Article

Dim and Small Space-Target Detection and Centroid Positioning Based on Motion Feature Learning

Shengping Su [1,2], Wenlong Niu [1,2,*], Yanzhao Li [1,2], Chunxu Ren [1,2], Xiaodong Peng [1,2], Wei Zheng [1] and Zhen Yang [1]

[1] National Space Science Center, Chinese Academy of Sciences, Beijing 100190, China
[2] University of Chinese Academy of Sciences, Beijing 100049, China
* Correspondence: niuwenlong@nssc.ac.cn

Abstract: The detection of dim and small space-targets is crucial in space situational awareness missions; however, low signal-to-noise ratio (SNR) targets and complex backgrounds pose significant challenges to such detection. This paper proposes a space-target detection framework comprising a space-target detection network and a k-means clustering target centroid positioning method. The space-target detection network performs a three-dimensional convolution of an input star image sequence to learn the motion features of the target, reduces the interference of noise using a soft thresholding module, and outputs the target detection result after positioning via the offsetting branch. The k-means centroid positioning method enables further high-precision subpixel-level centroid positioning of the detection network output. Experiments were conducted using simulated data containing various dim and small space-targets, multiple noises, and complex backgrounds; semi-real data with simulated space-targets added to the real star image; and fully real data. Experiments on the simulated data demonstrate the superior detection performance of the proposed method for multiple SNR conditions (particularly with very low false alarm rates), robustness regarding targets of varying numbers and speeds, and complex backgrounds (such as those containing stray light and slow motion). Experiments performed with semi-real and real data both demonstrate the excellent detection performance of the proposed method and its generalization capability.

Keywords: space-target detection; dim and small target; centroid positioning; deep neural network; k-means clustering

check for updates

Citation: Su, S.; Niu, W.; Li, Y.; Ren, C.; Peng, X.; Zheng, W.; Yang, Z. Dim and Small Space-Target Detection and Centroid Positioning Based on Motion Feature Learning. *Remote Sens.* **2023**, *15*, 2455. https://doi.org/10.3390/rs15092455

Academic Editors: Yang Yang, Sanat K. Biswas and Xiaofeng Wu

Received: 27 February 2023
Revised: 29 April 2023
Accepted: 4 May 2023
Published: 7 May 2023

1. Introduction

The deterioration of the outer-space environment owing to the increasing number of space-targets in orbit, such as spacecraft, rocket bodies, and space debris, poses severe threats to spacecraft safety and operations [1,2]. In addition, near-Earth asteroids may pose a potential threat to Earth [3]. Consequently, these targets must be monitored. In this context, dim and small space-target detection is of substantial value. However, the detection of such targets entails several limitations, including the low signal-to-noise ratio (SNR) of the targets, which causes difficulties in discriminating these small targets from background noise; the minute size of the targets, which often corresponds to a few pixels with only gray-scale information and lacks features such as texture and shape; and the simultaneous appearance of multiple targets with different speeds and directions in the field of view.

To overcome the aforenoted limitations, this paper proposes a novel framework for dim and small space-target detection. The framework comprises a dim and small space-target detection network and a k-means target centroid positioning method. The detection network is an end-to-end model that accepts a sequence of star images as input and outputs coarse centroid-positioned bounding boxes containing targets, which is the interim result of the detection framework. This interim result is then fed into the k-means centroid positioning algorithm to obtain the final precisely located target detection result. Figure 1 llustrates the flowchart of the detection framework. The target detection network comprises two main structures: the backbone network and the bounding box offset branch. The backbone network consists of several modules called shrinkage convolution (SC) blocks, which perform a three-dimensional (3D) convolution of the input and reduce the noise effects using soft thresholding modules. At the end of the backbone network, there is a fully connected layer that obtains the initial bounding boxes of the target. These bounding boxes are then offset by the offset branch to derive an interim coarse positioning result. The k-means centroid positioning algorithm comprises bilinear interpolation, k-means foreground/background segmentation, and target centroid and window iterations, resulting in a highly accurate subpixel-level centroid positioning result and a corrected target bounding box.

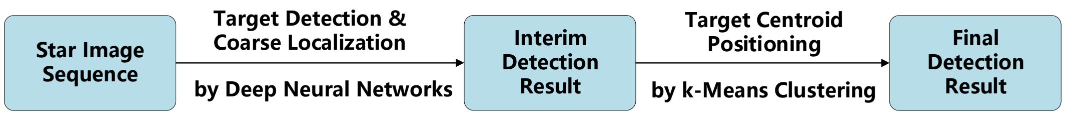

Figure 1. Flow chart of the proposed dim and small space-target detection framework.

The proposed detection framework is tested using simulated data, semi-real data, and fully real data. The simulated data contain dim and small space-targets with various attributes such as number; speed; SNR; multiple noises; and complex backgrounds with lots of stars, stray light, and slow motion. Semi-real data are the data of simulated targets with multiple attributes added to the real captured background star image. The fully real data are optical images containing space-targets taken by a professional observatory. Experiments on simulated data show that the detection network has excellent detection performance for targets with different SNRs (the detection rate reaches 99.75% when the SNR is three) and has extremely low false alarm rates (10^{-4} level) under all SNR conditions. The algorithm also has good robustness regarding different target numbers, speeds, and complex star backgrounds that include stray light and slow motion. The k-means centroid positioning method achieves an accuracy of 0.209 pixels when the target SNR is as low as 1.5 and can output accurate target bounding boxes. In addition, the overall detection algorithm shows excellent detection performance and generalization abilities for both semi-real and fully real data.

This paper is organized as follows: Section 2 provides a brief overview of related works; Section 3 introduces the proposed method; Section 4 presents the experimental validation of the proposed method; Section 5 discusses the experimental results; and finally, Section 6 presents the conclusions.

2. Related Work

This section provides a brief overview of the existing techniques for deep learning target detection, unsupervised semantic segmentation using k-means clustering, space-target detection using image data taken by optical telescopes, and point-like target centroid positioning.

2.1. Deep Learning Target Detection

In recent years, deep learning has achieved remarkable progress in target detection, particularly with the introduction of convolutional neural networks (CNNs). One of the earliest and most influential deep learning-based target detection methods is the R-CNN (Region-CNN) [4], which consists of three main stages: region proposal, feature extraction using a CNN, and classification using support vector machines (SVMs). Although the R-CNN method achieved state-of-the-art performance at the time, it was computationally expensive, with a slow detection speed. To overcome the limitations of R-CNN, the Fast R-CNN method [5] was proposed, which achieved a faster detection speed by sharing the feature extraction process across all target proposals. Then, the Faster R-CNN [6] method was proposed, which introduced a region proposal network (RPN) to generate target proposals and improved the accuracy of the detection task. Another significant development is the You Only Look Once (YOLO) [7] method. YOLO takes an image as input and predicts bounding boxes and class probabilities for targets in a single forward pass of a neural network, making it very fast and efficient. In 2018, the RetinaNet [8] method was proposed, which addressed the issue of class imbalance in target detection by introducing a novel focal loss function that focuses on hard examples during training. RetinaNet achieved state-of-the-art performance on various target detection benchmarks and is currently one of the most widely used target detection methods.

2.2. Unsupervised Segmentation Using K-Means Clustering

K-means is an unsupervised clustering algorithm with a simple idea: for a given sample set, divide the sample set into k clusters according to the distance between the samples, and let the points within the clusters be as closely connected as possible while letting the distance between the clusters be as far as possible. This idea can be expressed as follows: assume that the cluster is divided into $(C_1, C_2, \ldots, C_k)$; then, the goal is to minimize the squared error, E:

$$E = \sum_{i=1}^{k} \sum_{x \in C_i} \|x - \mu_i\|_2^2 \tag{1}$$

where x is the sample point, and μ_i is the mean vector of cluster C_i, also known as the cluster center, and it can be expressed as

$$\mu_i = \frac{1}{|C_i|} \sum_{x \in C_i} x \tag{2}$$

Since solving directly for the minimum of E is an NP-hard problem [9], the following iterative method is used to minimize E:

1. Randomly select k samples from all samples as initial cluster centers.
2. Update the cluster centers: iterate through all sample points and calculate the distance of each sample point from all cluster centers. For a sample point, the cluster center with the smallest Euclidean distance is taken as the class to which the sample point belongs.
3. Calculate the mean square error of the updated cluster center coordinates and the cluster center coordinates before the update. If the mean squared error is still greater than a certain threshold, repeat step 2; otherwise, end the iteration.

The segmentation of images using k-means is a simple, effective, and widely used approach [10,11]. The k-value, which is the number of regions to be segmented in the image, is preset, and the k-means algorithm will classify each pixel in the image. In this paper, k-means will be used to perform a two-class segmentation of targets and backgrounds for the images in the bounding box of the target.

2.3. Optical Space-Target Detection Methods

In general, optical telescopes have two primary modes of operation: target-tracking and sidereal-tracking modes. The former involves following a target along its trajectory or maintaining a stationary focus on a target that is in geostationary Earth orbit (GEO), where the target observed appears as a point-like object, and stars may appear as streak-like or point-like objects, depending on the exposure time. In contrast, the sidereal tracking mode aims to maintain the stable tracking of stars in the field of view, resulting in stars appearing as point-like objects and targets appearing as streak-like or point-like ones, depending on the exposure time. Figure 2 is a schematic diagram of point-like and streak-like targets imaged in this mode. Existing methods for detecting space-targets can generally be categorized into two groups: point-like target and streak-like target detection methods. Additionally, based on the specific technology employed for detection, these methods can be classified as model-driven or data-driven. The following sections focus on point-like target detection methods, categorized from both the model- and data-driven perspectives, as the proposed method falls into this category. A brief overview of streak-like target detection methods has also been provided.

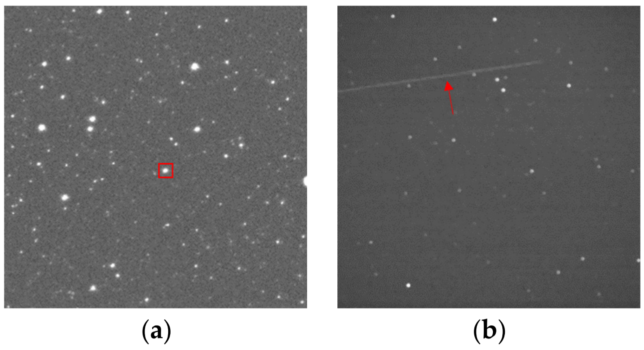

Figure 2. Schematic diagram of point-like and streak-like targets in a star image in sidereal tracking mode: (**a**) point-like target (marked by a square) and (**b**) streak-like target (marked by an arrow).

2.3.1. Model-Driven Method

Target detection is necessary because the "target-tracking" employed by telescopes merely fixates on a target region in near space. The targets have a point-like form due to the usually small relative velocity between a target in a particular region and the camera [12].

Point-like target detection using model-driven approaches typically involves capturing multiple successive images, detecting potential targets (referred to as "candidate targets"), and removing pseudo-targets by exploiting the distinctive motion patterns of targets relative to the background stars [13,14]. Frame-by-frame differencing and thresholding [15] are used for this purpose; however, they are unsuitable for detecting small and slow-moving space-targets. Reed et al. [16] proposed a weak motion target detection algorithm using a 3D-matched filter; however, this method required a specific known velocity profile. Several algorithms have been proposed to address this problem. Mohanty [17] demonstrated assembling potential trajectories of a weak point target by searching possible trajectories in frames and outputting the results based on the maximum likelihood ratio. Yanagisawa et al. [18] proposed the stacking method, which extracts sub-images from the image sequence based on the assumed trajectory of the space debris and removes stars by generating the median of all the sub-images. However, it takes significant time to detect invisible targets with unknown motions because of the need to assume and examine a range of possible paths. Addressing this limitation, Yanagisawa et al. [13] proposed the LINE method, which finds any series of targets arranged in a straight line in an image sequence. Although this approach analyzes data more quickly, it is not as effective as the stacking method for detecting faint targets. Sara et al. [14] proposed an efficient algorithm for detecting the trajectories of slow-moving targets using a trajectory sequence detection method similar to LINE, performed using the RANSAC robust method. Liu et al. [19] first use the maximum projection method and calculate the median image to remove stars; then, they use a target detection method based on projection time information to detect moving targets; and finally, they use inter-frame trajectory correlation to obtain the complete trajectory of the target. A two-stage detection algorithm was proposed by Jiang et al. [20]. The first stage uses wavelet transform and guided filtering to eliminate the effect of stars, and the second stage uses a robust principal component analysis approach to attribute the target detection problem to the separation of the target and background in a single frame image, which can effectively detect the target. Gao et al. [21] proposed a space debris detection method that first performs median filtering and improves top-hat filtering on the image, then extracts the centroid of the suspected target using connected region analysis while removing spurious targets using saliency features, and finally performs image alignment and data correlation to obtain the trajectory of the target. Zhang et al. [22] proposed a space object detection algorithm based on motion information using video image data as the input, and the space object trajectories are associated with the Kalman filter for all frames.

In point-like target detection, the track-before-detect (TBD) method can be used to handle darker targets. The TBD method first tracks possible targets in the image and then applies a discriminant based on the target characteristics to filter out the incorrect trajectories, finally arriving at the correct trajectory. The particle filter (PF) [23–25] is a natural framework for TBD solutions in which the probability of selecting a particle relies on the evaluation value of the likelihood function. Throughout the iterative process, the particles are gradually driven to regions with high likelihood values. Multilevel hypothesis testing (MHT) [26] is a common TBD method that organizes targets in an image sequence into candidate trajectory segments, constructs a tree structure, and prunes it using hypothesis testing to retain only consistent targets. Many subsequent improvements to MHT have been proposed, including Structured Branching-MHT [27], Multiple-MHT-Tracking [28], time-index multistage quasi-hypothesis-testing (TMQHT) [29], and Spatiotemporal Pipeline-MHT (SPMHT) [30]. However, good hypotheses are hard to design when little prior knowledge of the target and background models is available, with the resulting algorithms being complex and difficult to apply directly.

In streak-like object detection, matched filters are widely used to identify targets by detecting image regions with high responses to streak-like filters [31–33]. However, this method can produce numerous stripe templates with different directions and lengths, leading to long computation times and high computational costs. An alternative approach is to apply the TBD framework to locate streak-like objects by accumulating a quantity, such as pixel intensity values along the direction of the stripes, to enhance the final response [34,35]. Takayuki et al. [36] proposed a Trajectory Segment Integration Method for GEO space debris detection based on an inverse filter by multiplying the frequency axis. Other techniques have also been proposed, such as the Hough transform [37,38], Radon transform [39], and boundary tensor [40], for streak-like target detection. However, these methods are not effective for detecting diverse targets.

2.3.2. Data-Driven Method

The current study focuses on such a method. In recent years, data-driven methods for optical space-target detection have attracted the attention of several researchers. Do et al. [12] proposed a GEO faint target detection method based on Gaussian process regression (GPR) supplemented with topological scanning and linear fitting. GPR is used to remove the noise-laden background from the original image, resulting in good detection results. However, the linear fitting algorithm requires traversing all point pairs and has high time complexity. Several methods that apply convolutional neural networks (CNNs) to space-target detection have emerged to bypass the process of removing stars and constructing a first-level detection pipeline. Abey and Gupta [41] proposed a feature pyramid network (FPN)-based GEO target detection method with mask preprocessing and a custom vector mathematics-based post-processing algorithm to track objects across frames, achieving relatively high detection performance that minimized false detections. This method performs target detection on a single frame, and since it uses data where targets are point-like and stars are streak-like, it essentially classifies targets and stars based on shape features and is not applicable to data where both targets and stars are point-like. A number of neural network-based detection algorithms using single-frame star images have also been proposed. Bertin and Arnouts [42] trained a neural network on simulated images, and it enables the separation of stars and galaxies from the background. Xue et al. [43] proposed a fully convolutional neural network to implement pixel-by-pixel classifications. This method can quickly complete target background separation in a single stage and is robust regarding noisy and inhomogeneous backgrounds, with a high detection rate and fewer false alarms. Jia et al. [44] proposed a detection and classification framework based on deep neural networks for images obtained with wide-field small-aperture telescopes. This method borrows ideas from Faster R-CNN by first extracting image features using a modified ResNet-50; then, it proposes regions containing celestial objects using a regional proposal network (RPN) and subsequently performs classification and position regression on these regions. Experiments demonstrate that this method has two times better detection performance for dim targets than for traditional methods and can accurately classify all celestial targets. The latter three algorithms [42–44] do not directly detect specific space-targets of interest in star images (such as orbiting satellites, space debris, or near-Earth asteroids), instead detecting all targets in the image (including stars). This is because they use only a single image for detection and do not learn the motion features of the target from the image sequence, thus failing to distinguish the space-targets of interest from the stars.

Tao et al. [45] proposed a spatial small-fragment saliency detection method based on deep convolutional networks, which input the local contrast map of space debris together with the frame pairs in the video into a convolutional network to obtain a spatiotemporal domain saliency map of space debris to detect the target. Continuing the idea of saliency detection, Tao et al. [46] very recently proposed another end-to-end space debris detection network called SDebrisNet, which takes video as the input and obtains detection results for moving space-targets by extracting and fusing spatial and temporal features separately. This network uses MobileNet as the backbone and incorporates an attention mechanism to effectively detect dim and small targets, but its applicability to multi-targets or complex backgrounds containing stray light or slow-moving stars is unclear.

For streak-like object detection, Varela et al. [47] proposed a YOLO-based approach, demonstrating superior results compared with classical methods, such as the Hough transform. Duev et al. [48] presented a deep learning system called DeepStreaks, a CNN designed to efficiently identify fast-moving streaking near-Earth objects. This system achieved a detection rate of 96–98% while maintaining a false-positive rate of less than 1%.

2.4. Centroid Positioning Methods for Point-like Target

Several previous studies on centroid positioning have yielded significant research findings. These centroid positioning methods include grayscale weighting [49], Gaussian surface fitting [50], and ellipse fitting [51].

Among these, the grayscale weighting method considers the relationship between the pixel values and weights of the centroid positioning. The traditional method calculates the centroid of the target by using the grayscale values of the pixels as weights. The threshold grayscale method improves upon the traditional method by adding a background suppression threshold, which reduces the influence of stray light and enhances the accuracy of the centroid positioning process. The squared weighted method uses the square of the grayscale value of a pixel as the weight, increasing the target weight and decreasing the noise weight, resulting in more accurate positioning. This method also reduces the influence of the edges.

The Gaussian surface-fitting method utilizes a Gaussian function to simulate the point spread function in the imaging process, with the point spread function used to locate the center of mass of the stellar field. Although this method is stable, its calculations are relatively complicated.

Finally, the ellipse fitting method considers the stellar field as an ellipse, extracts the edges of the stellar field using morphological methods, and locates the centroid using a least-squares fit. However, this method is unstable.

3. Proposed Method

This section presents the details of the proposed method. The framework of the proposed method is shown in Figure 1 and comprises two main steps: (1) space-target detection using a deep neural network and providing a coarse localization of the target and (2) accurate centroid positioning of the space-target using a k-means clustering algorithm. This section first introduces the star image data used in the proposed method for training and testing. Then, the structure of the deep neural network for space-target detection is described (step 1). Next, the proposed method for accurate target centroid positioning using k-means clustering is presented (step 2). Finally, the implementation and training details of the target-detection network are highlighted.

3.1. Image Modelling

According to [30], an optical image can be modeled in the form of

$$G(x,y) = T(x,y) + S(x,y) + B(x,y) + N(x,y), \tag{3}$$

where $G(x,y)$ represents the grayscale value of the pixel at the coordinate (x,y). $T(x,y)$ and $S(x,y)$ denote the space-target and stars, respectively. The background, denoted by $B(x,y)$, is frequently nonuniform because of the presence of stray light and signals from different charge-coupled device (CCD) channels. Noise, $N(x,y)$, is usually a combination of several types of noise, such as photonic noise generated by an optical system, dark current noise, and readout noise generated by the detector. In addition, there is hot pixel noise, which is an individual pixel with a particular brightness well above the main distribution of the sensor noise. Later, simulation datasets based on the model in this section are also generated, as presented in Section 4.1.1.

All blobs of stars and targets can be modeled using a 2D Gaussian distribution [43], as defined in Equation (4):

$$G(x,y) = A\ exp\left[-\left(\frac{(x-x_c)^2}{2\sigma_x^2} + \frac{(y-y_c)^2}{2\sigma_y^2}\right)\right], \tag{4}$$

where $G(x,y)$ denotes the grayscale value of the pixel at (x,y), and the coefficient A is the amplitude. (x_c, y_c) indicates the coordinates of the center of a star or target, and σ_x, σ_y is the standard deviation that represents the spread of the blob and controls its size. For stars in the simulation images, their positions and grayscale values were generated based on the SAO Star Catalog [52,53].

For noise in the simulated image, Gaussian noise was added to simulate the CCD dark current noise, readout noise, and Poisson noise to simulate photon noise, with hot-pixel noise also added.

For the background, stray light and slow motion were primarily considered. Two types of stray light were simulated: linear and Gaussian. The slow movement of the background was simulated by subtly changing the pointing coordinates of the center of the field of view.

For each target in the simulation image, its size was arbitrarily set to 1×1, 3×3, 5×5, or 7×7 pixels. According to [43], SNR was used to indicate the darkness of the target, calculated as

$$SNR = \frac{\mu_T - \mu_B}{\sigma_B}, \tag{5}$$

where μ_T and μ_B are the average grayscale values of the star and neighboring regions, respectively, and σ_B represents the standard deviation of the grayscale value in the neighboring background region, as shown in Figure 3. The shaded area between adjacent rectangles of the target area and the larger rectangle was considered the background. In the simulation, $r = 5$ was set to obtain μ_T, μ_B, and σ_B. The SNR of the target can also be set arbitrarily.

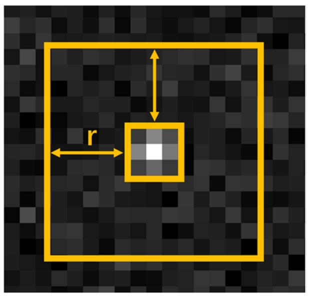

Figure 3. Schematic representation showing the calculation of the target SNR.

3.2. Target Detection Network

The architecture of the proposed detection network is shown in Figure 4a. The input of the detection network is the original star image sequence, and the output is the detected image sequence, where the target position on each frame is marked with a bounding box. The proposed network contains two main architectures: the backbone network and the branch path. The backbone network is responsible for detecting moving space-targets in the input sequence and can determine the position of the moving target in each frame, configuring a bounding box with a very coarse position. The branch path is responsible for offsetting the center of the bounding box to obtain a more precisely positioned bounding box, which is the interim result of the entire detection framework.

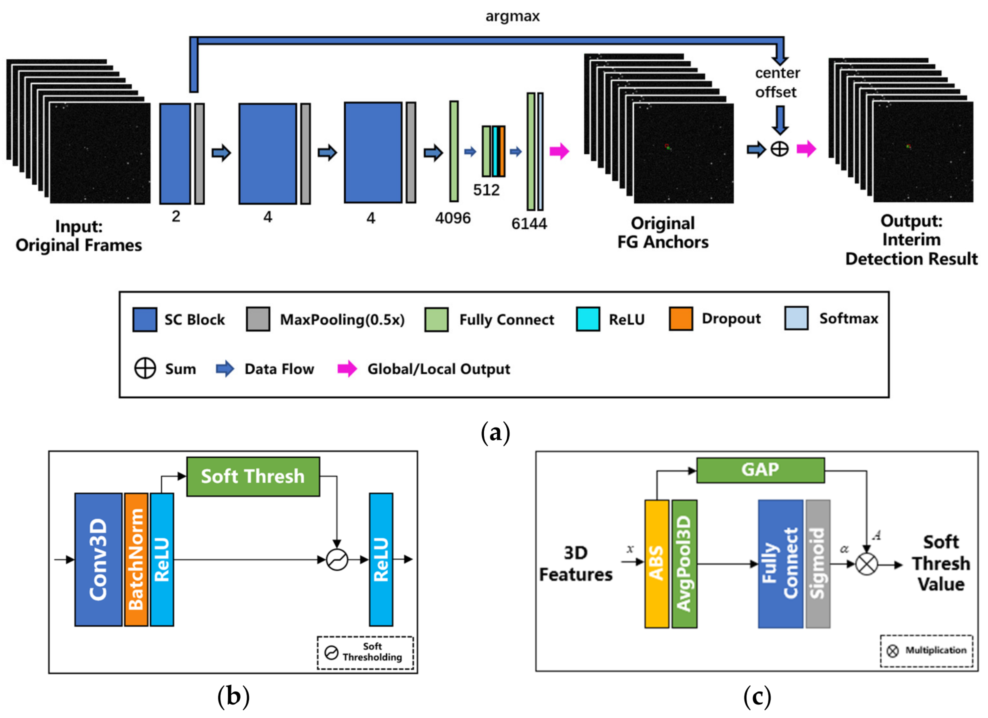

Figure 4. (**a**) Architecture of the proposed detection network, (**b**) the internal architecture of an SC Block, and (**c**) the internal architecture of a soft thresholding module.

The backbone network borrows the design idea of the RPN in Faster RCNN. Several square boxes (called anchors) were pre-placed with preset sizes on each frame of the input image sequence. The function of the backbone network is to extract the motion features of the targets in the input image sequence using 3D convolutional layers and predict the anchors on each frame that contains the moving targets, that is, the positive anchors. Specifically, each input image frame of size $H \times W$ pixels is divided into $n \times n$ grids so that each grid has a size of $(H/n) \times (W/n)$, and k square anchors with different scales are placed at the center of each grid. By default, $H = W = 256$, $n = 32$, and $k = 3$ were set, with the sizes of the three anchors being 16×16, 8×8, and 4×4, respectively, as shown in Figure 5. This design ensures that targets with different sizes and locations are likely to be detected by a particular anchor; for example, targets located at grid junctions or corner points can be surrounded by anchors of size 16×16 without being missed. With this design, 3072 anchors were placed on each input image frame.

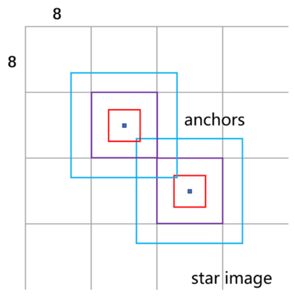

Figure 5. Diagram of the anchors placed on the star image. The sizes of the blue, purple, and red anchors in the figure are 16 × 16, 8 × 8, and 4 × 4, respectively, with dots representing the center of the grid.

The backbone network was designed using the architecture described below. It comprises several repeatedly applied modules called SC Blocks, each followed by a 3D max-pooling layer for down-sampling. Because too many pooling layers can lead to the loss of the spatial features of dim targets, this study used only a combination of three SC Blocks and 3D max-pooling layers in series for feature extraction. Subsequently, a fully connected neural network was used to implement positive/negative classification for each anchor. The details of the network architecture are described below, and the hyperparameter settings are listed in Table 1.

Table 1. Hyperparameter settings for the architecture of the backbone network.

Components of Backbone	Module Inside	Hyperparameters	Output Size (C × D × H × W)
Input	-	-	1 × 8 × 256 × 256

SC Block 1	Conv 3D
	
	Soft Thresh
		kernel = (3 × 3 × 3) × 2
		stride = 1 × 1 × 1
		padding = 1 × 1 × 1
		See Table 2
			2 × 8 × 256 × 256

Max Pool 3D 1	-	kernel = 2 × 2 × 2
		stride = 1 × 1 × 1
			2 × 4 × 128 × 128

SC Block 2	Conv 3D
	
	Soft Thresh
		kernel = (3 × 3 × 3) × 4
		stride = 1 × 1 × 1
		padding = 1 × 1 × 1
		See Table 2
			4 × 4 × 128 × 128

Max Pool 3D 2	-	kernel = 2 × 2 × 2
		stride = 1 × 1 × 1
			4 × 2 × 64 × 64

SC Block 3	Conv 3D
	
	Soft Thresh
		kernel = (3 × 3 × 3) × 4
		stride = 1 × 1 × 1
		padding = 1 × 1 × 1
		See Table 2
			4 × 2 × 64 × 64

Max Pool 3D 3	-	kernel = 2 × 2 × 2
		stride = 1 × 1 × 1
			4 × 1 × 32 × 32

FC	FC Layers	layer = {4096, 512, 6144}	4096

The internal structure of an SC Block is shown in Figure 4b, where the 3D convolutional layer and soft thresholding module are the two core components. The 3D convolutional layer is the foundation for the network to achieve moving target detection. As shown in Table 1, the number of 3D convolution kernels was set to two or four because, empirically, the number of convolutional kernels is related to the number of features that may be abstracted from the image, and space-targets do not have rich features such as texture and shape. The detection results of the network prove that the settings used are sufficient.

Table 2. Hyperparameter settings for the structure of the soft thresholding module.

Soft Thresh Module	Components	Hyperparameters	Output Size
In SC Block 1	Input	-	$2 \times 8 \times 256 \times 256$
	GAP	$1 \times 1 \times 1$	$1 \times 1 \times 1$
	Average Pool 3D	{kernel = $2 \times 2 \times 2$, stride = $2 \times 2 \times 2$} $\times$ 3	$2 \times 1 \times 32 \times 32$
	FC	layer = {2048, 256, 2}	2×1
	Output	-	2×1
In SC Block 2	Input	-	$4 \times 4 \times 128 \times 128$
	GAP	$1 \times 1 \times 1$	$1 \times 1 \times 1$
	Average Pool 3D	{kernel = $2 \times 2 \times 2$, stride = $2 \times 2 \times 2$} $\times$ 2	$4 \times 1 \times 32 \times 32$
	FC	layer = {4096, 256, 4}	4×1
	Output	-	4×1
In SC Block 3	Input	-	$4 \times 2 \times 64 \times 64$
	GAP	$1 \times 1 \times 1$	$1 \times 1 \times 1$
	Average Pool 3D	{kernel = $2 \times 2 \times 2$, stride = $2 \times 2 \times 2$} $\times$ 1	$4 \times 1 \times 32 \times 32$
	FC	layer = {4096, 256, 4}	4×1
	Output	-	4×1

The image sequences input into the backbone network contain a large amount of noise, which can drown out dim targets and thus affect the target detection performance of the network. Inspired by the deep residual shrinkage network [54], a soft thresholding module was added to the SC Block to reduce the interference of noise-related features. Soft thresholding is a frequently used denoising technique in signal processing, used to set the absolute value of features below a certain threshold to zero and adjust other features toward zero, i.e., "shrinkage". Figure 6a shows the relationship between the input and output of soft thresholding.

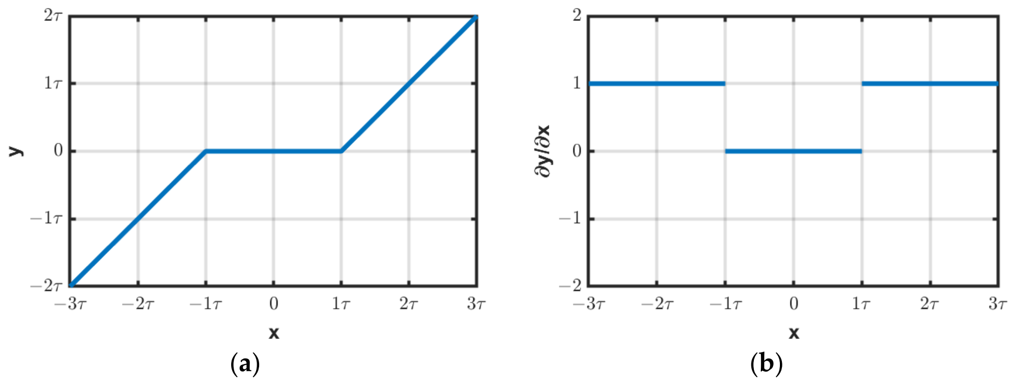

Figure 6. (**a**) Relationship between the input and output of soft thresholding and its (**b**) derivative.

Figure 6b shows that the derivative values of this function are zero and one, which can effectively prevent the gradient from exploding or vanishing. The basic module of the backbone network is the "Shrinkage Convolution Block", or "SC Block" for short, because of the "shrinkage" property of soft thresholding. The soft thresholding module is a small subnetwork with the structure shown in Figure 4c, and its hyperparameters are listed in Table 2.

The role of this subnetwork is to adaptively set the threshold. The 3D feature map input into this subnetwork is first taken at absolute value and then fed into two paths. One performs a global average pooling (GAP) operation to obtain an average of the absolute values of the feature map, A, which is a scalar. The other path first performs a 3D average pooling operation, with the pooling result then fed into a fully connected network and activated by sigmoid to obtain tensor α, where each element is a number in the range (0, 1). Subsequently, A is multiplied by α to obtain an output soft threshold, τ, each element of which represents the soft threshold of the corresponding channel. The feature map output by the 3D convolutional layer will then be soft-thresholded according to τ as the output feature map of an SC Block.

The feature map derived in the feature extraction stage is flattened into a 4096-d vector and fed into a fully connected network, which is used to classify each anchor placed on each input image frame to determine whether it contains a target. All anchors predicted to be positive are retained.

However, the original positive anchors derived at this point are only a rough localization result of the target. Therefore, a branch in the network structure was designed to offset the center of the roughly localized positive anchors to optimize localization accuracy. As shown in Figure 4a, the working process of this branch is simple: it finds the coordinates of the pixels with the maximum value (one or more, depending on the number of targets) on each frame in the feature map output by the first SC Block and considers this coordinate to be the target position. This is because the backbone network extracts the motion features of the target in the spatial–temporal domain; thus, the moving targets will have the maximum feature responses in the shallow feature map. Then, the difference between the found target positions and the center positions of the positive anchors in each frame is calculated, and the centers of the positive anchors are moved to the found positions according to the difference. Finally, nonmaximal suppression (NMS) is performed on these anchors based on the predicted probabilities. This positioning approach is similar to a common method used in critical point detection tasks: locating the target position based on the location of extreme points in a heatmap [55–57]. Note that only the center of the positive anchor was moved, and the bounding area of the anchor was not adjusted. The operation for adjusting the bounding region of the anchor is described in Section 3.3. After offsetting the center of the positive anchors, the network output is obtained, as shown in the part of Figure 4a denoted as "Output". This result is the interim result of the overall target detection framework.

3.3. K-Means Clustering Target Centroid Positioning

The interim results obtained via the detection network must be further improved in terms of their positional accuracy. This section presents a novel centroid positioning method based on k-means clustering, which can obtain subpixel-level positioning for the target centroid and accurately correct the range of the bounding box. This method has the following main characteristics: (a) because the target contains few pixels, which results in low centroid positioning accuracy, bilinear interpolation was performed on the image in the bounding box to increase the number of pixels in the target to improve the centroid positioning accuracy; (b) k-means clustering was used to segment the foreground and background of the interpolated image and find the target region; and (c) because noise pixels may affect the segmentation, the study adopted the iterative target region window to reduce the centroid positioning error caused by noise. The specific process of the centroid positioning method is as follows:

1. Bilinear interpolation is performed on the image inside the positive anchor to expand both the rows and columns by a factor of four, as shown in Figure 7a,b. Usually, the positive anchor output from the detection network surrounds the target or is already very close to the target, as shown in Figure 8.
2. Two-class clustering is performed on the interpolated image using k-means to segment the target and background regions and to obtain the respective cluster centers, which are two grayscale values representing the average of the pixel grayscale values of the two regions. The segmentation results of the target and background regions obtained from clustering are shown in Figure 7c.
3. The centroid coordinates of the target are derived via the cluster center and the window of the target region, as shown in Figure 7d,e.
4. The error, e, between the centroid coordinates of the target and the geometric center coordinates of the target region window is calculated. If e is greater than the threshold t, the center of the window is shifted to the centroid; $t = 0.1$ is set by default.

5. The k-means clustering is performed again in the new window, and new target centroid coordinates and a new window are obtained, along with a new error, e'.
6. This process is iterated until $e' < t$ to obtain the final centroid coordinates and the final window. This step generates the output of the centroid coordinates and converts the window to the original image for output as a bounding box.

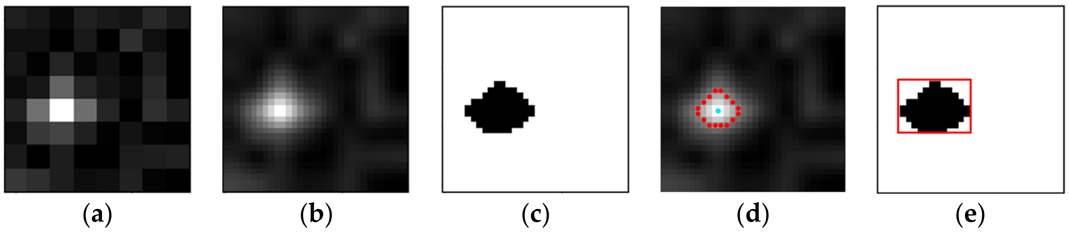

Figure 7. (**a**) Original image within the positive anchor; (**b**) image after bilinear interpolation; (**c**) results of target/background segmentation after k-means clustering; (**d**) the red dots represent the m pixels whose grayscale value is closest to the cluster center of the target region, and the blue dot represents the center derived by averaging the coordinates of the m pixels; and (**e**) the window of the target region (the red rectangle) obtained by calculating the outer rectangle of the outer contour of the target region.

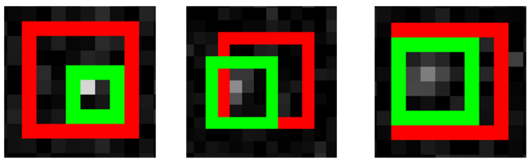

Figure 8. Examples of the positional relationship between the positive anchor output by the detection network (the red anchors) and the ground truth of the target (the green boxes).

Specifically, in step 3, the method to obtain the centroid coordinates is as follows: First, the cluster center of the target region is determined from the two cluster centers of the k-means, based on the premise that the value of the cluster center of the target region is greater than the value of the cluster center of the background, which also indicates that this method needs to be used on images where the SNR is greater than one to obtain the best performance. Then, the coordinates of the m pixels, whose grayscale value is closest to the cluster center of the target region, are averaged to obtain the centroid coordinates of the target. $m = 16$ was set as the default, as shown in Figure 7d. The window of the target region was obtained by calculating the outer rectangle of the outer contour of the target region, as shown in Figure 7e.

3.4. Training the Detection Network

3.4.1. Label Assignment, Handling Imbalance, and Loss Function

To train the detection network, a binary label was assigned to each anchor to indicate whether the anchor contained a target. A label assignment strategy was adopted similar to Faster RCNN: if the intersection-over-union (IoU) of an anchor with any of the ground truth boxes is greater than 0.5, a positive label is assigned to this anchor, or, if the IoU of an anchor with any of the ground truth boxes is less than 0.1, then a negative label is assigned to the anchor. Anchors that are neither positive nor negative are discarded and do not participate in training. Usually, an image contains only one target, and the rest of the image is background, which makes the small space-target detection task an extremely unbalanced problem in terms of classes. For all anchors placed on the image, the proportions of positive and negative anchors were approximately 0.09% and 8.1%, respectively. To solve this problem, focal loss [8] was used as a loss function, which is defined as

$$L(p_t) = -\alpha(1 - p_t)^{\gamma}\log(p_t), \tag{6}$$

where α and γ are the parameters used to balance the positive and negative samples and balance the hard and easy samples, respectively. p_t is defined as

$$p_t = \begin{cases} p, & \text{if } y = 1 \\ 1 - p, & \text{otherwise} \end{cases}, \tag{7}$$

where p is the probability predicted by the model for class with $y = 1$, and y denotes the binary label. In the training, $\alpha = 0.02$ was set to balance imbalanced samples, and γ was set to two according to [8].

3.4.2. Training Details and Implementation

The training process adopted the same approach as Faster RCNN to handle the anchors that cross the image boundaries; they do not contribute to the loss. After ignoring the cross-boundary anchors, 2947 anchors were left on the image to participate in the training. However, during testing, all anchors on the image are predicted, which may produce positive anchor predictions across the boundary, with these anchors cropped into the image boundary.

The proposed detection network was implemented using PyTorch, with the Adam optimizer [58] employed for network weight optimization. The training lasted 100 epochs, with the initial learning rate set to 0.01. The learning rate decay was set at every 10 epochs with a decay factor of 0.5. The weights of the 3D convolution kernel were initialized, and the fully connected layer was located in front of the rectified linear unit or ReLU in the network based on He initialization [59], with the parameters of the other fully connected layers based on Xavier initialization [60]. The training and test sets contained 20,000 and 2000 simulated star image sequences, respectively, each with 8 frames and a size of 256 × 256 pixels per frame. It took approximately 10 h to train on a single NVIDIA GTX1080Ti GPU.

4. Experimental Results

This section details the experiments performed to verify the performance of the proposed space-target detection method. Experiments on simulated, semi-real, and real data were conducted in the entire detection framework. These experiments were performed, respectively, on a simulated dataset, a semi-real dataset comprising real captured star images with the simulated target added, and fully realistic images with space-targets. In the experiments on simulated data, the detection performance of the proposed method was first verified under different conditions by controlling different simulation parameters and comparing it with other algorithms for dim and small target detection. Then, the ablation experiments were performed to verify the effectiveness of the key structures of the detection network. Finally, experiments were conducted on the performance of the centroid positioning method in the detection framework and compared with other classical centroid positioning methods. In the experiments on semi-real and real data, experiments concerning the performance of detection and centroid positioning were also carried out on the entire detection framework.

To assess the detection performance in a quantitative manner, evaluation metrics were used, including the detection rate, P_d; the false alarm rate, P_f; receiver operating characteristic (ROC) curves; and the area under the curve (AUC).

The detection rate and false alarm rate are calculated based on true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN) and are defined as

$$P_d = \frac{TP}{TP + FN} = \frac{TP}{GT} \tag{8}$$

$$P_f = \frac{FP}{FP + TN} = \frac{FP}{BG}, \tag{9}$$

where *TN* denotes the number of false alarms correctly classified, *FN* denotes the number of true targets not correctly predicted, *TP* denotes the number of true targets detected, *FP* denotes the number of false alarms misclassified as targets, *GT* denotes the total number of true targets, and *BG* denotes the number of background pixels.

The ROC curve is a widely used performance metric for target detection tasks and reflects the detection rate of an algorithm at different false-alarm rates. The AUC is the area under the ROC curve and is an evaluation metric that combines the probability of detection and the false alarm rate, reflecting the global performance of the detection algorithm. A larger AUC value indicates a higher overall detection performance.

To quantitatively evaluate the centroid positioning performance, the average positioning error of multiple positioning tests was used to characterize the performance. The positioning error, e_p, is the distance between the target centroid position calculated by the algorithm and the true centroid position of the target, which is defined as

$$e_p = \sqrt{(x - x_{true})^2 + (y - y_{true})^2}, \tag{10}$$

where (x, y) represents the centroid coordinates of the target calculated by the algorithm, and (x_{true}, y_{true}) represents the true centroid coordinates of the target.

4.1. Experimental Image Preparation

4.1.1. Simulated Image Generation

Because the number of real star images is very limited, a large number of simulated star images were generated for training and testing the target detection network based on the image model described in Section 3.1. The training and test sets contained 20,000 and 2000 image sequences, respectively, as described in Section 3.4.2.

Some of the simulation results are shown below. Figure 9 shows the simulation results for typical elements of the image. Figure 10 shows three examples of the simulated images.

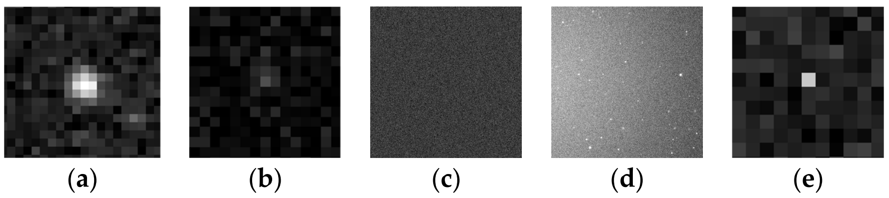

Figure 9. Simulation results for typical elements of the image: (**a**) bright star; (**b**) dim target; (**c**); noise; (**d**) background; and (**e**) hot-pixel.

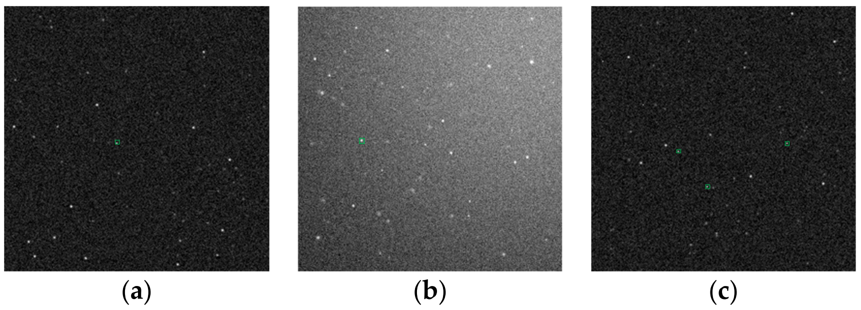

Figure 10. Examples of simulated images: (**a**) a target and a background containing no stray light; (**b**) a target and a background containing stray light; and (**c**) three targets and a background containing no stray light. The targets are marked with green boxes.

4.1.2. Real Star Image Preparing

In the experiments, in addition to the simulated star images, real star image data captured by an optical telescope (employed in this study) were used. The telescope had an aperture of 279.4 mm, a focal length of 2800 mm, a focal ratio of f/10, and a field of view of

0.8°. A Canon EOS R5 camera was attached to a telescope to capture images of the stars using a CMOS sensor with a single pixel size of 4 µm.

The shots were taken on a clear night, and the telescope worked in sidereal-tracking mode, such that the stars appeared point-like in the images. Examples of real star images are shown in Figure 11.

Figure 11. Examples of real star images.

4.2. Experiments on Simulated Data

The simulation experiments were performed on the simulation dataset, which allowed for free control of many of the simulation parameters, including the SNRs, number, size, speed of targets, presence or absence of stray light, and slow motion of the background. In the simulation, we set the SNR of the target = {3, 1.5, 1, 0.7}, the number of targets = {1, 2, 3}, the size of the target = {1, 3, 5, 7} square pixels, and the speed of targets = {1, 3, 6, 9} pixels between two adjacent frames and generated backgrounds with and without stray light or slow motion. Figure 12 shows examples of the simulated targets with different SNRs.

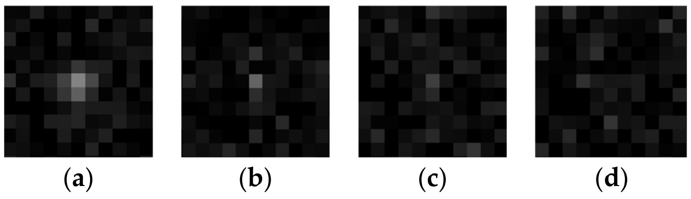

Figure 12. Examples of simulated targets with different SNRs. (**a**) SNR = 3; (**b**) SNR = 1.5; (**c**) SNR = 1; and (**d**) SNR = 0.7.

4.2.1. Experiments of Target Detection Performance

Experiments were conducted using the proposed overall detection framework to test its detection performance under different simulation parameters and compared with other detection methods for small, dim targets such as Max–Mean [61], Top-Hat [62], the Local Contrast Method (LCM) [63], the Multiscale Patch-based Contrast Measure (MPCM) [64], the Spatial-Temporal Local Difference Measure (STLDM) [65], and Infrared Patch-Image (IPI) [66]. The STLDM uses image sequences as inputs, while other methods use single-frame images as inputs. Here, only the target detection performance of the algorithm was focused on and not its centroid positioning performance, which is evaluated in Section 4.2.3.

Detection Performance of Targets with Different SNRs

Experiments were conducted on the test data with different SNR targets, and the parameter settings for the experiments are listed in Table 3.

Table 3. Parameter settings for the experiments of detection performance of targets with different SNRs.

Target				Background	
SNR	**Number**	**Size**	**Speed**	**Stray Light**	**Slow Motion**
3, 1.5, 1, 0.7	1	7, 5, 3, 1	1	No	No

Figure 13a–d show example results for four different target SNRs. ROC curves for various SNRs are illustrated in Figure 14, drawn using the raw output of each algorithm. Table 4 shows the comparative experimental results of the detection performance, which are the average results of 2000 image sequences from the test set. The detection rate of each comparison method was made as close as possible to our detection rate by adjusting the binarization threshold to compare the false alarm rates. Figures 15–18 show the binarized trajectory plots of the output of each algorithm.

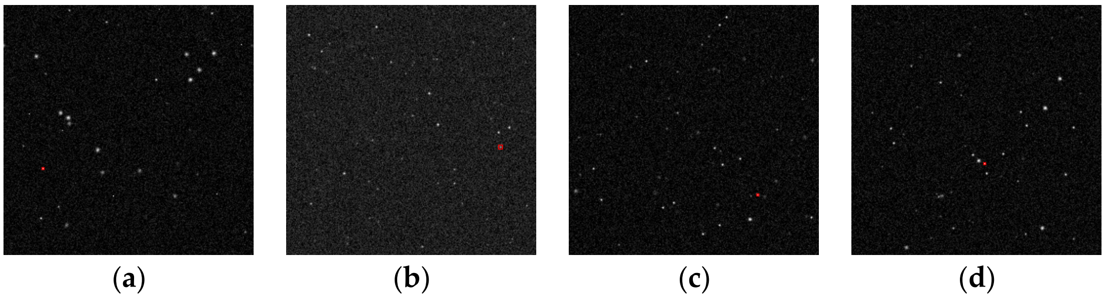

Figure 13. Detection results on the first frame of four image sequences containing targets with different SNRs. (**a**) SNR = 3; (**b**) SNR = 1.5 (**c**) SNR = 1; and (**d**) SNR = 0.7. The targets were enclosed by the red bounding box output using our algorithm.

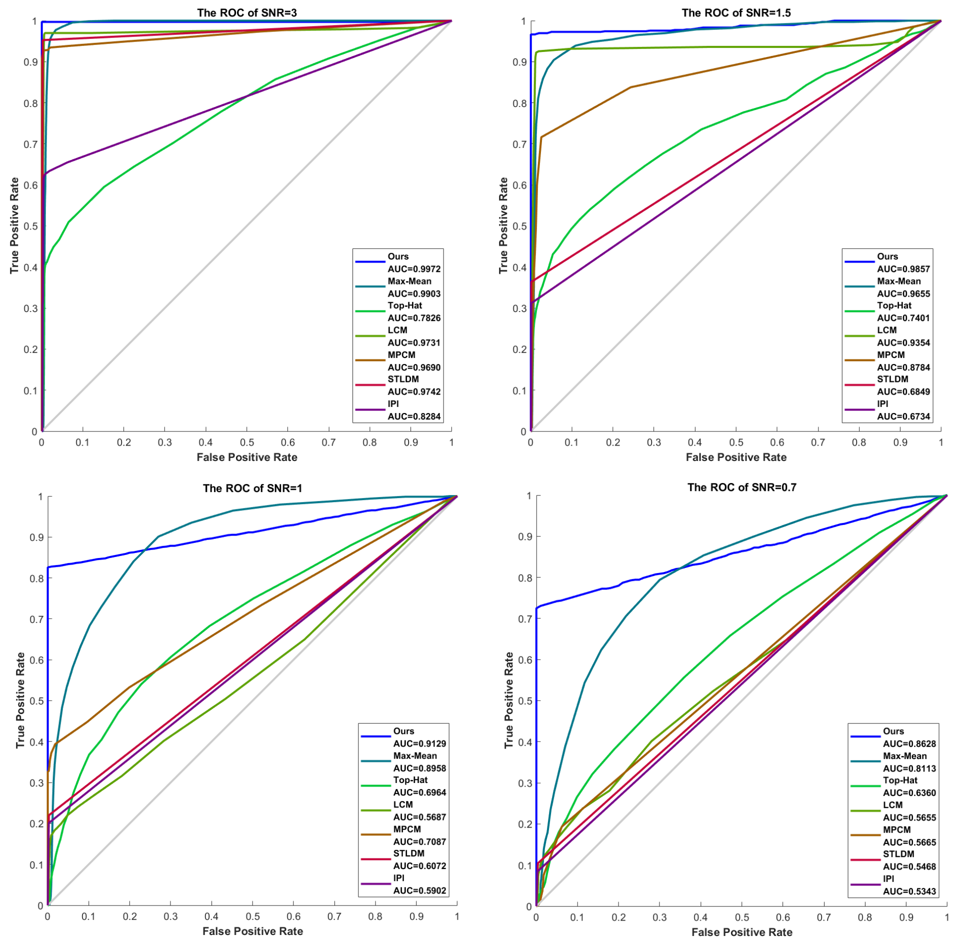

Figure 14. The ROC curves for different SNRs.

Table 4. Detection performance results of targets with different SNRs.

SNRs	Metrics	Ours	Max–Mean	Top-Hat	LCM	MPCM	STLDM	IPI
3	Detection Rate (%)	99.75	96.5	95.25	98.875	96.375	98.5	98.125
	False Alarm Rate (%)	**0.0243**	0.5955	37.1474	1.2897	1.2183	0.2328	0.2074
1.5	Detection Rate (%)	96	96.375	96.625	93	97.25	95.75	93.875
	False Alarm Rate (%)	**0.0206**	0.7234	39.0416	1.4999	8.7156	0.3196	0.2179
1	Detection Rate (%)	83	81.875	85.25	82.5	82.75	84.875	68.875
	False Alarm Rate (%)	**0.0260**	0.8782	8.2603	1.6380	3.0551	2.1185	0.2378
0.7	Detection Rate (%)	61.5	61.25	61.25	60.5	61.75	62.5	30.25
	False Alarm Rate (%)	**0.0662**	5.5468	4.5269	2.5264	4.0263	2.2766	0.2646

Bold numbers represent the best data in the row.

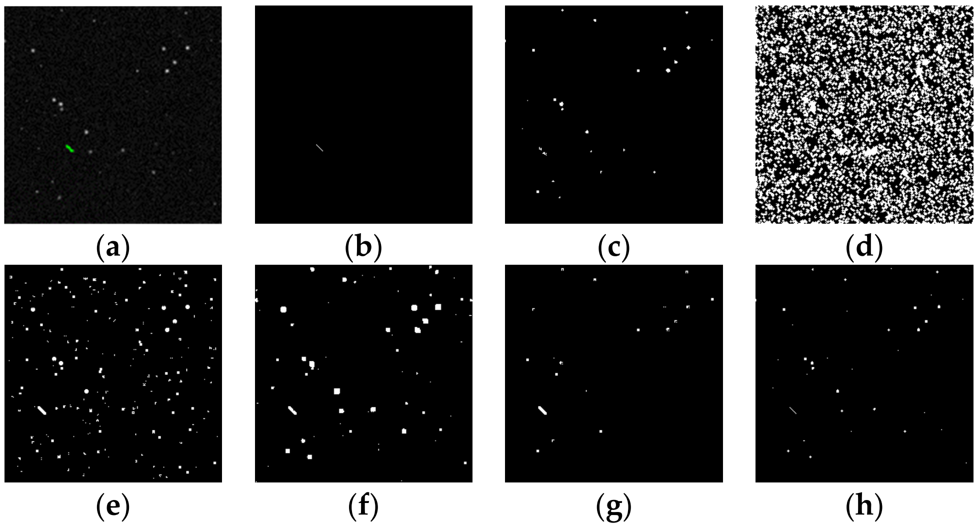

Figure 15. Detection results of target SNR = 3: (**a**) ground truth; (**b**) ours; (**c**) Max–Mean; (**d**) Top-Hat; (**e**) LCM; (**f**) MPCM; (**g**) STLDM; and (**h**) IPI.

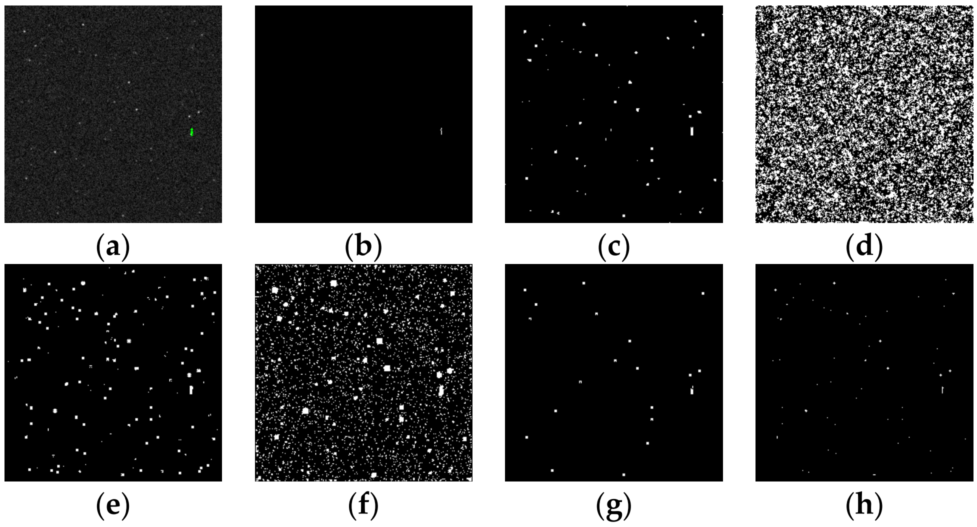

Figure 16. Detection results of target SNR = 1.5: (**a**) ground truth; (**b**) ours; (**c**) Max–Mean; (**d**) Top-Hat; (**e**) LCM; (**f**) MPCM; (**g**) STLDM; and (**h**) IPI.

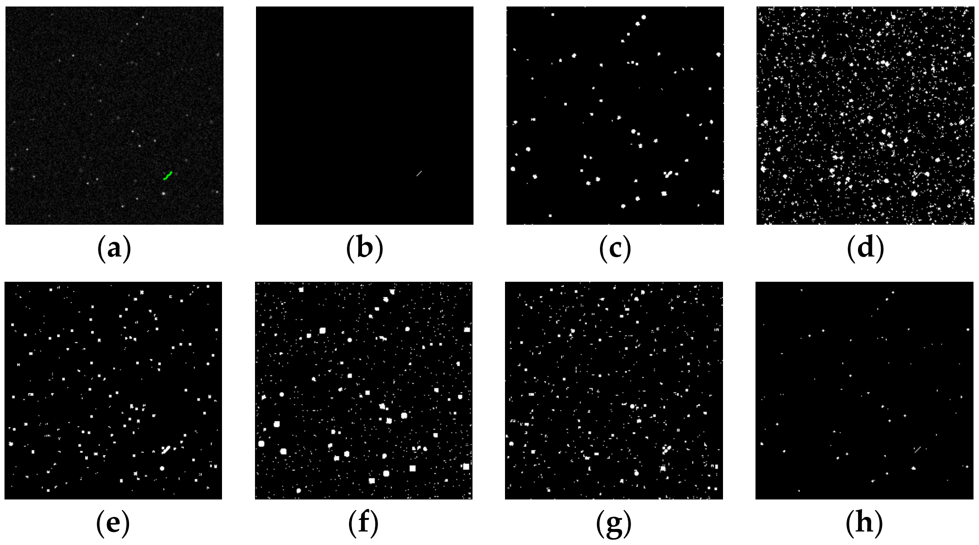

Figure 17. Detection results of target SNR = 1: (**a**) ground truth; (**b**) ours; (**c**) Max–Mean; (**d**) Top-Hat; (**e**) LCM; (**f**) MPCM; (**g**) STLDM; and (**h**) IPI.

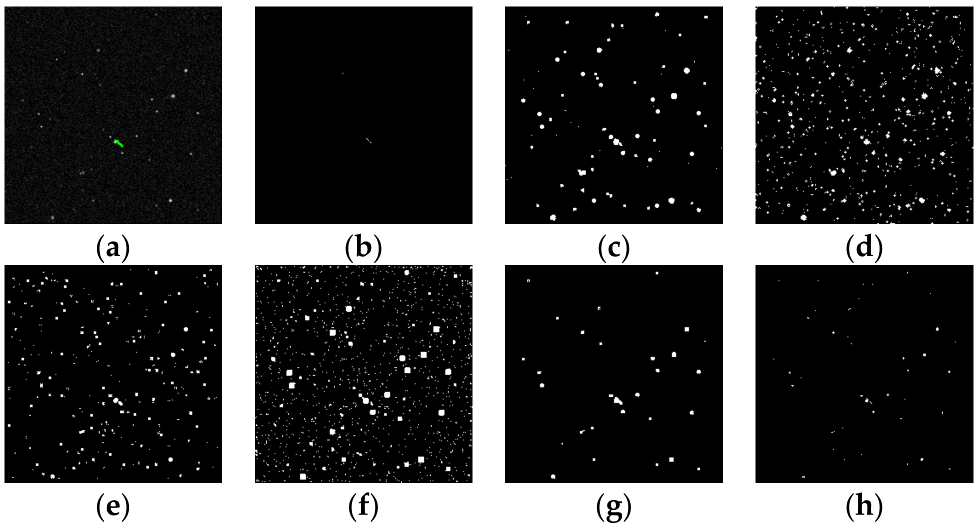

Figure 18. Detection results of target SNR = 0.7: (**a**) ground truth; (**b**) ours; (**c**) Max–Mean; (**d**) Top-Hat; (**e**) LCM; (**f**) MPCM; (**g**) STLDM; and (**h**) IPI.

The results of the ROC curves show that, although the detection rate decreases as the SNR of the target decreases, our approach can always achieve the best AUC performance: 0.9972, 0.9857, 0.9129, and 0.8628 for SNR = 3, SNR = 1.5, SNR = 1, and SNR = 0.7, respectively.

The results in Table 4 show that the proposed method achieved good detection rates for both sets of test data at SNR = 3 and SNR = 1.5, with average detection rates of 99.75% and 96%, respectively. When SNR ≤ 1, the target is extremely dim, at which point, the average detection rate of our algorithm decreases relatively more, but it is still 83% at SNR = 1, which is an acceptable result for a neural network approach. At the same time, the proposed method still outperformed the IPI method in terms of the detection rate and magnitude of its decline at SNR ≤ 1.

Table 4 also shows that the proposed method has the lowest false alarm rate for all SNRs when achieving the same detection rate and can be 1–3 orders of magnitude lower than the false alarm rate of the comparison method. An example of the feature map output by the third max-pooling layer of the network (final feature map) is shown in Figure 19. After processing by the network, the noise, stars, and stray light in the original image are filtered out, and only the moving target responds in the first channel of the feature map, which is the reason for the low false alarm rate and proves the effectiveness of the network for extracting the motion features of the target. However, this feature map contains coarse-grained semantic information that does not provide accurate target location information. Therefore, fine-grained information from the shallow feature map is also required to assist in locating the target. An example of the feature map output of the first SC Block is shown in Figure 20. The response results for each frame are derived from the convolution operation of three adjacent frames in the original image sequence. It can be seen that the responses to the moving target appear in the first channel in frames 2 to 7, that only the responses to the target are present in these frames, and the positions of these responses are more accurate than the positions of the responses in the final feature map. The response to the star appears on the eighth frame of the feature map, but as mentioned earlier, this response is filtered out in the subsequent network. Thus, the branch of our network used frames 2 to 7 of the feature map to localize targets in the response frames of the original image sequence, whereas frames 2 and 7 of the feature map were also used to localize targets in the first and eighth frames of the original image sequence, respectively. A quantitative analysis of the localization performance of the branch is presented in Section 4.2.2.

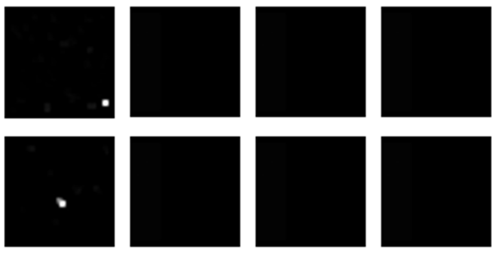
Figure 19. Each row is an example of the final feature maps when SNR = 1 and SNR = 0.7, respectively. Each of them has a size of 4 × 1 × 32 × 32 (4 channels with 1 frame of size 32 × 32 for each channel).

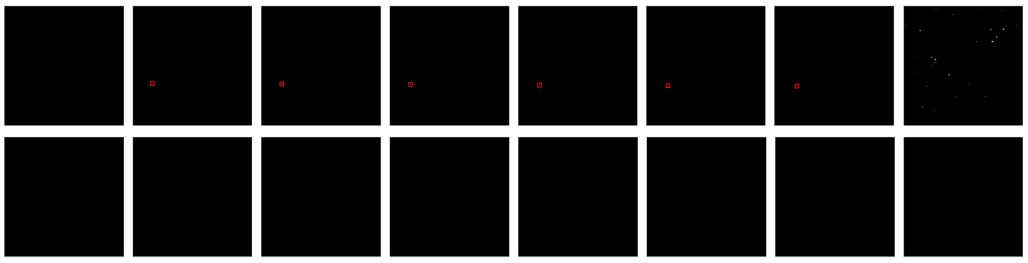
Figure 20. Example of the feature maps output by the 1st SC Block when SNR = 3, with a size of 2 × 8 × 256 × 256 (2 channels with 8 frames of size 256 × 256). The red boxes indicates the feature response of the target.

Detection Performance of Different Target Numbers

Experiments were conducted on the test data with different target numbers in each image sequence, and the parameter settings for the experiments are listed in Table 5.

Table 5. Parameter settings for the experiments of detection performance of different target numbers.

Target				Background	
SNR	**Number**	**Size**	**Speed**	**Stray Light**	**Slow Motion**
3	3, 2	3, 1	1	No	No

Figure 21 shows an example of the detection results of the proposed method on the first frame of two image sequences containing two and three targets. Figure 22 shows an example of a target motion trajectory detected by the proposed method in two image sequences containing two and three targets. Table 6 shows the detection performance results for different target numbers, which are the average results of 2000 image sequences from the test set. The results show that the proposed method achieved good detection performance on test data containing both two and three targets, and it is the same as on the SNR = 3 test data with one target; therefore, the detection performance of the proposed method is independent of the number of targets. Figure 23 shows an example of a feature map of the output of the first SC Block when the number of targets was three. It can be observed that three maximum responses for the three targets were generated in frames 2 to 7 of the first channel.

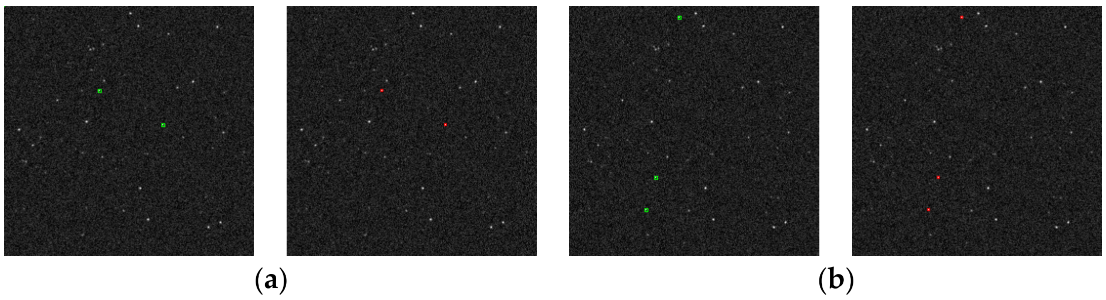

Figure 21. Example of the detection results on the first frame of two image sequences. (**a**) contains two targets and (**b**) contains three targets. In each pair, the left is the ground truth of the target, and the right is the detection result.

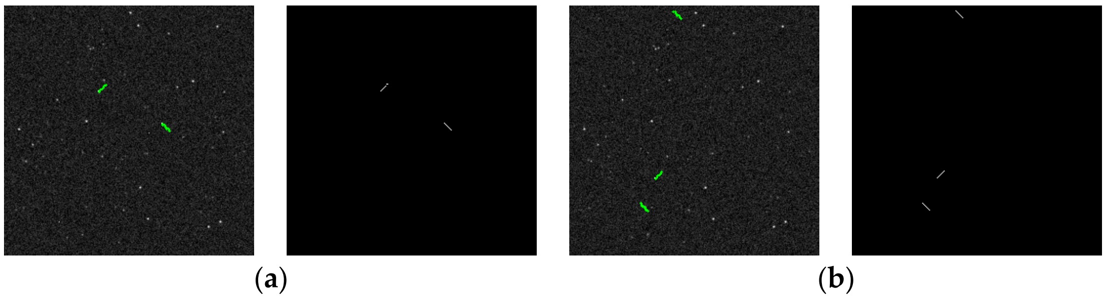

Figure 22. Example of a target trajectory output by the proposed method for two image sequences. (**a**) contains two targets and (**b**) contains three targets. In each pair, the left is the ground truth of the target, and the right is the detection result.

Table 6. Detection performance results of different target numbers.

Target Number	Detection Rate (%)	False Alarm Rate (%)
2	99.75	0.0212
3	99.625	0.0257

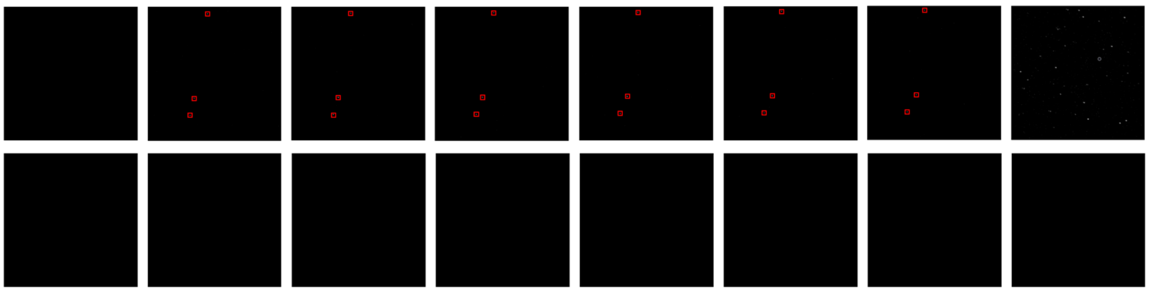

Figure 23. Example of the feature maps output by the 1st SC Block when target number = 3, with a size of 2 × 8 × 256 × 256 (2 channels with 8 frames of size 256 × 256). The red boxes indicates the feature response of the target.

Detection Performance of Different Target Speed

We conducted experiments on the test data with different target speeds in each image sequence, and the data contain different targets with different speeds in the same sequence. The parameter settings for the experiments are listed in Table 7.

Table 7. Parameter settings for the experiments of detection performance of different target speed.

Target				Background	
SNR	**Number**	**Size**	**Speed**	**Stray Light**	**Slow Motion**
3	1	5, 3, 1	12, 9, 6, 3	No	No
3	3, 2	5, 3, 1	6, 3, 1	No	No

Figure 24 shows an example of a target trajectory detected using the method presented for different target speeds. Table 8 shows the detection performance results for different target speeds, which are the average results of 2000 image sequences from the test set. It can be seen that at a target speed of three, when the target is still relatively slow, the proposed method achieves a good detection performance with a detection rate of 99.5% and a false alarm rate of only 0.0122%. As the speed of the target increased, the detection performance of the proposed method decreased slightly, and there were a few mislocated points in the output trajectory results. However, at a target speed of 12, there was still a detection rate of 89.625% and a false alarm rate of 0.0443%.

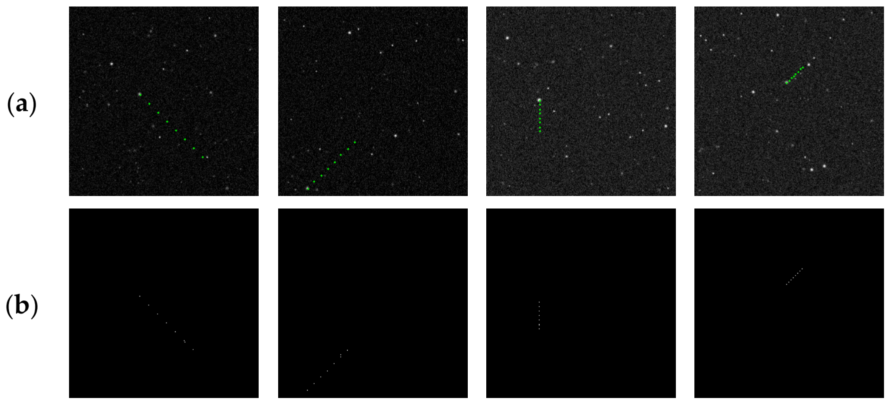

Figure 24. Example of target trajectories of different speeds. From left to right: speed = 12, speed = 9, speed = 6, and speed = 3. The first row (**a**): ground truths; the second row (**b**): results.

Table 8. Detection performance results of different target speeds.

Target Speed	**Detection Rate (%)**	**False Alarm Rate (%)**
12	89.625	0.0443
9	91.75	0.0372
6	93	0.0267
3	99.5	0.0122

Figure 25 and Table 9 show the experimental results for different targets with different speeds in the same sequence. It can be seen that the proposed method can still effectively detect the target in this type of data, achieving a detection rate of over 90% and a false alarm rate at the same level as the previous experiments. However, the detection rate decreases significantly compared with the results of the multi-target experiments (Table 6) at speed = 1. This again indicates that fast-moving targets can negatively affect the detection performance of the algorithm, and future improvements are needed to address this issue.

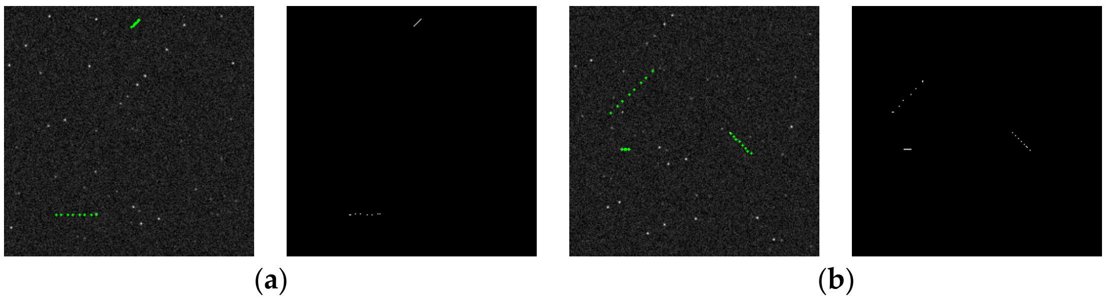

Figure 25. Example of a target trajectory output by the proposed method for two image sequences containing two and three targets with different speeds in the same sequence: (**a**) speed = 6 and 1; (**b**) speed = 6, 3, and 1. In each pair, the left is the ground truth of the target, and the right is the detection result.

Table 9. Detection performance results of different targets with different speeds in the same sequence.

Target Number	Target Speed	Detection Rate (%)	False Alarm Rate (%)
3	6, 3, 1	90.125	0.0350
2	6, 1	93.5	0.0277

Robustness to Nonuniform and Slowly Moving Backgrounds

We conducted experiments on test data with nonuniform or slowly moving backgrounds, and the parameter settings for the experiments are listed in Table 10.

Table 10. Parameter settings for the experiments of robustness to nonuniform and slowly moving backgrounds.

Target				Background	
SNR	Number	Size	Speed	Stray Light	Slow Motion
3	1	3, 1	1	Yes	No
3	1	3, 1	3	No	Yes

Tables 11 and 12 show the comparative experimental results of the detection performance of data with background stationary light and slow motion, respectively, which are the average results of 2000 image sequences from the test set. The detection rate of each comparison method was made as close as possible to our detection rate by adjusting the threshold; Figures 26 and 27 show the corresponding trajectory outputs.

Table 11. Detection performance results of image sequences with background stray light.

Metrics	Ours	Max–Mean	Top-Hat	LCM	MPCM	STLDM	IPI
Detection Rate (%)	98.25	98.75	98.125	98.25	98	98	97
False Alarm Rate (%)	**0.0241**	34.5864	40.4796	1.0717	10.3268	1.4859	0.4318

Bold numbers represent the best data in the row.

Table 12. Detection performance results of image sequences with slowly moving backgrounds.

Metrics	Ours	Max–Mean	Top-Hat	LCM	MPCM	STLDM	IPI
Detection Rate (%)	99.25	98.75	95.125	90	95.75	91.25	91.25
False Alarm Rate (%)	**0.0336**	1.2689	58.0269	5.3546	13.624	1.8536	0.3933

Bold numbers represent the best data in the row.

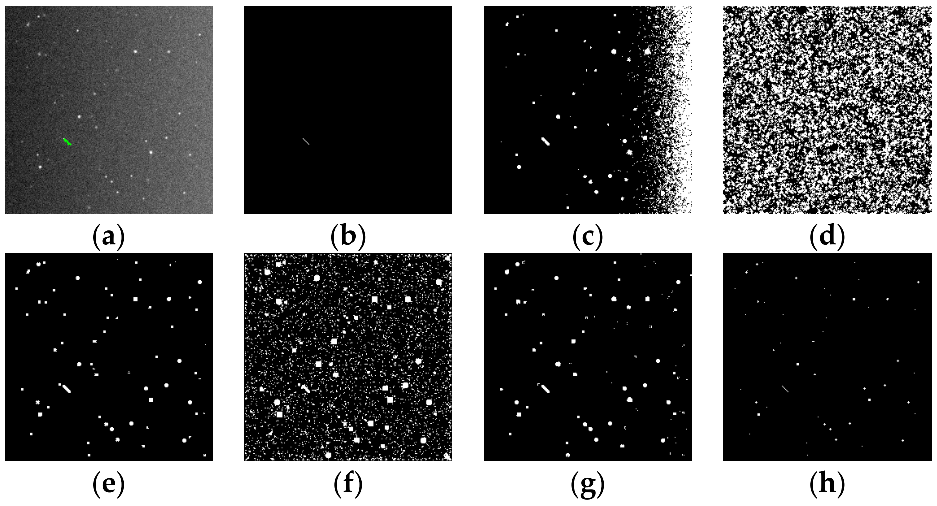

Figure 26. Detection results of image sequences with stray light: (**a**) ground truth; (**b**) ours; (**c**) Max–Mean; (**d**) Top-Hat; (**e**) LCM; (**f**) MPCM; (**g**) STLDM; and (**h**) IPI.

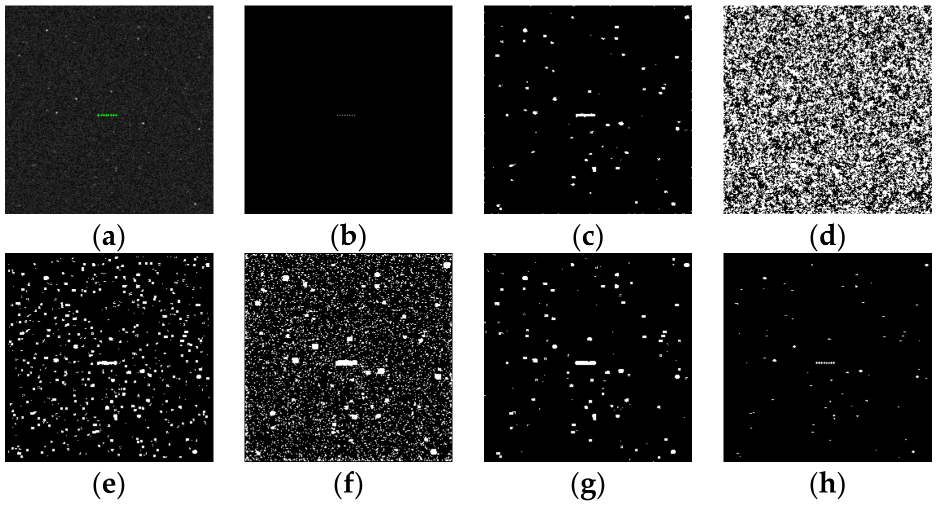

Figure 27. Detection results of image sequences with slowly moving backgrounds: (**a**) ground truth; (**b**) ours; (**c**) Max–Mean; (**d**) Top-Hat; (**e**) LCM; (**f**) MPCM; (**g**) STLDM; and (**h**) IPI.

Table 11 and Figure 26 show that the proposed method achieves the best detection performance even when the images contain stray light, which has only a small effect on the detection rate of the method compared with the results of tests without stray light under equivalent SNR conditions. However, other methods are significantly affected by stray light, particularly in the form of a significant increase in false alarm rates.

As can be seen from Table 12 and Figure 27, the proposed method also achieved the best results for the slow background movement data, with the slow movement of the background having no effect on the detection rate of the proposed method and only a very small effect on the false alarm rate. In contrast, the detection rates of LCM, STLDM, and IPI drop by approximately 8% compared with the absence of background motion, the false alarm rate increases significantly, and the movement of the background stars can be seen in the output plots of almost all the compared methods.

4.2.2. Ablation Experiments

In this section, two ablation experiments are conducted to verify the effectiveness of the soft thresholding module in the SC Block and the branch of the target detection network.

The Effectiveness of the Soft Thresholding Module

Two experiments were conducted to test the effect of the soft threshold module on noise and stray light. The parameter settings for the experiments are listed in Table 13.

Table 13. Parameter settings for the experiments of the effectiveness of the soft thresholding module.

Target				Background	
SNR	**Number**	**Size**	**Speed**	**Stray Light**	**Slow Motion**
3, 1.5, 1, 0.7	1	3, 1	1	No	No
3	1	3, 1	1	Yes	No

As can be seen from Table 14, the soft threshold module can improve the detection rate of the network to some extent, and the lower the SNR, the greater the improvement in the detection rate, but it is of little help in the suppression of false alarms. This conclusion is also reflected in the experimental results for the stray light data in Table 15. It appears that the soft threshold module is effective in suppressing noise interference and thus improving detection rates, but the network suppresses false alarms mainly by recognizing the different motion characteristics of the target and other elements (stars, noise, and stray light) in the image sequence.

Table 14. Results of the effect of the soft threshold module for different SNRs.

SNRs	Metrics	With Soft Thresholding Module	Without Soft Thresholding Module
3	Detection Rate (%)	**99.75**	99.625
	False Alarm Rate (%)	**0.0243**	0.0244
1.5	Detection Rate (%)	**96**	95.25
	False Alarm Rate (%)	**0.0206**	0.0209
1	Detection Rate (%)	**83**	81.5
	False Alarm Rate (%)	**0.0260**	0.0262
0.7	Detection Rate (%)	**61.5**	59.875
	False Alarm Rate (%)	**0.0662**	0.0674

Bold numbers represent the best data in the row.

Table 15. Results of the effect of the soft threshold module for stray light.

Stray Light	Metrics	With Soft Thresholding Module	Without Soft Thresholding Module
Yes	Detection Rate (%)	**98.25**	98
	False Alarm Rate (%)	**0.0241**	0.0248

Bold numbers represent the best data in the row.

The Effectiveness of the Branch of the Detection Network

We conducted an experiment to verify the effectiveness of the branch of the network in optimizing the position of the original anchor output of the network. This was tested using four sets of data with different SNRs. The parameter settings for the experiment are listed in Table 16.

Table 16. Parameter settings for the experiment of the effectiveness of the branch of the detection network.

Target				Background	
SNR	**Number**	**Size**	**Speed**	**Stray Light**	**Slow Motion**
3, 1.5, 1, 0.7	1	3, 1	1	No	No

Table 17 shows the average positioning error results for 2000 image sequences with different SNRs, from which it can be observed that the branch can significantly improve the positioning accuracy of the original anchors. The original anchors are fixed to the image, so the positioning error is relatively large, around 5 pixels for all SNRs, while the average positioning error of the branch-offset anchors is around 1.7 pixels for all SNRs. Figure 28 shows two examples of the effects of branch offsetting on the original anchors. Branch-offset anchors were used as an interim result of the overall detection framework.

Table 17. Results of the effect of the branch for different SNRs.

SNRs	Metric	With Branch	Without Branch
3	Positioning Error	**1.718**	5.326
1.5		**1.729**	5.532
1		**1.735**	5.466
0.7		**1.799**	5.890

Bold numbers represent the best data in the row.

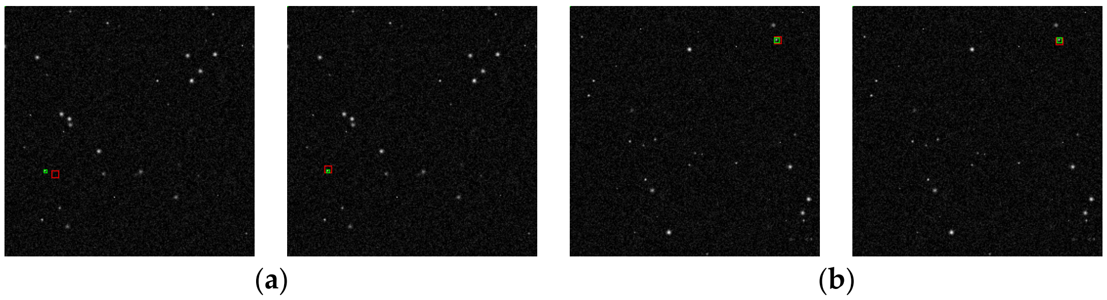

Figure 28. Two examples of the effect of branch offsetting the original anchors (pair (**a**,**b**)). The green boxes are the ground truth, and the red boxes are anchors. In each pair, the left is the original anchor and the right is the result after the branch offset.

4.2.3. Experiments of Target Positioning Performance

This section presents the results of the performance test of the proposed k-means centroid positioning method in the overall detection framework. The performance of the algorithm was evaluated by testing the average positioning error over 2000 image sequences and comparing it with the threshold grayscale centroid method and squared weighted centroid method. For this experiment, the parameter settings are presented in Table 18.

Table 18. Parameter settings of the experiments of target positioning performance.

Target				Background	
SNR	Number	Size	Speed	Stray Light	Slow Motion
3, 1.5	1	3, 1	1	No	No

As can be seen from Table 19, the positioning error of all three methods gradually increases as the SNR decreases, but the proposed method always maintains the lowest positioning error, and the increase in the positioning error is smaller than those of the other two methods, which can mainly be attributed to the excellent foreground/background segmentation capability of the k-means and the error reduction effect of the window iteration in the algorithm. At SNR = 1.5, the average positioning error of our method is 0.209 pixels, which allows for the high-accuracy positioning of the target centroid.

Figure 29 shows two examples of the positioning of targets with different SNRs (shown in the original window size); from left to right: the original target, the interpolated target, and the segmentation result of the k-means of the interpolated image. In the third column of the plot, the red and blue dots represent the pre- and post-iteration centroid positions, respectively. A high SNR target is shown in Figure 29a and it can be seen that the difference in distance between the red and blue dots is very small, meaning that the initial positioning results of the centroid positioning algorithm are accurate, and the iterations do not improve the accuracy significantly. In Figure 29b, for a lower SNR target (stronger background noise), it can be seen that the distance between the red and blue dots is very different, and the blue dot is a more accurate location for the centroid; therefore, for low-SNR targets, centroid iteration can effectively improve the localization accuracy.

Table 19. Results of the performance of our centroid positioning method for different SNRs compared with classic centroid positioning methods.

SNRs	Metric	Ours	Threshold Grayscale Method	Squared Weighted Method
3	Positioning Error	**0.146**	0.163	0.197
1.5		**0.209**	0.230	0.292

Bold numbers represent the best data in the row.

(**a**)

(**b**)

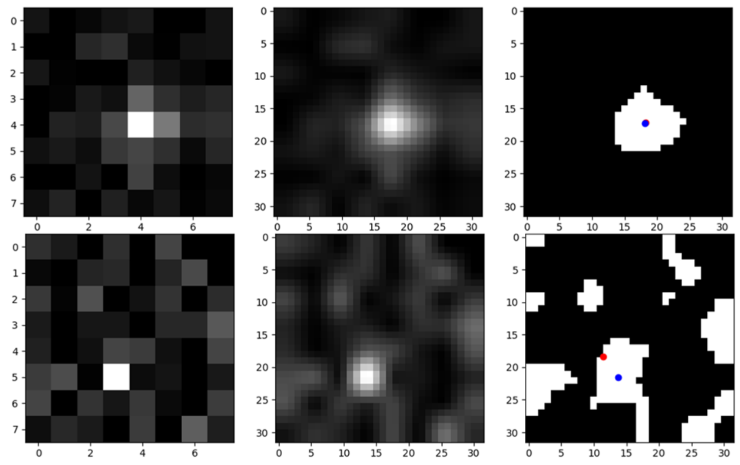

Figure 29. Examples of the validity of centroid iterations: (**a**) the 1st row: a target with higher SNR, and the effect of the precision enhancement of the iteration is not obvious; (**b**) the 2nd row: a target with lower SNR (stronger background noise), and the effect of precision enhancement of the iteration is obvious. The red dot and the blue dot represents the pre and post-iteration centroid positions, respectively.

The effect of target positioning after adding the k-means centroid positioning method to the network is shown in Figure 30 (third column of the figure). It can be seen that, after the k-means centroid positioning algorithm, the target centroid positioning effect is significantly improved, and not only is the centroid position more accurate but the bounding box also fits more precisely to the ground truth box.

4.3. Experiments on Semi-Real Data

This section describes the following two experiments: In the first experiment, a test set containing targets with SNR = 3 and speed = 3 was used and added to the real background image to simulate low-orbit bright targets. In the second experiment, a test set containing targets with SNR = 0.7 and speed = 1 was used to simulate a high-orbit dim target. The performance of the proposed method was tested under these two conditions. The parameter settings are shown in Table 20.

As can be seen from the results in Table 21, the average detection performance of the algorithm for the SNR = 3 test set is consistent with the results in Table 4, and the results for the test set with SNR = 0.7 are slightly better than the results in Table 4. Thus, the proposed method is suitable for generalizing the performance of semi-real data. Figure 31 shows the detection results for the first frame of the image sequence, and Figure 32 shows the trajectory output of the target.

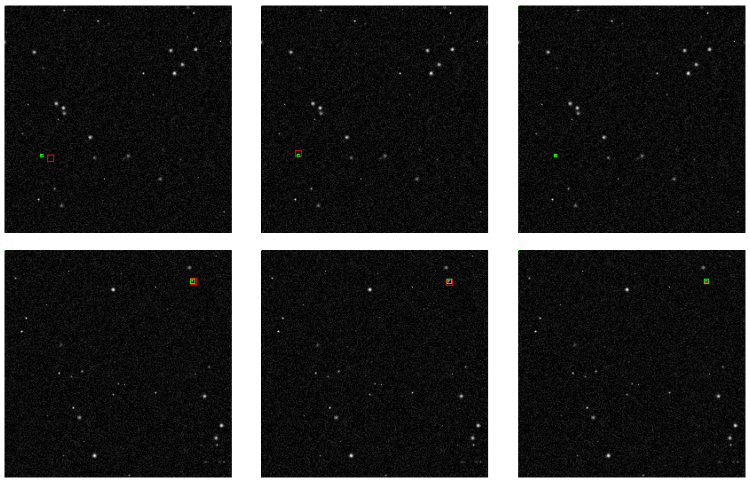
Figure 30. Examples of the effect of the overall positioning process in this study. The third column reflects the changes after processing via the k-means centroid positioning method. The green boxes are ground truth, and the red boxes are anchors.

Table 20. Parameter settings of the experiments on semi-real data.

Target			
SNR	**Number**	**Size**	**Speed**
3	1	3, 1	3
0.7	1	3, 1	1

Table 21. Detection performance of the two experiments.

SNRs	Target Speed	Detection Rate (%)	False Alarm Rate (%)
3	3	99.5%	0.0233
0.7	1	62.75%	0.0628

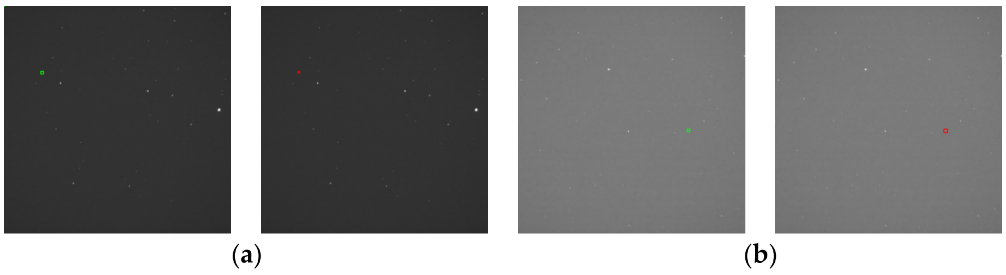
Figure 31. An example of the detection results for the first frame of two image sequences containing targets with (**a**) SNR = 3, speed = 3 and (**b**) SNR = 0.7, speed = 1. In each pair, the left is the ground truth of the target, and the right is the detection result.

4.4. Experiments on Real Data

This section includes three sets of experiments conducted on real-life captured data. The first set of experimental data was tested using a dataset containing asteroid 194, which was captured using a professional telescope at Lijiang Observatory on 24 April 2023 at 1721 UTC. This image sequence contains eight frames, each with a size of 3520 $\times$ 3520 and a field of view angle of approximately 1. The center of the field of view points to a right ascension of 12 h 30 m 09.23 s and a declination of +13°06′34.6″. The exposure time of each frame is 120 s, and the interval between each frame is approximately 24 min.

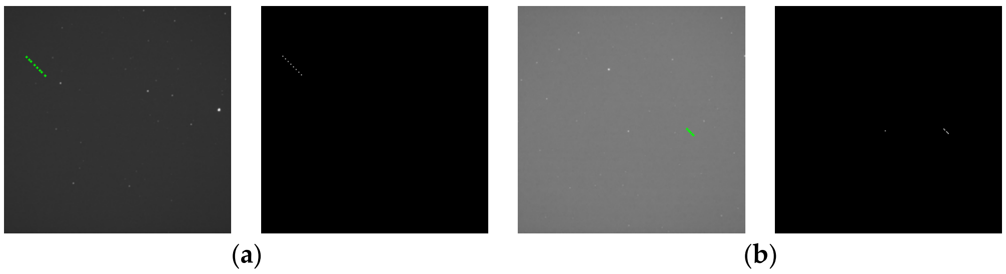

Figure 32. An example of a target trajectory output by the proposed method of two image sequences containing targets with (**a**) SNR = 3, speed = 3 and (**b**) SNR = 0.7, speed = 1. In each pair, the left is the ground truth of the target, and the right is the detection result.

During the experiment, we changed the size of each frame to 256 × 256, input our algorithm for detection, and restored the detection results back to the original image. Figure 33 shows the trajectory results of the asteroid detected using our method in the image sequence and displays the trajectory in the first frame of the image sequence. It can be seen that, because the asteroid is very far from Earth, its motion trajectory in the image is very short, and our algorithm detected this trajectory very well.

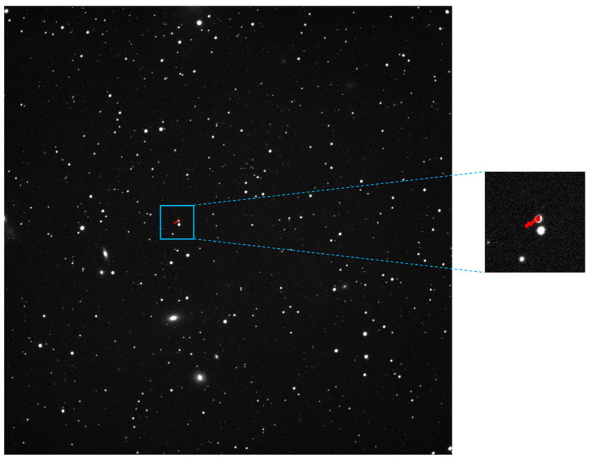

Figure 33. The target detection results of asteroid 194 via the proposed method; the trajectory area has been enlarged for display, with the trajectory marked by a red dotted line.

The second group of experiments was filmed using the same equipment as the first group, targeting asteroid 145. The data were captured on 27 April 2023 at 1806 UTC, with a right ascension of 13 h 41 m 27.71 s and a declination of +06°16′34.1″ at the center of the field of view. The exposure interval between frames is approximately 15 min, and the other settings are the same as in the first group of experiments.

Figure 34 shows the detection results, and it can be seen that our method detected some of the correct motion trajectories of the asteroid (marked with blue boxes), but there are still three incorrectly located trajectory points (marked with yellow boxes). These false detection points are mainly located on stars near the target. This is because the background of this data has significant jumps caused by unstable telescope pointing, resulting in the significant displacement of the background stars in the last three frames of the image sequence, resulting in algorithm false detection. In the future, we should pay attention to improving the robustness of algorithms with respect to jumping backgrounds.

The third group of experiments was carried out on a sequence of real images taken by a professional observatory provided by the "Tianzhi Cup" competition. The sequence consists of eight images, each measuring 4096 × 4096 (taken at 1520 UTC on 21 August 2020) with a large field of view centered at 291.38° right ascension and −13.94° declination, each with an exposure time of 200 ms and an interval of 4 s. The first image of the real data is shown in Figure 35.

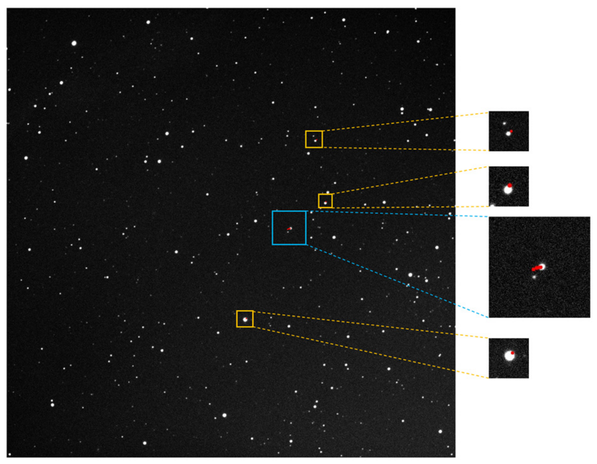

Figure 34. The target detection results of asteroid 145 via the proposed method; the trajectory area has been enlarged for display, with the correct trajectory marked in blue boxes and false detection points marked in yellow boxes with a red dotted line.

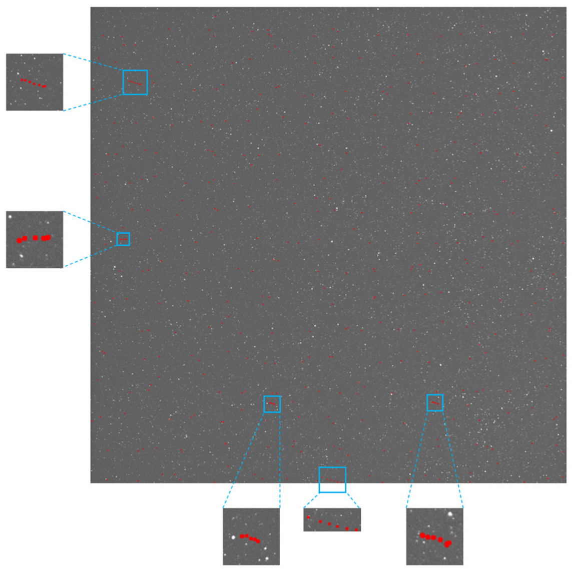

Figure 35. The target detection results via the proposed method; the trajectory area has been enlarged for display, with the trajectory marked in blue boxes with a red dotted line.

Because of the large size of the images, each original image of this sequence was first cropped into 16 × 16 pieces of small images of size 256 × 256 to match the input requirements of the detection network. Then, 256 new individual image sequences were generated by using small images at the same position as each original image. Each new image sequence was then detected using the detection framework. Finally, all detection results were combined and restored back to the original image.

Figure 35 also shows the detection results, and it can be observed that the proposed method clearly detects the five space-targets contained in the graph. In addition, there are sparse false alarm points in the result (456 in total), which is acceptable for an image of size 4096 × 4096, with a false alarm rate of approximately 0.002%. This is better than the average false alarm rate measured from the simulated dataset, indicating that our method has good generalization properties. If we had sufficient real data, we could build a real dataset and fine-tune our detection network to obtain a better detection performance for the real data.

5. Discussion

The experimental results show that the overall target detection framework can achieve high detection rates and low false alarm rates for conditions (such as SNR = 3 and SNR = 1.5) and detect targets of varying numbers and speeds, and it is robust regarding complex backgrounds. However, the detection rate of the overall detection framework at SNR ≤ 1 was not satisfactory. Although this is related to the inherent shortcomings of CNNs for dim–small target detection, it may still be possible to improve the detection rate under extremely low SNR conditions by optimizing the network structure. Concurrently, the overall detection framework is not a fully end-to-end model but still contains a separate centroid positioning step. Therefore, it will be a direction of future research to design a fully end-to-end model with better detection capabilities concerning extremely low-SNR targets and accurate centroid positioning.

In addition, our method in this article was mainly trained on simulation datasets containing point-shaped targets, which leads to two limitations: firstly, the algorithm's detection performance for other forms of moving targets (such as moving short streak-like targets generated under longer exposure times) may not be satisfactory, and secondly, the algorithm's detection performance may also decrease for other real data with attributes that have not been simulated before. At the same time, the scenarios we are currently simulating are also relatively limited, and more scenarios should be considered, such as deep space scenes or backgrounds where some parts of the Earth enter the field of view. Therefore, in subsequent work, we will make every effort to obtain more real data, annotate them, and train our algorithm to improve its ability to detect real data. We will increase the shape type of the target (such as short streak-like targets) and the real scene types (such as observing deep space targets from LEO, or observing low-Earth-orbit targets or deep space targets from Earth–Moon space) so that the algorithm can learn additional knowledge and better adapt to target detection tasks in the real world.

6. Conclusions

This study constructed a simulated dataset of dim and small space-targets and proposed a space-target detection framework that includes a space-target detection network and a k-means centroid positioning method. The experiments demonstrated that the proposed method has superior detection performance for targets with different SNRs, in particular achieving a very low false alarm rate while being robust with respect to the number of targets, their speed, and complex backgrounds containing stray light or slow movement. Moreover, the proposed method exhibited good detection performance and generalization for both semi-real and real star map data.

Author Contributions: Conceptualization, S.S.; methodology, S.S.; software, S.S. and C.R.; validation, S.S.; formal analysis, S.S.; investigation, S.S.; resources, W.N. and X.P.; data curation, S.S.; writing—original draft preparation, S.S.; writing—review and editing, S.S., W.N. and Y.L.; visualization, S.S.; supervision, W.N., X.P., W.Z. and Z.Y.; project administration, W.N., X.P., W.Z. and Z.Y.; funding acquisition, W.N., X.P., W.Z. and Z.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This work was partly supported by the Youth Innovation Promotion Association under Grant E1213A02 and the Key Research Program of Frontier Sciences, CAS (Grant No. 22E0223301).

Data Availability Statement: Not applicable.

Acknowledgments: Thanks to Lu Mingyu from Huairou Youth Palace in Beijing for his valuable assistance in shooting real data.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Wirnsberger, H.; Baur, O.; Kirchner, G. Space Debris Orbit Prediction Errors Using Bi-Static Laser Observations. Case Study: ENVISAT. *Adv. Space Res.* **2015**, *55*, 2607–2615. [CrossRef]
2. Esmiller, B.; Jacquelard, C.; Eckel, H.-A.; Wnuk, E. Space Debris Removal by Ground Based Laser Main Conclusions of the European Project CLEANSPACE. *Appl. Opt.* **2014**, *53*, I45–I54. [CrossRef] [PubMed]
3. Pelton, J. *Space Debris and Other Threats from Outer Space*; Springer: New York, NY, USA, 2013; ISBN 978-1-4614-6713-7.
4. Girshick, R.; Donahue, J.; Darrell, T.; Malik, J. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. In Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition, Columbus, OH, USA, 23–28 June 2014; pp. 580–587.
5. Girshick, R. Fast R-CNN. In Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV), Santiago, Chile, 7–13 December 2015; pp. 1440–1448.
6. Ren, S.; He, K.; Girshick, R.; Sun, J. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Trans. Pattern Anal. Mach. Intell.* **2017**, *39*, 1137–1149. [CrossRef]
7. Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You Only Look Once: Unified, Real-Time Object Detection. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 779–788.
8. Lin, T.-Y.; Goyal, P.; Girshick, R.; He, K.; Dollar, P. Focal Loss for Dense Object Detection. In Proceedings of the 2017 IEEE International Conference on Computer Vision (ICCV), Venice, Italy, 22–29 October 2017; pp. 2999–3007.
9. Aloise, D.; Deshpande, A.; Hansen, P.; Popat, P. NP-Hardness of Euclidean Sum-of-Squares Clustering. *Mach. Learn.* **2009**, *75*, 245–248. [CrossRef]
10. Forsyth, D.A.; Ponce, J. *Computer Vision: A Modern Approach*, 2nd ed.; Pearson: London, UK, 2017; ISBN 978-93-325-5011-7.
11. Hyun Cho, J.; Mall, U.; Bala, K.; Hariharan, B. PiCIE: Unsupervised Semantic Segmentation Using Invariance and Equivariance in Clustering. In Proceedings of the 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Nashville, TN, USA, 20–25 June 2021; pp. 16789–16799.
12. Do, H.N.; Chin, T.-J.; Moretti, N.; Jah, M.K.; Tetlow, M. Robust Foreground Segmentation and Image Registration for Optical Detection of GEO Objects. *Adv. Space Res.* **2019**, *64*, 733–746. [CrossRef]
13. Yanagisawa, T.; Kurosaki, H.; Banno, H.; Kitazawa, Y.; Uetsuhara, M.; Hanada, T. Comparison between Four Detection Algorithms For GEO Objects. In Proceedings of the 13th Annual Advanced Maui Optical and Space Surveillance Technologies Conference, Maui, HI, USA, 19–22 September 2012.
14. Sara, R.; Matoušek, M.; Franc, V. Ransacing Optical Image Sequences for GEO and Near-GEO Objects. In Proceedings of the Advanced Maui Optical and Space Surveillance Technologies Conference, Maui, HI, USA, 10–13 September 2013.
15. Piccardi, M. Background Subtraction Techniques: A Review. In Proceedings of the 2004 IEEE International Conference on Systems, Man and Cybernetics (IEEE Cat. No.04CH37583), The Hague, Netherlands, 10–13 October 2004; Volume 4, pp. 3099–3104.
16. Reed, I.; Gagliardi, R.; Shao, H. Application of Three-Dimensional Filtering to Moving Target Detection. *IEEE Trans. Aerosp. Electron. Syst.* **1983**, *AES-19*, 898–905. [CrossRef]
17. Mohanty, N.C. Computer Tracking of Moving Point Targets in Space. *IEEE Trans. Pattern Anal. Mach. Intell.* **1981**, *PAMI-3*, 606–611. [CrossRef]
18. Yanagisawa, T.; Nakajima, A.; Kimura, T.; Isobe, T.; Futami, H.; Suzuki, M. Detection of Small GEO Debris by Use of the Stacking Method. *J. Jpn. Soc. Aeronaut. Space Sci.* **2003**, *51*, 61–70. [CrossRef]
19. Liu, M.; Wang, H.; Yi, H.; Xue, Y.; Wen, D.; Wang, F.; Shen, Y.; Pan, Y. Space Debris Detection and Positioning Technology Based on Multiple Star Trackers. *Appl. Sci.* **2022**, *12*, 3593. [CrossRef]
20. Jiang, P.; Liu, C.; Kang, Z.; Yang, W.; Li, Z. Faint Space Debris Detection Algorithm Based on Small Aperture Telescope Detection System. *Res. Astron. Astrophys.* **2022**, *22*, 105003. [CrossRef]
21. Gao, J.; Wang, L.; Liu, W.; Cheng, B.; Jiang, S. A Space Debris Detection and Tracking Method for Wide-Field Surveillance. In Proceedings of the Ninth Symposium on Novel Photoelectronic Detection Technology and Applications, Hefei, Anhui, China, 4 April 2023; Volume 12617, p. 1261708.
22. Zhang, X.; Xiang, J.; Zhang, Y. Space Object Detection in Video Satellite Images Using Motion Information. *Int. J. Aerosp. Eng.* **2017**, *2017*, 1024529. [CrossRef]
23. Salmond, D.J.; Birch, H. A Particle Filter for Track-before-Detect. In Proceedings of the 2001 American Control Conference, Arlington, VA, USA, 25–27 June 2001; Volume 5, pp. 3755–3760.
24. Uetsuhara, M.; Ikoma, N. Faint Debris Detection by Particle Based Track-before-Detect Method. In Proceedings of the Advanced Maui Optical and Space Surveillance Technologies Conference, Wailea, HI, USA, 9–12 September 2014; Ryan, S., Ed.; The Maui Economic Development Board: Kihei, HI, USA, 2014; p. E54.
25. Rutten, M.G.; Gordon, N.J.; Maskell, S. Recursive Track-before-Detect with Target Amplitude Fluctuations. *IEE Proc. Radar Sonar Navig.* **2005**, *152*, 345–352. [CrossRef]
26. Blostein, S.; Huang, T. Detecting Small, Moving Objects in Image Sequences Using Sequential Hypothesis Testing. *IEEE Trans. Signal Process.* **1991**, *39*, 1611–1629. [CrossRef]
27. Demos, G.C.; Ribas, R.; Broida, T.J.; Blackman, S. Applications of MHT to Dim Moving Targets. In Proceedings of the Defense + Commercial Sensing, Signal and Data Processing of Small Targets, Orlando, FL, USA, 16–20 April 1990. [CrossRef]

28. Blostein, S.; Richardson, H. Richardson A Sequential Detection Approach to Target Tracking. *IEEE Trans. Aerosp. Electron. Syst.* **1994**, *30*, 197–212. [CrossRef]
29. Xi, J.; Wen, D.; Ersoy, O.K.; Yi, H.; Yao, D.; Song, Z.; Xi, S. Space Debris Detection in Optical Image Sequences. *Appl. Opt.* **2016**, *55*, 7929–7940. [CrossRef] [PubMed]
30. Li, M.; Yan, C.; Hu, C.; Liu, C.; Xu, L. Space Target Detection in Complicated Situations for Wide-Field Surveillance. *IEEE Access* **2019**, *7*, 123658–123670. [CrossRef]
31. Dawson, W.; Schneider, M.; Kamath, C. Blind Detection of Ultra-Faint Streaks with a Maximum Likelihood Method. In Proceedings of the Advanced Maui Optical and Space Surveillance Technologies Conference, Kihei, HI, USA, 19–22 September 2016.
32. Levesque, M.P.; Buteau, S. *Image Processing Technique for Automatic Detection of Satellite Streaks*; Defense Research and Development Canada Valcartier: Quebec, QC, Canada, 2007.
33. Vananti, A.; Schild, K.; Schildknecht, T. Streak Detection Algorithm for Space Debris Detection on Optical Images. In Proceedings of the AMOS Technical Conference, Maui, HI, USA, 15 September 2015.
34. Tagawa, M.; Yanagisawa, T.; Kurosaki, H.; Oda, H.; Hanada, T. Orbital Objects Detection Algorithm Using Faint Streaks. *Adv. Space Res.* **2016**, *57*, 929–937. [CrossRef]
35. Sara, R.; Cvrček, V. Faint Streak Detection with Certificate by Adaptive Multi-Level Bayesian Inference. In Proceedings of the European Conference on Space Debris, Darmstadt, Germany, 18–21 April 2017.
36. Takayuki, I.; Takumi, T. *Trajectory Detection Method Using Blind-RL for GEO Space Debris Detection, Proceedings of the SANE, 18 August 2022*; IEICE: Tokyo, Japan, 2022; Volume SANE2022-42, pp. 50–55.
37. Cowart, A.E.; Snyder, W.E.; Ruedger, W.H. The Detection of Unresolved Targets Using the Hough Transform. *Comput. Vis. Graph. Image Process.* **1983**, *21*, 222–238. [CrossRef]
38. Jiang, P.; Liu, C.; Yang, W.; Kang, Z.; Li, Z. Automatic Space Debris Extraction Channel Based on Large Field of View Photoelectric Detection System. *Publ. Astron. Soc. Pac.* **2022**, *134*, 024503. [CrossRef]
39. Zimmer, P.; Ackermann, M.; McGraw, J. GPU-Accelerated Faint Streak Detection for Uncued Surveillance of LEO. In Proceedings of the 2013 AMOS Technical Conference, Maui, HI, USA, 19–22 September 2013.
40. Vallduriola, G.V.; Trujillo, D.A.S.; Helfers, T.; Daens, D.; Utzmann, J.; Pittet, J.-N.; Lièvre, N. The Use of Streak Observations to Detect Space Debris. *Int. J. Remote Sens.* **2017**, *39*, 2066–2077. [CrossRef]
41. Abay, R.; Gupta, K. GEO-FPN: A Convolutional Neural Network for Detecting GEO and near-GEO Space Objects from Optical Images. In Proceedings of the 8th European Conference on Space Debris, Darmstadt, Germany, 13–20 April 2021.
42. Bertin, É.; Arnouts, S. SExtractor: Software for Source Extraction. *Astron. Astrophys. Suppl. Ser.* **1996**, *117*, 393–404. [CrossRef]
43. Xue, D.; Sun, J.; Hu, Y.; Zheng, Y.; Zhu, Y.; Zhang, Y. Dim Small Target Detection Based on Convolutinal Neural Network in Star Image. *Multimed. Tools Appl.* **2020**, *79*, 4681–4698. [CrossRef]
44. Jia, P.; Liu, Q.; Sun, Y. Detection and Classification of Astronomical Targets with Deep Neural Networks in Wide-Field Small Aperture Telescopes. *Astron. J.* **2020**, *159*, 212. [CrossRef]
45. Tao, J.; Cao, Y.; Zhuang, L.; Zhang, Z.; Ding, M. Deep Convolutional Neural Network Based Small Space Debris Saliency Detection. In Proceedings of the 2019 25th International Conference on Automation and Computing (ICAC), Lancaster, UK, 5–7 September 2019; pp. 1–6.
46. Tao, J.; Cao, Y.; Ding, M. SDebrisNet: A Spatial–Temporal Saliency Network for Space Debris Detection. *Appl. Sci.* **2023**, *13*, 4955. [CrossRef]
47. Varela, L.; Boucheron, L.; Malone, N.; Spurlock, N. Streak Detection in Wide Field of View Images Using Convolutional Neural Networks (CNNs). In Proceedings of the Advanced Maui Optical and Space Surveillance Technologies Conference, Maui, HI, USA, 19–22 September 2019; p. 89.
48. Duev, D.A.; Mahabal, A.; Ye, Q.; Tirumala, K.; Belicki, J.; Dekany, R.; Frederick, S.; Graham, M.J.; Laher, R.R.; Masci, F.J.; et al. DeepStreaks: Identifying Fast-Moving Objects in the Zwicky Transient Facility Data with Deep Learning. *Mon. Not. R. Astron. Soc.* **2019**, *486*, 4158–4165. [CrossRef]
49. Jahne, B. *Practical Handbook on Image Processing for Scientific and Technical Applications*, 2nd ed.; CRC Press, Inc.: Boca Raton, FL, USA, 2004; ISBN 0-8493-1900-5.
50. Wang, H.; Xu, E.; Li, Z.; Jingjin, L.; Qin, T. Gaussian Analytic Centroiding Method of Star Image of Star Tracker. *Adv. Space Res.* **2015**, *56*, 2196–2205. [CrossRef]
51. Yuan, X.; Li, G.; Tang, X.; Gao, X.; Huang, G.; Li, Y. Centroid Automatic Extraction of Spaceborne Laser Spot Image. *Acta Geod. Et Cartogr. Sin.* **2018**, *47*, 135–141.
52. Roman, N.G.; Warren, W.H., Jr. Smithsonian Astrophysical Observatory Star Catalog (SAO) (SAO Staff 1966). Documentation for the Machine-Readable Version. *Mach. Readable Version SAO Star Cat.* **1989**. Available online: https://heasarc.gsfc.nasa.gov/W3Browse/star-catalog/sao.html (accessed on 18 March 2023).
53. Xia, S.; Chen, J.; Lei, X.; Sang, J. Research on Image Simulation for Space-Based Space Debris Surveillance (In Chinese). *Chin. J. Space Sci.* **2020**, *40*, 1084–1090. [CrossRef]
54. Zhao, M.; Zhong, S.; Fu, X.; Tang, B.; Pecht, M. Deep Residual Shrinkage Networks for Fault Diagnosis. *IEEE Trans. Ind. Inf.* **2020**, *16*, 4681–4690. [CrossRef]

55. Kowalski, M.; Naruniec, J.; Trzcinski, T. Deep Alignment Network: A Convolutional Neural Network for Robust Face Alignment. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), Honolulu, HI, USA, 21–26 July 2017; pp. 2034–2043.
56. Merget, D.; Rock, M.; Rigoll, G. Robust Facial Landmark Detection via a Fully-Convolutional Local-Global Context Network. In Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 781–790.
57. Newell, A.; Yang, K.; Deng, J. Stacked Hourglass Networks for Human Pose Estimation. In Proceedings of the Computer Vision —ECCV 2016, Amsterdam, The Netherlands, 26 July 2016.
58. Kingma, D.P.; Ba, J. Adam: A Method for Stochastic Optimization. In Proceedings of the 3rd International Conference for Learning Representations, San Diego, CA, USA, 7–9 May 2015.
59. He, K.; Zhang, X.; Ren, S.; Sun, J. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. In Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV), Santiago, Chile, 7 December 2015.
60. Glorot, X.; Bengio, Y. Understanding the Difficulty of Training Deep Feedforward Neural Networks. *J. Mach. Learn. Res.* **2010**, *9*, 249–256.
61. Deshpande, S.D.; Er, M.H.; Venkateswarlu, R.; Chan, P. Max-Mean and Max-Median Filters for Detection of Small Targets. In Proceedings of the Signal and Data Processing of Small Targets 1999, Denver, CO, USA, 4 October 1999; Volume 3809, pp. 74–83.
62. Tom, V.T.; Peli, T.; Leung, M.; Bondaryk, J.E. Morphology-Based Algorithm for Point Target Detection in Infrared Backgrounds. In Proceedings of the Defense, Security, and Sensing, Orlando, FL, USA, 16 April 1993. [CrossRef]
63. Chen, C.L.P.; Li, H.; Wei, Y.; Xia, T.; Tang, Y.Y. A Local Contrast Method for Small Infrared Target Detection. *IEEE Trans. Geosci. Remote Sens.* **2014**, *52*, 574–581. [CrossRef]
64. Wei, Y.; You, X.; Li, H. Multiscale Patch-Based Contrast Measure for Small Infrared Target Detection. *Pattern Recognit.* **2016**, *58*, 216–226. [CrossRef]
65. Du, P.; Hamdulla, A. Infrared Moving Small-Target Detection Using Spatial–Temporal Local Difference Measure. *IEEE Geosci. Remote Sens. Lett.* **2020**, *17*, 1817–1821. [CrossRef]
66. Gao, C.; Meng, D.; Yang, Y.; Wang, Y.; Zhou, X.; Hauptmann, A.G. Infrared Patch-Image Model for Small Target Detection in a Single Image. *IEEE Trans. Image Process.* **2013**, *22*, 4996–5009. [CrossRef]