Advanced Algorithms for Segmentation of Space Debris Astronomical Images

Daniel Kyselica
Faculty of Mathematics Physics and Informatics
Comenius University Bratislava
842 48 Bratislava, Slovakia
daniel.kyselica@fmph.uniba.sk

Stanislav Krajčovič
Faculty of Mathematics Physics and Informatics
Comenius University Bratislava
842 48 Bratislava, Slovakia
stanislav.krajcovic@fmph.uniba.sk

Jiří Šilha
Faculty of Mathematics Physics and Informatics
Comenius University Bratislava
842 48 Bratislava, Slovakia
jiri.silha@fmph.uniba.sk

Roman Ďurikovič
Faculty of Mathematics Physics and Informatics
Comenius University Bratislava
842 48 Bratislava, Slovakia
durikovic@fmph.uniba.sk

2022 26th International Conference Information Visualisation (IV) | 978-1-6654-9007-8/22/$31.00 ©2022 IEEE | DOI: 10.1109/IV56949.2022.00063

***Abstract*—During astronomical observations, images of selected part of the sky are made by the Slovak 70cm telescope specialized on space debris tracking. Every pixel of this frame can be represented by three data: position on the horizontal X axis, vertical Y axis, respectively and the intensity value that can range from 0 to 65536. The intensity value in the order of thousands or higher indicates presence of an orbital or extraterrestrial object such as a star, planet, space debris, or even electromagnetic field interference, celestial plane background and other artefacts.**

In this paper, we present the methodology and proof of concept of our design for processing of astronomical images and a novel space debris tracklet building process using a machine learning method by exploiting Long Short Term Memory (LSTM) architectures. Machine learning models need a fair amount of data examples for training. However, there are not enough sequences captured by the telescope, therefore we train a neural network with synthetic artificial training data based on known sky observations. Information about moving objects in the Earth's orbit is visualized as sequences of positions in time.

***Index Terms*—LSTM architectures, neural network, object tracking, processing of astronomical images, space debris**

I. Introduction

Even though there are novel policies that all new satellites must be "cleaned up", the old ones simply have no capabilities to de-orbit themselves. These factors, and many other, contribute to the rising population of non-functional artificial objects, which, in turn, increases the probability of creating new space debris.

However, there exist solutions for this problem. One of them is active debris removal [1], involving satellites that actively and purposefully target selected objects and clean them up. Unfortunately, such an approach requires the known tracklet of the object. A tracklet is a data structure containing consecutive observations of an object in time, we can refer to the tracklet also as the trajectory of the object. The tracklet is often the result of astrometric measurements, which yield, information about an object's position, velocity, trajectory, etc. [2], [3].

As we are observing only a small part of the sky and due to a small field of view the object's orbit appears as a line. Therefore, we can represent an object's trajectory by a line segment. Blue points represent positions of moving objects in time and red points represent positions that are of other objects in Fig. 1. The problem is to find the set of blue points that can be interpolated by a line representing the trajectory of an object in time.

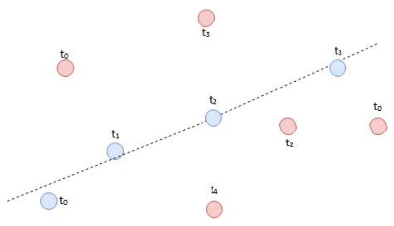

Fig. 1. A trajectory and tracklet of an object. The observed object positions are shown as blue points.

ESA Plan For European Cooperating States

II. Related Works

The existing system, developed by Krajčovič et al. [2], [4], is capable of tracking objects from a series of observations. The current analytic solution might produce false-positive tracklets, due to its architecture. Because of the nature of the system, false positives are undesirable. We propose utilizing a neural network to create tracklets that can remove the downsides of the current system.

The position of an object after astrometric reduction is given in equatorial coordinates, see Fig. 2, representing the positions of objects on the celestial sphere [5].

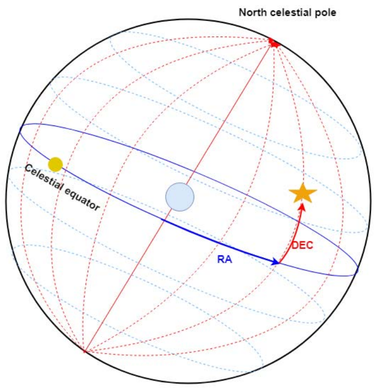

Fig. 2. Equatorial coordinates of a star on the celestial sphere.

The novelty of the paper is the **tracklet building** process using neural networks and its validation with the ground truth analytic solution [6]. The analytic solution uses linear regression to create tracklets. A neural network will be used to find the positions of a moving object from the input image sequence. This approach improves the system in two ways:

- We can detect different types of object trajectories using trained networks without a need to change the design of the algorithm.
- The proposed methodology returns a confidence value for each tracklet. This helps us detect false positive examples.

Support-vector machine (SVM) model is a supervised machine learning model used for classification. It projects original input data into a higher dimension and performs maximal-margin classification in this higher dimension space [7]. We can train SVM to classify input tuples of size k into two classes: *On line class*. This class represents tuples of points with size k laying on a line with an equal distance between points. Every two adjacent points in a tuple have to be equidistant and lay on the same straight line. *Not on line class*. This class represents tuples with size k, which are not colinear.

Finding moving objects using SVM method turns out to be unreliable. While this method is simple, it comes with a cost of a large number of points in *Not on line class* using real data. SVM classifier trained on 50 thousand synthetic input tuples with size 3 had above 96% accuracy. Only 3% of validation data were classified as a false negative. We considered a single moving object in the real world data, as a result, we classified many tuples belonging to *On line class*, but only one truly belongs there.

With so many false-negative examples we are not able to detect a true, moving object. Due to these problems, SVM method has been rejected.

The paper is structured as follows: in Section III we introduce the concept of machine learning used in trackled building process, then in Section IV we propose the model prediction step as a trained neural network and set several network parameters, we move on with system behavior and validation in Section V, and conclude with Section VI.

III. Methodology of Trackled building using Machine Learning

The recurrent neural network (RNN) model is a supervised learning model commonly used for time series forecasts and in natural language processing. In RNNs, information about previous inputs is accumulated in the hidden state. This information is then used to predict the next position of a point in time [8] chapter 10.

We can use Long short-term memory (LSTM) cells to predict the current position of a moving object from previous positions [9] we can analogically use the predicted positions to build tracklet of a moving object. LSTM is a good choice for this problem [10].

A. *Long Short-Term Memory - LSTM*

The RNNs accumulate information over a long time. However, in some cases, we want to forget part of the information we learned. For example, if a sequence consists of sub-sequences. Information from the previous sub-sequence accumulated in the hidden state can lead to a wrong prediction about the next sub-sequence. This problem was solved by introducing gated units like LSTM, Fig. 3. A gated unit uses gates that control the amount of information passing through. LSTM has two inner states. The cell state $c^{(t)}$ represents a

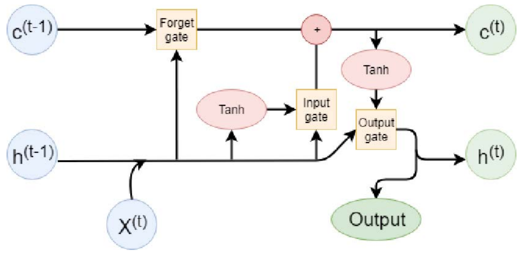

Fig. 3. Illustration of LSTM unit. Arrows represent information flow. Blue cells represent inputs, green cells represent outputs. Red cells stay for simple operation on data. Yellow cells represent gates.

path for information to run down through the whole network like lateral connections in leaky units. The hidden state $h^{(t)}$ represents the previous output of LSTM. LSTM has three gates: input, output, and forget gate, respectively. Each gate is a point-wise multiplication of the first input and output of nonlinear function from the second input, Fig. 4. A gate has two weight matrices U for the input vector and W for the hidden state vector. An input $x^{(t)}$ is at first concatenated with hidden state $h^{(t-1)}$ and goes with $c^{(t-1)}$ through the forget gate. Let $f^{(t)}$ be the output from sigmoid layer σ from the gate:

$$f^{(t)} = \sigma\left(bias^f + \sum_j U^f_{i,j} x^{(t)}_j + \sum_j W^f_{i,j} h^{(t-1)}_j\right) \tag{1}$$

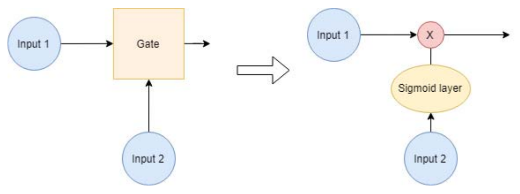

Fig. 4. The gate.

The next cell state is then computed [10]:

$$I^{(t)} = \sigma\left(bias^I + \sum_j U^I_{i,j} x^{(t)}_j + \sum_j W^I_{i,j} h^{(t-1)}_j\right) \quad (2)$$

$$l^{(t)} = tanh\left(bias^l + \sum_j U^l_{i,j} x^{(t)}_j + \sum_j W^l_{i,j} h^{(t-1)}_j\right) \quad (3)$$

$$c^{(t)} = f^{(t)} c^{(t-1)} + I^{(t)} l^{(t)}, \quad (4)$$

where $I^{(t)} l^{(t)}$ is the output of the input gate and $l^{(t)}$ is the LSTM internal unit. Next hidden state is computed

$$O^{(t)} = \sigma\left(bias^O + \sum_j U^O_{i,j} x^{(t)}_j + \sum_j W^O_{i,j} h^{(t-1)}_j\right) \quad (5)$$

$$h^{(t)} = tanh(c^{(t)}) O^{(t)}. \quad (6)$$

LSTM networks have been shown to learn long-term dependencies more easily than the simpler recurrent neural networks. One disadvantage of LSTMs is the training time, even with relatively small input and small hidden dimensions.

B. Tracklet Building System

The overall tracklet building system is described by flow chart in Fig. 5. An **input sequence** contains a list of positions in text form for each corresponding image. The **Masking** step removes stationary objects such as stars from the input sequence. From the first two images in lists, we **Create tuples**. Tracklets are later created from these tuples. In the next step, the **Model prediction** predicts the next position for each tracklet in the next image. For the prediction of the next position, we propose an LSTM network in the next Section IV. The **Matching function** finds the closest positions from the next image to already predicted positions. If the function is not able to find any position within the closed neighborhood, the tracklet marks the position as missing. If we have already predicted the **last position** we return the remaining tracklets. Otherwise, if at least k **positions in a row are marked as missing** in tracklet we discard that tracklet. The remaining tracklets are updated. The **Update Tracklet** step updates tracklet's confidence. The tracklet contains predicted and matched positions in order. The matching confidence number represents the ratio between the number of correctly found positions to the total number of positions. On the other hand, the vector confidence number is given by Eq. 7, where $\vec{t}$ is the true vector and $\vec{p}$ is the predicted vector.

$$\text{vector\_confidence} = \frac{\|\|\vec{t}\| - \|\vec{t} - \vec{p}\|\|}{\|\vec{t}\|} \quad (7)$$

Then the model will predict positions for tracklets again using the LSTM network, described in the next section.

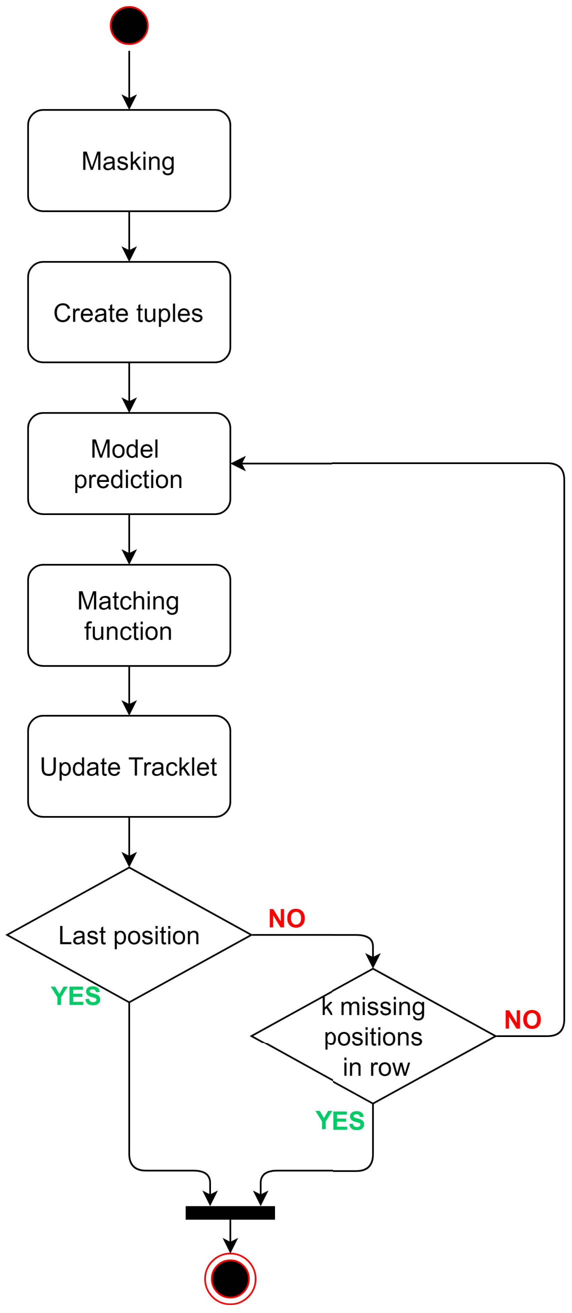

Fig. 5. The block diagram of the tracklet building system. A system input is a sequence of positions and tracklet is the output. Prediction model is based on LSTM neural network.

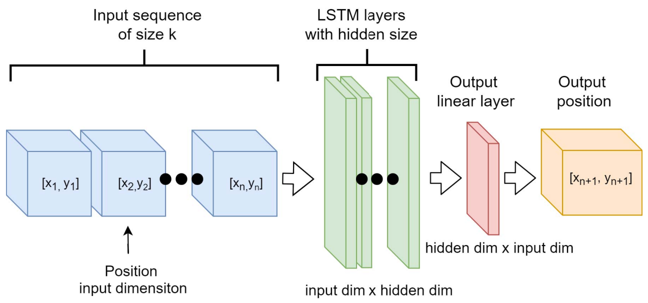

Fig. 6. Topology of the neural network. The input sequence is processed by LSTM layers. Output is produced by linear layer to transform LSTM output to the input dimension space.

IV. Position Prediction with LSTM Neural Network

The model prediction step is a trained neural network. For each tuple of positions and the tracklet, the model outputs the predicted position in the next image. An input to model has to be normalized and similarly, the model output is normalized, too. To get back the real positions in an image we need to denormalize output positions. We normalize data by subtracting the minimum for each coordinate and dividing by variance for each coordinate.

To function properly, the neural network has to be designed well. For a trajectory forecast, we chose a network with LSTM cells. It accumulates information about the movement of an object from its previous positions to predict the current position in the image. The neural network topology is illustrated in Fig. 6. Positions from an input sequence are fed into LSTM layers. LSTM projects points from **input dimension** into **hidden dimension** space. A nonlinear activation function (tanh, ReLU) [8] is applied before entering the linear output layer. The output layer projects data from the hidden layer back to the input space. Thus, the result is a vector of the same dimension as the input.

A. Data generator

Training data are crucial for neural network performance. A large amount of data is needed for the network to be properly trained. In our case, only an insufficient amount image sequences from real observations were available. We resolve this problem with a synthetic training data generator. It is crucial, for the tracklet building system, how we generate the trajectory of moving objects in synthetic data. A trained model prediction will be able to recognize only objects with a trained specific type of trajectory.

The algorithm can generate synthetic data similar to real observation series taken by a telescope based on the observed distribution function of trajectories. Synthetic training data is a series of 8 images consisting of the starfield and the variable number of moving objects. Their trajectories are straight lines given as lists of positions in each image, see Fig. 7. Positions of both stars and objects in each frame are stored into *TSV* file with the same structure as is required for the input in the Masking step.

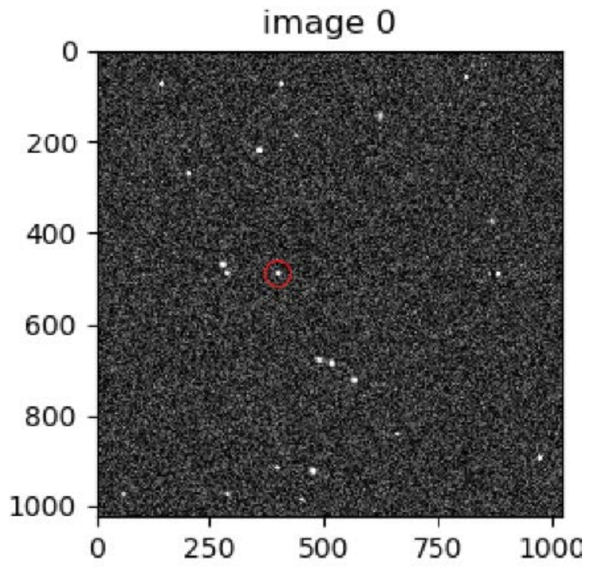

Fig. 7. The first image of the series. The moving object is in the red circle moving downwards.

B. Model selection

The neural network with described topology has two adjustable parameters: hidden dimension and number of LSTM layers. To choose the best parameters, we trained multiple models for 20 epochs i.e. learning iteration with 500 batches each containing 32 sequences with the length of 2. Our strategy is to create a complex model, which is complex enough to solve this problem with an acceptable training time.

Hidden dimension. LSTM projects inputs to the hidden dimension space. To accumulate enough information in the hidden layer, it must have a higher dimension than the input space. Higher dimension than 50 lead to higher training error as shown in Tab. I. For this reason, we choose the highest hidden dimension without significantly increasing the training error after 20 epochs. We set the hidden dimension equal to 50.

TABLE I
Training error versus the dimension of the hidden layer.

Hidden Dimension	Training Error [px]
2	72
5	20
15	6
50	6
100	9

Number of LSTM layers. Using multiple LSTM cells leads to the increase of training time required. Training error is Euclidean mean distance between predicted and next position in pixels. The training error is also converging slower, see Tab. II. For these reasons, we chose the number of layers equal to 1.

TABLE II
CHANGE IN TRAINING ERROR AND EPOCH TIME FOR DIFFERENT NUMBERS OF LSTM LAYERS IN NETWORK.

Number of Layers	Training Error [px]	Epoch Time [s]
1	6	1.9
2	11	2.55
3	13	3.13
5	18	4.5

C. *Length of Training Input Sequence*

The input sequence of length 2 consists of the last two positions of the moving object. In the Fig. 8, model is learning quickly. Training loss after 20 epochs is around 3 pixels. However, we use only a fraction of known information about the object's movement.

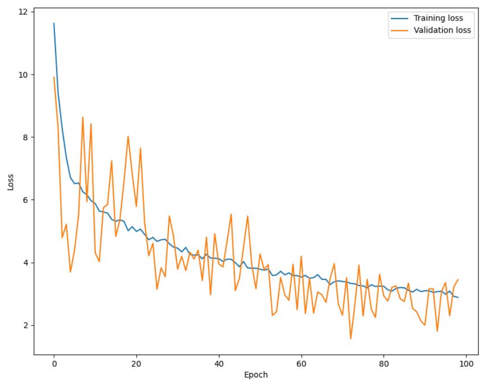

Fig. 8. Training and validation loss during training in the Model prediction step using input sequence of length 2.

We trained the model for 500 epochs to get the best performance. A model trained on an input sequence of length 2 cannot be used to process the longer sequence. As shown in Tab. III, the testing error is increasing with an input sequence length as was expected.

D. *Variable Length of Training Input Sequence*

A model trained on the variable input sequence consists of multiple known positions of the object. We use every available information about the movement using this approach. Thanks to that, the prediction of the new position is less sensitive to local deviations.

TABLE III
TESTING ERROR FOR INPUT SEQUENCES WITH DIFFERENT LENGTHS. EACH LENGTH WAS TESTED ON 100000 EXAMPLES 5 TIMES TO GET AN AVERAGE ERROR.

Test Input Sequence Length	Error [px]
2	0.1393
4	16.6966
6	23.68
8	28.0441
10	31.2441

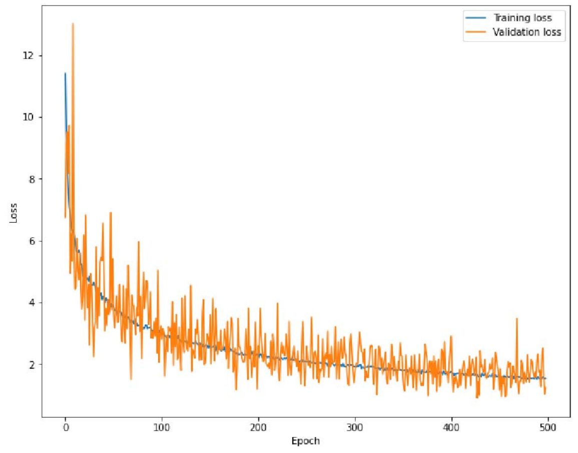

Fig. 9. Training loss (in blue) and validation loss (in orange) during the training process of 500 epochs with variable length input.

The training process is a little slower with the variable input sequence length as we can, see Fig. 9. If we trained a neural network with variable length sequences for 500 epochs, training loss will drop to around 2.5 px. Behavior of training loss and validation loss in time is visualised in Fig. 9. This approach can perform quite well on inputs with length from 2 to 7, Tab. IV.

Fig. 10 shows the model performance on real data. We can observe that the model with variable input sequence length is more consistent for long sequences. Predicted positions 8, 9, 10, 11 have the same direction as real positions.

V. Results

The Department of Astronomy and Astrophysics has provided 26 processed image series from AGO70 telescope. Each series contains positions of one of the 5 real moving objects. They are the GPS satellites Navstar 51, Navstar 60, Glonass satellites Cosmos 2434, Cosmos 2460, and broadcasting satellite AMC-14, for more information seen [11]. These data are the real data, used as testing data of the proposed system, they contain verified ground-truth tracklets.

TABLE IV
TESTING ERROR FOR INPUT SEQUENCES WITH DIFFERENT LENGTH. EACH LENGTH WAS TESTED ON 100000 EXAMPLES 5 TIMES TO GET AVERAGE ERROR.

Test Input Sequence Length	Error [px]
2	2.284
3	0.2325
4	0.181
5	0.15
6	0.1418
7	0.1525
8	0.4566

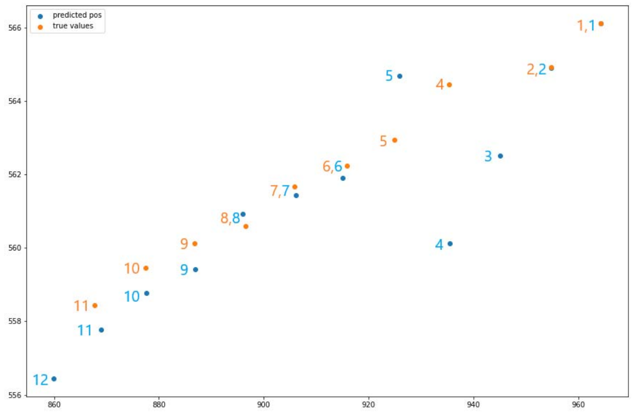

Fig. 10. Positions of the moving object. Object is moving from top right corner to bottom left. Blue points are predicted positions by the model and the red dots are the real positions.

A. Default Parameters and System Behavior

Masking threshold parameter is the maximum distance between positions in masking step. It is relative number to the image size. The default value is 0.002%.

A matching threshold parameter is the maximum distance between the predicted position and position in an image. This value can influence the total amount and accuracy of found tracklets. The bigger the threshold is, the more tracklets will be found, but many with low confidence. The matching threshold is a relative number to image size. The default value is 0.003%.

Parameter K is the maximum number of missing consequent positions of moving object. Default set to $K = 2$. The final tracklet could have missing positions in at most K consecutive images. Many times, objects in an image are not recognized by the segmentation algorithm and their positions are not in the input sequence to our system. Therefore, the value of K allows the system to find tracklets with missing positions. If $K \geq 3$ the system can find random tracklets thanks to many missing positions with low confidence value.

The system was trained to create tracklets from input sequences with maximal length 8. To process longer sequences we can create multiple short subsequences with a slight overlap. The minimal value of the overlap is 2 because we need at least two positions for prediction.

B. Validation

To validate the proposed system we use the existing analytic system as the ground truth method and the testing data as the ground truth data. The ground truth data were also manually evaluated.

Our new system is the system with embedded RNN using parameters: masking threshold = 0.002, matching threshold = 0.003, $K = 2$ and overlap = 2. The RNN system was trained on artificial data.

For every test sequence from real object data set, tracklets found by both systems were the same in other words tracklets are the equal or analytic solution is a part of the RNN solution. In most cases tracklets found by the RNN system are longer, as shown in the table V. Because of this conclude that the proposed system outperforms the analytic system on tested data.. The resulting tracklets of Cosmos 2460 satellite from both the proposed and the analytic method are the same, see the left image of Fig. 11. The tracklet of Navstar 51 satellite has additional 4 positions found by the proposed system while the analytic ground truth method missed them, see the right image in Fig. 11. The blue dots are the positions found by the RNN system, only. The output tracklets based on real observation of AMC-14 and Navstar 60 demonstrate that the proposed RNN system is capable to find the longer tracklets with new additional positions compared to the ground truth method as shown in Fig. 12. Green points are positions corresponding to the given measurement time that is the identical result of both the proposed and the ground truth

TABLE V
DIFFERENCE BETWEEN TRACKLETS LENGTH COMPARING THE PROPOSED RNN SYSTEM AND THE ANALYTIC SYSTEM.

Difference [# points]											
-1	0	1	2	3	4	5	6	7	8	9	10
Count											
1	2	8	4	6	1	0	1	1	0	0	2

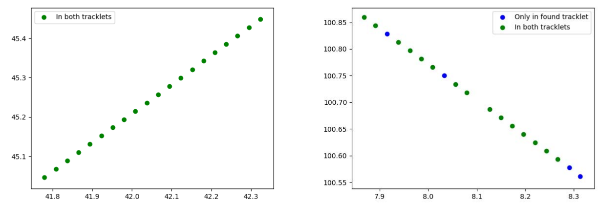

Fig. 11. Satellite tracklets. Left: Cosmos 2460 satellite. The resulting tracklets of Cosmos 2460 satellite from both systems are the same. Right: Resulting tracklets of Navstar 51 satellite. The result from the proposed system is about 4 positions longer than one from the analytic system.

method. The blue dots are additional positions found by the RNN system, only. The error position from the ground truth analytical system is the red dot in right image of Fig. 12. Another real observation data that includes the long time

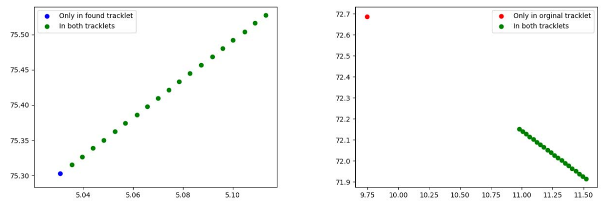

Fig. 12. Output satellite tracklets based on real data. Left: Tracklets for AMC-14 satellite. Tracklet from the proposed system is 1 position longer than tracklet from the analytic system. Right: Tracklets for Navstar 60 satellite. Tracklet from the analytic system is 1 position longer than tracklet from the RNN system. But the additional position (red) is an error made by the analytic system.

observation of AMC-14 satellite were used to test the system, see Fig. 13. The tracklet from the proposed system is 10 positions longer than tracklet from the ground truth analytic system. The vector confidence value for the output tracklet is 0.90. Green points are positions corresponding to the given measurement time that is the identical result of both the proposed and the ground truth method. The blue dots are additional positions found by the RNN system, only.

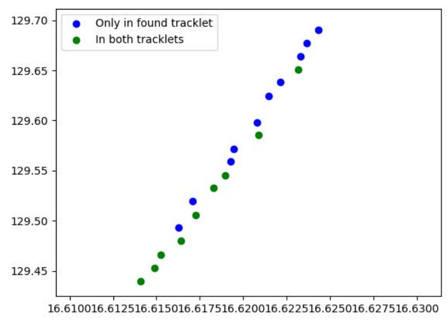

Fig. 13. Tracklets from the long time observation of AMC-14 satellite. The blue dots are additional positions found by the RNN system, only. The green dots are the position found successfully by both the analytic and proposed methods, respectively.

VI. Conclusion

This paper is the proof of concept of the proposed processing pipeline with the newly included position prediction step based on RNN. We have shown that the RNN method is capable of finding moving object positions in the sequence of images. Moreover, this method performs well on testing data. In comparison to the ground truth analytical solution, the RNN method was capable to find longer tracklets, which could lead to more precise identification of the object trajectory. We conclude that the proposed system detects different types of object trajectories using trained networks without a need to change the design of the complex pipeline.

The proposed methodology returns a confidence value for each tracklet. This helps the user to detect false positive examples. Such information is often missing in the existing analytical systems.

In future work, the RNN method can be further improved to find moving objects with more complex trajectories and missing positions in the input image sequence.

Acknowledgment

The presented work was performed under the programme of SK6-26-400013667221 NLSC-ESA PECS 6 activity AMOS Reentry, as well as additional funding from the Comenius University in Bratislava.

References

[1] B. Taylor, G. Aglietti, S. Fellowes, S. Ainley, T. Salmon, I. Retat, C. Burgess, A. Hall, T. Chabot, K. Kanan, A. Pisseloup, C. Bernal, F. Chaumette, A. Pollini, and W. H. Steyn, “Remove Debris Mission, From Concept to Orbit,” in *SmallSat 2018 - 32nd Annual AIAA/USU Conference on Small Satellites*, Logan, United States, Aug. 2018, pp. 1–10. [Online]. Available: https://hal.inria.fr/hal-01877662

[2] J. Silha, S. Krajčovič, M. Zigo, D. Zilkova, P. Zigo, J. Simon, J. Toth, L. Kornos, S. Setty, T. Flohrer *et al.*, “Development of the slovak 70-cm optical passive system dedicated to space debris tracking on leo to geo orbits,” *Advanced Maui Optical and Space Surveillance Technologies Conference (AMOS)*, p. 85, 2019.

[3] J. Silha, S. Krajcovic, J. Tóth, L. Kornos, J. Világi, P. Zigo, J. Simon, D. Kalmancok, M. Hamara, D. Zilková, L. Novak, M. Zigo, F. Duris, V. Nagy, R. Durikovic, T. Schildknecht, E. Cordelli, A. Vananti, C. D. Paccolat, A. Rachman, H. K. Mann, M. Trujillo, and T. Flohrer, “Slovakian optical sensor for hamr objects cataloguing and research,” in *69th International Astronautical Congress (IAC2018)*, 2018.

[4] S. Krajčovič, R. Ďurikovič, and J. Šilha, “Selected modules from the slovak image processing pipeline for space debris and near earth objects observations and research,” in *2019 23rd International Conference Information Visualisation (IV)*. IEEE-CS, 2019, pp. 112–117.

[5] W. Thompson, “Coordinate systems for solar image data,” *Astronomy & Astrophysics*, vol. 449, no. 2, pp. 791–803, 2006.

[6] K. Stanislav, R. Ďurikovič, and J. Šilha, *Masking and Tracklet Building for Space Debris and NEO Observations: The Slovak Image Processing Pipeline*. IGI Global, 2020, pp. 38–56.

[7] N. Cristianini and J. Shawe-Taylor, *Support Vector Machines*. Cambridge University Press, 2000, p. 93–124.

[8] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[9] I. Hasan, F. Setti, T. Tsesmelis, A. D. Bue, F. Galasso, and M. Cristani, “Mx-lstm: Mixing tracklets and vislets to jointly forecast trajectories and head poses,” *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, p. 10, 2018.

[10] T. V. Eli Stevens, Luca Antiga, *Deep Learning with PyTorch*. Manning Publications Co., 2020.

[11] N2yo - live real time satellite tracking and predictions. https://www.n2yo.com/, visited 27.1.2021.