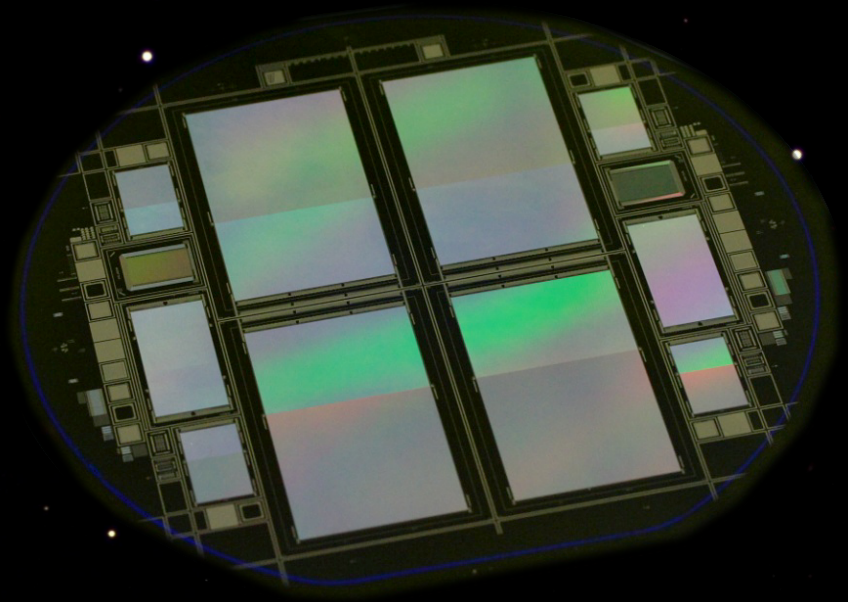
Detector performance of eROSITA

Michael Wille



Friedrich-Alexander-Universität
Erlangen-Nürnberg

ERLANGEN CENTRE
FOR ASTROPARTICLE
PHYSICS

Detector performance of eROSITA

Diploma thesis

Michael Wille

Dr. Karl-Remeis Sternwarte Bamberg & ECAP
Friedrich-Alexander-Universität Erlangen-Nürnberg

Supervisor:
Prof. Dr. Jörn Wilms

April 2011

Friedrich-Alexander-Universität Erlangen-Nürnberg

ECAP ERLANGEN CENTRE FOR ASTROPARTICLE PHYSICS

The illustration on the titlepage shows a wafer with four eROSITA CCDs on it (Picture: Meidinger et al., 2009). Underneath, "X-ray imaging data" containing two bad pixels and the signature of an AGN is drawn. The galaxy which can be seen is NGC2683 and was photographed with a MEADE 40 cm telescope and a SBIG STL-11000M CCD camera (Credit: Fürst, Kühnel, Wille)

Contents

List of acronyms

ABRIXAS A Broadband Imaging X-ray All-Sky Survey

ADU Analog Digital Unit

AGN Active Galactic Nuclei

Aux Auxiliary

CCD Charge Coupled Device

CCF Current Calibration File

CPU Central Processing Unit

ECAP Erlangen Centre for Astroparticle Physics

EPIC European Photon Imaging Camera

EXOSAT European X-Ray Observatory Satellite

eROSITA extended ROentgen Survey with an Imaging Telescope Array

FITS Flexible Image Transport System

FoV Field of View

FPIPE FITS-pipeline

GSFC Goddard Space Flight Center

GTI Good time interval

HDU Header Data Unit

HEASARC High Energy Astrophysics Science Archive Research Center

HK Housekeeping

HST Hubble Space Telescope

HXD Hard X-ray Detector

IAAT Institut für Astronomie und Astrophysik Tübingen

ISDC *INTEGRAL* Science Data Center

ISS International Space Station

KS Kolmogorow-Smirnow

MIPS Minimum-Ionizing Particles

MOS Metal Oxide Semiconductor

MPE Max Planck Institut für Extraterrestrische Physik

NASA National Aeronautics and Space Administration

NRTA Near Real Time Analysis

PCA Proportional Counter Array

PSF Point Spread Function

ROSAT Roentgensatellit

RXTE Rossi X-ray Timing Explorer

SASS Standard Analysis Software System

SIXT Simulation of X-ray Telescopes

SRG Spectrum Roentgen Gamma

WFPC Wide Field and Planetary Camera

WMAP Wilkinson Microwave Anisotropy Probe

XML eXtensible Markup Language

XMM-Newton X-Ray Multiple Mirror-Newton

Typographic Conventions

In this diploma thesis, several typographic conventions are used. The following list describes their meanings.

- Titles of satellites and satellite missions are typeset in *slanted font* (e.g., *X-Ray Multiple Mirror-Newton (XMM-Newton)*)
- Names of programs and software libraries are typeset in *slanted font* (e.g., *SIXT*)
- Function names are typeset in *italic font* (e.g., *fillImageArray*)
- Parameters and corresponding values are typeset in `typewriter font` (e.g., `RAWX=384`)
- Listings such as console output, are typeset in `typewriter font`

Schematic Conventions

In this work, flow diagrams illustrate the logical structures of functions which are part of the software being discussed. The following conventions apply to these diagrams:

- Colored flow diagrams resemble functions which are directly called by the *main*-function whereas gray ones are one or more steps further down the hierarchy
- Single-headed arrows indicate the application flow
- Double-headed arrows indicate the direction of data input/output
- Diamonds resemble points where a decision has to be made. Depending on the result the application flow will continue following the *yes*- or the *no*-line respectively
- Boxes which have double lines at their top and left border indicate structures (which in turn may contain further variables, arrays, structures etc.)
- Depending on their caption, simple boxes resemble either function calls or a particular action which will be performed immediately. In case of function calls the box colour helps to find the respective flow diagram in order to have a closer look at the function
- Cylindric shapes stand for input/output FITS-files

Abstract

The extended ROentgen Survey with an Imaging Telescope Array (eROSITA) mission, which is scheduled for launch in 2013, will perform an all-sky survey up to an energy of 10 keV with unprecedented resolution. The mission is realised by a collaboration of German institutes. eROSITA will be part of the instrumentation aboard the Russian *Spectrum Roentgen Gamma (SRG)* satellite and reach its final destination in Earth's L_2 orbit. According to estimates, eROSITA will detect up to 100 000 galaxy clusters which can provide valuable information on the validity of current cosmological models and on the properties of Dark Energy. eROSITA will be equipped with 7 Wolter telescopes, each of them having a pn-CCD chip in its focal plane. With the *NRTA*, a software framework is in development which is intended to monitor instrument health and detect hardware problems as soon as possible.
In the course of this thesis, a method was worked out which is applicable for the detection of bad pixels in CCD chips. This is made possible by statistical analysis of the event lists which are generated by the detector. The realisation of this concept yielded two programs capable of recognising and cataloguing bad pixels and which can be integrated to the existing *NRTA* framework without any problem. Both programs underwent extensive testing with data from the *XMM-Newton* satellite mission.
Additionally, an extension to the *SIXT* software was developed. *SIXT* (programmed by C. Schmid) is capable of simulating various X-ray optics and -detectors and thus can provide valuable information on, e.g., their imaging properties. The extension allows *SIXT* to incorporate realistic particle background to its simulations. This is achieved by selecting background events from a prefabricated list in a random fashion, according to the particle statistics which is to be expected. Test simulations have shown that the background library fits seamlessly into the existing *SIXT* framework and that it works as expected.

Deutsche Zusammenfassung

Die für das Jahr 2013 geplante eROSITA Mission, die durch Zusammenarbeit zahlreicher deutscher Institute realisiert wird, soll den Himmel im Röntgenbereich bis zu einer Energie von 10 keV mit bislang unerreichter Auflösung kartographieren. Zu diesem Zweck wird eROSITA an Bord des russischen *SRG* Satelliten in einen Orbit um den L_2 Punkt (im System Erde – Sonne) gebracht. Danach wird eROSITA Schätzungen zufolge bis zu 100 000 Galaxienhaufen detektieren, deren Wechselwirkungen wertvolle Informationen über die Konsistenz kosmologischer Modelle und die Eigenschaften der Dunklen Energie liefern können. eROSITA wird mit 7 Wolter-Teleskopen und je einem bildgebenden pn-CCD Chip in den Fokalebenen ausgestattet sein. Um den Betriebszustand von eROSITA zu überwachen und Hardwarefehler sowie kleinere Defekte rechtzeitig zu erkennen, wird mit der *NRTA* eine Softwarelösung zur schnellen Analyse von Telemetriedaten entwickelt.
Im Rahmen dieser Diplomarbeit wurde ein Konzept erdacht, das eine automatisierte, softwareseitige Erkennung von defekten Pixeln in CCD Chips ermöglicht. Dies geschieht durch statistische Analyse der vom Detektor gelieferten Ereignisdaten. Im Zuge der Umsetzung dieses Konzepts entstanden zwei Programme, die sich problemlos in das bestehende *NRTA* Framework integrieren lassen und defekte Pixel detektieren und katalogisieren. Die korrekte Funktion beider Programme wurde anhand umfangreicher Tests mit Daten des *XMM-Newton* Satelliten überprüft und bestätigt.
Zusätzlich wurde eine Erweiterung für die *SIXT* Software erarbeitet. *SIXT* (entwickelt von C. Schmid) ermöglicht die Simulation verschiedener Röntgenoptiken und -detektoren und liefert damit z.B. wertvolle Erkenntnisse über deren Abbildungseigenschaften. Mit der Erweiterung besitzt *SIXT* nun zusätzlich die Fähigkeit, Teilchenhintergrund in die Simulation zu integrieren. Dabei werden, gemäß der zu erwartenden Teilchenstatistik, Hintergrundereignisse aus einer vorgefertigten Liste zufällig ausgewählt und an das Simulationsprogramm weitergereicht. Durch eine Testsimulation konnte bestätigt werden, daß sich die Hintergrundbibliothek nahtlos in *SIXT* einfügt und funktioniert wie erwartet.

1 Introduction

Beauty and mysteries of space have always been intriguing to man. Evidence was found that peoples like the Maya, who vanished long ago, already performed observations of the sky. Egyptian civilisations built daunting monuments such as the Great Pyramid of Giza whose alignments are evidently linked to astronomy. Today, astronomy is not only the research topic of many scientists all over the world but is performed by a huge community of amateur and hobby astronomers who all share a common commitment and enthusiasm for this field. Whereas observations of space were bound to optical wavelengths and to ground in former times, technological advance has enabled scientists to overcome these restrictions and build satellites capable of exploring the universe in a variety of other energy bands. Not till then the real potential of astronomy and especially its relevance to physics were unveiled.
This thesis will cover astronomy exploring the highly energetic X-ray band of the spectrum with particular emphasis on detector technics and simulations.

1.1 X-ray astronomy

Compared to an observation in visible light, imaging the X-ray regime, which starts at about 0.1 keV and reaches up to several tens of keV, turns out to be much more demanding. Earth's atmosphere absorbs virtually 100% of incident X-rays (see Fig. 1.1), thus providing an invaluable shielding to all living creatures on this planet as radiation with this energy has the capability to ionize atoms and therefore destroy cells. Unfortunately this makes it impossible to perform ground-based X-ray astronomy at the same time. Detectors have to be lifted above 99% of the atmosphere in order to start receiving X-ray photons, which translates into a height of more than 50 km above ground level. For this reason observations in the X-ray band just began in the 1950s when technology had gotten far enough to construct adequate detectors and vehicles to carry them to high altitudes (Friedman et al., 1951).
According to Charles & Seward (1995), the Sun was the only object in the sky X-ray astronomers were interested in for a long time. This was due to the fact that the Sun, being relatively near to Earth and thus very luminous, completely dominated the sky in this energy band. Additionally, detectors were not yet sensitive enough in order to have a realistic chance of distinguishing other sources from detector background. With the discovery of the first cosmic X-ray source Sco X-1 during a rocket experiment in 1962 (Giacconi et al., 1962), the search for additional X-ray sources began to attract far more attention and efforts. Astronomers were staggered by the luminosity of the objects they detected as many of them had to be 10 000 times brighter than the sun, given their distance from Earth. Out of the unforeseen discovery of these powerful sources scientists were able to gain valuable physical insights.
One type of objects which can be studied this way are Active Galactic Nuclei (AGN), cores of galaxies which supposedly accommodate a supermassive black hole in their centre with masses of several times $10^8 M_\odot$ or even more. Surrounding matter, attracted by the tough gravitational pull of the black hole, forms a hot accretion disk

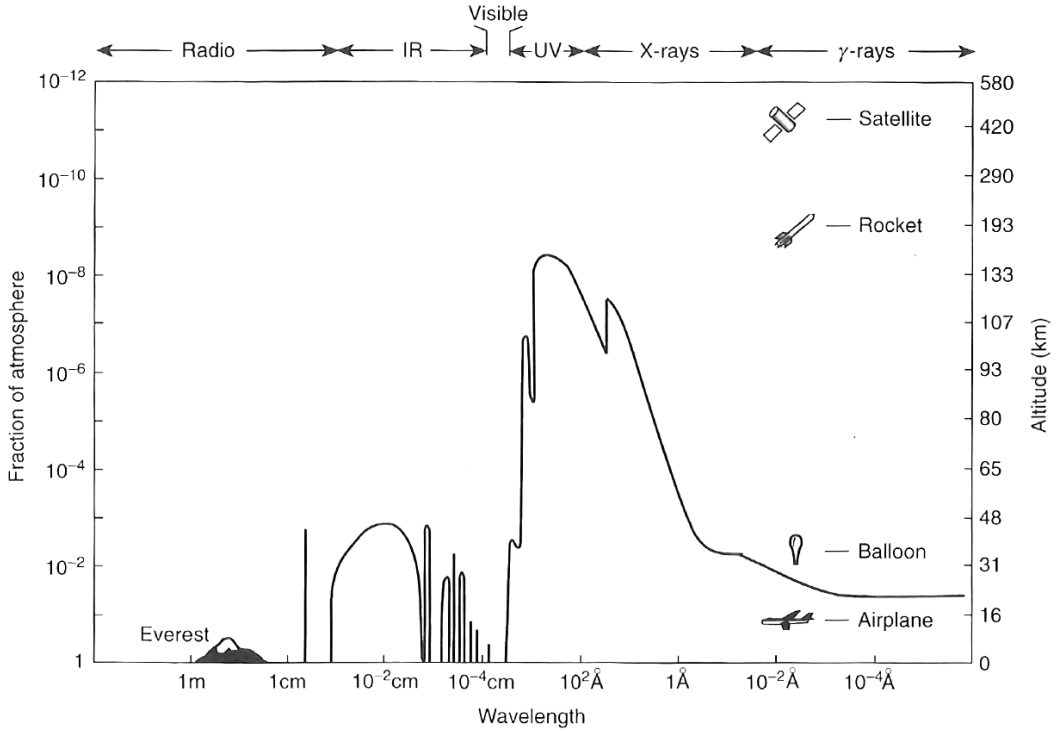


Figure 1.1: Visualisation of the wavelength dependent absorption of radiation penetrating Earth's atmosphere. The black line shows the height where half of the initial flux is left. Optical light passes the atmosphere nearly untouched while photons in the UV and X-ray band are subject to massive absorption. (Figure: Charles & Seward, 1995)

around it which then emits radiation in various wavelengths. These emissions can get powerful enough to be still detectable on Earth despite their enormous distances. For example the nearest galaxy with an active core, Cen A in the constellation Centaurus, is already 4.8 Mpc away from Earth (Müller et al., 2010). Due to mechanisms which are not yet fully understood, collimated polar streams of ejected matter can form to both sides of the black hole, so called jets. The speed of the jets often amounts to a considerable fraction of the speed of light which boosts their already powerful emission of radiation even more due to relativistic effects (Begelman et al., 1984). Since their discovery AGN have been among the most important objects for the studies of highly energetic astrophysical phenomena and without any doubt will stay important in the future.

Despite far from ejecting as much energy as AGN, binary systems were also identified as contributors to the numerous X-ray sources in the sky. They usually consist of a stellar black hole (which is several orders of magnitude less massive than a supermassive black hole in galactic cores), a neutron star or a white dwarf and a companion star, with the two of them orbiting each other (see, e.g. Schwarm (2010) and Fuerst

(2008)). Depending on the mass of the companion two different models are applied nowadays for this class of objects (Charles & Seward, 1995):

- In High Mass X-ray Binaries (HMXBs) the massive companion star, often being of spectral type O or B, emits material through powerful stellar winds. Matter falling onto the orbiting neutron star will then cause extensive emission of radiation due to the release of gravitational energy.
- The companions in Low Mass X-ray Binaries (LMXBs) lose matter because of having filled their Roche lobe. As a consequence, the "overflowing" matter can be accreted by the nearby neutron star. The resulting hot accretion disk can be seen in the X-ray band, among a broad variety of other wavelengths.

As already indicated, the observations permitting the ongoing discovery and comprehension of these phenomena make high demands on man and machine. The most important detector technologies and optics which can be used for astronomical X-ray studies will be briefly described below.

1.2 X-ray optics

The following list of optics and detectors makes no claim to be complete. It rather shows a short selection of devices with either historical importance or which are of great relevance to modern X-ray satellite missions.

Honeycomb collimator

When trying to localise X-ray sources in the sky it is mandatory to put constraints on the direction of the incoming photons which will be detected. To achieve this one can use collimators which are made up of a collection of tubes consisting of highly absorbing material (e.g., lead). The arrangement of the tubes resembles the structure of honeycombs which gave the collimators their name. The only way for a photon to get into the detector without being absorbed is to pass through one of the tubes, given that the other sides of the detector are shielded properly. Honeycomb collimators are often used in combination with proportional counters.

Wolter optics

Focusing X-rays by deflection is far more complicated than in the case of optical light. Instead of being reflected in an ordered fashion, e.g., as one would expect from light hitting a mirror, X-rays scatter or simply go through untouched when hitting surfaces under normal incidence (Wolter, 1952). In order to achieve total reflection on a mirror with refraction index n in vacuum, the critical incidence angle θ_c can be obtained with Snell's law, yielding

$$\cos\theta_\mathrm{c} = n. \tag{1.1}$$

Consequently, total reflection is only possible when $n < 1$. Considering a perfect mirror with no absorption, the refraction index n is given by (Aschenbach, 2009)

$$n = 1 - \mathrm{N_A}\frac{Z \cdot r_\mathrm{e}}{A \cdot 2\pi} \cdot \rho \cdot \lambda^2 \tag{1.2}$$

where N_A is the Avogadro number, r_e the classical electron radius, A the atomic weight, Z the atomic number, ρ the mass density and λ the wavelength of incident radiation. With wavelengths typical for X-rays ($\lambda \approx 1\,\mathrm{nm}$) and a mirror material consisting of heavy elements ($Z/A \approx 0.5$), the critical angle yields

$$\theta_\mathrm{c} \approx 5.6' \cdot \sqrt{\rho\,[\mathrm{g/cm^3}]} \tag{1.3}$$

which translates into an angle of about 1° or even less for materials common for X-ray mirrors such as, e.g., gold. For this reason, proper reflection of X-rays will only work if they hit the mirror under grazing incidence. The higher the energy of the X-ray the smaller the angle has to be. Additionally the mirrors should be composed of a heavy material with high density and Z/A ratio, as Equation 1.2 suggests (Jackson, 1999). Hans Wolter, a German physicist, invented a new X-ray optics in the 1950s which was able to take care of this problem. Interestingly, his initial purpose was to construct a device which would permit X-ray microscopy (Wolter, 1952).

Wolter optics consist of a nested structure of numerous cylindrical mirrors, each of them being separated by a slim slit only. The form of the mirrors is not perfectly cylindrical but resembles a parabolic shape migrating to a hyperbolic shape in order to focus incident photons on the detector plane. X-ray photons hitting the slits will graze the mirror surface twice, get deflected in doing so and will reach the detector afterwards (see Fig. 1.2). The resolution achieved with Wolter telescopes is clearly superior to, e.g., that of honeycomb collimators. For this reason and despite their complex and expensive manufacturing they are frequently adopted aboard X-ray satellites such as, e.g., on *XMM-Newton*, *Chandra* and *Suzaku*.

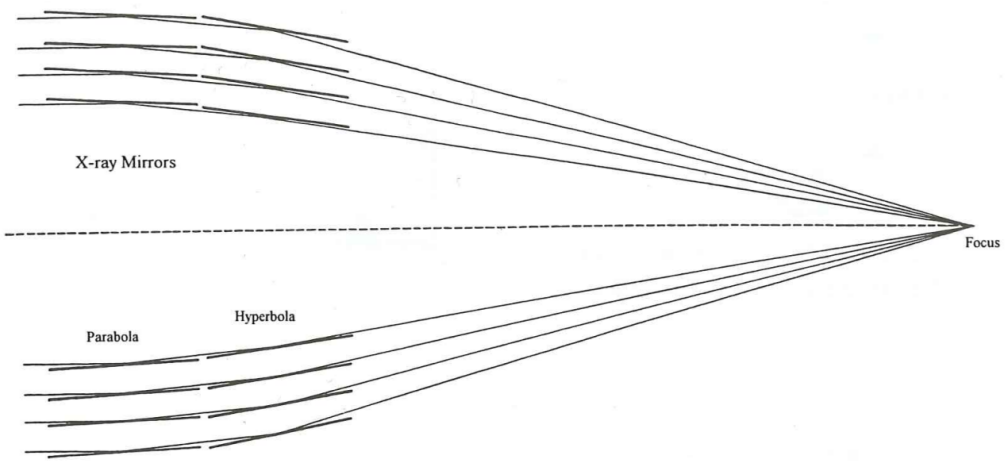


Figure 1.2: Illustration of the working principle of a Wolter telescope: incident radiation gets deflected under grazing incidence by nested paraboloid and hyperboloid shells (Figure: Charles & Seward, 1995).

1.3 X-ray detectors

Proportional counters

The concept behind proportional counters, which were invented in 1948 by S. Curran (Curran & Craggs, 1949), has much in common with their nearest relative, the Geiger counter. Basically the device consists of a gas filled chamber with a thin window to one side which can easily be penetrated by X-rays. Inside the chamber one or more energised wires are streched across. Energetic photons which enter the chamber will ionize gas molecules along their track. The number of electrons set free in this process is proportional to the initial energy of the photon. Due to the high voltage the electrons will drift towards the nearest anode wire where they can be detected as a short current pulse. Other than the Geiger counter, the anode voltage is kept low enough to prevent electrons from forming an avalanche as this would void the proportionality of photon energy and measured current. Proportional counters are still in use aboard modern X-ray satellites today such as, e.g., the Proportional Counter Array (PCA) (*Rossi X-ray Timing Explorer (RXTE)*; Jahoda et al., 1996).

Scintillation counters

For the detection of hard X-rays with energies above around 20 keV, proportional counters are rather inefficient as the photons are not completely absorbed by the detector gas. For this reason solid state detectors consisting of crystals such as sodium iodide which have the ability to fully absorb even energetic X-rays have to be used (Charles & Seward, 1995). Photons entering these so-called scintillation counters will ionize the surrounding atoms which, however, will recombine nearly instantly. The energy yielded by this recombination process will be emitted as scintillation light which can be detected by a photomultiplier. Arrival time and amount of the light pulse provide information about the initial photon (Knoll, 2000). A recent example for the use of a scintillator aboard an X-ray satellite is the Hard X-ray Detector (HXD) on *Suzaku* which covers a wide energy range of 10 keV to 600 keV (Takahashi et al., 2007).

CCDs

Thanks to the quickly emerging semiconductor industry, X-ray astronomy was soon provided with yet another useful detection device which clearly outdid the other detector technologies at least in terms of imaging. CCD chips rely on the principle of pn-junctions known from diodes and can for example be used to obtain images from X-ray sources with superior resolution. For nearly 20 years they have been favoured detectors aboard many kinds of scientific satellites and have provided astronomers with data of sparkling quality. CCDs will be discussed in greater detail in Chapter 3.

1.4 X-ray missions

Below, several chosen X-ray missions and their most remarkable achievements will shortly be presented. Most of the information in this chapter was extracted from NASA's High Energy Astrophysics Science Archive Research Center (HEASARC) website on past satellite missions (see Gibb, 2009).

Rocket and balloon experiments

The first missions dedicated to the discovery of cosmic X-ray sources started in the early 1960s while all missions of earlier years focused on the investigation of solar X-rays. These days the detectors were not launched into orbit but were attached to balloons or rockets which carried them high enough to overcome absorption of X-rays by the atmosphere. As X-rays with higher energies ($> 20\,\mathrm{keV}$) can penetrate farther, balloons equipped with scintillation detectors were sufficient for their detection. In contrast, the measurement of softer X-rays required rockets which could reach much higher altitudes. The time to perform measurements, however, was limited to a few minutes then (Charles & Seward, 1995). The start of the first X-ray satellites solved these problems for good.

UHURU

With the launch of *UHURU*[1] in Dec. 1970, the first X-ray satellite went into service. It was equipped with two proportional counters and performed a survey of the whole sky, detecting 339 X-ray sources with intensities of $2-6\,\mathrm{keV}$. In early 1973 its battery failed and the device went out of operation (Forman et al., 1978).

European X-Ray Observatory Satellite (EXOSAT)

One of the most exceptional properties of the European *EXOSAT* was its highly eccentric orbit around Earth which permitted long observations without interruption. *EXOSAT* was equipped with two Wolter type telescopes and had an energy range reaching from the very soft X-ray regime up to $50\,\mathrm{keV}$. Having been launched in May 1983, it remained functional for nearly three years and delivered valuable data, e.g., on variabilities in the spectra of X-ray binary systems and AGN.

Roentgensatellit (ROSAT)

During its first six months of operation, *ROSAT* which was launched in June 1990 performed an all-sky survey in the soft X-ray regime with a sensitivity about three orders of magnitude better than that of *UHURU*. The resulting survey catalogue contained information about more than 150 000 objects in the end. In the years after that, another 100 000 sources were investigated with deep sky pointed observations. Among other interesting things, *ROSAT* was the first to discover X-ray emission from comets. Its lifetime ended in Feb. 1999. With the European eROSITA mission currently under development, another successor to the profitable *ROSAT* could launch in near future.

[1] *UHURU* is the swahili word for "freedom"

XMM-Newton

Being equipped with three co-aligned Wolter telescopes, two of them having an array of 7 Metal Oxide Semiconductor (MOS) CCDs and the third one 12 pn-CCDs in their focal plane, *XMM-Newton* is capable of performing X-ray imaging and spectroscopy with unprecedented resolution and quality. Its energy range of $0.1 - 15\,\mathrm{keV}$ allows both the detection of soft and hard X-ray sources in the sky. It was launched in Dec. 1999 and had a nominal mission time of 10 years. The satellite is still in orbit and fully functional though.

Chandra

NASA's *Chandra X-ray Observatory*, named in honor of the famous US-American astrophysicist Subrahmanyan Chandrasekhar, was launched in July 1999. Having long since survived its nominal mission time of 5 years, it is still in operation today. Its payload consists of an iridium-coated Wolter telescope with a Field of View (FoV) of $30'$ and four devices which can be inserted into the focal plane, namely an imaging spectrometer (energy range: $0.2 - 10\,\mathrm{keV}$), a high-resolution camera ($0.1 - 10\,\mathrm{keV}$) and two transmission gratings for low ($0.08 - 6\,\mathrm{keV}$) and high ($0.5 - 10\,\mathrm{keV}$) energies. Chandra has a very high spatial resolution of $< 1''$.

Suzaku

With its enormous energy range of $0.2 - 600\,\mathrm{keV}$ the Japanese satellite *Suzaku* can serve as a versatile tool in observations of many different kinds. Its optics consists of five nested Wolter mirror shells with an angular resolution of about $2'$ and a focal length of 4.75 m. Soon after the launch of *Suzaku* in July 2005 it became clear that the X-ray Spectrometer (XRS) was inoperable due to a cooling problem (Mitsuda et al., 2007). The Hard X-ray Detector (HXD; $10 - 600\,\mathrm{keV}$) and the X-ray Imaging Spectrometer (XIS; $0.2 - 12\,\mathrm{keV}$) are fully functional though. *Suzaku* was the first satellite to be equipped with an X-ray microcalorimeter.

2 eROSITA

2.1 Overview

eROSITA is the name of an X-ray mission which is currently being planned and realized by a collaboration of German institutes under the leadership of the Max Planck Institut für Extraterrestrische Physik (MPE). The main purpose of the mission is to perform a survey of the whole sky in the X-ray band and measure the distribution of galaxy clusters. The corresponding telescope array and ART-XC, which is a Russian coded mask X-ray telescope, will be the two main assemblies aboard the Russian *SRG* satellite, which is scheduled for launch from Baikonur (Kazakhstan) using a Soyuz-2 rocket in mid 2013.
eROSITA is the successor of several different X-ray missions. Among them are the successful *ROSAT* mission which was the first to perform a survey of the whole sky with an imaging X-ray telescope, the ROSITA mission whose detector was planned to be attached to the International Space Station (ISS) but was abandoned due to the unsuitable radiation levels in this area and the *ABRIXAS* mission which had failed due to critical problems with the power supply (Greiner et al., 2001).
The optics of the eROSITA device will consist of seven identical Wolter type X-ray telescopes and are based on existing concepts of the *ABRIXAS* telescopes with additional improvements (Cappelluti et al., 2011). Other than *ABRIXAS* whose telescopes had a joint focus, the orientation of the eROSITA telescopes will be parallel. Consequently each of the seven telescopes will focus incident radiation on its own detector chip. The advantage of this design is obvious: in case of detector chip failure or damage, there will still be a large number of redundant chips left.
The chips will be pn-CCDs resembling those which have already been in use for more than 10 years aboard *XMM-Newton*. Several refinements to the CCDs have been achieved in the meantime (Cappelluti et al., 2011).
Fig. 2.1 shows the eROSITA instrument on its hexapod mounting which will be used to attach the device to the *SRG* satellite. The seven mirror modules inside the telescope housing and the cameras on the bottom are also visible.

2.2 Mission goals

Efforts in trying to understand origin, geometry and evolution of the universe are still ongoing and nowadays are taken more seriously than ever before. Around the turn of the millennium, observational data convinced many astronomers that the dynamics of the universe can be explained by the existence of so-called "Dark Energy" but nobody has managed to observe it directly yet. As Dark Energy is believed to interact on scales which are comparable with the size of the universe, the observation of large scale structures like galaxy clusters seems to be an appropriate method of studying its properties and its nature (Predehl et al., 2007).
Data from *WMAP* have shown that Dark Energy accounts for about 72% of the energy-matter density of the Universe in current cosmological models. Compared to

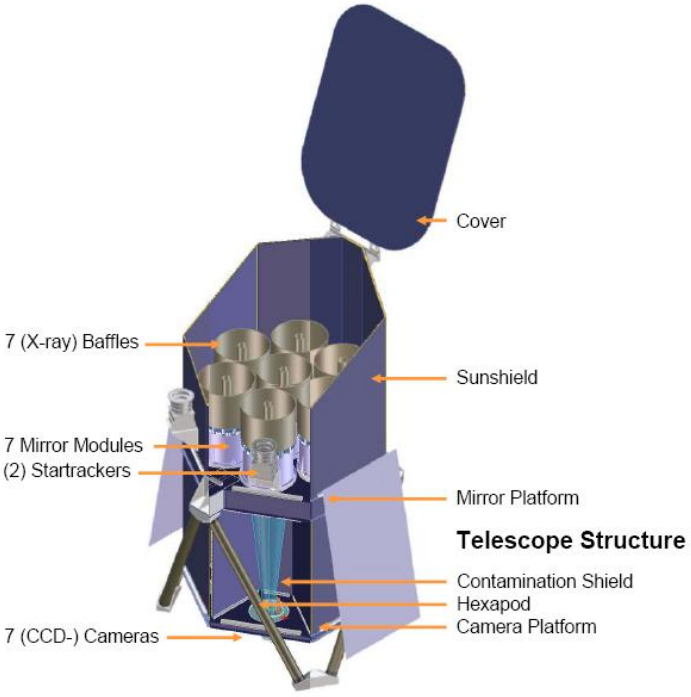


Figure 2.1: The assembled telescope in its housing (Figure: Fürmetz et al., 2008)

that the fraction of "normal" matter (consisting of atoms) with about 5% is rather small. The remaining part of 23% is occupied by Dark Matter (Komatsu et al., 2010). Due to galaxies being bright in the X-ray regime it is planned to detect $50\,000-100\,000$ galaxy clusters up to a redshift $z \sim 1.3$ by performing a survey of the whole sky in the energy band from $0.3-10\,\mathrm{keV}$ (Meidinger et al., 2009). The data gained from the survey will not only provide information about Dark Energy but will also allow tests of current cosmological models. For this reason the question whether the expansion of our universe will come to a stop some day or will go on forever could finally come to an answer (Predehl et al., 2007).
According to Cappelluti et al. (2011) detailed analysis of galaxy clusters would provide us with the following information:

- The mass distribution of galaxy clusters holds information on the matter density of the universe.
- Analyses of the growth of structures in the universe constrain the properties of Dark Matter and Dark Energy.
- Spectra of galaxy clusters depend indirectly on Dark Matter and Dark Energy.
- The large-scale distribution of clusters contains information about the curvature of space.

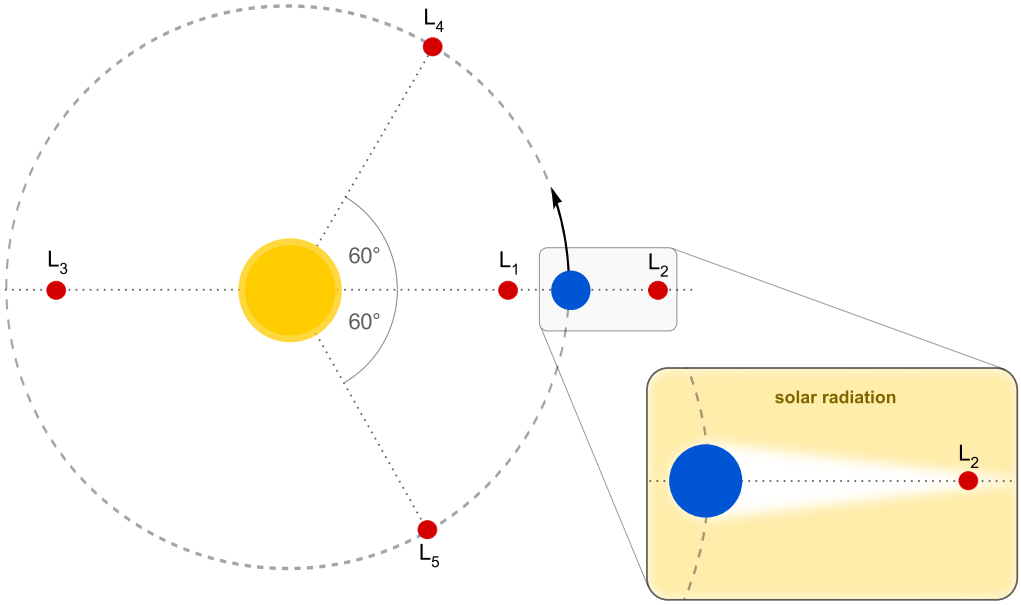


Figure 2.2: Illustration of the five Lagrangian points (red) around Sun (yellow) and Earth (blue) (Figure inspired by Karttunen, 2003). The L_2 point is partially shielded from solar radiation, as visualised in the small box. This figure is not to scale.

As soon as the *SRG* satellite with eROSITA onboard has reached its destination orbit in Earth's L_2 point and instrument calibration is done, the instrument will begin its first task and perform a survey of the whole sky with a scheduled duration of 4 years. The survey data is intended to extend the *ROSAT* imaging all-sky survey up to energies of 10 keV and will provide better spectral and angular resolution than ever before (Fürmetz et al., 2010). Data packets which have been finished will periodically be sent to the ground station on Earth. The all sky survey will be followed by pointed observations which will provide the opportunity to have a closer look at selected objects of interest with longer exposure time.

2.3 Orbit

After launch the *SRG* satellite will travel for 110 days until it reaches its destination orbit around the 2nd Lagrangian point (L_2) at a distance of 1.5 mil. km from Earth (Fürmetz et al., 2010). In the L_2 point (see Fig. 2.2) the combined gravity of Earth and Sun outweigh the centrifugal force on a lighter object orbiting the Sun with the same speed as Earth (Karttunen, 2003). Because of the co-moving Earth constantly deflecting part of the solar wind with its magnetic field, exposure to solar radiation in L_2 is lower than it would be, e.g., in a low Earth orbit. Additionally, the L_2 point is out of reach of Earth's radiation belts. The net radiation, however, is higher than in a low Earth orbit due to a lack of geomagnetic shielding from cosmic radiation. For this reason the L_2 point is a reasonable place for satellites to perform all-sky surveys

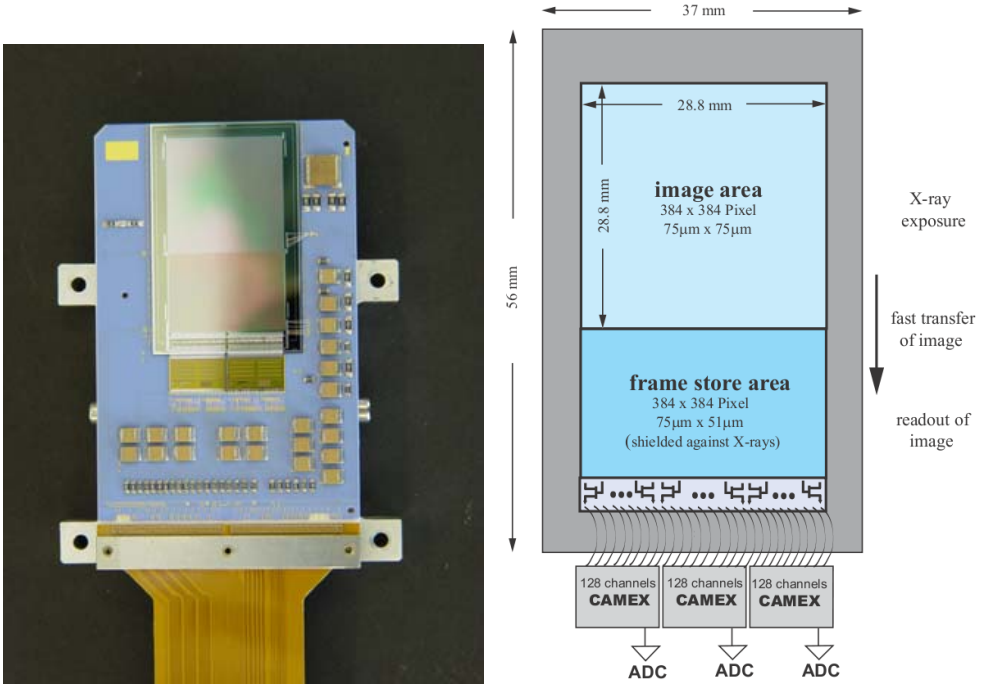


Figure 2.3: *Left image:* An eROSITA frame store CCD mounted on a blue ceramic printed circuit board. The frame store area which is insensitive to light can be seen below the quadratic exposure area (Predehl et al., 2007). As its single purpose is the temporary storage of signal charges the frame store area can be kept smaller than the light sensitive part of the chip. *Right image:* Schematics of an eROSITA CCD (Meidinger et al., 2009).

because their line of sight is mostly unobstructed.
The orbit of *SRG* will be shaped elliptically, having a semi-major axis of about 300 000 km, a semi-minor axis of 250 000 km and a period of 180 days. As the local effective potential in the L_2 point is dynamically unstable it will be necessary to perform course corrections on a regular basis. In an earlier stage of mission design it was intended to launch *SRG* into a low earth orbit. This plan, however, was changed due to technical difficulties in favour of an orbit around L_2 (Fürmetz et al., 2010).

2.4 Technical details

eROSITA will have a total weight of 735 kg with the telescope having a diameter of 1.9 m and a height of 3.2 m. The telescope consists of seven parallel Wolter optics with each of them having a focal length of 1.6 m and being equipped with a frame store pn-CCD in its focal plane. The Wolter optics of eROSITA are assembled out of 54 gold-coated, nested mirror shells with the outermost shell having a maximal diameter of 360 mm. The mirror shape, fading from a parabolic into a hyperbolic form,

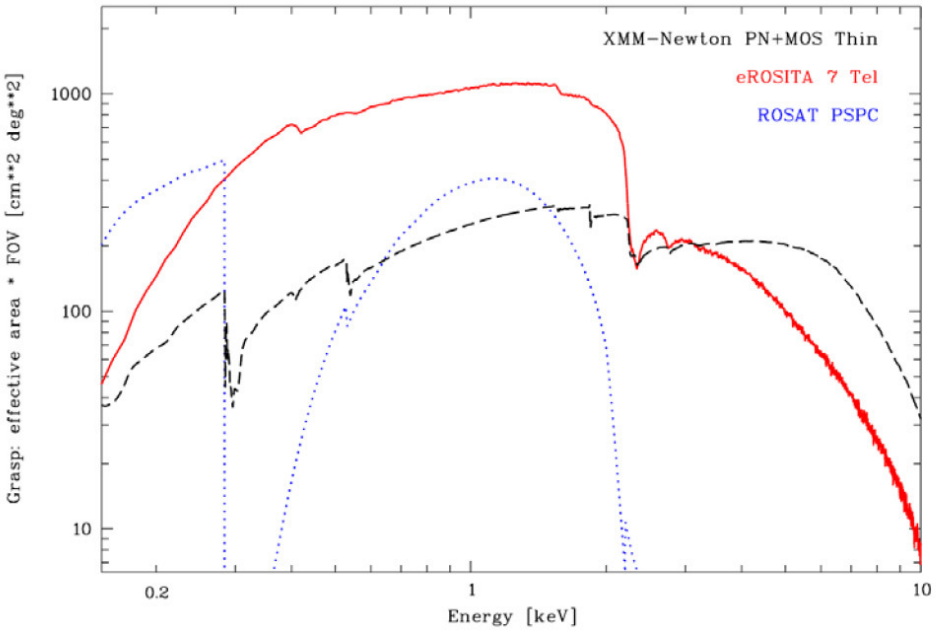


Figure 2.4: Comparison of the grasp (eff. area × FoV) of eROSITA, *XMM-Newton* and *ROSAT* (Figure: Cappelluti et al., 2011).

allows in conjunction with the nested structure the deflection of even highly energetic X-ray photons. In order to meet the tough requirements arising from the mission goal to detect 100 000 galaxy clusters, the initial design of the *ABRIXAS* telescopes was extended by adding 27 outer mirror shells. Consequently, the sensitivity at low energies increased by a factor five compared to *ABRIXAS* (Predehl et al., 2007).

Due to very limited resources being available aboard, data sorting and reduction have to take place on Earth. Consequently, raw data will be sent in a continuous, unordered telemetry stream which has to be disentangled and rearranged by specialised software tools. This process will be discussed in Chap. 4.3 in more detail.

The pn-CCDs are based on those of the *XMM-Newton* satellite and have experienced further improvements in the meantime such as the capability to store the whole frame in an area which is shielded against X-ray photons before final read out. This has the advantage that no out of time events (see Chapter 3.3) will occur even during long read out times while, in the meantime, the next frame can be exposed.

The seven pn-CCDs have a sensitive area of 28.8 mm×28.8 mm and a pixel size of 75 μm, yielding a resolution of 384 × 384 pixels. The integration time of one frame is 50 msec with an additional 100 msec needed for shifting the signal charges to the frame store area. The actual read out will take about 5 msec. In order to achieve optimal performance, the CCDs have to be cooled down to −80°C which is achieved by making use of heatpipes and radiators (Fürmetz et al., 2008). A sample of one of the CCDs is shown in Fig. 2.3.

Compared to its predecessor *ABRIXAS*, the effective area of eROSITA has been increased by a factor of five and now is of the order of $1500\,\mathrm{cm}^2$ at 1.5 keV. The FoV of about $0.83\,\mathrm{deg}^2$ and angular resolution of $28''$ (averaged over the FoV) have been improved by a factor of two respectively. Fig. 2.4 shows the energy dependency of the grasp (i.e., the effective area times the FoV) which is nearly one order of magnitude better than that of *XMM-Newton* for energies below 2 keV. This high sensitivity is important for the detection of galaxy clusters, one of the mission's main objectives (see Chap. 2.2).
Each camera is equipped with a Fe^{55} source and an aluminum target which produces two spectral lines at 5.9 keV (Mn-K_α) and 1.5 keV (Al-K_α) (Cappelluti et al., 2011). The source can be moved into and out of the FoV by making use of a mechanical filter wheel, allowing tests and recalibration of the CCDs if necessary.

3 CCD

The success story of CCDs goes way back to the early 1970s when they were first used to record images in visible light, e.g., in video cameras. Their potential for science has not been utilised before the 1990s though (except for optical CCDs which found their way into science earlier), when they were turned into much more versatile radiation detectors, which could be used for imaging and even spectroscopy (Knoll, 2000). Before the invention of CCDs, astronomers had to rely on photographic plates or films in order to take images of the sky. Their usage, however, was not as comfortable due to their smaller sensitivity (i.e. longer exposure times were needed), complicated development, and often much worse quality of the final images.

3.1 Basics

3.1.1 Semiconductors

Understanding the principle behind CCDs requires at least basic knowledge about semiconductors. In insulators, valence electrons have virtually no chance to get enough energy to overcome the band gap. The case looks different for semiconductors, whose band gap is small enough to permit such processes. As soon as a valence electron has gained sufficient energy (e.g., by thermal excitation or incident radiation) it will switch to the conduction band and can move freely through the material.
By injecting atoms into the semiconductor, whose number of valence electrons differs from that of the native material, one can reduce the band gap even further. This process is called “doping”. If the dopant has more valence electrons than the base material (“n-doping”), it is especially easy to lift the excess electrons to the conduction band as they have no binding partner in the crystal lattice. Accordingly, in “p-doping” there are less valence electrons, thus leaving a positively charged hole in the material. If this hole gets occupied by an electron of the conduction band, a positive excess is left behind, which can contribute to charge transport as well.

3.1.2 Diodes and CCDs

If n-doped materials are brought into physical contact with p-doped materials, a pn-junction (also called “diode”) is created. Consequently, excess electrons of the n-material recombine with the holes of the other. This recombination creates an opposing potential difference between the two substances which sooner or later compensates the attractive force between electrons and holes and hence will lead to an equilibrium. The area around the contact layer of the two materials, which does not have charge carriers anymore, which could easily be freed, is called “depletion zone”. By application of a bias voltage to the diode, the size of the depletion zone and thus its ability to conduct (“forward direction”) or block (“reverse direction”) current can be varied.

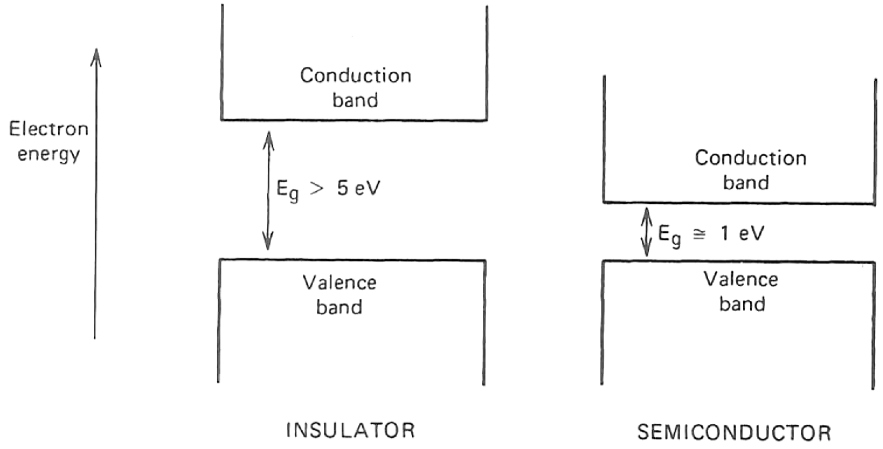


Figure 3.1: Comparison of the band gap size of insulators and semiconductors (Figure: Knoll, 2000).

Although CCDs come in different types (pn-CCDs, MOS CCDs) and lots of different shapes (linear, matrix, cylindrical...), the physical principle they are all based on is similar. Basically, CCDs resemble a collection of reversely biased diodes. As a consequence, depending on the strength of the bias voltage, CCDs have a more-or-less broad depletion zone. Photons which travel through the depletion zone have an energy-dependent chance to lift valence electrons to the conduction band. When this happens, the freed electrons drift, due to the electric field spanning across the depletion zone, to the top or the bottom of the CCD. It is, however, necessary to move them in a controlled manner towards the readout electronics, which is located at one side of the CCD chip and which is used to convert the electron current into a digital signal. To achieve this, each pixel of the CCD is equipped with wiring, making it possible to alter the local potential and thus create wells for the electrons. Additionally, if these potentials are triggered in such a manner, that the electrons in the wells are "handed over" from pixel to pixel (just like in a bucket chain), they can be transported towards the readout node bit by bit. This shifting can be achieved by varying the depth of adjacent potential wells periodically. In Fig. 3.5 three potentials $\varphi_1, \varphi_2, \varphi_3$ with a small time-shift in between them illustrate this method.

3.2 Benefits

With about 50 years of experience in the manufacturing of semiconductor devices (Charles & Seward, 1995), the fabrication of CCDs with ever increasing complexity and superior quality is technically quite feasible nowadays, while at the same time keeping the costs in affordable regions.
One of a CCD's most obvious advantages, when compared to former imaging methods, are its ease of use and its versatility. Due to their highly flexible exposure times,

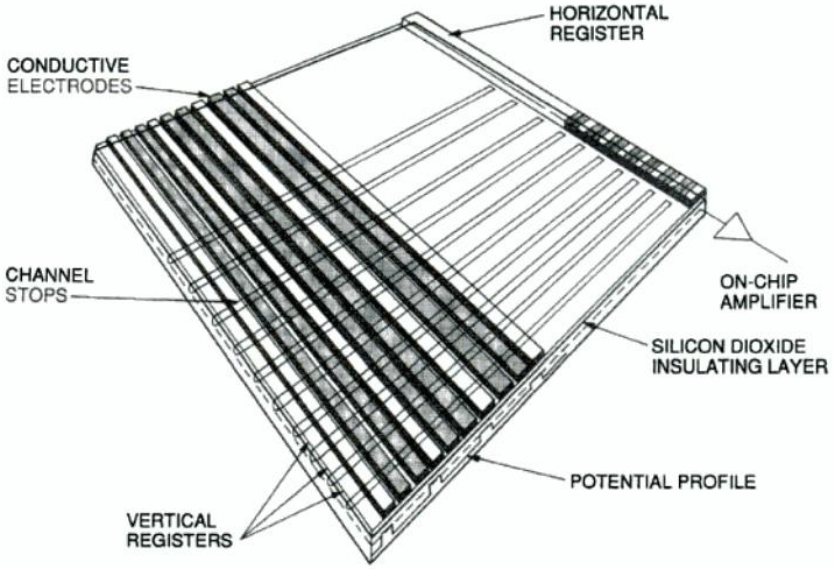


Figure 3.2: A very common shape of CCDs is the matrix structure: vertical strips of diodes, separated by insulating SiO_2 channel stops are aligned next to each other. This structure is covered by perpendicular electrode strips, which apply the readout potentials to the channels and thus "guide" the signal electrons sidewards to the horizontal readout register (Figure: Janesick, 2001)

CCDs are applicable in studies of highly time-dependent phenomena (e.g., oscillating neutron stars) in addition to their capability of recording images by integrating photons over longer periods. In X-ray astronomy, CCDs are the best and most convenient way to perform imaging because of their good resolution, compared to, e.g., scintillators. Furthermore, due to being nearly maintenance-free, lightweight and rather small, CCDs have quickly become an integral part of many devices, such as consumer digital cameras, astronomical imaging cameras and even scientific satellites. Unlike, e.g., photographic plates or the human eye, the sensitivity of CCDs depends linearly on the amount of incident radiation and thus on exposure time. This behaviour is useful for comparing the brightnesses of different objects which may have even been recorded with different (yet known) exposure times.

3.3 Disadvantages

3.3.1 Blooming

As the potential wells which are used to gather the photo-electrons are not infinitely deep and therefore can hold only a limited amount of charge, unwanted effects will occur when exposure time is chosen too large. When exposure time is chosen too long, the potential wells will no longer be able to ingest additional electrons at a

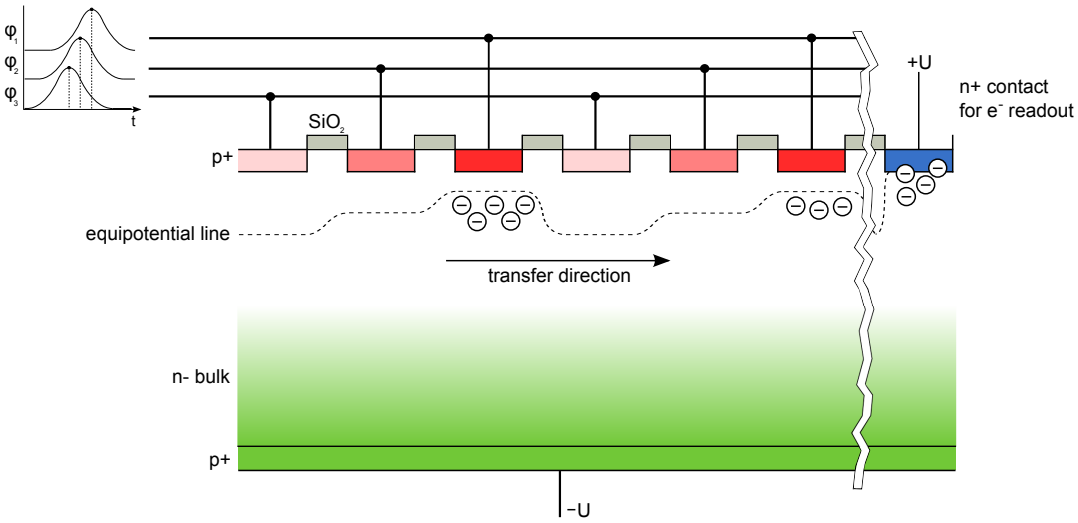


Figure 3.3: Cross-section of a CCD: regions marked $p+$ show highly positive doping while the n-bulk experienced much weaker negative doping and thus is nearly fully depleted of free charge. As indicated, signal electrons gather in potential wells, which are created by application of bias voltages to the pixels. The electrons can be shifted towards the readout node (marked in blue), if the potentials φ_1, φ_2 and φ_3 vary with time in an appropriate way (Figure inspired by Knoll, 2000)

certain point and start to flow over into neighbouring wells. This causes drop-like artifacts ("blooming") in the image which will render all data taken in this region useless. Overexposed pixels have to be avoided in any case though, as they always go along with data loss.

3.3.2 Out of time events

Due to their finite readout time, CCDs have a short time span during which they are not able to interpret the incoming photons correctly. As illustrated in Fig. 3.5, signal electrons are moved towards the readout node. Depending on the chip's architecture, the time needed for one full readout varies, but typically lies in the order of a few tens of milliseconds for fast X-ray CCDs or even seconds in case of slow ones. Photons, however, which hit the chip during this time, will free additional charges and create an unwanted excess signal. Unfortunately these events cannot be distinguished from the real ones and thus their origin will not be reconstructed correctly. In X-ray astronomy such incidents are called out of time events (optical astronomers refer to this phenomenon as "smearing") and have to be avoided or at least minimised. Many optical CCDs are equipped with a shutter in front of the chip, which shields it from further exposure while the readout is in progress. As photons can no longer hit the detector during this time, the usage of shutters reduces smearing to nearly zero. In case of X-ray CCDs it is often possible to speed up the readout process by changing

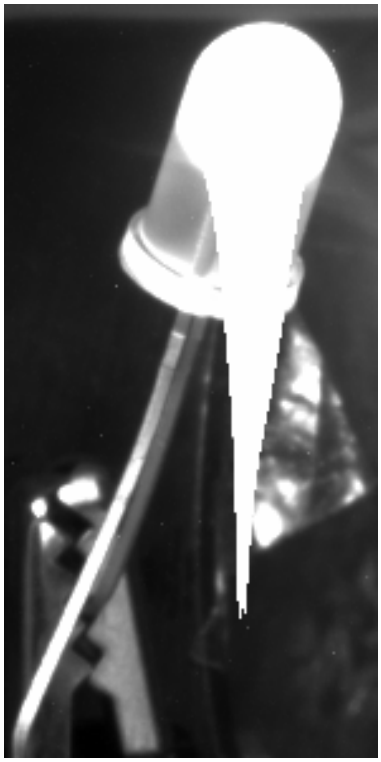


Figure 3.4: An example for blooming in a picture of a LED. Exposure time was 20 sec (Picture was taken with a SBIG ST-9XE camera which contains an optical CCD).

the imaging mode (e.g., one can constrain the readout region to only a fraction of the chip) which helps reducing out of time events.

3.3.3 Split events

Normally, in readout direction the pixels of a CCD are not separated from each other by any kind of insulating material. And even the "channel-stop" which is a thin, insulating layer between columns is not sufficient to avoid so-called split events. These occur when an incoming photon hits the chip near the boundary of one or more pixels. The cloud of charge carriers which are set free by the photon may therefore reach over several adjacent pixels and hence will make correct event and energy reconstruction much more difficult. This is a problem especially in spectroscopy.

3.3.4 Pile-up

When doing spectroscopy with CCDs, it is mandatory to be able to reconstruct the exact energy of the photons which hit the detector. To achieve that, incident photons have to be distinguishable from each other, i.e. there must not be more than one photon per readout cycle in a certain area on the chip. If the observed source is

Figure 3.5: An example for smearing in a picture of a LED. Exposure time was 0.12 sec with the shutter being deactivated (Picture was taken with a SBIG ST-9XE camera which contains an optical CCD).

too bright, too many photons will arrive during one time frame and hence energies of different photons, which just have hit the same pixel, will pile up. Thinking of split events, it is not sufficient to ask for one photon per pixel, as it is not possible to tell apart, e.g., two adjacent photon hits from the hit of one energetic photon, whose energy was spread among these pixels. In order to minimise these effects, the time needed for one readout cycle has to be kept as short as possible. For example, so-called interline CCDs have an extra readout column adjacent to every normal column, which speeds up the readout process significantly. These readout columns are shielded and are therefore not exposed to incident radiation.

3.3.5 Dark current

A phenomenon unfortunately every CCD suffers from is unwanted excitation of electrons by thermal energy. At any temperature $T > 0$ there is a non-vanishing probability $p(T)$ for valence electrons to get sufficient energy to overcome the band gap and move from the valence to the conduction band, thus leaving a hole behind. According to Knoll (2000), the probability to generate an electron-hole pair by thermal excitation is given by

$$p\left(T\right) = CT^{3/2}\exp\left(-\frac{E_g}{2kT}\right) \tag{3.1}$$

with T the absolute temperature, E_g the band-gap energy, k the Boltzmann constant and C a proportionality constant which is characteristic for the material.
Therefore, the strength of the dark current mainly depends on the ratio of band-gap energy E_g to temperature T.
As the presence of dark current adds noise to the output signal which reduces data quality, it is of great interest to keep this effect as low as possible. For this reason scientific CCDs are kept at fairly low temperatures, e.g., $-83°$C in case of the Wide Field and Planetary Camera (WFPC) aboard the *Hubble Space Telescope (HST)* (Hill et al., 1997). Additionally, one can record so called dark frames, i.e., images without exposing the CCD to incident radiation. Consequently the resulting data will just contain the signal of the dark current (and to a small fraction noise of the readout node). By subtracting correctly scaled dark frames from the real data it is possible to reduce dark current influence to a minimum.
In Eq. 3.1 we have seen that the size of the band gap plays a significant role for the probability of electrons to get thermally excited to the conduction band. While all pixels of a CCD show a certain amount of dark current, there are always a few pixels where it is exceptionally high, the so-called "hot" pixels. These and their counterparts, the "cold" pixels, will be subject to detailed discussions in the following chapters.

3.3.6 Particle background

An additional contributor to noise and thus to a reduction of signal quality are background events in the CCD chip caused by incident charged particles. These particles may originate from different sources such as, e.g., solar winds, outer space or secondary radiation.
Minimum-Ionizing Particles (MIPS), mainly protons with energies around 1 GeV (Danner, 1993), will leave trails of ionized electrons along their track when penetrating the detector. Due to these characteristic streaks, detector events caused by MIPS can in principle be discriminated from X-ray events. Their energy deposit can also give a clue if it is higher than the energy of X-ray photons.
According to Danner (1993), it is possible for low-energy electrons to be reflected by the telescope mirrors onto the detector plane. If they additionally happen to have an energy in the sensitive range of the CCD they can hardly be distinguished from X-ray events. A similar problem arises from photons and charged particles which originate as secondary radiation from interactions of primary energetic particles with matter such as, e.g., bremsstrahlung emitted by electrons penetrating the satellite material (Tenzer et al., 2010). Secondary radiation, however, can be minimized by equipping the detector with proper shielding.

4 Bad pixel recognition

4.1 Bad Pixels

Basically, bad (or dead) pixels tend to show an anomalous signal compared to other pixels. This behaviour is caused by local defects in the crystal structure of CCDs. As the data which are produced by these defective pixels would have non-negligible influence on the products (e.g., light-curves, spectra...) of the whole dataset, it is of great interest to both minimise the number of crystal defects and recognise and flag existing ones.

4.1.1 Origin and classification

As already mentioned in Chap. 3.3, dark current adds additional charge carriers to the signal due to thermal excitation of electrons. If the crystal has suffered local damage, e.g., by highly energetic radiation the dark current signal may get even higher. This is due to the pixel's band gap having reduced further, making it easier for electrons to get excited to the conduction band by thermal energy. Pixels which show this behaviour can get significantly brighter than others or may even get completely saturated, i.e. no further charge carriers can be set free during that cycle. These pixels are called "hot" or "bright" pixels.
To the contrary, if crystal defects give rise to an enlargement of the band gap much less or even no electrons at all will be set free, leading to a signal near zero. This may also happen if the insulating structure of the pixel (often a SiO_2 layer) is damaged and produces a short-cut which consequently will drain all free charge carriers from this pixel (and possibly of all subsequent pixels in this channel) before they can get read out. This type of defect is called a "dead" or "cold" pixel.
The effects of both types of defects not necessarily have to be constrained to the extent of just a single pixel. CCD chips have a readout direction to which the collected charge carriers are shifted step by step (see Chap. 3.1). Now, one can imagine that charge gets shifted across a bad pixel during the readout cycle and hence charge will possibly be added to or removed from the initial signal, according to the nature of the respective defect. As a result all pixels of this line whose charges have to cross the bad pixel in order to reach the readout node will show an anomalous signal. This effect is called bright or cold line respectively. Note, however, that still only one pixel is actually broken, in fact rendering the valid signal of many other pixels useless.

4.1.2 Impact on data quality

Bad pixels voiding the data of a whole line, of course, belong to the worst cases that may happen. But even isolated defects which do not tamper with the data being shifted across, can be very irritating in data reduction. This is especially the case when the pixel happens to obscure parts of the object of interest such as, e.g., a galaxy.

If not recognized, the additional or missing charge carriers of the pixel (depending on its type) will be interpreted as photons coming from the source with corresponding energies. Hence, spectra or lightcurves of these data will be shifted or show an offset accordingly which can have significant impact on scientific results.
Another unpleasant side-effect particularly in terms of hot pixels is an increase of data rate. The unceasing excess signals hot pixels produce, bloat data volume and consequently the storage space necessary. This is a big problem especially aboard satellites where only very limited technical infrastructure and telemetry bandwidth are available. For this reason, it is common practice to instruct the satellite software to deactivate or mask out pixels which have shown unwanted behaviour over a longer period.

4.1.3 Annealing

Pixels which have been rendered useless due to local damage in the crystal are not necessarily bound to stay in this condition forever. Observations have shown that the total number of bad pixels on a CCD can decrease again, given that the cause of their formation (e.g., irradiation with energetic rays) has vanished (Lutz, 1999). This effect, called "annealing", shows a strong dependence on temperature, i.e. the warmer the CCD gets, the more crystal defects are "repaired".
According to Lutz (1999), annealing is caused by the fact that defect complexes lose their stability in the crystal lattice when temperature reaches a critical level. They transform into other types of defects with different characteristics, which perhaps do not influence their contribution in charge collection and transport anymore. As a consequence the pixel can appear to be fully functional again. Unfortunately annealing does not work arbitrarily well, as it will reach a saturation level sooner or later. After that a further decrease in bad pixel numbers cannot be expected.
The physics behind annealing is way too complex to describe it in just a few words and many processes being involved are not thoroughly understood yet (Lutz, 1999). In the course of this thesis the explanations above shall suffice though.

4.2 Objectives

One of the most important things when doing research with "real" data (i.e. data obtained with any kind of measuring device or detector) is to minimise systematic errors as they would rise uncertainties of the scientific outcome needlessly. In Chap. 3.3, the demand for methods to keep the total number of bad pixels as low as possible was already mentioned. Unfortunately it is impossible to avoid the occurrence of bad pixels completely. As a consequence existing bad pixels have to be identified and their type and position stored in a database. Afterwards one can remove the faulty data which was produced by these pixels from the dataset.
Independent of the method which is applied to detect bad pixels, several requirements have to be considered:

- Detection accuracy has to be as high as possible, i.e. the probability of false detections must be minimised. Otherwise data are thrown away which would have been actually valid.
- Algorithm speed should be kept as high as possible. Event files can grow very large which would lead to unnecessarily long processing times otherwise.
- Computer resources (main memory, disk space) should be used economically
- Values and settings which are needed by the algorithm for correct operation have to be parametrised, i.e. they must be adjustable at program start and/or through a configuration file.
- The recognition software has to be designed such that it can be re-used for different detector sizes and geometries.

Of course good detection accuracy should be most important to achieve reliable results and may therefore outweigh other points to a certain extent.
The main focus of this thesis was to develop and implement software which is capable of processing event lists obtained from satellite data and testing them for bad pixels. If found any, their position and further information (like type or mean signal) should be archived for later use during data reduction. Two bad pixel detection tools were developed out of these requirements, *FPFindHotPixel* and *FPFindColdPixel*. They are written in ANSI C, make use of the *CFITSIO* library and have become part of the *FITS-pipeline (FPIPE)* which will be incorporated to the eROSITA *NRTA* in turn. Both programs as well as the libraries will be explained in the following chapters.

4.3 *NRTA* Software

The eROSITA *Near Real Time Analysis (NRTA)* is a software pipeline, which is currently developed by the Dr. Remeis Sternwarte Bamberg / Erlangen Centre for Astroparticle Physics (ECAP) in close collaboration with MPE. Its purposes are monitoring satellite health and performing first quick-look analyses of satellite data (Wilms et al., 2009). The insights obtained by this initial appraisal of the dataset can also be useful in the "real" scientific analysis which will be performed by the *Standard Analysis Software System (SASS)* afterwards[2].
Fig. 4.1 illustrates the steps data will go through after having been transmitted from the satellite in a telemetry stream to the ground station. The continuous telemetry data are split up to three different kinds of files, depending on their content:

- Housekeeping (HK) information about the current satellite state (e.g., voltages, temperatures, diagnostic maps...)
- Scientific Data which have been recorded recently by the telescope

[2]The *SASS* is developed by the MPE and will be used for final data reduction

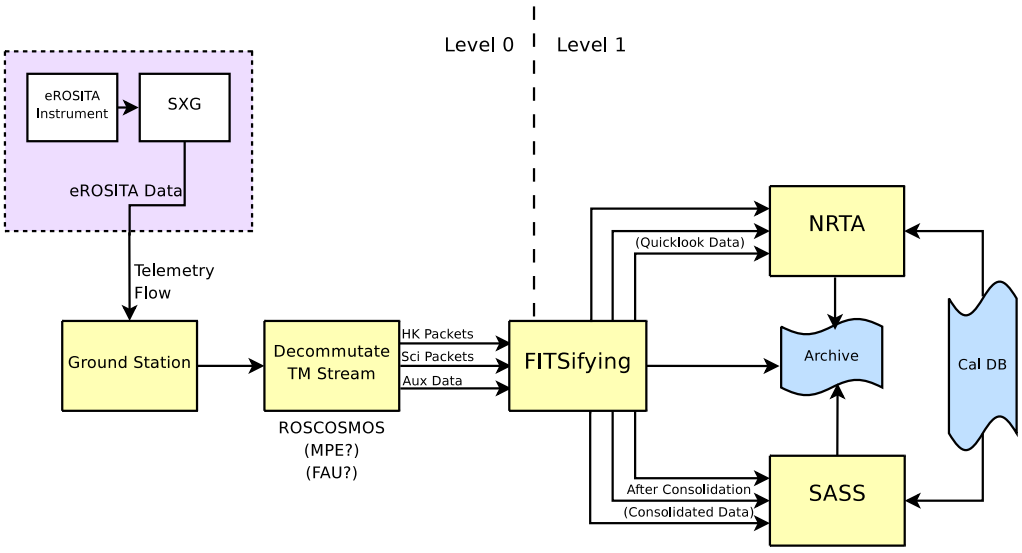


Figure 4.1: Illustration of the different stages of data processing, starting with satellite telemetry data being received by the ground station and ending with *NRTA* and *SASS*, which save their results to an archive (Figure: Wilms et al., 2009).

- Auxiliary (Aux) information needed for correct operation of *NRTA* and *SASS* (e.g., satellite attitude, mission planning files, orbit data...)

The disentangled data are stored in the Flexible Image Transport System (FITS) format which was developed in the late 1970s and has become very popular especially among astronomers due to its widespread support and multifunctionality (Pence et al., 2010). Afterwards the dataset – be it already completed or not – is fed into the *NRTA* pipeline with the results being saved to an archive.
The *NRTA* is designed in a modular way such that great flexibility is achieved. It consists of various small modules with each module focusing on one particular task (e.g., Good time interval (GTI) determination, bad pixel detection, data plotting etc.). Depending on the current analysis, modules can be combined in different ways or even started in parallel with subsequent modules working on output data of preceding modules. Fig. 4.2 shows an example of a spectrum which was displayed and updated in real-time during data reduction. To achieve this, satellite event files were fed piecewise into the first pipeline task and underwent further processing by several consecutive tools with one of them showing the graphical output. This can be very useful for having a first quick-look on the data in order to check their quality and guarantee pipeline sanity.

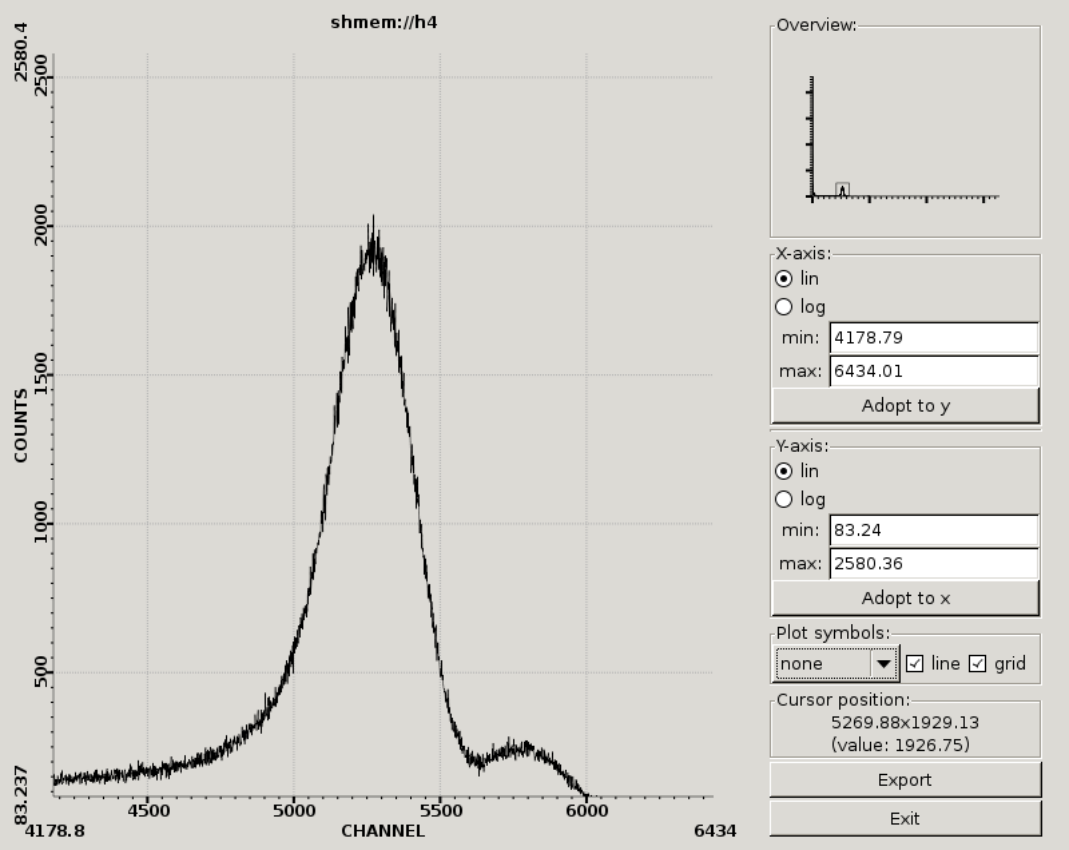


Figure 4.2: A source spectrum plot in *FPPlot2*. Despite its data source having real-time nature, the user has several possibilities to adjust data visualisation on the fly.

4.4 *CFITSIO*

CFITSIO is a free software library written in ANSI C and was developed by the HEASARC at NASA Goddard Space Flight Center (GSFC). Its functionality provides programmers with a convenient way to create, manipulate and delete files which follow the FITS standard (Pence, 2009). The *CFITSIO* is the successor to the *FITSIO* which is based on FORTRAN and was developed in 1992 with the goal to "insulate the programmer from having to deal with the complicated formatting details in the FITS file" (Pence, 1992). Since v2.0 other institutes such as, e.g., the *INTEGRAL* Science Data Center (ISDC) have started making contributions to the software. Source code, documentation, tutorials and further information on the CFITSIO library can be obtained from NASA's HEASARC website[3].

[3] *http://heasarc.nasa.gov/docs/software/fitsio/fitsio.html* (Mar. 2011)

4.5 *FPIPE*

In Chap. 4.3 several advantageous properties of the *NRTA* software, such as its modular structure and its flexibility, were mentioned. This functionality is achieved by making use of the *FPIPE* which was developed by S. Schwarzburg from the Institut für Astronomie und Astrophysik Tübingen (IAAT) in the context of his diploma thesis. According to Schwarzburg (2005), the *FPIPE* was developed from scratch with the goal to create a software framework which would prove to be useful for real-time data analysis in future satellite missions. Big efforts were made to keep Central Processing Unit (CPU) load at a minimum in order to realise the intended real-time capabilities while at the same time keeping great functionality and adjustability.
The resulting *FPIPE* software not only offers the possibility to construct pipelines consisting of a chain of arbitrary tools (may they be self-provided or existing, external binaries) but already comes with a whole bunch of programs which are very useful for data extraction and visualisation. For example, the functionality to create spectra, lightcurves and intensity diagrams out of event lists is – among others – already provided. Additionally it is possible for each tool to continue processing data one or more of its predecessors have produced as output. As this could lead to race conditions causing a dead lock, the *FPIPE* is equipped with a semaphore handler which can account for this problem automatically.
Individual pipelines can be specified with so-called pipeline definition files being written in eXtensible Markup Language (XML). An example for a simple pipeline which would generate and display a lightcurve is illustrated in Fig. 4.3. All information provided in the pipeline definition must be enclosed by `<pipelinedefinition>` tags. Further division into sections which logically belong together such as, e.g., a tool which generates a lightcurve and the corresponding plotting program, can be done with arbitrarily named, additional tags. For each component of the pipeline, the corresponding program call with all necessary parameters has to be specified, as well as input and output files. For further details about the different components and capabilities of pipeline definition files see Schwarzburg (2005).
As the *FPIPE* is used by the *NRTA* software, eROSITA will be the first mission where the *FPIPE* can demonstrate its abilities.

4.6 Detection principle

At first glance bad pixel recognition seems to be a pretty easy task. The only thing which has to be done is to analyse the data and mark all pixels which show odd values (i.e. no signal at all or unreasonably high) compared to the other pixels near them. But that is the point where problems already begin: when exactly does a signal start to look "odd"? In the case of cold pixels, the answer is pretty easy: the pixel value has to be equal to 0 (this introduces other problems though, which will be discussed in Chap. 4.8). In constrast, hot pixels produce a more-or-less bright fake signal which can not necessarily be detected that easily. Depending on the Point Spread Function (PSF) of satellite optics, size and shape of a point-like source will

```
<pipelinedefinition>
  <GetEvents>
    <Events>
      <status>0</status>
      <fpipeserverstart>FPFitsEventFeeder frametime=0.001 geiger=n outfile=##OUTFILE[0]##
      infile=##INFILE[0]## semid=##SEMID## wlocks=##WLOCKS,/## or=##O_R,/##</fpipeserverstart>
      <infile sema="false" source="true">Pix.fits</infile>
      <outfile sema="true">PixList.fits</outfile>
    </Events>
  </GetEvents>

  <ProcessEvents>
    <Lightcurve>
      <status>0</status>
      <loopstart>FPLightcurve infile="##INFILE[0]##" outfile="##OUTFILE[0]##" servermode=yes inmode=5
      intcol=FRAME outncol=COUNTS outtcol=TIME binsize=1</loopstart>
      <loopstop>xpaset -p FPLightcurve EXIT</loopstop>
      <single>xpaset -p FPLightcurve PROCESS_DATA</single>
      <infile sema="true" source="false">PixList.fits</infile>
      <outfile sema="true">shmem://h2</outfile>
    </Lightcurve>

    <Plot2>
      <status>0</status>
      <loopstart>FPPlot2 infile=##INFILE[0]## servermode=y socket_id=##SOCKET_ID## xcolumn=TIME
      ycolumn=COUNTS xerrcolumn="" yerrcolumn="" psym=1</loopstart>
      <loopstop>xpaset -p FPPlot2 EXIT</loopstop>
      <single>xpaset -p FPPlot2 PROCESS_DATA</single>
      <infile sema="true" source="false">shmem://h2</infile>
      <outfile sema="false"></outfile>
    </Plot2>
  </ProcessEvents>
</pipelinedefinition>
```

Figure 4.3: An example for a pipeline definition file which would read in the event list *Pix.fits*, generate a lightcurve out of it and make it accessible through shared memory, allowing the plotting tool *FPPlot2* to display the results. The plot would look similar to the sample shown in Fig. 4.2.

vary significantly in the final image. Hence, the challenge to distinguish bad pixels and sources can get very demanding.

Another aspect which has to be considered is the format of the input data. The recognition of bad pixels being contained in a static, single image which already contains all available information will work different from an algorithm which operates on event lists. This is due to the fact that event lists additionally involve information about the time when the events occurred. Under certain conditions this can be very helpful for bad pixel search. When checking, e.g., data of an all-sky survey, real sources will move from one side of the FoV to the other because of the satellite having scanned the sky bit by bit. Due to their immobile nature bad pixels can be easily detected in this case.

The data which are processed by *NRTA* and *SASS* are formatted as time- or frame-dependent event lists. For this reason, both bad pixel detection tools have been optimized to work with this kind of data and do not support static images. As the methods for the detection of bright and cold pixels differ nearly completely from each other, it was reasonable to split the software in two separate parts.

Bad Pixels

According to Chap. 4.1 bad pixels are restricted to the dimension of one single pixel. Hence, it would be possible to distinguish hot pixels from real sources if the PSF were bigger than one pixel (which is normally the case) and had a sufficiently smooth slope on either side. In order to account for different PSF shapes, detector geometries and other effects, which could have impact on data quality and therefore on hot pixel detection, there is need for an adjustable threshold, which varies the sensitivity of the pixel-to-neighbour test.
Thinking about the comparison of pixels with their neighbours another problem arises. Is it sufficient to look at pixels which are direct neighbours to those in question? Certainly not if very bright sources are observed, whose core spans across more than one or two pixels and thus boost their signal to nearly equal values. Shape and size of the PSF are also important again. Just like before, it seems reasonable to introduce another recognition parameter which makes it possible to adjust the size of an area whose pixels' values should be compared to each other.

Bad Lines

In principle, recognising a bright line works similar to the detection of hot pixels: by comparing a line with its neighbours it should be possible to sort out those which behave strangely. A bright line, however, may not necessarily be bright over its whole extent but, e.g., to just one third. A possible solution is to cut the line to imaginary pieces and compare each piece with the corresponding piece of neighbouring lines - in analogy to the approach in terms of hot pixels.

4.7 *FPFindHotPixel*

Compared to cold pixel detection the search for bright (or hot) pixels proved to be the more difficult task. This is, e.g., due to the fact that one has to distinguish them from real sources which sometimes are not much bigger than a few pixels. Additionally there are different types of bright pixels (see Chap. 4.1) which do not behave equally. Thus, a lot more time and efforts had to be spent on the development and the testing of the program *FPFindHotPixel* whose first duty will be to find and catalogue bright pixels in the eROSITA CCDs by analysing event lists.

4.7.1 Overview

FPFindHotPixel consists of two files: the executable binary file itself and a file called *FPFindHotPixel.par* ("the PAR-file"), serving as a configuration file for a lot of parameters which are needed during runtime.
Fig. 4.4 illustrates the three most fundamental steps the program is built on: an event list (which is sorted with respect to frame number or time and contains background, bad pixels, bad lines, sources etc.) is fed in some way into the recognition algorithm. Eventually, it produces two output files in the form of a table and a map which contain tabular and visual information about the recognised bright pixels and bright lines.

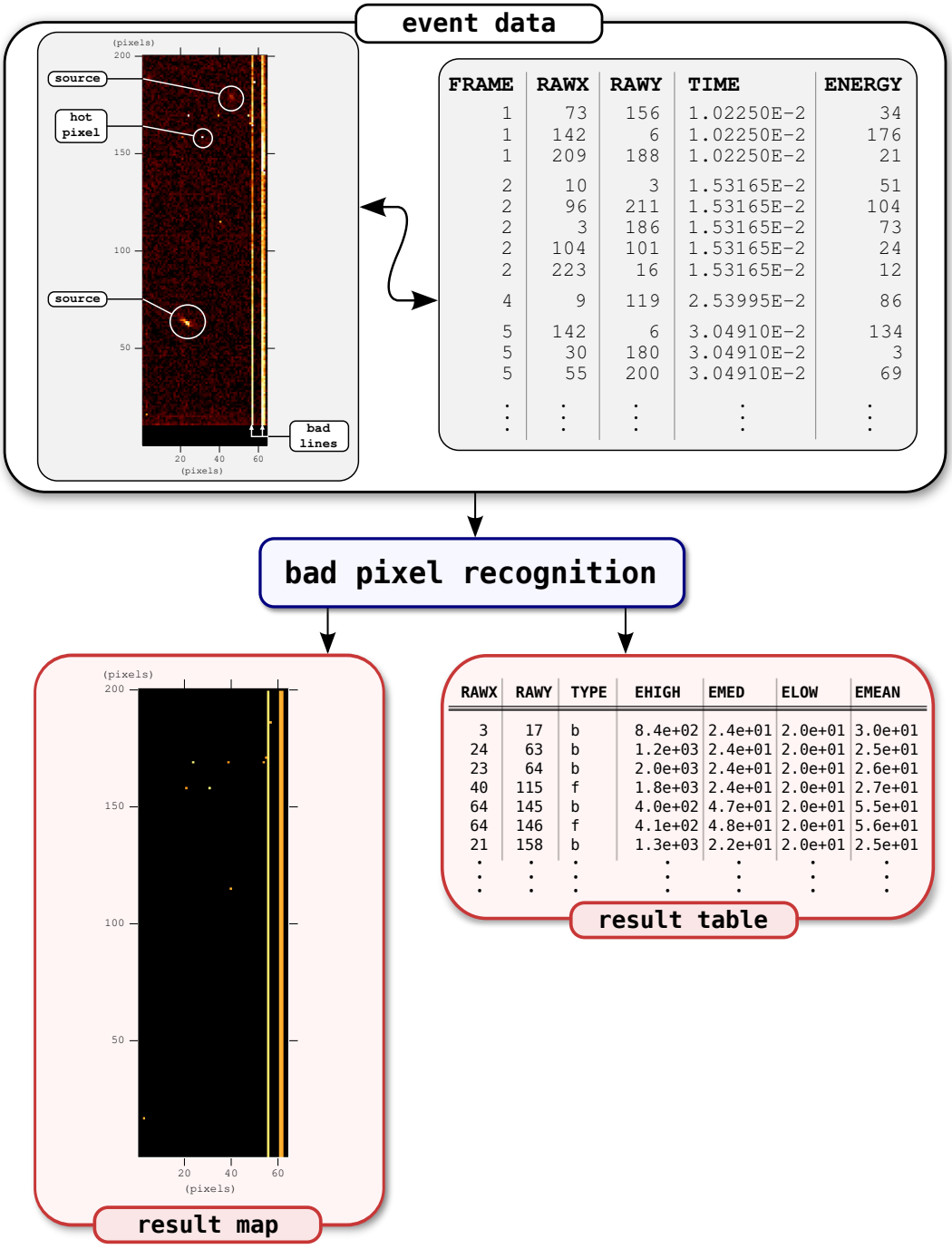


Figure 4.4: General overview: input event data is fed into bad pixel recognition which, in turn, produces a bad pixel map and a summary table.

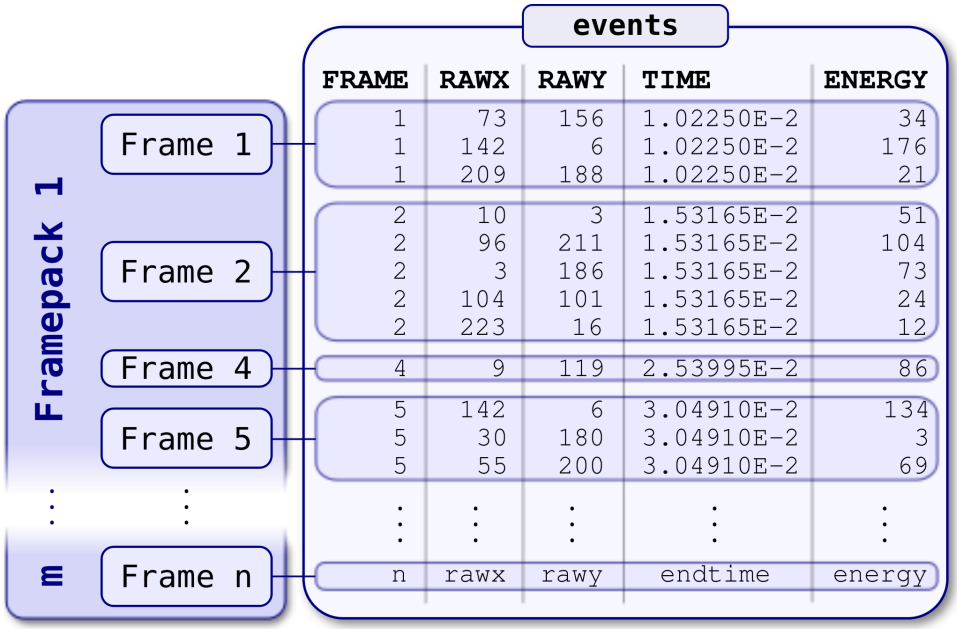


Figure 4.5: Splitting of event data with respect to frame (or time) affiliation.

FPFindHotPixel uses several data reduction and filtering steps. These are arranged consecutively with each step working with the results of preceding functions. First of all, the event data are read in and split up into logical chunks (see Fig. 4.5). Depending on whether the data contain frame or time information, the split is performed with respect to events belonging to the same frame or to the same time slice. Neither frame numbering nor time necessarily have to be consecutive, which has to be taken into account when combining a certain number of slices. The respective amount of frame or time slices which should be "glued" together are adjustable through the `BUNCHSIZE` parameter (see Table 4.2).
All events which are part of one of these "frame bunches" are gathered in a two-dimensional array, which has the same dimensions as the CCD chip. As the data will be split up into blocks by the detection algorithms later on, the detector dimensions have to be a multiple of `XBLOCKSIZE` and `YBLOCKSIZE` respectively. This constraint is necessary to assure that all blocks have the same size. However, this should be a problem neither for testing with *XMM-Newton* data (64×200 pixels) nor for analysing eROSITA data (384×384 pixels) when choosing, e.g., quadratic blocks with an edge length of 4 or 8. These values also proved to produce reasonable results.
As soon as gathering data for the first frame bunch is finished and the data have been marked as being available for analysis, the multithreaded recognition routines will start. The algorithm itself will be discussed in Chap. 4.7.4 and 4.7.5 in more detail.
After the last frame has been processed, all bright pixel and bright line findings will be filtered for relevance, i.e. only those which exceed a certain threshold for an adjustable fraction of all data will be considered as valid detections. If a bad pixel search for

Table 4.1: Example for a valid FITS input table

Row	FRAME	RAWX	RAWY	PHA
1	1	259	146	533
2	1	319	375	728
3	1	266	145	294
4	1	223	197	47
5	1	295	118	1324
6	2	4	327	1175
7	2	98	161	887
8	2	65	59	68
9	2	194	186	1139
10	3	280	314	60
⋮	⋮	⋮	⋮	⋮

the current detector has already been performed in the past, there is likely some kind of calibration file which contains the results of past runs. If so, these former results will be compared to the recent upshot and, if necessary, the calibration file updated. Afterwards, information about the recognised bright pixels and lines are stored in two additional output files (a FITS table and a FITS-image in which only hot pixels and bright lines show up). A simplified overview of the whole process is illustrated in Fig. 4.6.

4.7.2 Data input

The program expects the input event data to be in *FITS* tabular format and to contain the columns `FRAME` and/or `TIME`, `RAWX`, `RAWY` and `PHA`. There have to be either a `FRAME`- or a `TIME`-column (or both of them), being used by the detection software to determine how many frames the data consists of. If there is neither of them, the program will exit. Furthermore, it is absolutely mandatory that the input file is sorted by frame number (or by time respectively). For example the beginning of an event file could look like this:
For reasons of compatibility, especially in terms of testing with data from other satellite missions like *XMM-Newton*, there is the possibility to change the captions of all input and output columns through the PAR-file or during program launch. For example, in the case of *XMM-Newton* event files, the energy column is named `ENERGY` instead of `PHA`.
During launch, several parameters have to be specified by the user, which have influence on the way data are processed and on detection accuracy. Moreover, there are several "hidden" parameters in the PAR-file, which will not be queried but can be altered on demand. This is possible either by setting them explicitly via the command line during start (e.g., `./FPFindHotPixel numthreads=2`) or by modifying the PAR-file before running the program. All parameters are explained in Table 4.2.

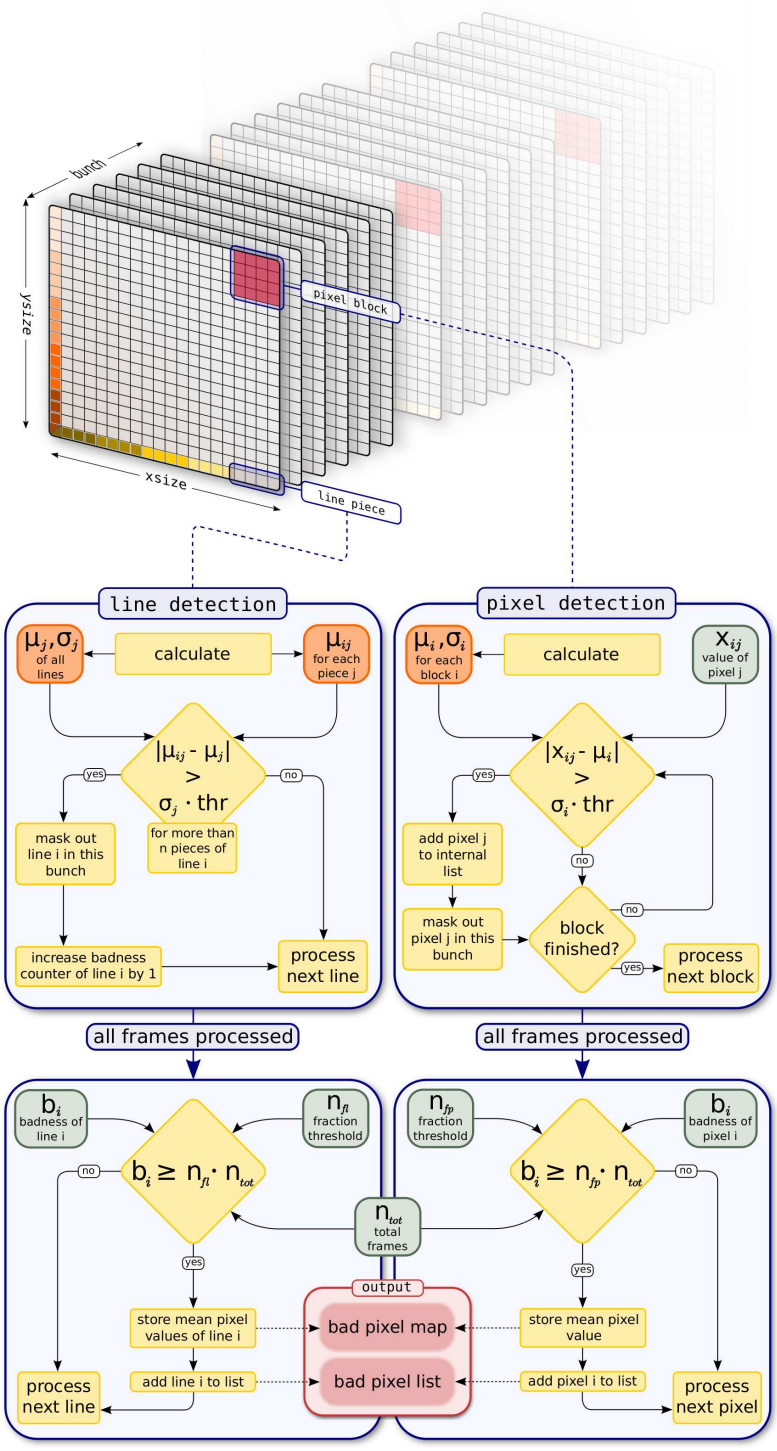


Figure 4.6: Overview of data splitting, processing and bad pixel recognition. Due to reasons of simplicity, data skipping and flicker-tests are not included here.

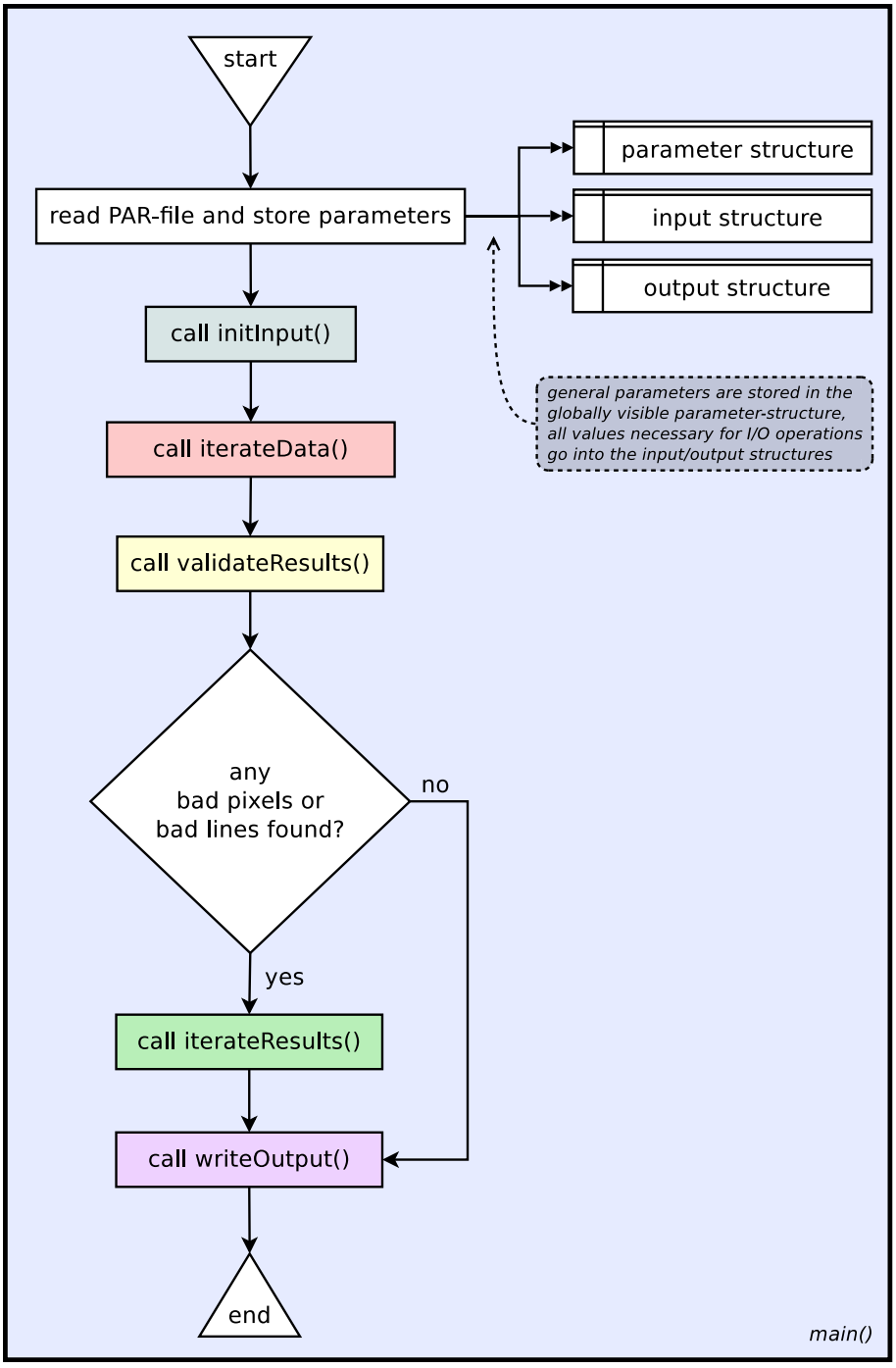


Figure 4.7: Raw schematics of the basic application flow of *FPFindHotPixel*. The main worker function which is responsible for staging input data and administering the recognition threads is marked in red.

The "type"-column indicates, whether the value will be queried during launch ('q' for query, 'h' for hidden) and if the input should be remembered permanently ('l' for learn).
Note: the `NUMTHREADS` parameter should be handled with care. Tests have shown that performance can decrease significantly when the value exceeds the actual number of physical cores in the machine. Moreover, if `NUMTHREADS` is set even higher the program may freeze or crash.

4.7.3 Data processing

Having successfully opened the input file, checked its sanity and determined how many data frames it contains, the program will check if it is possible to skip parts of it during bright pixel detection (see Fig. 4.8). This makes sense if the size of the dataset easily exceeds the amount necessary to perform reasonable statistical calculations. The parameter `MINEVENTS` can be used for setting the minimum number of events one frame bunch has to contain before being processed on. This value not only influences the statistical quality of the dataset (the more events the more reliable the results of bright pixel checks will be) but it is also used for determining the fraction of data which will be skipped. If `BUNCHSIZE` is set to 0 the program will try to guess a reasonable value automatically, depending on the number of iterations `NUMITERS` (i.e. the number of frame bunches) which should be used for bright pixel search.
For example, let's think of an input file with 5 million rows and 10 events per frame. That is, the file contains 500.000 frames in total. The data will be split such that the resulting number of frame bunches matches the value specified in `NUMITERS`. If `NUMITERS` equals, e.g., 50 and `MINEVENTS` is set to 5000, there will be a total of

$$\texttt{BUNCHSIZE} = \frac{\texttt{MINEVENTS}}{\text{events per frame}} = \frac{5000}{10} = 500$$

frames in one frame bunch.

$$\Rightarrow \text{fraction of frames to process} = \frac{\texttt{NUMITERS} \cdot \texttt{BUNCHSIZE}}{\text{total frames}} = \frac{50 \cdot 500}{500.000} = 0.05$$

Consequently only 5% of the whole dataset would be part of bright pixel search in this example. However, it is up to the operator's opinion and to tests if the size of a frame bunch will suffice to yield reasonable results.
If `BUNCHSIZE` is manually set to, e.g., 5000 events the program can construct 100 frame bunches out of the data. If we further assume that `NUMITERS` is set to, e.g., 50 bunches the program will skip every 2nd bunch accordingly. If `NUMITERS` exceeds the number of available bunches, the `BUNCHSIZE`-value will be adjusted to a lower value in order to get enough iterations.
As processing time is an important issue and event lists can grow very big, *FPFind-HotPixel* is capable of working with multiple threads in order to take advantage of multicore machines. The number of worker threads can be set through the parameter

Table 4.2: Overview of all parameters which are adjustable through the PAR-file

parameter	type	flag	default	description
INFILE	s	ql	--	path to the input event file
OUTTABLE	s	ql	--	path to the designated output file
THRESHOLD	s	ql	2.0	multiplier to define detection threshold
DETWIDTH	i	ql	384	width of the detector in pixels
DETHEIGHT	i	ql	384	height of the detector in pixels
BADPIXDB	s	ql	--	path to the bad pixel database
OUTMAPPAT	s	h	_map	a short pattern which will be appended to the filename of the output table in order to set the output map filename
UPDATEDB	b	h	n	flag to set whether bad pixel database should be updated or not
EXTNAME	s	h	EVENTS	name of the input file extension which contains event data
LINEDIV	i	h	8	number of chunks one line will be split to
BUNCHSIZE	i	h	0	specifies how many frames a frame bunch should consist of. If set to 0, a reasonable size will be determined automatically
NUMITERS	i	h	200	number of iterations (framebunches) which should be used for calculations
MINEVENTS	i	h	5000	minimum number of events one frame-bunch must contain
LINEFRAC	i	h	30	percentage of the chunks of one line which have to exceed the threshold in order to get the line marked as bright
LINEFRMFRAC	i	h	30	percentage of all frames a bright line must exceed the threshold
PIXFRMFRAC	i	h	60	percentage of all frames a bright pixel must exceed the threshold
XBLOCKSIZE	i	h	8	width of the blocks the image should be split to (has to be factor of DETWIDTH)
YBLOCKSIZE	i	h	8	height of the blocks the image should be split to (has to be factor of DETHEIGHT)
CHIFLICONF	i	h	95	value in percent which defines the desired confidence level of the χ^2-flicker-test
KOLFLICONF	i	h	95	value in percent which defines the desired confidence level of the KS-flicker-test
STARTVAL	i	h	1	lowest coordinates used in event files (normally 0 or 1)
NUMTHREADS	i	h	1	number of CPU cores used for doing calculations

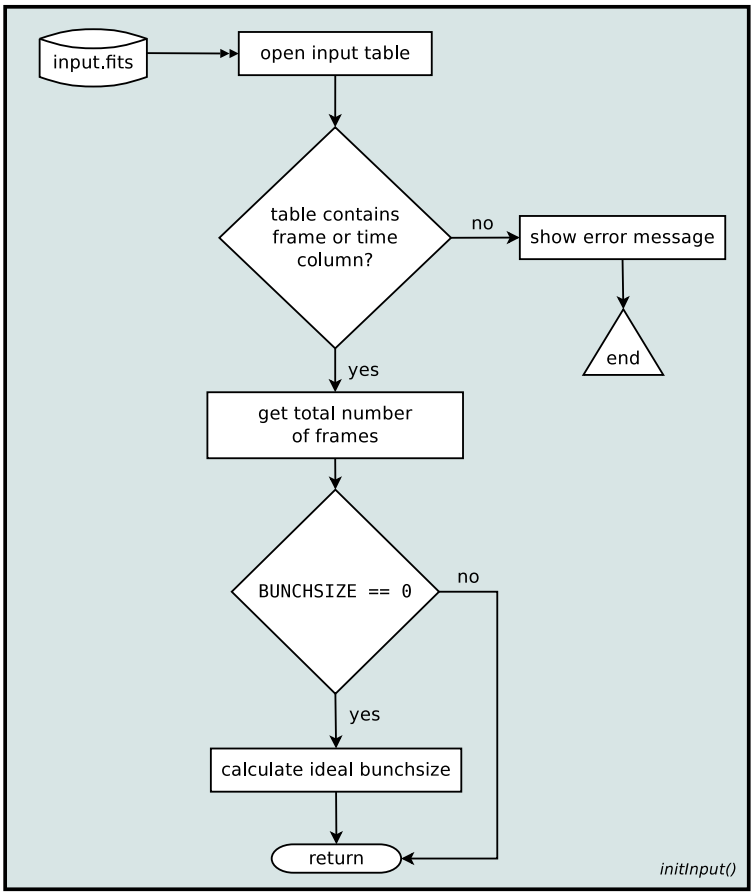


Figure 4.8: *initInput* cares for correct initialisation of the input event file, checks if it contains all columns necessary for the detection process, determines the number of total frames and calculates the ideal size of the frame bunches to be used.

`NUMTHREADS` and should not be larger than the number of physical cores of the CPU. Otherwise there may be a significant performance loss or even a program crash in the worst case.

Fig. 4.9 shows the function *iterateData* which basically controls data input to the worker threads it created in the beginning. As soon as it has finished reading in another frame bunch it will mark the bunch as being ready for processing. This continuous data supply (taking possible data skipping into account) will continue until end of file is reached. Meanwhile, the threads will check frequently if new frame bunches are available. If so, one of the threads will successfully acquire a mutual exclusion (mutex) lock on the data and start working on it (further explanations can be found in Chap. 4.7.4 and 4.7.5).

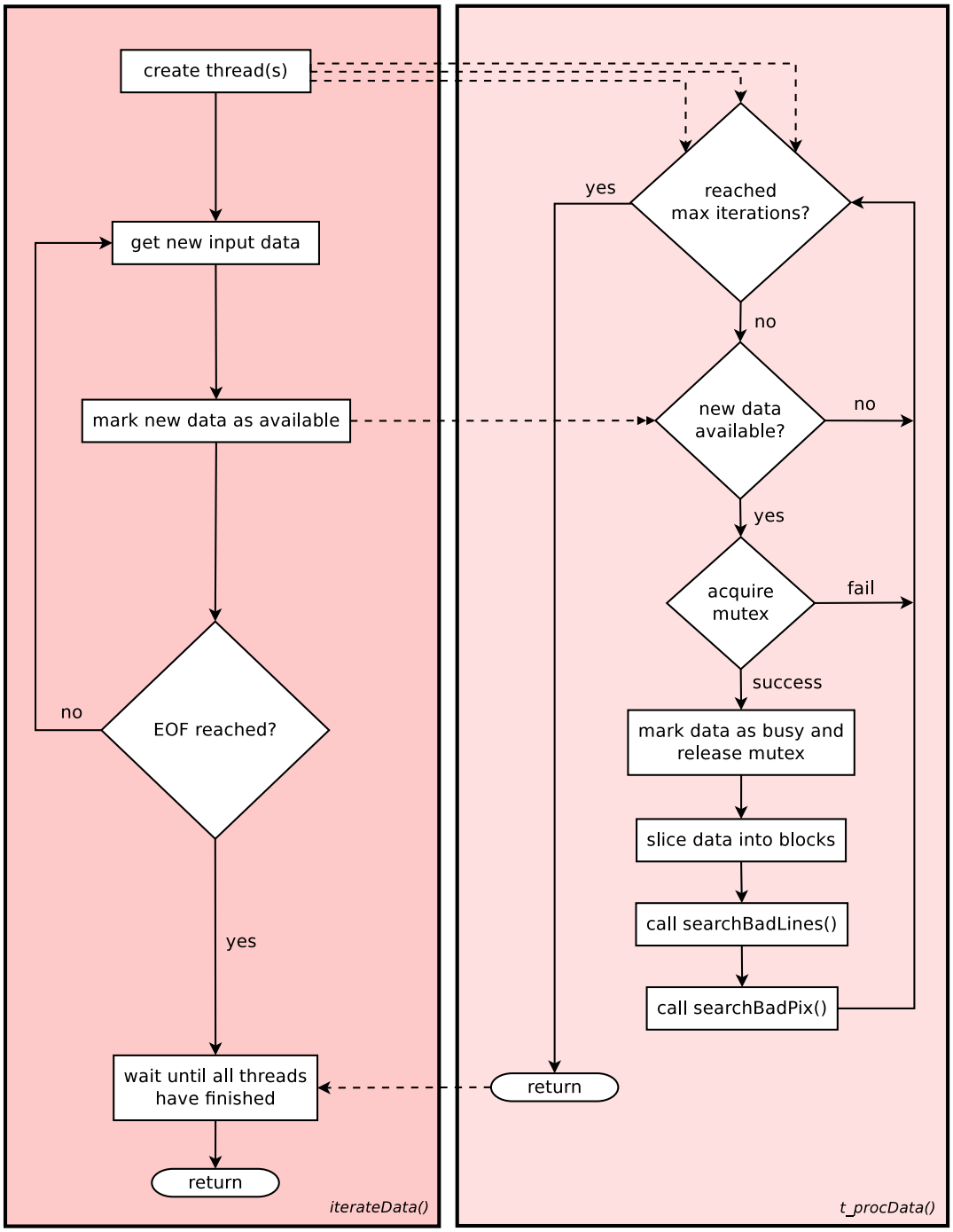


Figure 4.9: *iterateData* creates threads which start working in *t_procData* as soon as new data are available. In the end, *iterateData* waits for all threads to finish properly before returning.

4.7.4 Detection of bright lines

When the first frame bunch has been marked as "available", the worker thread(s) will start processing data. Their first purpose is the detection of bright lines. To achieve this each image line[4] is divided into `LINEDIV` sections (see Fig. 4.6). This sectioning is reasonable because of the existence of bright lines which are not necessarily bright over their whole extent, but only in part. Afterwards mean value and standard deviation of each part are calculated, as well as the total mean of all lines.
The actual decision whether a line part contains abnormal data will then be made by the function *searchBadLines* (see Fig. 4.10) using the criterion

$$(\mu_{ij} - \mu) > (\vartheta \cdot \sigma_{ij}) \tag{4.1}$$

with μ_{ij} the mean value of all pixels in section j of line i, μ the mean value of all lines, ϑ the factor used to adjust detection sensitivity and σ_{ij} the standard deviation of section j in line i.
If this holds true for more than `LINEFRAC` % of all sections the line will not only be catalogued but its values in the internal data array will be adjusted such that it will not influence further detection cycles anymore. If the whole line were nullified completely though, the chance that adjacent pixels suffer a false positive detection would rise as the local mean value (see Chap. 4.7.5) would decrease. To account for that and keep the impact on the detection accuracy low, the line will be overwritten with the median of all median values of all lines. As data have been modified by this operation, all statistical values will be re-calculated afterwards by another call to *calcStripeStats*.
As soon as no more bright lines are detected hot pixel detection begins. The advantage of masking out bright lines beforehand is an increase in hot pixel detection accuracy as pixels of bright lines which lie in a hot pixel's neighbourhood (i.e. in the same block) cannot influence the detection threshold anymore.

4.7.5 Detection of bright pixels

Once *searchBadLines* has finished its work, the actual hot pixel detection *searchBadPix* starts. For each block in the current frame bunch, both mean value and standard deviation of the contained pixels are calculated and stored for further usage. Afterwards, each block of data is iterated step by step and the respective pixels marked if their properties match the expression

$$\left(\Sigma_{bi} - \overline{\Sigma_b}\right) > (\vartheta \cdot \sigma_b) \tag{4.2}$$

with Σ_{bi} the value of pixel i in block b, $\overline{\Sigma_b}$ the mean value of all pixels which lie in block b, ϑ the factor used to adjust detection sensitivity and σ_b the standard deviation of the current block.
Once having found a pixel whose value is high enough to meet the condition, its position is stored. Having done this, the pixel's value is set to the median of all pixels

[4] bright lines are recognised in vertical direction only

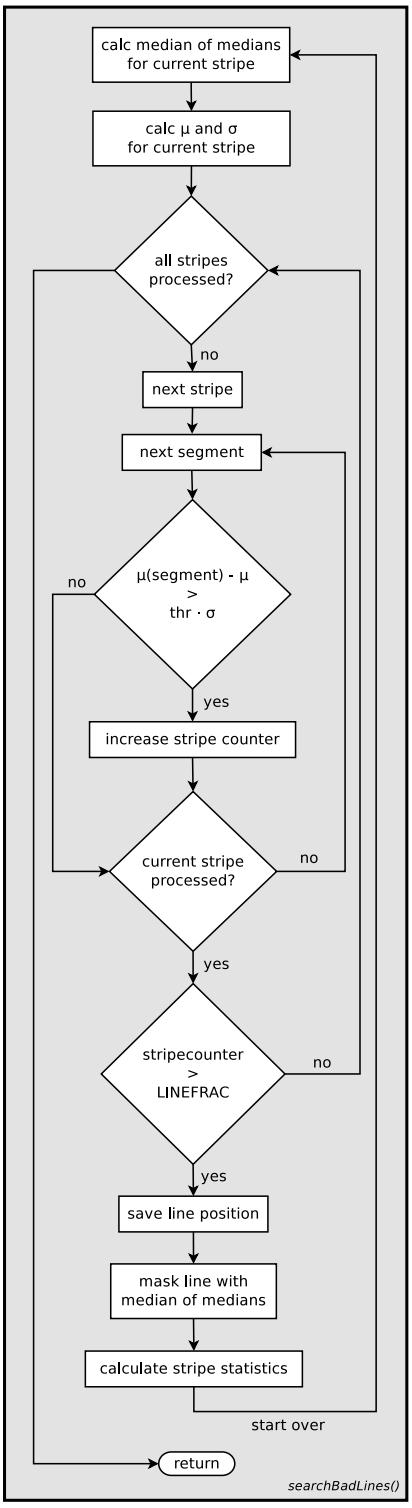


Figure 4.10: *searchBadLines*: the core of bad line detection.

in the current block and the function *calcBlockStats* is called once again in order to compute the changed mean and standard deviation for the current block. As soon as all pixels of all blocks of the current bunch have been dealt with, the routine returns to the underlying work-function and continues with the next iteration.
When the input work-function *iterateData* has finished, it will wait until all worker threads are done. Afterwards *validateResults* checks if the abundance of the found bad pixels and bad lines exceeds the specified thresholds `PIXFRMFRAC` and `LINEFRMFRAC`, i.e. only findings which were detected in at least a given percentage of all frames will be considered to be valid (see Fig. 4.12). Bad pixels and bad lines which pass this last test will be incorporated into the output files.

4.7.6 Postprocessing

Despite having collected data about the coordinates of the bright pixels and bright lines, further information (e.g., type or minimum and maximum Analog Digital Unit (ADU) value) has still to be gained. Due to possible data skipping and the breakup of data into bunches this information cannot be obtained during bad pixel search itself. Apart from that it will not be clear from which pixels additional data will be needed until the detection is complete.
In case any defects were found, the function *iterateResults* will generate one or more worker threads (depending on the value of `NUMTHREADS`) which will start selecting specific data on bad pixels from the input file. This is achieved by making use of the extended filename syntax which is a feature of the *CFITSIO* library. Mean and median as well as highest and lowest ADU value will be calculated for each bright pixel. In case of bright lines no additional information will be added to the output. In the map, however, bad lines will show up with the mean value of all pixels in that line in order to compare their brightnesses to other findings.
Additionally, for bright pixels another test is performed which tries to determine whether they have to be classified as flickering or as constantly bright. For this purpose all data of the respective pixel are divided into equal time slices (their size is adjustable through the `FLITIMESTEPS` parameter). For each pixel the number of events per time slice are determined and fed into a histogram. Depending on the data volume of the histogram, either a χ^2-test (in case of many counts per bin) or a Kolmogorow-Smirnow (KS)-test are carried out.

4.7.7 Flickering and static bright pixels

Both χ^2- and KS-test are realised in the style of the flickering tests of the *badpixfind* tool which is part of the *XMM-Newton SASS*.

χ^2-test

Given the fact that an "events per time slice" histogram with N bins contains enough counts n_i per bin (i.e. a value greater than 0) a χ^2-test will be performed on the histogram data. The test tries to decide whether the count-rate resembles a normal distribution with a certain confidence level. This confidence level is adjustable

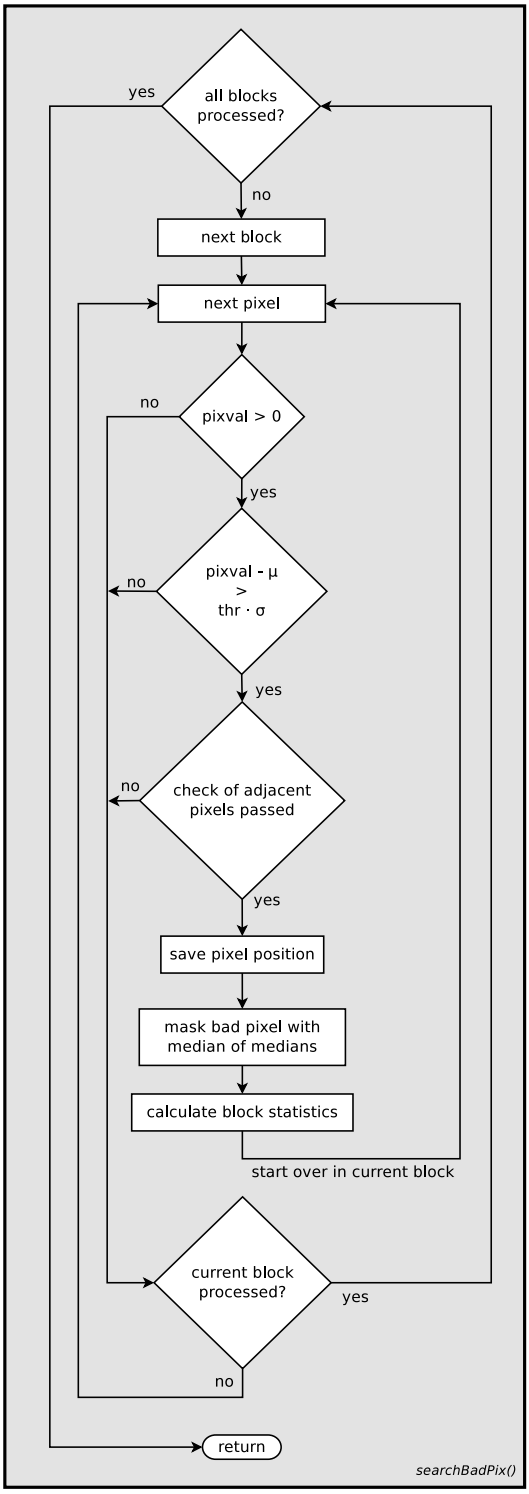


Figure 4.11: *searchBadPix*: the core of the bad pixel recognition algorithm.

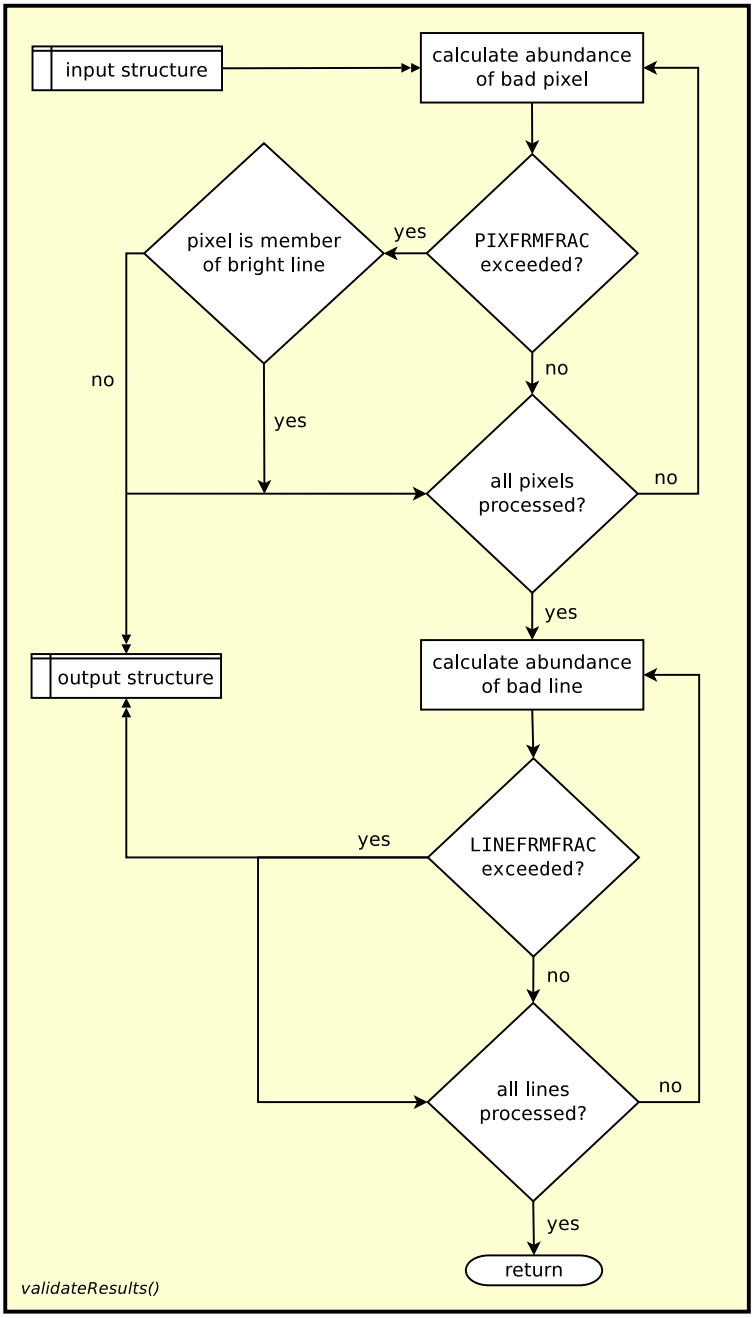


Figure 4.12: *validateResults* compares the abundances of all bright pixels and bright lines with the thresholds specified by the parameters. Only defects whose abundance lies above the threshold will be incorporated to the output structure.

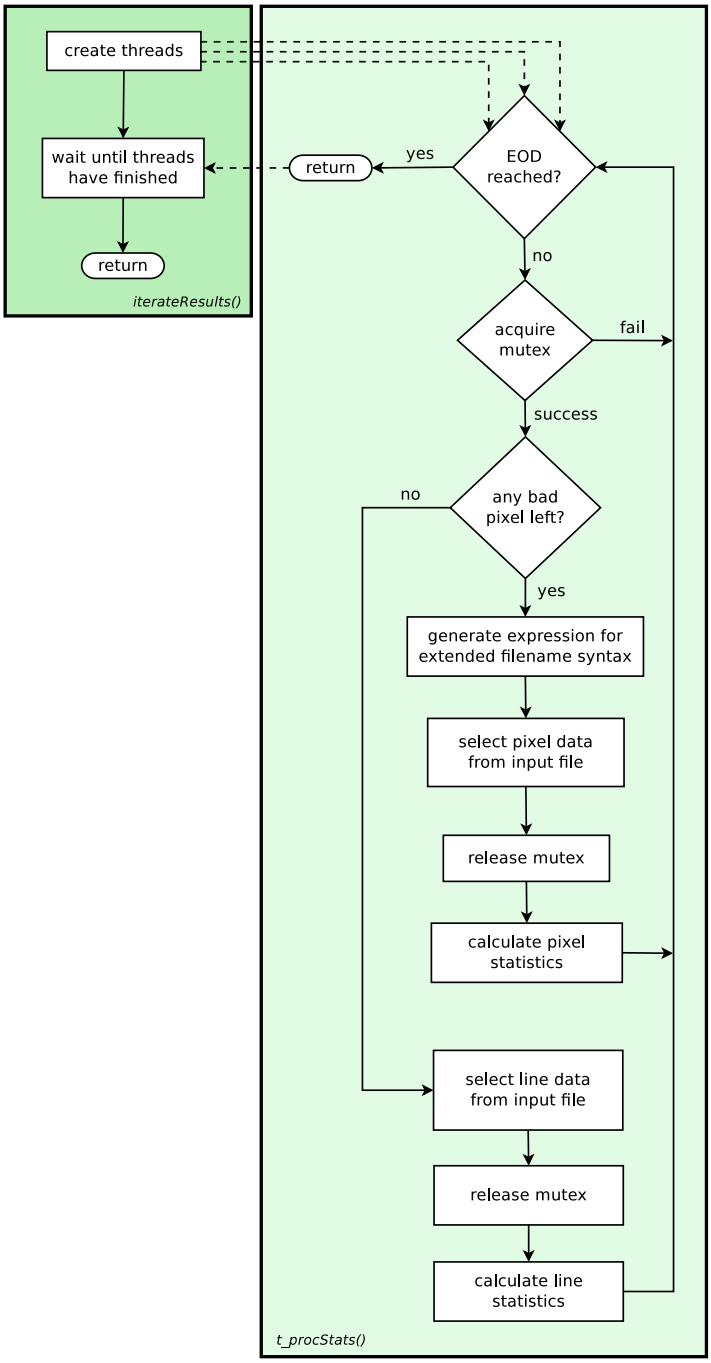


Figure 4.13: *iterateResults* creates worker threads which collect further data on each bright pixel and line. Mutexes are used to prevent different threads from accessing the same data. With the functionality of the extended filename syntax it is possible for each thread to filter the whole input file just for those events which were caused by the particular defect it is working on.

through the parameter `CHIFLICONF`. The value for χ^2 is calculated as follows (Agarwal, 2009):

$$p_i = \Phi\left(b_{i+1}, \mu, \sigma\right) - \Phi\left(b_i, \mu, \sigma\right) \tag{4.3}$$

$$\chi^2 = \sum_{i=1}^{N} \frac{\left(n_i - \left(p_i \cdot n\right)\right)^2}{\left(p_i \cdot n\right)} \tag{4.4}$$

with $\Phi\left(x, \mu, \sigma\right)$ the cumulative probability density for a random variable x, N the total number of histogram bins, b_i the value at left border of bin i, n_i the number of events in bin i, n the total number of events and p_i the probability to get $b_i \leq n_i < b_{i+1}$ events in bin i.
The critical value c_{χ^2} is computed by integration of the cumulative χ^2-distribution function in the interval $[k; +\infty]$ with k being the confidence `CHIFLICONF` and ν the number of degrees of freedom (= number of events minus the number of fitted parameters):

$$c_{\chi^2} = \int_k^{+\infty} p\left(x\right) \mathrm{d}x = \int_k^{+\infty} \frac{1}{2\Gamma\left(\frac{\nu}{2}\right)} \left(\frac{x}{2}\right)^{\frac{\nu}{2}-1} \exp\left(-\frac{x}{2}\right) \mathrm{d}x. \tag{4.5}$$

If $\chi^2 > c_{\chi^2}$, the event distribution of the respective pixel does not fit a normal distribution, which indicates a flickering behaviour. Hence, in the output table the pixel will be marked as `'f'` ('flickering'). If the χ^2-test is passed the pixel will be marked as `'b'` ('bright'). The χ^2-test is implemented through the function *checkChiSquare* which is illustrated in Fig. 4.14.

Kolmogorow-Smirnow-test

If the events per time slice histogram does not contain enough data to perform a χ^2-test (i.e., there are bins with no counts at all) a Kolmogorow-Smirnow-test (KS-test) will be run which compares the histogram's event distribution with a Poisson distribution (see Fig. 4.15. For this purpose, the histogram will be sorted in ascending direction, making it comparable to a cumulative probability density. A confidence level k which can be set through the parameter `KOLFLICONF` is used to adjust how strict (in terms of a critical value c_{KS}) the algorithm should be.

$$c_{\mathrm{KS}} = \sqrt{\frac{1}{n} \log\left(\sqrt{\frac{2}{1-k}}\right)} \tag{4.6}$$

The KS-test as such compares the "real" event counts n_i of bin i and additionally the counts n_{i-1} of the previous bin with the probability obtained by the cumulative probability density of the Poisson distribution (with μ being the mean; Agarwal, 2009).

$$\Phi\left(n_i, \mu\right) = \int_0^{n_i} \frac{\mu^k}{k!} \exp\left(-\mu\right) \mathrm{d}k \tag{4.7}$$

That is, if the conditions of Eq. 4.8 and/or Eq. 4.9 (with N the total number of events) are not met, the test failed and the pixel will be marked `'f'` for 'flickering'

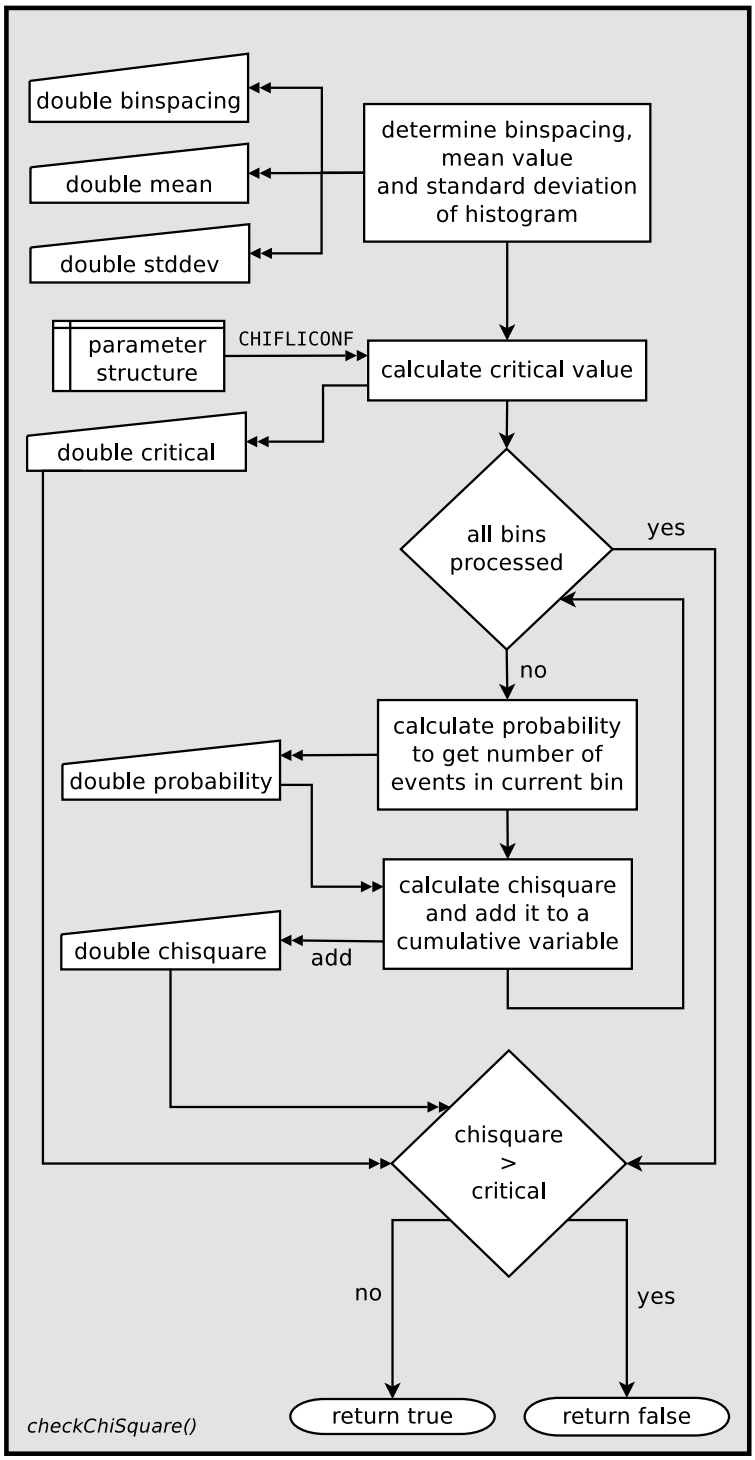


Figure 4.14: The function *checkChiSquare* can perform a χ^2-test on the histogram data, given the fact that the histogram contains enough counts per bin.

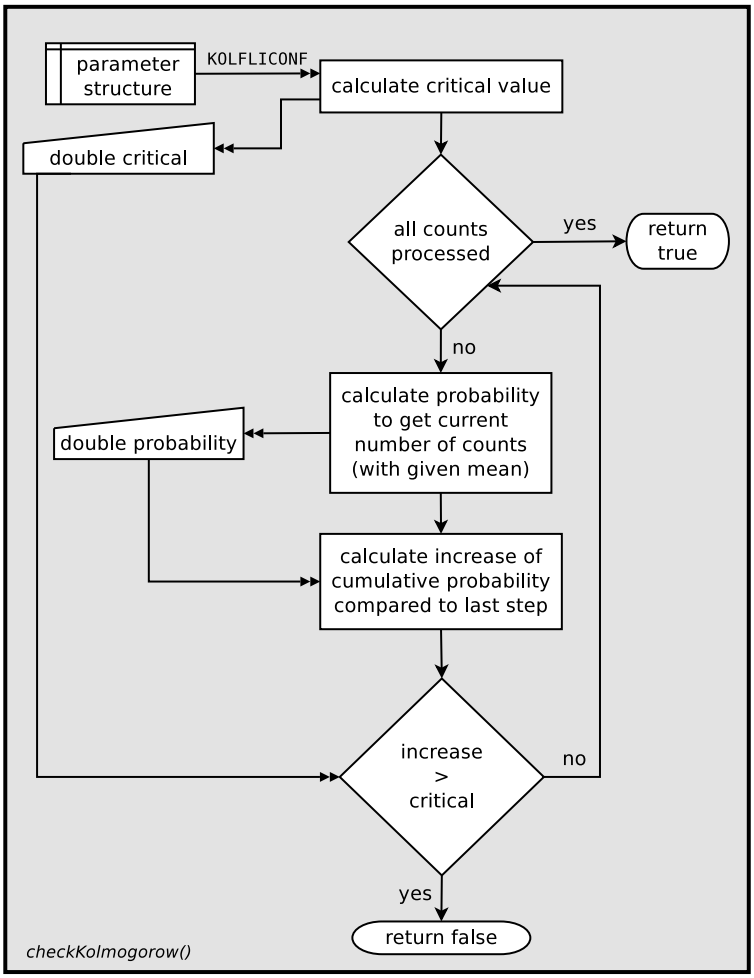


Figure 4.15: In case the histogram does not contain enough data, the function *check-Kolmogorow* will perform a KS-test.

(otherwise `'b'` for 'bright').

$$\frac{n_i}{N} - \Phi\left(n_i, \mu\right) \leq c_{\mathrm{KS}} \tag{4.8}$$

$$\frac{n_{i-1}}{N} - \Phi\left(n_i, \mu\right) \leq c_{\mathrm{KS}} \tag{4.9}$$

4.7.8 Output

Running *FPFindHotPixel* on sample data from *XMM-Newton* (see Fig. 4.17) with the starting parameters listed in Table 4.3 produced output as shown in Fig. 4.18. Most of the console output, however, will only be visible if the debugging mode of the *FPIPE* is active. Otherwise only runtime errors and warnings will trigger an output

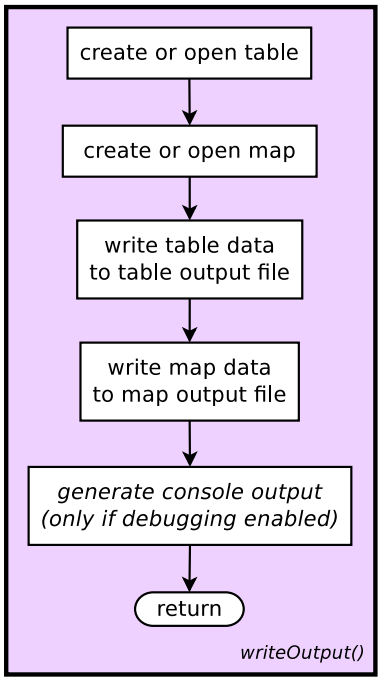


Figure 4.16: If not existing, the output files will be created. Otherwise they will be opened and data which were already present in the files will be overwritten. In case of debugging mode being active the results will also be shown on the console.

Table 4.3: The starting parameters which were used for a test run on *XMM-Newton* data.

parameter	value
EXTNAME	PNIME1
ENERGYCOL	ENERGY
THRESHOLD	2.0
DETWIDTH	64
DETHEIGHT	200
STARTVAL	0

to the console. Independent from that, the results will be written to the FITS table and be visualized in the FITS map additionally (Fig. 4.16).
A comparison with Fig. 4.17 shows that all three bright lines have been detected correctly. Moreover the algorithm found 13 presumably bright/flickering pixels.

- `RAWX/RAWY` is the position of the bright/flickering pixel
- `TYPE` shows the type ('b' for bright or 'f' for flickering) of the pixel

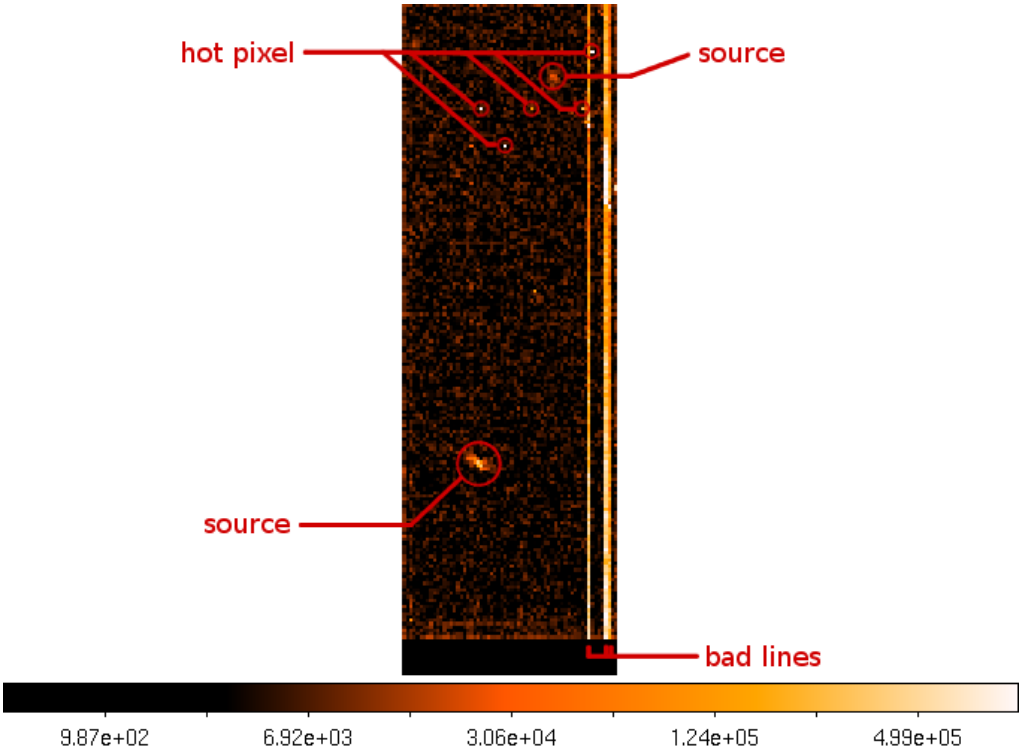


Figure 4.17: Integrated data of more than 120 k frames for CCD 10 of *XMM-Newton*. Bad lines, several hot pixels and presumable sources are highlighted.

- `EHIGH / EMED / ELOW / EMEAN` show the highest/median/lowest/mean value of the pixel in ADU

If it is not clear at first glance whether a pixel is a false positive or if its type was chosen correctly, it can be reasonable to have a closeer look at its ADU values before making a decision. A flickering bright pixel might for example have a very high maximum ADU while the other values will be much smaller. In contrast to that, the mean and median of a constantly bright pixel will be significantly bigger.

```
No TIME-column found. Trying to use FRAME-column instead...
total rows: 5780101
total frames: 112133
events per frame: 51.55
Bunchsize automatically set to 97.
Processing every 5-th bunch.

*** RESULTS ***

possible bright lines:

###  RAWX
  1   56
  2   61
  3   62

total hot pixel candidates found: 13

###  RAWX    RAWY    TYPE    EHIGH   EMED    ELOW    EMEAN
  1    3      17     b       8.4e+02 2.4e+01 2.0e+01 3.0e+01
  2   24      63     b       1.2e+03 2.4e+01 2.0e+01 2.5e+01
  3   23      64     b       2.0e+03 2.4e+01 2.0e+01 2.6e+01
  4   40     115     f       1.8e+03 2.4e+01 2.0e+01 2.7e+01
  5   64     145     b       4.0e+02 4.7e+01 2.0e+01 5.5e+01
  6   64     146     f       4.1e+02 4.8e+01 2.0e+01 5.6e+01
  7   21     158     b       1.3e+03 2.2e+01 2.0e+01 2.5e+01
  8   31     158     f       1.4e+03 4.7e+01 2.0e+01 5.4e+01
  9   24     169     b       1.8e+03 4.6e+01 2.0e+01 5.1e+01
 10   39     169     b       1.6e+03 2.3e+01 2.0e+01 2.5e+01
 11   54     169     f       1.3e+03 2.6e+01 2.0e+01 2.8e+01
 12   55     171     b       1.9e+02 2.1e+01 2.0e+01 2.2e+01
 13   57     186     f       3.0e+03 2.9e+01 2.0e+01 3.1e+01
```

Figure 4.18: Example for an output of a successful run of *FPFindHotPixel* as it would appear on the console. Three bright lines and 13 bright pixels of different types were found. As the first couple of lines indicate, the event file was big enough to permit skipping of data bunches. Consequently every 5th bunch was processed and the others were thrown away.

4.8 *FPFindColdPixel*

Other than the search for bright pixels, the approach to cold pixel detection was taken in quite a different way. For this reason cold pixel search has been implemented separately from bright pixel recognition, as these tasks don't have much in common. The program *FPFindColdPixel* has been developed under the assumption that cold pixels will never show any signal at all, no matter how long the exposure time was (I. Kreykenbohm, 2010, priv. comm.). The problem seems to get nearly trivial from this point of view, as it is only necessary to look for pixels which are completely dark. The size of the dataset, however, has significant influence on the quality of the

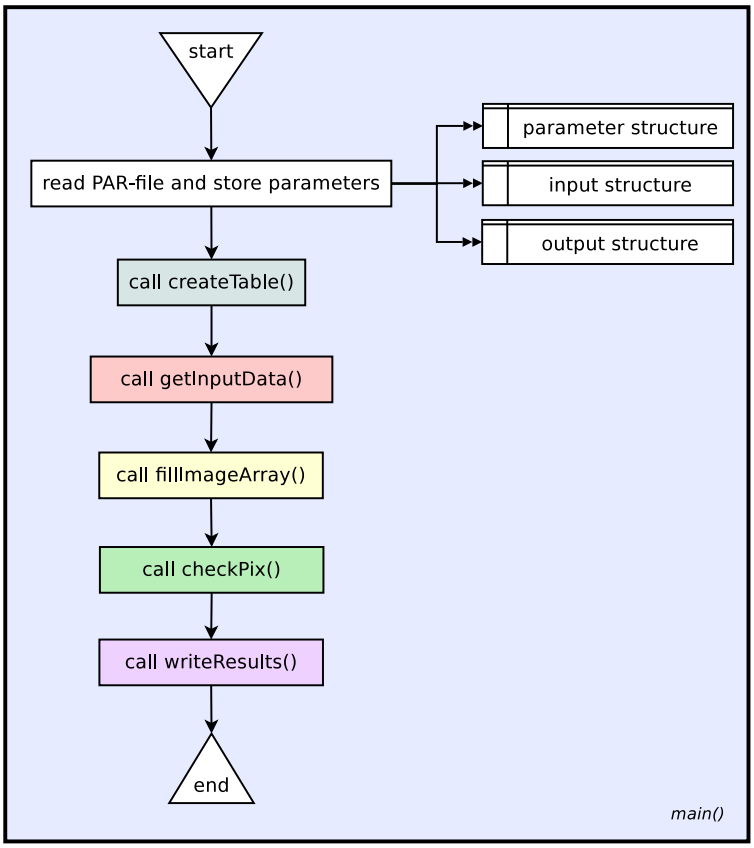


Figure 4.19: The *main* function: first of all, parameters and structures are initialized and the output table is created. After that the input data are assembled to an image which will be analyzed for cold pixels by *checkPix* and *checkBorder*. Finally, all results will be written to the output file.

detection. This limitation made it worth while implementing an additional filtering method, which could account at least partially for these effects.

4.8.1 Overview

In analogy to *FPFindHotPixel*, cold pixel detection requires a couple of parameters during program start. As *FPFindColdPixel* uses a simpler detection method, much less parameters are needed though.
After parameter initialisation the input data will be imported and the output file opened (i.e., if not existing it will be created. Otherwise all data already present will be removed.). As soon as all input data has been read in and stored to an internal data structure, the detection algorithm will start with the analysis. The coordinates of cold pixel and cold line findings, if any, will then be written to both standard output (which will be the console normally) and output file. Additional information

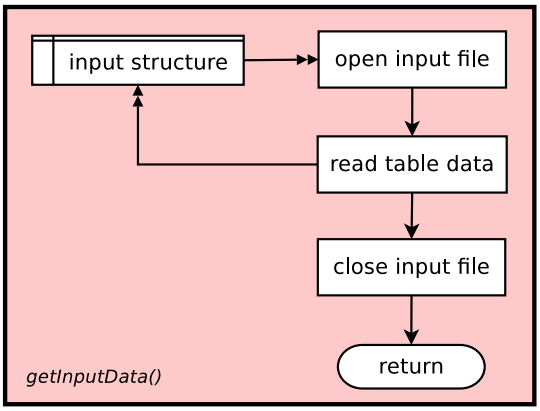


Figure 4.20: *getInputData* opens the input file from the path specified in the input structure, imports its data into the structure and closes the file.

about the extent of the pixel defect as well as its type (single cold pixel or cold line) will be included also.

The *main*-function of *FPFindColdPixel* is illustrated in Fig. 4.19. Its subfunctions are marked in different colours which eases their assignment to the corresponding flow diagram.

4.8.2 Data input

Just like in *FPFindHotPixel*, several parameters are read in at program start which will later be needed during runtime. For this reason, they are stored in an internal input structure and hence will be easily accessibly anytime. Some of the parameters will be queried directly on the command line when the program is starting. The others must be changed manually by specifying them as arguments in the program call, or by editing the PAR-file directly. More information about all parameters and their meaning can be looked up in Table 4.4.

After parameter initialisation the input data will be imported (see Fig. 4.20) and the output file opened (i.e., if not existing it will be created). Otherwise all data already present will be removed.).

A significant of a reliable detection of cold pixels is to distinguish them from pixels which show no signal because they just have nothing to show. A “real” cold pixel is different from that, as it loses signal electrons due to local crystal defects. Consequently, reasonable cold pixel analysis requires a sufficient amount of data to work with. Even background noise which normally is an unwanted accompaniment of detectors shows to be a valuable help in this task. For this reason *FPFindColdPixel* sums up all available input data to one single image before starting the detection on it (see Fig. 4.21).

Table 4.4: Overview of all parameters, which are used in *FPFindColdPixel*. All of them are adjustable through the PAR-file or directly via command line.

parameter	type	flag	default	description
VERSION	i	h	1	version of *PAR* file
XPANAME	s	h	FPFindColdPixel	name which is used by the XPA to identify the program
INFILE	f	ql	--	path to input event file
OUTFILE	f	ql	--	path to designated output file
BADPIXDB	f	ql	--	path to cold pixel database file
UPDATEDB	b	h	n	update database with new bad pixels and remove vanished ones
FRMCOL	s	h	FRAME	name of frame column in input event file
TIMECOL	s	h	TIME	name of time column in input event file
PHACOL	s	ql	PHA	name of energy column in input event file
XCOL	s	h	RAWX	name of x-coordinate column in input event file
YCOL	s	h	RAWY	name of y-coordinate column in input event file
RANGECOL	s	h	RANGE	name of range column in output event file
TYPECOL	s	h	TYPE	name of type column in output event file
DETWIDTH	i	ql	384	width of the detector in pixels
DETHEIGHT	i	ql	384	height of the detector in pixels
LINEPERCENT	i	h	10	threshold for cold line trigger (value in percent)

4.8.3 Detection algorithm

The cold pixel detection algorithm itself is fairly simple compared to the more sophisticated algorithm of *FPFindHotPixel*. Consequently there was no need to implement, e.g., multi-threading support, as the program needs only a few seconds to fully process even very large files with several tens of millions of rows.
The function *checkPix* (see Fig. 4.22) iterates pixel by pixel through the image, firstly checking in each step if the value of the current pixel is equal to 0. If so, a call to *checkBorder* (illustrated in Fig. 4.24) will perform another check, showing whether the respective pixel is located in a corner, at a border or in the midst of the frame. Depending on the outcome of the second check, a third check decides whether the

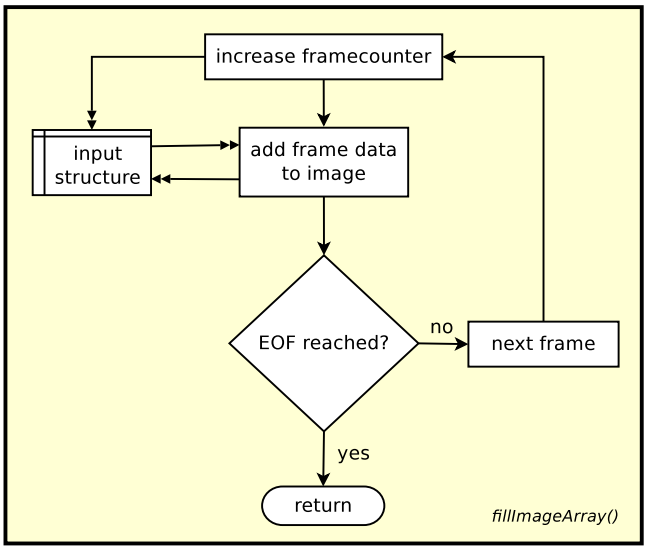


Figure 4.21: The purpose of *fillImageArray* is to assemble all input data to a single image. This is achieved by a simple loop which iterates frame after frame through the available data, adding them to the image and increasing a frame counter accordingly. All results are stored in the input structure again.

current pixel supposedly is cold or not:

- if located in a corner, the pixel will be considered as cold when at most one of its three neighbours shows no signal at all
- if located at a border and not in a corner, the pixel will be considered as cold if at most two of its five neighbours show no signal at all
- if located somewhere in the midst of the frame (i.e. not at a border and not in a corner), the pixel will be considered as cold if at most three of its eight neighbours show no signal at all

Some of the possible combinations of the second and third check are shown in Fig. 4.23. The checks on pixel position relative to the image boundaries and on cold neighbours were introduced in order to confine the number of false positive detections. Typically, false positives occur when analyzing datasets which are not voluminous enough to supply every pixel with at least a small background signal. Hence the algorithm cannot distinguish between cold pixels and "empty" healthy pixels. Due to the additional checks, at most widespread blank regions can be recognized and ignored while, e.g., detections of cold lines are still possible. Nevertheless the results of cold pixel checks on small data sets which possibly lack of background noise (e.g., simulation data) tend to show lots of false positives and should be generally distrusted. Another option to circumvent this problem would be to repeat the cold

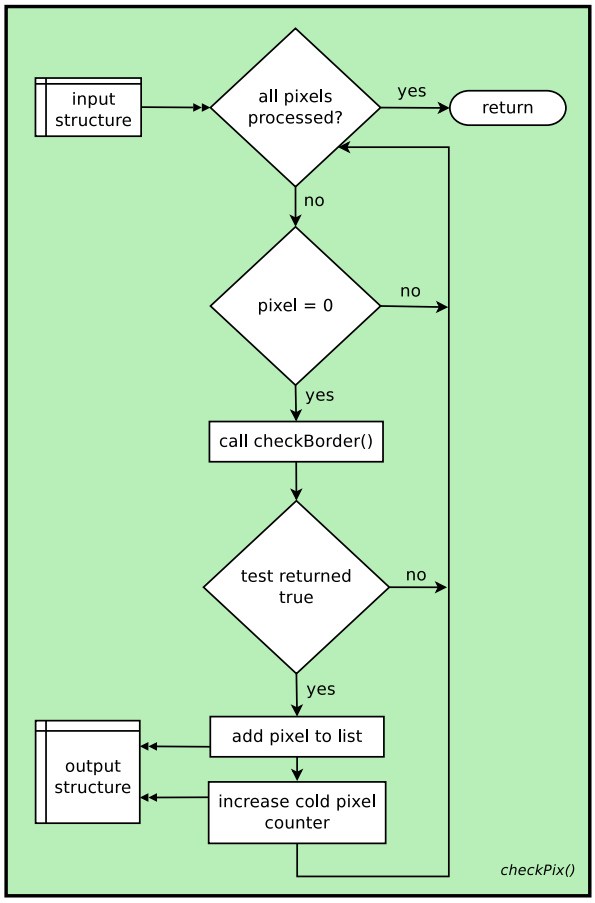


Figure 4.22: *checkPix* reads data from the input structure pixel by pixel, checking if any of the pixels has signal 0. If so, further checks will be performed by calling *checkBorder*. Pixels which pass all tests will be catalogued in a list in the output structure.

pixel check several times with different datasets (of the same detector, of course) and compare the results. Pixels which are found persistently are likely to be cold then.

4.8.4 Output

Initialisation of the output files works quite similiar to *FPFindHotPixel*. Two kinds of output files will be created:

- A FITS table containing coordinates, vertical extent and type of the cold pixels/lines
- A FITS image resembling a visual map of the cold pixels/lines. They will show up as white dots and stripes, depending on their position and type, while the rest of the normal data will be masked out

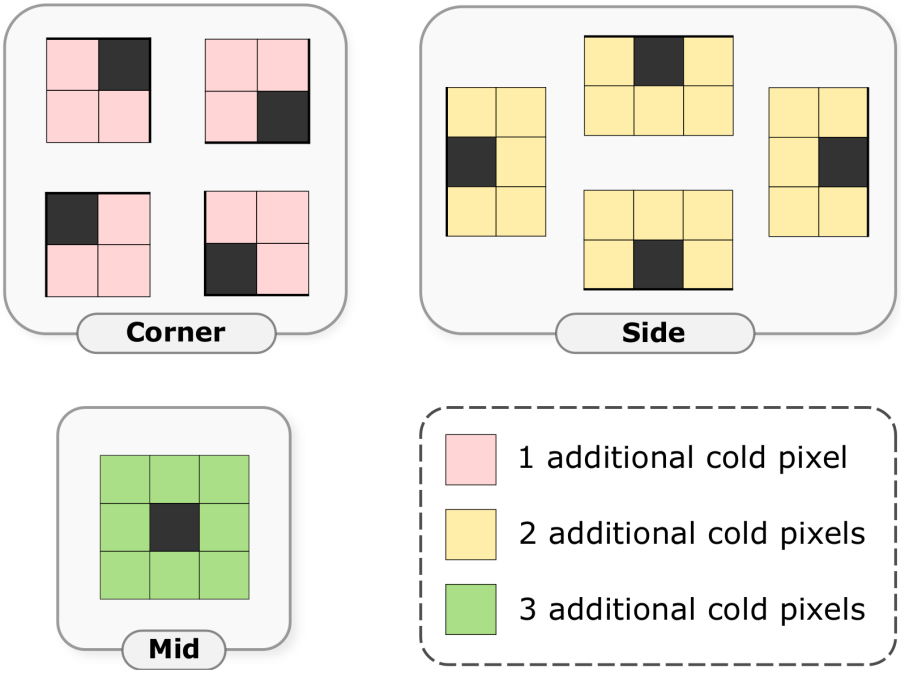


Figure 4.23: The gray boxes illustrate the different possibilities *checkBorder* will test on, depending on the location of the current pixel (marked in black). Bold lines mark image boundaries. Depending on their color, boxes suggest the number of surrounding pixels which may also be cold without violating the filtering rules. For example only one out of the three pixels which are next to a cold pixel in a corner may be cold too. Otherwise the algorithm will ignore the current pixel.

If the output files are already existing, it will be checked whether they are valid FITS files. In case of one or both of them being invalid the program will exit, giving the user a chance to change the output filenames or have a look at the files in question by himself. Otherwise all existing Header Data Unit (HDU) of the files except for the header unit will be deleted. These steps are visualised in Fig. 4.25.
Although the algorithm which was described in Chap. 4.8.3 will have accounted for the recognition of cold pixels at this stage, the decision whether some of these cold pixels (if found any at all) could be interpreted as a cold line is still owing. Unlike bright line detection, a simple occurrence check is sufficient to determine if a line contains cold pixels only or if these cold pixels can be said to form a cold line. The corresponding condition may be expressed as

$$h_i \geq \frac{l_p \cdot \Lambda_y}{100} \tag{4.10}$$

with h_i the number of cold pixels in line i, l_p the parameter `LINEPERCENT` which defines the pixel fraction having to be cold in a line and Λ_y the size of the detector in y-direction as specified in `DETHEIGHT`.

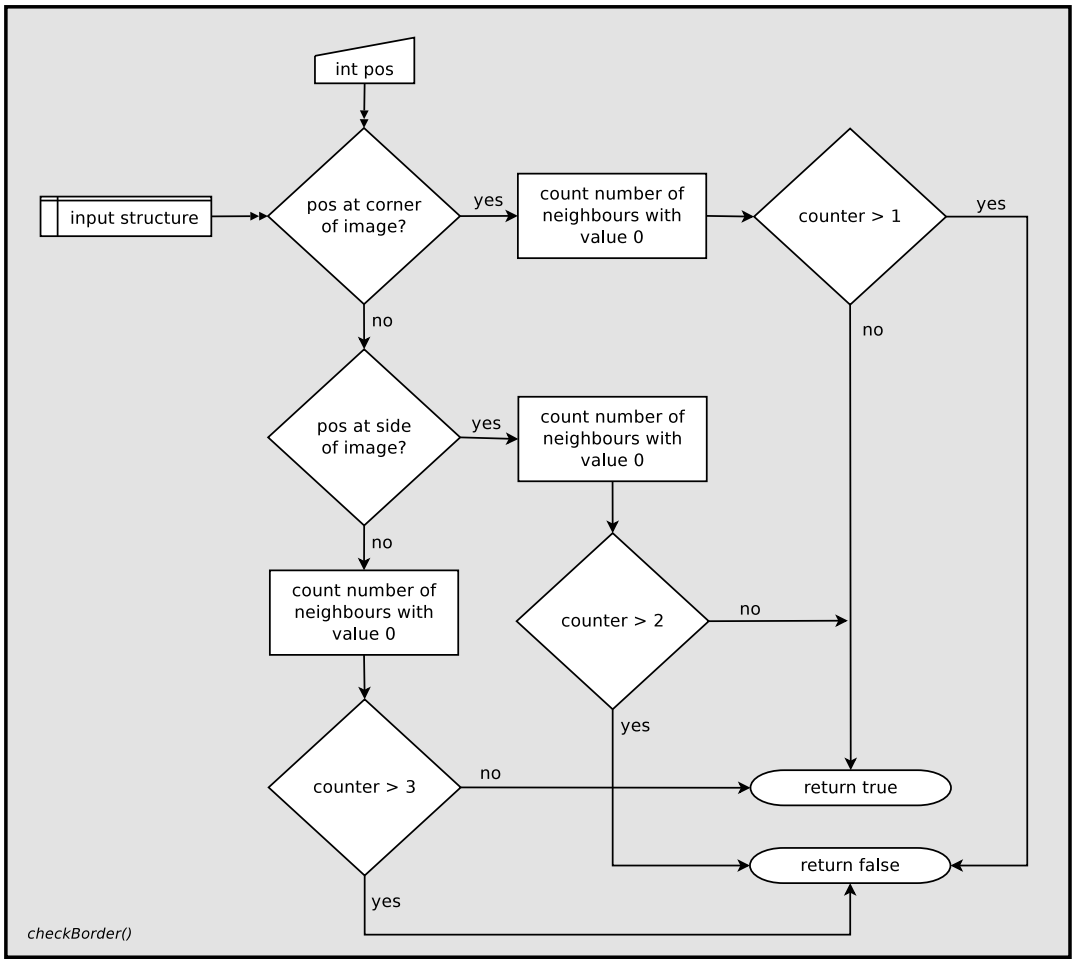


Figure 4.24: Two simple checks determine whether the pixel at *pos* is located in a corner or at a border of the image. Depending on which case applies, the number of adjacent pixels which may be cold too, varies. If this number is exceeded, the cold pixel in question will be rejected and the function returns *false*. Otherwise, it will return *true* for this pixel.

If there are enough cold pixels h_i in line i such that condition 4.10 is met, line i will be considered cold. The results will be written to the output table, the output map (see Fig. 4.27) and to standard output (usually to console, see Fig. 4.28) afterwards. In Fig. 4.26 all these steps are visualized.

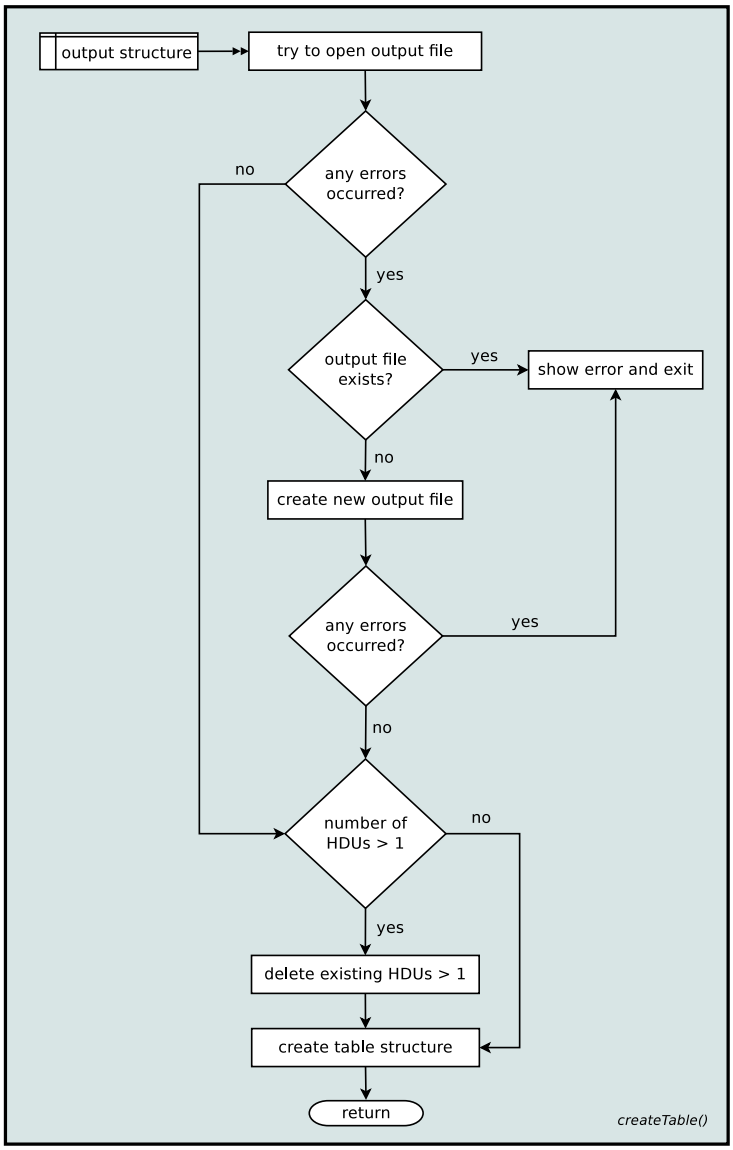


Figure 4.25: At first, *createTable* will try to open the output file. In case of errors, it will check if the file actually exists and, if so, will force an exit because of the file being corrupted likely. If it doesn't exist, a new output file with a binary table header will be created. Otherwise, all data HDUs of the existing file will be deleted and an empty table structure will be created.

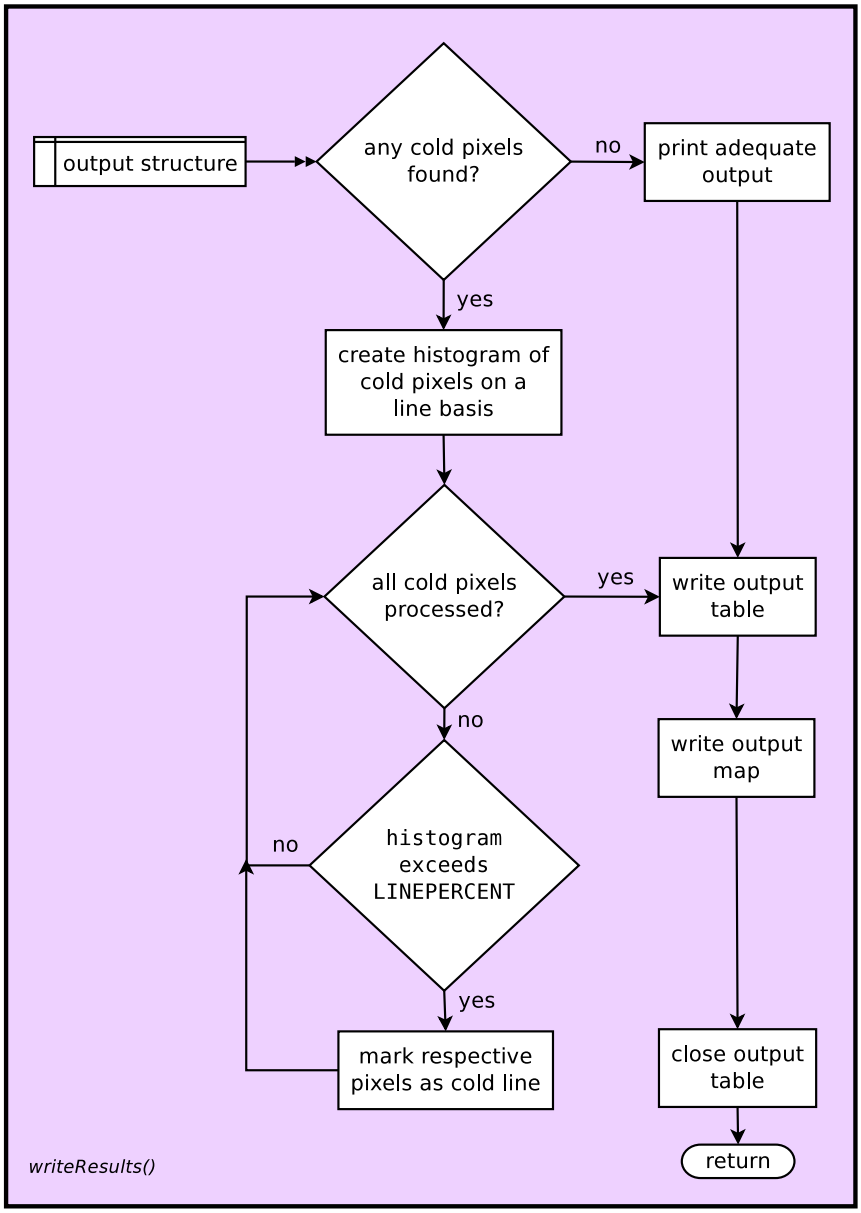


Figure 4.26: If no cold pixels/lines were found, *writeResults* will write a short notification to standard output and close the output table. Otherwise a histogram for each line will be created, showing how many cold pixels each line contains. By testing the histograms with the condition 4.10, cold lines will be identified and the respective cold pixels marked. Once having finished all checks the results (containing coordinates of cold pixels/lines, their extent and their type) will be written to standard output and to the output table.

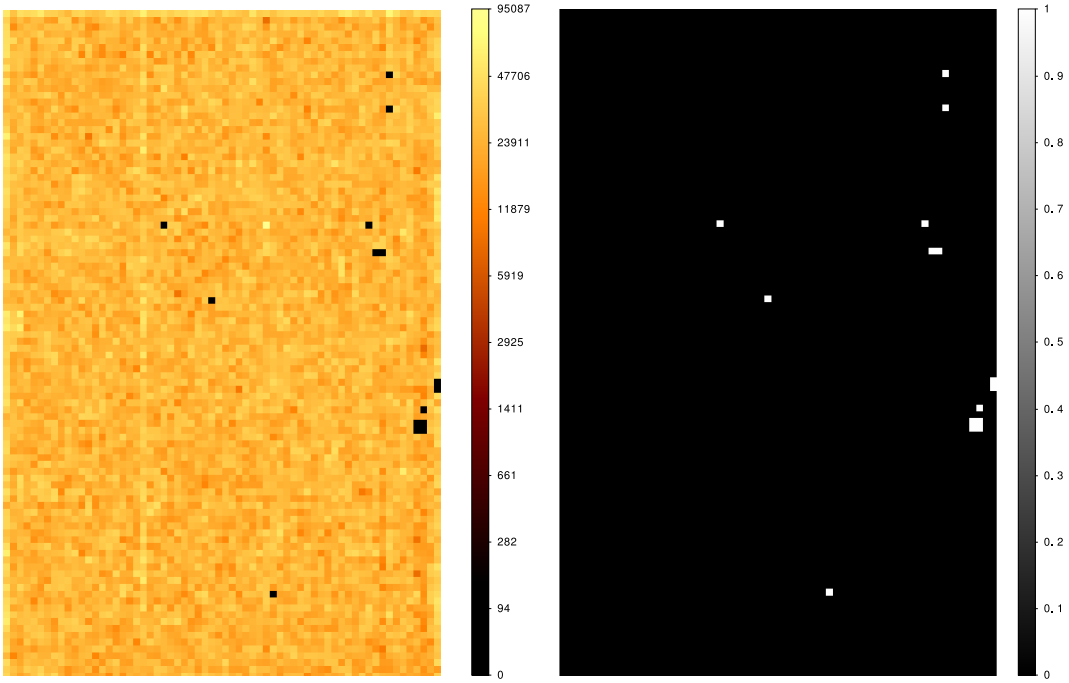


Figure 4.27: Comparison of image data taken during rev. 1235 by pn-CCD 10 with the resulting cold pixel map. The cold pixels, clearly visible as black spots on the left side, appear as white spots in the map. The lower halves of the images have been cut away as they did not contain any data. The corresponding console output of the cold pixel analysis is illustrated in Fig. 4.28.

4.9 Performance tests

In order to test the detection algorithms under realistic conditions, "real" data from the *XMM-Newton* satellite mission were used. *XMM-Newton* is equipped with 12 pn-CCDs, each having a width of 64 pixels and a height of 200 pixels. The advantage of using data from *XMM-Newton* for testing is the possibility to compare the results with an existing Current Calibration File (CCF) afterwards. Among other things, these files contain the outcome of the bad pixel detection performed by the *badpixfind* tool, which is part of the *XMM-Newton SASS*.

The data which were at hand contain several dark spots with signal 0 which can be exploited for cold pixel search. A deeper look into the corresponding CCF reveals, however, that these pixels had actually been masked out in an earlier iteration of *badpixfind*, in which they were apparently classified as defective.

There is also a black bar at the bottom of each image (see, e.g., Fig. 4.30), which measures 11 pixels in height and spans across the whole width of the chip. Due to defects, these areas were already masked out aboard the satellite and thus contain no data (M. Stuhlinger, 2010, priv. comm.). The bars do not interfere with cold pixel detection though, as the filtering mechanism described in Chap. 4.8.3 ignores them.

```
*** RESULTS ***

###   RAWX    RAWY
  1     40     115
  2     61     139
  3     62     139
  4     61     140
  5     62     140
  6     62     142
  7     64     145
  8     64     146
  9     31     158
 10     55     165
 11     56     165
 12     24     169
 13     54     169
 14     57     186
 15     57     191

numnewpix: 0
numoldpix: 0
database file will NOT be updated.
Finished. Freeing memory...
Done.
```

Figure 4.28: Final console output of *FPFindColdPixel* after the analysis of *XMM-Newton* data from rev. 1235, pn-CCD 10. Data and cold pixel map are shown in Fig. 4.27. As the output suggests, comparison with an existing cold pixel table was turned off.

It is, however, possible that the bars influence the sensitivity of the detection of bright pixels lying near their top border. This is due to image blocks being partially covered by the bar which would lead to a higher standard deviation of the block values and thus to reduced detection sensitivity. This effect is negligible for this performance test though.

The diagram in Fig. 4.29 shows the results of bad pixel search on *XMM-Newton* data with an exposure time of 40 ksec, taken during revolution 159 in Oct. 2000. Bright pixel search was performed for different thresholds $2.0 \leq \vartheta \leq 3.4$ (in steps of 0.1) and the respective results merged with those of cold pixel search. Of course, the latter had to be started only once per CCD.

On the whole, the sum of the solid and the dash-dotted curves in Fig. 4.29 reach a minimum in the area around $\vartheta \approx 2.7$ which is marked in dark gray. When increasing the threshold any further, detection seems to get too insensitive and thus leads to a miss of some bright pixels which are even part of the CCF. On the other hand, lowering ϑ may quickly cause a very steep increase in detected pixels which are not in agreement with the CCF list. The fact that the data of all CCDs react very similar to changing ϑ can be explained by the common background level of all chips: when a certain threshold is reached, most of the bright pixels cannot be detected anymore

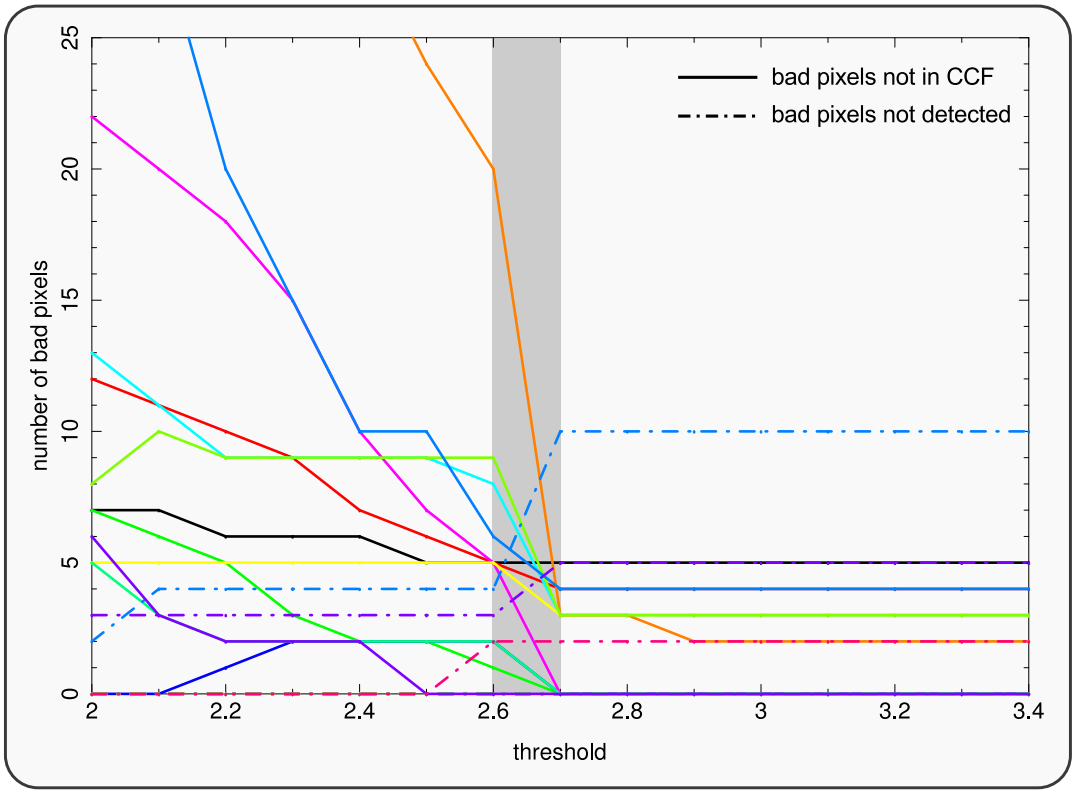


Figure 4.29: Results of the tests with data from rev. 159: Each color represents one of the 12 pn-CCDs. The outcome was compared to the "official" bad pixel list in the corresponding CCF-file. The solid lines show the number of bad pixels which were detected but not found in the CCF, while the dash-dotted lines show the amount of pixels present in the CCF but not detected by the search tools. The shaded area at $2.7 \leq \vartheta \leq 2.8$ marks the best agreement with the CCF as the sum of solid curves and dash-dotted curves reaches a minimum.

against the background which causes a joint drop of the solid curves.

Unfortunately, results of tests with the official CCF files are not absolutely accurate. This problem arises from incompatibility issues due to the fact that nearly all bad pixels which are indexed in the CCF files have already been blackened in the data. Consequently, most of the indexed bad pixels will be detected by the cold pixel analysis despite actually being catalogued as "bright" in the CCF. Although this problem is indeed quite irritating, the fact that the CCF files not necessarily contain all bad pixels which can actually be found is much worse. In Fig. 4.30 an example image taken by CCD 8 during revolution 159 is shown, which contains a great deal of bright pixels (more than 15 of them can be counted by bare eye). In the corresponding CCF, however, only 8 of them are indexed. This problem is caused by the volatile nature of many bad pixels. A comparison with data from revolution 179, which was

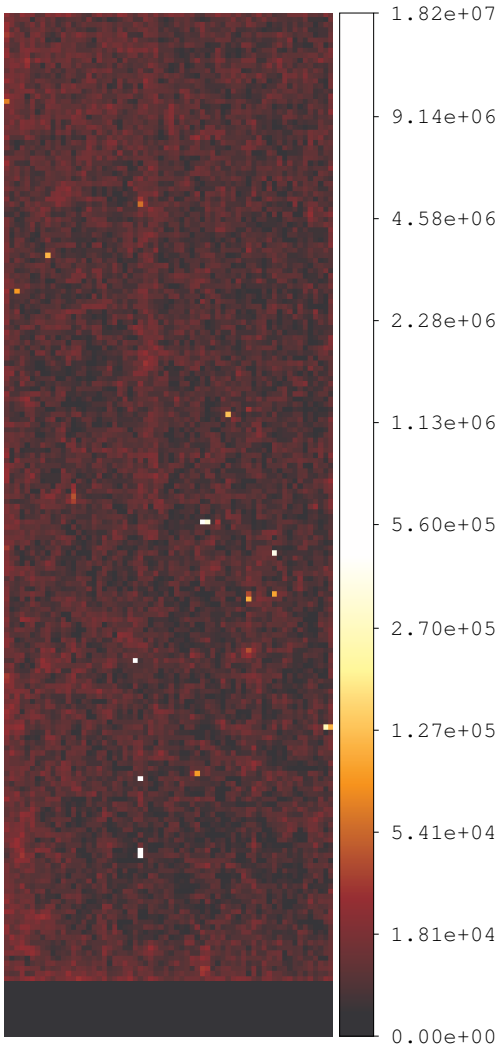


Figure 4.30: Image obtained by integrating 40 ksec of data taken by pn-CCD 8 during rev. 159 of *XMM-Newton*. A large number of bright pixels are clearly visible.

about 1 month later, showed that most of the bright pixels of revolution 159 behaved normally again. Pixels which are recorded in the CCF files are considered to be beyond repair because of having stayed bad for a long time.

This concept makes it difficult to compare the results of the bad pixel search tools with the official list. The great discrepancy for thresholds $\vartheta < 2.6$ in Fig. 4.29 can be explained with that. This implies, however, that the best fit may differ from the shaded area to a certain extent.

4.10 Data merge

The differences between the detection algorithms of bright and cold pixels made it reasonable to split the task into two separate programs with both of them producing their own output files. The main goal, however, was to obtain a single list containing information about all bad pixels which are in the data. For this reason, the output files of *FPFindHotPixel* and *FPFindColdPixel* have been kept compatible to each other, making it fairly easy to merge them into one single file.
This task is done by a small perl script called *badpixmerge.pl*, which has been implemented as part of the fpipe, but which can also be used as a standalone utility. It makes use of the FTOOL *ftmerge* which is part of the FTOOLS software package (Blackburn, 1995), being available at NASA's HEASARC website[5].

4.11 Integration into *NRTA*

The three steps of bad pixel search which were discussed in the last few chapters (bright pixel detection, cold pixel detection and a final merge of the output tables) would all have to be started manually and separately if there were no means of somehow connecting and automatising them. As shown in Chap. 4.5 the *FPIPE* offers a convenient solution. Pipeline definition files allow the user to define a "schedule" which controls the order of execution of all tools and input/output data flows. Fig. 4.31 shows how such a pipeline definition file could look like when considering the simplified case that bad pixel search is the only task of the pipeline.
Each pipeline needs a so-called pipeline server which is basically necessary for semaphore handling (other tasks like, e.g., feeding input data to subsequent tools would also be possible) and therefore has to be started in the first place. In case of the sample pipeline shown in Fig. 4.31 the program *FPPrePro*, written by C. Grossberger, is used for this purpose. It acquires an initial write lock on the semaphore `FileFeed`, which is actually not a real file but just a virtual handle in order to control the execution order of programs which have dependencies between each other. Not till then the underlying pipeline server command framework will start *FPFindHotPixel* and *FPFindColdPixel* as both of them require the virtual infile `FileFeed` and do not interfere with each other because they have different virtual write locks on `HotPix` and `ColdPix`. The merging script *badpixmerge* will be started in the last place consequently, needing all of the virtual output files of preceding tools.
With this concept, the initial intention of finding a versatile, modular way to perform first quick-look analyses of satellite data without requiring much user interaction has been achieved. Furthermore the definition of multiple pipelines with appropriate tools can account for the various areas of responsibility of the NRTA, as discussed in Chapter 4.3.

[5] *http://heasarc.gsfc.nasa.gov/docs/software/ftools/ftools_menu.html* (Sep. 2010)

```
<pipelinedefinition>
  <NRTA>
    <FPPrePro>
      <fpipeserverstart>FPPrePro semid=##SEMID## wlocks=##WLOCKS,/## or=##O_R,/##</fpipeserverstart>
      <infile sema="false" source="false" />
      <outfile sema="true" temporary="true">FileFeed</outfile>
    </FPPrePro>

    <FPFindHotPixel>
      <status>0</status>
      <single>FPFindHotPixel startval=0 numthreads=2 infile=/scratch1/wille/fits/0159/0159_0112980201_PNS00301IME.FIT
      outtable=/scratch1/wille/fits/hotpixout.fits badpixdb=/scratch1/wille/fits/hotpixdb.fits energycol="ENERGY"
      threshold="2.3" detwidth=64 detheight=200 extname=pnime1 updatedb=y</single>
      <infile sema="true" source="false">FileFeed</infile>
      <outfile sema="true" temporary="true">HotPix</outfile>
    </FPFindHotPixel>

    <FPFindColdPixel>
      <status>0</status>
      <single>FPFindColdPixel infile=/scratch1/wille/fits/0159/0159_0112980201_PNS00301IME.FIT
      outfile=/scratch1/wille/fits/coldpixout.fits outmap=/scratch1/wille/fits/coldpixout_map.fits
      badpixdb=/scratch1/wille/fits/coldpixdb.fits phacol="ENERGY" detwidth=64 detheight=200 updatedb=y</single>
      <infile sema="true" source="false">FileFeed</infile>
      <outfile sema="true" temporary="true">ColdPix</outfile>
    </FPFindColdPixel>

    <badpixmerge>
      <status>0</status>
      <single>badpixmerge.pl /scratch1/wille/fits/hotpixout.fits /scratch1/wille/fits/coldpixout.fits
      /scratch1/wille/fits/badpixout.fits</single>
      <infile sema="true" source="false">FileFeed</infile>
      <infile sema="true" source="false">HotPix</infile>
      <infile sema="true" source="false">ColdPix</infile>
      <outfile sema="false"></outfile>
      <ntimes>1</ntimes>
    </badpixmerge>
  </NRTA>
</pipelinedefinition>
```

Figure 4.31: A sample pipeline definition which would trigger an automated search for bad pixels and bad lines (first bright, then cold) and merge the output tables to a result table in the end.

5 Simulations

Ever ongoing technological advance allows us to build satellites with better sensitivity and even more sophisticated detectors. However, this progress is accompanied by an increase in complexity and thus in error rate. In order to deal with this challenge, it is advantageous to perform as much testing as possible and hopefully recognise potential problems and design faults prior to satellite launch. Unfortunately, not every single aspect can be simulated, such as for example the final PSF of the optics. After launch simulations can also provide information which are essential for the understanding of the detector, e.g., in terms of pile-up.

In terms of mirror and detector quality assurance, extensive simulations with specialised software can be performed nowadays. Theoretically one can feed the whole mirror and detector structure into the simulation software, given that there is enough processing power available. Afterwards it is possible to visualise the signal which the detector would actually receive from a source by tracking paths and incidence of randomly generated photons. It makes sense though, to constrain simulations to aspects which are physically reasonable and make simplifying assumptions where applicable. Even in the mirror design phase, the outcome of simulations can already be very useful, e.g., for checking the designated PSF and minimising vignetting effects (obscuration or mutation of image parts by telescope components).

5.1 SIXT Software

In the course of his diploma thesis, C. Schmid has created an advanced simulation software called *Simulation of X-ray Telescopes (SIXT)*, which "provides several models to simulate the imaging and detection process of individual photons in Wolter telescopes with different kinds of detectors" (Schmid et al., 2010). Having initially been developed with particular regard on simulations for eROSITA, the *SIXT* software has evolved to a more general simulation framework which can be used with many different detector types and geometries. Due to its modular design, each component of the *SIXT* software can easily be readjusted to match different environments.

Fig. 5.1 shows an overview of the different simulation steps, starting with an X-ray source list, which specifies spectra, positions and brightnesses of the sources to be simulated. After having provided additional information about effective area and attitude of the satellite, a photon generation module creates a list of photons which were "emitted" by the virtual sources. This photon list is processed by another module, simulating which of these photons were detected with given PSF, vignetting and attitude. The resulting list of photons which hit the detector, in turn, is reprocessed by a third module which accounts for effects produced by the nature of the detector, such as its response function or bad pixels.

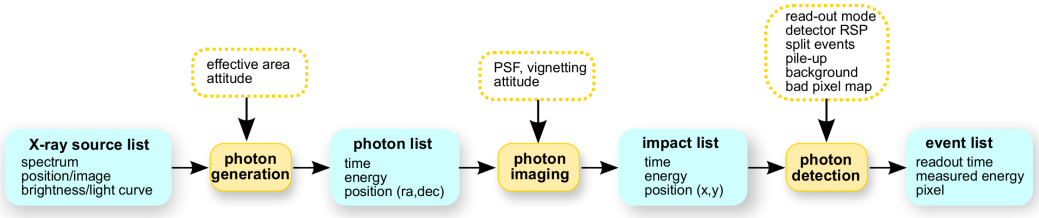


Figure 5.1: Scheme of the individual steps of the *SIXT* pipeline (Figure: C. Schmid, 2011).

5.2 Detector background simulation

Detector background, mostly caused by cosmic rays and corresponding scattering events (see Chap. 5.2.1), unfortunately is an integral part of X-ray images and has non-negligible impact on data quality. The signal of faint sources, which could easily be spotted if there were no background events in data, may quickly get indistinguishable from background noise in real data or in simulations which take it into account.

Simulation of detector background has recently become a feature of the *SIXT* software (see Fig. 5.1, step 3) in the form of an additional library called *erodetbkgrndgen*. Its development and implementation into the existing framework were performed in the course of this thesis.

5.2.1 Issue

Cosmic rays are not really "rays" but actually ionised particles (primarily protons) with energies partially higher than 10^{21} eV (Gaisser, 1990). Most of them, however, barely exceed 10^{12} eV. In total, Earth's atmosphere is hit by about 1000 cosmic rays per square meter per second. Consequently X-ray satellites are subject to cosmic ray radiation to its full extent.

Aside from their generally harmful effects on electrical components, cosmic rays might, e.g., reach the CCD chip of a detector and contribute to background noise. Therefore, satellites are equipped with shielding which is capable of screening the detector from most of the primary particles. Interactions with the shielding material will cause manifold secondary radiation such as electrons and photons though. These can hit the detector and add to the background signal. Hits of the detector plane by primary particles may occur particularly if energy and direction of the cosmic ray are such that it can be deflected by the grazing incidence optics of Wolter-telescopes and thus get through to the focal plane directly.

While travelling through the CCD bulk, cosmic rays generate showers of secondary particles along their track, with the major part of them being electrons. Consequently, the signal appears in the form of long and short, randomly distributed streaks (see Fig. 5.2). Length and brightness of the streaks depend both on particle energy and incidence angle.

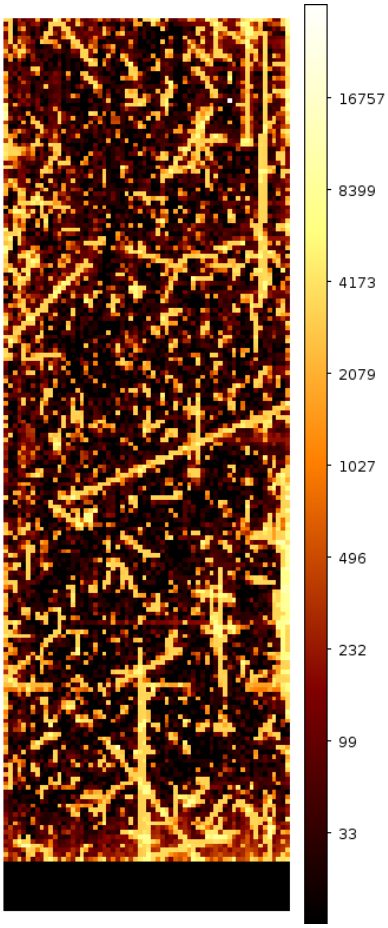


Figure 5.2: picture taken by *XMM-Newton* EPIC-pn, showing extensive pollution with cosmic rays (clearly visible as bright streaks with variable lengths and arbitrary directions).

The simulation, which yielded Fig. 5.6, made use of background event data provided by C. Tenzer et al. from IAAT. They collected sample data of the expected detector particle background for the eROSITA detector. The data were obtained by performing simulations with the *Geant4* software, which is a toolkit for the simulation of the passage of particles through matter (Tenzer et al., 2010), over a period of 1309 sec.

5.2.2 Photon statistics

Background simulations require three basic information in order to work as expected:

- how many background events n_i occurred during time slice t_i?
- at which position (x_j, y_j, z_j) with respect to the detector did event j occur?

- which energy E_j did event j have?

Energy and position of the events can be obtained from the sample background event file. The number of background events has to be calculated with photon statistics, though. According to Berry & Burnell (2009), the distribution of the mean number of photon events per time slice follows Poisson statistics and thus the probability P_i to get n_i photons in time slice i, with $\bar{n}_i$ the mean, is:

$$P_i\left(n_i\right) = \frac{\exp\left(-\bar{n}_i\right)\bar{n}_i^{n_i}}{n_i!} \tag{5.1}$$

The library *erodetbkgrndgen* was designed to take these issues into account and therefore is capable of performing simulations compliant to photon statistics.

5.2.3 *erodetbkgrndgen*

Like all other components of the *SIXT* software, the background simulation is implemented as a modular library called *erodetbkgrndgen*. Before being able to generate particle background events for arbitrary time intervals, the function *eroBkgInitialize* (see Fig. 5.3) must be invoked once. It opens the sample file, initialises data structures and allocates memory needed by the following operations.

Fig. 5.5 illustrates the functioning of the main routine *eroBkgGetBackgroundList*, which has to be called in order to get a list of background events returned. The desired time interval (the unit depends on the preset of the sample file), for which background events should be generated, must be specified in the function call.

The number of background events to "occur" during a specific time interval of the simulation, depends on the mean event rate of the sample file and on the length of the interval. Based on the Poisson probability density (Eq. 5.1), the function *gsl_ran_poisson* generates event numbers for given mean values, i.e., if invoked a lot of times, the function's return values will resemble a Poisson distribution.

Afterwards, the number of events is chosen randomly from the sample file (following a simple flat random distribution this time) and added to a background event list, which will be returned to the calling function in the end. If any chosen event is part of a bunch of secondary events, the whole corresponding event shower will be incorporated to the background list, as, in principle, all of its events belong to the same primary particle.

When the list of background events is completed, it will be returned to the calling function and *eroBkgGetBackgroundList* will exit. Data of the sample background file and the initialised random number generator will be still in memory though, ready to process further requests for background lists from the ongoing simulation. If *erodetbkgrndgen* should be shut down entirely, e.g., when the whole simulation is done, the function *eroBkgCleanUp* should be called once, which will take care of de-allocating memory and closing the input file.

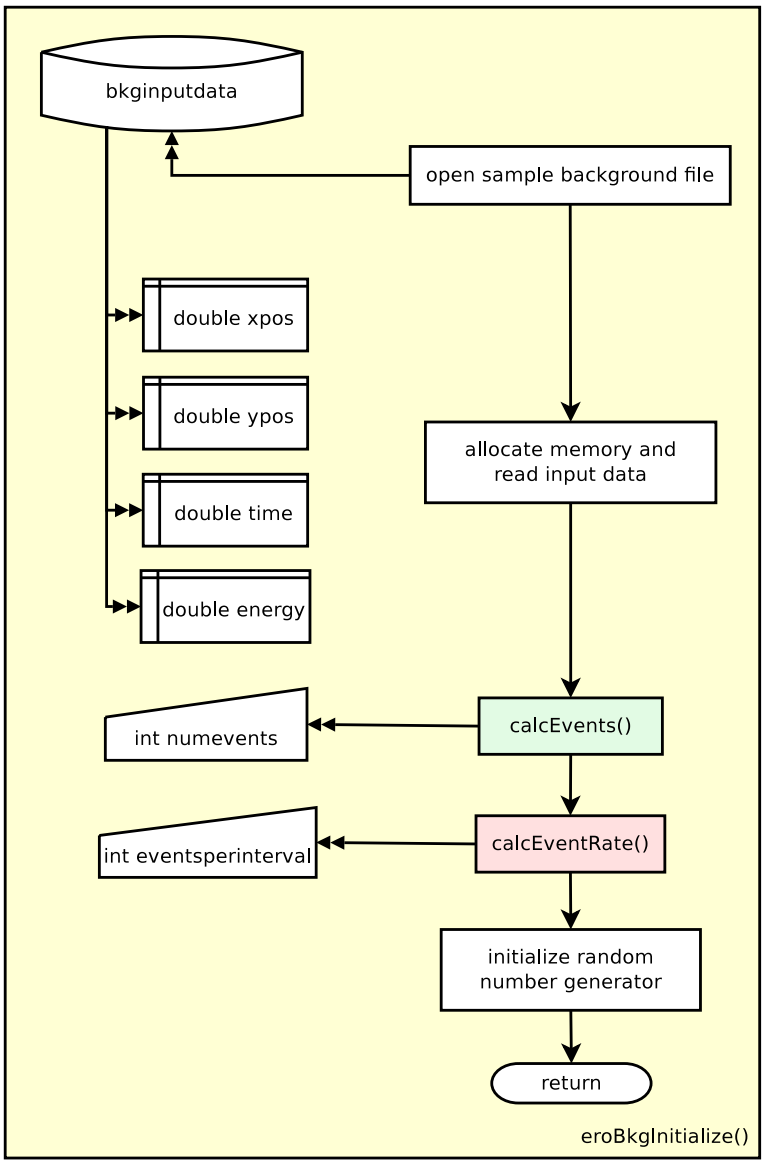


Figure 5.3: The sample background event file is opened, some of its data extracted and stored in memory. After that, function calls to *calcEvents* and *calcEventRate* determine total number and mean rate of background events. In a final step, the random number generator is initialised, involving several calculations with system time on a millisecond basis, in order to generate a seed as unique as possible.

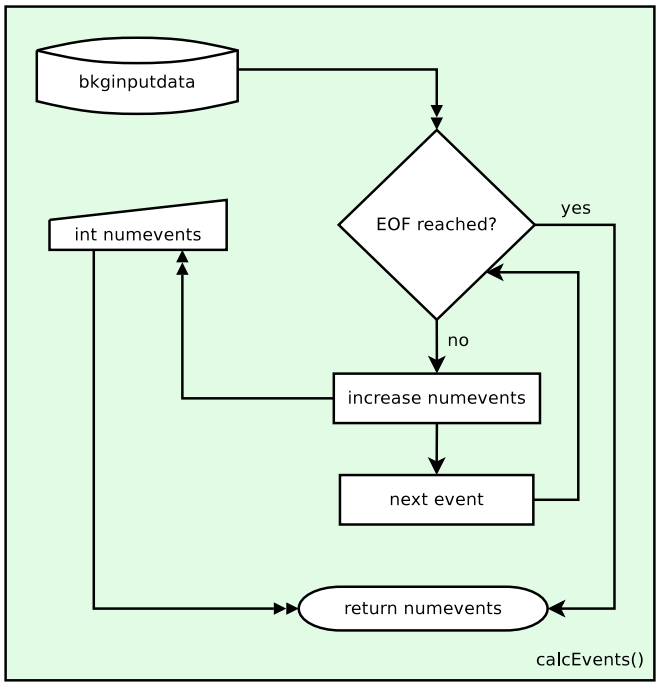


Figure 5.4: *calcEvents* simply counts the total number of events in the sample background file.

5.2.4 Results

Several tests, performed by C. Schmid, have shown that the background event simulation module seems to work as expected. The results of a simulation of a point source with an exposure time of 1000 sec can be seen in Fig. 5.6, clearly showing a random distribution of background events all over the detector plane. The detector model which was used for this particular simulation resembles the structure of the eROSITA detector.
Unfortunately, the data of the sample file arose from a background simulation of the very short timespan of 1309 sec only. As a consequence, many background events in the resulting 1000 sec simulation in Fig. 5.6 originate from the same entries in the sample file and were just incorporated multiple times, due to a lack of data. In order to achieve a more realistic background, it is mandatory to improve the amount of sample data significantly.

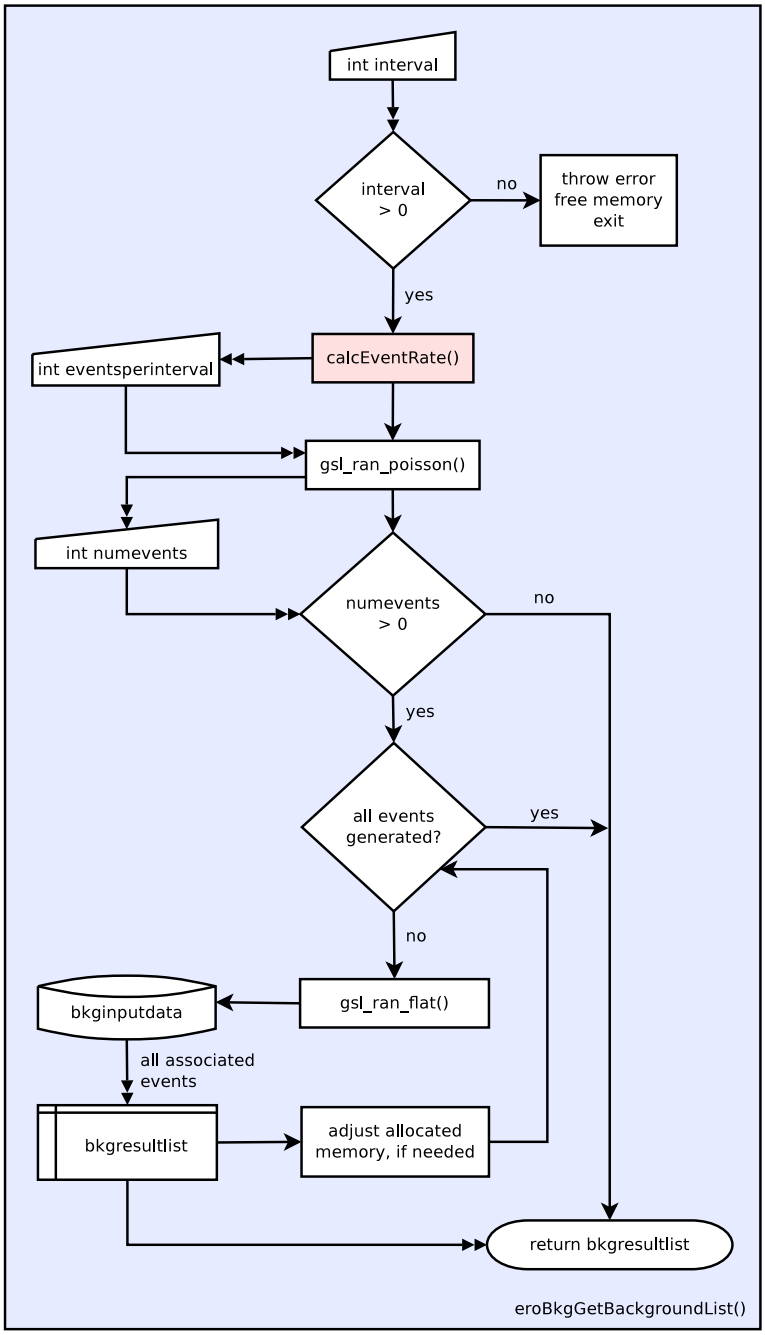


Figure 5.5: Having been passed a valid, non-zero time interval for background event generation, the function *calcEventRate* is called, which detects the mean number of events one can expect for the given time interval and sample background file. Afterwards, the number of events to be generated is determined by Poisson statistics. A corresponding number of random events are read from the sample background file and returned as the resulting background list afterwards.

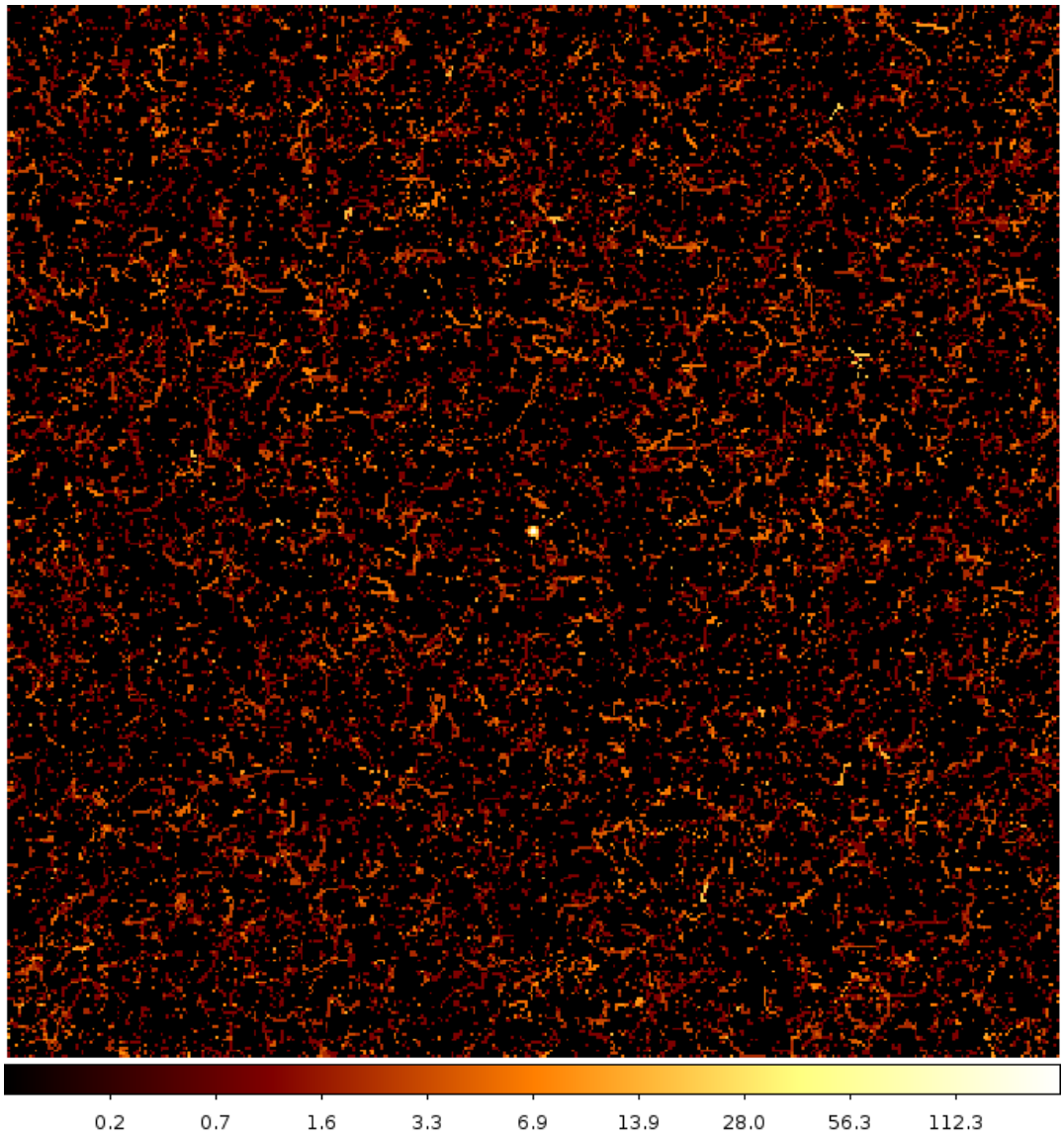


Figure 5.6: *SIXT*-simulation of a 1000 sec exposure of a point-source (in the centre) with detector background included. The strip-like trails of cosmic rays are clearly visible.

6 Conclusion and Outlook

Despite its sometimes irritating peculiarities, CCD technology has become an integral part not only of astronomy but of everyday life long ago. The intention of this thesis was to give a peek into the very common pn-CCD subspecies with eROSITA being one of its next areas of application in X-ray astronomy. Particular emphasis was set on the development of a toolset which is capable of providing valuable information on data sanity and detector health in terms of bad pixels. In their current state both tools produce results which are quite satisfactory and allow seamless integration into the FPIPE framework. Software, however, is never perfect[6]. Even if it were possible that a program became absolutely foolproof after endless hours of testing and code reviewing, many new bugs would be introduced soon with the next feature expansion. Software development is an iterative process with the implementation of new features in the first place, followed by testing and bug hunting before the whole cycle restarts. From this point of view the bad pixel search tools have reached the end of such an iteration now, with all important features being implemented and working.

Of course there are a couple of things that would be nice to have such as, e.g., an interactive graphical user interface which sums up the results of bad pixel search and allows the user to reject obvious false positives by a simple click with his mouse. During development of the bad pixel tools I was often asked by colleagues if the software was also capable of recognizing bad pixels in static images. This is unfortunately not the case as the primary objective was to achieve reasonable results with event lists as input files. Additionally, the analysis of static images requires a different approach because of the data volume being smaller than in the case of an event list and due to a complete loss of temporal information. For example Ghosh et al. (2008) offer an advanced algorithm to deal with this problem. In the very beginning of my research work I led myself get carried away to try to adapt their solution but soon had to realise that it would not fit my needs.

Another interesting topic which does not stop coming up every now and then is the integration of bad pixel search into the *SASS*. Despite having been developed with the purpose of performing first quick-look analyses (which is completely sufficient in terms of the NRTA) both bad pixel tools could be improved in such a way that their capabilities meet the requirements of the SASS, given that there is enough time. Tests with realistic data sets and close teamwork with SASS developers of the MPE will show if this project would be worthwhile.

In terms of testing a great deal of work still has to be done. The extensive parameter set of bright pixel search allows the recognition algorithm to be adapted to different detectors and conditions of operation (e.g., background levels, survey mode, etc.). Up to now a significant number of tests were made only with data from the *XMM-Newton* satellite. Despite the similarities of its detector chips with eROSITAs pn-CCDs, the results of the tests most likely can only be used to get a rough estimation for the performance of bad pixel search on eROSITA data. Several other tests were done

[6] TEX(version 3.1415926 since March 2008) is no exception to this rule as bug hunting is still ongoing

with data from the TRoPIC camera, which is a dedicated testing camera used by the MPE for on-ground calibration (Freyberg et al., 2008). Its CCDs resemble those which will be used in the eROSITA detector. The outcome was promising but due to a lack of "natural" defects, bad pixels had to be injected into the data artificially. In any case the final adjustment of the bad pixel search tools will have to wait until the satellite is launched in 2013.
Concerning the background simulation library for *SIXT*, tests with the available background event data indicated that the event selection algorithm works as expected. As already mentioned in Chapter 5.2.4 the background event list must be extended considerably in order to permit simulations longer than 1 ksec without data suffering from similar background events. This is useful especially when simulating deep surveys or pointed observations which tend to have very long exposure times. With the help of its new background simulation capabilities the *SIXT* software will be able to perform even more sophisticated simulations of observations. These data are, among other things, valuable for designers and engineers of new X-ray optics and detectors to find technical constraints which can satisfy scientific expectations later on. Functionality to incorporate simulated pixel defects into the data was implemented into the *SIXT* software recently (C. Schmid, 2011, priv. comm.). For this reason *SIXT* could also provide useful data for testing the bad pixel detection tools.
I agree with Ghosh et al. (2008) saying in their very first sentence that "detection of defective pixels in solid-state detectors/sensor arrays has received limited research attention". During my own research I was surprised by the difficulty of finding information on existing bad pixel detection methods. Considering the popularity of CCD detectors and their extensive usage in science, more efforts should be made to find applicable ways of dealing with bad pixels and filtering them from data. Perhaps this work can be a modest contribution to this objective.

Acknowledgements

In this final chapter I would like to thank all the people who supported me during the long time of writing this thesis. It was not an easy task but I gained interesting insights into a lot of different facets of astro- and solid state physics, satellite operations and programming.
First of all I want to express my gratitude to my advisor, Jörn Wilms, whose vivid, informative lectures on astronomy and astrophysics encouraged me to stay with this intriguing subject. His ability to grasp the core of a problem with a single glance and give helpful advice straight away never ceased to amaze me. Despite being often snowed under with work he always takes his time when there are questions.
Especially in the early phase of my research work and during the development of the bad pixel software, Christoph Großberger provided me with valuable information about *FPIPE* and *CFITSIO* and how to deal with them from a programmer's point of view. Furthermore his advice concerning the bad pixel detection algorithm and the various discussions we had about the structuring of the *NRTA* and the usage of *FTOOLS* were very helpful for my work. Finally he did not hesitate to proofread part of my thesis even during the Easter holidays which I appreciate very much. I also want to thank Ingo Kreykenbohm who shared his experience on other satellite missions and his extensive knowledge about scientific data processing with us.
Apart from maintaining a pleasant working atmosphere, my likable office colleagues also supplied me with manifold support. Whenever I had a question concerning LaTeX, Felix Fürst had an appropriate answer at hand. Moreover, due to his dedication to astrophotography we took beautiful snapshots of the night sky and I learned very much about the astronomical equipment and the telescopes of the observatory. I also want to thank Christian Schmid who, in addition to Felix, proofread part of my thesis and provided helpful feedback. Natalie Hell, being always game for anything, not only made the office day even more enjoyable but also contributed valuable tips to my thesis and proofread it entirely. During the time of her stay in the United States I will miss her defeating me continually in Kinect bowling and in our joint movie sessions.
Basically I have to express my gratitude to the whole Remeis gang. All of them have done their bit to turn the observatory into a place lovely to work and even spend part of one's spare time in. Especially the crazy guys of the "Knigge Room" deserve an honorable mention due to their apparently inexhaustible source of entertaining projects and bad (yet mostly creative) jokes. Due to serendipitous happenstance I became part of this pleasant community and am looking forward to staying for another couple of years while working on my PhD.
Last but not least I want to thank my family for their continued confidence and support (also from a financial point of view) on my long way to graduation. They always had a sympathetic ear for my problems, helped me wherever they could and showed patience when we did not see each other over a longer period, especially in the final phase of my diploma thesis. Without them, I would not have made it this far.

References

Agarwal B., 2009, Basic Statistics, New Age International, New Delhi

Aschenbach B., 2009, Experimental Astronomy 1-3, 95

Begelman M.C., Blandford R.D., Rees M.J., 1984, Reviews of Modern Physics 56, 255

Berry R., Burnell J., 2009, The Handbook of Astronomical Image Processing, William-Bell Inc.

Blackburn J.K., 1995, In: R. A. Shaw, H. E. Payne, & J. J. E. Hayes (ed.) Astronomical Data Analysis Software and Systems IV, Vol. 77. Astronomical Society of the Pacific Conference Series, p. 367

Cappelluti N., Predehl P., Böhringer H., et al., 2011, Memorie della Societa Astronomica Italiana Supplementi 17, 159

Charles P.A., Seward F.D., 1995, Exploring the X-ray Universe, Cambridge University Press

Curran S., Craggs J., 1949, Counting tubes: theory and applications, Laboratory technique monograph, Academic Press

Danner R., 1993, Experimental Astronomy 4, 105

Forman W., Jones C., Cominsky L., et al., 1978, The Astrophysical Journal Supplement Series 38, 357

Freyberg M.J., Budau B., Burkert W., et al., 2008, In: Turner M.J.L., Flanagan K.A. (eds.) Space Telescopes and Instrumentation 2008: Ultraviolet to Gamma Ray, Vol. 7011. Proc. SPIE, SPIE, Bellingham, WA, p. 701117

Friedman H., Lichtman S.W., Byram E.T., 1951, Physical Review 83, 1025

Fuerst F., 2008, Investigations of the SAA and the long-time behavior of Vela X-1, Friedrich-Alexander-Universität Erlangen-Nürnberg & ECAP

Fürmetz M., Pfeffermann E., Predehl P., et al., 2008, In: Turner M.J.L., Flanagan K.A. (eds.) Space Telescopes and Instrumentation 2008: Ultraviolet to Gamma Ray, Vol. 7011. Proc. SPIE, SPIE, Bellingham, WA, p. 70113Y

Fürmetz M., Predehl P., Eder J., Tiedemann L., 2010, In: Arnaud M., Murray S.S., Takahashi T. (eds.) Space Telescopes and Instrumentation 2010: Ultraviolet to Gamma Ray, Vol. 7732. Proc. SPIE, SPIE, Bellingham, WA, p. 77323K

Gaisser T.K., 1990, Cosmic rays and particle physics, Cambridge University Press

Ghosh S., Froebrich D., Freitas A., 2008, Applied Optics 47, 6904

Giacconi R., Gursky H., Paolini F.R., Rossi B.B., 1962, Physical Review Letters 9, 439

Gibb M., 2009, NASA's HEASARC: Observatories, http://heasarc.gsfc.nasa.gov/docs/heasarc/missions/past.html

Greiner J., Friedrich P., Liebscher D., et al., 2001, In: H. Sawaya-Lacoste (ed.) Space Debris, Vol. 473. ESA Special Publication, p.415

Hill R.S., Landsman W.B., Lindler D., Shaw R., 1997, In: S. Casertano, R. Jedrzejewski, T. Keyes, & M. Stevens (ed.) The 1997 HST Calibration Workshop with a New Generation of Instruments., p. 120

Jackson J., 1999, Classical electrodynamics, Wiley, New York

Jahoda K., Swank J.H., Giles A.B., et al., 1996, In: O. H. Siegmund & M. A. Gummin (ed.) UV, X-Ray, and Gamma-Ray Instrumentation for Astronomy VII, Vol. 2808. Proc. SPIE, SPIE, Bellingham, WA, p.59

Janesick J., 2001, Scientific charge-coupled devices, SPIE Press monograph, SPIE Press

Karttunen H., 2003, Fundamental astronomy, Physics and Astronomy Online Library, Springer, Berlin

Knoll G.F., 2000, Radiation Detection and Measurement, Wiley, New York

Komatsu E., Smith K., Dunkley J., et al., 2010, The Astrophysical Journal Supplement Series 192, 18

Lutz G., 1999, Semiconductor Radiation Detectors, Springer, Berlin

Meidinger N., Andritschke R., Ebermayer S., et al., 2009, In: Siegmund O.H. (ed.) UV, X-Ray, and Gamma-Ray Space Instrumentation for Astronomy XVI, Vol. 7435. Proc. SPIE, SPIE, Bellingham, WA

Mitsuda K., Bautz M., Inoue H., et al., 2007, Publications of the Astronomical Society of Japan 59, 1

Müller C., Kadler M., Ojha R., et al., 2010, In: 38th COSPAR Scientific Assembly. COSPAR Conf.Proc.38, Bremen, Germany, p. 2310

Pence W., 2009, CFITSIO User's Reference Guide, http://heasarc.gsfc.nasa.gov/FTP/software/fitsio/c/cfitsio.pdf

Pence W.D., 1992, In: D. M. Worrall, C. Biemesderfer, & J. Barnes (ed.) Astronomical Data Analysis Software and Systems I, Vol. 25. Astronomical Society of the Pacific Conference Series, p. 22

Pence W.D., Chiappetti L., Page C.G., et al., 2010, Astronomy and Astrophysics 524, A42

Predehl P., Andritschke R., Bornemann W., et al., 2007, In: Siegmund O.H.W. (ed.) UV, X-Ray, and Gamma-Ray Space Instrumentation for Astronomy XV, Vol. 6686. Proc. SPIE, SPIE, Bellingham, WA, p. 668617

Schmid C., Martin M., Wilms J., et al., 2010, In: Comastri, A. and Angelini, L. and Cappi, M. (ed.) X-ray astronomy 2009; Present status, multi-wavelength approach and future perspectives, Vol. 1248. AIP Conference Proceedings, p.591

Schwarm F.W., 2010, Cyclotron resonant scattering features in the line forming region of highly magnetized neutron stars, Friedrich-Alexander-Universität Erlangen-Nürnberg & ECAP

Schwarzburg S., 2005, Eine Software zur Echtzeitanalyse von experimentellen Daten im Flexible Image Transport System (FITS), http://www.xubuntix.org/media/paper/DiplomarbeitStefanSchwarzburg.pdf

Takahashi T., Abe K., Endo M., et al., 2007, Publications of the Astronomical Society of Japan 59, 35

Tenzer C., Warth G., Kendziorra E., Santangelo A., 2010, In: Holland A.D., Dorn D.A. (eds.) High Energy, Optical, and Infrared Detectors for Astronomy IV, Vol. 7742. Proc. SPIE, p. 77420Y

Wilms J., Kreykenbohm I., Schmid C., 2009, Near Real Time Data Analysis Software Design, eRO-RSB-DD-63-01_2

Wolter H., 1952, Annalen der Physik 445, 94

Erklärung

Hiermit erkläre ich, dass ich die Diplomarbeit selbständig angefertigt und keine anderen als die angegebenen Hilfsmittel verwendet habe.

Bamberg, 25. April 2011

_____

(Michael Wille)