Noname manuscript No.
(will be inserted by the editor)

SPIN: Cleaning, Monitoring, and Querying Image Streams Generated by Ground-Based Telescopes for Space Situational Awareness

Keke Chen · Bharath Avusherla · Sarah Allison · Vincent Schmidt

the date of receipt and acceptance should be inserted later

Abstract With the increasing number of objects in Earth orbits, space situational awareness (SSA) becomes critical to space safety. As an economical option, ground-based telescopes can be deployed around the world and continuously provide imaginary information of space objects. However, they also raise unique challenges regarding big, noisy, and streaming data processing. In this paper, we present the SPIN system to address these challenges. The core algorithms process image sequences generated by ground-based telescopes and conduct: (1) image quality classification for data cleaning, (2) stream-based key-object identification and anomaly detection, and (3) efficient query processing on large image sequence repositories. Our goal is to design or adopt algorithms that handle the domain-specific image streams most efficiently and effectively. We use a 17-inch telescope to collect a large real dataset for evaluating the core algorithms, which covers more than ten satellites in one month and contains about 16,400 images. The experimental results show that the developed algorithms are fast enough for stream-based real-time processing and also yield high-quality results for all the primary tasks.

Keywords Space Situational Awareness · Data Stream · Image Processing · Machine Learning · Image Query Processing

Keke Chen, Bharath Avusherla, Sarah Allison
Data Intensive Analysis and Computing Lab
Department of Computer Science and Engineering
Wright State University, OH, USA
E-mail: {keke.chen, avusherla.2, allison.24}@wright.edu

Vincent Schmidt
Air Force Research Laboratory
Wright-Patterson Air Force Base, OH, USA
E-mail: vincent.schmidt@us.af.mil

1 Introduction

With the increasing number of satellites, space crafts, and debris around the earth, space situational awareness (SSA) becomes critical to national security and space safety. According to the recent data from CelesTrak (celestrak.com), there are over 13,000 satellites in orbit, and over 20,500 satellites have decayed since 1957. The near-earth orbits are more crowded than people can imagine (Figure 1). Among the satellites in orbit, there are just under 3,500 satellites that are functioning in their correct orbit, compared to nearly 10,000 that are classed as debris but have not yet decayed. With the increasing number of debris, the chance of collision increases, too. In 2009, the defunct Russian communications satellite crashed into an operational Iridium spacecraft, creating a new debris cloud comprising about 700 objects [1]. There is an urgent need to monitor space objects to predict and avoid collisions.

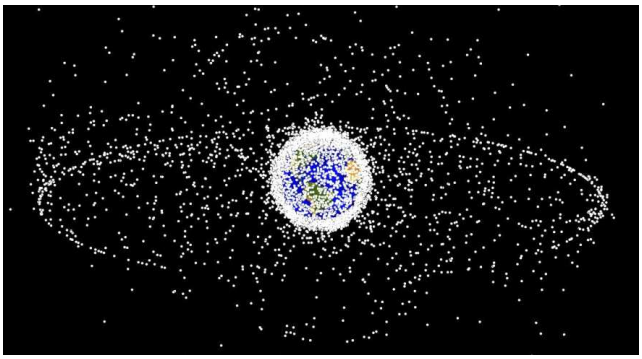

Fig. 1 The crowded near-earth orbits. Credit: NASA

[1] Satellites Crash in Space, Debris Scatters, http://www.tomsguide.com/us/Space-Satellite-Collision-Sibera,news-3477.html

The recent advances in remote sensing and optical devices make it possible to collect large amounts of space data using multiple methods. In this paper, we consider those data streams generated by ground-based telescopes (GBTs). Equipped with high-resolution cameras, they generate image streams, which can be processed for monitoring space objects in earth orbits. Compared to other more expensive instruments such as space-based telescopes and radars, GBTs are cheaper to obtain, install, and maintain[2]. They can be installed in multiple locations around the world and used to observe the same space object from different locations continuously and simultaneously. It is then possible to utilize images of the same space object taken from different times, or at the same time from different locations, to gain more reliable information to improve the quality of space situational awareness. Such a multi-source data fusion can be achieved economically with the deployment of low-cost GBT stations. However, there are several challenges to be addressed so that an automatic economical GBT-based solution become realistic.

Challenges. Space image data from ground-based telescopes have several unique features, which raise several challenges in effectively utilizing them.

- **Noisy Data.** Because of the changing meteorological conditions, the long distance to the target objects, and equipment setup, the quality of images from ground-based telescopes can vary from time to time. As a result, not all the raw images contain useful information, which needs to be sorted and cleaned before entering meaningful processing.
- **Large Data.** A GBT can generate enormous amounts of image data. A common image in FITS format (fits.gsfc.nasa.gov) captured by a 17-inch telescope has 1.4 Megabytes(MB), which is generated at a rate of one image every few seconds continuously for a period, e.g., 8 hours at night, easily producing gigabytes of data. The number of space objects of interest can be hundreds to thousands, and thus the aggregated historical data will lead to a large image repository. Furthermore, when multiple image sources from different GBT stations are integrated, the data will be even larger. Analytical queries on such a huge image repository should be efficiently and accurately processed.
- **Streaming Data.** In addition to offline analyses of large image datasets, online monitoring of specific objects is also important to detect anomalies and generate reaction plans promptly. Thus, images have to be produced at a relatively high rate, e.g., a few images per minute, to create an "image stream". This stream is processed and mined in real time to detect possible anomalies, including missing monitored objects or emerging new objects around the monitored ones.

[2] A high-end GBT costs about only about $50K (check planewave.com).

Scope of Research. To address these challenges, we develop the SPIN (SPace Image cleaning, moNitoring, and querying) system. This system will try to answer the following key questions. (1) How to identify bad images automatically? (2) How to process a stream of images, identify the key objects, and detect possible anomalies? And (3) how to query the large space image repository to find relevant images? Our goal is to develop or adopt the most *efficient* algorithms that can give *accurate* results for these major tasks. The focus is not to invent new primary image processing algorithms, but to find the best integration of the existing techniques to answer the key questions well.

A unique feature of the SPIN system is that both real-time monitoring and offline analytics are based on sequentially captured images sequences, not on individual images. The redundant information in a sequence of images can be effectively used to simplify the algorithm design. We will extract raw image and object features from each image as fast as possible. However, they do not provide accurate assertions until they are linked to the image sequences. The image repository is organized by image sequences and indexed by features extracted from the sequences, rather than from individual images. Correspondingly, query processing is also sequence-based: both the query and the result are image sequences. In stream-based monitoring, the key objects are identified and anomalies are detected from sequences (i.e., sliding windows in the image stream).

By carefully studying the characteristics of space object images, we design (or adopt) a set of simple image processing techniques that give fast processing speed and guarantee accurate results. Specifically, our approach has several unique contributions.

- We design a simple method for rapidly extracting (possibly noisy) objects from individual images. The high accuracy of key object detection and noise reduction is effectively achieved by processing the recurring objects in sequential images.
- We develop an effective online image-quality classifier for space images, based on a few basic image features (e.g., histogram features) and object features of an individual image.
- The geometry of multiple key objects is utilized in indexing and query processing, which leads to high speed and high accuracy query processing.
- We have conducted extensive experiments with a large real dataset (about 16,400 images covering more than ten satellites) collected by ourselves. The results show that we have satisfactorily achieved our design goals.

We describe more details of the SPIN approach in the following sections. Section 3 gives the system framework

and the roles of the core algorithms. Section 4 describes the core algorithms for this framework. Section 5 presents experimental results to show the efficiency and accuracy of the algorithms. Section 2 includes some related work.

2 Related Work

Several types of instruments can be used to observe space objects. Among them, space-based telescopes can generate the highest quality images without atmospheric turbulence, which is critical for observing deep space objects. The Hubble space telescope has been the most successful project in the history of astronomy. However, this type of resource is very scarce, often dedicated to scientific studies. It is very expensive to deploy, operate, and maintain space-based telescopes for space situational awareness.

The ground-based techniques include telescopes and radars. Ground-based telescopes are more economical and also highly available in the market. However, they may generate lower quality images due to atmospheric turbulence, can only be used in good weather conditions [1] for monitoring medium-orbit or geostationary satellites. Another well-known ground-based instrument is Synthetic Aperture Radar (SAR) [4], which is also more expensive than ground-based telescopes. Compared to a commodity 17-inch telescope such as CDK17[3], which often costs less than $50,000, an SAR costs millions. SARs have their unique advantages - they can detect objects as small as tens of centimeters and are not subject to atmospheric conditions. Thus, they are critical to monitoring low-orbit satellites. Note that the image stream techniques developed in our approach can also be used for SAR images.

Space situational awareness systems have two types of monitoring sensors[4]: surveillance and tracking sensors. The surveillance sensor sees a vast area of the sky at one time. It is not actively looking for objects, but rather passively waiting for objects to pass over. In contrast, tracking sensors usually have a very small field of view. It is particularly useful when the orbit of an object is already known, and the object needs to be precisely tracked. Our system is designed for tracking sensors.

Object detection in images has achieved many significant results during the past decade. For example, the Viola-Jones method [15] can quickly detect human faces in an image at a rate of 15 frames per second for small images (384x288). The recent advances in deep learning have been applied to object detection [13], resulting in detectors for multiple types of objects. Due to the rich content of images, image feature extraction is often discussed under a certain application [14]. However, there are also some general features widely used by many applications such as the pixel-block features [15] and scale-invariant features [8]. However, our focus is not to develop more advanced general object detection or feature extraction techniques, but to find the simplest (thus fastest) method that takes advantage of the characteristics of space surveillance images and works nicely.

Content-based image retrieval is a challenging problem. It is closely related to visual recognition [11] - a classical problem in computer vision. A promising way is to classify an image into multiple categories [3], where the deep learning techniques can be effectively applied [16]. Jing et al. [7] show that the link structure on the web can also be helpful in identifying the topics of an online image. These techniques typically require a large labeled training dataset and intensive computing resources, making it expensive to develop and apply models. The general problem of content-based image retrieval is difficult because the content can be anything in the world. However, space images have very restricted types of content, making it possible to use much simpler yet still highly effective retrieval techniques. Since we can accurately identify the key objects in image sequences with our proposed techniques, we can use the geometric information of the objects for query processing.

3 System Framework

To fully utilize the images generated by ground-based telescopes for the SSA purpose, we have several goals for this system. (1) Build up a repository of space object images, where the images should be of good quality and the primary objects are well identified. (2) Be able to monitor specific space objects to find possible anomalies such as the missing targets or emerging new objects around the targets. And (3) Query the image repository with a captured image sequence to find the similar ones.

The system is intended to collect, monitor and query specific space objects, whose trajectory models are already known. The trajectory model of a space object precisely defines the location of the object at any time. It thus makes possible for ground-based telescopes to automatically and continuously track the target object. The ultimate goal is to develop a maintenance-free, remotely controllable system for ground-based telescope stations.

Figure 2 shows the workflow and the major components in the system. Specifically, multiple ground-based telescopes are set up in different locations to monitor specific objects with known trajectory models. The captured images are coming in streams, cleaned and processed at computational nodes to check whether the observed situations are normal or abnormal. Selected good-quality images may enter the image repository for future analysis, where more advanced features might be extracted and analytic algorithms be applied. This

[3] http://planewave.com/products-page/telescopes/

[4] http://goo.gl/oNlRjS

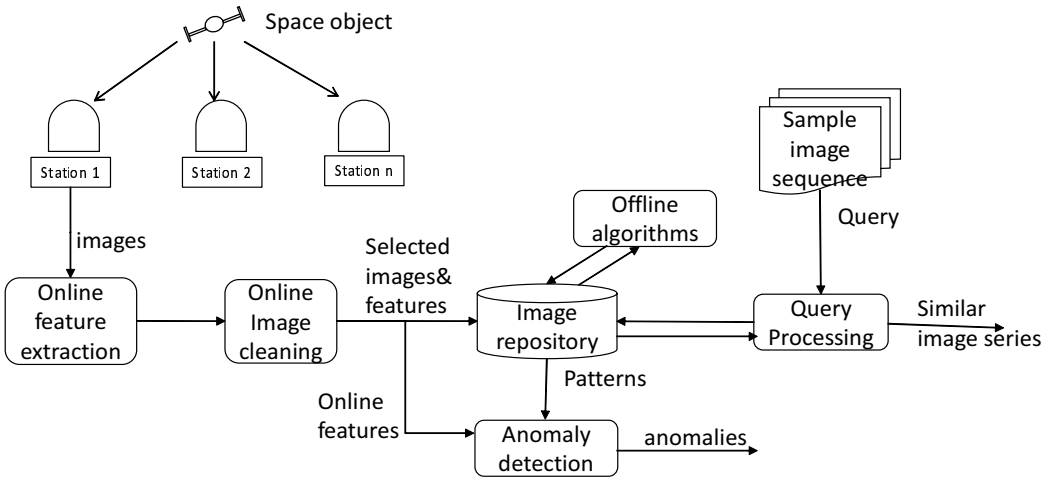

Fig. 2 SPIN System Framework

image repository might be queried with image sequences to find the most relevant sequences in the database.

We identify several core algorithms that enable this system framework. (1) Efficient online feature extraction algorithms that can extract low-cost features for quickly and accurately conducting automated tasks such as image quality determination and key-object identification. (2) A sequence-based key object identification algorithm for reliably extracting key objects from the noisy objects in a series of sequentially captured images. (3) Algorithms for finding object geometric patterns for indexing and querying image sequences. These core algorithms are mapped to the key components of the framework. Note that there are unique problems on correlating images from multiple locations, which, however, is not covered by our current work. In the following, we will discuss the core algorithms in detail.

4 Core Algorithms

In this section, we describe the design of three key algorithms: object-based online feature extraction, stream-based key object identification, and geometry-based indexing and query processing.

4.1 Object-Based Online Feature Extraction

The first key operation of the framework is to extract features from images, which will be used in the subsequent tasks. There are two special requirements. (1) The feature extraction process needs to be efficient enough for online processing of image streams, such as image quality classification and anomaly detection. (2) It should capture all the major objects so that the features can be re-used by subsequent tasks. The algorithm may generate noisy object extraction, but we need it to include all key objects as possible (i.e., very high recall).

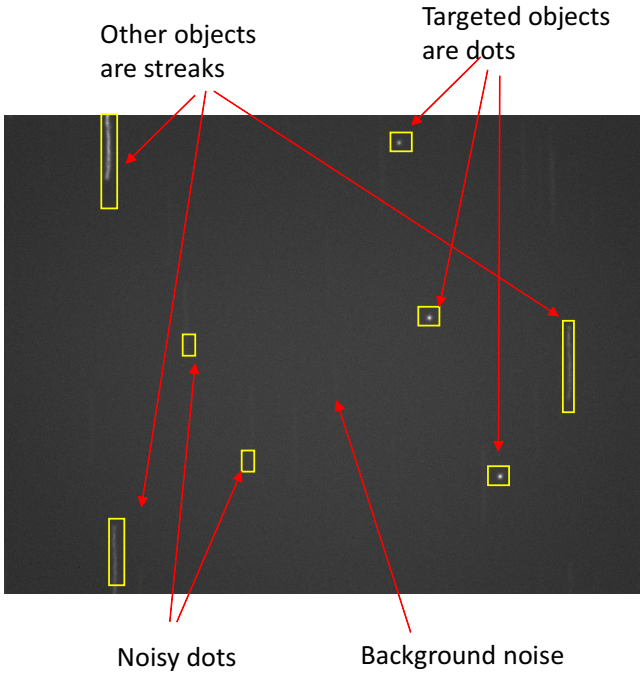

Fig. 3 A sample image captured by a ground-based telescope about satellites. The three dots are the targeted satellites and the streaks are other objects passing the scene.

Images from ground-based telescopes have several important characteristics that distinguish their processing from normal ones. First, objects on space images are presented as dots and streaks, as shown in Figure 3. No special shape in-

formation available except for size and intensity[5]. Second, these images are often noisy with varying quality such as low intensity, low contrast, and out-of-focus. As the telescope is set to track the targets synchronously, the key objects are captured as dots in images. Sometimes they might be too low intense to be visually detected - however, can be still detected via algorithms. Streaks are objects with different speeds moving through the scene, which may include unexpected objects.

These unique features demand special processing algorithms. Our strategy is to first identify the pixels with relatively high intense, and then use a cost-effective clustering algorithm to identify the primary objects (i.e., the dots and streaks). Experiments show that this design works satisfactorily for the major tasks: image quality classification and key object identifications with image sequence. The statistics of the basic object features, together with the histogram information, are used directly to identify good images in the image classification and cleaning task. The basic object features fed to the sequence-based analysis to identify the key objects among the noisily identified objects accurately.

In the following, we describe the design of the algorithm. The basic idea is to use the simplest techniques to achieve the most effective results.

Customized Histogram Based Thresholding. Histogram-based methods are the most popular for converting the grayscale images into binary images ready for efficient object extraction. The solutions for general cases has been studied intensively [5]. However, the unique features of space image may allow us to develop a specialized and more efficient customized algorithm, which works best for the streaming case. We consider a simplified histogram based thresholding algorithm. The well-known histogram based algorithm is the Otsu algorithm [10], which checks each gray level (e.g., 0 to 255) to find one that optimally separates the foreground and background pixels. However, it can be unnecessarily expensive for our application as we can quickly narrow down the target gray level for space images. One intuition is that the light portion in a space image is often tiny, compared to the dark area. The histogram of pixels can quickly capture this unbalanced distribution. As Figure 4 shows, most pixels have a grayscale (intensity) around 60; only a very small portion of pixels have high intensity. In practice, we find that the threshold of gray level can be set so that the left side of histogram contains very high percentage (e.g., about 90%) of all pixels and the right side represents the pixels of interest. It is possible to fine tune this percentage to minimize the inclusion of background noise. However, it is not necessary since the late "noise removal" process will effectively remove them. In experiments, we found

[5] The shape information may be available for low-orbit objects captured by radars [4], which, however, are not the objects targeted by ground-based telescopes, due to the objects' high orbital velocity.

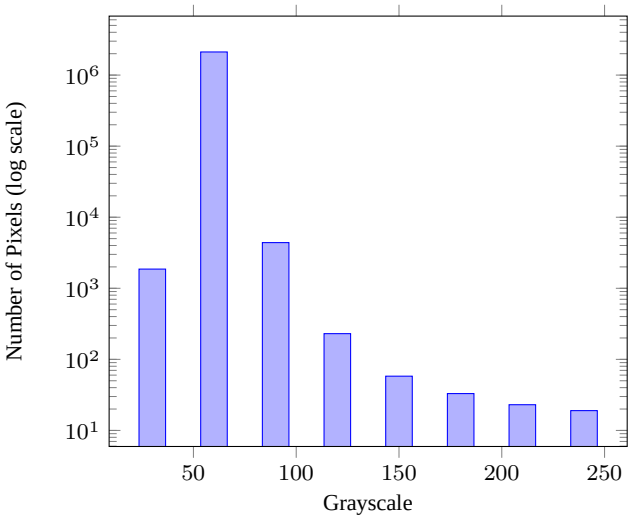

Fig. 4 The histogram of a sample image captured by a ground-based telescope.

that 90% works just fine. Note this procedure is highly efficient, with time complexity $O(n)$, linear to the number of pixels n.

Noise Removal. The result of fast thresholding may leave small noises in the binary image, which can be further removed by the classical method - the combination of *erosion* and *dilution* [12], with a cost of $O(n)$. We found this

Pixel Grouping. The remaining white pixels are grouped by a simple eight-neighbor approach. Specifically, if a pixel is one of the eight neighbors of another pixel, we consider these two pixels are connected, which forms connected pixel graphs. A bread-first search is applied to such graphs to identify the connected components, which only costs $O(m)$, where m is the number of white pixels. m is usually very small (e.g., < 100) after the threshold and noise removal steps.

Clustering of Pixel Groups. The pixel groups formed in the last step are further checked by the distances between the groups. A simple agglomerative clustering method is applied. It starts with each pixel group as a cluster. The distance is defined as the minimum distance between the bounding boxes of the pixel groups. In each step, if a pair of clusters has their distance less than a threshold δ, they are merged. Let the number of pixel groups be p. The complexity of this step is about $O(p^2 \log p)$. Since p is often small (e.g., typically less than ten), the actual cost of this step is low. According to the shape and size of the cluster, we label them with "dot", "streak", or "invalid". Invalid objects are those with very large size likely caused by hardware problems such as an ill-tuned camera.

Algorithm 1 gives the details of these steps. In experiments, we show that this approach is much faster than other candidate methods.

Algorithm 1 Fast Online Object Extraction

Input: image I, histogram-threshold τ, cluster-threshold δ;
$h \leftarrow$ get the histogram of I;
$b \leftarrow$ find the grayscale that cuts the left of the histogram h to have $\approx \tau$ percent of the total pixels;
$B \leftarrow$ convert the image I to binary with the threshold b;
erode(B) and then dilute(B);
apply Breadth-First-Search to find the connected white pixels, which forms p pixel groups;
apply hierarchical clustering on p pixel groups to find the final clusters and label them according to their shapes;
return bounding boxes of the labeled objects;

The identified objects can effectively support other tasks such as image quality classification and key object tracking. In image quality classification, features are extracted from the captured objects together, and the basic histogram information, and then a classifier is trained to separate the bad quality images from good ones. We will describe the details in experiments.

ID	feature description
1	intensity at 50% of histogram
2	intensity at 90% of histogram
3	total number of objects
4	number of dots
5	number of streaks
6	average size of dots
7	maximum size of dots
8	minimum size of dots
9	average intensity of dots
10	maximum intensity of dots
11	minimum intensity of dots
12	average size of streaks
13	maximum size of streaks
14	minimum size of streaks
15	average intensity of streaks
16	maximum intensity of streaks
17	minimum intensity of streaks

Table 1 Feature design for image cleaning.

Note that the objects identified by this algorithm are quite noisy, which is a trade-off we make for fast online processing. Figure 3 shows an example, where the identified objects are boxed. However, this information together with the basic image statistics is sufficient for image quality classification. Table 1 shows the features used by the quality classification. The noisy objects can also be effectively eliminated with a stack of sequentially captured images as shown in the next section.

4.2 Key Object Identification and Anomaly Detection in Image Streams

A single image may contain objects other than the target ones, such as passing satellites, meteors, and background stars, which are difficult to distinguish by processing only one image. However, the key objects (identified as small square-shaped boxes in the images) should continuously appear in a series of sequentially captured images, while others may not. Also, as the telescope is precisely tracking the objects according to their trajectory model, the key objects must show up in each image around the same position. In contrast, the moving objects such as passing satellites and meteors may only appear in a few images or form streaks, and the background stars that have much lower intensity should have been eliminated from the images after the preprocessing steps.

Key-Object Identification with an Image Sequence. The key-object identification algorithm extracts the key objects from a sequence of images. It is used in the setup stage of stream monitoring (i.e., in the first 10 minutes of observation), and the image query processing algorithm that will be discussed later.

The algorithm needs to handle some special situations observed from real datasets. With an appropriate setup, the key objects should appear around the same position in the sequentially captured images. However, in practice, we often observed small levels of deviation from sequences to sequences, which should be tolerated by the algorithm. Deviations may be caused by some system errors, e.g., the mechanic and optical systems. Figure 5 and 6 show different levels of deviations observed in real image streams.

Considering such normal small deviations, we design the key-object detection algorithm as follows. First, we process images by a sliding window, which is a typical method for streaming data processing [2]. Each image is called a "frame". The window size is denoted as s frames. Objects may only be detected in a few of the s frames in a window, which leads to the *frame missing rate*, i.e., the number of missed frames divided by s. We set the maximum frame missing rate as r. Second, we collect all objects detected from the s frames in one dataset for clustering. It aims to find the object clusters that satisfy the following conditions: (1) the cluster contains more than $(1-r)s$ objects, and (2) the diameter of cluster [6] is smaller than δ. Intuitively, it is like to stack all frames together in one frame and find the repetitively appearing objects around certain locations. The diameter δ defines the allowed deviation range and $(1-r)s$ means the object appears at least in $(1-r)s$ frames in the window.

The key reference objects and the reliable parameter settings of s and r are established in the first few (e.g., 100-200) frames of the observation period, to adapt to different obser-

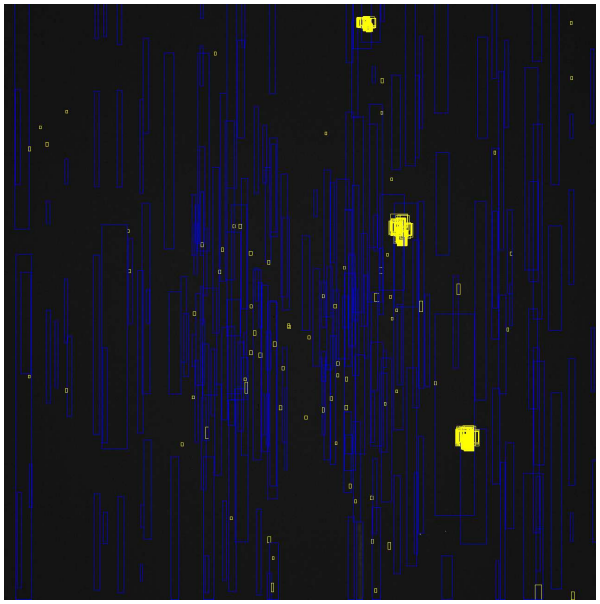

Fig. 5 Typical image sequences, where the key objects appear in almost every frame and they stack in the small areas. Yellow boxes represent the identified dot objects in the first stage processing; blue boxes are the streak objects.

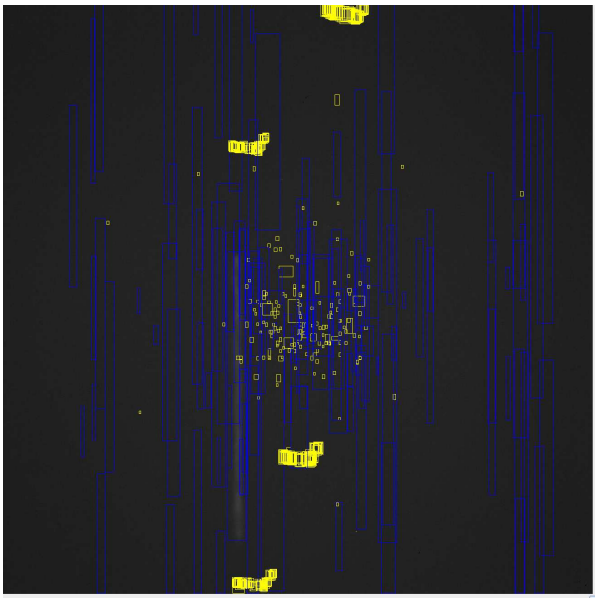

Fig. 6 In some sequences, the key objects may drift about in some irregular tracks.

vational conditions. The details of the algorithm are given in Algorithm 2. Specifically, we observed that the cluster deviation diameter δ can be fixed (e.g., $\delta = 40$) based on a large set of normal sequences. However, the setting of s is complicatedly related to the noisiness of the sequences (i.e., the missing rate r). To adaptively find the stable settings, we consider the following approach to handle this uncertainty. First, the analyst gives the bounds of candidate window size s: s_{min} and s_{max}. The aim is to find the appropriate window size that gives low missing rate *stably* over the sliding windows in the first n frames. Then, a probing process is applied for each setting of window size s_i, to compute the *maximum* missing rate, r_i of the top-k clusters sliding over the n sequential images. The clusters are ranked by the number of frames that the contained "dot" objects in the cluster are from. Finally, among all pairs (s_i, r_i), we find the pair that gives the lowest maximum missing rate r_{min}.

Algorithm 2 Probing the best setting of window size and missing rate

Input: image $\{I_i, i = 1..n\}$, window size range $[s_{min}, s_{max}]$, cluster-threshold δ, and the number of key objects k;
for s_i from s_{min} to s_{max} **do**
  find the maximum missing rate r_i over sliding windows (Algorithm 3);
end for
return (s_{opt}, r_{opt}), where
$s_{opt} = argmin_{s \in [s_{min}, s_{max}]}\{r_i | (s_i, r_i)\}$;

Algorithm 3 Find the maximum missing rate over sliding windows

Input: image $\{I_i, i = 1..n\}$, window size s, cluster-threshold δ, and the number of key objects k;
$S \leftarrow \{I_1, .., I_s\}$;
clustering the "dot" objects in the sequence S (Algorithm 4);
find the maximum missing rate of the top k clusters, r_1;
for i from 2 to n-s **do**
  remove I_{i-1} from S and append I_{i+s-1} to S;
  remove the objects of I_{i-1} from the clusters;
  assign the objects of I_{i+s-1} to the clusters (Algorithm 4);
  find the maximum missing rate of the top k clusters r_i;
end for
return $\max\{r_1, .., r_{n-s}\}$;

References and Anomaly Detection. In processing an image stream, we first establish the reference sequences that represent the normal situations. The first few windows in the whole observation period (e.g., one night of observations) are used to find the best parameter settings and the stable key objects (Algorithm 2). It establishes the "normal cases" for anomaly detection, which can also be cross-validated with the historical information in the image repositories (or verified by the analyst for objects tracked at the first time). Each stable key object is represented as a cluster of key objects in the stacked images. Once the key objects are established, the objects in the late windows are compared to the key objects to detect anomalies.

We define "anomaly cases" with two categories: *the disappearance of the target objects*, and *the emergence of new objects*. Both can be captured with the identified object clus-

Algorithm 4 Clustering "dot" objects in one image to previous clusters

```
Input: image I, cluster-threshold δ, a set of existing clusters C;
for each "dot" object o_i in I do
    for each cluster c_j in C do
        if distance (o_i, c_j) ¡δ then
            add o_i to c_j and break;
        end if
    end for
    if o_i does not belong to any cluster in C then
        create a new cluster that contains only o_i and add the cluster
        to C;
    end if
end for
return the updated C;
```

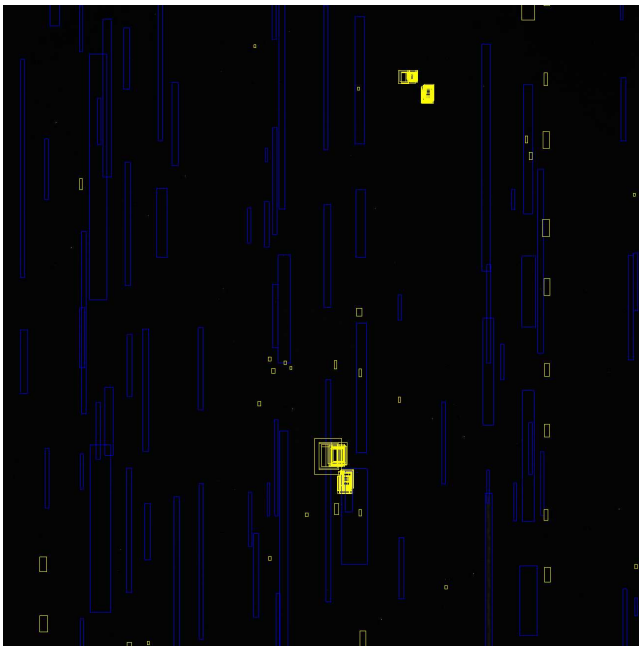

Fig. 7 Significantly shifting key objects.

ters. Specifically, for the first category, we flag all frames that miss any of the identified key objects. Once a sequence of continuously flagged frames (α frames) is detected, a key-object-missing alarm is raised. Meanwhile, we also keep detecting *emerging key objects* by applying the key-object identification algorithm to the sliding window $[i - s, i]$, where i is the index of the current frame. Once a new emerging key-object cluster is detected, we also raise the alarm.

We have observed in real datasets that, in very rare cases, the deviation is so large (e.g. Figure 7) that causes both disappearance and emerging alarms. Thus, once a frame is detected with both types of alarms, it is very likely that large deviation happens, which will need to be visually verified by the analyst. After the verification, the key-object identification process needs to be restarted.

Note that the overall computation cost for tracking the information is low, only related to the number of objects detected in the feature extraction stage, which is typically small with our highly effective object extraction algorithm in the early stage.

4.3 Geometry-Based Indexing and Query Processing

High-quality image sequences are sampled, processed, and saved to the image repository for future analysis. Supporting image queries is one of the major functions for the image repository. In the following, we focus on processing the similarity query, i.e., finding the historical images sequences that are similar to a given sequence of image.

In the preliminary study, we have observed that typically more than one geostationary satellite [6] can be captured in the same frame for each observation period, which provides important geometry information for accurately indexing and querying sequences. Specifically, the distances between the key objects can be used to create a pattern of object distribution, which are then used for indexing and query processing. This method is very reliable for geostationary objects. Non-geostationary objects can also be possibly described with multiple patterns, combined with their trajectory models and observation times.

Geometry-based indexing is necessary for fast query processing. Specifically, we define the problem as follows. We assume some *historical sequences of images* are stored in the image repository for query and analysis. These sequences are of good quality, well cleaned, and explicitly labeled by the detected target objects. A *query* consists of a sequence of sequentially captured images (in typical cases) or one single image (in unusual cases). We want to find the most relevant sequences of images in the image repository. It will be an important tool for many analytical tasks.

The key problem is to define the features for indexing and query processing. We consider two types of information: the geo-location of a telescope station and the geometry information of the key objects. The index is a hierarchical tree-like structure, designed as follows. The first level is the geo-location of a station. For the query images not taken by the stations in the repository, it is best to search for the images related to the closest station. More sophisticated methods can be used to transform the object geometry between stations for the same target object. However, they are out of the scope of our current study. The second level is the number of key objects in the image sequences. Finally, each node in the third level represents an image sequence that was taken from the specific geo-location and has the specific number of key objects, containing the geometric information of the key objects in the image sequence. Figure 8 illustrates this indexing structure.

[6] http://solarsystem.nasa.gov/basics/bsf5-1.php

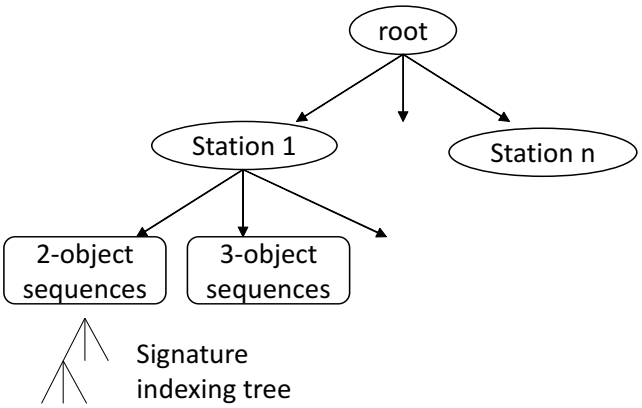

Fig. 8 Index structure for geometry-based query processing.

For each sequence in the third level, we compute the pairwise Euclidean distances between the key objects, and sort them to form a distance vector for the specific image sequence, denoted as $d_{l,k,i}$, where l represents the location id, k is the number of key objects, and i is the sequential id of the image sequence in this specific category. For two objects in the sequence, there is only one pairwise distance, and thus the $d_{l,k,i}$ vector has only one element. Three objects will have three elements, and n objects will have the vector length $\binom{n}{2}$. These distances are also normalized to the range $[0, 1]$ according to the image size[7]. Similarly, we also compute pairwise cosine similarity values and sort them to form a vector, denoted as $c_{l,k,i}$, with the same definition of l, k, i. Elements of $c_{l,k,i}$ are in the range $[0, 1]$. Specifically, the Euclidean distances define the relative position of the objects, and the cosine similarity measures define the relative angles between the objects. The two vectors together form the signature vector $s_{l,k,i} = (d_{l,k,i}, c_{l,k,i})$ used in similarity based sequence search. A multidimensional spatial index [9] can be built with the signature vectors for the specific branch.

Query processing is straightforward with such an index. If a query consists of a series of sequentially captured images, each image in the sequence is preprocessed with the previously discussed basic object extraction algorithm, and then the sequence-based key-object identification algorithm is used to spot the key objects. Then, the signature vector of the query sequence, s_{query}, can be computed. With the station location and the number of key objects, the branch in the third level is identified. We then conduct a similarity search between the signature vectors in the branch and the query signature vector using Euclidean distance. The top k most similar sequences are returned as a result.

[7] For an image of size $x \times y$, the maximum distance is $\sqrt{x^2 + y^2}$. Each computed distance is divided by the maximum distance to get the normalized value.

5 Experiments

As mentioned in the framework, the system has three major tasks: (1) image cleaning, (2) anomaly detection, and (3) sequence querying, while object detection is the key component for all the three tasks. In the following, we will first describe the setup for experimental evaluation and then report the evaluation results for the tasks.

5.1 Implementation

To evaluate the components of the framework, we have built several auxiliary tools with C++, Java, and Python.

Image Browsing and Labeling Tool. This tool provides a GUI for developers to efficiently browse and label the sample images for image cleaning. Each image is labeled by "good", "bad", and "uncertain". With labels and the designed features, we can learn classifiers.

Image Stacking and Visualization Tool. This tool is used to examine the extracted objects and visually evaluate the quality of sequence-based key-object identification. Figure 5 and 6 are from this tool.

Image Query Processing System. It provides a test platform for evaluating query performance. We build the sequence database with the sample sequences collected in one month. Users can submit an image sequence as a query and get the most related sequences in the database.

In addition to these tools, we have implemented the core algorithms with C++ or Java to achieve better performance, which are used by these tools via some glue code in Python.

5.2 Datasets

The real datasets were collected during October 2013 with a 17-inch telescope, containing about 16,400 images in total. Each observation period is about 6 hours during the night. Table 2 shows the date, targeted satellites, the number of collected images, and their quality labeled by the editors. The code names of the tracked satellites can be found on www.lyngsat.com. The captured images have a resolution of 1676×1266 and are stored in the FITS format[8].

These data are visually examined and labeled by two editors with the labeling tool[9]. The last two columns also list the number of good and bad quality images, respectively. For some sequences, there are also uncertain quality images, which are not included in this table.

[8] fits.gsfc.nasa.gov/

[9] The dataset is first labeled by one editor. Then, the other editor verifies the result and identifies the inconsistent ones. Finally, the inconsistent ones are reviewed by both editors and assigned to the agreed category.

Date	Targeted Satellites	# images	good	bad
10/1	AMC18, AMC15	1010	978	32
10/7	DTV8, SES1	1355	1346	9
10/8	DTV4S, SES1, DTV9S	788	763	25
10/9	AMC18, AMC15	1401	1318	83
10/10	DTV12, DTV10	1111	963	147
10/11	DTV4S, DTV9S	1257	1206	40
10/13	DTV8, SES1	989	862	99
10/14	DTV10, DTV12, SW1	816	598	169
10/16	DTV8, SES1	1214	1143	57
10/17	DTV4S, DTV9S	1434	1130	202
10/18	DTV12, DTV10	141	52	72
10/19	DTV8, SES1	1072	1044	16
10/20	SES1, DTV9S, DTV4S	1242	991	220
10/24	DTV12, DTV10	495	470	17
10/25	AMC18, AMC15	1091	916	149
10/27	DTV12, DTV10	603	261	267
10/28	DTV12, DTV10	383	287	62

Table 2 Dataset Description

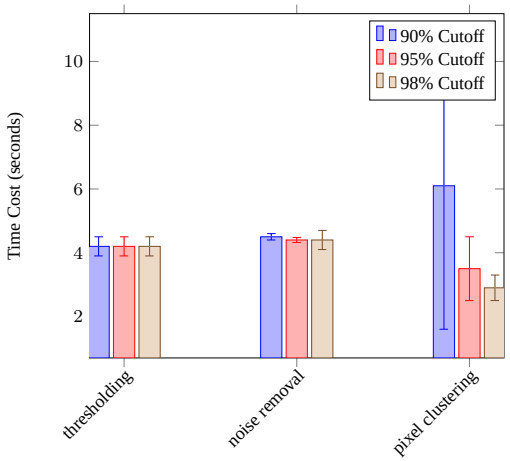

Fig. 9 The average costs of the key steps in online object extraction, based on 10 1000-image batches.

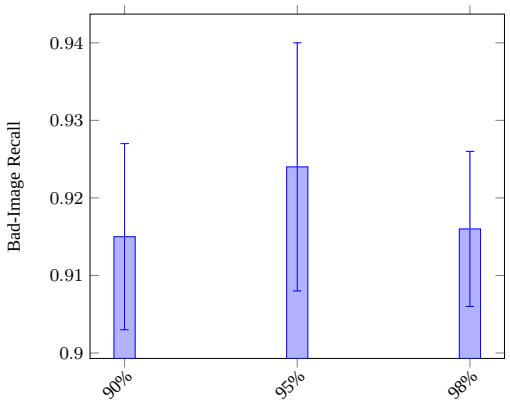

Fig. 10 Thresholding affects the quality of cleaning. Bad-Image recall is important for the task.

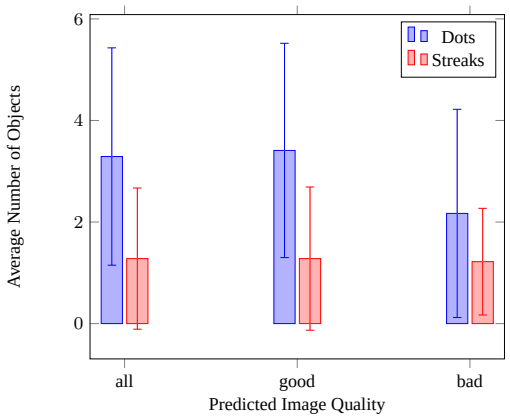

Fig. 11 The average number of "dot" and "streak" objects in all/good/bad images, respectively. Cutoff is 95%.

5.3 Results

Online Object Feature Extraction. We have done experiments on the datasets in 10 batches of 1000 images to find the average and standard deviation of time costs for the simplified thresholding method. We compared the results with the default Otsu algorithm for thresholding [10] and observed that the Otsu algorithm generates too many noisy objects and thus much slower than our algorithm. Overall, for 1000 image batches, our method take 12 seconds, while the default Otsu takes 290 seconds. The high cost is due to the noisy results from thresholding and noise removal steps, which result in the high cost of pixel clustering.

We also compute the cost distribution of the three key steps in object extraction: thresholding, noise removal, and pixel clustering. Figure 9 shows the the average costs of the steps for different thresholds. Again, the numbers are based on 10 batches of 1000 images. With the initial non-optimized implementation, this algorithm can already achieve a speed about 80 frames per second for high resolution images.

Figure 9 also shows that higher cutoff rates for thresholding do not change the cost of the first two steps much, but can significantly reduce that of the last step: pixel clustering. It is easy to understand that higher cutoff rates result in less pixels for clustering and thus lower the cost of clustering. However, overaggressive cutoff (e.g., 98%) may also affect the quality of extracted objects and thus the performance of later tasks. Figure 10 shows that the 98% cutoff rate will downgrade the quality of the image cleaning task. Thus, the later tasks will be based on the features extracted with the 95% cutoff rate.

Figure 11 shows the comparison of the two types of objects, "dots" and "streaks", in images of different quality (categorized as "good" and "bad"). It appears the bad-quality images contains significantly less "dot" objects; while the "streaks" are about the same. Furthermore, it also shows that the average number of extracted objects per image is also quite small, which help reduces the costs of the subsequent tasks: key-object identification and anomaly detection.

Image Cleaning. Based on the detected objects and the histogram information, we generate a feature vector for de-

termining the quality of each image. Table 1 gives the extracted features for image cleaning. The intensity (or gray level) at different positions of the histogram shows the overall brightness and object brightness of the image, respectively.

We then create a balanced training dataset by using all the "bad" examples, plus randomly sampled the same number of "good" examples. Adaboost classifier in the Python Sklearn package is used to learn the classification model with five-fold cross validation. Table 3 lists the results of the five folds. Both precision and recall are higher than 0.9 or around 0.9, which indicates high model quality and the designed features are very effective in modeling.

fold	prediction	precision	recall
1	good	0.933131	0.913690
	bad	0.913947	0.933333
2	good	0.903323	0.952229
	bad	0.955090	0.908832
3	good	0.931548	0.907246
	bad	0.902736	0.928125
4	good	0.895210	0.897898
	bad	0.897281	0.894578
5	good	0.917404	0.928358
	bad	0.919003	0.902141
avg.	good	0.92 ± 0.01	0.92 ± 0.02
	bad	0.92 ± 0.02	0.91 ± 0.02

Table 3 Result of five-fold cross validation in image quality classification.

By using the utility in the Python sklearn package, we can also obtain the importance of the top-10 features, which is given in Table 4. They match our intuition on judging the quality of space-object images.

rank	feature ID	feature Name	importance
1.	feature 10	maximum intensity of dots	0.167867
2.	feature 2	intensity at 90% of histogram	0.093028
3.	feature 1	intensity at 50% of histogram	0.091448
4.	feature 9	average intensity of dots	0.091435
5.	feature 7	maximum size of dots	0.063282
6.	feature 6	average size of dots	0.061405
7.	feature 11	minimum intensity of dots	0.061277
8.	feature 4	number of dots	0.057875
9.	feature 13	maximum size of streaks	0.046410
10.	feature 3	total number of objects	0.042003

Table 4 Feature importance computed by sklearn.

Key-Object Identification. We apply the parameter probing algorithm for key-object identification to all the sample image sequences. The first 150 images of the testing sequences are used for setting up the parameters and identifying key objects. The cluster-diameter is set to 40 pixels, and the number of continuous sliding windows is set to 50. The window sizes are probed in the range [20, 100] - thus 150 frames are sufficient for this setting. By searching the parameter space using the method described in Section 4.2, we find the best parameter settings. We have manually verified The correspondingly identified key objects. Table 5 presents the parameter settings for different sequences.

sequences	window size	max. missing rate
10/01	68	0.22
10/07	90	0.01
10/08	26	0.04
10/09	45	0.08
10/10	50	0.02
10/11	28	0.04
10/13	70	0.09
10/14	100	0.11
10/16	40	0.08
10/17	100	0.01
10/18	52	0.38
10/19	100	0.19
10/20	58	0.14
10/24	62	0.02
10/25	66	0.55
10/27	90	0.30
10/28	100	0.13

Table 5 Adaptive parameter settings for finding the key objects.

Indexing and Query Processing. The query processing algorithm depends on the geometry-based indexing. It analyzes the query sequence first to extract the geometry of the key objects and then searches the index to find the answers. We use the key objects detected by the last experiment to build the geometry-based image-sequence index. To test the query performance, we randomly sample the whole set of image sequences to generate the query sequences, each of which consists of 100 sequential images. The same key-object identification algorithm is applied to detect the key objects in the query, and then their geometry is used to query the sequence index. We repeat the experiment for five times - each time, we generate 50 random queries and compute the query accuracy. Recall that the similarity between the query and a sequence in the database is defined as the Euclidean distance between their feature vectors. We can rank the related sequences by their similarity to the query sequence. Finally, the top 2 query results are used in the evaluation.

We list the experimental result in Table 6. Since the query sequence has a limited length, it is possible that the key object cannot be reliably identified. Specifically, if no key object or only one key object is identified in the query sequence, the query cannot be processed due to the incomplete geometry information. Table 6 lists the percentage of such cases in the first two columns "0-obj" and "1-obj" queries.

The top-k accuracy is defined as follows: if the desired result appears in the top-k results, we consider the query processing is successful; the number of successful queries divided by the total number of queries is the top-k accuracy. Apparently, the top-2 results will have equal or higher accuracy than top-1 results. Note that the top-1 and top-2 accuracies are computed by excluding the 0-obj and 1-obj queries that cannot be processed. The overall accuracy of top-2 results is 0.92, which is considered highly acceptable

by domain experts. Further studies will be needed to reduce the number of 0-obj and 1-obj queries, and improve the top-k accuracy.

test	0-obj queries	1-obj queries	top-1 accuracy	top-2 accuracy
1	3	2	84%	96%
2	1	5	77%	93%
3	1	7	81%	90%
4	2	6	79%	88%
5	1	4	82%	91%
avg.	1.6±0.8	4.8 ± 1.7	81% ±3%	92% ±3%

Table 6 Testing results for image-sequence query processing. 5 test batches are used. Each batch has 50 randomly selected query sequences from the repository, each of which consists of 100 sequentially captured images.

6 Conclusion

Space surveillance with ground-based telescopes is an economical option complementary to other space monitoring techniques. Due to its low cost, many ground stations can be possibly deployed around the world to automatically and continuously produce space object images with little or without human intervention. The challenging problem is to process and utilize these image streams effectively. The proposed SPIN framework aims to develop effective and efficient methods to address several important problems: (1) automatically cleaning image streams generated by the ground-based telescopes, (2) monitoring the known space objects and detecting anomalies such as disappeared key objects and emerging new objects, and (3) querying existing image sequence databases with sample image sequences.

The proposed algorithms have been evaluated with a large collection of space-object images that was collected during one month, containing about 16,400 images in total. The experimental evaluation has several results. (1) The image quality classification algorithm can achieve higher than 90% in both precision and recall. (2) We can accurately identify all the key objects in the testing sequences with adaptive parameter settings. (3) The geometry-based image-sequence indexing and query processing algorithm can give above 90% accuracy on top-2 results. These results show that the developed SPIN framework has achieved the design goals satisfactorily.

In the ongoing work, we will further improve the efficiency and accuracy of the core components. More datasets will be collected from multiple sites around the world. We will also develop methods to build up the correlations among the image sequences of the same object observed from multiple sites.

Acknowledgements This project was supported by a DAGSI/AFRL award. Many thanks to the AFRL colleagues who helped set up the instruments and collect the images.

References

1. of Universities for Researches in Astronomy, A.: Space-based vs. ground-based telescopes with adaptive optics (ao). http://www.aura-astronomy.org/news/archive/hst_vs_ao_2.pdf
2. Babcock, B., Babu, S., Datar, M., Motwani, R., Widom, J.: Models and issues in data stream systems. In: Proceedings of ACM Conference on Principles of Database Systems (PODS) (2002)
3. Deng, J., Berg, A.C., Li, K., Fei-Fei, L.: What does classifying more than 10,000 image categories tell us? In: Proceedings of the 11th European Conference on Computer Vision: Part V, ECCV'10, pp. 71–84. Springer-Verlag, Berlin, Heidelberg (2010)
4. Ender, J., Leushacke, L., Brenner, A., Wilden, H.: Radar techniques for space situational awareness (2011)
5. Glasbey, C.A.: An analysis of histogram-based thresholding algorithms. CVGIP: Graph. Models Image Process. **55**(6), 532–537 (1993)
6. Jain, A., Murty, M., Flynn, P.: Data clustering: A review. ACM Computing Surveys **31**, 264–323 (1999)
7. Jing, Y., Baluja, S.: Pagerank for product image search. In: Proceedings of the 17th International Conference on World Wide Web, WWW '08, pp. 307–316. ACM, New York, NY, USA (2008)
8. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. Int. J. Comput. Vision **60**(2), 91–110 (2004)
9. Manolopoulos, Y., Nanopoulos, A., Papadopoulos, A., Theodoridis, Y.: R-trees: Theory and Applications. Springer-Verlag (2005)
10. Otsu, N.: A threshold selection method from gray level histograms. IEEE Trans. Systems, Man and Cybernetics **9**, 62–66 (1979). Minimize inter class variance
11. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A.C., Fei-Fei, L.: ImageNet Large Scale Visual Recognition Challenge. International Journal of Computer Vision (IJCV) **115**(3), 211–252 (2015)
12. Soille, P.: Morphological Image Analysis: Principles and Applications, 2 edn. Springer-Verlag New York, Inc., Secaucus, NJ, USA (2003)
13. Szegedy, C., Toshev, A., Erhan, D.: Deep neural networks for object detection. In: C.J.C. Burges, L. Bottou, M. Welling, Z. Ghahramani, K.Q. Weinberger (eds.) Advances in Neural Information Processing Systems 26, pp. 2553–2561. Curran Associates, Inc. (2013). URL http://papers.nips.cc/paper/5207-deep-neural-networks-for-object-detection.pdf
14. Trier, Ø.D., Jain, A.K., Taxt, T.: Feature extraction methods for character recognition - a survey. Pattern Recognition (PR) **29**(4), 641–662 (1996)
15. Viola, P., Jones, M.: Rapid object detection using a boosted cascade of simple features. In: Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on, vol. 1 (2001)
16. Wan, J., Wang, D., Hoi, S.C.H., Wu, P., Zhu, J., Zhang, Y., Li, J.: Deep learning for content-based image retrieval: A comprehensive study. In: Proceedings of the 22Nd ACM International Conference on Multimedia, MM '14, pp. 157–166. ACM, New York, NY, USA (2014)