margin (then $\xi_i \in \langle 0, 1\rangle$), is incorrectly classified ($\xi_i > 1$) or correctly classified ($\xi_i = 0$). With inclusion of these slack variables the optimization problem is rewritten as:

$$\begin{array}{rll} \text{minimize} & J(\mathbf{w}, w_0, \xi) \equiv \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{N}\xi_i & \\ \text{subject to} & y_i(\mathbf{w}^T\mathbf{x_i} + w_0) \geq 1 - \xi_i, & \\ & \xi_i \geq 0, & i = 1, 2, \ldots, N. \end{array} \tag{4.28}$$

This is still a linear classifier, but it can (even not maximally correctly) classify nonlinearly separable data. Nevertheless, it is possible to adapt SVM to classify nonlinear data by extending feature vector $\mathbf{x}$ to higher dimension and making it nonlinear. In the case of SVM this can be achieved more efficiently than just adding nonlinear combinations of existing features with the use of kernel methods.

4.7 Hyperparameter tuning and evaluation

All the previous information is put together in this section. Because of a large number of possible combinations of different feature preprocessing steps and classification algorithms, best candidates were chosen automatically with a grid search. These candidates were further tuned and validated. The available dataset was divided into training and testing subset (with a ratio $0.75 : 0.25$). The training dataset was used for training of the model and parameter search with the use of a 10-fold stratified crossvalidation. The performance of the model was evaluated on a testing dataset after training.

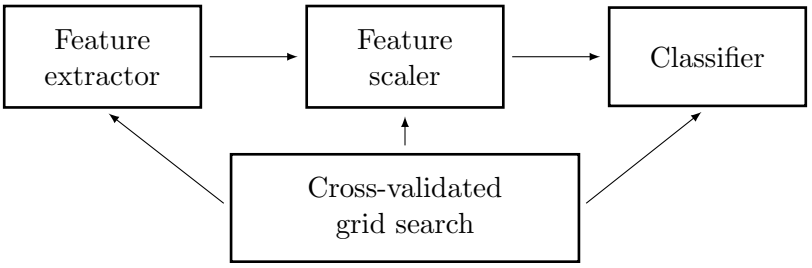


Figure 4.5: Classification setup for selection of best parts of the pipeline.

4.7.1 Gridsearch and scoring

The grid search is a procedure where a grid of parameters of the tuned estimator is defined. Every combination of these defined parameters is then tested and the classifier is scored with an appropriate metric of success. The combination with the highest score is selected as optimal for a given problem.

On a basis of empirical experiments, best three classifiers were identified. Different methods were automatically tried (and tuned) in a place of blocks in figure 4.5 for each classifier and best combinations were selected for each classification algorithm. Scoring was performed with the use of a brute-force search over a discrete grid of model parameters (that is, parameters of each function block in figure 4.5). 10-fold stratified cross-validation was used for scoring of each combination of parameters. This reduces risk of biasing the classifier toward one of the classes (which are unbalanced) and also reduces the risk of overfitting due to the small size of available dataset.

The metric chosen for evaluation of the pipeline performance is Matthews correlation coefficient (MCC), which is suitable for both binary and multiclass problems, even though its definition for multiclass problems is less intuitive. Its advantage is that it summarizes whole confusion matrix into a single number and it also gives reliable scores in a case of unbalanced classes [41].

Suppose that we have matrices $\mathbf{X}, \mathbf{Y}$ and $\mathbf{X}, \mathbf{Y} \in \{0,1\}^{S \times N}$. Both of them are filled with 1's and 0's according to the result of classification. Field of matrix $\mathbf{X}$, X_{sn}, is filled with 1 if sample s was predicted to belong to class n, zero otherwise. Field Y_{sn} is filled with 1 if sample s really belongs to the class n. Matthews correlation coefficient is then given as:

$$MCC = \frac{\mathrm{cov}(\mathbf{X}, \mathbf{Y})}{\sqrt{\mathrm{cov}(\mathbf{X}, \mathbf{X}) \cdot \mathrm{cov}(\mathbf{Y}, \mathbf{Y})}}, \tag{4.29}$$

where

$$\mathrm{cov}(\mathbf{X}, \mathbf{Y}) = \sum_{k=1}^{N} \mathrm{cov}(\mathbf{X}_k, \mathbf{Y}_k), \tag{4.30}$$

where $\mathbf{X}_k$ and $\mathbf{Y}_k$ is the k-th column of matrix $\mathbf{X}$ and $\mathbf{Y}$. Values of MCC lie between -1 and 1, where 1 means perfect classifier and -1 bad classifier. In a binary case the meaning of MCC is even more explanative, since it is also known that $MCC = 0$ means that classifier performs in a same way as a random classifier. This meaning is lost in our multiclass case.

The best identified parameters of the estimators are summarized in tables 4.1, 4.2 and 4.3. Corresponding scores are in table 4.4. Clearly the best estimator is the one based on a random forest, which was chosen as a solution to the classification task.

Part of the pipeline	**Variants**	**Parameters**
Feature preprocessing	Polynomial combinations	degree 2
Scaling	Standardization to $\mathcal{N}(0,1)$	-
Multiclass extension	One-vs-one	-
Classifier hyperparameters	Regularization penalty	L1
	C (inv. reg. strength)	1.6

Table 4.1: Optimized parameters of logistic regression.

Part of the pipeline	**Variants**	**Parameters**
Scaling	Standardization to $\mathcal{N}(0,1)$	-
Multiclass extension	Not needed	-
Classifier hyperparameters	Number of estimators	240
	Maximum tree depth	∞
	Minimum samples per split	5
	Minimum samples per leaf	1

Table 4.2: Optimized parameters of random forest.

Part of the pipeline	**Variants**	**Parameters**
Scaling	Standardization to $\mathcal{N}(0,1)$	-
Multiclass extension	One-vs-one	-
Classifier hyperparameters	C	4
	Kernel	linear

Table 4.3: Optimized parameters of support vector classifier.

Estimator	**CV test (MCC)**	**CV train (MCC)**	**Test score (MCC)**
Random forest	0.83216 ± 0.020	-	0.7905
Logistic regression	0.8115 ± 0.0228	0.8755 ± 0.0039	0.7438
SVM	0.8060 ± 0.0236	0.8483 ± 0.0047	0.7689

Table 4.4: Cross-validated scores (Matthews correlation coefficients, MCC) with standard deviations and final test scores for best candidates of classifiers based on grid search.

Chapter 5

Reconstruction and visualization

This chapter describes solution of the reconstruction problem. The goal is to create model able to estimate number of each of particle trace types in binned frames and to use these results to make visualizations of detected particle fluxes over the Earth. Part of this chapter is also concerned with the design of a generator of simulated frames, which were used for pretraining of one of the tested estimators.

Similar counting problem is faced in computer vision as the count estimation. It appears in areas like microbiology, industry and also surveillance (e.g., the crowd counting problem, in which the goal is to count people in an image). According to the review of crowd counting methods [42], common strategies to deal with this problem can be divided into three groups:

- counting by clustering,
- counting by detection,
- counting by regression.

Counting by clustering is based on a temporal sequence of images, where it is assumed that each moving object can be represented by local features, which all follows similar trajectory, so the number of same objects equals to the number of trajectory clusters. *Counting by detection* is applicable when we have a classifier which directly recognizes objects of interest. In this case, the counting is a trivial operation applied after detection of objects. *Counting by regression* is based on training of a mapping from image (feature) space to counts.

In our case, the problem is more complex, since we cannot apply detection on binned images because it is not possible to identify and localize particular types of particles on these frames. Unless it is possible to reconstruct the original images from binned frames, we must rely on training a black-box regression model. The training data for this model are generated by the counting by detection approach from relatively few full resolution available images. To increase the size of the dataset, artificial data generator was also implemented.

5.1 Overview of proposed counting estimator via regression

According to the previous discussion, the most promising solution to our problem is to train a regression model which receives binned image and its energetic histogram as its input and outputs estimated counts of all particle trace types. From the mathematical point of view, we assume that there is a function $f(\mathbf{x})$, which receives binned image described by some features $\mathbf{x}$ and predicts number of particles of different classes. The goal is to find (or approximate) this function with the use of a training set.

The proposed pipeline is depicted in figure 5.1. The full resolution images are fed into the classifier (described in the previous section), particle traces are detected, classified and counted

in each image. This counting gives us a set of particle counts $\{C_i\}$. Each full resolution image is also processed through a pipeline which is the same as on the satellite, so we get set of energetic histograms $\{I_{hi}\}$ and binned frames $\{I_{bi}\}$. Then each pair from $\{I_{bi}, I_{hi}\}$ can be associated with counts of particles for each full resolution frame I_i from $\{C_i\}$.

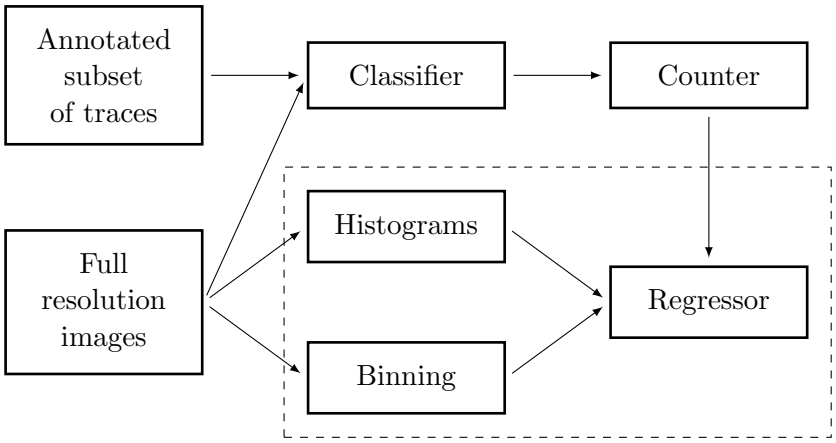


Figure 5.1: Illustration of the regression pipeline. The dashed rectangle marks core part, which is used after training for prediction.

Two types of regressors were tested: neural network and random forest regressor. Moreover, two-phase training of the neural network with artificially generated data was conducted due to the small amount of original training data. In the first phase, a large dataset was generated artificially and used to pre-train the neural network. The model parameters were then fixed and used to initiate the training in the second step. The pre-training dataset was generated using rotations and translations of individual segments on the original frames.

5.2 Dataset analysis

The training dataset contains three sets of data: full resolution images, binned images from the satellite and also artificially generated data, used for pretraining of the regressor based on a neural network. All of these data are described. Part of the artificial data description is also description of the designed simulator which was used to generate them.

5.2.1 Full resolution original training dataset

The training dataset for the regressor contains 493 full resolution images. Each frame contains particles classified into 8 classes. However, some of these classes are very rare, which leads to the following consequences:

1. the classifier (from the previous stage of processing pipeline) can not always classify them correctly, which causes noise in the training dataset,
2. the dataset is small and high-dimensional (relative to its size), so the regression model needs as clean and simple data as possible. Regression for these rare noisy events may cause that the error from the (counting) classifier will by further amplified by error from regression model with very unreliable results in the end of the pipeline.

In a favor of prediction reliability maximization, labels of classified particles from the training dataset are remapped according to the table 5.1. Aim of this remapping is to generalize the meaning of particles by merging particles which are similar into larger classes. All the blob-type

particles are generalized as a "blob" class. Class of drop-shaped particles is also included under simplified class of blobs, since it is likely caused by similar type of heavier particle. Class of particles of type "Other" is discarded. All the track-like classes are kept the same since the datasets contains sufficient number of representative examples.

Class of classifier	**Simplified class**
Dot	Dot
Blob small	Blob
Blob big	Blob
Blob branched	Blob
Track straight	Track straight
Track curly	Track curly
Drop	Blob
Other	-
Track lowres	Track lowres

Table 5.1: Mapping between classes used by the classifier and the regression model.

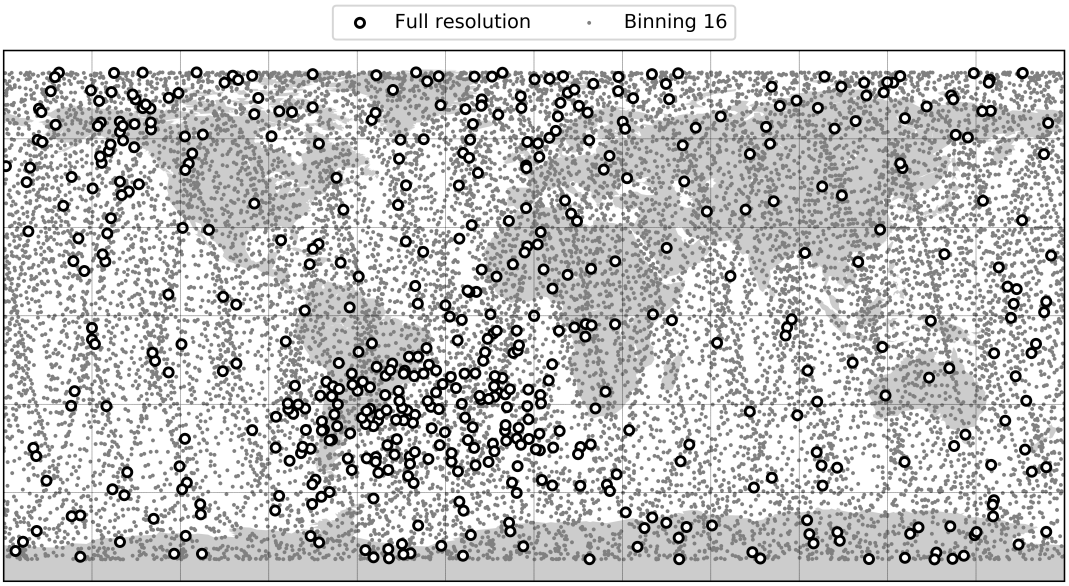


Figure 5.2: Locations of captured frames from April 30, 2017 till May 15, 2018.

5.2.2 Artificially generated dataset

Because the number of full resolution frames is too small for training of a neural network and the task of regression is rather complex, two-phase training has been used to reduce the error of the model. The model parameters are initially set from large number of the simulated images to the values that are used as initial values for the subsequent training on original frames.

Segments classified in the previous part of the thesis are used to generate simulated images for the pretraining stage. These segments are randomly chosen and placed into the new frames using only translation and rotation. Rotation is performed at an angle that is an integer multiple of 90 degrees, otherwise it would be necessary to interpolate the rotated segment into the rectangular grid of the new image, which could severely distort the appearance of the trace of the particle due to the low resolution.

The appearance of the generated images is highly dependent on the statistical distribution of particle counts. It is hypothetically possible that the best results (the regressor with maximum generalization ability) would be obtained when using uniform distribution, which in fact forces the regressor to be able to predict particle counts for many distributions which are subset of the uniform one.

However, this would imply big performance penalty since many training samples would be needed, so the distributions of particle counts were modeled on the basis of the original training set to minimize this issue. The core assumption behind fitting of the model is that probability of observing image with vector of particle counts $\mathbf{c} = [c_1, c_2, c_3, c_4, c_5]^T$ is treated as some (maybe nontrivial) joint distribution $P(\mathbf{c})$, which can be estimated. The estimation is performed by the kernel density estimator, which is a nonparametric technique that estimates probability density $p(\mathbf{c})$ expressed as [43]:

$$\hat{p}_n(\mathbf{c}) = \frac{1}{nh^d} \sum_{i=1}^{n} K\left(\frac{\mathbf{c} - \mathbf{c_i}}{h}\right), \tag{5.1}$$

where $\mathbf{c}$ is vector of particle counts for some arbitrarily chosen frame and $\mathbf{c_i}$ is vector of particles on image i in the training dataset. It holds for both $\mathbf{c}$ and $\mathbf{c_i}$ that the vector elements are integers numbers and dimension of both vectors is d. There are n samples in the training dataset, h is a parameter which specifies amount of smoothing of the estimator. K is a kernel function, $K : \mathbb{R}^d \mapsto \mathbb{R}$. The Gaussian kernel, defined as

$$K(x) = \frac{\frac{\exp(-||x||^2)}{2}}{\int \exp(\frac{-||x||^2}{2})\mathrm{d}x}, \tag{5.2}$$

was chosen in our case. Examples of resulting images are in figure 5.3.

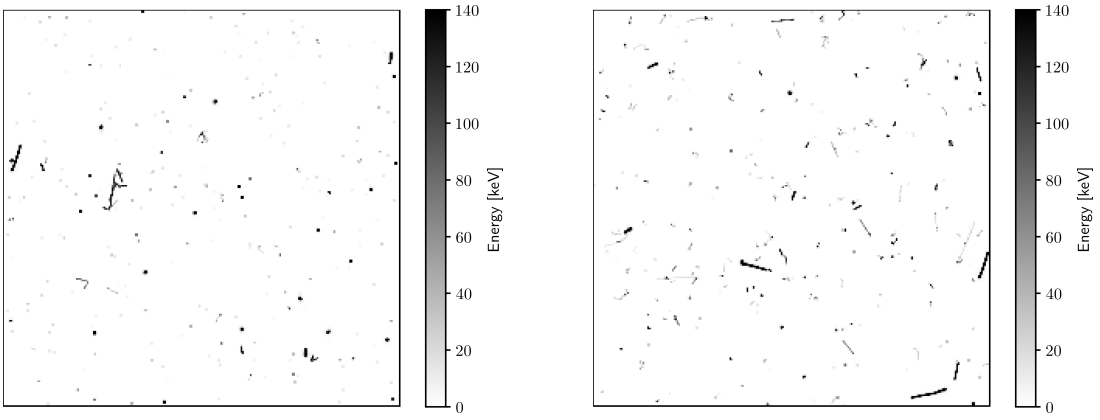


Figure 5.3: Demonstration of output of the data generator.

5.2.3 Binned dataset for application of regression

The trained and verified regression model is applied on a set of binned images, which were downloaded from the satellite. Important decision, which was taken before design of the regression model, was to choose binning mode for which the regression model will be optimized. It is necessary to consider available numbers of frames collected in each mode during the service of the satellite.

Frame type	Frames in dataset
Binning 8	424
Binning 16	16962
Binning 32	1610
Histograms	18344
Sums	76

Table 5.2: Number of available compressed frames in dataset for different compression modes (at time of regressor application when these numbers were relevant).

It can be seen from the table 5.2 that the frames with binning 16 compression mode were captured most frequently (and were captured uniformly in time during lifetime of a satellite, approximately once per week). Almost every frame captured with the binning 16 mode is also accompanied by the energy histogram, which can be used to calculate features for regression model. From that it follows that the pipeline should be optimized for the binning 16 mode.

5.3 Regressor design

The elementary scheme of the regressor is similar to the classifier architecture. The first step is description of the binned images with condensed representation with a lower dimension – the feature extraction. The features are calculated mainly on the basis of pixel values of binned images, together with the energetic histogram, which is available almost for each image. Binned images described by the features are, after scaling to a zero mean and unitary variance, put into the estimator.

Two estimation methods were chosen to be tested: a neural network and a random forest regression. The neural network was trained in two steps. The aim of the first step was to find initial values of the model parameters based on a large number (40000) of simulated images. The second step involved training on the real images with initial parameter values taken from the previous step.

5.3.1 Feature extraction

Each image with binning 16 is represented by a 256-dimensional vector in a space of all these images. Since our real training dataset contains only a few hundreds of images, regression model trained just from image pixel intensities would be susceptible to overfitting. Feature representation with 27 features was proposed to overcome this issue. These features, described below, are mostly of a statistical nature since binned images often have a texture-like appearance with no clearly identifiable shapes.

Texture features

Co-occurrence gray-level matrix is a tool used for texture description [44]. This matrix is constructed from all the pairs of pixels (p_i, p_j) which are separated by defined translation d with angle θ. For image with N intensity levels it is a matrix of a size $n \times n$:

$$P(n_i, n_j|d, \theta) = \begin{bmatrix} P(0,0|d,\theta) & \dots & P(0,N|d,\theta) \\ \vdots & \ddots & \vdots \\ P(0,N|d,\theta) & \dots & P(N,N|d,\theta) \end{bmatrix}, \tag{5.3}$$

whose entry with coordinates n_i, n_j can be interpreted as a probability that values n_i and n_j occurs in any two pixels separated by translation d with angle θ. Several features, mostly with quantitative meaning, are calculated from this matrix, like contrast:

$$C = \sum_{i,j=0}^{N-1} P_{i,j}(i-j)^2, \tag{5.4}$$

dissimilarity:

$$D = \sum_{i,j=0}^{N-1} P_{i,j}|i-j|, \tag{5.5}$$

homogeneity:

$$H_o = \sum_{i,j=0}^{N-1} \frac{P_{i,j}}{1+(i-j)^2}, \tag{5.6}$$

angular second moment:

$$ASM = \sum_{i,j=0}^{N-1} P_{i,j}^2, \tag{5.7}$$

energy:

$$E = \sqrt{ASM}, \tag{5.8}$$

and correlation between gray levels:

$$\rho = \sum_{i,j=0}^{N-1} P_{i,j}[\frac{(i-\mu_i)(j-\mu_i)}{\sqrt{(\sigma_i^2)(\sigma_j^2)}}]. \tag{5.9}$$

In a feature extraction pipeline, this matrix is calculated for $d = 1$ pixel and angles 0°, 45° and 90°. All the resulting matrices are then averaged and this averaged matrix is then used as a source for the feature calculations above.

Pixel features

If a density of tracks on an image is sufficiently small so it is possible to assume that each bin is crossed by at maximum one particle, we can assume that the values (basically occupancies) of bins might be connected with the incoming particle types. To account for this, numbers of bins with values in intervals $\langle 0;3\rangle$, $\langle 4;7\rangle$, $\langle 8;15\rangle$, $\langle 16;47\rangle$ and $\langle 47;256\rangle$ were taken as features.

Energetic histogram

The energetic histograms of images contain 16 bins, maximum energy captured in the histogram is 140 keV. All the pixel energies higher than this are put to the last bin. All the bins of a histogram were taken and stacked together with features described in the above section.

5.3.2 Estimator

Two estimators were tested: neural network and regression random forest. The neural network was trained using an artificial data generator, for random forest regressor it is sufficient to use only original satellite data to get acceptable results. Both of these estimators with their performance analysis are described below.

Feedforward neural network

Feedforward neural network is a learning scheme that is composed of base units, neurons, that are arranged in layers. The neurons constitute oriented acyclic computational graph. Multilayer neural networks, such as those used in this thesis, consist of at least three layers: input, hidden, and output layers. More hidden layers can be present.

Artificial neurons remotely resemble biological neurons. Each such neuron can have multiple inputs (each with its own weight – a tunable parameter), body (where calculations are performed) and one output. The body of the neuron performs weighted sum over the inputs and passes the result to the activation function f. Output of this function is written to the output of a neuron and passed to all neurons in the following layer.

In mathematical terms, each neuron computes function

$$y = f\left(\sum_{i=1}^{n} w_i x_i + w_0\right), \tag{5.10}$$

where y is its output value and $\mathbf{x} = [x_1, x_2, \ldots, x_n]^T$ are inputs. Each input has one weight w_i associated with itself. There is also one weight – bias w_0 – which has no associated input. Function f can be of many different forms. Commonly used ones are, e.g., sigmoid, hyperbolic tangent, and variations of linear function.

The neurons are connected into layers in a way such that output of each neuron in each layer (except the output layer) is connected with input of each neuron in the following layer. This complex structure can be trained to approximate continuous function of inputs with the use of optimization technique which tunes the weights to appropriate values. Example of such algorithm is the backpropagation algorithm, which minimizes the error between real and predicted output values for the given training inputs. The numerical optimizer which was used to tune the weights in this thesis was the Adam algorithm.

The mean square error was used as a metric of the network performance. The network was instantiated for prediction of each class independently, so in total 5 neural networks were trained. Architecture of each of these networks is identical. All hidden layers of all these networks uses rectified linear function as their activation:

$$f(x) = \max(0, x). \tag{5.11}$$

Output layer uses linear activation function. A dropout[1] layer (with dropout rate 0.2) was used on the input of the neural networks. The network structure can be seen in figure 5.4.

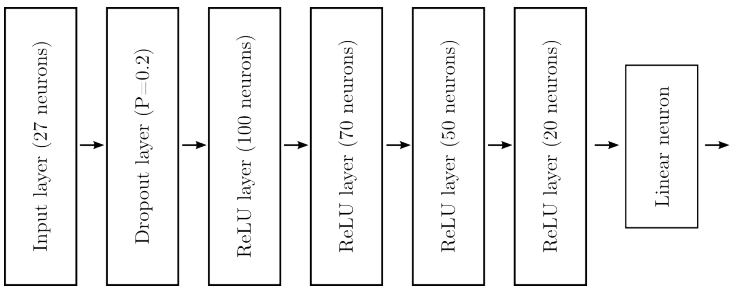


Figure 5.4: Developed structure of regression neural network.

The estimator was trained in two phases. The first phase was pretraining on an artificial data with the goal of setting of weights of networks to values which are near to the optimal

[1]Dropout is a technique used during training of neural networks. It randomly deactivates certain fraction of neurons (dropout rate) each training step. It can be used to prevent overfitting.

ones for a real data. 50000 artificial samples were generated during this phase. These samples were divided into training (40000 samples) and validation (10000 samples) datasets. Because of a length of the training and large amount of available data no crossvalidation was performed. After each epoch the network was scored both on training and validation dataset and results were plotted in figure 5.5.

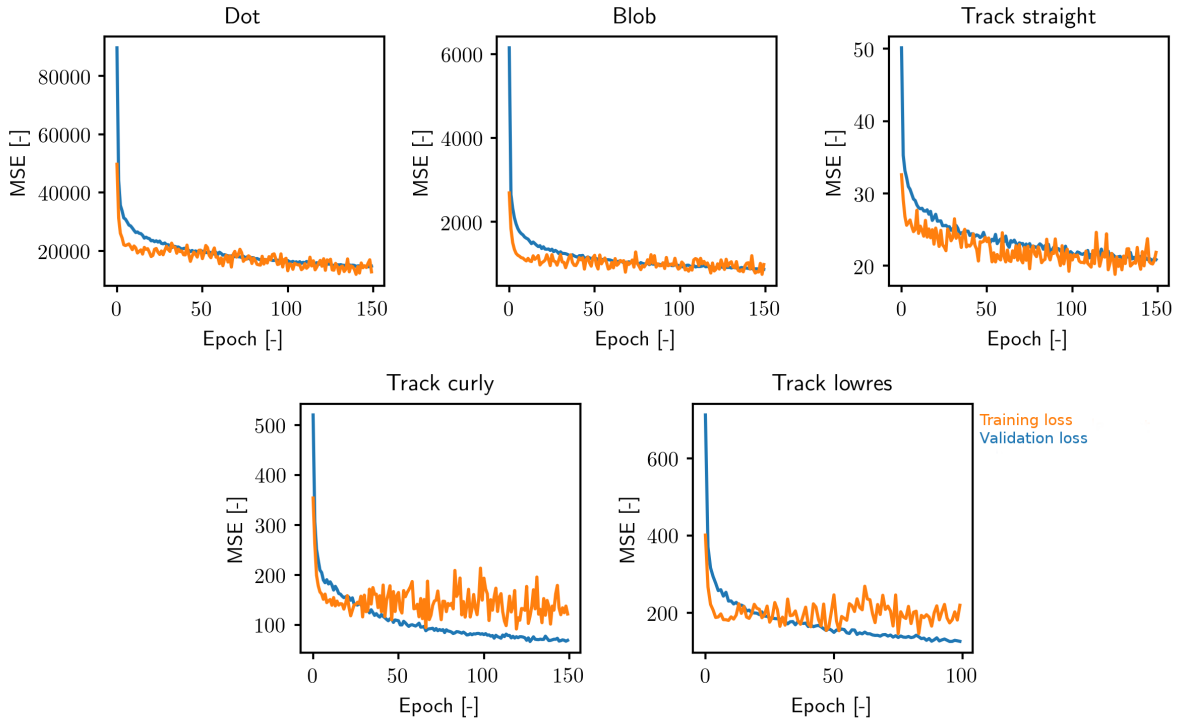


Figure 5.5: Validation curves for the neural networks pretraining: dependence of mean squared error (MSE) of regression on the number of training epochs. The blue curve is the error on the training dataset, orange one is the error on the validation dataset.

Stopping epochs were chosen in order to minimize training time and regression error according to the table 5.3.

Class	Stopping epoch	Training error (MSE)	Validation error (MSE)
Dot	150	14724	12850
Blob	150	870	986
Track straight	125	21.14	18.75
Track curly	18	153.8	131.1
Track lowres	9	228.29	179.02

Table 5.3: Results of neural networks pretraining.

During the second training phase the networks were initialized with weights from the pretraining phase. The training was performed with the goal of minimizing mean squared error (MSE) of the network on the validation dataset during 10-fold crossvalidation. The stopping epoch was selected with the use of the validation curves (in figure 5.6). The final testing was performed on a testing set (with a size of 25 % real full resolution images). The results are presented in table 5.4. Qualitative evaluation of the neural network regression can be performed with the use of figure 5.7.

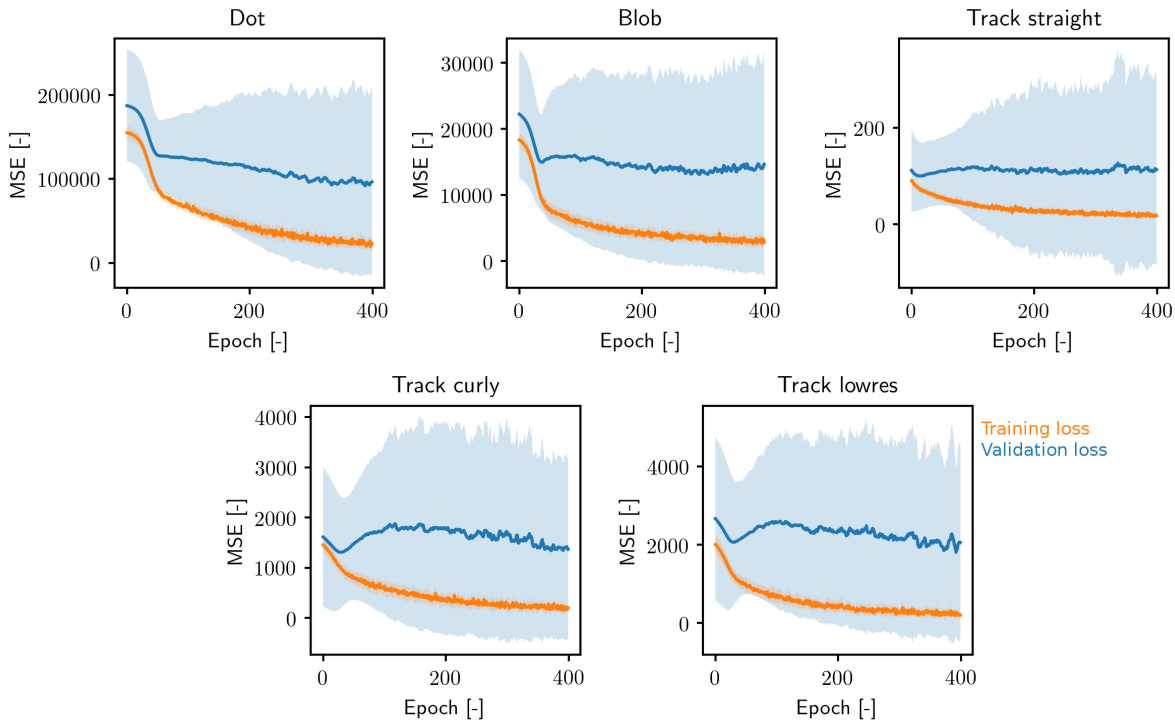


Figure 5.6: Validation curves for training of the neural networks on the real data. Curves were created with the use of a 10-fold crossvalidation on the training subset of the regression dataset. Blue curve is validation loss, orange training loss. Colored area around losses is a standard deviation of the loss during crossvalidation.

Class	Stopping epoch	Test error (MSE)
Dot	50	98633.68
Blob	37	3291.32
Track straight	12	57.61
Track curly	29	220.82
Track lowres	30	452.84

Table 5.4: Test errors of the neural networks after final training on the real data.

Random forests

The random forest regression is similar to the random forest classification as described in previous chapter. Again, the forest consists of ensemble of decision trees combined with the use of bagging. However, because the task of regression is to model some continuous function (as opposed to classification), the trees are of a different nature [45]. The first difference is different node impurity measure considered when choosing a suitable split in a feature space during tree generation. Second difference is that each leaf outputs not a categorical, but a numerical value. The type of this output can be different for different tree algorithms, e.g., in CART algorithm this value is a constant. Estimate of the function modeled by a such decision tree regressor is then a "stairway-like", piecewise constant function.

The bagging principle must be also altered to reflect these changes. It is no longer possible to vote for a final regression result among the ensemble of trees, since each tree can output one of infinitely many values. Average from all ensemble results is calculated instead of voting.

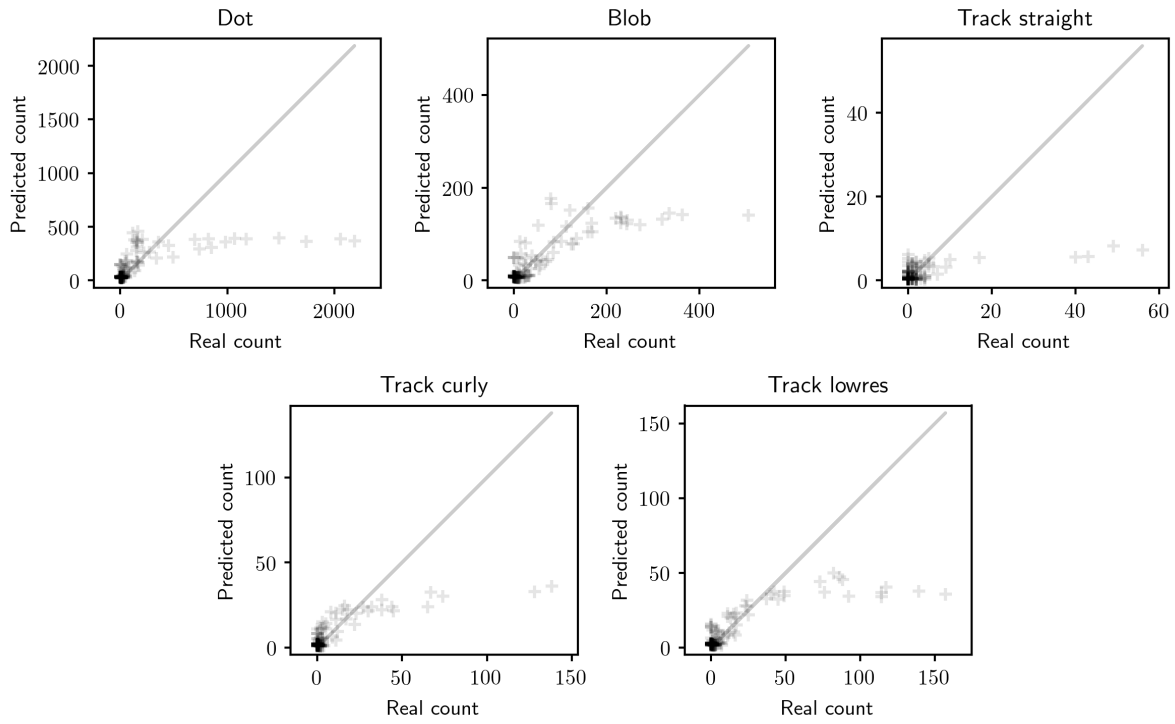


Figure 5.7: Plot of real versus predicted counts for neural network. The closer the points to the identity line the better the regression. The plots were generated with the use of a testing subset of a regression dataset.

One instance of random forest regressor (implemented in the scikit-learn library) was trained for each class of particles. During empiric trials it was identified that good hyperparameter values for this problem are the default ones, except number of estimators in each forest. These identified values are shown in table 5.5.

Hyperparameter	**Value**
Number of estimators	400
Maximum tree depth	∞
Mininum samples per split	2
Minimum samples per leaf	1

Table 5.5: Table of parameters of random forests used as regression estimator

Test of models based on random forest with parameters from table 5.5 resulted in regression errors presented in table 5.6. The errors were obtained during a final testing on a test subset, which contained 25% of all the real full resolution images. Qualitative analysis can be performed with the use of figures 5.8.

Class	**Test error (MSE)**
Dot	15730.18
Blob	508.95
Track straight	16.32
Track curly	37.82
Track lowres	62.57

Table 5.6: Test errors (mean square errors, MSE) for the random forest regression model.

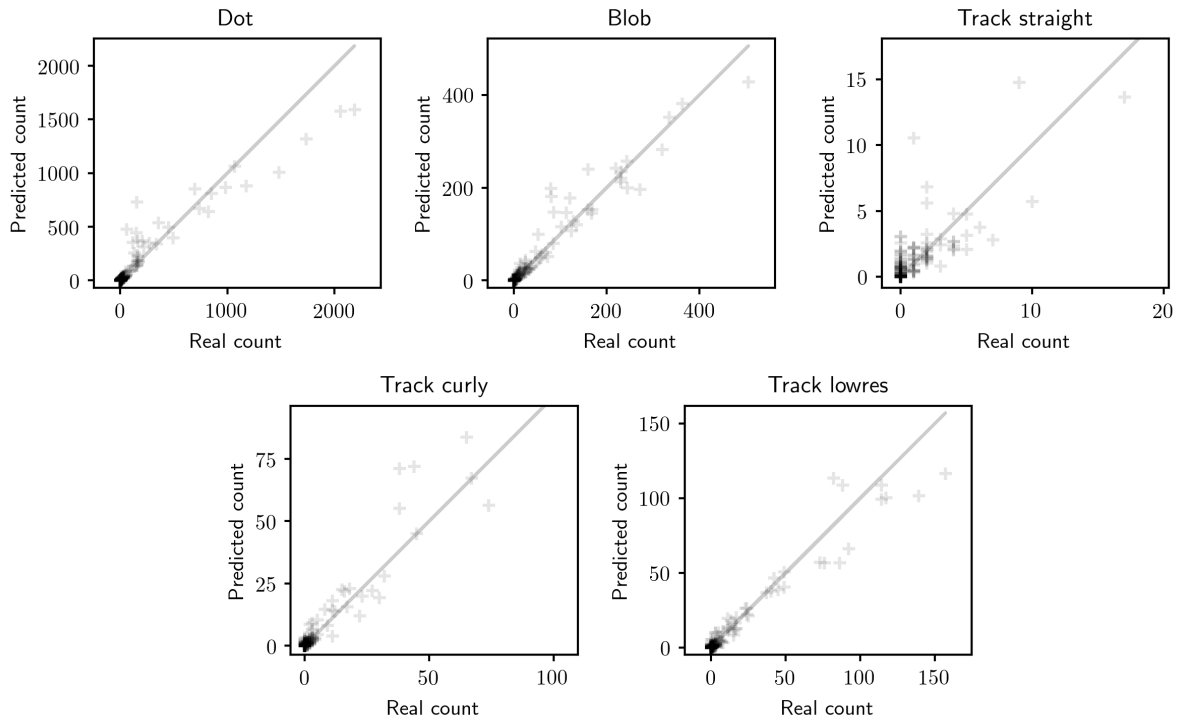


Figure 5.8: Real versus predicted counts for random forest. The closer the points to the identity line the better the regression. Generated with the use of the testing subset of the regression dataset.

5.4 Visualization of results

This section explains the last step of the processing pipeline. The input to this stage is a collection of data points sampled approximately 500 km above the surface of the Earth. Each point consists of a timestamp with estimated counts of particles without information about latitude and longitude. These coordinates must be inferred from the timestamp with the use of tracking data provided by NORAD (North American Defense Command). With known latitude and longitude it is possible to plot measurements into the map. Since data are sampled relatively sparsely and nonuniformly both in time and space, measurement results are interpolated with a suitable algorithm and plotted onto the surface of the Earth, which is then projected to 2D map.

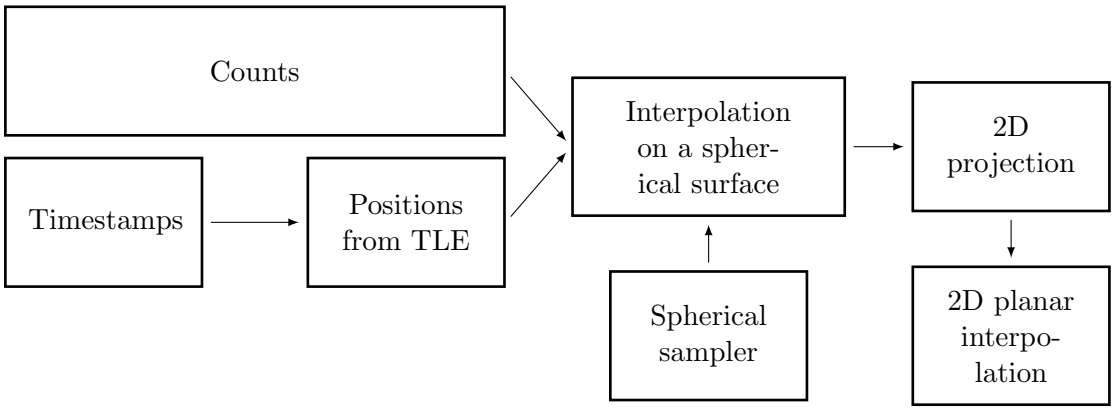


Figure 5.9: Diagram of visualization pipeline.

Operations leading to visualizations are shown in the diagram 5.9. The input to this pipeline is a set of counts of particles in images together with the set of times when these images were exposed. The position of the satellite must be estimated from the timestamp before spatial interpolation. After that, the data passes through two interpolation steps. The first one takes spherical geometry of the satellite's orbit into account, the second one is applied on a 2D projection of the Earth to transfer datapoints to a uniform rectangular grid (bitmap), which can be directly displayed.

It would be also possible to use only one-step interpolation, where the latitudes and longitudes are chosen such that they directly corresponds to points on a bitmap which will be displayed. Nevertheless, there are two main drawbacks of such approach:

1. Desired resolution of the final bitmap can be very high. Interpolation of such number of points with some (spherical interpolation) algorithms have shown not to be feasible.
2. Because of the projection of a curved surface, the projected interpolated points with equal spacing will represent areas of variable density on a resulting bitmap.

Both of these problems are solved by a proposed two-step interpolation.

5.4.1 Position of the satellite

The satellite orbits around the Earth on orbit with perigee 499.2 km and apogee 514.9 km (in March 29, year 2018), but its altitude slowly decreases. No spatial coordinates of the satellite are known, only timestamp of exposure of each incoming frame. Position can be recovered from the known time with the use of a parametrized model of the satellite's orbit. Specifically, parameters for our purpose are available in a form of TLE (two-line elements) and are intended for use with the so-called "simplified perturbation models" [46], which is a family of mathematical orbit models useful for prediction of space object positions in various distances from the Earth with accounting for orbital changes caused by atmospheric drag and influence of other space bodies.

TLE specification of spacecraft's orbit consists of three lines. The first line is name of the orbiter, the second and the third line describes particular orbit. Second and third line also have line number as first element and their last element. Each line ends with checksum. Other elements are identificators of the object, time specifier and parameters for the mathematical model (derivatives, drag coefficient, perigee etc.). Example for the VZLUSAT-1 satellite can be seen below:

```
VZLUSAT-1
1 42790U 17036AB  17178.94283943 +.00001416 +00000-0 +67467-4 0  9991
2 42790 097.4493 238.6523 0010556 232.2753 127.7526 15.20723572000655
```

Figure 5.10: Example of TLE for VZLUSAT-1 satellite.

The mathematical model is available in specialized software libraries. The routines for recovery of the VZLUSAT-1 latitude and longitude were supplied as ready-to-use functions with the software which parses incoming data from the satellite.

5.4.2 Spherical data interpolation

The first interpolation through which the data passes is the interpolation on surface of a 3D shape which contains all orbits of the satellite. The satellite orbits along orbit with perigee in 499.2 km and apogee 514.9 km (in March 29), so this surface can be approximated by a sphere. Positions of all datapoints, estimated from the timestamps on this spherical surface are specified,

in a latitude-longitude format, where latitude is expressed in degrees in range $\langle -90°, 90° \rangle$ and longitude in range $\langle -180°, 180° \rangle$. Interpolation in this coordinate system is complicated by the curvature of the surface and also by singularities (longitude is not defined for polar areas). Both of these issues can be mitigated by the use of interpolation algorithm which allows us to choose distance metric of our choice.

Distance metric on a curved surface

The most intuitive choice of distance measure in our case is the orthodromic distance, which is defined for given two points as a length of the shortest curve connecting these points. This distance corresponds to the common distance as measured in geography. More possibilities exist for its calculation. It is often important (particularly when working with surface of the Earth) to consider that shape of the Earth is not a sphere but rather ellipsoid. This distance can be calculated with the use of a Vincenty formulas [47], which consists of a two methods, the first one solves problem of finding distance between two points on a surface of an ellipsoid and the second one solves problem of finding second point from known latitude and longitude of the first point together with azimuth and distance. There is also an analytical solution based on haversines[2] to calculate the orthodromic distance on a spherical surface, which handles issues with polar singularities.

Since it is unnecessary to work with the ellipsoidal model of the Earth in our case, the haversine formula was used as a distance measure for interpolation.

Spherical interpolation algorithms

This thesis deals with scattered data interpolation[3] problem, where each point of N data points (x_i, y_i) consists of coordinates y_i from some domain Ω and from a scalar value y_i, measured on these coordinates. Then we can define the scattered data interpolation as follows [48]:

Definition 5.4.1. *Given $(x_i, y_i) \in \Omega \times \mathbb{R}, i = 1, 2, \ldots, N$, $\Omega \subset \mathbb{R}^k$, find $s \in S(\Omega)$ such that $s(x_i) = y_i$, $i = 1, 2, \ldots, N$. $S(\Omega)$ is the interpolating space. It is also assumed that the measurements y_i were generated by some unknown function $f(x_i)$.*

The domain Ω is a spherical surface and every x_i is 2-dimensional in our case, because each point is spatially determined by its latitude and longitude. We have multiple values of y_i (counts of particles of different types) to interpolate in each x_i. It is possible to solve this by performing multiple interpolations, one for each particle type.

Many algorithms exist for solution of this problem. Popular methods are often based on an assumption that value $z^\star(s_0)$ in some unknown point s_0 is based on a linear combination of known values $z(s_i)$:

$$z^\star(s_0) = \sum_{i \in N(s_0)} a_i z(s_i), \tag{5.12}$$

where N is a set of known values in a neighborhood of point s_0. N can be set of all available points, but it can be also only a subset, e.g., when the interpolated surface is not stationary and points far away from s_0 carry little or no information about this point. Coefficient a_i is calculated for each interpolated point in a way which is characteristic for each interpolation

[2] Haversine $\mathrm{hav}(\theta)$ is a function $\mathrm{hav}(\theta) = \sin(\frac{\theta}{2})^2$

[3] It must be noted that in geostatistical and practically oriented literature the term interpolation is sometimes used to denote also function approximation, which does not forces the approximating function to pass exactly through the points with known values. It also holds that some interpolation algorithms can be used both for interpolation and approximation. For these reasons this thesis do not make a difference between these two in practical applications.

method. Methods of primary interest in thesis are inverse distance weighting (IDW), nearest neighbor and kriging:

IDW Values of a_i are determined with the use of inverse distance weighting, which calculates interpolated value $z^\star(s_0)$ as a weighted average of p-th powers of distances of a known points from unknown point:

$$z^\star(s_0) = \frac{\sum_{i \in N(s_0)} z_i(s_i) d_i^{-p}}{\sum_{i \in N(s)} d_i^{-p}}. \tag{5.13}$$

This method is, according to [49], often used with $p = 2$.

Nearest neighbor The efficient and simple method is to partition the interpolated surface into Voronoi cells V such that cell V_i is generated by sample with position s_i. Then the weights, according to the equation 5.12, are chosen in a following way:

$$a_i = \begin{cases} 1 & \text{if } s_0 \in V_i \\ 0 & \text{otherwise} \end{cases}. \tag{5.14}$$

The target value $z^\star(s_0)$ can be also calculated with the use of multiple nearest neighbors – multiple nearest neighbors are found and averaged. This method is not an interpolation from mathematical point of view since it smooth the interpolated surface. The mean can also be weighted by inverse distance of nearest neighbors, which is a variant of IDW.

Kriging Kriging is a whole family of more sophisticated methods used in geostatistics [50]. Kriging assumes that function z is a random field. Basic kriging interpolation equation is similar to 5.12:

$$z^\star(s_0) - m(s_0) = \sum_{n \in N(s_0)} a_i (z(s_i) - m(s_i)), \tag{5.15}$$

where weights a_i are chosen such that they minimize variance:

$$\sigma^2(s_0) = \mathrm{var}(z^\star(s_0) - z(s)) \tag{5.16}$$

with

$$\mathbf{E}\{Z^\star(s) - Z(s)\} = 0. \tag{5.17}$$

It is also assumed that $z(s)$ consists of two components: residual with zero mean and stationary covariance, which is a function of distance, and trend. Kriging variants differ in assumptions about these components.

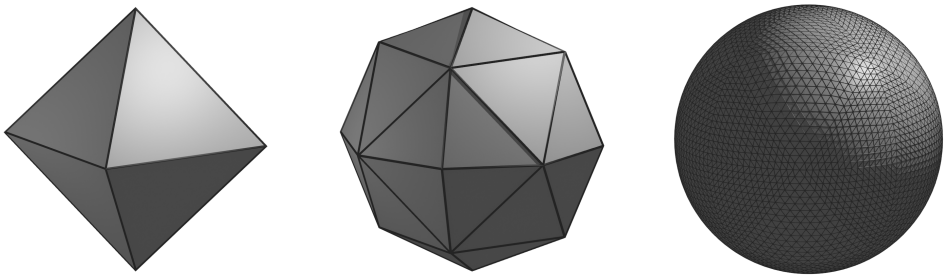
Sampling of interpolation points

The (fitted) interpolator can be imagined as some function f, which returns predicted value of particle counts in some latitude and longitude for each interpolated point. The last thing remaining to solve before application of algorithms described above is how the coordinates, where the interpolation will be performed, should be generated.

Output of this interpolation step is a set of interpolated points on a surface of a sphere, same as the one on which original measurements are located. The interpolated points can be placed anywhere, but they must cover the sphere at least perceptually uniformly without clusters or holes.

Stochastic and deterministic solutions to this problem were considered. Even though the stochastic solution based on uniform sampling of points on the surface of the sphere is computa-

tionally less expensive, large amount of generated points is needed to be close to uniformly cover the interpolated surface. Because of this constraint the deterministic solution was proposed and implemented, which recursively subdivides the faces of a geodesic octahedron and normalizes their vertices in a way such that they lie on a spherical surface so the octahedron is inscribed in a sphere (see algorithm 2). It is possible to set a level of subdivision, where the algorithm stops. Sample of output for three different subdivision levels can be seen in the figure 5.11.



(a) Subdivision level 0 (b) Subdivision level 1 (c) Subdivision level 5

Figure 5.11: Demonstration of output for different levels of subdivision.

Algorithm 2 Geodesic grid generator

```
faces ←geodesicOctahedron()
subdivided ← list()
while subdivisionThreshold not reached do
    for face in faces do
        temp ←subdivideFace(face)
        for face in temp do
            for vertex in face.vertices do
                normalize(vertex)
            end for
        end for
        subdivided.append(temp)
    end for
end while
return subdivided
```

The resulting grid is a simple variant of a discrete global grid (DGG), where the surface of the Earth is discretized into cells [51]. Each vertex on this grid represents one point for which the spherical interpolation will be performed. Since vertices are expressed in cartesian coordinates x, y, z (as a points on a unit sphere with center in the origin of the coordinate system), it is necessary to perform conversion to latitude-longitude coordinate system:

$$\begin{aligned} \phi &= \arcsin(\mathrm{z}) \\ \lambda &= \arctan 2(x, y), \end{aligned} \tag{5.18}$$

where ϕ is latitude and λ is longitude.

5.4.3 2D projection

All the data points processed according to the previous sections were represented as points on a two dimensional surface curved in a three dimensional space. For final visualization purposes it

is necessary to project data onto a flat surface. Solution of this task is to choose the right map projection, which converts latitude-longitude coordinates of data points to a (cartesian) position on a two dimensional map.

We are interested in global maps depicting counts of detected particles above different locations of the Earth. It implies need for a map projection which is easy to understand in a global scale and which is without significant distortion. There is also a need for complete projection[4], since the measurements were taken literally everywhere (above the equator, above the poles, etc.). The chosen projection should also preserve area in different places so that areas with different measured particle fluxes can be compared.

The selected projection which fulfills these requirements is the Mollweide projection. It is equal-area pseudocylindrical projection which preserves parallel lines. This projection is suitable for plotting of global phenomena and was used, e.g., for visualization of results of the microwave anisotropy probe WMAP [52]. The projection is shown in figure 5.12 together with the Tissot indicatrix to visualize distortion.

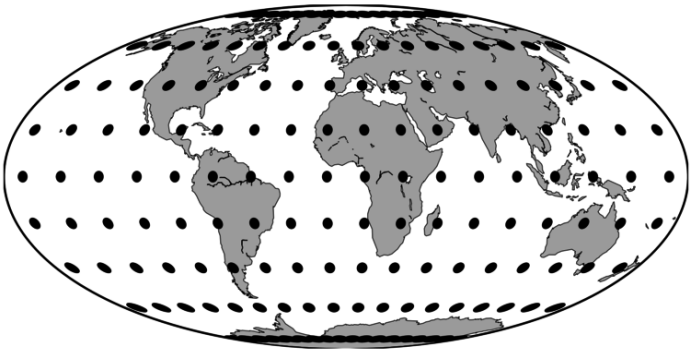


Figure 5.12: Tissot indicatrix of Mollweide projection: visualization of distortion. All the black blobs were circles with equal area before the projection.

5.4.4 2D interpolation

Points projected from spherical coordinates to a flat map using the Mollweide projection, as described in the previous section, are unevenly and still relatively sparsely spaced. The last step of processing before the results are drawn into bitmap file is interpolation of points to a regular rectangular grid. This smooths the map (the exact smoothing effect depends on the resolution of the geodesic grid generated by the algorithm described above) and allows for a computationally cheap increase of resolution, this time ignoring the spherical geometry.

The algorithm involved in this step is barycentric linear interpolation, which performs triangulation of the scattered data and then interpolates values in each triangle with the use of values of its vertices. Let the point p be located inside triangle with vertices p_1, p_2 and p_3. Then we can create three new, smaller triangles T_1, T_2 and T_3 with vertices (p, p_1, p_2), (p, p_1, p_3) and (p, p_2, p_3). Each of the triangles T_i has an area A_i. Then the value in point p can be written as:

$$p_{value} = \frac{A_1 p_{1value} + A_2 p_{2value} + A_3 p_{3value}}{A}, \tag{5.19}$$

where p_{ivalue} is value in point p_i and A is area of the large triangle created during triangulation of a scattered data. The efficient implementation of this algorithm was taken from the SciPy library.

[4]Projection showing the whole Earth.

Chapter 6

Results and discussion

This chapter is focused on presentation of the final results of the whole pipeline described in chapters above – visualizations of estimated distribution of five particle trace types on the low Earth orbit. In the beginning, the processing of measurements is quickly summarized from the practical point of view. The next parts present obtained results in graphical form and the last part of this chapter is focused on discussion of practical problems encountered during work on this thesis and ideas for possible future missions.

6.1 Processing results

The proposed pipeline (which consists of data manager, classifier, regression model and visualizer) was used to process the total of 21 whole-Earth scans from the VZLUSAT-1 satellite. In addition, approximately 500 full resolution images were analyzed. The scans mainly consist of "binning16" images along with the histograms, allowing estimation of number of detected particles using the regression model. The images of each sequence were sampled unevenly over time and space and different exposure times were used, depending on location of the satellite at the time of exposure. Scans consist of a variable number of sampled images, ranging approximately from a few hundreds to one thousand. The scans were processed in two ways. First processing run created maps of distribution of ionizing particles for each scan separately. During the second one all the scans were merged and all of the images were analyzed together. Focus was also given on full resolution images, which were processed to obtain energy spectra of different particles in different regions.

6.1.1 Processing of separate scans

The separate-scan based processing resulted in a large number of generated images that can not all be presented in this part of the thesis. A scan containing 887 images (in figure 6.2) was selected as a representative example. This scan belongs to scans with higher quality outputs – not all of them were so successful[1] Spherical interpolation algorithm applied in this case was the IDW combined with the 7-nearest neighbors algorithm, which guarantees that the interpolated surface passes exactly through the sampled points. Nevertheless, in our case this assumption was broken by the use of a second interpolation step – the planar triangulation, which further smoothed the data.

[1]Parts of several whole-Earth scans were irreversibly damaged or completely missing due to on-board memory corruption, mostly because of single event error [27].

6.1.2 Processing of merged scans

In separate dosimetry scans it is possible to identify interesting and important physical effects like the South Atlantic Anomaly and the Northern and Southern Radiation Horns, but still these scans have low resolution and estimated values in same places are often inconsistent between them. To mitigate these problems all the scans were merged into one set of images. The visualizations were then generated from this set with the use of k-nearest neighbor regression algorithm with uniform weights, which do not suffer from the artifacts of IDW caused by forcing the interpolated surface to pass through the measured datapoints. The high spatial density of measurements in merged dataset allowed to apply higher degree of smoothing with the choice of $k = 200$. The results are available in figure 6.3.

6.1.3 Energetic spectra

The full resolution frames could not be used for reasonable dosimetry maps estimation since there is only a small number of them and majority of them was sampled above the South Atlantic Anomaly. Nevertheless, over 100000 particles were identified on the full resolution frames by the classifier. These particles were used for analysis of energetic spectra over several regions above the Earth: the South Atlantic Anomaly (SAA in the figure 6.1), north and south radiation belts (NB, SB) and equatorial area (EQ).

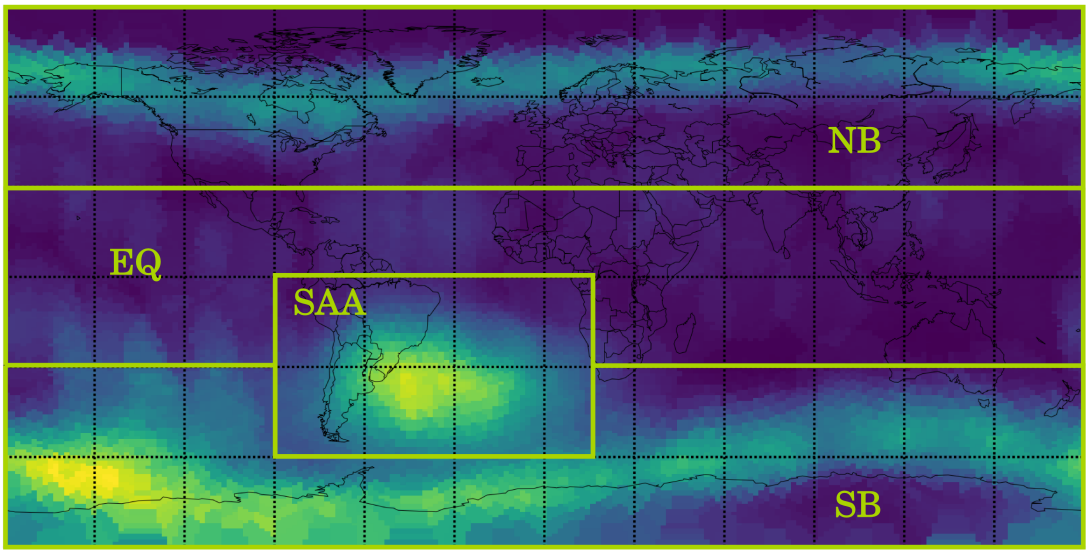
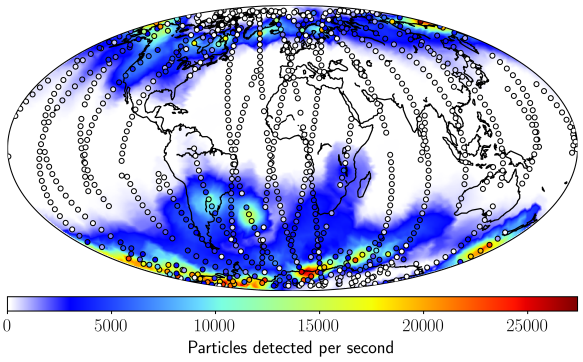
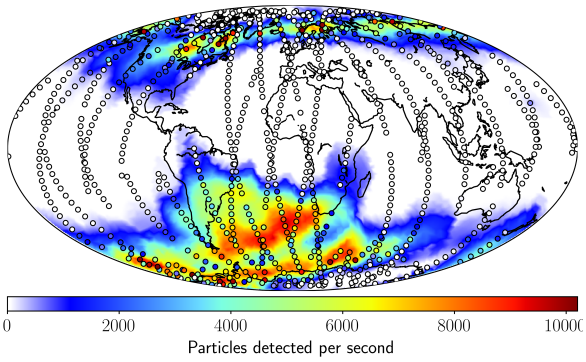


Figure 6.1: Regions (rectangles in latitude-longitude coordinate system) into which the map was divided for analysis of energetic spectra of classified particles identified on full resolution images. This particular map was generated from the binned data because of clear visibility of interesting features of different regions.

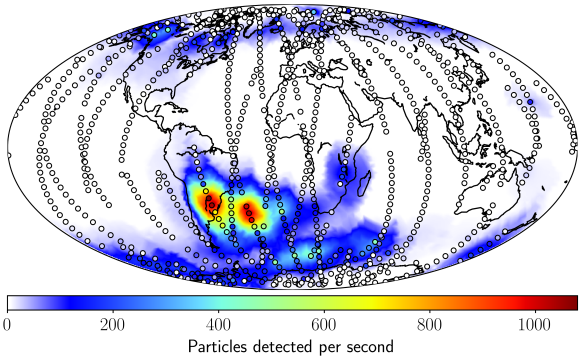
The energetic spectra were calculated for the original particle classes – the ones the classifier worked with (dot, track straight, track curly, track lowres, blob small, blob big, blob branched, drop). Spectra for the class “other” were not calculated since this class contains mainly corrupted segments without direct physical meaning. All spectra are in figure 6.4.



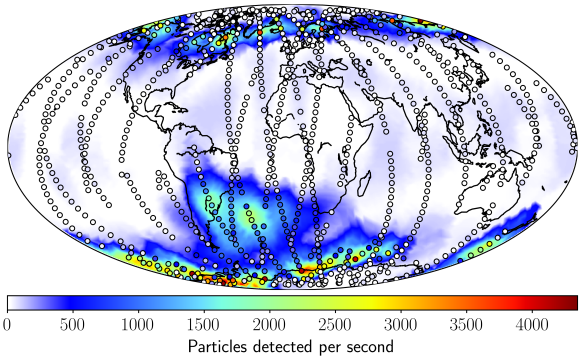
(a) Dot



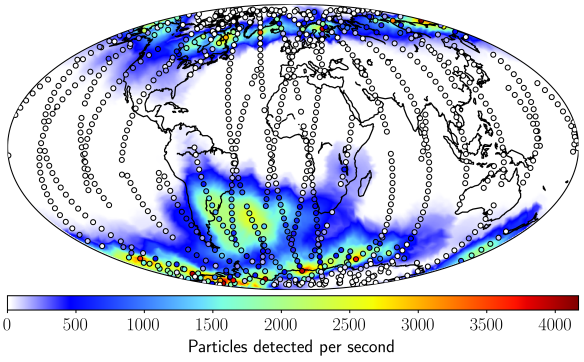
(b) Blob



(c) Track straight

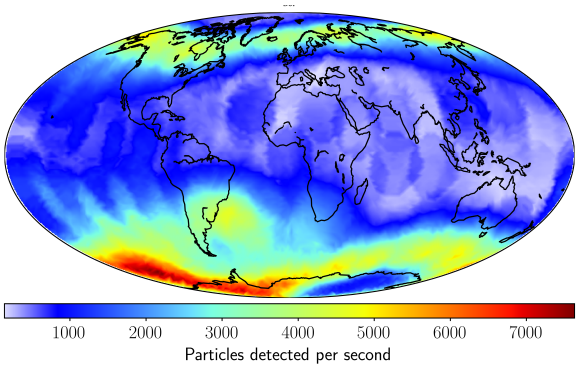


(d) Track curly

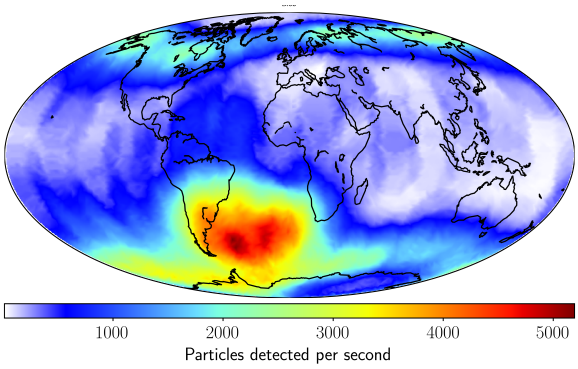


(e) Track lowres

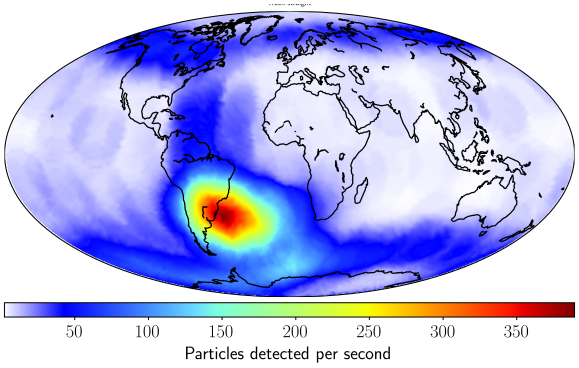
Figure 6.2: Global radiation maps generated with the use of the developed reconstruction pipeline applied on low resolution data from one whole-Earth scan. The scan consists of 887 images in binning 16 compression mode with histograms. Circles on maps are actual satellite measurements. The scan was performed from 14. 11. 2017 to 15. 11. 2017.



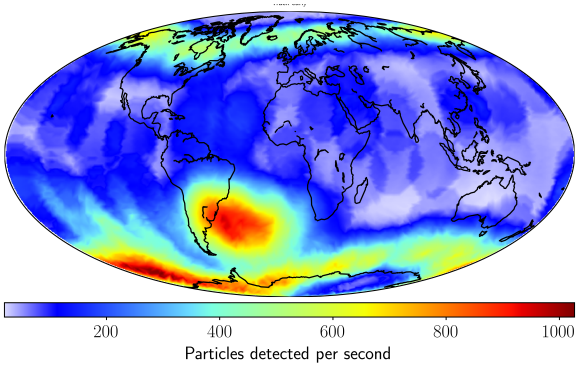
(a) Dot



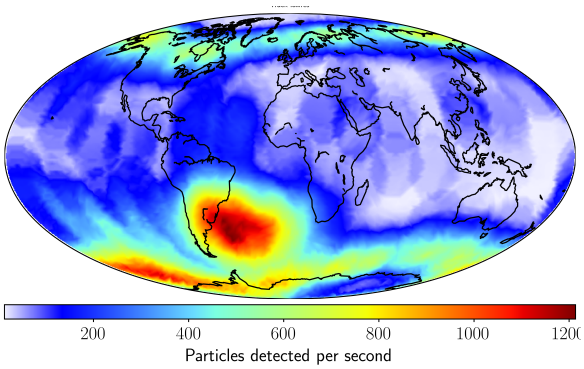
(b) Blob



(c) Track straight

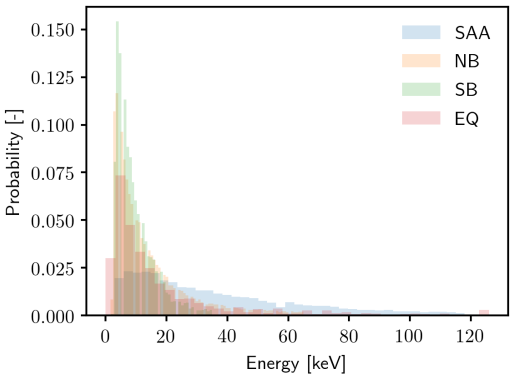


(d) Track curly

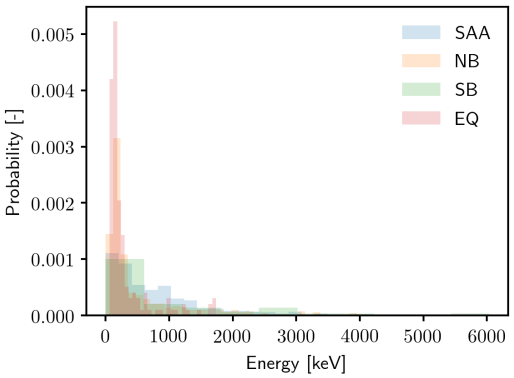


(e) Track lowres

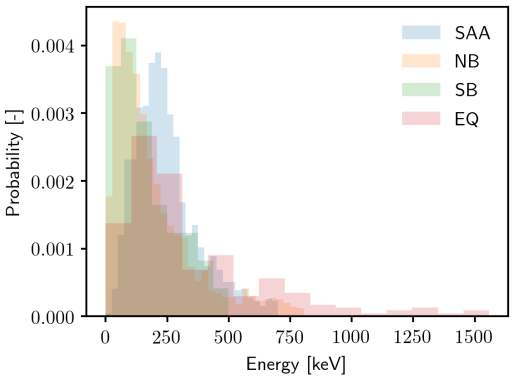
Figure 6.3: Global radiation maps generated with the use of developed reconstruction pipeline applied on low resolution data from 21 dosimetry sequences. The data were sampled from April 2017 to March 2018. Approximately one scan per week was performed.



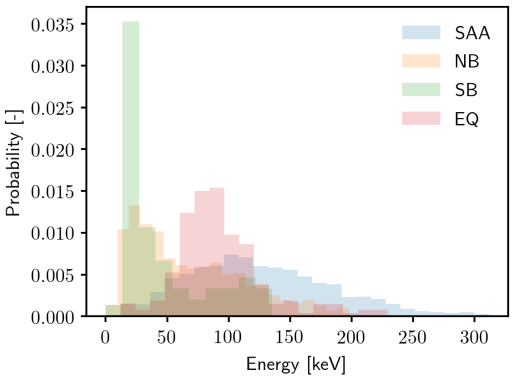
(a) Dot



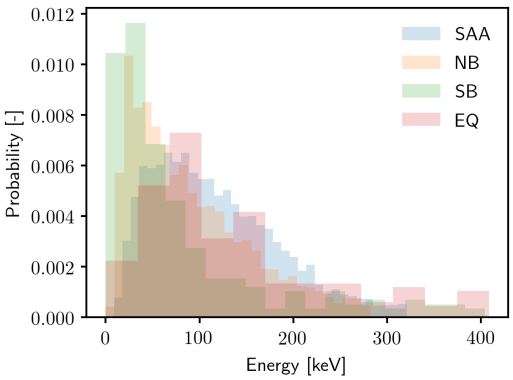
(b) Track straight



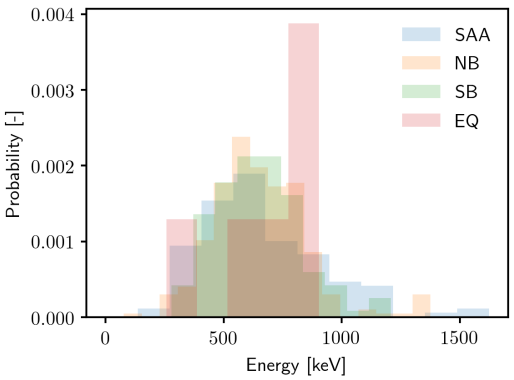
(c) Track curly



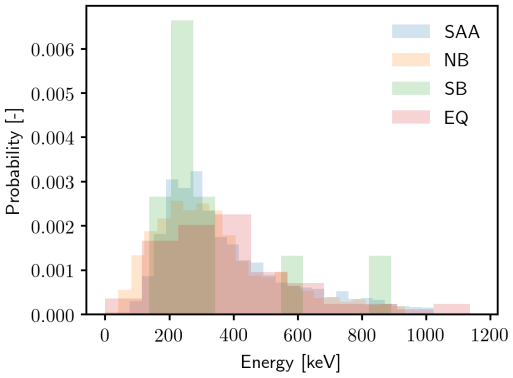
(d) Track lowres



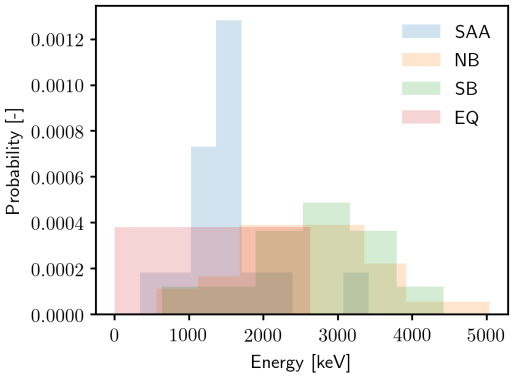
(e) Blob small



(f) Blob big



(g) Blob branched



(h) Drop

Figure 6.4: Energetic spectra for all particles.

6.2 Discussion of results and solution

This section is focused on a discussion of the obtained results. Since work on this thesis was highly experimental and full of dead-ends, the practical background is introduced together with reasons which led to the solution described in chapters above. Moreover, some interesting ideas which were developed (but not used in the final solution) are presented. The final solution is discussed in several logical parts: the data organization, classifier, regressor and visualizer. Improvement proposals for future missions are made as a part of discussion.

6.2.1 Uncertainity of the space project and time constraints

The pipeline for processing of VZLUSAT-1 dosimetry measurements was successfully created, despite several technical challenges. The whole work was complicated by the fact that very few resources were found in the available databases on the classification of charged particles. Moreover, all the available papers were concerned only with laboratory experiments. Since no available source dedicated to processing of data from experiments in the space environment was found, there was a problem with annotation of the training set, as it was not clear how the traces of different particles would look in the pixel detector. Because of that, the detected particles were described according to the shape characteristics of their tracks. It was expected that these characteristics would reflect physical reality, e.g., that all particles leaving blob-like track will have similar physical properties, but the precise classification of particle traces will be left to future analysis by particle physics experts when more (and better) data from other experiments will be collected.

The original assumption was that a subset of the images downloaded in full resolution would be annotated by the proprietary state-of-the-art method and these results will be used for the design of the classifier in this thesis. However, no such method was found, despite the effort to collaborate with the team of ESA's Probe-V researchers, who process data from the same sensor from the SATRAM experiment. Thus the data were annotated manually.

Meantime, an artificial image generator was created, which did not require the original images. Artificial images were generated from parameterized 2D functions. Classification work has been postponed and outputs of this generator used to design a regression model. Based on the generator outputs, a regression model was created that was able to accurately estimate the number of particles. This model was built on the a priori knowledge of energy histograms of individual particles, which it tried to combine so that the energy histogram of the image (which we know) equals the sum of energy histograms of individual particles. Due to the promising results, it was planned to apply the model to the real images when a functional classifier was available, but then it turned out that real images have so complex energy histograms that the method could not be used and was replaced by a black-box machine learning techniques. The inefficient generator itself has also been replaced by a solution that uses a generative model trained on full resolution images and which is described in this thesis.

6.2.2 Implementation and DataManager

All the software was implemented in the Python language, which allowed for easy and fast prototyping. Although, due to its interpreted nature, it is a relatively slow language, most of the necessary operations are fast enough due to the availability of efficient modules such as numeric library NumPy, scikit-image image processing library or a machine learning library, scikit-learn. A major issue (from a Python implementation point of view) has been the data visualization, especially interpolation in a spherical map. Python is probably currently not sufficiently equipped with geography analysis tools (if we limit ourselves to open-source projects) and the proposed implementation of some algorithms (such as kriging) in Python has shown to

be very inefficient. It also became obvious that the dynamic nature of Python complicates the development of programs with a more complex structure which contains many data types and classes. For future production-level work in the field of particle physics with more time available, it would maybe pay off to explore the `C++` ROOT toolkit developed in CERN.

Most of the key functionality has been implemented using an object-oriented approach. The basis of all of the program infrastructure has become a data organizer that is reliable and memory-efficient – this is mainly achieved by the use of the SQLite database as a backend. The solution makes it possible to carry out all the operations presented in this thesis using an average hardware (although some may take hours to complete). SQL solution is more complex (as opposed to storing data in a .csv file or simple tables), but it allows quick data filtering by attributes, which is very important in this type of work. Because the database file contains only relative references to the image files, it can be easily transfered between users or archived with the use of a version control system.

6.2.3 Classifier

Number of annotated particle examples is relatively small and ranges between tens to hundreds of examples, depending on the particular classes. The number of examples on which the classifier can be trained is even smaller, since part of the available data must be put aside and used for testing. Cross-validation was used to partially alleviate the problem, which made it necessary to divide the available dataset not into three standard sets (training, validation and testing) but only two (training, testing). However, there was still a problem with particle classes that are relatively rare (as presented in table 6.1). Three classifiers (which were assumed to perform relatively well with small amount of data) were tested. The random forest-based solution showed to be the best (with test Mathews correlation coefficient (MCC) 0.7905). It was followed by the support vector machine with MCC 0.7689 and the logistic regression with MCC 0.7438.

	Dot	**Blob small**	**Blob big**	**Blob branched**	**Track straight**
Counts	705	382	288	448	332
	Track curly	**Drop**	**Other**	**Track lowres**	
Counts	473	49	93	134	

Table 6.1: Numbers of particle types in the training dataset.

With the use of a confusion matrix (figure 6.5) it is possible to evaluate misclassification rates for each class. Clearly the classifier performance is very good for particles of types Dot (0), Track small (1) and Blob heavy (2). The classifier often confuses classes Blob branched (3) and Track curly (5). Classes Drop, Track lowres and Other are problematic since there were not enough data for their classification.

6.2.4 Regression model

The regression model, which was accepted as the final solution, is built around a random forest estimator (with training errors depicted in table 5.6). The second model with which the random forest was compared is based on a feed-forward neural network that was pretrained on a set of data generated by the generative model based on kernel density estimate of distribution of counts in real frames. Surprisingly, a neural network pretrained on an artificial dataset with 40000 training samples has achieved much worse results than a random forest (5.4). None of the results are truly optimal, but, at least, the random forest regressor is able to deliver results that make physical sense. The random forest is known to be able to work with data with little preprocessing. Maybe the neural network would be able to surpass its performance when more

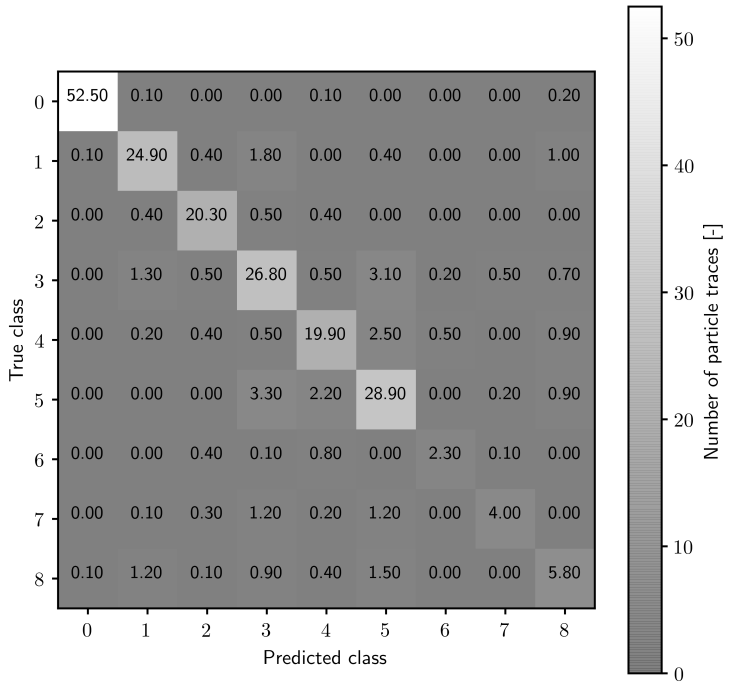


Figure 6.5: Confusion matrix of the classifier, generated with the use of 10-fold crossvalidation from testing part (25%) of the whole full resolution dataset.

care is given to feature preprocessing, but because of the length of neural network training, the experimentation was difficult.

However, the main problematic point is probably again the lack of training data (the whole real dataset contained only 493 training samples). This claim is supported by the fact that the neural network validation curve (figure 5.5) generated from the large artificial dataset converged quickly to relatively low error – lower than the error of both network and random forest trained on the actual dataset. The large final error of the pretrained neural network on the real data also suggests that the generated data are probably different from the real ones.

6.2.5 Interpolation and visualization

The task of interpolation was mainly a task of trial-and-error with the goal of finding interpolating (or approximating) algorithm which produces results which contain as small number of artifacts as possible. Several techniques were considered. For spherical splines only one working Python implementation (in SciPy library) was found. This implementation uses splines only in smoothing mode and it was not possible to get reasonable results during parameters setting. Radial basis function approach (also from SciPy library) performed better, but created false misleading local minima in interpolated surface due to the low density of points. Experiments were also carried with (ordinary) kriging interpolation, common geostatistical method, but because it was implemented from scratch it was very slow. Moreover, this method needs proper parameter tuning to work well and the results are not intuitive. Finally, IDW combined with k-NN algorithm (for interpolation of separate scans) and pure k-NN regression algorithm (for interpolation of merged scan with high density of the measurements) were chosen for interpolation. Although these algorithms are not as sophisticated as current state-of-the-art, their advantage is intuitive behavior and speed.

Throughout the work, it was not certain whether the entire pipeline for information reconstruction from compressed images would have a sufficiently low error rate to create visualizations. Finally the pipeline performs impressively, at least from qualitative point of view (sections 6.1.2,

6.1.1). Unfortunately, there is not enough full resolution data available to perform accurate quantitative verification of the entire chain's functionality. However, when looking at visualizations (figures 6.3 and 6.2), it is obvious that the results make sense physically – the visualization contains all the theoretically expected phenomena such as the South Atlantic Anomaly and the North and South Radiation Horns.

According to the visualizations, it is evident that each particle class is governed by its own spatial distribution. The occurrence of Track curly (6.2c, 6.3c) and Track lowres (6.2e, 6.3e) class is spatially dependent. It may either be a regression model error or it is due to the physical nature of the matter. We can probably exclude the classifier error because the track curly energy spectrum is very different from the spectrum of track lowres.

Spatial particle distribution of Dot is also interesting. Since photons do not interact with Earth's magnetosphere, it would be expected that their distribution over the Earth's surface will be uniform, which is not the case. The North and South Radiation Horns contain large amounts of photons and there is also a small amount of photons in region of the South Atlantic Anomaly. Hypothetically, it might be that the observed photon flux inhomogeneities are caused by interaction of charged particles with magnetosphere. These photons might be also produced as secondary radiation by the satellite's hull.

6.2.6 Energetic spectra

The energetic spectra of different particle types were calculated on the basis of particles that were identified in the full resolution images. Although the spectra were generated for different areas, the most informative are the ones for particles in the South Atlantic Anomaly. This is probably due to the fact that during sampling of full resolution images the emphasis was put mainly on this area. Northern, southern belt and images from equatorial region contain small amounts of particles, and, probably due to low sample density in these areas, they do not adequately capture the structure of radiation belts.

From the available data, it is possible to say that the energy spectra are predominantly of the Poisson distribution. Photons and tracks with the highest energies (up to 100 keV for photons and several MeV for track straight) are found mostly in the SAA region (figure 6.4). Energy spectra of blob-like particles are more or less similar in all areas (6.4). For drop type particles, the analysis is difficult because this particle is a rare phenomenon which was observed only a few dozen times in all the full resolution frames. Nevertheless, it is interesting that this class contains particles with highest energies observed – up to several MeV.

Chapter 7

Conclusion

This thesis, which deals with a design of pipeline for partial reconstruction of heavily compressed Timepix images of ionizing particle traces from the VZLUSAT-1 satellite, successfully fulfilled all requirements of the assignment.

During initial steps, the software tools for efficient data organisation and filtering were developed. The software design is centred around data manager entity which forms a universal interface between SQL database on the one side and arbitrary routines and user interfaces on the other side. This fundamental infrastructure is used as a bridge for data generated by subsequently designed steps of the reconstruction pipeline: the classifier, regressor and visualizer. It also allows efficient storage and filtering of incoming data from the satellite.

The classifier gets full resolution images on its input, performs segmentation, feature extraction and classification. The goal of the classifier is to assign one label to each segment, which represents one particle trace. Nine particle classes are considered, all of them are based purely on geometrical properties of segmented tracks of detected particles. Each segment is described by 25 features, which are based on statistical properties of individual pixel values, geometrical analysis of images and also on mathematical morphology. Random forest, support vector machine and logistic regression were tested in place of classification algorithms. Random forest achieved the best score with Matthews correlation coefficient 0.83.

The classifier was used to perform counting-by-detection task for images in a full resolution dataset. These counts and full resolution images served two main purposes. The full resolution images were processed by routines mimicking the internal satellite compression methods and, together with the counts, formed input-output training pairs for regressor, whose task is to perform counting-by-regression task on compressed data from the satellite. The counts also form a basis for a generative model (based on a kernel density estimation) which estimates the probability distribution of observing an image with given particle counts. This model can generate new samples from this distribution. These samples (vectors of particle counts) were used to generate new full resolution images via geometric transformations of particle tracks from real full resolution images.

Both the real and the artificial image datasets were (after feature extraction) used in the design process of a counting regressor. The extracted features are based on simple pixel values, texture analysis method and energetic histograms of the images. Since the generative model generated 50000 artificial images, it was originally assumed that the best model will be a neural network implemented in Keras, pretrained on this huge dataset and then fine-tuned on the small, real dataset (which contained only 493 samples). Nevertheless, the neural network (even though structure optimization was performed) achieved very high error rates and was overcome by a random forest regressor which was trained only on a real dataset (numerical values are in figures 5.4, 5.6). Because of this result the random forest regressor was chosen as a regression algorithm in the final implementation.

Since the whole-Earth dosimetry data sampled by the satellite were sparse in space and non-uniformly sampled on a spherical surface, interpolation and projection was performed. Algorithm for a geodesic grid generation with two-step spherical-planar interpolation was proposed and designed with the goal of smoothing the data with a minimum of artifacts. After experimenting with different interpolation and approximation methods (such as kriging, spherical splines and rbf interpolation) the spherical interpolation method was based on k-NN algorithm combined with inverse distance weighting.

Two outputs of the pipeline were finally collected after application on compressed data of the VZLUSAT-1 satellite. The primary outputs of this thesis – ionizing particles densities estimated from the compressed data sampled above the whole Earth – contain all the characteristic features as they theoretically should: South Atlantic Anomaly and North and South radiation belts. Energetic spectra of detected particles, which were generated in different regions of orbit, are the secondary quantitative outcome of this thesis.

Since this project seems to be one of the first ones of its kind in the given field (at least between projects with publicly available results), the obtained results can be hardly validated and compared with results of other, similar projects. It must also be noted that the results were obtained in extreme conditions, from data which were mostly supposed to be used for another purpose (exposures with the X-ray telescope) – so the results should be judged mainly from the qualitative point of view.

Although this thesis should not directly suggest better compression methods for further missions, it is worth pointing out that the current compression solution (binning) is highly unsuitable for transferring images which contain particles other than X-ray photons. Better solution for dosimetry measurements would be to move data evaluation pipeline to the satellite, ideally with the possibility of remotely modifying the processing chain from the Earth. Nevertheless, due to the limited computing capabilities of the satellite, it might happen that only the elementary on-board preprocessing would be feasible.

For example, the images might be segmented and categorized into several elementary shape-based classes by the satellite. Each segment would be compressed by a method that would be optimized for a given particle class and sent to the Earth. It might also be possible to apply a more coarse quantization of amplitude of each segment pixels, which would probably not seriously degrade the performance of a robust classifier located on the Earth, when the quantization levels will be chosen wisely.

Bibliography

[1] The Von Karman Institute for Fluid Dynamics, "Qb50 project." `https://www.qb50.eu/`.

[2] T. Baca, M. Platkevic, J. Jakubek, A. Inneman, V. Stehlikova, M. Urban, O. Nentvich, M. Blazek, R. McEntaffer, and V. Daniel, "Miniaturized X-ray telescope for VZLUSAT-1 nanosatellite with Timepix detector," *Journal of Instrumentation*, vol. 11, no. 10, p. C10007, 2016-10-01.

[3] R. A. Darwish, A. H. Staudacher, E. Bezak, and M. P. Brown, "Autoradiography imaging in targeted alpha therapy with Timepix detector," *Computational and Mathematical Methods in Medicine*, vol. 2015, pp. 1–7, 2015.

[4] "Design and development of petipix," *2013 IEEE Nuclear Science Symposium and Medical Imaging Conference (2013 NSS/MIC)*, pp. 1–4, 2013.

[5] A. S. Tremsin, J. B. McPhate, J. V. Vallerga, O. H. W. Siegmund, W. Kockelmann, A. Steuwer, and W. B. Feller, "High-resolution neutron counting sensor in strain mapping through transmission bragg edge diffraction," *IEEE Sensors Journal*, vol. 11, no. 12, pp. 3433–3436, 2011.

[6] C. Granja, S. Polansky, Z. Vykydal, S. Pospisil, A. Owens, Z. Kozacek, K. Mellab, and M. Simcak, "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*, vol. 125, pp. 114–129, 2016.

[7] N. Stoffle, L. Pinsky, M. Kroupa, S. Hoang, J. Idarraga, C. Amberboy, R. Rios, J. Hauss, J. Keller, A. Bahadori, E. Semones, D. Turecek, J. Jakubek, Z. Vykydal, and S. Pospisil, "Timepix-based radiation environment monitor measurements aboard the International Space Station," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 782, pp. 143–148, 2015.

[8] T. A. collaboration, "A neural network clustering algorithm for the ATLAS silicon pixel detector," *Journal of Instrumentation*, vol. 9, no. 09, pp. P09009–P09009, 2014-09-01.

[9] S. Hoang, *A pattern recognition approach to learning tracks of heavy-ion particles in Timepix detectors.* Dissertation, University of Houston, Houston, 2013.

[10] J. Čermák, "Particle track recognition in Timepix pixel detector," diploma thesis, MFF UK, Prague, 2013.

[11] J. Bartovsky, D. Schneider, E. Dokladalova, P. Dokladal, V. Georgiev, and M. Akil, "Morphological classification of particles recorded by the Timepix detector," in *2011 7th International Symposium on Image and Signal Processing and Analysis (ISPA)*, pp. 343–348, Sept 2011.

[12] W. Spjedlvik and P. Rothwell, *Handbook of geophysics and the space environment.* Air Force Geophysics Laboratory, 1 ed., 1985.

[13] *Guide to modeling Earth's trapped radiation environment.* Reston, VA: American Institute of Aeronautics and Astronautics, 1999.

[14] M. Moldwin, *An introduction to space weather.* Cambridge: Cambridge University Press, 2008.

[15] D. N. Baker, P. J. Erickson, J. F. Fennell, J. C. Foster, A. N. Jaynes, and P. T. Verronen, "Space weather effects in the Earth's radiation belts," *Space Science Reviews*, vol. 214, p. 17, Dec 2017.

[16] NASA, "Radiation belts with satellites." `https://www.nasa.gov/mission_pages/sunearth/news/gallery/20130228-radiationbelts.html`.

[17] E. Daly, "The radiation belts," *Radiation Physics and Chemistry*, vol. 43, no. 1-2, pp. 1–17, 1994.

[18] Amsler *et al.*, "Review of Particle Physics, 2008-2009. Review of Particle Properties," *Phys. Lett. B*, vol. 667, no. 1-5, pp. 1–6, 2008.

[19] S. Tavernier, "Interactions of particles in matter," *Experimental Techniques in Nuclear and Particle Physics*, pp. 23–53, 2010.

[20] M. Platkevič, *Signal processing and data read-out from position sensitive pixel detectors.* Dissertation, ČVUT v Praze, Praha, 2014.

[21] R. Ballabriga, M. Campbell, and X. Llopart, "ASIC developments for radiation imaging applications," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 878, pp. 10–23, 2018.

[22] J. Jakůbek, "Pixel device TimePix." `http://aladdin.utef.cvut.cz/ofat/others/Timepix/`.

[23] J. Jakubek, "Precise energy calibration of pixel detector working in time-over-threshold mode," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 633, pp. S262 – S266, 2011. 11th International Workshop on Radiation Imaging Detectors (IWORID).

[24] J. Bouchami, A. Gutiérrez, A. Houdayer, J. Idárraga, J. Jakůbek, C. Lebel, C. Leroy, J.-P. Martin, M. Platkevič, and S. Pospíšil, "Study of charge sharing in a silicon pixel detector with heavy ionizing particles interacting with a Medipix2 device," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 607, no. 1, pp. 196 – 198, 2009. Radiation Imaging Detectors 2008.

[25] Z. Vykydal and J. Jakubek, "Usb Lite—miniaturized readout interface for Medipix2 detector," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 633, pp. S48 – S49, 2011. 11th International Workshop on Radiation Imaging Detectors (IWORID).

[26] M. Urban, O. Nentvich, V. Stehlikova, T. Baca, V. Daniel, and R. Hudec, "VZLUSAT-1," *Acta Astronautica*, vol. 140, 2017.

[27] J. Masopust, I. Veřtát, R. Linhart, A. Voborník, V. . Daniel, and P. Svoboda, "První český nanosatelit," 2018.

[28] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'12, (USA), pp. 1097–1105, Curran Associates Inc., 2012.

[29] T. Holy, E. Heijne, J. Jakubek, S. Pospisil, J. Uher, and Z. Vykydal, "Pattern recognition of tracks induced by individual quanta of ionizing radiation in Medipix2 silicon detector," pp. 287–290, 06 2008.

[30] N. M. Zaitoun and M. J. Aqel, "Survey on image segmentation techniques," *Procedia Computer Science*, vol. 65, pp. 797 – 806, 2015. International Conference on Communications, management, and Information technology (ICCMIT'2015).

[31] C. Fiorio and J. Gustedt, "Two linear time union-find strategies for image processing," *Theoretical Computer Science*, vol. 154, no. 2, pp. 165 – 181, 1996.

[32] W. E. Snyder and H. Qi, *Machine vision*. New York: Cambridge University Press, c2004.

[33] S. Roman, *Advanced linear algebra*. New York [N.Y.]: Springer, 3rd ed. ed., c2008.

[34] "Scipy documentation." `https://docs.scipy.org/doc/scipy/reference/tutorial/ndimage.html#distance-transforms`, 2017.

[35] S. Aksoy and R. M. Haralick, "Feature normalization and likelihood-based similarity measures for image retrieval," *Pattern Recognition Letters*, vol. 22, no. 5, pp. 563 – 582, 2001. Image/Video Indexing and Retrieval.

[36] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *J. Mach. Learn. Res.*, vol. 3, pp. 1157–1182, Mar. 2003.

[37] S. Shalev-Shwartz and S. Ben-David, *Understanding machine learning*. New York, NY, USA: Cambridge University Press, 2014.

[38] S. Theodoridis and K. Koutroumbas, *Pattern recognition*. Burlington, Mass.: Academic Press, 4th ed ed., c2009.

[39] J. R. Quinlan, "Induction of decision trees," *Machine Learning*, vol. 1, pp. 81–106, Mar 1986.

[40] L. Khaidem, S. Saha, and S. R. Dey, "Predicting the direction of stock market prices using random forest," *CoRR*, vol. abs/1605.00003, 2016.

[41] G. Jurman, S. Riccadonna, and C. Furlanello, "A comparison of mcc and cen error measures in multi-class prediction," *PLOS ONE*, vol. 7, pp. 1–8, 08 2012.

[42] C. C. Loy, K. Chen, S. Gong, T. Xiang, C. C. Loy, K. Chen, S. Gong, T. Xiang, C. C. Loy, K. Chen, S. Gong, and T. Xiang, "Crowd counting and profiling: Methodology and evaluation."

[43] Y.-C. Chen, "A Tutorial on Kernel Density Estimation and Recent Advances," *ArXiv e-prints*, Apr. 2017.

[44] S. van der Walt, J. L. Schönberger, J. Nunez-Iglesias, F. Boulogne, J. D. Warner, N. Yager, E. Gouillart, T. Yu, and the scikit-image contributors, "cikit-image: image processing in Python," *PeerJ*, vol. 2, p. e453, 6 2014.

[45] W. Loh, "Classification and regression trees," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 1, no. 1, pp. 14–23.

[46] F. R. Hoots and R. L. Roehrich, "Spacetrack report no. 3," 31 December 1988.

[47] T. Vincenty, "Direct and inverse solution of geodesics on the ellipsoid with application of nested equations," *Survey review*, vol. 13, 1975.

[48] P. Alfeld, "Scattered data interpolation in three or more variables," in *Mathematical Methods in Computer Aided Geometric Design* (T. LYCHE and L. L. SCHUMAKER, eds.), pp. 1 – 33, Academic Press, 1989.

[49] I. B. Gundogdu, "Usage of multivariate geostatistics in interpolation processes for meteorological precipitation maps," *Theoretical and Applied Climatology*, vol. 127, pp. 81–86, Jan 2017.

[50] G. Bohling, "Kriging." `http://people.ku.edu/~gbohling/cpe940/Kriging.pdf`, 2005.

[51] K. Sahr, D. White, and A. Jon Kimerling, "Discrete global grid system," vol. 30, pp. 121–134, 04 2003.

[52] G. Hinshaw, D. Larson, E. Komatsu, D. N. Spergel, C. L. Bennett, J. Dunkley, M. R. Nolta, M. Halpern, R. S. Hill, N. Odegard, L. Page, K. M. Smith, J. L. Weiland, B. Gold, N. Jarosik, A. Kogut, M. Limon, S. S. Meyer, G. S. Tucker, E. Wollack, and E. L. Wright, "Nine-year wilkinson microwave anisotropy probe (wmap) observations: Cosmological parameter results," *The Astrophysical Journal Supplement Series*, vol. 208, no. 2, p. 19, 2013.

Appendices

Appendix A

Contents of the CD

The CD (figure A.1) contains source code, results of the thesis in pdf format and main results obtained after application of designed method to all data with sufficient quality which were available in May, 2018. The CD does not contain original images from the satellite and database file with information about them. For details, please refer to supervisor of this thesis, Ing. Tomáš Báča.

```
/
├── thesis.pdf
├── source_code...source code of the developed tools
└── results
    ├── dosimetries_binning16...generated maps of particle fluxes for
    │   whole-Earth scans
    ├── maps_full_resolution..generated maps of particle fluxes obtained from
    │   full resolution frames captured during the mission
    └── spectra_full_res...energy spectra calculated from all from full
        resolution frames captured during the mission
```

Figure A.1: Contents of the CD.

Appendix B

The source code

This appendix serves as a brief guide the source code shipped with this thesis. The code is located in the folder `source_code` on the CD. The whole software solution is composed of multiple subprograms whose main functionality can be accessed from this folder. All the source files are described in their header and in the comments.

The workflow for data processing consists of multiple steps. For basic database manipulations, the simplest solution is to run the script `database_gui.py`:

```
python database_gui -f <path to the image folder> -d <path to the database>
```

Simple GUI shows up. It allows to initiate the following operations:

- **full database update:** the database will be updated with new images and the segmentation and preprocessing will be performed. New database will be created when the the database is not present in the database path,
- **segments update:** the segments will be updated without loading new images,
- **initialization of GUIs for data exploration and annotation:** two GUIs can be accesses, the fully-featured Image explorer and the lightweight Segment explorer, which is dedicated for fast segment annotation.

The database updating is managed by the code in the file `main_preprocessing.py`. The preprocessing is executed in batches of images to prevent memory problems. The number of batches can be configured when this script is run from the command line – details are available in the source code.

The classifier can be executed after the database is populated with the processed data. This task is achieved by calling the script `main_classifier.py`. The script can be configured to perform various experimental tasks, but in the default implementation it only performs classification of segments in the database with classifier which was described in this thesis.

The visualizations can be generated by calling `main_map.py`. Before using see the source code – there is a more detailed guide to this operation.

The root folder also contains several additional files: `main_regressor.py`, `crossval_keras.py`, `regressor_experiments.py` and `genFromReal.py`. First two files are not important from the user's point of view–they were used for training and experiments with the regressor, whose final version is saved in file `regressor.pkl`. The file `genFromReal.py` starts the process of artificial data generation. The artificial data are needed for execution of the `crossval_keras.py` script.

The source code on the CD does not contain the database. For details, please refer to supervisor of this thesis, Ing. Tomáš Báča.

Appendix C

Alternative graybox regression technique

One of the additional attempts to the count estimation problem proposed in this thesis was the development of a graybox estimation, which relies on an intuitive physical explanation of energetic histograms of the images.

The core assumption of this method is that the energetic histogram of a received frame is a linear combination of energetic histograms of all the detected particles. If we assume there will be, e.g., three types of particles, whose pixels will have values according to their known energetic histograms E_1, E_2, E_3, we can find linear combination of these histograms, which must sum to the composite energetic histogram E_c, according to the equation

$$n_1 S_1 E_1 + n_2 S_2 E_2 + n_3 S_3 E_3 = E_c, \tag{C.1}$$

where n_i is the number of a i-th class of particles and S_i is the expected area of the i-th kind of particles. Goal is to find all counts n_i. We formulate this idea as a matrix equation

$$\begin{bmatrix} E_{1_{bin1}} & E_{2_{bin1}} & E_{3_{bin1}} \\ E_{1_{bin2}} & E_{2_{bin2}} & E_{3_{bin2}} \\ E_{1_{bin3}} & E_{2_{bin3}} & E_{3_{bin3}} \\ E_{1_{bin4}} & E_{2_{bin4}} & E_{3_{bin4}} \\ \vdots & \vdots & \vdots \\ E_{1_{binN}} & E_{2_{binN}} & E_{3_{binN}} \end{bmatrix} \begin{bmatrix} S_1 & 0 & 0 \\ 0 & S_2 & 0 \\ 0 & 0 & S_3 \end{bmatrix} \begin{bmatrix} n_1 \\ n_2 \\ n_3 \end{bmatrix} = \begin{bmatrix} E_{c_{bin1}} \\ E_{c_{bin2}} \\ E_{c_{bin3}} \\ E_{c_{bin4}} \\ \vdots \\ E_{c_{binN}} \end{bmatrix}. \tag{C.2}$$

The vector $[n_1, n_2, n_3]^T$ is found by multiplication of the whole equation by the appropriate pseudoinverse matrix from the left. The energetic histograms of particles are not exactly known, but can be estimated from the dataset, e.g., with the use of the kernel density estimator. The energetic distributions of particles possibly depend on many other factors like activity of the Sun and probe's position, so we should work with the energetic histogram of the particle class i as with conditional probability distribution $E_i \equiv P_i(E|I)$, so $p_i(e|im)$ is a probability that pixel activated by the particle of a class i will have an energetic level e given that this histogram was generated by image im. If we approximate distributions $P_i(I, E)$ and $P_i(I)$, we then calculate the $P_i(I, E)$ in the following way:

$$P_i(E|I) = \frac{P_i(E, I)}{P_i(I)}. \tag{C.3}$$

This methodology was tested on a dataset generated by one of the oldest versions of the parametric simulator, which can produce only particles of Track straight and Dot types.

Dataset for estimation of energetic histogram consisted of 100 images with approximately 40000 segments. Then the estimator was tested on previously unseen images and for each of these images the system C.2 was solved. Example of resulting decomposition of the original histogram into histograms of Tracks and Dots can be seen in figure C.1.

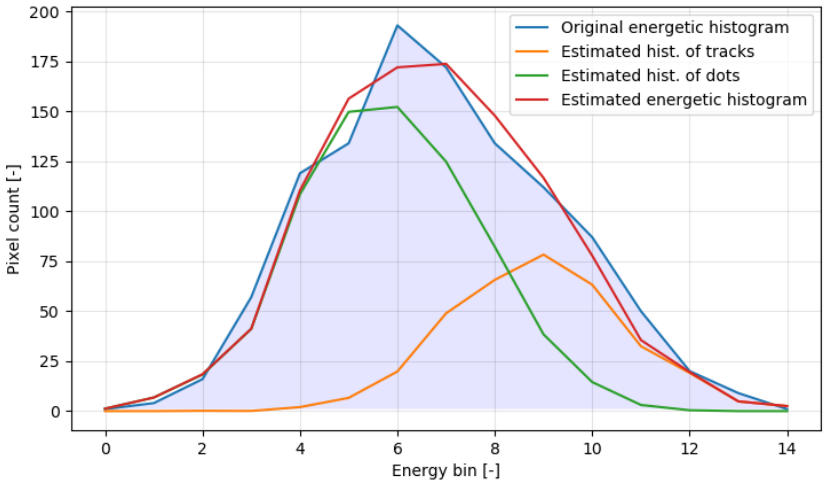


Figure C.1: An example of estimated energetic histograms for image, generated by an early version of artificial image generator. The red estimated energetic histogram is a sum of estimates of the histogram of dots (green) and the histogram of tracks (orange). The blue histogram is a ground truth.

Estimation was repeated for 15 images and error metrics were calculated (table C.1).

Particle type	**RMSE**	**Mean relative error**
Track straight	3.75	-0.011
Dot	3.14	-0.012

Table C.1: Performance of the gray-box approach. MSE stands for mean squared error, mean relative error is mean ratio of true and predicted particle counts in the whole test dataset.

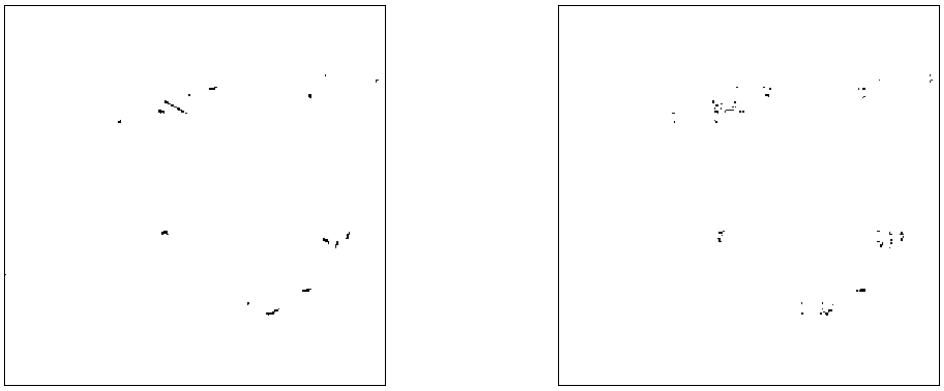
The initial results exceeded expectations. Nevertheless, when tested on the real dataset when the annotated data were available, it performed unsatisfactorily. The predicted counts were in order of tens or thousands of particles, sometimes the counts were lower then zero. It was possibly caused by higher number of classes and complicated energetic histograms. The poor performance on the real dataset is the reason why this method was not described in the chapter 5. Nevertheless, maybe it would be possible to add more constraints to solution of the system C.2 which would make this method perform well on the real dataset in the future.

Appendix D

Reconstruction of the full resolution images

An attempt was made to reconstruct the complete full resolution images from its downloaded binned version together with the row and column sums. The problem was formulated as a constraint satisfaction problem (CSP).

The constraint satisfaction problem tries to find values of a set of variables from some finite domain, which satisfies given constraints. The task of full resolution image reconstruction can be formulated as CSP where the variables are all the 256×256 individual pixels where each pixel has value of 0 or 1. The constraints can be obtained from reversing the binning and row/column summing process. The binning constraint can be expressed as a constraint that the sum of values of (binarized) full resolution image pixels in a given square (which corresponds to one bin) must be equal to the corresponding binned "superpixel". The row sum constraint expresses that the sum of pixels in particular row of the (binarized) full resolution image must be equal to the corresponding entry of the row sum vector. The column sum constraint is similar.



(a) Original image (b) CSP reconstruction

Figure D.1: Example of reconstruction of an image captured with binning 8 mode. The image D.1a shows the original full resolution image is visible, D.1b depicts the reconstructed variant.

Results of this method can be observed in figure D.1. Clearly, the locations of the particles were approximately recovered, but still it is not possible to identify the original particle trace classes. This is probably a consequence of the fact that we have (much) less constraints than variables. Nevertheless, this problem may be solved by introducing more constraints into the problem.