Compact Solution for Low Earth Orbit Surveillance

Radu Danescu, Razvan Itu, Mircea Paul Muresan
Computer Science Department
Technical University of Cluj-Napoca
Cluj-Napoca, Romania
radu.danescu@cs.utcluj.ro, razvan.itu@cs.utcluj.ro,
mircea.muresan@cs.utcluj.ro

Vlad Turcu
Astronomical Observatory Cluj-Napoca
Astronomical Institute of the Romanian Academy
Cluj-Napoca, Romania
vladturcu@yahoo.com

***Abstract*—Low Earth Orbit Objects (LEOs) are objects that circle our planet at a distance of less than 2000 km from the surface. Due to their small orbital radius, they move fast and are sometimes affected by atmospheric drag, meaning that their orbit will change in time. This orbit includes communication satellites, Earth observation satellites, but also space debris such as rocket bodies which will eventually reenter the atmosphere. The fast motion, the changing nature of the orbit, their sheer number, and the periodic reentry events, lead to the need of intense observation of their position. This paper presents a compact, portable system for surveillance of the LEO objects. The system is built with commercially available, low-cost items, and is capable of on-site acquisition and real time processing of images. The acquired images are processed by background subtraction, analysis of the difference between frames, extraction of elongated objects corresponding to the satellite streaks, and forming trajectories (tracklets) from consecutive detections. The emphasis on trajectories instead of individual object properties allows successful detection of faint objects, without a significant increase in false positives.**

Keywords—space surveillance, embedded image processing, tracking

I. Introduction

According to the European Space Agency (ESA) vision, a Space Surveillance and Tracking (SST) system detects space objects, catalogues objects, and determines and predicts their orbits. The data generated by an SST system can be used to predict hazards to operational spacecraft, such as a potential collision with a debris objects, or to infrastructure on the ground, in the case of a re-entering object. A SST system can be considered a 'processing pipeline' based on observation data acquired by sensors – the telescopes, radars or laser-ranging stations – and provide derived applications and services, comprising collision warnings, fragmentation detection and re-entry predictions [1].

The Earth Orbiting objects, or satellites, can have a wide range of orbits, of different radii and eccentricities. The Low Earth Orbit (LEO) objects are the closest to the ground, and many of them are visible to the naked eye. They are also the most ubiquitous, they move very fast, both in absolute and in angular (perceived) speeds, and their orbit can change in time due to their proximity to the atmosphere. Also, due to the proximity to Earth, the field of view for observing them is limited.

This orbit includes communication satellites, Earth observation satellites, but also space debris such as rocket bodies which will eventually reenter the atmosphere. The fast motion, the changing nature of the orbit, their sheer number, and the periodic reentry events, lead to the need of intense observation of their position.

II. Related work

Many of these LEO objects are space debris, and they are of special interest for the space agencies. In detecting them, optical approaches are easy to employ. In [2], a survey of observation strategies and image processing techniques is presented, and we can find that the most popular approaches are the ones that track the sky, causing the background stars to be fixed in the image sequence, and the satellite to be perceived as a streak, or the ones which track the previously known object, causing it to be a point in the image and the stars to be the streaks. While the target tracking strategy is easier, the satellite can also have variable intensity over time, or can deviate slightly from its predicted position, as shown in [3].

When the starry background is fixed, the satellite is seen as a linear streak. If detection is desired for a single frame, the streak can be detected using matched filters, as presented in [4] and [5], or using a transformation that emphasizes the linear aspect of the streak, such as the Hough transform [6] or the Radon transform [7], [8]. If multiple frames are available to be processed consecutively, the streaks can be detected as differences followed by validation based on shape [9]. A more complex approach, which assumes neither sidereal tracking (fixed background) nor target tracking, uses image registration to match the stars between frames, is shown in [10].

This paper presents a complete system for space surveillance in the LEO region. The first part describes the compact, low-cost acquisition system and computing platform, and the second part describes the image processing algorithm for detecting the satellite as a sequence of streaks (a tracklet). The main idea is based on detecting differences, but without the use of sidereal tracking, which means that the background is not completely static, followed by geometric properties initial streak validation, and final validation based on the trajectory across multiple frames. The simple approach allows real time processing of large images on the embedded board which also triggers the camera, and yet it is sensitive enough to detect dim satellites from an urban location, and also to have a minimum of false positives even in the presence of clouds.

III. Image acquisition and processing system

A. Hardware architecture

The observation instrument is based on a commercial DSLR camera, Canon EOS 800D, equipped with a 24-megapixel CMOS sensor. The camera is equipped with a Sigma EX 20 wide-angle lens, having a focal distance of 20 mm.

The camera and lens assemble is mounted on a fixed photographic tripod, without any tracking system to compensate for the Earth rotation. While this solution impacts on accuracy, causing the background stars to move slightly between shots and to deviate from the point-like or circular shape for long exposures, it greatly increases the ease of use and portability, as the system can be set up anywhere without any preparation.

The system's core is a nVidia Jetson Nano embedded board, featuring a Quad-core ARM® A57 CPU, 128-core NVIDIA Maxwell™ GPU and 4 GB of RAM. The board features four USB ports, needed for the communication with the camera and with other peripherals, and general purpose I/O pins (GPIO) which are used for camera triggering and for interface buttons. While initially we believed that the processing power of the GPU will be needed for the image processing tasks, the CPU proved to be enough.

The camera is connected to the processing board by two interfaces: the USB interface for image transfer, and a GPIO interface (trigger wire + ground wire, TTL level, active low) for triggering.

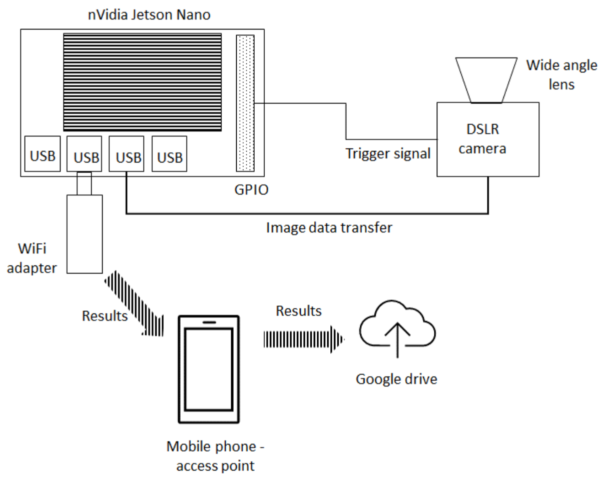

Fig. 1. Hardware architecture of the system.

The system needs to be connected to the Internet for delivering detection results and for time synchronization. The most convenient solution was to equip the board with a WiFi adapter, and connect it to a mobile phone set up as a mobile access point. As the system does not send whole images, but only detection results as text files, it needs neither high speed nor high volume of transferred data. The detection result files will eventually be sent to a cloud storage location, such as Google Drive, where they will be accessible for post-processing and astrometric reduction.

For starting and stopping of the system, simple push buttons were connected to GPIO pins of the Jetson board. Mouse, keyboard and monitor can also be connected for debugging or even on-site programming. The system is powered either by a AC 5V/3A adapter, or by a LiPo 5000 mAh battery, connected to the Jetson board through a step-down voltage stabilizer capable of delivering up to 5 A.

The complete architecture is shown in Fig. 1, and the working assembly is shown in Fig. 2.

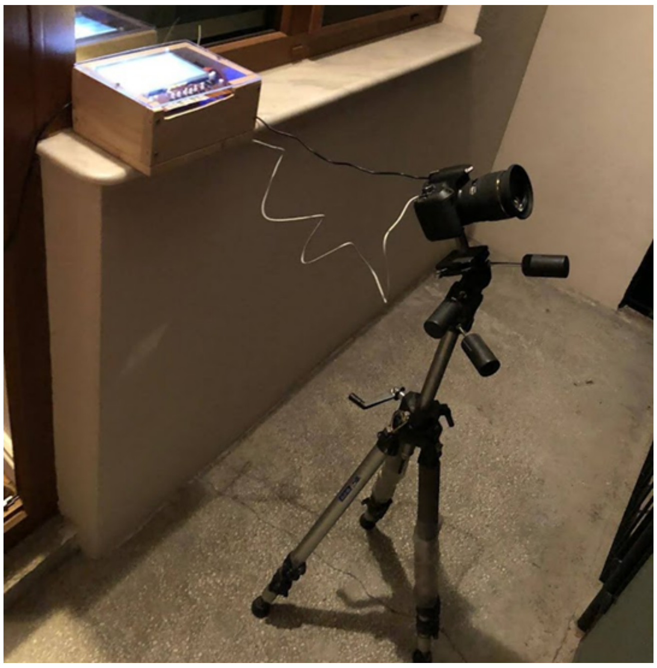

Fig. 2. The assembled system.

B. Software architecture

The main activity of the acquisition and processing system is organized in two threads, as shown in Fig. 3. One thread needs to be accurately synchronized with the system time, which is also synchronized with an external time source (internet time in the current solution, with the possibility of using GPS time for increased accuracy). This thread ensures accurate periodic generation of the trigger signal, which controls the starting time of the exposure and the exposure duration. The camera is set up in the "Bulb" mode, and the exposure is controlled by the pulse length. The exposure length and the interval between shots can be configured by the user. For each start of the exposure interval, the triggering thread registers the current timestamp of the system, to be matched with the image file.

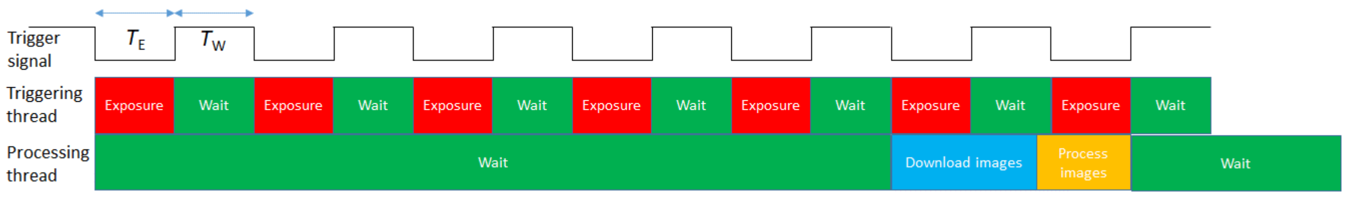

Fig. 3. Time diagram of the exposure and processing threads, including the exposure signal.

Because the camera's image download interface was not designed for real-time transfer after each shot, we need re-mount the drive to discover the newly written files. We have found that this process, if executed too often, will eventually crash the system. Therefore, we have chosen to make the transfer after five images are captured. The processing thread will be started after five exposure triggers have been sent, will read the images, associate to them the timestamps and name the files IMG_YYYY_MM_DD_HH_MM_SS.jpg, so that the timestamp will be recorded directly in the file name. Each captured image will be processed, and the results will be written as a text file, containing the pixel coordinates and the timestamps. Due to the large interval between frames (currently 6 seconds), there is plenty of time for processing, and the processing thread will finish in time for the next batch of 5 frames.

IV. Detection of Satellite Streaks

A. *Extracting movement features*

The basic idea for detecting the satellites is to take advantage of their moving nature, against the (almost) fixed starry background. In a single frame, a satellite will cause a linear streak, as it moves during camera exposure, and in consecutive frames the streak will change position. This establishes the basic strategy: find regions that differ from consecutive frames and see if they are elongated shapes. This approach is not new and has been used by other researchers as well [2][9]. The basic approach can detect clearly visible streaks moving on dark, stationary background, but we want the system to work for faint, short streaks, with uneven background, when the images are taken from the city as opposed to a secluded observatory, without a star tracking mount to make the background fully stationary.

The first problem to be solves is the unevenly lit background. This problem is especially relevant in the case of observing LEO satellites, as they are visible only near sunset or near sunrise, otherwise they are either in the Earth's shadow or the sunlight is too powerful and renders them invisible. Unfortunately, these times are exactly the times the sky is unevenly lit. Figure 4 shows a full frame with uneven background light, and figure 5 shows a detail around a satellite streak.

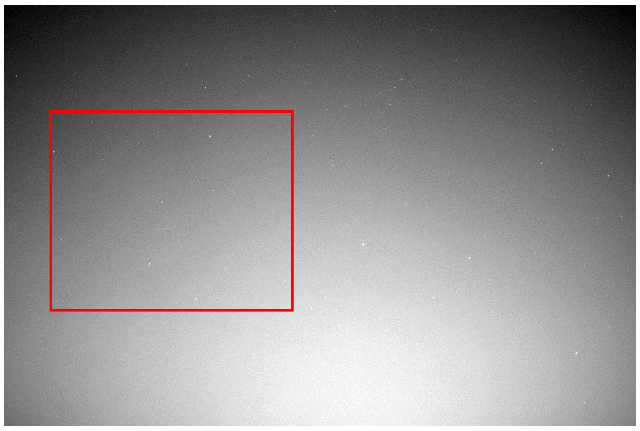
Fig. 4. Captured image, converted to grayscale. The red rectangle shows the area around the satellite streak. The contrast is enhanced for display, but not for processing.

Fig. 5. Detail showing the satellite streak against the unevenly lit background. Contrast enhanced for display – the intensity of the streak is only 5-6 units above the local background.

Some researchers [5] model the background light as a parametric surface. However, we have found that a median filter of a sufficiently large size will extract the light model accurately, as shown in figure 6.

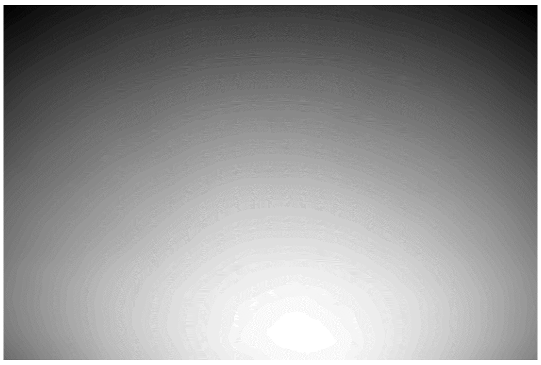
Fig. 6. Background light model, extracted by median filtering.

Thus, assuming I_t to be the captured image at time *t*, we subtract the background B_t, to get the dark-background image D_t.

$$B_t = median(I_t, 55) \quad (1)$$

$$D_t = I_t - B_t \quad (2)$$

A detail of the dark image is presented in figure 7.

Next, the moving features in the image are emphasized by subtracting the current background-free image D_t from the past image D_{t-1}, obtaining the movement image M_t.

$$M_t = D_t - D_{t-1} \quad (3)$$

All subtractions so far are performed using saturation, meaning that any pixel difference below 0 is set to 0. The movement image is then thresholded with a very low threshold (in our current implementation we use T=2, set experimentally, meaning that each difference that is above or equal to 2 will be set to 1 in the binary image). This way, any small variation of intensity is taken into consideration.

The result for our detail window is seen (in negative) in figure 8.

Fig. 7. Detail region after background subtraction.

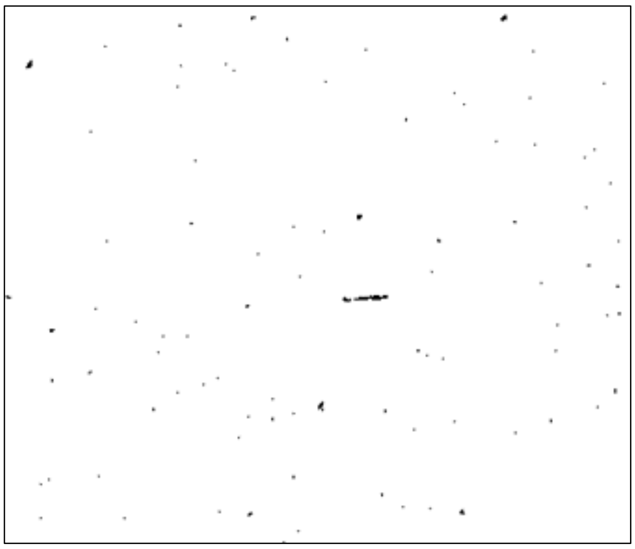
Fig. 8. Binary image, following thresholding of the movement image.

As we can see, many features, including parts of the stars, are present in the binary image. This is due mostly to the fact that we do not use a star tracking mount for the camera, to compensate for the Earth's rotation. The satellite streaks will be extracted by further analyzing the size and shape of the binary objects.

B. Extracting streak candidates

The binary image is further processed by extracting the connected components (labeling). First, for each connected component we compute the area (the number of pixels included in the object), and the ones with a very small area (in our implementation the area threshold is 15 pixels) are discarded. For the remaining objects, we compute the following geometric properties:

- Center of mass,
- Major ellipse axis length,
- Minor ellipse axis length,
- Eccentricity, computed from the axes' lengths,
- Orientation angle.

The objects having the major axis length above 30 pixels, and the minor axis length less than half the major axis, and the eccentricity above 0.95 will be selected as streak candidates.

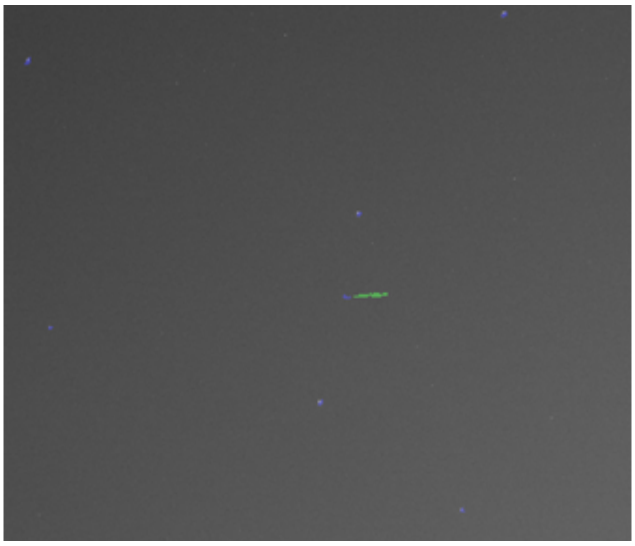
Fig. 9. Objects classification: the streak candidate is shown in green, and the other large area objects in blue.

For every possible streak candidate, we'll compute three key points, which will define the streak as a line segment. The first point is the center of mass, already known, which will be denoted as O, having the coordinates x_O and y_O. The other two points are denoted as A and B, and their coordinates are computed by starting from the streak's center and going along its orientation, progressively increasing the distance from the center r until the current point passes beyond the pixel set of the streak. The search is performed in both directions, and therefore the coordinates of the streak ends are computed as:

$$x_A = x_O - r_A \cos\varphi \quad (4)$$

$$y_A = y_O - r_A \sin\varphi \quad (5)$$

$$x_B = x_O + r_B \cos\varphi \quad (6)$$

$$y_B = y_O + r_B \sin\varphi \quad (7)$$

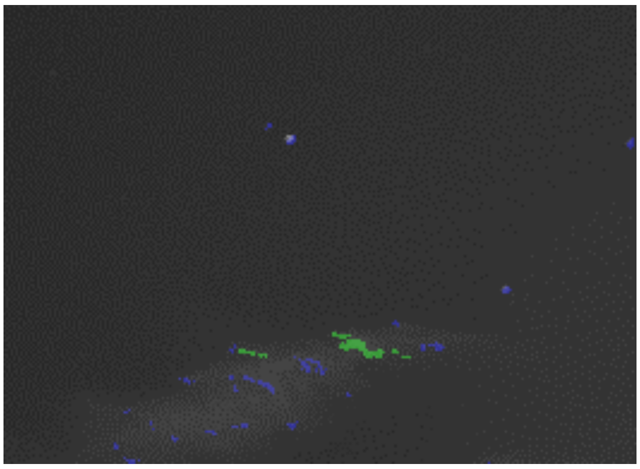
Fig. 10. False streaks caused by clouds.

The conditions for the streak candidate are not strict, because the distance of a LEO from the observation site varies greatly, along with the perceived angular speed, and therefore the streak can be as short as 30 pixels or as long as 200. The variable brightness of the satellite also affects the thickness of the streak. Thus, strict criteria will lead to a lot of missed targets. However, our lax criteria will lead to a lot of false positives when other moving features are present in the image, such as clouds, as seen in figure 10. In order to have the best of both worlds, high sensitivity and low number of false positives, we'll validate the streaks by their trajectory.

C. *Trajectory analysis and tracklet formation*

The trajectories will be formed based on individual, consecutive detected streaks. Each streak will be defined by the center point O, and the end points A and B, therefore we can define two types of distances between two streaks:

- The Euclidean distance between centroids, due to velocity, d_V;

$$d_V = \sqrt{(x_O - x_{O'})^2 + (y_O - y_{O'})^2} \qquad (8)$$

- The distances between one streak's points and the line defined by the other streak, d(P, A, B), where P can be either the centroid of the new streak, O', or one of the end points, A' and B'.

$$d(P,A,B) = \frac{|(x_B - x_A)(y_A - y_P) - (x_A - x_P)(y_B - y_A)|}{\sqrt{(x_B - x_A)^2 + (y_B - y_A)^2}} \qquad (9)$$

The two types of distances are shown in figure 11.

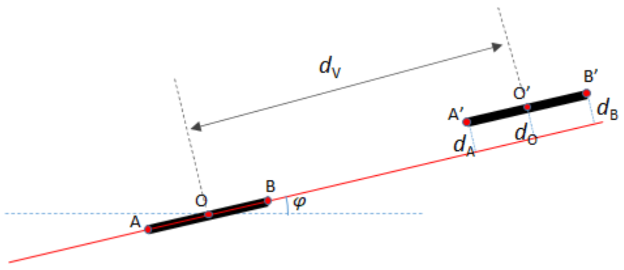

Fig. 11. Computing distances between two streaks.

A tracklet is a sequence of streaks that depict the same LEO object at different moments in time. A tracklet is started when a new streak is detected and it cannot be associated to an existing tracklet.

The tracklet can have the following states:

State 0 – empty tracklet.

State 1 – a new streak has been detected, and a new tracklet has been initialized. In this state, the speed of the streak is not yet known, and neither is the orientation along the trajectory line. This state is depicted in figure 12: the gray areas are the locations of the possible new streaks to be matched to this track. Even though we don't know the speed of the tracklet, we can assume that it is similar to the length of the streak, as the exposure time is equal to the time between frames. Thus, we will impose an acceptable interval for the distance d_V.

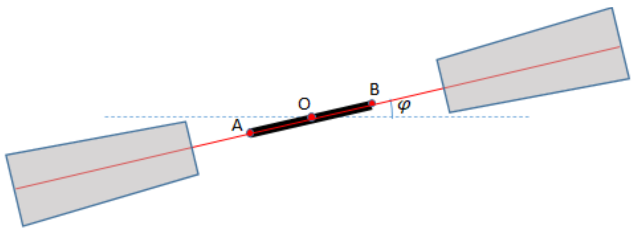

Fig. 12. A tracklet in state 1: a single streak is detected, and the following streak can be on either side.

State 2 – at least two streaks are associated to the tracklet, and therefore the speed and the orientation of the trajectory are known. In this state, the area allowed for the new detections is limited to the side pointed by the speed vector, as shown in figure 13.

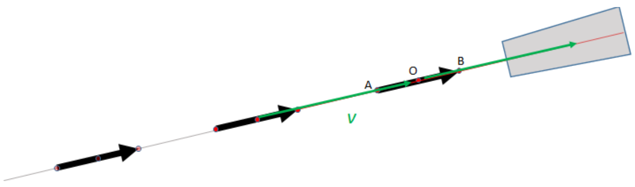

Fig. 13. A tracklet in state 2: the tracklet now has speed and orientation, and new detections can only match in one direction.

State 3 – a state 2 tracker will eventually pass beyond the borders of the image, or no new streaks will be added to it for a significant number of frames. This state represents a "closed" tracker, which will be delivered as output.

The process is illustrated step by step in figures 14 – 17. We can also see false streaks caused by clouds, which are not transformed into tracklets.

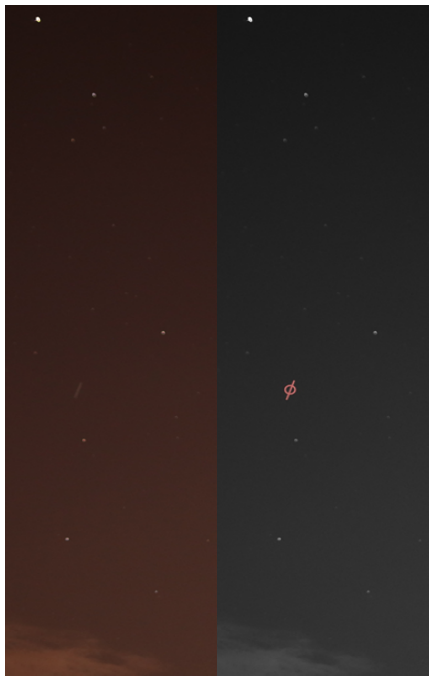
Fig. 14. A newly detected streak initializes a tracklet, which goes into state 1.

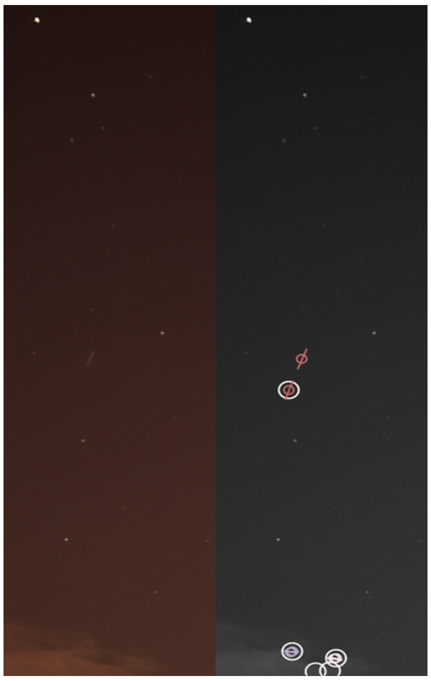
Fig. 15. A second detection is assigned to the state 1 tracklet, which now goes into state 2.

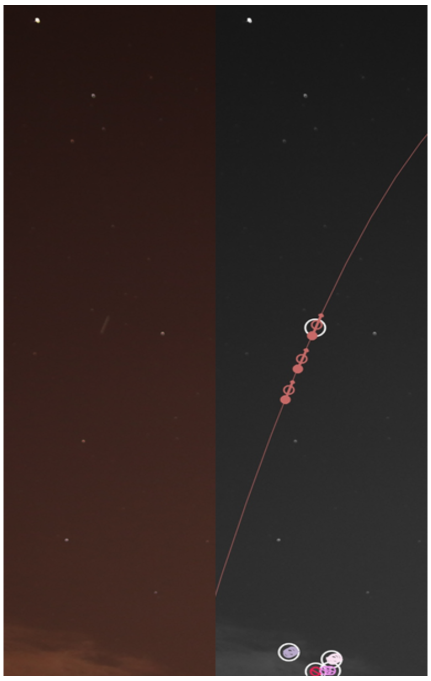
Fig. 16. A state 2 track gets a new measurement. A trajectory curve can now be fitted to the tracklet (for display purposes only for now).

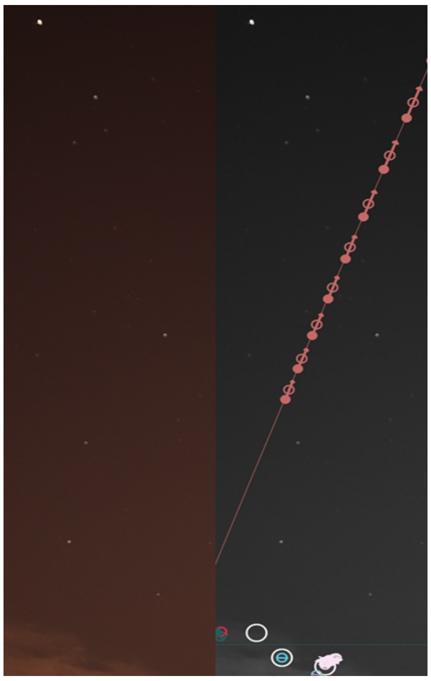
Fig. 17. The tracklet passes the boundary of the image and goes into state 3 (finished).

V. Tests and Results

A. Test scenarios

The system was tested in challenging conditions, from an urban location in Cluj-Napoca, usually in uneven sky illumination conditions near the sunset. Many times the observed area of the sky had moving clouds. Each test sequence consists of 1000 frames. Initially the frames were acquired with a 5 second exposure, and 10 seconds between exposure starts, but later this time was reduced to a 3 second exposure and 6 seconds between exposures. The main reason for reducing the exposure time was the observation of the Starlink satellite groups, which were very close together and a 5 second exposure time led to the merging of the streaks of multiple satellites into a single streak.

For the average sequence length of 1h40m (at 6 seconds interval between frames) the system typically detects around 40 satellites, while covering an angular sky area of 60x40 degrees. Figure 18 an example of trajectories extracted during a typical observation period.

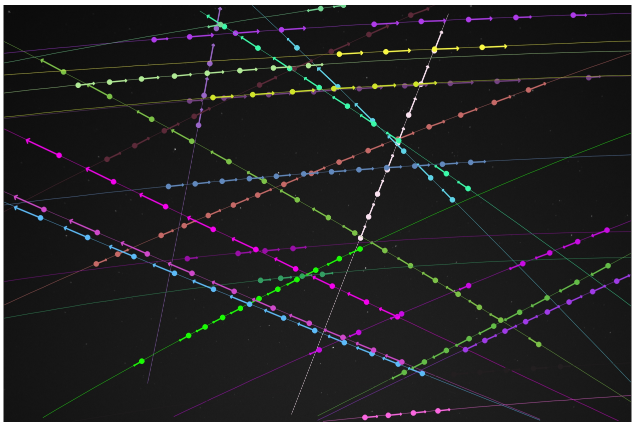
Fig. 18. Satellite trajectories extracted in a typical observation session.

B. *Limitations of the detection system*

The satellites need to be visible to be detected, therefore they need to be slightly brighter than the background sky. We have recorded sequences which start immediately before sunset, and we have found that the system starts to detect when the ambient light is still significant (for a 0…255 intensity, the system starts detecting when the average sky intensity is 65). The streaks need to be just 2-3 units brighter than the surrounding background to be reliably detected.

A very difficult challenge is the presence of clouds. While the system works reliably when a few cloudy areas are present in the image, some false positives will be validated as tracklets when the cloudy area is large, as seen in figure 19. We can also see that the number of streak hypotheses vastly outnumbers the number of validated tracks by more than 100:1.

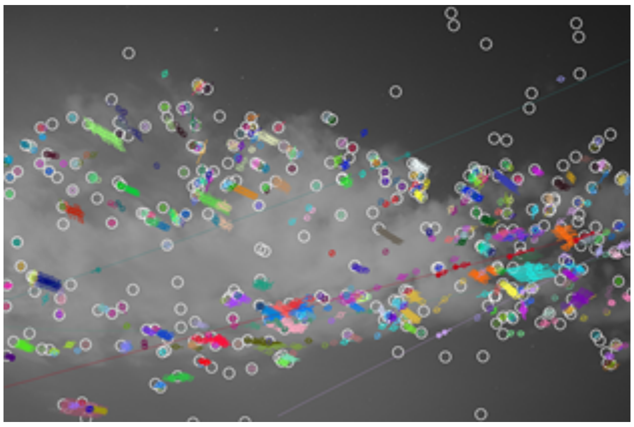
Fig. 19. Cloud dominated scenario: the false positives cannot be all avoided.

C. *Preliminary testing the accuracy of the detection*

The detection results must be converted into angular coordinates (Right Ascension – RA, Declination – DEC), to be useful for space surveillance. Since this work is preliminary, this conversion, called Astrometric Reduction, was done offline, using the reduction tool from Astrometry.net [11]. The converted coordinates were then compared to the predicted coordinates of a known satellite. Most of the satellites are tracked by the US and EU authorities, and their known parameters are described in the TLE (Two Line Element) format. Based on this format, specialized software such as TheSkyX [12] can be used to predict the angular coordinates at specific moments of time.

The test scenario satellite is a space debris, a rocket body having the following TLE description:

```
0 CZ-2C R/B
1 37766U 11039B  21158.87867492  .00000036  00000-0  15813-4 0  9992
2 37766  98.3347  24.6498 0045379 348.6624  11.3553 14.66571447527545
```

The object’s distance from the observation point was higher than 1000 km, which caused its contrast in the image to be low (the average intensity of the streak was 44 units, while the background intensity was 38 units, with a standard deviation of 6), and the length of the streak to be less than 40 pixels. A detail showing the streak in one of the frames is shown in figure 20. One advantage of a distant satellite is, however, the low apparent angular motion, which allow us to detect it in many frames, as shown in figure 21.

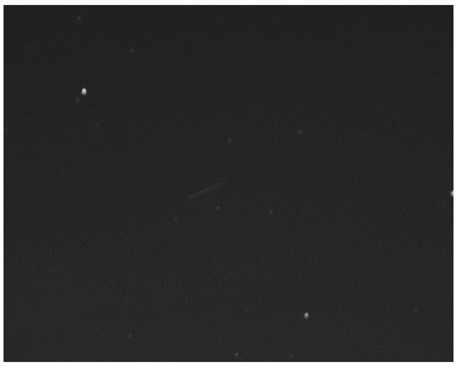
Fig. 20. Detail streak of CZ-2C R/B.

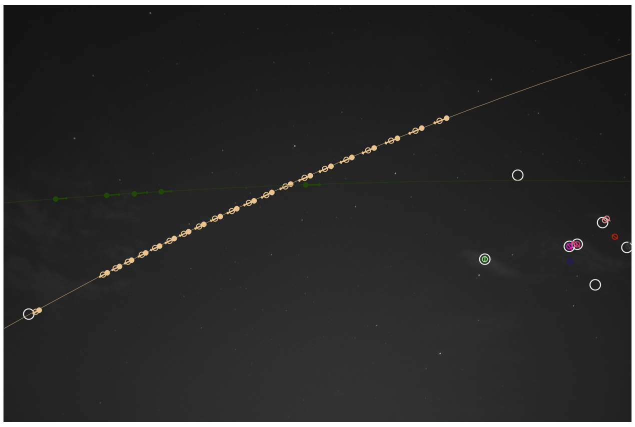
Fig. 21. Complete trajectory of CZ-2C R/B.

The angular coordinates estimated based on the detected tracklet are shown in figures 22 and 23, against the frame number since detection, and they clearly match the predicted trajectory. The errors, of arc minute order of magnitude (for a 40x60 degrees field) are shown in figures 24 and 25. The errors are larger at the edge of the image due to the lens distortion.

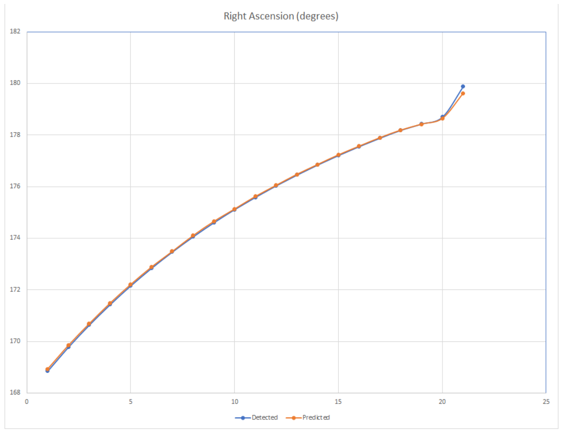

Fig. 22. Detected vs predicted Right Ascension angles.

D. *Execution time*

The execution time per frame, on board of the nVidia Jetson Nano, is between 300 and 400 ms, without the use of GPU or other specific optimizations. Taking into consideration that the interval between frames is 6 seconds, plenty of processing time is left available, which means that we can also perform, in the future, astrometric reduction directly on the Nano, or we can choose a cheaper and less powerful solution for processing, such as a Raspberry Pi board.

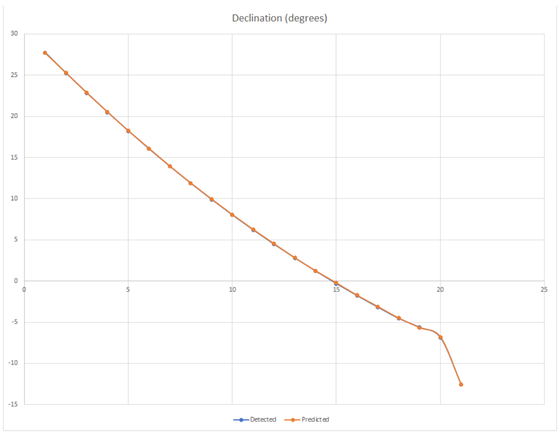

Fig. 23. Detected vs predicted Declination angles.

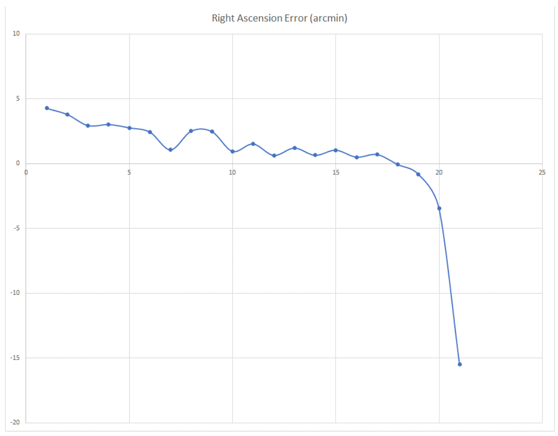

Fig. 24. Error of Right Ascension (arc minutes).

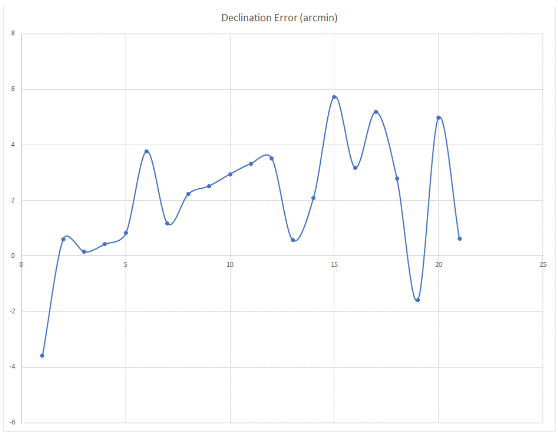

Fig. 25. Error of Declination (arc minutes).

VI. Conclusions and Future Work

We have presented a compact, lightweight space surveillance solution based on off the shelf components, which can be easily set up anywhere to detect Low Earth Orbiting objects. To make up for the lack of perfect observation conditions, and to allow for low contract streaks to be detected, the system relies mostly on coherent trajectories instead of the geometric properties of the streaks, thus reducing the number of false positives even without sidereal tracking systems, and in the presence of clouds. The detection works in real time, on board of the acquisition system.

The future work will be focused on reducing the number of false positives in severe conditions by imposing additional constraints on the tracklets, merging tracklets if they are detected as fragments for the same satellite, and performing automatic astrometric reduction on board of the acquisition system.

Acknowledgment

The work was supported by a grant of Ministry of Research and Innovation, CNCS–UEFISCDI, project number PN-III-P2-2.1-PED-2019-4819, within PNCDI III.

References

[1] European Space Agency, “Space Safety”, online: https://www.esa.int/Our_Activities/Space_Safety/Space_Surveillance_and_Tracking_-_SST_Segment, accessed: 2021.

[2] E. Stoveken, T. Schildknecht, “Algorithms for the optical detection of space debris objects”, Proceedings of the 4th European Conference on Space Debris, Darmstadt, Germany, pp. 637-640, 2005.

[3] S. Maksim, “A comparison between a non-linear, poisson-based statis-tical detector and a linear, gaussian statistical detector for detecting dim satellites”, Advanced Maui Optical and Space Surveillance Technologies Conference, p. 44, 2012.

[4] R. Sara, V. Cvrcek, “Faint streak detection with certicate by adaptive multi-level bayesian inference”, European Conference on Space Debris, 2017.

[5] M. P. Levesque, S. Buteau, “Image processing technique for automatic detection of satellite streaks. Technical Report”, DEFENCE RESEARCH AND DEVELOPMENT CANADA VALCARTIER (QUEBEC), 2007.

[6] F. Diprima, F. Santoni, F. Piergentili, V. Fortunato, C. Abbattista, L. Amoruso, “Efficient and automatic image reduction framework for space debris detection based on gpu technology”, Acta Astronautica, vol. 145, pp. 332-341, 2018.

[7] A. Ciurte, R. Danescu, “Automatic detection of meo satellite streaks from single long exposure astronomic images”, 2014 International Conference on Computer Vision Theory and Applications (VISAPP), pp. 538-544, 2014.

[8] P. Hickson, “A fast algorithm for the detection of faint orbital debris tracks in optical images”, Advances in Space Research, vol. 62, 3078-3085, 2018.

[9] R. Danescu, F. Oniga, V. Turcu, O. Cristea, “Long baseline stereo-vision for automatic detection and ranging of moving objects in the nightsky”, Sensors, vol. 12, pp. 12940-12963, 2012.

[10] H. N. Do, T. J. Chin, N. Moretti, M. K. Jah, M. Tetlow, “Robust foreground segmentation and image registration for optical detection of geo objects”, Advances in Space Research, vol. 64, pp. 733-746, 2019.

[11] D. Lang, D. W. Hogg, K. Mierle, M. Blanton, S. Roweis, “Astrometry.net: Blind astrometric calibration of arbitrary astronomical images”, The Astronomical Journal, vol. 139, pp. 1782–1800, 2010.

[12] Software Bisque, “TheSky Professional”, online: https://www.bisque.com/product/theskyx-pro/, accessed 2021.