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 difficult 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.

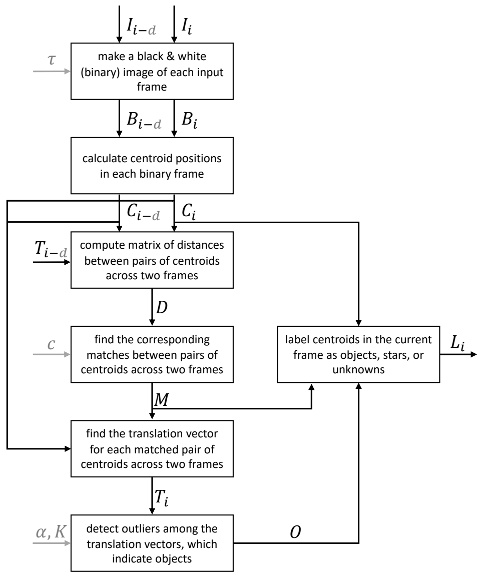
1. INTRODUCTION

After more than 50 years of human space activities, orbital debris has become a serious problem in the near-Earth environment [3, 7]. 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 greater 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 the 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 to train a 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 values are debris objects. The algorithm should perform accurately and robustly in challenging environments such as moving

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-1} refers to one frame in the past).
- $\tau \rightarrow$ 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 ith 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* (not 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$\}$.
- $T \rightarrow$ Translation vectors. 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 ith frame. Each label could be *Object*, *Star*, or *Unknown*.



I_{i-d}
I_i
τ
make a black & white (binary) image of each input frame
B_{i-d}
B_i
calculate centroid positions in each binary frame
C_{i-d}
C_i
T_{i-d}
compute matrix of distances between pairs of centroids across two frames
D
c
find the corresponding matches between pairs of centroids across two frames
label centroids in the current frame as objects, stars, or unknowns
L_i
M
find the translation vector for each matched pair of centroids across two frames
T_i
α, K
detect outliers among the translation vectors, which indicate objects
O

Fig. 1: Block diagram of classification algorithm.

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, signal to noise ratio (SNR), movement patterns, etc.

Moreover, we optimize and validate the performance of the algorithm using testing capabilities and space debris images provided by the MSFC. The MSFCs 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, 11].

2. METHODS

2.1 Statistical Framework

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 case the algorithm is not be 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 input intensity images over time, which represents the captured space videos. Algorithm 1 and Fig. 1 illustrate the classification algorithm developed in this work.

Algorithm 1 Classification

```
 1: function CLASSIFY($I$)
    Input: Video $I$                                   ▷ Sequence of intensity images
    Output: Labels $L$, centroids $C$                  ▷ Each label could be 'Object', 'Star', or 'Unknown'
 2:     constant $\tau$                                ▷ Threshold of binarization
 3:     constant $c$                                   ▷ Cost of Unknown label (not matching)
 4:     constant $d$                                   ▷ Delay between two consecutive frames
 5:     constant $\alpha$                              ▷ Significance level for detecting outliers
 6:     constant $K$                                   ▷ Maximum number of outliers

 7:     for $i = 1$ to $d$ do
 8:         $T_i \leftarrow \{[0,0]\}$                 ▷ Translation vectors of the first $d$ frames
 9:         $L_i \leftarrow \emptyset$                 ▷ Labels of the first $d$ frames
10:         $C_i \leftarrow$CENTROIDS($I_i, \tau$)     ▷ Centroids of the first $d$ frames
11:     end for

12:     for $i = d + 1$ to $\|I\|$ do
13:         $C_i \leftarrow$ CENTROIDS($I_i, \tau$)
14:         $D \leftarrow$ DISTANCEMATRIX($C_{i-d} + T_{i-d}, C_i$)
15:         $M \leftarrow$ MATCHES($D, c$)
16:         $T_i \leftarrow$ TRANSLATIONVECTORS($C_{i-d}, C_i, M$)
17:         $\Theta \leftarrow \{\arctan(u) \mid u \in T_i\}$      ▷ Vector directions
18:         $R \leftarrow \{\|u\|_2 \mid u \in T_i\}$              ▷ Vector magnitudes
19:         $O \leftarrow$ OUTLIERS($\Theta, \alpha, K$) $\bigcup$ OUTLIERS($R, \alpha, K$)
20:         $L_i \leftarrow$ LABLE($C_i, M, O$)
21:     end for
22:     return $L, C$
23: end function
```

In order 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 much noisy pixels as possible in all the input frames. Using this threshold value, input grayscale images are converted into binary black and white images [12].

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 [9]. 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 as previous frame), where d represents the history of the centroids information that the algorithm uses for its computations.

Algorithm 2 Spot Centroids

```
1: function CENTROIDS($I, \tau$)
   Input: Intensity image $I$, binarization threshold $\tau$
   Output: Centroids $C$                         ▷ Each centroid is an $(x, y)$ pixel position
2:     $B \leftarrow$ Binarize intensity image $I$ by using threshold $\tau$
3:     for all connected component $W \in B$ do
4:         append $(x, y)$ coordinate of the centroid for the connected component $W$ to $C$
5:     end for
6:     return $C$
7: 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 [4]. The distance values represent the cost associated with matching two centroids between the current and previous frame).

Algorithm 3 Distance Matrix

```
 1: function DISTANCEMATRIX($C_1, C_2$)
    Input: Centroids of previous and current frames $C_1$, $C_2$
    Output: Distance matrix $D$
 2:     $n_1 \leftarrow \|C_1\|$
 3:     $n_2 \leftarrow \|C_2\|$
 4:     for $i = 1$ to $n_1$ do
 5:         for $j = 1$ to $n_2$ do
 6:             $D_{i,j} \leftarrow \left\|C_{1,i} - C_{2,j}\right\|_2$
 7:         end for
 8:     end for
 9:     return $D$
10: 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 [5, 6]. 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 $M$
```

$$\begin{aligned} &\text{minimize } \sum_{i=1}^{m}\sum_{j=1}^{n} D_{ij}x_{ij} + c\cdot(m+n-2\|M\|)\,. \\ &\text{subject to } \sum_{j=1}^{n} x_{ij} \le 1 && i = 1,\dots,m; \\ &\sum_{i=1}^{m} x_{ij} \le 1 && j = 1,\dots,n; \\ &x_{ij} \in \{0,1\} && i = 1,\dots,m, \quad j = 1,\dots,n; \\ &M = \left\{(i,j)\,|\,x_{i,j} = 1\right\} && i = 1,\dots,m, \quad j = 1,\dots,n. \end{aligned}$$

```
    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 [15].

Algorithm 5 Translation Vectors

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

Using a hypothesis testing approach [10,13,14], 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.

Algorithm 6 Outliers
Generalized (extreme Studentized deviate) ESD test

```
1: function OUTLIERS($X, \alpha, K$)
   Input: Sample data $X$, significance level $\alpha$, maximum number of outliers $K$
   Output: Index of outliers $O$
2:     $XX \leftarrow X$                                            ▷ Save a copy of $X$
3:     for $i = 1$ to $K$ do
4:         $\bar{X} \leftarrow \text{mean}(X)$
5:         $s \leftarrow \text{std}(X)$
6:         $x^* \leftarrow \underset{x \in X}{\text{argmax}} |x - \bar{X}|$
7:         $N \leftarrow \|X\|$
8:         $\nu \leftarrow N - i - 1$
9:         $p \leftarrow 1 - \frac{\alpha}{2(\nu+2)}$
10:        $t_{p,\nu}$ denoting the critical value of the $t$ distribution with $\nu$ degrees of freedom and a significance level of $p$
11:        $\tau \leftarrow \frac{(\nu+1)t_{p,\nu}}{\sqrt{(\nu+2)(\nu+t_{p,\nu}^2)}}$
12:        if $\frac{|x^* - \bar{X}|}{s} > \tau$ then
13:            $l \leftarrow \bar{X} - s \cdot \tau$                          ▷ Lower bound
14:            $u \leftarrow \bar{X} + s \cdot \tau$                          ▷ Upper bound
15:        end if
16:        Remove $x^*$ from $X$
17:    end for
18:    $X \leftarrow XX$                                            ▷ Restore $X$
19:    for $i = 1$ to $\|X\|$ do
20:        if $X_i < l$ or $X_i > u$ then
21:            append $i$ to $O$
22:        end if
23:    end for
24:    return $O$
25: end function
```

Algorithm 7 Labeling

```
1: function LABEL($C_i, M, O$)
   Input: Centroids of current frame $C_i$, match pairs $M$, and index of outliers $O$
   Output: Labels $L$
2:     for $j = 1$ to $\|C\|$ do
3:         if $j \in O$ then
4:             $L_j \leftarrow$ 'Object'
5:         else if $j \in \{j \mid (i, j) \in M\}$ then
6:             $L_j \leftarrow$ 'Star'
7:         else
8:             $L_j \leftarrow$ 'Unknown'
9:         end if
10:    end for
11:    return $L$
12: end function
```

The software implementing these algorithms is publicly available at [8].

3. RESULTS

Information about the sample video files is summarized in Table 1. For all the sample videos *Frame size* is 1080 (pixel) Height × 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 has been shown in Fig. 2.

Table 1: Input sample data specifications

Sample data	**Frame rate (fps)**	**Duration (sec)**	**Frame count**	**Spots (mean ± std)**	**Objects (mean ± 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

Four different data sets were used to evaluate the performance of the algorithm, each of which is described in Table [**?**]. Besides, each of these datasets has different complexities in their foreground (number of objects and their motion pattern) and foreground (stars motion pattern). For example, in the dataset named *simple foreground*, there are only two objects, but in the *complex foreground* dataset, the number of objects is increased to ten objects. In the data set called the *simple background* stars do not move but in the *complex background* data set there are moving stars. 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 the parameters unmatched cost (c) and delay between two consecutive frames (d) 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 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 (the number of spots which are classified as unknown class will decrease). On the other hand, the delay between two consecutive frames increases, the greater the likelihood that there will be no corresponding in the previous frame (the spots will be classified in an unknown class). The bottom right figure shows the variance of this rate. The low values of this index indicate that the algorithm operates consistently on different frames (the error is not propagated over time). The middle row represents the mean (left panel) and variance (right panel) of the ratio of the number of spots correctly classified as stars to the total number of spots in the current frame (true negative rate per each frame). According to previous arguments, the higher c and lower d, the algorithm will have a better performance on this index, but since in each frame there is usually a much

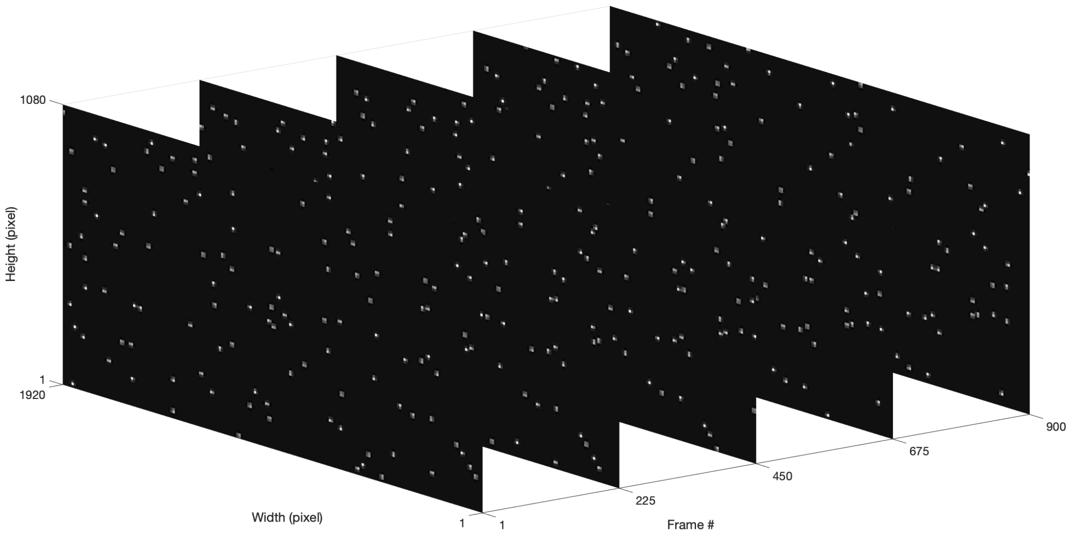


Fig. 2: Sample input data. Shown is a sequence of grayscale intensity image frames over time, represeting a few snapshots of the captured video sequence.

higher number of stars than the number of objects, this index will be more robust to the parameter c than previous index. The bottom row shows the mean (left) and variance (right) of the average ratio of the number of spots classified as unknown in the current frame (the algorithm could not find a corresponding match for these spots between the current and previous frames) to the total number of spots in that frame.

The source code for these results is available at [8].

4. DISCUSSION

ACKNOWLEDGMENT

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

REFERENCES

[1] E. Delande, C. Frueh, J. Franco, J. Houssineau, and D. Clark, “Novel multi-object filtering approach for space situational awareness,” *Journal of Guidance, Control, and Dynamics*, vol. 41, no. 1, pp. 59–73, 2017.

[2] 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*, 2016, p. 5500.

[3] 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.

[4] 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.

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

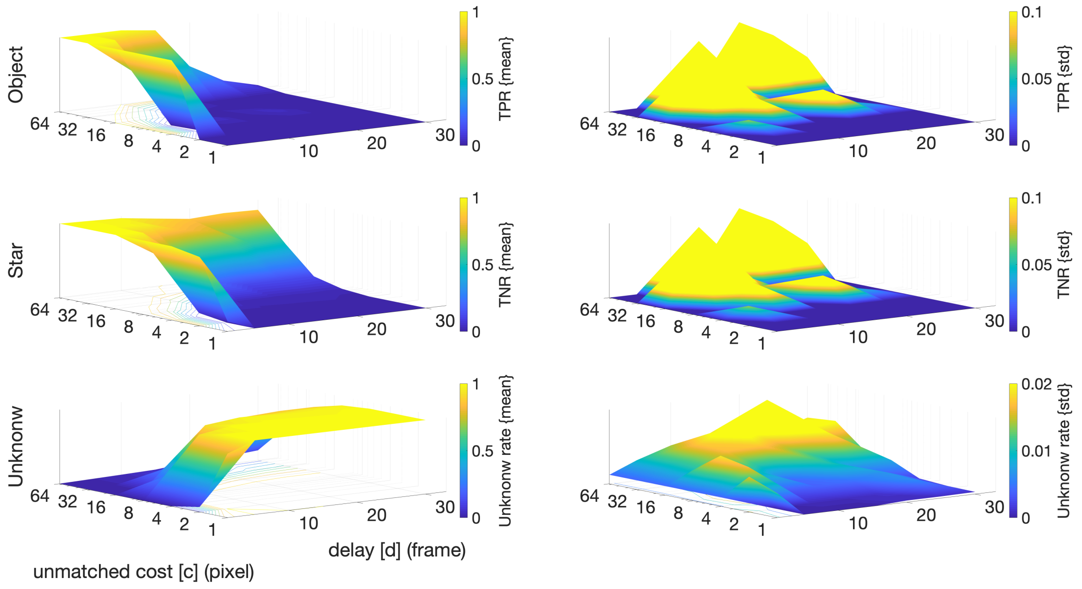


Fig. 3: Simple foreground (Two objects with similar moving speed and trajectory).

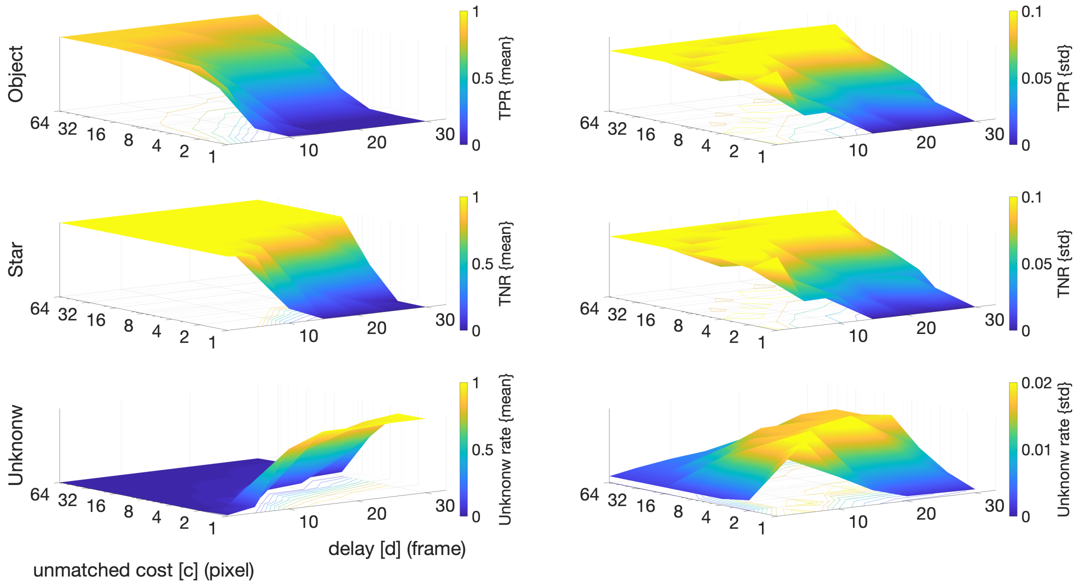


Fig. 4: Complex foreground (Ten objects moving in various directions and at various speeds).

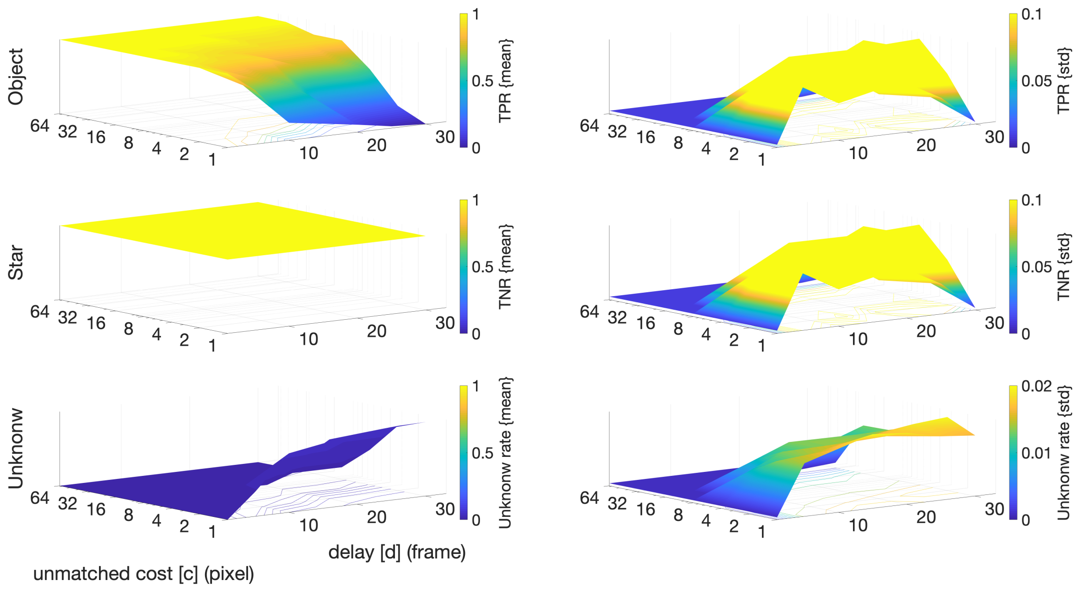


Fig. 5: Simple background (static stars).

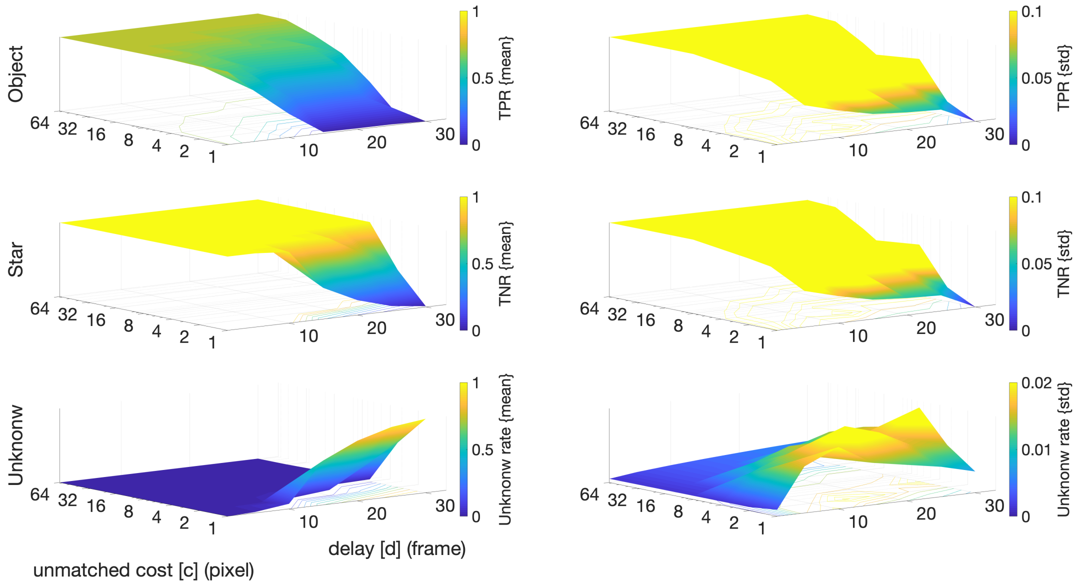


Fig. 6: Complex background (moving stars).

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

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

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

[9] 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.

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

[11] 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.

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

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

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

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