ISSN 1213-2365

Streak Detection in Astronomical Images

Michal Šustr

sustrmic@fel.cvut.cz

CTU–CMP–2013–11

May 22, 2013

Available at
ftp://cmp.felk.cvut.cz/pub/cmp/articles/navara/Sustr-TR-2013-11.pdf

Thesis Advisor: Mirko Navara , Radim Šára

Research Reports of CMP, Czech Technical University in Prague, No. 11, 2013

Published by

Center for Machine Perception, Department of Cybernetics
Faculty of Electrical Engineering, Czech Technical University
Technická 2, 166 27 Prague 6, Czech Republic
fax +420 2 2435 7385, phone +420 2 2435 7637, www: http://cmp.felk.cvut.cz

České vysoké učení technické v Praze
Fakulta elektrotechnická

katedra řídicí techniky

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student: **Michal Šustr**

Studijní program: Kybernetika a robotika
Obor: Systémy a řízení

Název tématu: **Detekce pohyblivých objektů v astronomických snímcích**

Pokyny pro vypracování:

Seznamte se se základními metodami zpracování obrazové informace. Na reálných snímcích z astronomických dalekohledů vyzkoušejte metody detekce pohyblivých objektů (družice, kosmické smetí, meteory). Dle možnosti implementujte rovněž odhad drah pozorovaných objektů (kruhových nebo eliptických, podle toho, co umožní získaná data). Posuďte použitelnost metody, její citlivost, rušivé vlivy apod.

Seznam odborné literatury:

[1] Sonka, M., Hlavác, V., Boyle, R.: Image Processing, Analysis and Machine Vision. Thomson, 2007.
[2] Schlesinger, M.I., Hlavác, V.: Deset prednásek z teorie statistického a strukturního rozpoznávání. CVUT, Praha, 1999.

Vedoucí: Prof.Ing. Mirko Navara, DrSc.

Platnost zadání: do konce zimního semestru 2013/2014

[illegible]

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
Fakulta elektrotechnická
L.S.
4

[illegible]

prof. Ing. Michael Šebek, DrSc.
vedoucí katedry

prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 20. 12. 2012

Abstrakt

Táto práca popisuje metódy detekcie objektov v obrázkoch z optických ďalekohľadov, ktoré sa zobrazujú ako čiary. Detekcia je vykonaná pomocou vlastného algoritmu a výsledky sú porovnané so známym súčasným technickým riešením. Dáta použité na testovanie pochádzajú z rôznych observatórií s rôznymi spôsobmi zachytávania obrazu a sú použité aj simulované dáta. Keďže súčasné riešenie predpokladá určité vlastnosti čiar, metódy sú porovnané na obrázkoch, kde by mali byť úspešné oba algoritmy. Pre porovnanie rozdielnosti metód budú tiež použité na obrázkoch, pre ktoré súčasné technické riešenie nie je navrhnuté.

Abstract

This thesis describes methods of detection of objects in optical telescope images that manifest themselves as streaks. The detection is done with a custom algorithm and the results are compared with the known state of the art. The data used for testing comes from various observatories with different modes of capture, and simulated data is used as well. As the current state of the art presumes certain qualities of the streaks, methods are compared on images where they should both perform well. To contrast the methods, they are also applied to images for which the state of the art is not designed for.

Acknowledgement

I would like to thank to my thesis advisors Radim Šára and Mirko Navara and to my family and friends for their support.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze, dne 23. 5. 2013

Contents

1 Introduction **1**

2 Detection algorithms **7**

2.1 Assumptions about the data 7

2.2 The proposed algorithm 7

2.2.1 Structure of the algorithm 10

2.2.2 Image shearing 12

2.2.3 Selecting optimal angle 16

2.3 The YanKuro algorithm 18

3 Data **20**

3.1 Input images 20

3.2 Annotation of data 21

3.3 Simulated streaks 22

4 Testing of the algorithms **27**

4.1 Threshold amplitude 29

4.2 Classification of images 30

4.3 Detection of streaks 31

4.4 Time complexity 33

5 Conclusions **35**

Chapter 1

Introduction

Images from optical telescopes contain much information that can be processed automatically by computers. The objects that can be found in the images are various: Stars, galaxies, nebulae, material from our solar system, meteors, various atmospheric processes, passing aeroplanes or satellites and much much more. A collection of defunct objects in orbit around Earth called orbital debris (also known as cosmic debris, space junk or space waste) is of principal interest in this thesis.

Orbital debris includes everything from remnants after rocket carriers, old satellites, fragments from disintegration, erosion, lost equipment to other objects. Since their orbits overlap with orbits of operational spacecraft, a collision can happen. The first major space debris collision was on February 10th, 2009, at 16:56 UTC. The deactivated 950 kg Kosmos 2251 and an operational 560 kg Iridium 33 collided 800 km over northern Siberia [5]. The relative speed of impact was about 11.7 km/s, or approximately 42,120 km/h [8]. Both satellites were destroyed and the collision scattered considerable debris. Numerous collisions have occurred in the history and operational satellites have to actively avoid debris for their safety. Debris can also land on Earth. In the year 2000, remnants of an American rocket landed in farm area close to Cape Town in the Republic of South Africa [7]. To this date, there was only one reported incident of a human being struck by space debris: In 1997, a woman in Tulsa, Oklahoma, was hit on the shoulder by a 6-inch piece of metal, and fortunately, it did not lead to any serious injury [1].

The great majority of debris consists of smaller objects, 1 cm or less, which are difficult to detect. Their magnitudes vary around ~ 16 mag and lower. Since space debris comes from man-made objects, the total possible mass of debris is easy to calculate: It does not exceed the total mass of all spacecraft and rocket bodies that have reached the orbit. The actual mass of debris will be necessarily less than that, as the orbits of some of these objects have since decayed. As debris mass tends to be dominated by larger objects, most of which have long ago been detected, the total mass has remained relatively constant in spite of the addition of many smaller objects. Using the figure of 8,500 known debris items from 2008, the total mass is estimated at 5500 tons [9]. The most suitable time for their capture for observers on Earth is after sunset or before sunrise, when the sunlight is reflected the most from their surface.

According to the NASA debris FAQ, the number of large debris items over 10 cm is 19000, between 1 and 10 cm approximately 500,000, and debris items smaller than 1 cm exceed tens of millions [2].

Based on the distance from the surface of the Earth most of the debris can be separated into two major categories: Low Earth Orbit (LEO) and Geostationary Earth Orbit (GEO) debris. LEO is generally defined as an orbit below an altitude of approximately 2,000 kilometers. It is the most concentrated area for orbital debris. GEO is a circular orbit 35,786 kilometres above the Earth's equator. A visualisation of debris density is in Figure 1.1.

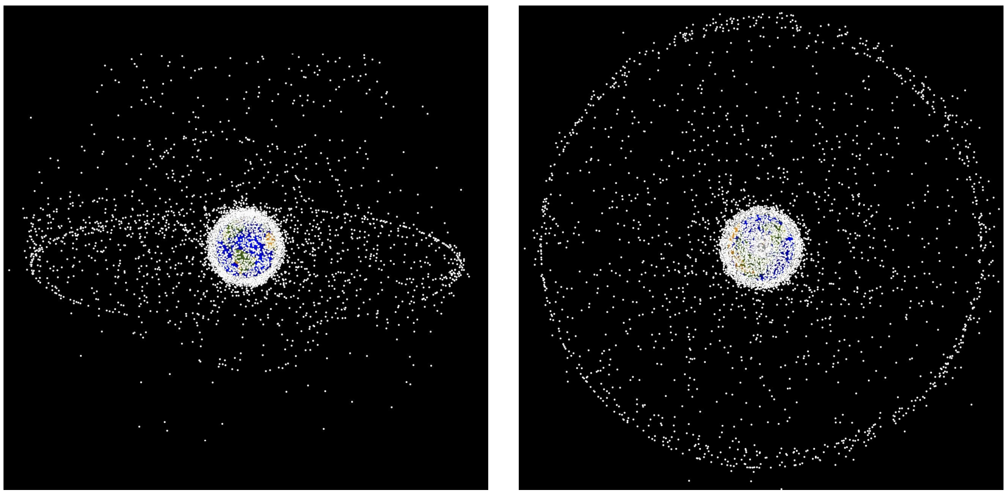
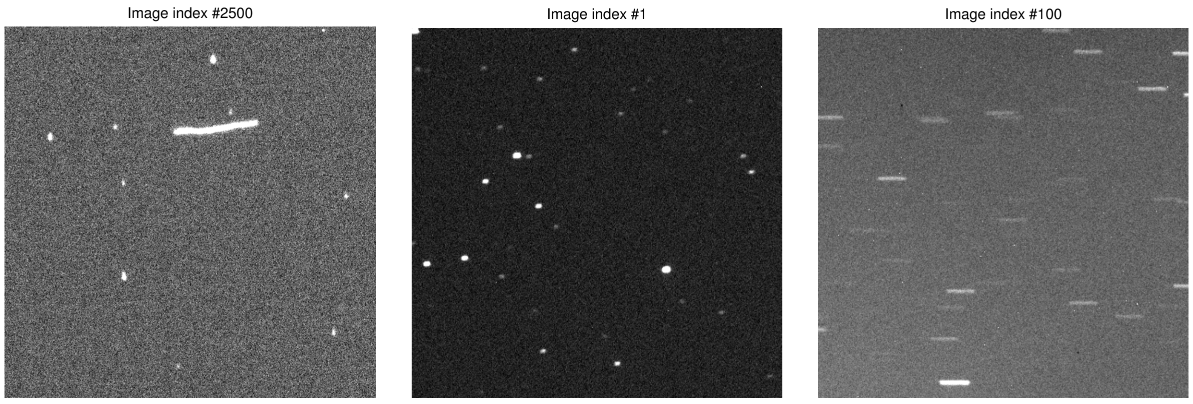

(a) Side view (b) Top view

Figure 1.1: Visualisation of the density of the orbital debris [3].

Prediction and modelling of the trajectories of debris is difficult and is of concern of various organizations such as USSTRATCOM's mission Space Surveillance Network (SSN)[1] or Europe Space Agency (ESA)[2] and subject of major conferences as European Conference on Space Debris (ECSD) organized by European Space Operations Centre (ESOC)[3] or Advanced Maui Optical and Space Surveillance Technologies Conference (AMOS)[4]. Under good observation conditions LEO objects can be detected in images from optical telescopes. They show as streaks when length is proportional to their angular speed, as they reflect light from the Sun. With efficient streak detection algorithms, faint debris streaks can be found and identified much faster.

Recognition of objects in the images is computationally time-consuming because of the size of the images, which usually starts at the resolution of 2000 $\times$ 2000 pixels. The pictures are captured by CCD cameras. There are two most common modes of capture of images: With or without the compensation for sidereal movement of the sky, as in

[1]http://www.stratcom.mil/factsheets/USSTRATCOM_Space_Control_and_Space_Surveillance/
[2]http://www.esa.int
[3]http://www.esa.int/About_Us/ESOC
[4]http://www.amostech.com



(a) Example of a streak (b) Compensation for the sidereal movement (c) No compensation

Figure 1.2: Example of a streak and two modes of capture of images.

Figures 1.2(b) and 1.2(c) respectively. We will be searching for streaks, objects that are segment-like without exceeding curvature and longer than the motion blur of the surrounding stars (sometimes we will call them star streaks). One example is in Figure 1.2(a). Both modes of capture may contain streaks. The streaks may also represent other objects than orbital debris, such as meteors, passing aeroplanes, atmospheric processes, etc., but the identification, trajectory or property computation of the streak is not the aim of this thesis.

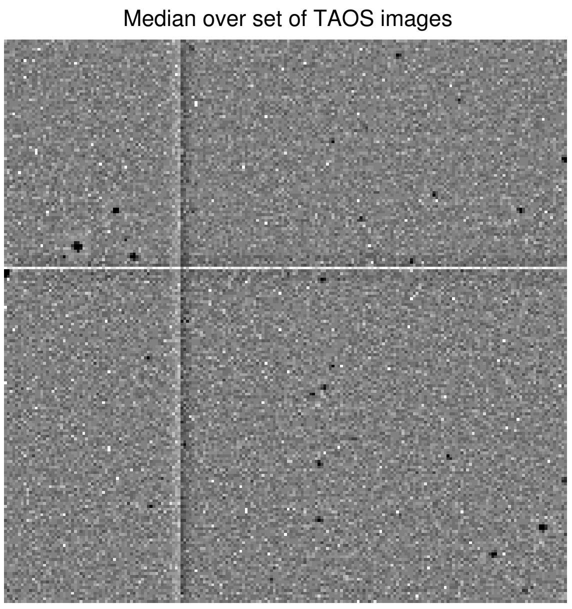
The images used for detection of the streaks are quite varied and they have different defects: Additive light reflected of clouds, multiplicative character of light transmission through clouds, cloud motion, internal light scattering in the telescope, cosmic particles, hot pixels and dark pixels (Figure 1.3(a)), nonuniform sensitivity (Figure 1.3(b)), dirt on sensors (Figure 1.3(c)), camera damage (Figure 1.3(d)), image noise and possibly others. These phenomena should not have a great impact on the performance of a good detector.

Some images are captured by CCD cameras that are composed of multiple segments and the median intensity level is different for each segment, which causes appearance of vertical/horizontal lines. An example of such defects is in Figure 1.3(a).

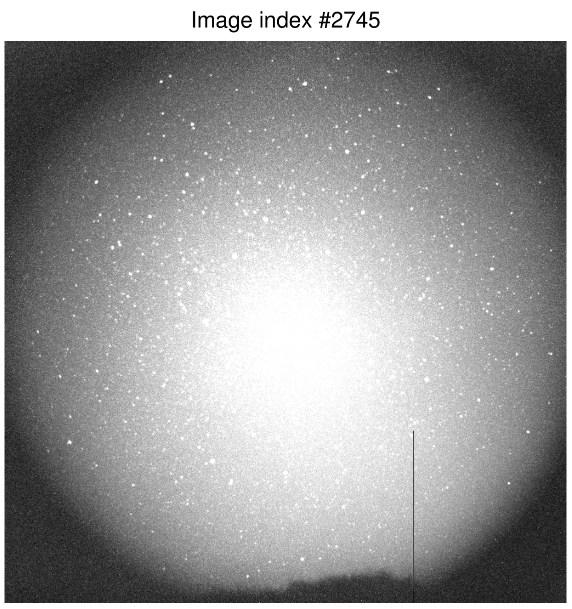
Some telescopes used a mode of capture without compensation for the sidereal movement (as in Figure 1.2(c)), which results in stars looking like small segments that can be interchanged for streaks. If a streak is very short and is in the same direction as this distortion it is difficult to recognize.

There exist numerous methods of detection of space debris using optical, radar or laser techniques. They have been discussed at the conference ECSD. A book of abstracts of these papers can be found in [4].

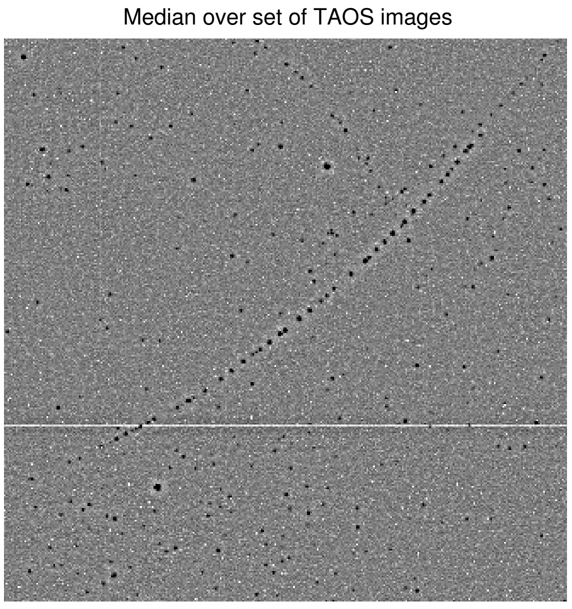
A standard method for detection of debris works similarly to this: It takes two time-consecutive images, subtracts them and rotates this difference in angles from 0 to 180 degrees [14]. This rotates the candidate for the streak into a vertical position. Because



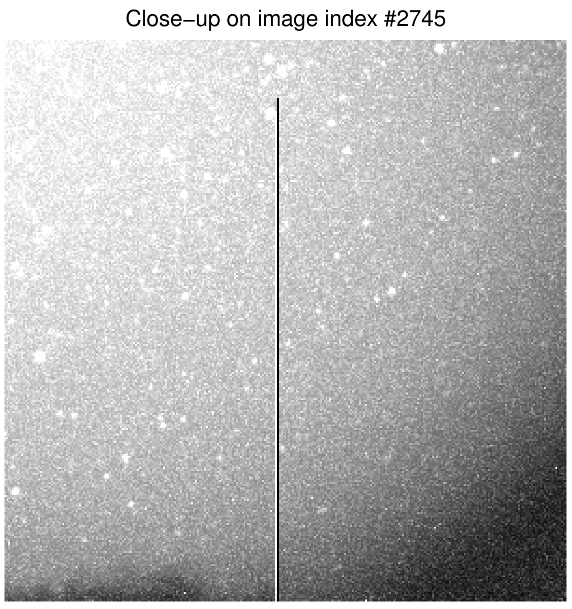
(a) Hot and dark pixels, tile boundaries



(b) Nonuniform sensitivity

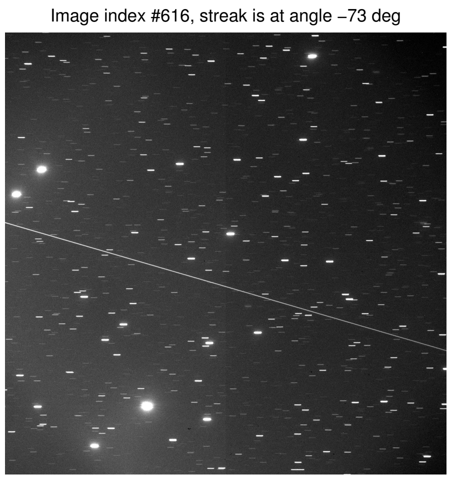


(c) Dirt on the sensor

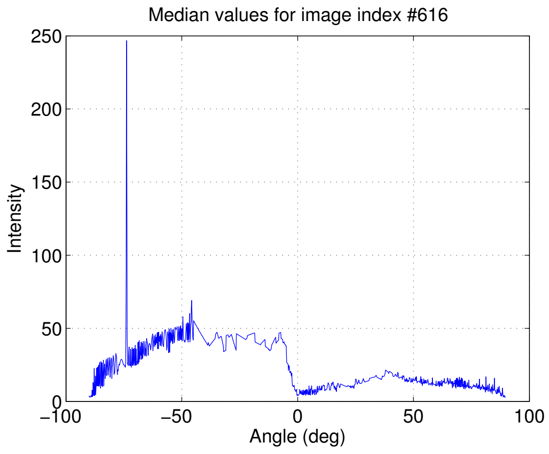


(d) Damage of the CCD sensor

Figure 1.3: Defects that can be found on the source images.



(a)



(b)

Figure 1.4: The source image is displayed on the left. On the right is a plot of maximal median values of intensities for each angle. The candidate for the streak is at the angle of -73°.

of the use of difference, only the streak should remain in the image and it should be very outstanding compared to the background, which in ideal case should be zero. In the direction where the median of column values in the rotated image is the largest we find the streak. An example can be seen in Figure 1.4.

The use of the image difference makes the algorithm robust with regard to corruptions of the CCD camera or other stationary defects. The detection is reliable for strong and long streaks. On the other hand, it has the following disadvantages:

- It assumes that in the two images there is no other change besides the streak.
- The positions of the stars and other static objects have to be the same.
- The exposition takes twice as long.
- Image rotation is computationally expensive (time-consuming).
- The streak has to be strong/long enough to be captured by the median.
- It doesn't detect where the streak begins and ends, it only gives equation of a line where the streak is located.

Implementation is simple, but the algorithm is very time-consuming. FPGA technology can be and was used for improving performance [13].

In this thesis two detection algorithms are going to be described and tested. "The CMP Algorithm" by Sara and Sustr [10] and the "YanKuro Algorithm" by Yanagisawa et al. [14], which uses the standard method.

The CMP Algorithm tries to avoid disadvantages of the standard method. It can find weak streaks in a single image, it is robust to atmospheric or other disturbances, it is adaptive (input can be a single image or the difference of two time-consecutive images, with or without sidereal movement). Although its current implementation is still too slow to be deployed, it can use parallelization to improve performance on multi-threaded CPUs or on hardware that supports massive parallelization, like the mentioned FPGA.

The purpose of this thesis is to implement and compare these algorithms on the acquired set of data in Matlab. As the current state of the art presumes certain qualities of the streaks, the methods are compared on images where they should both perform well. To contrast the methods, they are also applied on images for which the state of the art is not designed for. I have created parts of the proposed algorithm, implemented the image viewing, annotation and detection mechanism and compared the results.

Chapter 2

Detection algorithms

2.1 Assumptions about the data

There are certain assumptions about the data. If they are not fulfilled the algorithms might give unexpected results:

- There is at most one streak in the image. This assumption is reasonable: The average density of debris is 10 debris/hour/spherical degree [11], the acquisition of images is in interval of $\sim$10 seconds, the size of the field of view is $1.73° \times 1.77°$ (for source TAOS, see Table 3.1 on page 21), so the density of streaks per image is $\sim 0.084 \ll 1$.
- The mean intensity profile of the streak can be well approximated by a Gaussian distribution and its standard deviation σ is approximately known. The streak in Figure 2.1 shows that this is a justifiable assumption.
- A streak of angle ϕ with respect to star streaks of length d is significantly longer than $d \cos \phi$. A typical value for d is $\sim$ 32px for TAOS data (see Table 3.1 on page 21) with exposure time six seconds long.
- Streak is not saturated in the image (in other words, it does not have a flat profile at the maximal value).

2.2 The proposed algorithm

The principle of the algorithm is to rotate the image so that the streak becomes vertical as in the standard method mentioned in Section 1. It then solves a set of independent optimization problems, one per rotated image column. In each column the problem is to find a minimum-energy contiguous interval of unknown beginning and length. Since we do not know the angle at which the streak is located, the image must be rotated by all angles in the range of $\langle -90°, 90°)$.

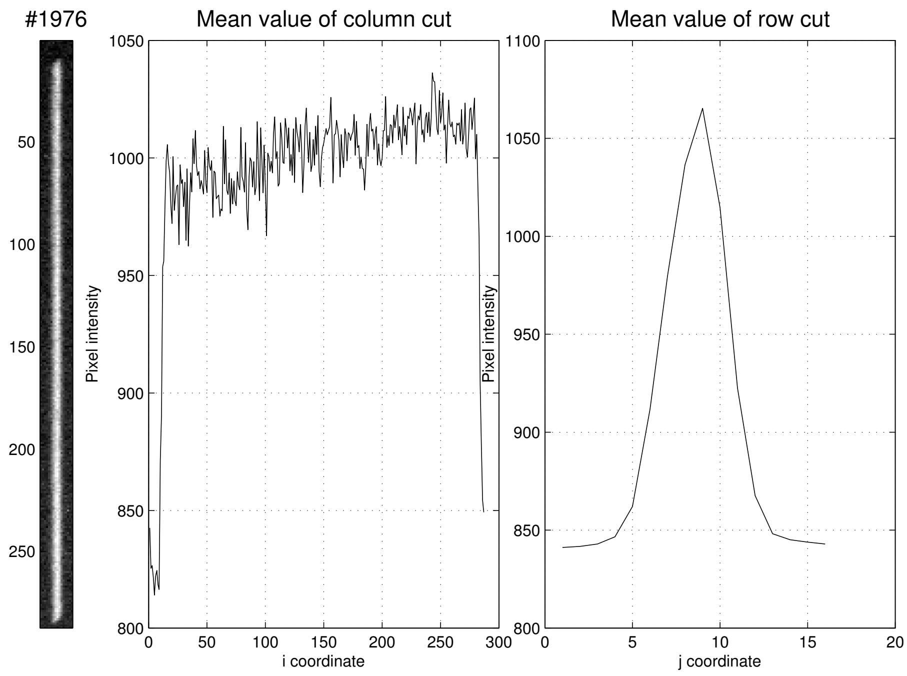


Figure 2.1: Cut-out of a streak from image. The mean value of a row cut approximates normal distribution with $\sigma^2 = 10.1$.

The energy function is constructed from the log-likelihood ratio for two classes: *background* and *streak*. The respective probability density functions (p.d.f.) are constructed over a local image feature called streakiness, not over the input image itself, like for instance in [6]. The streakiness map is computed by the normalized cross-correlation with a local vertical model called streaklet. Streaklet's kernel is visualized in Figure 2.2. Streakiness is a normalized feature, its range is $\langle -1; 1 \rangle$. The advantage of this approach over deconvolution methods is its simplicity, fast implementation, stability under a high level of noise present in optical telescope images, and the suppression of many local artifacts discussed in Section 1. An example of streakiness map is shown in Figure 2.3. Details on streakiness computation can be found in [10].

The background probability density function $p_B(x \mid \phi; \theta_B(\phi))$ is obtained by fitting a beta-distribution mixture model to the streakiness map [10]. The x is a scalar streakiness value computed for image rotation angle ϕ. The $\theta_B(\phi)$ are the parameters to be fitted (they are the beta distribution parameters α, β, and the mixture coefficients), as detailed in [10]. Typical probability density functions are shown in Figure 2.4. We can see that it is necessary to condition the distribution on the rotation angle. The dependence on the rotation angle is given by the fact that stars appear as streaks oriented at $\phi = 90°$ in the input image (cf. Figure 2.3(a)).

The streak probability density function $p_S(x \mid \phi; \theta_S(\phi))$ is obtained by marginalization from a posterior distribution constructed from the background distribution and a streak model [10]. Note that both the background and streak probability density functions are computed for each rotation angle ϕ. Given the background and streak probability density functions, one defines the log-likelihood ratio as

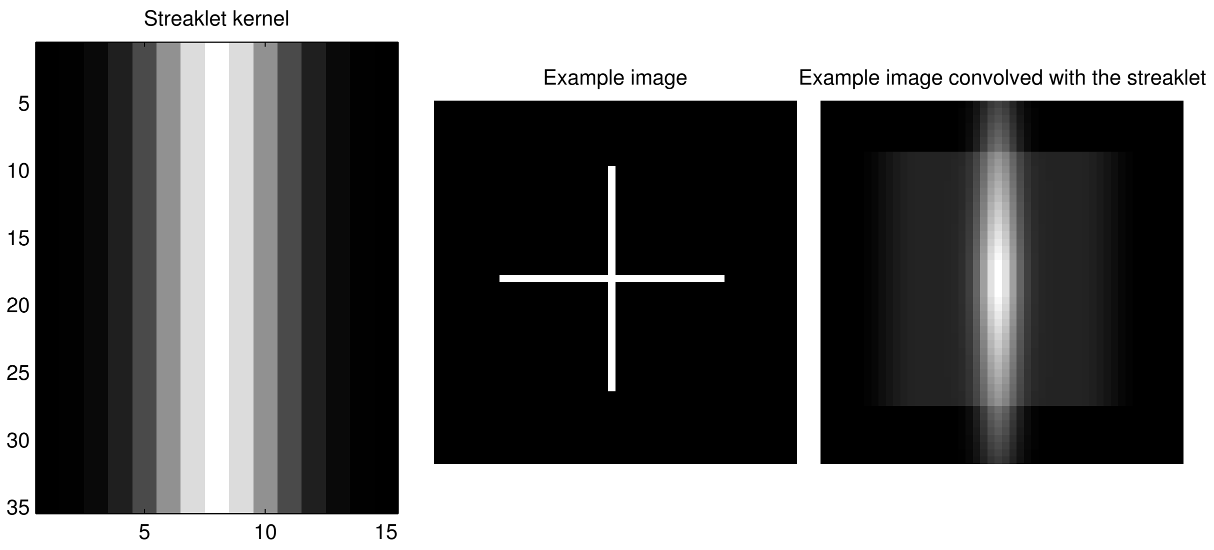
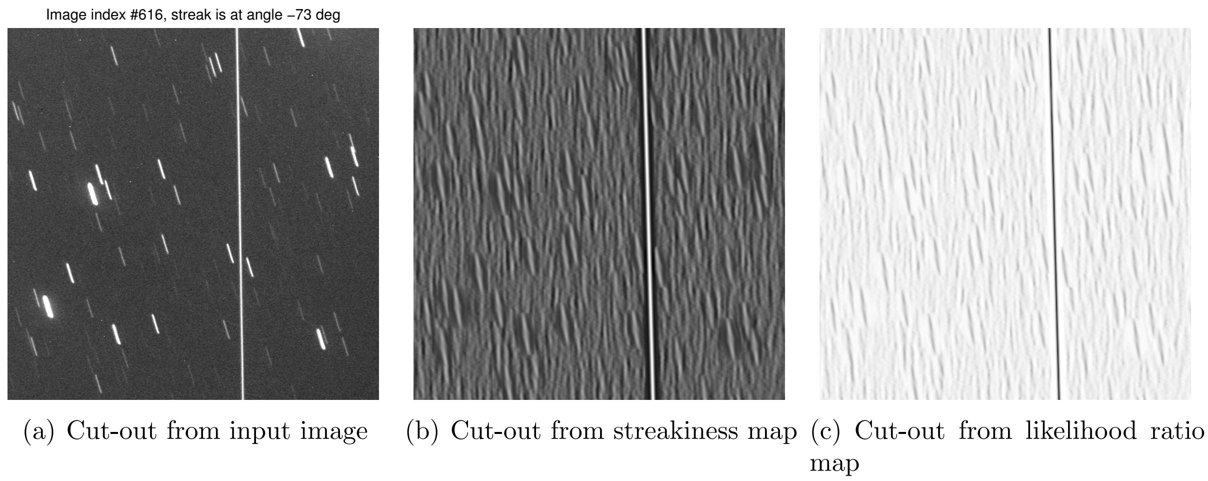
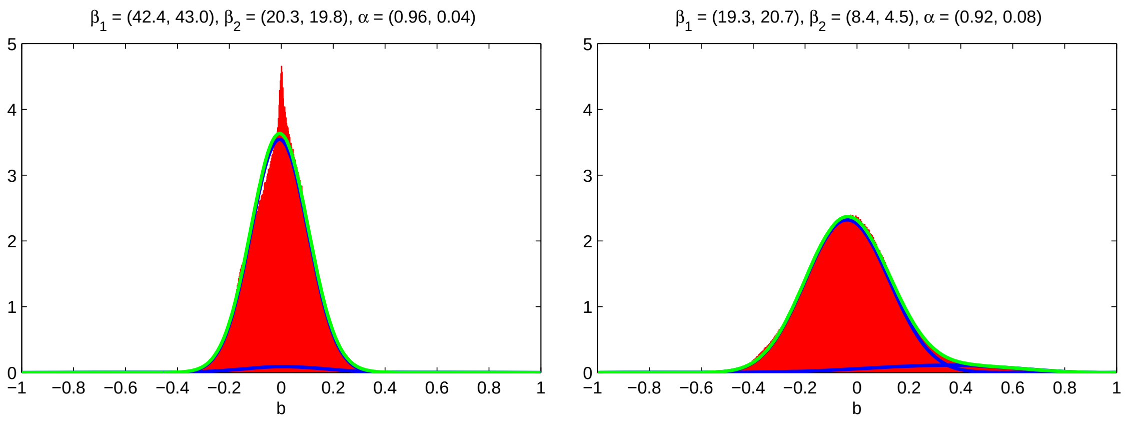


Figure 2.2: The streaklet kernel (left) and its normalized correlation response on a synthetic image (center, right).



(a) Cut-out from input image (b) Cut-out from streakiness map (c) Cut-out from likelihood ratio map

Figure 2.3: An example of a streakiness map and the corresponding likelihood ratio map after image shearig by the proper angle. The input image is displayed in Figure 1.4.



(a) $\phi = 0°$ (b) $\phi = 85°$

Figure 2.4: Typical histogram (red) and the fitted probability distribution $p_S(x \mid \phi; \theta_S(\phi)$ (green). The individual mixture components are in blue. The fitted mixture parameters are given in plot titles.

$$V(x \mid \phi;\, \theta_S(\phi), \theta_B(\phi)) = -\log \frac{p_S(x \mid \phi;\, \theta_S(\phi))}{p_B(x \mid \phi;\, \theta_B(\phi))}\,. \tag{2.1}$$

This function is computed per pixel of the rotated image. It will be called the *likelihood ratio map* in the subsequent text. From now on we omit the parameters and will write the likelihood ratio map value as $V(x(i,j) \mid \phi)$ where $x(i,j)$ is a sample of the random variable x at pixel (i,j). An example of the function $V(x \mid \phi)$ is shown in Figure 2.5.

Given the rotation angle ϕ and the corresponding likelihood ratio map $V(x(i,j) \mid \phi)$ the task is to find the minimum of

$$V(b,e,j,\phi) = \sum_{i=b}^{e} V(x(i,j) \mid \phi)\,, \tag{2.2}$$

over the angles $\phi \in \langle -90°, 90°)$, column indices $j \in I_n$, and the row index interval $(b,e) \in I_m^2$, $b \le e$, in which $I_n = \{1, 2, \ldots, n\}$ is the set of likelihood ratio map columns, and $I_m = \{1, 2, \ldots, m\}$ is the set of likelihood ratio map rows. This problem can be solved very fast, in $O(mn)$ time, by employing its column-wise independence and row-wise recursive properties, with the help of column-wise cumulative sums and cumulative maxima. A detailed description is given in [10].

2.2.1 Structure of the algorithm

The actual algorithm does not cycle over all rotation angles ϕ with a uniform step. If the diagonal length of an image was d then the required angular step would be

$$\Delta\phi \approx \frac{180}{\pi\, d}. \tag{2.3}$$

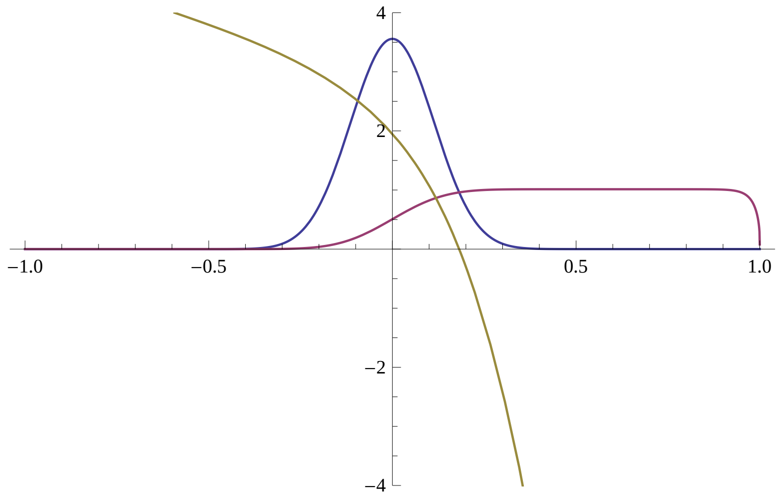


Figure 2.5: The likelihood ratio function $V(x \mid \phi)$ (brownish), together with the $p_S(x \mid \phi)$ (reddish) and $p_B(x \mid \phi)$(bluish). The abscissa of the plots is x.

For a typical 2000×2000 image this gives $\Delta\phi \approx 0.0203°$ requiring almost 9000 image rotations. Image rotation is computationally quite expensive. In Matlab on the CMP Grid (see Table 4.1 on page 29 for more information about the CPU), image rotation of this size (with bicubic interpolation) takes about 5.88 sec, which means the exhaustive search would need 14.7 hours to complete, even if the streak detection was not taken into account. As a result, such algorithm would be too slow. We wanted to avoid the brute-force approach of [13] that used an FPGA board to compute image rotations.

The proposed detection algorithm proceeds in three phases. The entire flow of the algorithm is outlined in Figure 2.7.

In Phase 1, the angular step $\Delta_{\phi 1}$ is set so that the image rotates by at most a single pixel under the streaklet kernel of height m_s

$$\Delta_{\phi 1} = \frac{2 \cdot 180}{\pi\, m_s}, \tag{2.4}$$

where we use $m_s = 40$, giving $\Delta_{\phi 1} \approx 2.9°$, hence 63 initial rotations. In Phase 1 the image is rotated, the streakiness map is computed, the p.d.f. model is fitted, and the minimization problem solved per each rotation. The typical profile of

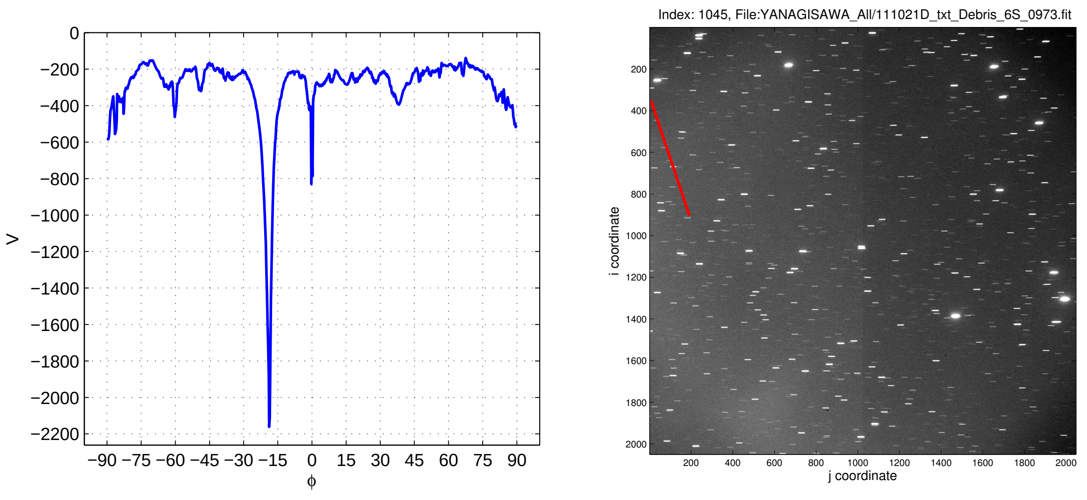
$$V^*(\phi) = \min_{j \in I_n} \min_{b,e \in I_m,\, b \le e} V(b, e, j, \phi)$$

as a function of ϕ is shown in Figure 2.6.

In Phase 2, the angular resolution is refined by a simplified computation. We use a smaller angular step

$$\Delta_{\phi 2} = \frac{1}{5}\,\Delta_{\phi 1} \tag{2.5}$$

and instead of rotating the input image and computing the streakiness map, we rotate the streakiness map itself within a small angular interval of $\langle \phi - 2\Delta_{\phi 2},\, \phi + 2\Delta_{\phi 2} \rangle$ with



(a) Plot of $V^*(\phi)$ (b) Source image with highlighted streak

Figure 2.6: The typical profile of $V^*(\phi)$ for an image with a faint streak. The angle of the streak corresponds to the global minimum. The secondary strong local minimum at $\phi = 0$ is caused by the chip layout artifact due to an inconsistent zero level for each half of the chip, as discussed in Section 1.

the step of $\Delta_{\phi 2}$. Then only the minimization problem per angle needs to be solved. The per-angle computation in Step 2 is about twice as fast than in Step 1, since the streakiness and likelihood ratio maps are not needed to be recomputed. The effective resolution after Step 2 is $\Delta_{\phi 2}$, which would correspond to 315 rotations in our running example.

Phase 3 then performs local optimization started from all local minima of $V(\phi)$ obtained from Phase 2. This is done by a local bracketing method with quadratic convergence. The assumption of this step is that if there is a sufficiently long streak in the image, the method is sufficient to find some of its short subsegment at the low angular resolution and then to refine its pose and length starting from the segment. We observed empirically that this assumptions holds well, typically, there is a short segment of length m_s that has a good response even if the entire streak does not.

The most expensive part of the algorithm is still the image rotation. But it is not necessary to perform the image rotation to achieve the detection task. A suitable image shearing suffices. Shearing is about 5× faster and easily parallelizable. This improvement is described in the next section.

2.2.2 Image shearing

The detection of the streak is done column-wise and because of this the image needs to be rotated by all angles in the interval $\langle -90°, 90°)$ to get the streak into a vertical position. Image rotation with bicubic or even bilinear interpolation is slow, hence image shearing transformation is used instead as it achieves the same qualitative result. On the CMP Grid (Table 4.1) a single bilinear rotation of a 2048×2048 image takes 3.54 seconds and

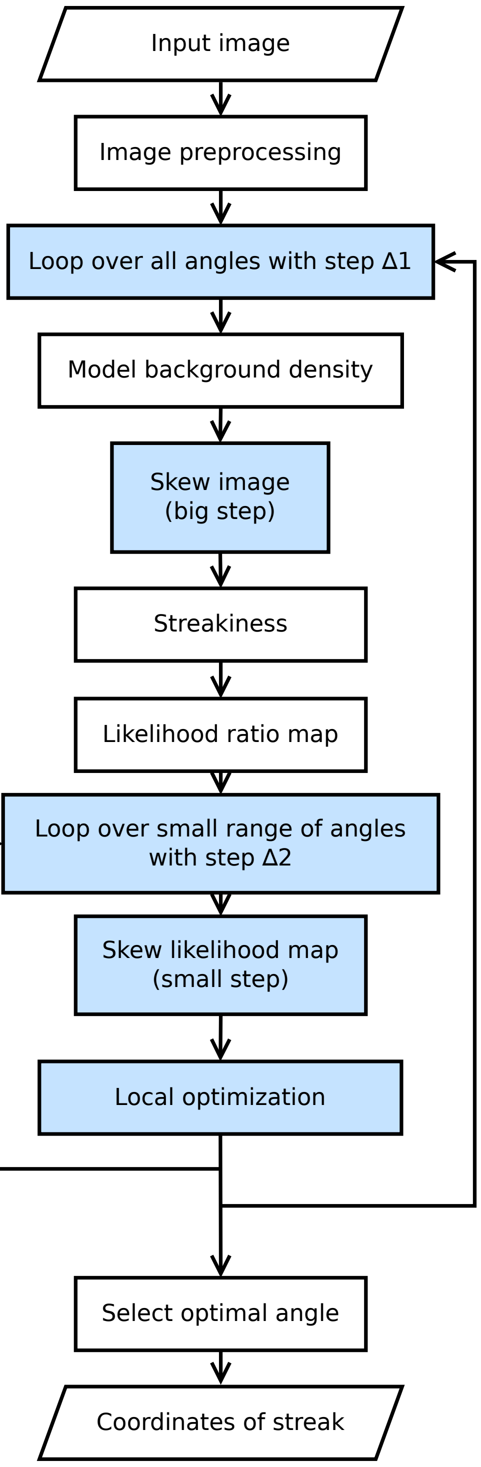


Figure 2.7: Outline of the algorithm. Highlighted processes can be easily parallelized. Small step is the $\Delta_{\phi 2}$ and big step is the $\Delta_{\phi 1}$ described in the main body of the thesis.

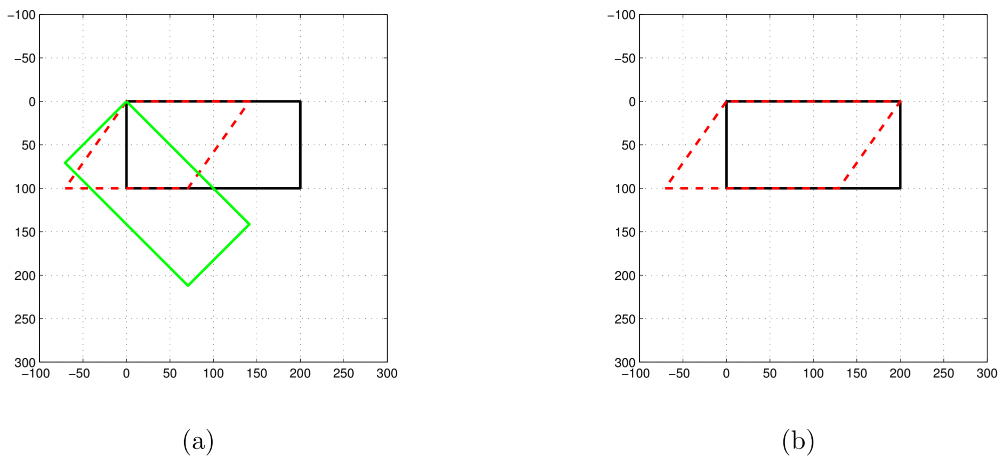


Figure 2.8: Illustration of the decomposition of the rotation transformation. On the left, original black rectangle is sheared by $\mathbf{S_1}$ into the red parallelogram and then transformed by $\mathbf{S_2}$ onto the green rectangle. On the right, the same black rectangle is sheared by $\mathbf{S}^*$.

bicubic 5.88 seconds. My implementation of image shearing takes 0.74 seconds, which is $4.8\times$ and $8\times$ faster, respectively.

We will need a transformation that can put a streak into a vertical position. The original idea was the following. Rotation matrix $\mathbf{R}(\phi)$ can be decomposed into two skew transformations $\mathbf{R}(\phi) = \mathbf{S_2}(\phi)\mathbf{S_1}(\phi)$ as follows:

$$\mathbf{R}(\phi) = \begin{bmatrix} \cos\phi & \sin\phi \\ -\sin\phi & \cos\phi \end{bmatrix}, \quad \mathbf{S_2}(\phi) = \begin{bmatrix} 1 & 0 \\ -\tan\phi & \sec\phi \end{bmatrix}, \quad \mathbf{S_1}(\phi) = \begin{bmatrix} \cos\phi & \sin\phi \\ 0 & 1 \end{bmatrix}. \quad (2.6)$$

An illustration of the application of the transformations is in Figure 2.8(a).

But this can be simplified further, only to shear transformation

$$\mathbf{S}^*(\phi) = \begin{bmatrix} 1 & \tan\phi \\ 0 & 1 \end{bmatrix}, \quad (2.7)$$

which preserves the row length as can be seen in Figure 2.8(b). We will call shearing a transformation with this matrix $\mathbf{S}^*(\phi)$.

Each row is shifted by a value $h(i) = \tan(\phi) \cdot i$ where ϕ is the shearing angle and i is a zero-based number of the row, like in Figure 2.9. If the value of $h(i)$ is not an integer, linear interpolation between neighbouring pixels is used to compute the value.

This approach works well in the interval of angles $\langle -45°; 45° \rangle$. If ϕ is outside of this interval (it belongs to $\langle -90°; -45°)$ or $(45°; 90°\rangle$), the image is first transposed and then sheared by the angle of $\phi \pm 45°$. Because of this interval restriction, the shear transformation $\mathbf{S}^*(\phi)$ is implemented as four partial transformations which use normalized homogeneous coordinates each pertaining to the proper interval:

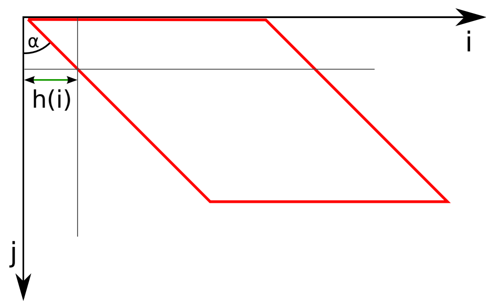


Figure 2.9: Illustration of image shearing by row shifting.

$$\mathbf{S}^*{}_{\langle -90°;-45°)}(\phi) = \begin{bmatrix} \tan(\phi+90) & -1 & m+1 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{2.8}$$

$$\mathbf{S}^*{}_{\langle -45°;0°)}(\phi) = \begin{bmatrix} 1 & \tan(\phi) & q-n \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{2.9}$$

$$\mathbf{S}^*{}_{\langle 0°;45°)}(\phi) = \begin{bmatrix} 1 & \tan(\phi) & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{2.10}$$

$$\mathbf{S}^*{}_{(45°;90°\rangle}(\phi) = \begin{bmatrix} -\tan(\phi-90) & 1 & 0 \\ -1 & 0 & n+1 \\ 0 & 0 & 1 \end{bmatrix}, \tag{2.11}$$

where the input image has size $m \times n$ and the sheared image has size $m \times q$. The dimension q is calculated as

$$q = \lceil |\Delta \cdot m| \rceil + n, \tag{2.12}$$

where $\lceil x \rceil$ is ceiling of x and

$$\Delta = \begin{cases} \tan(\phi+90) & \text{if } \phi \in \langle -90°; 45°), \\ \tan(\phi) & \text{if } \phi \in \langle -45°; 45°\rangle, \\ \tan(\phi-90) & \text{if } \phi \in (45°; 90°\rangle. \end{cases} \tag{2.13}$$

Image shearing is implemented in function `imshear`. It returns an image sheared by the specified angle using the linear interpolation in rows and corresponding transformation matrix. Since row interpolation is not dependent on other rows of the image, `imshear` can be easily parallelized.

Shearing affects the energy profile $V^*(\phi)$ because it does not preserve the line segment length. We use a multiplication factor that approximately corrects the change in $V^*(\phi)$, since the computation of $V^*(\phi)$ uses non-linear operations. The factor is

$$\frac{1}{\max(|\cos\phi|, |\sin\phi|)}. \tag{2.14}$$

The comparison of the profiles can be seen in Figure 2.10. We can see that the profiles are almost identical.

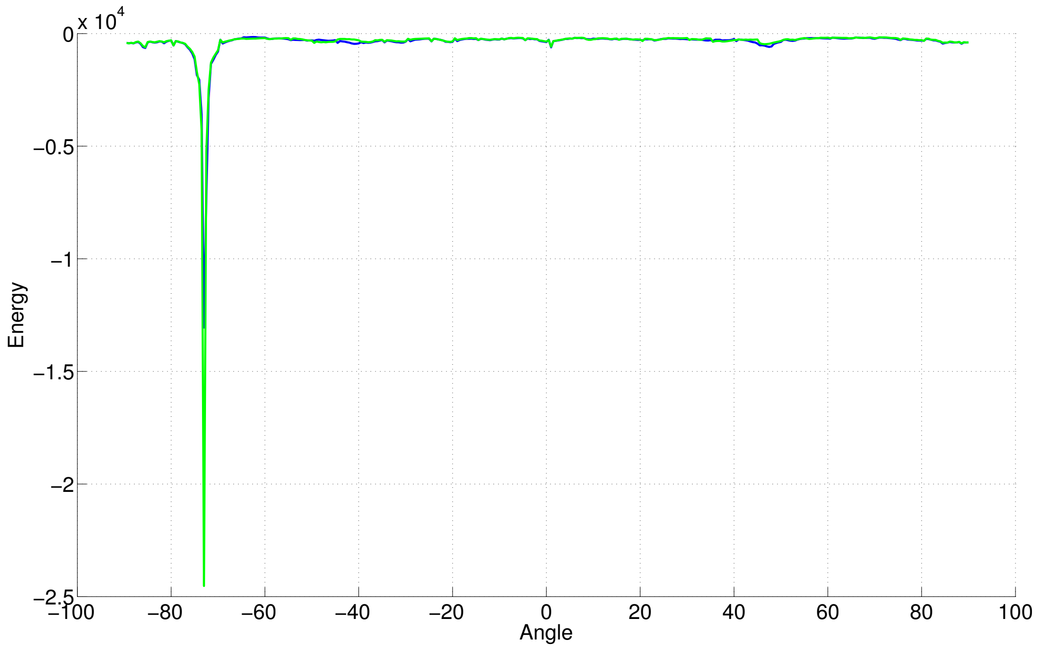


Figure 2.10: Energy profiles using rotational (blue) or shearing (green) transformations.

2.2.3 Selecting optimal angle

The optimal angle ϕ^* in the energy profile of $V^*(\phi)$ (Figure 2.6) represents the angle in which the streak is located. But before the angle selection, we must decide if the image contains a streak or not. The conditions that the energy plot must satisfy have been found empirically. The minimum of the energy at angle ϕ with neighbouring energies in interval $I = \langle \phi - 2^\circ; \phi + 2^\circ \rangle$ are thresholded using a multiple of median of $V^*(\phi)$. If ϕ is close to the start or to the end of the range of $V^*(\phi)$ (which is $\pm 90^\circ$) then values from the other end of the energy profile are used.

When the energies of interval I satisfy the threshold condition, image is classified as containing a streak at the angle $\phi^* = \phi$.

The appropriate threshold was found to be $2.1 \times \text{median}(V^*(\phi))$. The multiplication constant was found as the minimum of the classification error (see Figure 2.11) on an independent set of 100 random synthetic streaks which have been generated similarly to those described in Section 3.3.

Neighbouring values within I are used to prevent the tiling artefacts (Figure 1.3(a)) to be proclaimed as streaks. Energy profiles of two images are displayed in Figure 2.12. The first one (#100) is from the `taos` source which has no streak in it but it contains a tiling artefact and the second one (#2503) is from `odhi` and it contains a streak. We can see that although the `taos` image has a very distinguished minimum, it is only a very narrow peak, as the edges of the tiles are very narrow as well. Thus it does not have many neighbouring values that exceed the threshold. However, this approach is not very efficient and can be improved as discussed in Section 5.

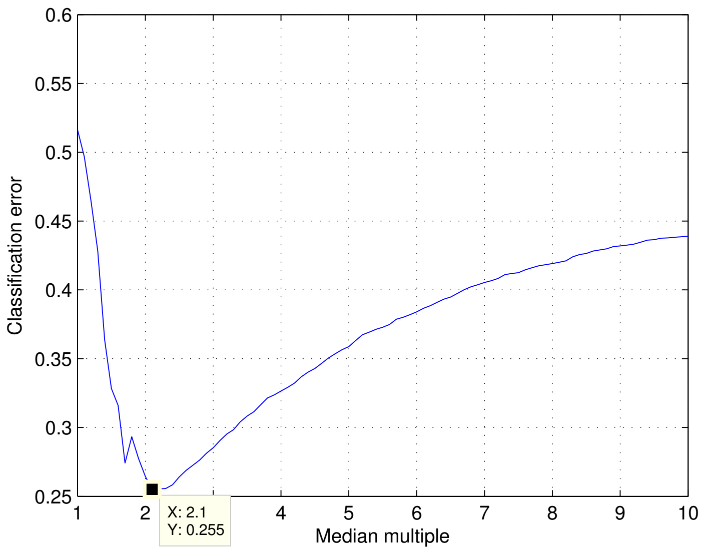


Figure 2.11: Learning of the threshold value. The minimum is located at 2.1 with the error of 25%.

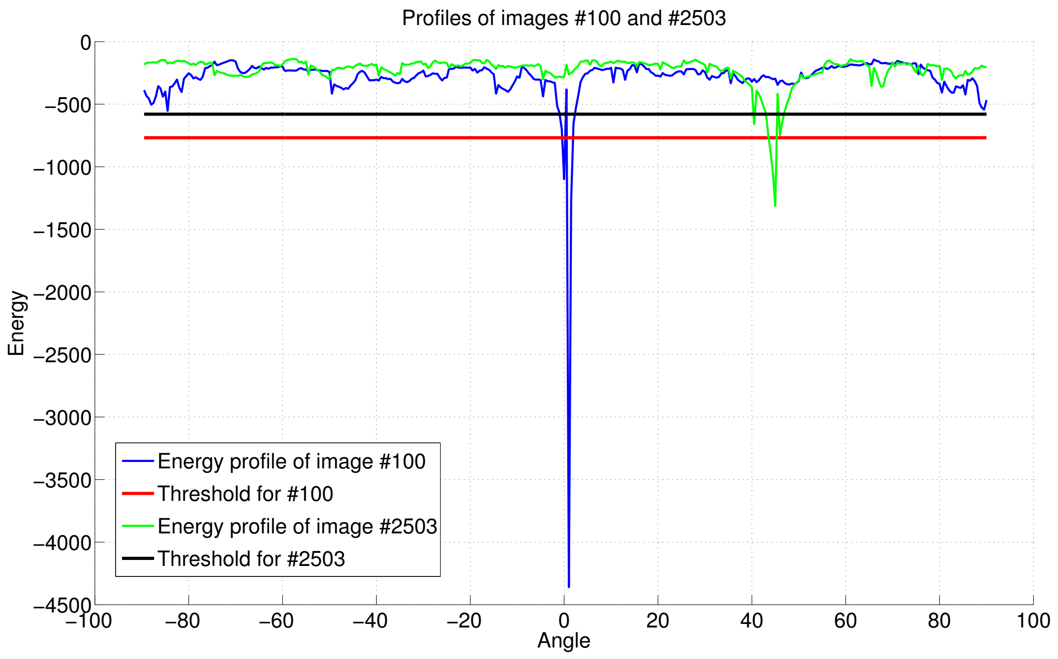


Figure 2.12: Energy profiles of the two images.

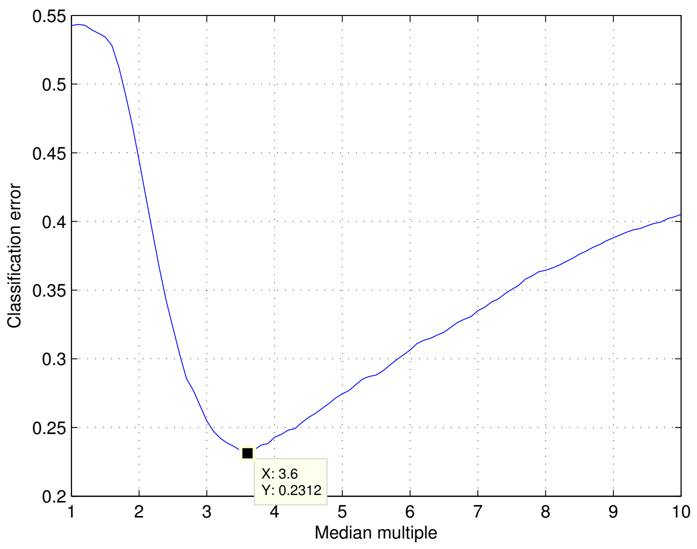


Figure 2.13: Learning of the threshold value. The minimum is located at 3.6 with the error of 23%.

2.3 The YanKuro algorithm

The principle of this algorithm was discussed already in Section 1. We will call $M^*(\phi)$ the maximal value of column-wise taken median in image rotated by angle ϕ. Because image rotation is slow, shearing has been used instead.

There are several issues that have to be solved before YanKuro can be deployed on tested data:

- We need to find the decision threshold that can be used for classification and angle detection.
- And we need to find the equation of the line that corresponds to the found angle and column.

The decision threshold has been empirically found as $3.6 \times \text{median}(M^*(\phi))$. The multiplicative constant was found as the minimum of the classification error (see Figure 2.13) on the same independent set of 100 random synthetic streaks as in the case of the CMP algorithm as discussed in the previous section. A higher value could have been considered though, because it would make a smaller false negative error.

The detected angle ϕ^* and column j^* can be found as

$$(\phi^*, j^*) = \arg\max_{\phi, j} M^*(\phi, j). \tag{2.15}$$

Column j^* corresponds to this angle in the rotated image. We would like to find the equation of the line in the original image. We will choose two points $\mathbf{A_{sk}}$ and $\mathbf{B_{sk}}$ in the column j^* and transform their positions to the positions in the original image:

$$\mathbf{A} = \mathbf{S}^{*-1}(\phi)\mathbf{A_{sk}}, \tag{2.16}$$
$$\mathbf{B} = \mathbf{S}^{*-1}(\phi)\mathbf{B_{sk}}. \tag{2.17}$$

Matrix $\mathbf{S}^{*-1}(\phi)$ is the inverse shearing matrix from (2.7). The equation of the line is created from the points $\mathbf{A}$ and $\mathbf{B}$.

Chapter 3

Data

3.1 Input images

Input images are in standard format FITS. FITS (Flexible Image Transport System) is an open standard defining a digital file format useful for storage, transmission and processing of scientific and other images. FITS is the most commonly used digital file format in astronomy. Its full description can be found in [12].

For the use of image processing it is important to note these characteristics:

- Images are grayscale.
- Range of colors is 16bit (from 0 to 65535).

There are numerous software and libraries to view or work with FITS,[1] for example ds9.[2] In Matlab the following functions are available:

- `fitsdisp` - Display FITS metadata.
- `fitsinfo` - Information about FITS file. This usually involves information like GPS position of the observer, time and coordinates of the observed area.
- `fitsread` - Read data from FITS file.
- `fitswrite` - Write data to FITS file.

Computer monitor displays support only 8-bit range of gray levels. To view the images only a range of the data can be used. This is done in a custom function `fitsview`. It displays the FITS image along with a histogram. The range of colors can be easily set

[1] List of various FITS software: http://tdc-www.harvard.edu/astro.software.html
[2] ds9 - recommended viewer: http://hea-www.harvard.edu/RD/ds9/site/Home.html

Table 3.1: Sources of data

Code name	Number of images	Number of streaks	Image size	Note
taos	1986	29	2050×2048	Data from TAOS Observatory http://taos.asiaa.sinica.edu.tw
ondrejov	422	5	1027×1056	Images from Ondrejov observatory http://www.asu.cas.cz
odhi	97	72	2048×3072	Capture of asteroid 2012DA14
tadn	143	54	736×1092	within Gloria Project
tadandor	102	1	1024×1024	http://live.gloria-project.eu
rme	26	20	1024×1024	Capture of satellite 23613
modra	12	9	1016×1038	Images from Modra observatory http://www.daa.fmph.uniba.sk
simulated	2000	2000	various	Artificially simulated streaks
Σ	4788	2190		→ 45.7% of images contains streaks

in the graphical user interface and the image and section of the histogram is updated automatically. The default display range is 0.5 to 99.5 percentile of the image values.

Since some images have a black border of invalid data, the images are cropped to valid data only. The extent of the border depends on the data source and must be specified beforehand. This is the reason why data is classified into different sources in Table 3.1.

3.2 Annotation of data

Eight different sources of data have been used and streaks have been manually annotated. Table 3.1 summarizes basic information about the data.

Since the data comes from multiple sources, with their own hierarchy and organization, an indexed list with data information was created for the purposes of the testing of the algorithms. It is saved in Matlab's `.mat` file type in matrix `data`, whose structure is

- `source` - code name of the source, as in Table 3.1,
- `file` - path to the FITS file,
- `generated_from` - path to the FITS file that was used for this image generation (this is valid only for the simulated data),

- `has_streak` - true/false,
- `i1`, `j1`, `i2`, `j2` - coordinates `(i,j)` of the start and end point of the streak,
- `phi` - angle at which the streak is located,
- `ampl` - amplitude of the streak, filled in only for the simulated streaks,
- `is_long` - true/false, determines if the streak is longer than 20% of image's diagonal.

The data was manually annotated except for the images with the simulated streaks. A graphical tool for the manual annotation is implemented in function `annotate`. It displays a given image in the standard Matlab interface (with zoom and other functions) and allows dragging of a line that represents the streak. Once satisfied with the streak location, its position is saved into the `(i,j)` coordinates.

3.3 Simulated streaks

We do not know the exact location of the streaks in the manually annotated images. Humans have a tendency to extrapolate line patterns to places where they are not located. To verify if the detection works properly, a custom dataset has been generated. This allows to estimate the deviation of detected positions from the real streaks.

We suppose that the input image I can be modeled as

$$I = x + as, \tag{3.1}$$

where x is a random variable corresponding to background process (stars, noise, etc.), a is the streak amplitude and s is a streak function.

The processs of generating the streaks uses the model (3.1) and it works as follows:

1. Choose a background image from a pool of 10 images that do not contain any streaks.
2. Create a blank "canvas," a zero-filled matrix of the same size as the background image.
3. Choose two random points with coordinates (i_1, j_1) and (i_2, j_2) from uniform distribution.
4. Draw a discrete line between these two points with a pixel intensity

   $$p \sim N(A, \sigma_L), \tag{3.2}$$

   where N is the normal distribution with mean amplitude A and deviation σ_L=10.
5. Apply Gaussian blur to line image with rotationally symmetric matrix of size 9×9 with $\sigma = 2$. The blur does not preserve the amplitude equally well along the line.

6. Add canvas to the background image (by pixel-wise addition).

An example of such a generated streak is in Figure 3.3.

The amplitude a is one of the set $X = 1,\ 2,\ ...,\ 9,\ 10,\ 20,\ 30,\ ...,\ 90,\ 100$ or 1000. Amplitudes of the streaks are chosen in this way to be able to identify what is the success threshold amplitude for the detector in Section 4.1.

An indexed list of ten different input images that do not contain any streaks and have no corruptions is created (they are shown in Figure 3.1 and continued in 3.2), ten random positions are selected and twenty different amplitudes are chosen from X, creating a set of 2000 simulated streaks.

The background images have been chosen randomly from the available data set which does not contain any streaks. Unfortunately, one image (index 2650) has a corruption in it which can mislead the detection algorithm. This fact was noticed too late and this is why for the testing of the simulated data only the first nine source images are used, thus the set is composed from 1800 images only.

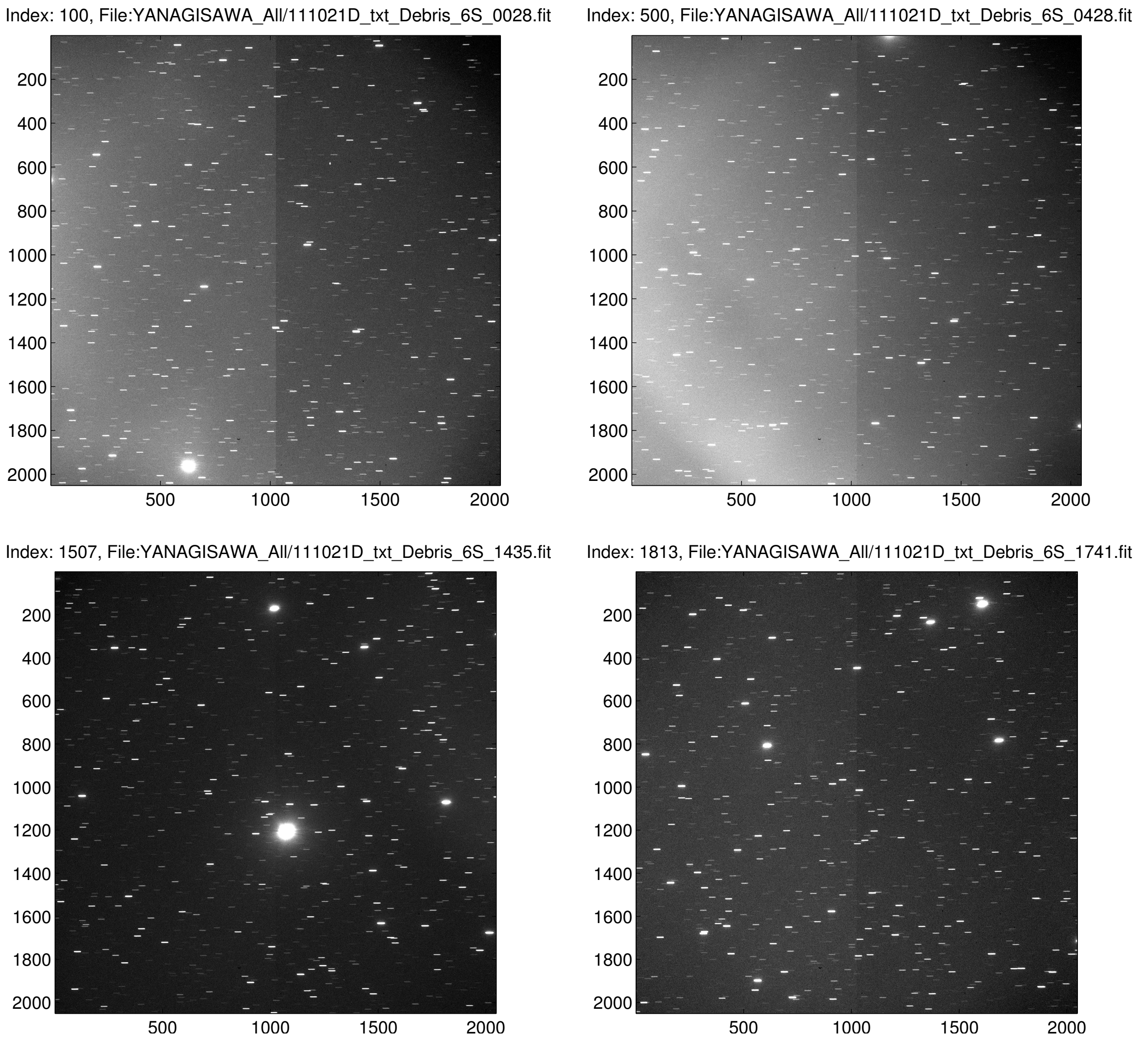

Index: 100, File:YANAGISAWA_All/111021D_txt_Debris_6S_0028.fit
Index: 500, File:YANAGISAWA_All/111021D_txt_Debris_6S_0428.fit
Index: 1507, File:YANAGISAWA_All/111021D_txt_Debris_6S_1435.fit
Index: 1813, File:YANAGISAWA_All/111021D_txt_Debris_6S_1741.fit

Figure 3.1: Background images used for generation of streaks.

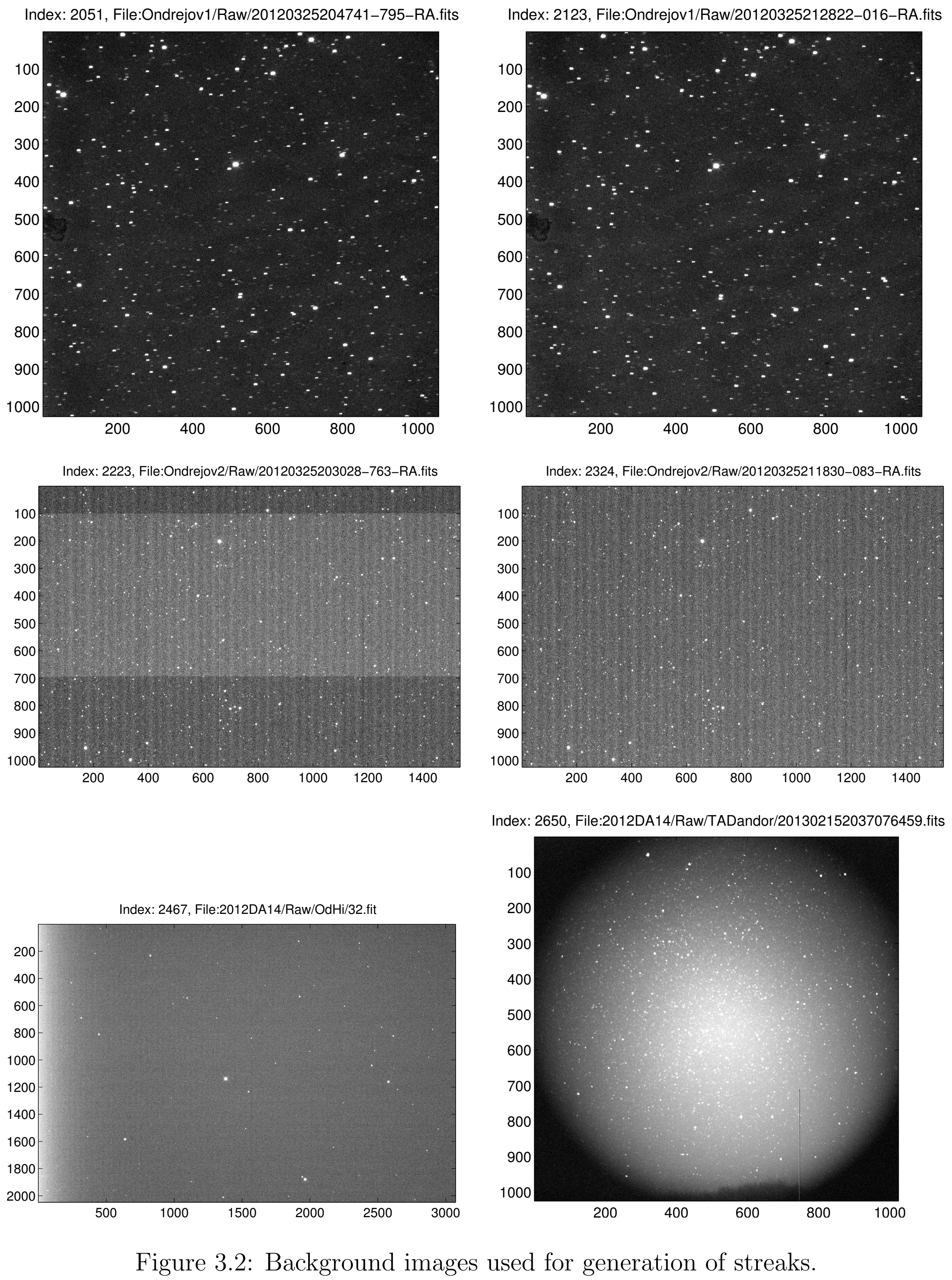

Index: 2051, File:Ondrejov1/Raw/20120325204741−795−RA.fits
Index: 2123, File:Ondrejov1/Raw/20120325212822−016−RA.fits
Index: 2223, File:Ondrejov2/Raw/20120325203028−763−RA.fits
Index: 2324, File:Ondrejov2/Raw/20120325211830−083−RA.fits
Index: 2467, File:2012DA14/Raw/OdHi/32.fit
Index: 2650, File:2012DA14/Raw/TADandor/201302152037076459.fits

Figure 3.2: Background images used for generation of streaks.

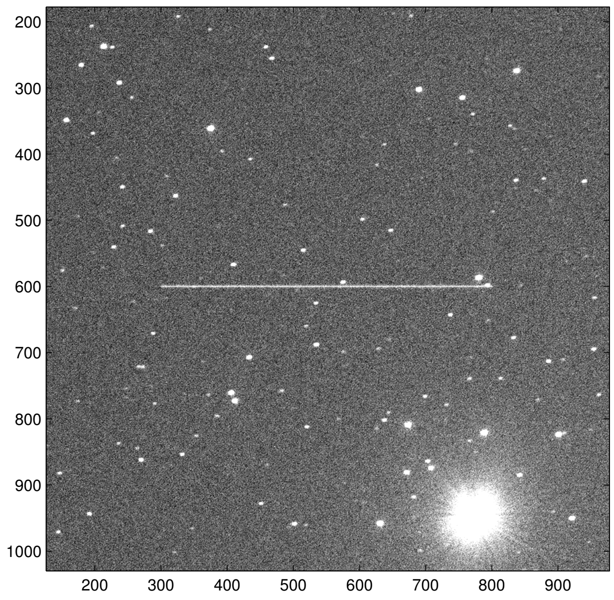


Figure 3.3: A close-up image of a generated streak. Amplitude varies along the streak, but does not overcome a maximal value.

Chapter 4

Testing of the algorithms

Comparison of algorithms can be divided into two categories: Testing of the classification of images (if there is a streak or not), and detection of the streak position.

The classification is based on thresholding energy profile $V^*(\phi)$ or median maxima $M^*(\phi)$ as discussed in Sections 2.2.3 and 2.3, respectively.

To qualify the streak as properly detected, three criteria have to be met:

- Small angular error e_ϕ. The angle of the detected streak is close to the angle of the annotated streak.
- Small perpendicular error $e_\perp$. It can happen that the detected streak has the same angle as the annotated streak, but they are in fact only parallel. Therefore we will measure the error of perpendicular distances of the end-points of detected streak to the line where the annotated streak lies.
- Small end-point error $e_\otimes$. The end-points of the detected streaks are close to the end-points of the annotated streak.

We need to define error measurements. For angle error e_α it will be

$$e_\phi = \min(|\phi_1 - \phi_2|, 180 - |\phi_1 - \phi_2|) \tag{4.1}$$

where ϕ_1 and ϕ_2 are angles of the annotated and detected streak, respectively.

Equation of a line can be expressed as

$$a_1x_1 + a_2x_2 + a_3 = 0, \tag{4.2}$$

and perpendicular distance of point $\mathbf{y} = (y_1, y_2)$ to the line characterized by $\mathbf{a}$ can be calculated as

$$\mathrm{dist}(\mathbf{a}, \mathbf{y}) = \frac{|a_1y_1 + a_2y_2 + a_3|}{\sqrt{a_1^2 + a_2^2}}. \tag{4.3}$$

The perpendicular error distance of the detected streak with end points $\mathbf{y_1}$ and $\mathbf{y_2}$ to the annotated streak characterized by $\mathbf{a}$ is then

$$e_{\perp}^2 = \mathrm{dist}^2(\mathbf{a}, \mathbf{y_1}) + \mathrm{dist}^2(\mathbf{a}, \mathbf{y_2}). \tag{4.4}$$

The end-point error is

$$e_{\otimes}^2 = \min(||\mathbf{x_1} - \mathbf{y_1}||^2 + ||\mathbf{x_2} - \mathbf{y_2}||^2,\ ||\mathbf{x_1} - \mathbf{y_2}||^2 + ||\mathbf{x_2} - \mathbf{y_1}||^2). \tag{4.5}$$

The minimum is used because we do not know the relative orientation of the line segments (whether they are concordant or not).

Streaks are said to be close enough and therefore correctly detected if both errors $e_{\perp}$ and $e_{\otimes}$ are less than 100 pixels and the angular difference e_{ϕ} of these two streaks is less than 2°.

Since the nature of the algorithms is slightly different, the fair comparison between the YanKuro and the CMP algorithm is done on images with `is_long` set to true.

The results of the tests with the CMP algorithm is saved into structure `data_detected_str` with the following fields:

- `has_streak` - true/false,
- `i1`, `j1`, `i2`, `j2` - coordinates `(i,j)` of the start and end point of the streak,
- `phi` - angle of the detected streak,
- `Vai` - profile of the energy $V^*(\phi)$,
- `Vai_phi` - angles to which the profile values pertain,
- `time` - length of the processing time.

Similarly, for the YanKuro algorithm the results are saved into structure `data_detected_yan` with following fields:

- `has_streak` - true/false,
- `a` - representation of a line for the detected streak,
- `phi` - angle of the detected streak,
- `maxv` - profile of the median of maximal values $M^*(\phi)$,
- `time` - length of the processing time.

The software was tested on two machines: Personal laptop and the computational grid at CMP, here referenced as the CMP Grid. The summary of information about their CPUs is in Table 4.1. The computation on the grid was distributed using `qsub`,[1] a job submission mechanism for clusters.

Note: In the following figures, the CMP algorithm is drawn in blue and YanKuro in green.

[1] https://wikis.nyu.edu/display/NYUHPC/Tutorial+-+Submitting+a+job+using+qsub

Table 4.1: Computers used for processing, a selected list of information available with `lscpu`.

Machine	CMP Grid	Personal laptop
Architecture:	x86_64	x86_64
CPU op-mode(s):	64-bit	64-bit
CPU(s):	8	4
Thread(s) per core:	1	2
Core(s) per socket:	8	2
Vendor ID:	AuthenticAMD	GenuineIntel
CPU family:	16	6
CPU MHz:	2000.295	782.000

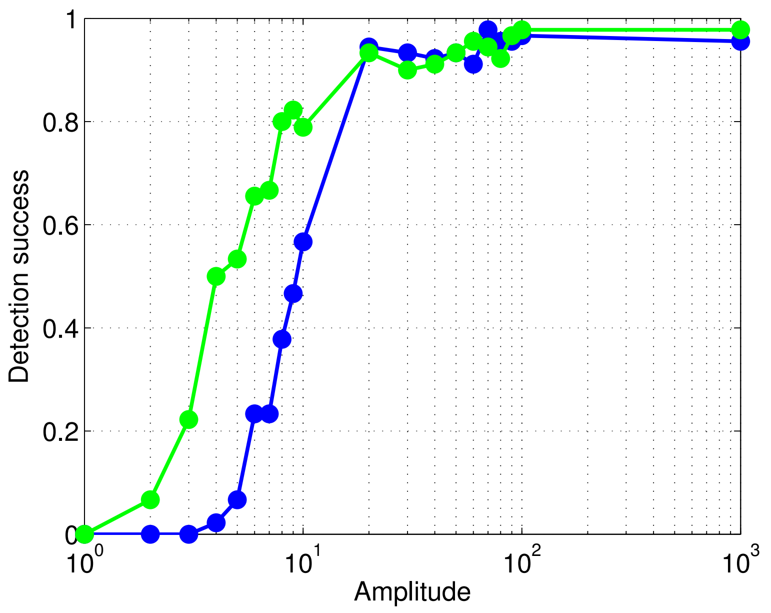


Figure 4.1: Plot of angular detection success based on amplitude of simulated streak for the CMP algorithm (blue) and YanKuro algorithm (green).

4.1 Threshold amplitude

Both detection algorithms have been tested for their threshold amplitude a from equation (3.1). The test is based on detection of streaks in simulated data. Since the Gaussian blur that is used for streak generation does not preserve the amplitude, the actual image amplitude is smaller. Figure 4.1 shows the plot of detection success. Each point in the graph shows how successful the method is in detecting simulated streaks of given amplitude. Based on this plot we will distinguish weak and strong simulated streaks. The breakthrough of detection success seems to be at the amplitude of 20, so we will call weak streaks those below this amplitude and strong streaks those over or equal to this amplitude.

Table 4.2: Results of the classification for the CMP algorithm (blue) and YanKuro (green). The column "Properly classified" is the sum of true and false positives.

Source	Input images	Annotated streaks	False negatives	False negatives	False positives	False positives	Properly classified	Properly classified	True positives	True positives	Classification success	Classification success
modra	12	9	1	5	0	0	11	7	8	4	0.91	0.58
odhi	97	72	0	59	1	7	96	31	72	13	0.99	0.32
ondrejov	422	5	0	3	14	42	408	377	5	2	0.97	0.89
rme	26	20	14	16	0	2	12	8	6	4	0.46	0.31
tadandor	102	1	1	1	0	9	101	92	0	0	0.99	0.90
tadn	143	54	32	48	2	19	109	76	22	6	0.76	0.53
taos	1986	29	4	21	23	331	1959	1634	25	8	0.99	0.82
Σ	2788	190	52	153	40	410	2696	2225	138	37	0.98	0.80
simulated strong	900	900	49	52	0	0	851	848	851	848	0.95	0.94
simulated weak	900	900	723	434	0	0	177	466	177	466	0.19	0.52
Σ	1800	1800	772	486	0	0	1028	1314	1028	1314	0.57	0.73

4.2 Classification of images

Table 4.2 contains the results of the classification based on thresholding as discussed in Sections 2.2.3 and 2.3.

Classification of the CMP algorithm:

- False positives in `ondrejov` are caused by vertical waviness of the image from CCD camera.
- False negatives in `rme` and `tadn` are caused by very short streaks.
- False positives in `taos` are caused by tiling defect.
- The great number of of false negatives in `simulated weak` data is caused by the insufficient amplitude of streak or artefacts in the backgrounds that are stronger that these streaks.

Classification of the YanKuro algorithm:

- False negatives in `modra` and `odhi` are caused by very short streaks.
- The great number of of false negatives in `simulated weak` data is caused by the insufficient amplitude of streak or artefacts in the backgrounds that are stronger that these streaks.

Table 4.3: Results of the angular detection for the CMP algorithm (blue) and YanKuro (green). The input samples are only the true positives from Table 4.2.

Source	Samples	Samples	Properly detected	Properly detected	Detection success	Detection success
modra	8	4	4	0	0.50	0.00
odhi	72	13	46	0	0.64	0.00
ondrejov	5	2	5	2	1.00	1.00
rme	6	4	5	0	0.83	0.00
tadandor	0	0	0	0	-	-
tadn	22	6	19	4	0.86	0.67
taos	25	8	25	2	1.00	0.25
Σ	138	37	104	8	0.75	0.22
simulated strong	851	848	851	848	1.00	1.00
simulated weak	177	466	177	455	1.00	0.98
Σ	1028	1314	1028	1303	1.00	0.99

4.3 Detection of streaks

Detection based on angle

Table 4.3 contains the results of the angular detection. The threshold for decision is the angular error $e_\phi = 2°$.

Angular detection of the CMP algorithm:

- Detection works well except for short streaks in `modra` and `odhi` where the angular deviation can be quite big.

Angular detection of the YanKuro algorithm:

- Low performance is caused by too low threshold on these sets of data. There is no significant peak in `modra`, `odhi` and `rme`.

Detection based on perpendicular distance

Table 4.4 contains the results of the perpendicular detection. The threshold for decision is error distance $e_\perp = 100\,\text{px}$.

Table 4.4: Results of the perpendicular distance detection for the CMP algorithm (blue) and YanKuro (green). The input samples are only the true positives from Table 4.2.

Source	Samples	Samples	Properly detected	Properly detected	Detection success	Detection success
modra	8	4	4	0	0.50	0.00
odhi	72	13	72	1	1.00	0.08
ondrejov	5	2	4	2	0.80	1.00
rme	6	4	6	0	1.00	0.00
tadandor	0	0	0	0	-	-
tadn	22	6	22	4	1.00	0.67
taos	25	8	25	1	1.00	0.13
Σ	138	37	133	8	0.96	0.21
simulated strong	851	848	736	765	0.86	0.90
simulated weak	177	466	113	408	0.64	0.88
Σ	1028	1314	849	1173	0.83	0.89

Perpendicular detection of the CMP algorithm:

- Detection has better results than angular detection since it can handle short streaks more appropriately.

Perpendicular detection of the YanKuro algorithm:

- Low performance is similar to the performance in angular detection. This is caused by the same reasons – low threshold value.

Detection based on end-point distance

Table 4.5 contains the results of the end-point distance detection. The threshold for decision is error distance $e_{\otimes} = 100\,\text{px}$. This detection was done only for the CMP algorithm. The low results are caused by imprecise angle detection which leads to non-optimal interval selection in (2.2), therefore the detected streak is too short.

Table 4.5: Results of the end-point distance detection. The input samples are only the true positives from Table 4.2.

Source	Samples	Properly detected	Detection success
modra	8	4	0.50
odhi	72	70	0.97
ondrejov	5	2	0.40
rme	6	5	0.83
tadandor	0	0	-
tadn	22	20	0.91
taos	25	17	0.68
Σ	138	118	0.85
simulated strong	851	651	0.76
simulated weak	177	91	0.51
Σ	1028	742	0.72

4.4 Time complexity

Graphs of time consumptions of the CMP and YanKuro algorithms are in Figure 4.2. The spread is caused by the fact that not all processing units used had the same CPU. The CMP algorithm is approximately twice as fast as YanKuro algorithm.

The complexity for the worst case scenario can be estimated as

$$\text{CMP: } O(N_\phi \cdot (n + m/2) \cdot m), \tag{4.6}$$
$$\text{YanKuro: } O(N_\phi \cdot (n + m/2) \cdot m \log m), \tag{4.7}$$

where N_ϕ is number of rotations (transformations) of image of size $m \times n$. The term $n + m/2$ corresponds to image skewing, $m \log(m)$ is computation of the median $M^*(\phi)$ and m is the search for the cumulative maximum in the inference algorithm [10]. If image is square ($m = n$), it could be simplified to

$$\text{CMP: } O(N_\phi \cdot n^2), \tag{4.8}$$
$$\text{YanKuro: } O(N_\phi \cdot n^2 \log n). \tag{4.9}$$

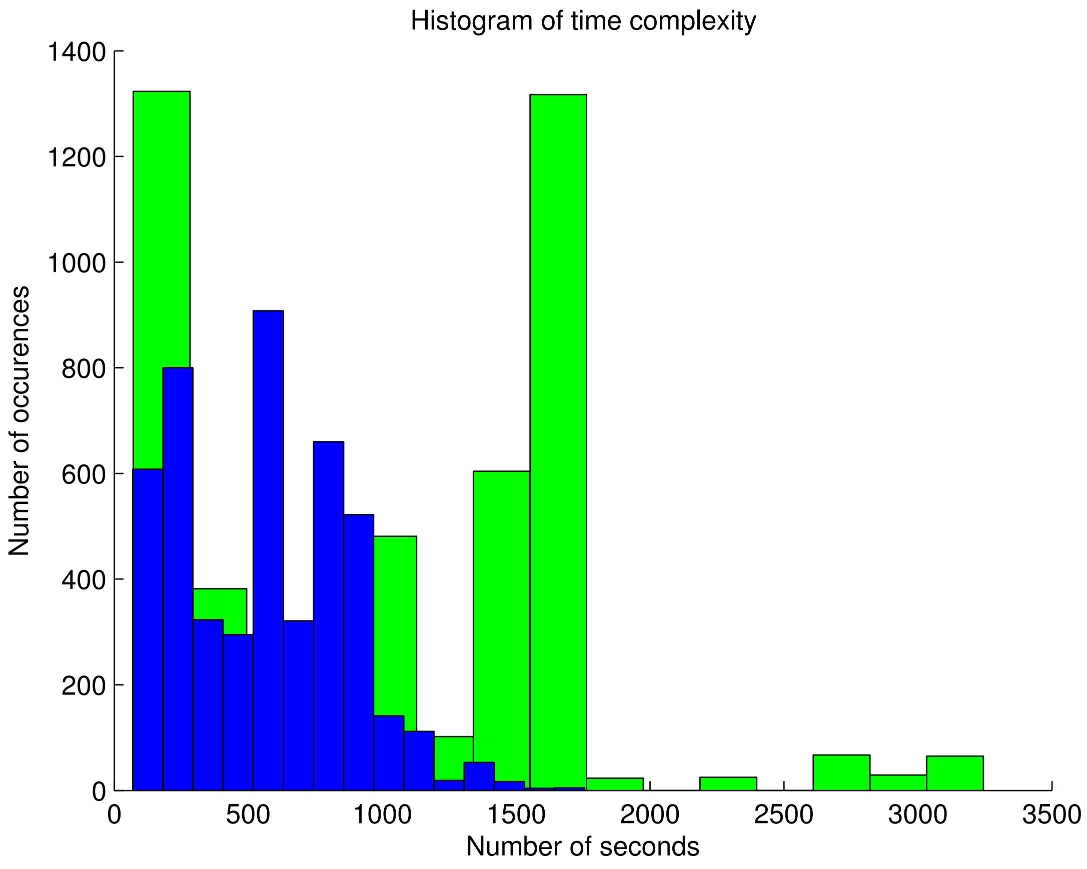


Figure 4.2: Time consumption of the CMP algorithm (blue) with mean=561 sec, std=305 sec and YanKuro (green) with mean=1010 sec, std=707 sec.

Chapter 5

Conclusions

I have successfully implemented a fast shearing transformation and angle selection for the CMP algorithm and reimplemented the YanKuro algorithm. I have created a generator of simulated streaks and I have created tests for comparison of classification of images and detection of streaks in the images and proposed the evaluation method.

The CMP algorithm has a 98% success of image classification on real data and 57% on simulated data, YanKuro has 80% success on real data and 73% on simulated data. Three detection tests have been used to study the behaviour of the algorithms: Based on angular error, error in perpendicular distance and end-point distance. The CMP algorithm does well for the perpendicular distance: It detects 96% of streaks in real data, 83% in simulated, compared to YanKuro with 21% for real data and 89% for simulated data. The angular error test is not so much in favor of the CMP algorithm as perpendicular distance for real data: 75% of streaks have been detected well, but 100% of simulated streaks are detected. YanKuro has success of 22% on real data and 99% on simulated data. The end-point distance detection test was done only for the CMP algorithm: there is 85% detection success for real data and 72% for simulated data.

The CMP algorithm runs about twice as fast as the YanKuro algorithm.

Some of the testing wasn't done as properly as it could be. The background images used for generation of simulated streaks contain artefacts that the algorithms can mistake as streaks, therefore many weak streaks could not be detected. This affects the graph of threshold amplitudes in Figure 4.1 for the CMP algorithm. The YanKuro algorithm takes as input the difference of two consecutive images, therefore these artefacts would not be present there.

This thesis was created for an older version of the CMP Algorithm. The newer version contains improvements that make the algorithm faster and more precise. It can find the streak with an arbitrary precision very fast as it uses the gradient method for the search of the angular position. This also leads to an energy profile with a very deep peak which is easy to distinguish.

Image preprocessing could be used to identify invalid data where known artefacts are located and therefore avoid erroneous results of the algorithm.

For real-world application the CMP algorithm should be implemented in faster environment, such as the C language instead of a fast prototyping tool like Matlab.

Bibliography

[1] *Near space environment*, [Online; http://www.faqs.org/espionage/Mo-Ne/Near-Space-Environment.html, accessed 8-May-2013].

[2] *Orbital debris FAQ: How much orbital debris is currently in Earth orbit?*, [Online; http://orbitaldebris.jsc.nasa.gov/faqs.html, accessed 8-May-2013].

[3] *Orbital debris graphics*, [Online; http://orbitaldebris.jsc.nasa.gov/photogallery/beehives.html, accessed 8-May-2013].

[4] *Proceedings of sixth European conference on space debris*, 2013, [In press].

[5] Becky Iannotta and Tariq Malik, *U.S. satellite destroyed in space collision*, (2009), [Online; http://www.space.com/5542-satellite-destroyed-space-collision.html, accessed 8-May-2013].

[6] A. E. Kolessa, *Detection of faint space debris elements with unknown orbits*, In Proceedings of Sixth European Conference on Space Debris, 2013, [In press].

[7] Paul Maley, *Space debris page*, [Online; http://www.eclipsetours.com/paul-maley/space-debris-2/, accessed 8-May-2013].

[8] Paul Marks, *Satellite collision more powerful than China's ASAT test*, (2009), [Online; http://www.newscientist.com/article/dn16604-satellite-collision-more-powerful-than-chinas-asat-test.html, accessed 8-May-2013].

[9] Robin McKie and Michael Day, *Warning of catastrophe from mass of 'space junk'*, (2008), [Online; http://www.guardian.co.uk/science/2008/feb/24/spaceexplorationspacejunk, accessed 8-May-2013].

[10] Radim Šára, *Random streak detection in optical telescope images*, Research Report CTU–CMP–2013–09, Center for Machine Perception, K13133 FEE Czech Technical University, Prague, Czech Republic, May 2013.

[11] Patrick Seitzer, Andrew Burkhardt, Tommaso Cardonna, Susan M Lederer, Heather Cowardin, Edwin S Barker, and Kira J Abercromby, *Observations of GEO debris with the Magellan 6.5-m telescopes*, In Proceedings of Sixth European Conference on Space Debris, 2012.

[12] Wells, Greisen, and Harten, *FITS – A flexible image transport system*, Astronomy and Astrophysics Supplement Series **44** (1981), 363.

[13] Toshifumi Yanagisawa and Hirohisa Kurosaki, *Detection of faint GEO objects using JAXA's fast analysis methods*, Transactions of The Japan Society for Aeronautical and Space Sciences, Aerospace Technology Japan **10** (2012), no. ISTS28, 29–35.

[14] Toshifumi Yanagisawa and Atsushi Nakajima, *Detection of small LEO debris by use of the line detection method*, Japan Society of Aeronautical Space Sciences **51** (2003), 564–572.