A Robust Vision-based Algorithm for Detecting and Classifying Small Orbital Debris Using On-board Optical Cameras

Yasin Zamani
University of Utah, Salt Lake City, UT
Joel Amert
NASA/MSFC, Huntsville, AL

Thomas Bryan
NASA/MSFC, Huntsville, AL

Neda Nategh
University of Utah, Salt Lake City, UT

ABSTRACT

This study develops a vision-based detection and classification algorithm to address the challenges of in-situ small orbital debris environment classification, including debris observability and instrument requirements for small debris observation. The algorithm operates in near real-time and is robust under challenging tasks in moving objects classification such as multiple moving objects, objects with various movement trajectories and speeds, very small or faint objects, and substantial background motion. The performance of the algorithm is optimized and validated using space image data available through simulated environments generated using NASA Marshall Space Flight Centers Dynamic Star Field Simulator of on-board optical sensors and cameras.

1. INTRODUCTION

After more than 50 years of human space activities, orbital debris has become a serious problem in the near-Earth environment [1]. The U.S. Space Surveillance Network is currently tracking more than 22,000 objects larger than about 10 centimeters (cm). Additional optical and radar data indicate that there are about 500,000 pieces of debris larger than 1 cm, and more than 100 million pieces of debris larger than 1 millimeter (mm) in orbit. NASAs statistical analysis predicts there are between 300,000 to 600,000 debris particles orbiting earth that are smaller than a softball. These small, untracked debris objects ($\sim$500,000) outnumber the larger and tracked objects ($\sim$20,000) by a factor 25 to 1. They are too small to track but too large to shield against. Therefore, the risk of the small untracked debris objects to operational spacecraft is much higher than the risk posed by the larger and tracked debris objects. Low Earth Orbit (LEO) has the highest concentration of both active satellites and debris, thus increasing the probability of objects colliding.

To address this challenge, this study, defined in conjunction with scientists at NASA Marshall Space Flight Center (MSFC), focuses on the problem of detection and tracking for small orbital debris ranging in size from 5 mm to 10 cm using a new low-cost space-based orbital debris tracking system. The algorithm technology development builds on the Small Orbital Debris Detection, Acquisition, and Tracking (SODDAT) conceptual technology demonstration concept developed by MSFC.

Here, we specifically focus on one of the major components in this space-based tracking system, which is a vision-based detection and classification algorithm. The algorithm takes the pixel location and brightness information of everything in the field of view (FOV) of an optical camera captured in a video sequence. This pixel information is used to compute the centroid location of each spot in each camera frame and their associated brightness values. The centroid data are then used by a statistical classifier for differentiating between stars and debris objects in the set of detected spots. The classification algorithm takes successive camera frames as its input and determines which detected

Block Diagram Notations:

- $I_i \rightarrow i^{th}$ intensity/gray-scale frame (in the block diagram, I_i refers to the current frame, and I_{i-d} refers to the d^{th} frame before the current frame).
- $\tau \rightarrow$ Threshold value selected to make a black and white (binary) image from the intensity image.
- $B_i \rightarrow i^{th}$ black and white (binary) frame.
- $C_i \rightarrow$ Spot centroids for the i^{th} binary frame.
- $D \rightarrow$ Distance matrix. $D_{p,q} \rightarrow$ Euclidean distance between the p^{th} spot centroid in the previous frame with q^{th} spot centroid of current frame.
- $c \rightarrow$ Cost value selected for labeling a spot as *Unknown* (no matching found).
- $M \rightarrow$ Matches.
  $M = \{(p,q) | p^{th}$ spot centroid in the previous frame is matched with the q^{th} spot centroid in the current frame$\}$.
- $T_i \rightarrow$ Translation vectors calculated for the i^{th} frame. For each matched pair, there is a translation vector defined from the centroid in the previous frame to the matched centroid in the current frame.
- $O \rightarrow$ Outliers. returns a logical array whose elements are true when an outlier is detected in the corresponding element of T.
- $\alpha \rightarrow$ Significance level. It is the probability of the study rejecting the null hypothesis, given that the null hypothesis were true.
- $K \rightarrow$ Maximum outlier count. It specifies the maximum number (upper bound) of outliers returned by the method.
- $L_i \rightarrow$ Label of the spots in the i^{th} frame. Each label could be *Object*, *Star*, or *Unknown*.

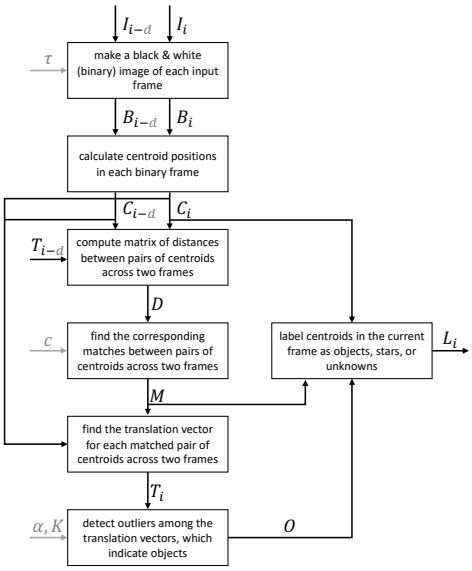

Fig. 1: Block diagram of classification algorithm.

values are debris objects. The algorithm should perform accurately and robustly in challenging environments such as moving star background, multiple moving objects, objects with various moving trajectories and speeds, occlusion of spots. The algorithm should also be computationally tractable for on-orbit calculations. The accuracy of the algorithm is evaluated across a variety of image conditions and movement patterns and dynamics.

Moreover, we optimize and validate the performance of the algorithm using testing capabilities and space debris images provided by the MSFC. The MSFC's one-of-a-kind Dynamic Star Field Simulator (DSFS) uses a high resolution large monochrome display and a custom collimator capable of projecting realistic star images with simple orbital debris spots (down to star magnitude 11-12) into a passive orbital detection and tracking system with simulated real-time angular motions of the vehicle-mounted sensor. The DSFS can be expanded for multiple sensors (including advanced star trackers), real-time vehicle pointing inputs, and more complex orbital debris images. Images from the DSFS serve as inputs to the detection algorithm to track simulated small orbital debris objects.

This vision-based detection and classification algorithm will be used to detect very small and faint space debris, very quickly and robustly, using videos captured by an optical camera from an orbit. The ultimate classification system can be used on a constellation of small satellites in LEO, which will enable detecting debris outside of the presence of atmospheric scintillation to achieve higher precision than existing systems [2, 3].

2. ALGORITHMS

The purpose of the proposed algorithm is to identify the bright spots in captured video sequences and to classify them as "objects" (representing orbital debris objects) or "stars". In cases the algorithm is not able to classify any of the detected spots into one of the object or star classes, that spot will be labeled as "unknowns", which is neither the objects class nor the stars class. The input to the algorithm is a sequence of intensity images over time, which represents the captured space videos. Algorithm 1 and Fig. 1 describe the classification algorithm developed in this work.

Algorithm 1 Classification

function CLASSIFY(I)
Input: Video I ▷ Sequence of intensity images
Output: Labels L, centroids C ▷ Each label could be *'Object'*, *'Star'*, or *'Unknown'*
  constant τ ▷ Threshold of binarization
  constant c ▷ Cost of *Unknown* label (not matching)
  constant d ▷ Delay between two consecutive frames
  constant α ▷ Significance level for detecting outliers
  constant K ▷ Maximum number of outliers

  for $i = 1$ **to** d **do**
    $T_i \leftarrow \{[0,0]\}$ ▷ Translation vectors of the first d frames
    $L_i \leftarrow \emptyset$ ▷ Labels of the first d frames
    $C_i \leftarrow$CENTROIDS(I_i, τ) ▷ Centroids of the first d frames
  end for

  for $i = d+1$ **to** $\|I\|$ **do**
    $C_i \leftarrow$ CENTROIDS(I_i, τ)
    $D \leftarrow$ DISTANCEMATRIX$(C_{i-d} + T_{i-d}, C_i)$
    $M \leftarrow$ MATCHES(D, c)
    $T_i \leftarrow$ TRANSLATIONVECTORS(C_{i-d}, C_i, M)
    $\Theta \leftarrow \{\arctan(u) \,|\, u \in T_i\}$ ▷ Vector directions
    $R \leftarrow \{\|u\|_2 \,|\, u \in T_i\}$ ▷ Vector magnitudes
    $O \leftarrow$ OUTLIERS$(\Theta, \alpha, K) \bigcup$ OUTLIERS(R, α, K)
    $L_i \leftarrow$ LABLE(C_i, M, O)
  end for
  return L, C
end function

To classify spots in each image frame, the first step is to determine the position of the spots in each frame as follows. Using grayscale images as the input, each pixel value between 0 and 1 in the image represents the light intensity at that pixel, with larger values representing brighter pixels. A global intensity threshold value is chosen such that it can remove as many noisy pixels as possible in all the input frames. Using this threshold value, input grayscale images are converted into binary black and white images [4].

To determine the centroid of each spot using the detected white pixels, we find clusters of white pixels that are one of the 8-connectivity neighborhoods of each other where each cluster represents a connected component marking an area in the image that each spot covers. Each spot is then identified by the center of its corresponding connected components, called centroids [5]. The centroids information is calculated for each frame (Algorithm 2). The classification algorithm uses the centroids information at a given frame (referred to as current frame in the text) as well as that of in the d-th frame before (referred to as previous frame), where d represents the history of the centroids information that the algorithm uses for its computations.

Algorithm 2 Spot Centroids

function CENTROIDS(I, τ)
Input: Intensity image I, binarization threshold τ
Output: Centroids C ▷ Each centroid is an (x, y) pixel position
  $B \leftarrow$ Binarize intensity image I by using threshold τ
  for all connected component $W \in B$ **do**
    append (x, y) coordinate of the centroid for the connected component W to C
  end for
  return C
end function

Next, the distance of each center location of spots in the current frame to those in the previous frame is calculated (Algorithm 3). The result will be a distance matrix whose entries evaluate the Euclidean distance between one centroid in the current frame to another one in the previous frame [6]. The distance values represent the cost associated with matching two centroids between the current and previous frame.

Algorithm 3 Distance Matrix

function DISTANCEMATRIX(C_1, C_2)
Input: Centroids of previous and current frames C_1, C_2
Output: Distance matrix D
    $n_1 \leftarrow \|C_1\|$
    $n_2 \leftarrow \|C_2\|$
    for $i = 1$ **to** n_1 **do**
        for $j = 1$ **to** n_2 **do**
            $D_{i,j} \leftarrow \left\|C_{1,i} - C_{2,j}\right\|_2$
        end for
    end for
    return D
end function

Using the distance matrix, we obtain the correspondence between centroids in the current frame versus those in the previous frame as described in Algorithm 4. It should be noted that these correspondences must be one-to-one, i.e., each centroid in the current frame corresponds to one and only one centroid at maximum in the previous frame; equivalently, each centroid in the previous frame should also have a maximum of one matched centroid in the current frame [7, 8]. Those centroids in the current frame for which a match cannot be found, for example in the case that a spot has just entered the image frame and was not present in the previous frame, are labeled as unknowns in that frame.

Algorithm 4 Match Pairs
Generalized Linear Assignment Problem

function MATCHES(D, c)
Input: Distance matrix D, unmatched cost c
Output: Match pairs M

$$\text{minimize} \sum_{i=1}^{m}\sum_{j=1}^{n} D_{ij}x_{ij} + c\cdot(m+n-2\|M\|)\,.$$

$$\text{subject to} \sum_{j=1}^{n} x_{ij} \le 1 \qquad i = 1,\ldots,m;$$

$$\sum_{i=1}^{m} x_{ij} \le 1 \qquad j = 1,\ldots,n;$$

$$x_{ij} \in \{0,1\} \qquad i = 1,\ldots,m, \quad j = 1,\ldots,n;$$

$$M = \left\{(i,j)\,|\,x_{i,j} = 1\right\} \qquad i = 1,\ldots,m, \quad j = 1,\ldots,n.$$

    return M
end function

Using the matched pairs, we calculate a translation vector for each centroid in the current frame relative to its corresponding matched centroid in the previous frame (Algorithm 5). Each translation vector represents a shift in the location of a centroid from the previous frame to the current frame [9].

Algorithm 5 Translation Vectors

function TRANSLATIONVECTORS(C_1, C_2, M)
Input: Centroids of previous and next frames C_1, C_2, and Match pairs M
Output: Translations T
  for all $(i, j) \in M$ **do**
    append $C_{2,j} - C_{1,i}$ to T
  end for
  return T
end function

Using a hypothesis testing approach [10–12], spot centroids with statistically different translation vectors in magnitude or direction are classified into the objects class, and the remainder of the spots are classified as stars. The statistical test and labeling procedure are described in Algorithm 6 and 7, respectively.

Algorithm 6 Outliers
Generalized (extreme Studentized deviate) ESD test

function OUTLIERS(X, α, K)
Input: Sample data X, significance level α, maximum number of outliers K
Output: Index of outliers O
  $XX \leftarrow X$ ▷ Save a copy of X
  for $i = 1$ **to** K **do**
    $\bar{X} \leftarrow \text{mean}(X)$
    $s \leftarrow \text{std}(X)$
    $x^* \leftarrow \underset{x \in X}{\text{argmax}} |x - \bar{X}|$
    $N \leftarrow \|X\|$
    $\nu \leftarrow N - i - 1$
    $p \leftarrow 1 - \frac{\alpha}{2(\nu+2)}$
    $t_{p,\nu}$ denoting the critical value of the t distribution with ν degrees of freedom and a significance level of p
    $\tau \leftarrow \frac{(\nu+1)t_{p,\nu}}{\sqrt{(\nu+2)(\nu+t_{p,\nu}^2)}}$
    if $\frac{|x^* - \bar{X}|}{s} > \tau$ **then**
      $l \leftarrow \bar{X} - s \cdot \tau$ ▷ Lower bound
      $u \leftarrow \bar{X} + s \cdot \tau$ ▷ Upper bound
    end if
    Remove x^* from X
  end for
  $X \leftarrow XX$ ▷ Restore X
  for $i = 1$ **to** $\|X\|$ **do**
    if $X_i < l$ **or** $X_i > u$ **then**
      append i to O
    end if
  end for
  return O
end function

Table 1: Input sample data specifications

Sample data	Frame rate (fps)	Duration (sec)	Frame count	Spots (mean $\pm$ std)	Objects (mean $\pm$ std)
Simple foreground	30	6	180	134.7 ± 2.4	1.3 ± 0.7
Complex foreground	30	30	900	117.7 ± 3.7	6.8 ± 2.0
Simple background	60	60	3600	118.3 ± 2.1	5.3 ± 2.1
Complex background	60	60	3600	114.0 ± 4.8	5.1 ± 2.3

Algorithm 7 Labeling

function LABEL(C_i, M, O)
Input: Centroids of current frame C_i, match pairs M, and index of outliers O
Output: Labels L
  for $j = 1$ **to** $\|C\|$ **do**
    if $j \in O$ **then**
      $L_j \leftarrow$ *'Object'*
    else if $j \in \{j \,|\, (i, j) \in M\}$ **then**
      $L_j \leftarrow$ *'Star'*
    else
      $L_j \leftarrow$ *'Unknown'*
    end if
  end for
  return L
end function

The software implementing these algorithms is publicly available at [13].

3. RESULTS

Information about the sample video files is summarized in Table 1. For all the sample videos *Frame size* is 1080 (pixel) Height $\times$ 1920 (pixel) Width, *Color format* is Grayscale, and the *Source data type* is Floating-point number between 0 and 1. Sample snapshots of one of the video datasets (Complex foreground data in Table 1) has been shown in Fig. 2.

Four different datasets were used to evaluate the performance of the algorithm, each of which is described in Table 1. Each of these datasets has different complexities in their foreground (number of objects and their motion pattern) and foreground (stars motion pattern). In the dataset termed *simple foreground*, there are only two objects, but in the *complex foreground* dataset, the number of objects is increased to ten objects. In the dataset called *simple background* stars background does not move (static background) but in the *complex background* dataset stars are moving in the background. The performance of the algorithm on each of the datasets is shown in figures 3 to 6, respectively.

The sensitivity of the performance of the classification algorithm to two parameters, unmatched cost (c) and delay between two consecutive frames (d) is shown in figures 3-6. In each figure, the top left plot, for a given c (x-axis) and d (y-axis), shows the ratio of the number of spots correctly classified as an object to the total number of spots in the current frame (true positive rate (TPR) per each frame). This plot shows that as c increases, the algorithm will make more effort to find the corresponding spots and classify them as objects or stars while reducing the number of spots classified as unknowns. On the other hand, as the delay between two consecutive frames increases, the greater is the likelihood that there will be found no correspondance in the previous frame, which will result in the spots being classified as unknowns. These plots show that for parameters c and d, there are range of values for which the algorithm has the maximum possible performance on the sample input data, indicating the feasibility of the algorithm. The top right plot shows the standard deviation of the TPR. The low standard deviation values indicate that the algorithm operates consistently on different frames (the error does not propagate over time). The middle row shows the mean (left panel) and standard deviation (right panel) of the ratio of the number of spots correctly classified as stars to

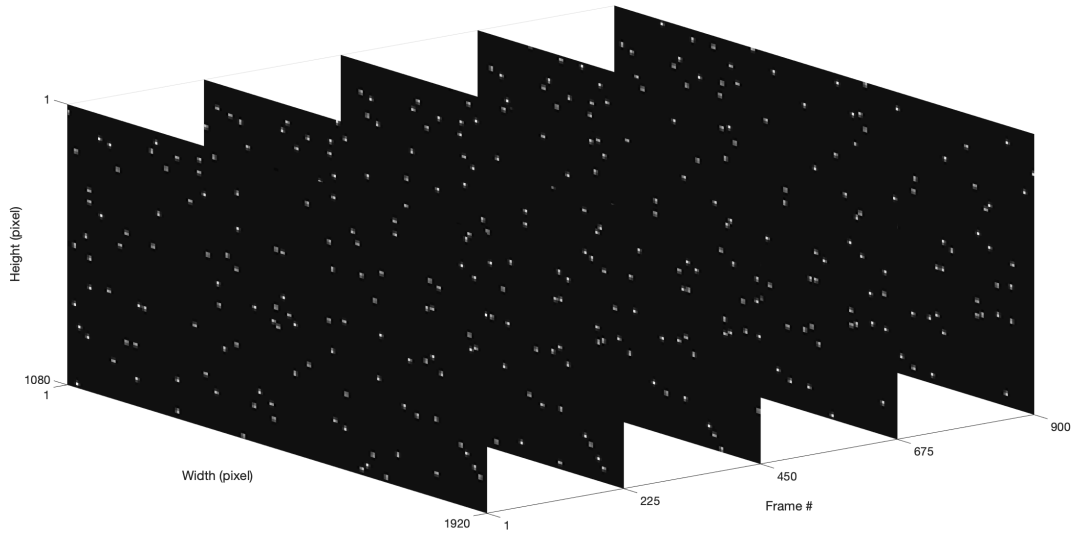

Fig. 2: Sample input data. Shown is a sequence of grayscale intensity image frames over time, represeting a few snapshots of the captured video sequence.

the total number of spots in the current frame (true negative rate (TNR) per each frame). Similar to the above, the higher the c is and the lower the d is, the algorithm will have a better performance on this measure too. In general, since in each frame there are usually much larger number of stars than the number of objects, this measure shows more robust to the parameter c for stars than that of for objects above. The bottom row shows the mean (left) and standard deviation (right) of the average ratio of the number of spots classified as unknowns in the current frame when the algorithm cannot find a corresponding match for these spots between the current and previous frames to the total number of spots in that frame. For this analysis the values of significance level parameter (α) and the maximum possible number of outliers (K) used for detecting the outliers are set respectively to 0.05 and 10% of the total number of spots in the current frame are selected.

The source code for generating these results is available at [13].

4. DISCUSSION

In this paper, a statistical classifier algorithm is proposed to detect and classify small bright spots in the input image sequence. When applied to the simulated videos of small orbital debris objects moving over a static or dynamic star background provided by the NASA MSFC, our algorithm showed successful in robustly detecting and classifying the objects versus stars. The proposed algorithm has several degrees of freedom which can affect the classification performance. Two of those degrees of freedom (DoF), cost of unknown labels (c) and delay between two consecutive frames (d) were analyzed in this study. For other DoF, values were chosen such that the algorithm showed the best performance on the data samples under stufy. However, these DoF can be tuned and optimized based on the statistics of the input images to be able to generalize to variety of image settings or complexities. For example, for the threshold of binarization parameter (τ), a constant global value was selected here, but othe dynamic thresholding methods could be used to adapt this value for any particular input image statistics [14]. In addition, the significance level parameter (α) and the maximum possible number of outliers (K) parameter used for detecting outliers are also among the DoF of the algorithm. Future studies can focus on optimizing possible DoF values based on certain applications and the desired performance measures.

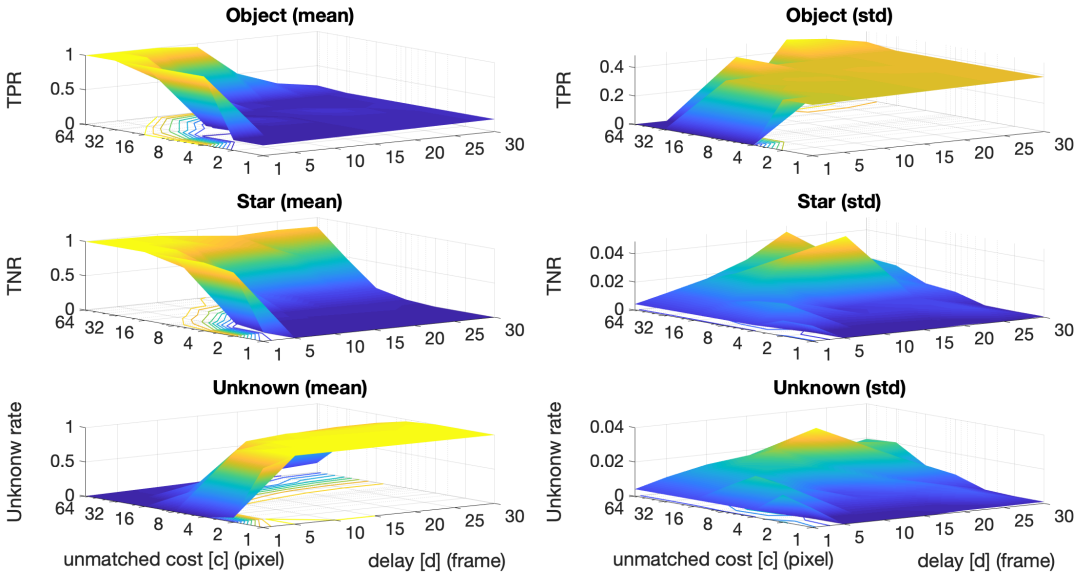

Fig. 3: Simple foreground (Two objects with similar moving speed and trajectory).

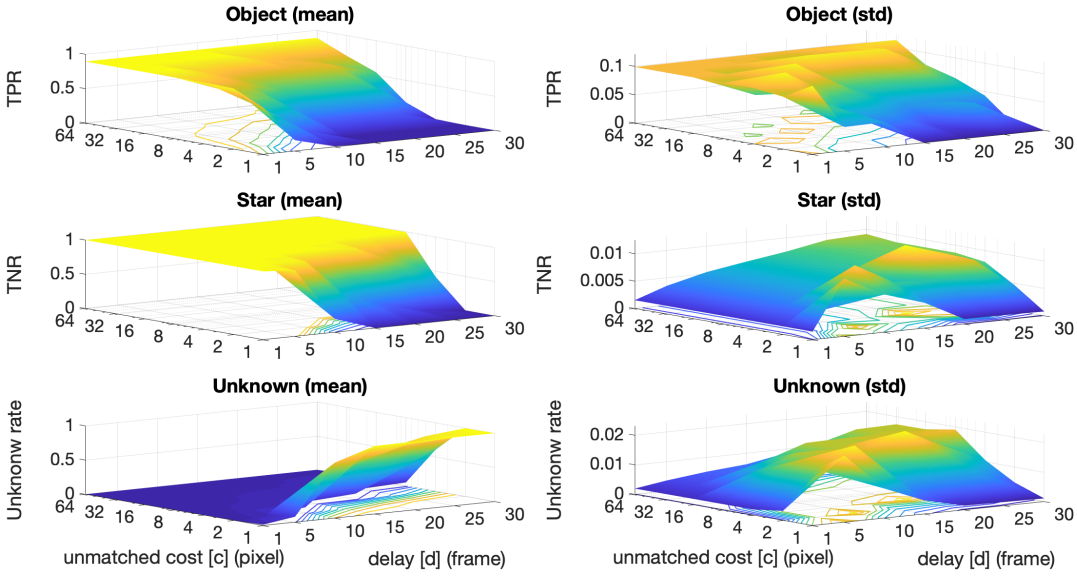

Fig. 4: Complex foreground (Ten objects moving in various directions and at various speeds).

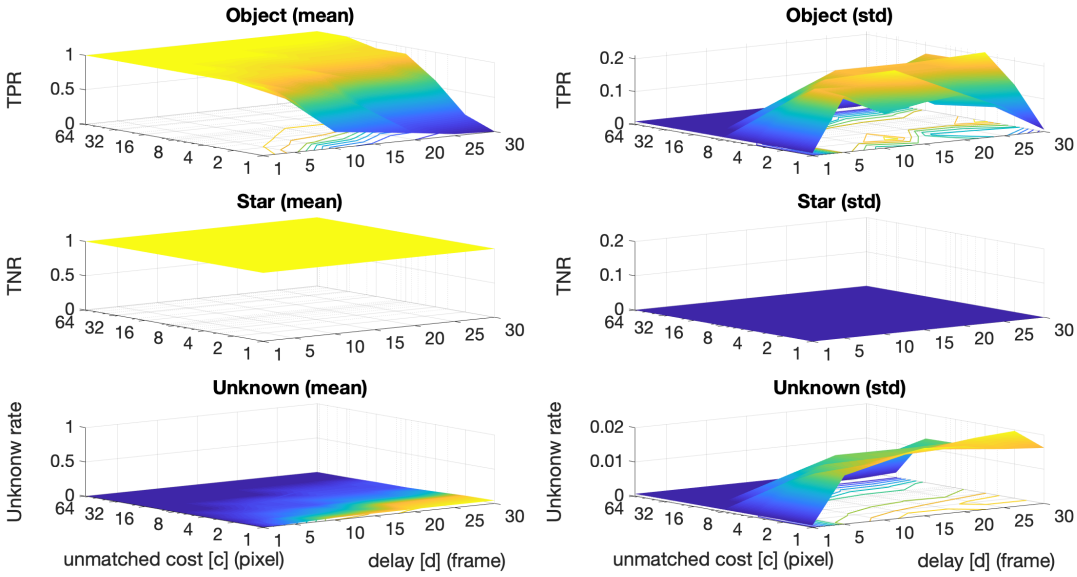

Fig. 5: Simple background (static stars).

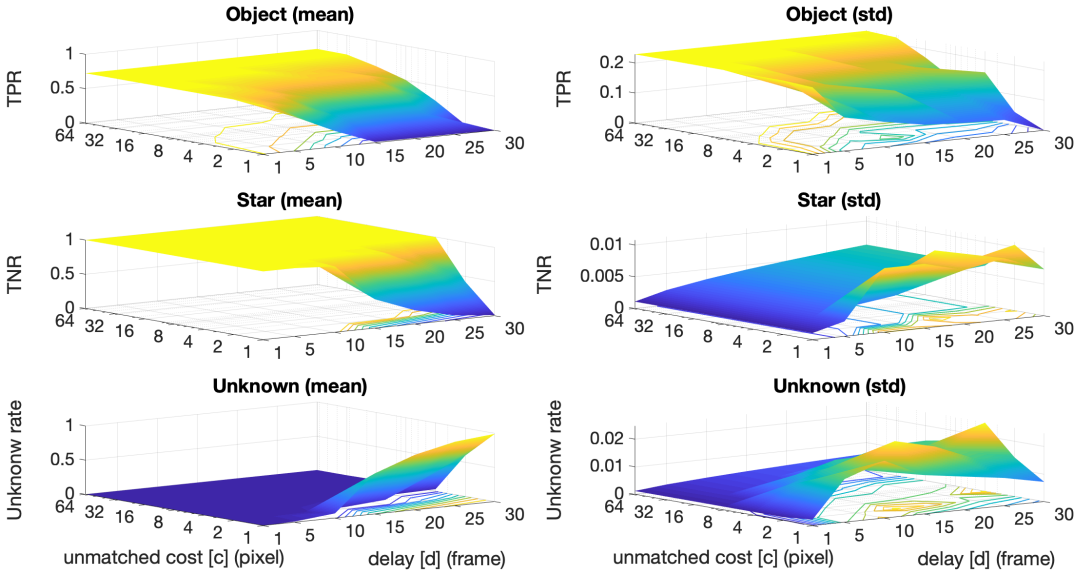

Fig. 6: Complex background (moving stars).

ACKNOWLEDGMENT

The work was supported by NASA 80MSFC18M0010 grant to N.N.

REFERENCES

[1] A. Nafi, A. Bernard, and K. Fujimoto, "A unified approach for optical survey strategy design of resident space objects," in *AIAA/AAS Astrodynamics Specialist Conference*, p. 5500, 2016.

[2] A. C. Snow, J. L. Worthy III, A. den Boer, L. J. Alexander, M. J. Holzinger, and D. Spencer, "Optimization of cubesat constellations for uncued electrooptical space object detection and tracking," *Journal of Spacecraft and Rockets*, pp. 401–419, 2016.

[3] A. D. Biria and B. G. Marchand, "Constellation design for space-based space situational awareness applications: an analytical approach," *Journal of Spacecraft and Rockets*, vol. 51, no. 2, pp. 545–562, 2014.

[4] C. Solomon and T. Breckon, *Fundamentals of Digital Image Processing: A practical approach with examples in Matlab*. John Wiley & Sons, 2011.

[5] A. Rosenfeld, J. L. Pfaltz, *et al.*, "Sequential operations in digital picture processing.," *J. ACM*, vol. 13, no. 4, pp. 471–494, 1966.

[6] J. E. Gentle, "Matrix algebra," *Springer texts in statistics, Springer, New York, NY, doi*, vol. 10, pp. 978–0, 2007.

[7] J. Matousek and B. Gärtner, *Understanding and using linear programming*. Springer Science & Business Media, 2007.

[8] R. Jonker and A. Volgenant, "A shortest augmenting path algorithm for dense and sparse linear assignment problems," *Computing*, vol. 38, no. 4, pp. 325–340, 1987.

[9] E. T. Whittaker, *A treatise on the analytical dynamics of particles and rigid bodies*. Cambridge University Press, 1988.

[10] G. L. Tietjen and R. H. Moore, "Some grubbs-type statistics for the detection of several outliers," *Technometrics*, vol. 14, no. 3, pp. 583–597, 1972.

[11] W. Stefansky, "Rejecting outliers in factorial designs," *Technometrics*, vol. 14, no. 2, pp. 469–479, 1972.

[12] B. Rosner, "Percentage points for a generalized esd many-outlier procedure," *Technometrics*, vol. 25, no. 2, pp. 165–172, 1983.

[13] N. Nategh and Y. Zamani, "Amos2019." `https://github.com/nnategh/AMOS2019`, 2019.

[14] D. Bradley and G. Roth, "Adaptive thresholding using the integral image," *Journal of graphics tools*, vol. 12, no. 2, pp. 13–21, 2007.

[15] E. Alpaydin, *Introduction to machine learning*. MIT press, 2009.