STREEK EVALUATION TOOL

Nina Marić[1], Noelia Sánchez-Ortiz[2], María Jesus Enriquez Pabon[2], Benjamín Murcia Sánchez[2], Jaime Nomen Torres[3], Laura Pedrouzo Rodriguez[1], Tim Flohrer[4], Jan Siminiski[4]

(1) DEIMOS Space UK Ltd, Building R103, Fermi Avenue, Harwell Campus, Oxfordshire, OX11 0QR, UK
spacesafety@deimos-space.com
(2) DEIMOS Space S.L.U., Ronda de Poniente 19, Tres Cantos, Madrid, 28760 Spain
spacesafety@deimos-space.com
(3) Deimos Engineering and Systems, Calle Francia 9, 13500 Puertollano, Spain
(4) ESA/ESOC Space Debris Office, OPS-SD, ESA/ESOC, Robert Bosh Strasse, Darmstadt, Germany
Tim.Flohrer@esa.int

ABSTRACT

Optical telescope images in the SST field may contain streaks (a list of pixels with higher local salient pixel density and more elongated shape than background noise and other artefacts). This is particularly common for very large Field of View (FoV) sensors and low Earth objects. Automatic processing of streaks is challenging due to many factors, including but not limited to: false streaks caused by artefacts, fading introducing difficulty establishing streak ends, and chained behaviours. Deimos created a database of images cataloguing streaks detected by the Deimos trail analyser tool for a future test of algorithms for an intelligent trail detector. The tool searched for trailed shapes on the images, analysed and collected features of the trails found, and classified them accordingly. The analysis showed that around 80% of images contained streaks, whereas streaks were detected automatically in 50-70% of cases, depending on the sensor features and image conditions.

1 INTRODUCTION

Optical telescope images in the SST field may contain streaks (a list of pixels with higher local salient pixel density and more elongated shape than background noise and other artefacts). It is typical for the observation with very large Field of View sensors and low Earth objects (which move faster than high orbiting objects). The processing pipeline for astrometric and data reduction of such images considered the variety of features of those streaks for a robust solution. Until now, the European Space Agency (ESA) has tested prototype algorithms (developed in a General Support Technology Programme) based on simulated images.
The streak detection algorithm was developed considering various features in the images to ensure a robust streak detection solution. The images with spurious artefacts like cosmic rays, hot pixels, etc., were considered (Fig. 1). Different sensor features were used during this activity, like FoV, pointing accuracy, time tag accuracy, resolution, etc. Additionally, different types of observations were accounted to reflect the different streak features.
Observing and processing images every night showed many trail features caused by fast rotators, particularly observing geometries, different object attitudes, and entering in Earth shadow conditions. Some of them were fading until disappearing. Others showed chained behaviours, making it difficult to establish the actual trail ends and the accurate measurement and time registry belonging to the start-end over the plate exposure (Fig. 2). Sometimes, it was impossible to detect them as streaks because they did not show as simple stripes.

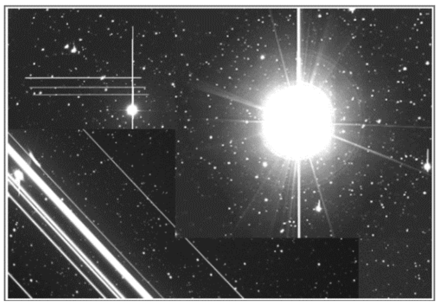
Figure 1: Artefacts in real optical images

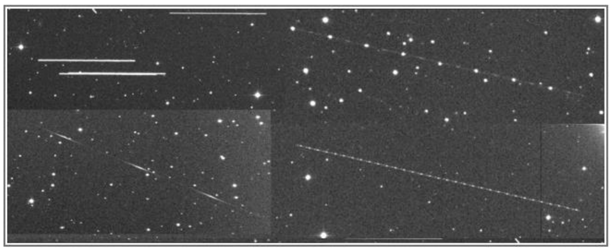
Figure 2: Variety of trails features in real optical images

The project objective was to identify the sensors and possible datasets of raw observation derived data products stored in the database and provided to ESA. The database was designed to support the archive of data acquired during the activity and support the future test and validation of streak detection algorithms.

Visualisation capabilities were also applied to ease the analysis of database contents.

2 OBSERVATION CAMPAIGN

Within the campaign, several sensors of different features were assigned to the observation, and more than 125 000 images were obtained:

- Two United Kingdom sensors from the Open University, COAST and PIRATE
- DEIMOS sensors ANTSY1 and TRACKER2
- Sybilla Technologies' sensor PANOPTES-MAM.

Different objects were observed within different orbital parameters. The observations were generated with varying exposure times to have streaks of various lengths for similar orbital regimes. Diverse background conditions were also considered, benefiting from observing varying periods during the month (with varying Moon light conditions). For the image generation, sensors were following scheduled known targets based on a two-line element set (TLE) and the online Satellite Catalogue (SATCAT) cross-section information [1]. Images were delivered in triplets over the same FoVs. The objective was to get at least three trails of the target inside the same FoV, trying to centre the middle one. However due to the angular speed, pointing uncertainties of the sensor and TLE accuracy, sometimes two, one, or none could be registered. After three shots, the sensor might slew to the next target to do the same tasks; they were all solved by World Coordinate System (WCS) [2]. In the case that some of them could not be resolved, it was usually because the observing conditions were too extreme for solving the plates. In those situations, the images were not further processed.

Tab.1 shows a summary of observation nights per each sensor. For some of them, it was necessary to recover observation nights to reach a minimum required number of images (600 images/night) due to the reasons mentioned above.

Table 1:Summary of observation night

Sensor	**Number of planned/ operated nights**	**% of nights completed**	**Num. of images**	**Num. of unsolved FITS**
TRACKER2	25/25	100%	53532	3200
ANTSY1	25/26	104%	33717	7808
PIRATE	25/32	128%	23406	-
COAST	25/37	148%	13141	-
PANOPTES-MAM	14/27	193%	5400	-

Image resolution for TRACKER2 and ANTSY1 was 512 x 512 pixels for binning2 and 1024 x 1024 pixels for binning1, while for COAST, it was 3032 x 3032 pixels (x1) and 1496 x 1496 pixels (x2). For PIRATE 4056 x 4056 pixels (x1), and 2008 x 2008 pixels (x2). Furthermore, for PANOPTES-MAM, for binning1 was 2048 x 2048 pixels and 4056 x 4056 pixels. For binning2 it was 1024 x 1024 pixels and 2008 x 2008 pixels.

Sensors were using sidereal tracking mode and longer exposure time to produce trails of different length, intensity, variations, etc. The following graphs show typical track lengths where initially they were requested (in pixels for the bin mode) from a few to > 100. The goal was to reach trail diversity longitudes to analyse different cases. The longer the length, the longer the exposure of the camera. Statistics of the generated data summarised:

- For TRACKER2 the most common target lengths were 84 and 96 pixels for x1 and 12, 36 and 48 pixels for x2 (Fig. 3).
- For sensor ANTSY1 more common target length was shorter for both binning modes. For x1 it is 12, 24, 36 pixels and for x2 12, 24 pixels (Fig. 4).
- For PANOPTES-MAM sensor, the most common trail length for x1 was 36 and 108 pixels, and 48 pixels for x2 (Fig. 5)
- For COAST (Fig. 6) and PIRATE (Fig. 7), the most common trail length for both binning modes was 156 pixels.

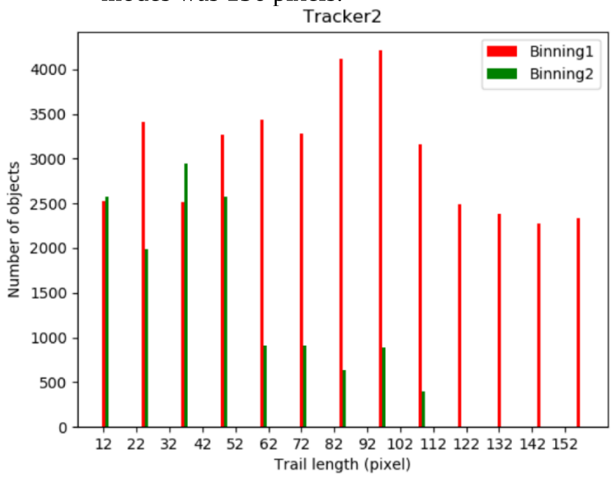

Figure 3: Summary of the number of the objects per trail length for TRACKER2

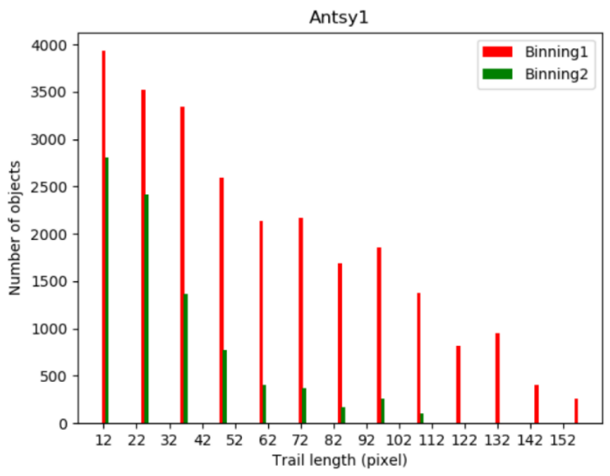

Figure 4: Summary of the number of the objects per trail length for ANTSY1

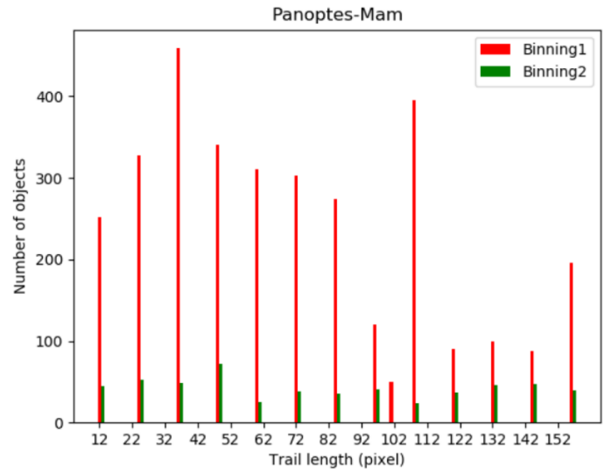

Figure 5: Summary of the number of the objects per trail length for PANOPTES-MAM

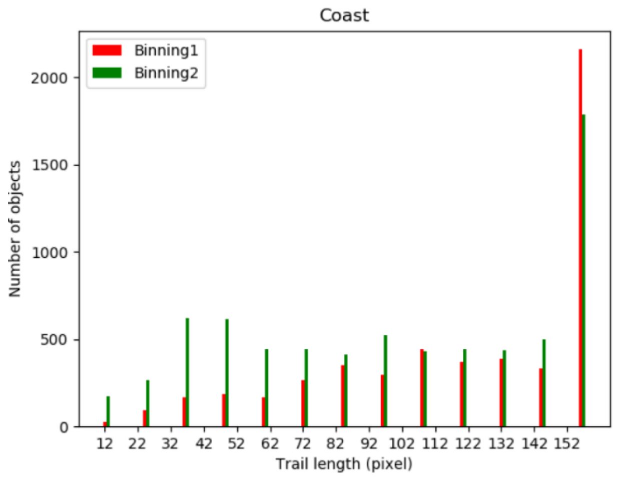

Figure 6: Summary of the number of the objects per trail length for COAST

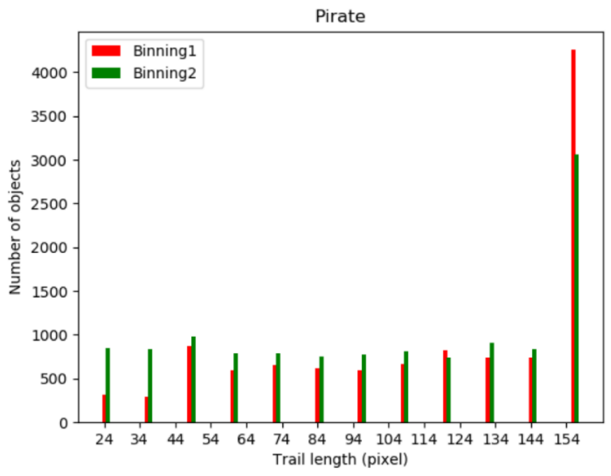

Figure 7: Summary of the number of the objects per trail length for PIRATE

3 IMAGE PROCESSING

Images were processed and analysed with two different tools and approaches:

- Deimos processing tool
- Astrometry24.net (A24N)

To verify Deimos tool efficiency, which was the baseline for characterising the images, a set of images was also manually analysed and processed with two additional tools:

- ESA's Streak Det tool
- Czech Technical University's detection tool

3.1 Deimos Processing Tool

Fig. 8 approximately shows a required software (SW) architecture for processing, extracting the data and cataloguing, from raw Flexible Image Transport System (FITS) images to user interfaces.

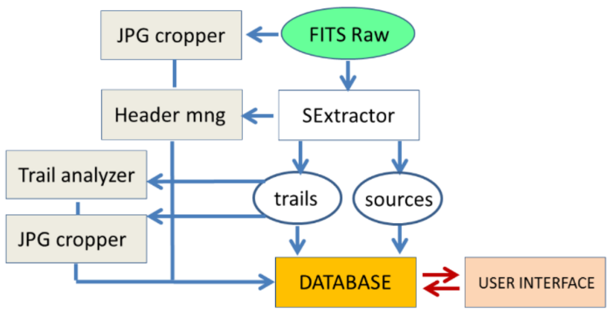

Figure 8: Software architecture for processing the raw FITS images during the activity

After obtaining FITS raw images, SExtractor was used for automated detection and photometry for sources in

FITS image files. The outputs were the catalogue with a range of characteristics of each detected object and maps of these objects. Some of the issues that occurred using SExtractor [3] were:

- It had significantly low precision in trail detection in crowded fields because it is mainly oriented towards reducing large-scale galaxy survey data.
- It had limited accuracy in the determination of a trail's barycentre and ends.
- It was also possible that trails were not detected by SExtractor at all.

For each sensor, SExtractor parameters had to be changed and adapted as much as possible according to sensor specification and FITS images that it generated. It was necessary to compromise between more accurate detection and loss of faint trails by changing some parameters.
SExtractor's catalogue was processed using Python scripts and its various libraries. The software applied filters for real streaks extraction. The output parameters were extracted, used for image processing and object light curve was obtained, a starting point for further detailed analysis and classification. The algorithm marked a streak as real or false based on that light curve and the trail's target length. Trails were not well categorised (false positive/false negative) in crowded, shacked images, in those close to the full Moon or Milky Way or when trails themselves were not well-detected by SExtractor (which caused inaccurate angle and/or barycentre).

It was necessary to smooth the light curve using convolution to determine the real length of a trail and detect them correctly. Fig. 9 shows a detected streak and its corresponding light curve in Fig. 10 (blue line), while the purple line is convolution. After smoothing, the algorithm looked for a target trail length over it. At the point where the purple line intersected the red line (the background value of the image), if a certain trail length was detected between these two intersections, it was detected as a trail. In this case, these lines did not intersect the red line at the same point, underestimating the streak length.

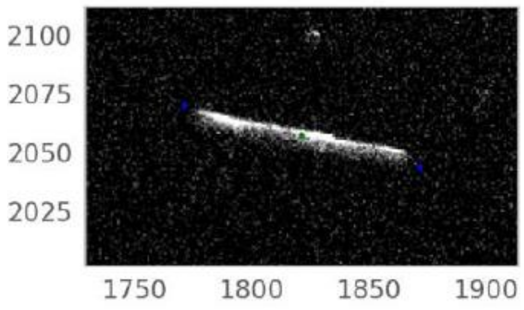

Figure 9: Detected streak.

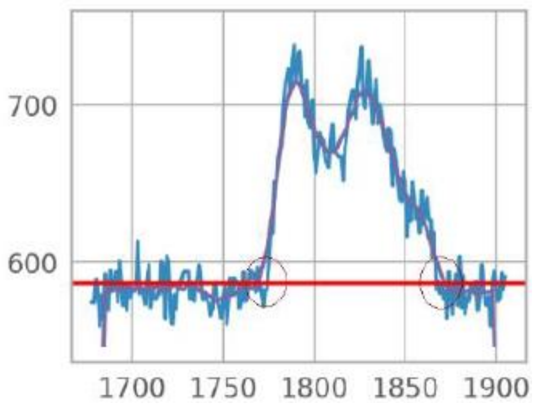

Figure 10: Corresponding light curve to detected streak in Fig. 9

Sometimes if an image had a high signal-to-noise ratio (SNR) or was crowded, it was difficult to detect real streaks using smoothed light curve. Fig. 11 represents an example of a detected streak using convolution and a target length. Fig. 12 shows corresponding light curve and original image (Fig. 13). In these cases, it was a false positive streak.

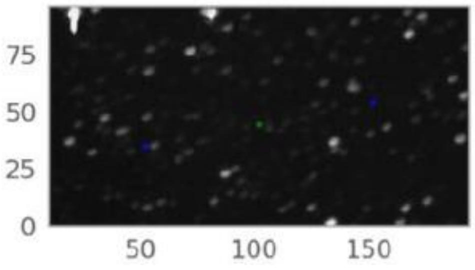

Figure 11: False positive detection

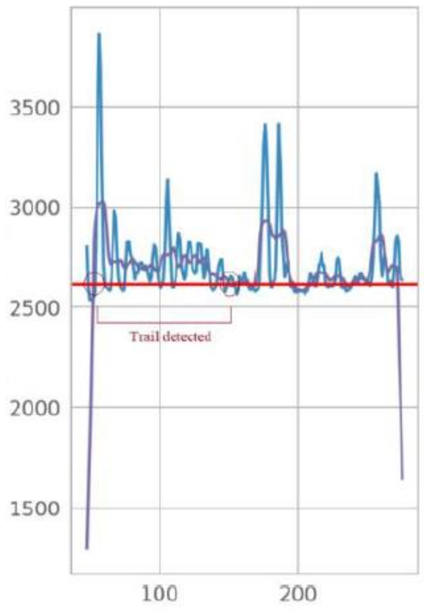

Figure 12: Corresponding light curve to detected streak in Figure 11

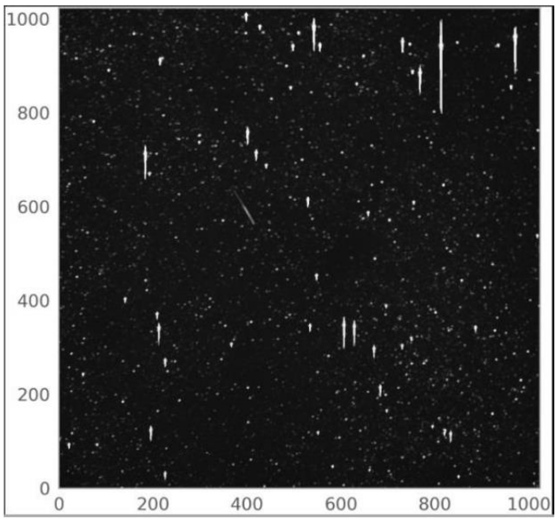

Figure 13: Original image of detected streak in Fig. 11

3.2 Astrometry24.net

Sybilla Technologies have developed Astrometry24.net (A24N). It is an astronomic web service that provides astrometric and photometric measurements and streak detection for SST and Near-Earth Object (NEO) observations. The use of A24N mainly allowed:

- Cross-check and parallel processing of all ingested images in the database. In this way, several approaches for data processing could be considered, and products from different processing algorithms could be stored for the sake of comparison.
- Ingestion of the data extracted from A24N solutions inside the database as part of the information related to all the images and trails.
- Comparison and convergence of commonly available outputs.

A24N, together with Deimos Processing Tool, has been integrated into the database for automated image processing. Its output was used to cross-check all the images ingested in the database. In each successfully processed image with A24N, at least one streak was detected, including false positives.

3.3 ESA StreakDet Tool

Deimos tested StreakDet tool on the data set obtained during the campaign on ESA request.

As a result, Deimos noticed that whether the image contained trails, a significant number of false positives were detected in most of them. For the images without trails, false detections still occurred. Fig. 14 shows the processed image by StreakDet Tool.

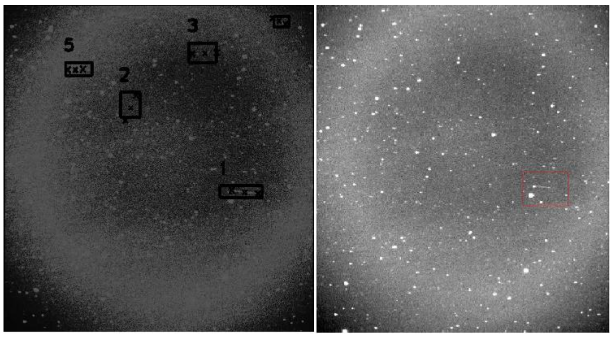

Figure 14: Detected streaks with StreakDet Tool (left) and original image (right)

3.4 Czech Technical University Group

Deimos provided a set of images to the Czech Technical University Group led by Vojtech Cvrcek, on ESA request. They analysed an image dataset detecting objects of interest with a significant angular error for short streaks, with their method being more suitable for longer streaks.

3.5 Manual Analysis

A subset of each sensor's images in a range of 1h (an arbitrary night) was manually checked on ESA request to verify the streak detection algorithm. Each image was verified if it had a trail, and, in the case that it did, length measurements of the trail (in pixel) was calculated and saved in the database including the trail coordinates of its ends. Tab. 2 shows the results of manual analysis for each sensor.

Table 2: Results of manual analysis

Sensor	Num. of images for 1h	Images with streaks	% of images with streaks	Mutual streaks
TRACKER2	393	277	70	Yes
ANTSY1	54	48	88	Yes
COAST	36	22	61	Yes
PIRATE	198	177	86	Yes
PANOPTES-MAM	47	38	80	Yes

4 DATABASE

The database was designed to support the archive of the data acquired during the activity and the future design, test and validation of streak detection algorithms. For that, the interface with such an algorithm was implemented. Visualisation capabilities were also required to ease the analysis of the database contents. It was based on ensuring that bottlenecks were properly tackled (as using very large files), data accessibility and robustness.

In order to comply with the objective, the following requirements and design drivers were identified:

- The database allows the ingestion of different levels of reduction products from the raw images.
- Pre-existing product metadata can also be ingested in the database, as well as the data can be traceable to its source and uniquely identifiable.
- The database allows easy access to the acquired data (at different processing levels) through a user interface.
- The basic image format in the database is FITS format
- The database allows general FITS headers to be included.
- PNG format is included.
- The database allows entering the processed information generated by different algorithms/mechanisms (for example, manually) to be used as a reference for trail detection algorithms). The solution encountered by different mechanism was stored.
- It allows the database's easy and fast extensibility to keep new contents without modifying manually at the database's low level.
- The system provides a REST API to the rest of the world that external third-party tools could use. This REST API provides automatic processing using the database contents.
- Different databases (RDBMS and/or non-RDBMS) are assessed regarding their performance when handling large files; external storage solutions are also evaluated, where only the metadata is kept in the database.
- It provides some minimal protection (through the usage of user roles).
- The STREET User interface accesses the database through this API, but also third-party entities may use the API directly.
- Software developed in this activity is ESA IPR, and it is considered prototype software.

The STREET system solution was implemented following a Service Oriented Architecture (SOA). The core modules that compound the whole system provide Web services to exchange the data and access their functionalities. The Web services are based on the REpresentational State Transfer (REST) architectural style compounding a commonly called RESTful API (Application Programming Interface). The data format to interchange the messages in the REST API is the open-standard (RFC7159) JavaScript Object Notation (JSON). The RESTful API's design and implementation follow the architectural constraints defined in this style.

The STREET system's high-level logical architecture is shown in Fig. 15.

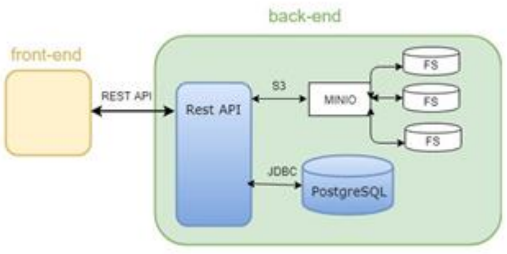

Figure 15: High-level architecture of STREET system

The Front-end is a web-based Human Machine Interface (HMI) to the STREET system. This module is responsible for providing remote access to accredited users to each profile's data and functionalities. All the data and actions to be executed from the front-end are retrieved and requested through the back-end REST API.

The Back-end is the module that, following the SOA paradigm, provides a REST API for data exchange between the persistence and the front-end layers. Additionally, the REST API allows the users to execute on-demand tools for processing products stored in the database. Finally, the back-end authenticates and authorises the users when the user signs in to the STREET system via the front-end.

The Database (PostgreSQL) is the module responsible for the persistence of the STREET system's data.

4.1 Back-end Database Design

PostgreSQL is used as a relational database for storing products metadata and Minio as an object database for files storage.

The relational database (PostgreSQL) stores a product repository or any other information required by the system. It allows making queries over the data to provide high-performance search capabilities. The database contains links to large size files, which are stored in external file storage. The choice of Minio for external file storage resolves the problem of managing large size files.

When a query is performed for obtaining a list of products matching the used constraints, the REST API retrieves links to the external files. External files can be downloaded directly from the Minio database using the provided links.

This sub-module manages the data interchange between the back-end database and the front-end. It handles data insertion (tools and products definitions, sensor configurations...), data browsing (search, filtering, sorting and numerical/graphical representations), data uploads/downloads (sensor observations load, raw data insertion ...), etc. The back-end REST API was built using the usual Spring layers (see Fig. 16) where:

- Controllers layer defines service entry points for each HTTP URL and how parameters are to be read from the HTTP request.
- Services layer contains business logic and provides interfaces for interaction with the repositories and other external Web Services.
- Repositories layer maps the database to/from in-memory domain objects

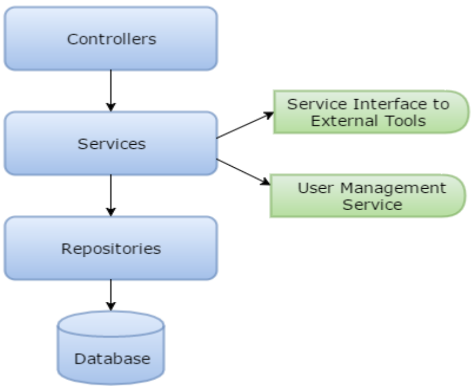

Figure 16: Rest API Architecture Layers

4.2 Front-end Architecture

STREET HMI is a Web-based Human-Machine Interface based on the Angular framework. The application architecture is described in Fig 17.

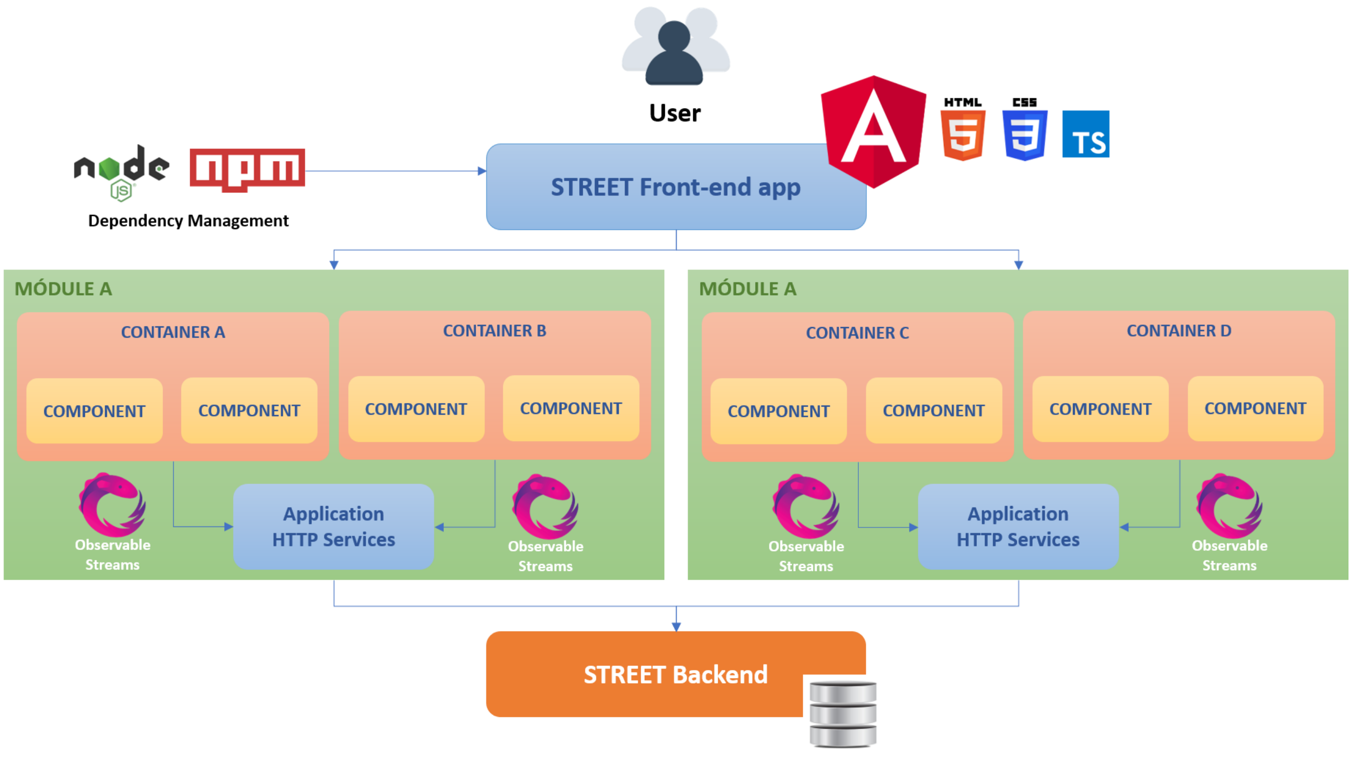

Figure 17: STREET Front-end Architecture

The application has been developed with the following technology:

- NodeJS and NPM for the installation and management of dependencies.
- Angular was a development framework with TypeScript, HTML and CSS.
- RxJS (used to maintain the Observable flow in queries via HTTP with the back-end).

The application was structured in modules that include components and containers (the basic difference was that the containers were those that interacted with HTTP Services, and the child components received input data and sent output data to the container).
The interaction with the back-end was done via HTTP through services, and reactive technology with RxJS was used to maintain the streams' flow.
User authentication was based on JWT (JSON Web Tokens).
The access to data stored in the STREET database was based on the RESTful API standard.
STREET HMI provides the main features described in Fig. 18:

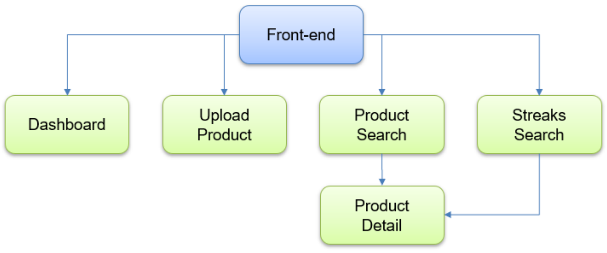

Figure 18: STREET HMI Main Features

The main access view (Fig. 19) gives access to the STREET front-end application using user credentials (email and password).

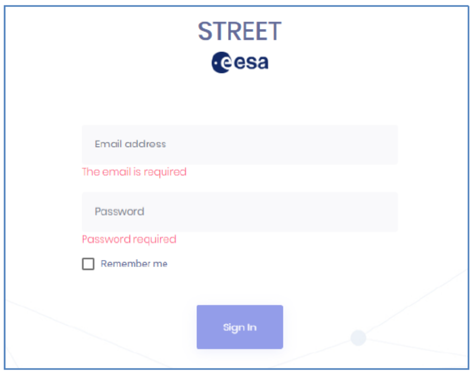

Figure 19: Login View

A dashboard view (Fig. 20) is a reporting tool showing a statistical summary of uploaded products and streak detection

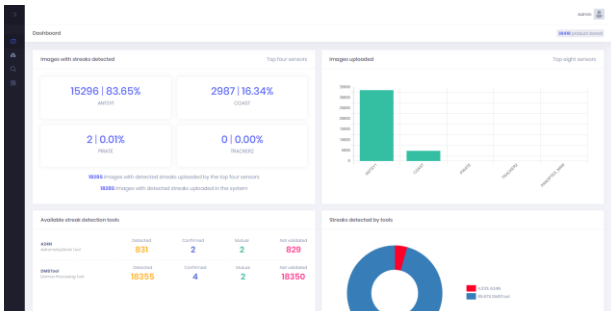

Figure 20: Dashboard view

It is possible to upload one or multiple FITS or JPEG/PNG products in order to be processed by the back-end (see Fig. 21). Choosing only one product view, custom metadata might be added.

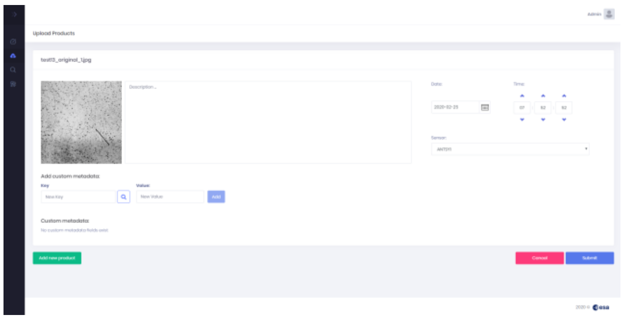

Figure 21: Upload Products view

It is possible to search any supported product applying different filters to get more specific results based on the entered criteria (see Fig. 22).

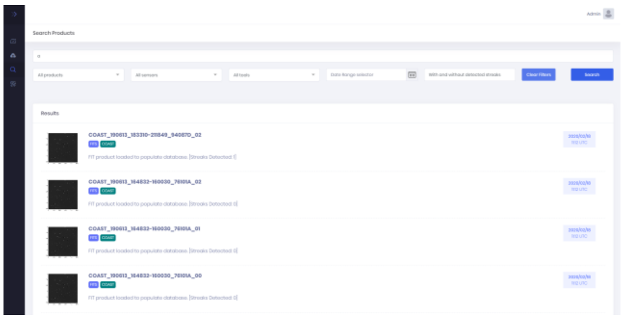

Figure 22: Search Products view

HMI shows detailed preview of a selected product from the Search Products view of one streak detected in the Search Streaks view (see Fig. 23). This view includes some features related to the custom metadata edit, the possibility of validating detected streaks or review the used tools input and streak feature metadata associated with the detection process.

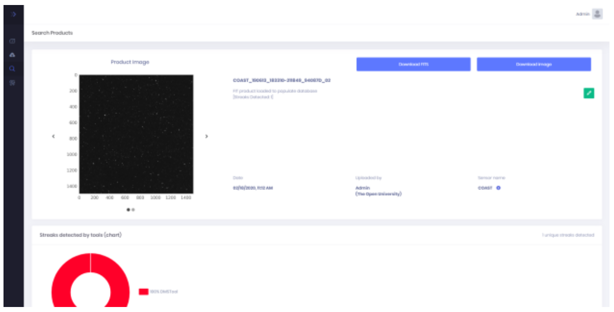

Figure 23: Product detail view

It is possible to search any streak in the STREET database, applying different filters based on the Streak Features related to the streak detection process (see Fig. 24).

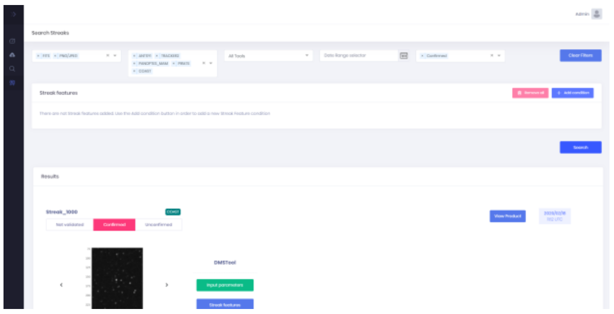

Figure 24: Search Streaks view

5 RESULTS

The average percentage of detected streaks by Deimos Processing Tool and the number of streaks in total are shown in Tab. 3, including the estimated number of images with streak extrapolated from a manually checked set of images.

Table 3: Results of the Analysis with Deimos tool for Tracker2

Sensor	Num. of images	Num. of FITS with at least one streak	% of FITS with streaks	Estimated number of streak images
TRACKER2	53532	26395	49.12	35220
ANTSY1	33717	25456	75.50	23253
PIRATE	23406	15897	67.92	20113
COAST	13141	7511	57.16	7999
PANOPTES -MAM	5400	3914	72.49	5074

TRACKER2 sensor has been observing for 25 nights. Observation nights were almost entirely successful. The average percentage of streaks per night (excluding unsolved FITS files) was 49.12%, and for those which include all FITS files was 46.87%. One of the reasons for the lower % of trails per image may be SExtractor detection's imprecision (as previously mentioned, we had to compromise where a lot of chained and faint trails would not be detected). However, it could also be due to bad, shaken images or nights close to the Moon, Milky Way or cloud base.

For ANTSY1, the average percentage of streaks per night (excluding unsolved FITs files) was 75.50% and for those which included all FITs files was 56.29%. The same reasons as those mentioned for TRACKER2 can cause fewer trails detections. However, it could also lead to the opposite result – a significant number of false positives. In the COAST sensor case, the average percentage of streaks per night was 57.16%, for PIRATE, it was 67.92%, and for PANOPTES-MAM 72.49%. For these three sensors, unresolved images were not provided to Deimos. The total number of trails/nights and detected trails/nights are shown graphically in the following figures (Fig. 25, Fig. 26, Fig. 27, Fig. 28, Fig. 29).

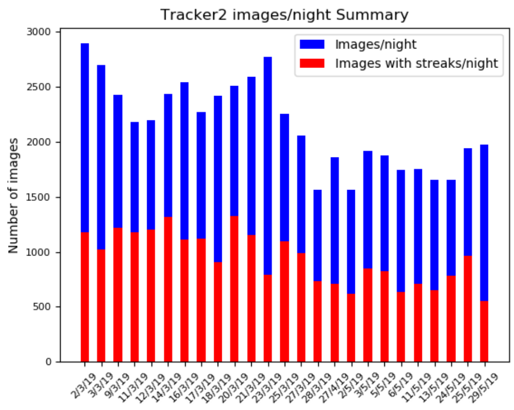

Figure 25: Images per night and the total number of detected trails per night for TRACKER2

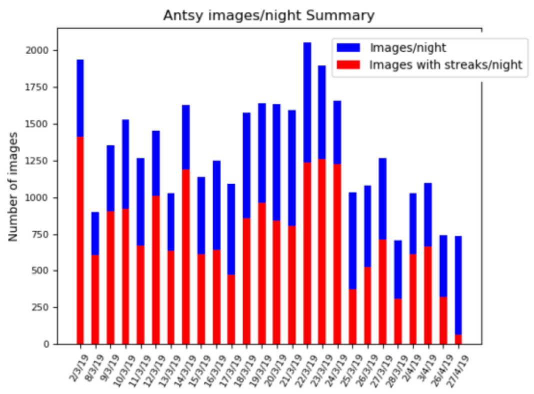

Figure 26: Images per night and the total number of detected trails per night for ANTSY1

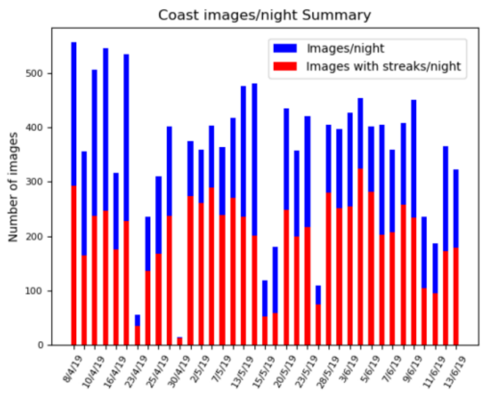

Figure 27: Images per night and the total number of detected trails per night for COAST.

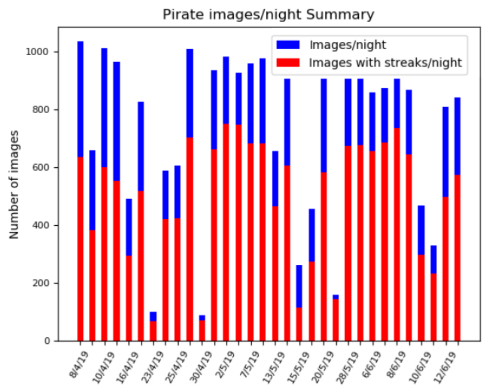

Figure 28: Images per night and the total number of detected trails per night for PIRATE.

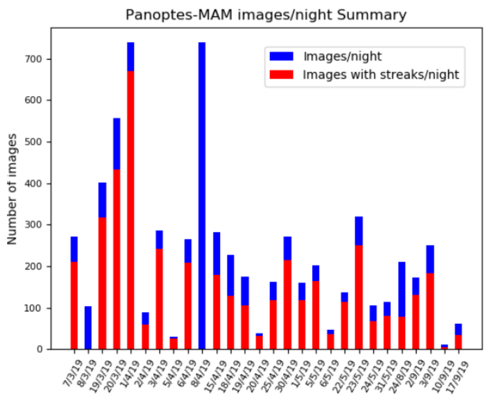

Figure 29: Images per night and the total number of detected trails per night for PANOPTES-MAM.

If SExtractor output catalogue was precise enough (in particular barycentre and angle), trail detection was easier and more accurate. Fig. 30, Fig. 31, Fig. 32 and Fig. 33 show some well-detected streaks for different sensors.

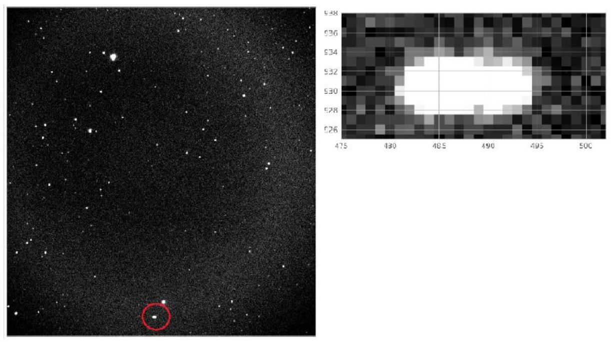

Figure 30: FITs file and image of the detected streak. The image obtained by TRACKER2.

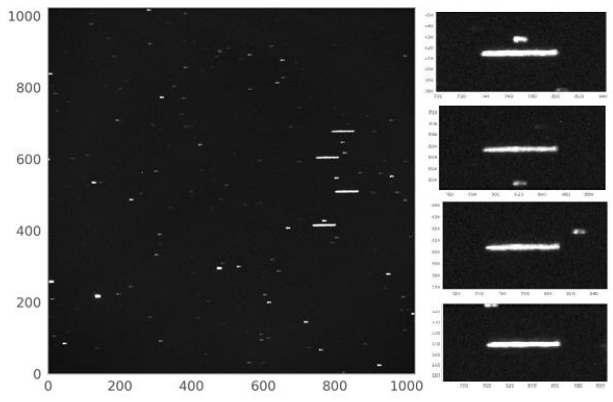

Figure 31: Multiple detected streaks. Image obtained by TRACKER2.

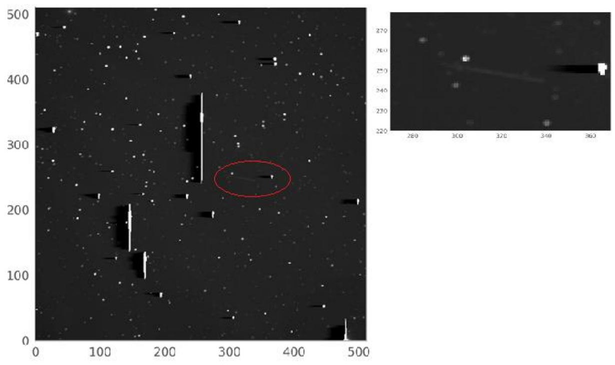

Figure 32: Original FITs file (left) and detected streaks (right). Image obtained by ANTSY1.

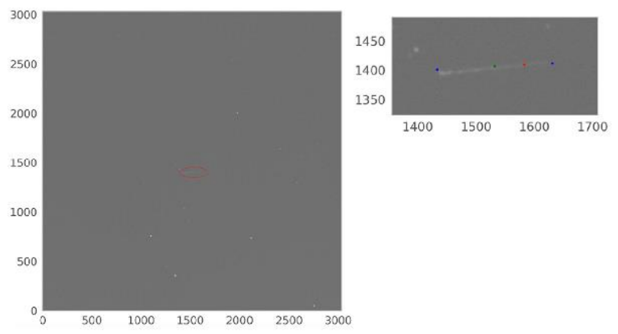

Figure 33: FITs file (left) and cropped image of s detected streaks (right). Image obtained by COAST.

Fig. 34 shows an example of both false-positive (right-top) and real streak (right-bottom) detections in the same FITS.

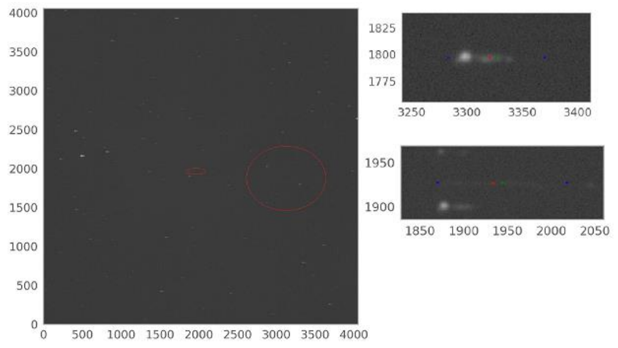

Figure 34: False-positive and real streak detection. Image obtained by PIRATE.

With all given conditions for the observation, false-positive streaks were also detected. Fig. 35 shows false-positive detected streaks in cropped images (right) and ANTSY1 png of original FITS. Fig. 36 is another example of false-positive detections. In the original FITS file, dark shadow-like features around trails displayed. In this case, those trails were stars with smears. These dark shadows happen on frame transfer CCD architectures when stars/trails are very bright and close to saturation.

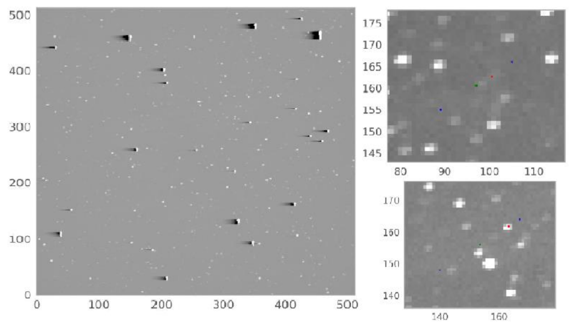

Figure 35: False-positive detections.

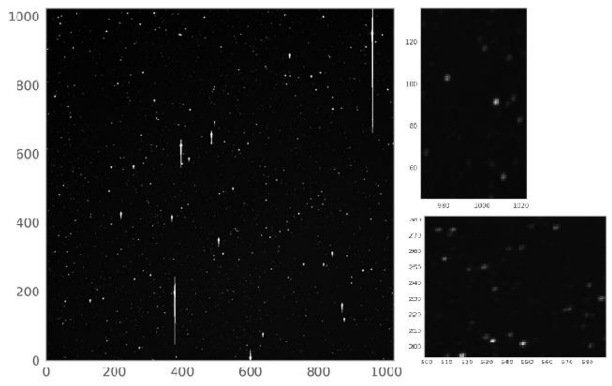

Figure 36: False-positive streak detection in ANTSY1 image.

6 CONCLUSION AND FUTURE WORK

Overall, around 80% of the images contained streaks, whereas streaks were detected automatically for 50-70% of the cases, depending on the sensor features and image conditions. The current database can integrate different streak detection algorithms and compare the output with already implemented software and stored data. This can give us a thorough insight into various tools' capabilities and ideas on achieving the highest possible accuracy in streak detection.

In future work:

- ESA might use STREET database in a challenge to find faint streaks.
- STREET may also be used in preparing for space-based observation endeavours by ESA.

7 REFERENCES

1. Celestrak website https://www.celestrak.com
2. FITS World Coordinate System (WCS) description and conventions https://fits.gsfc.nasa.gov/fits_wcs.html
3. SExtractor website https://www.astromatic.net/software/sextractor
4. CCSDS Hystorical Document, Tracking Data Message, Blue Book, Issue 1, November 2007, Washington
5. CCSDS 503.0-B-1 Cor. 1, Technical Corrigendum 1 to CCSDS 503.0-B-1, Issued November 2007, Blue Book, Issue 1 Cor. 1 Sepmteber 2010
6. CCSDS 502.0-B-2, Orbit Data Messages. Blue Book, Issue 2. November 2009
7. CCSDS 502.0-B-2 Cor. 1, Technical Corrigendum 1 to CCSDS 502.0-B-2, Issued November 2009. Blue Book, Issue 2 Cor. 1 May 2012