Parallel tracking and detection for long-term object tracking

Dan Xiong¹, Huimin Lu², Qinghua Yu², Junhao Xiao², Wei Han¹ and Zhiqiang Zheng²

Abstract
High tracking frame rates have been achieved based on traditional tracking methods which however would fail due to drifts of the object template or model, especially when the object disappears from the camera’s field of view. To deal with it, tracking-and-detection-combination has become more and more popular for long-term unknown object tracking, whose detector almost does not drift and can regain the disappeared object when it comes back. However, for online machine learning and multiscale object detection, expensive computing resources and time are required. So it is not a good idea to combine tracking and detection sequentially like Tracking-Learning-Detection algorithm. Inspired from parallel tracking and mapping, this article proposes a framework of parallel tracking and detection for unknown object tracking. The object tracking algorithm is split into two separate tasks—tracking and detection which can be processed in two different threads, respectively. One thread is used to deal with the tracking between consecutive frames with a high processing speed. The other thread runs online learning algorithms to construct a discriminative model for object detection. Using our proposed framework, high tracking frame rates and the ability of correcting and recovering the failed tracker can be combined effectively. Furthermore, our framework provides open interfaces to integrate state-of-the-art object tracking and detection algorithms. We carry out an evaluation of several popular tracking and detection algorithms using the proposed framework. The experimental results show that different tracking and detection algorithms can be integrated and compared effectively by our proposed framework, and robust and fast long-term object tracking can be realized.

Keywords
Parallel tracking and detection, online learning, discriminative model, open interface, long term

Date received: 3 August 2019; accepted: 2 January 2020

Introduction
As one of the major research topics in computer vision, object tracking is defined as estimating the location of a tracked object in continuous frames. With the springing up of high-power computers, the availability of high-quality and cheap cameras and the need for automated video analysis, it has been attracting increasing attention. The achievements in object tracking have been applied to many domains such as automated surveillance, traffic control, and augmented reality (AR). However, inevitable

¹Unmanned Systems Research Center, National Innovation Institute of Defense Technology, Academy of Military Science, Beijing, China
²Robotics Research Center, College of Intelligence Science and Technology, National University of Defense Technology, Changsha, China

Corresponding author:
Huimin Lu, Robotics Research Center, College of Intelligence Science and Technology, National University of Defense Technology, 47 Yanwachizheng Street, Changsha 410073, Hunan, China.
Email: lhmnew@nudt.edu.cn

Creative Commons CC BY: This article is distributed under the terms of the Creative Commons Attribution 4.0 License (https://creativecommons.org/licenses/by/4.0/) which permits any use, reproduction and distribution of the work without further permission provided the original work is attributed as specified on the SAGE and Open Access pages (https://us.sagepub.com/en-us/nam/open-access-at-sage).
difficulties would occur during object tracking, especially
long-term object tracking, such as occlusion, changes of
the scene illumination and the object's appearance, low-
quality or compressed images, fast and complex object
maneuvering and real-time requirements.

As is known to all, tracking and detection can be com-
bined to make up their own deficiencies. A tracker can
provide weakly labeled training samples for a detector
which can help to reinitialize or correct the former when
the tracked object is lost, or large drifts exist. In the
Tracking-Learning-Detection (TLD) algorithm, the track-
ing and detection are performed simultaneously to realize
long-term object tracking. However, due to the fact that
the tracking and detection tasks are placed in the same
thread and run sequentially at the same frame rates, thus
it presents several limitations in practical applications. For
example, the consumption of computing resources must be
limited to the right range so as to guarantee the real-time
performance; therefore, many popular and successful track-
ing, detection, and learning algorithms in the pattern rec-
ognition community with high computation cost cannot be
integrated into the TLD framework. What’s more, the
appearance of the object usually changes slowly during
long-term object tracking. Typically, it is unnecessary to
perform both multiscale object detection and object model
updating for all frames, from which a detector can be run at
a lower frame rate than that of a tracker.

Parallel tracking and mapping (PTAM) inspires us by
dividing tracking and mapping into two separate parts
which are processed in parallel threads on a multicore com-
puter.2 We here find that the front-end tracking thread can
realize real-time camera pose estimation due to lower com-
putation costs. The back-end mapping thread usually
requires more computing resources to map the perceived
environment, which can be processed at low frame rates.
PTAM has been successfully applied in AR systems and
simultaneous localization and mapping for mobile robots.

In this article, we propose a parallel tracking and detec-
tion (PTAD) framework which contains two parallel mod-
ules: a tracker as a front end is performed in real time with
low computation costs; a detector including online learn-
ing and object detection regarding as a back end needs
more computing resources and can be processed at a lower
speed. Furthermore, our framework provides an interface
to integrate and test different tracking and detection algo-
rithms as open source. In this article, several tracking and
detection algorithms are integrated to realize object track-
ing, and then evaluated and compared on several bench-
marking datasets.

The rest of this article is organized as follows. The
related work are presented in the second section. We
describe our whole framework and introduce several pop-
ular tracking, detection, and learning algorithms in the third
section. In the fourth section, we validate the effectiveness
of the proposed PTAD framework by integrating the algo-
rithms introduced in the third section and comparing their
performance. Finally, we give a brief summary of our pro-
posed framework and a statement of the future work.

Related work

As a major and challenging research topic in computer
vision community, object tracking has been studied for
decades. A large number of papers can be found, and read-
ers can refer to surveys or visual object tracking chal-
enges on various benchmarking datasets. In this section,
we just introduce some work related to tracking and detec-
tion modules of our PTAD framework briefly.

Object tracking

The core of object tracking is to robustly estimate the loca-
tion of an object in every frame of an image sequence.
Effective appearance modeling for the tracked object is a
key factor to realize robust and accurate object tracking,
which generally consists of two components: visual repren-
tation and statistical modeling.

Essentially, visual representation refers to extracting
salient visual features and designing appropriate descrip-
tors for a tracked object. Ideal visual representations should
be qualified to distinguish the tracked objects and the back-
ground, and preferably keep invariant to translation, scale,
and rotation as well as robust to affine transformations. It is
unrealistic to design a robust visual representation for all
types of tracked objects. Therefore, various visual repres-
entations have been proposed, which can be divided into
two categories: global visual representations and local
visual representations. Global representations describe glo-
bal characteristics of the object appearance through global
visual features and descriptors such as color histogram.5
They are generally simple and computationally efficient
but sensitive to global appearance changes. For local repres-
entations, local visual features are used to characterize the
object appearance. Although local visual features require
more computing resources, they have many advantages:
invariant to image translation, scale, and rotation; partially
invariant to illumination changes and affine 3-D projection;
robust to partial occlusion and relatively insensitive to
viewpoint changes. Many local visual feature methods
have been proposed for object tracking and detection.6
With the rapid development of artificial intelligence tech-
nology, deep visual features learned by convolutional
neural networks has been employed in object tracking.
Ma et al.7 exploited visual features extracted from deep
convolutional neural networks to improve tracking accu-
racy and robustness. Deep features were used to train cor-
relation filters, and good tracking performance have been
achieved. Furthermore, for object tracking, especially for
long-term object tracking, multiple types of features can be
combined to model the appearance of tracked objects,
which can help to improve the robustness.
According to statistical models, trackers can be typically classified into generative and discriminative trackers. Generative trackers build an object appearance model, and then find out the most similar region in the image by fitting the model. The generative model can be built off-line or online, while it should be able to be updated online for adapting to changes of the object appearance. Wang et al. proposed the spatial-color mixture of Gaussians appearance model by simultaneously encoding both of the color and the spatial layout information, which has a good discriminative ability. In recent years, sparse representation and compressive sensing techniques have been widely used to learn some generative models, which can help to improve the tracking robustness and accuracy. The major problem of online generative models is lack of consideration on the background information. As a result, these trackers suffer from failures in cluttered environments, especially when the appearance of the background regions are similar with that of the tracked object.

Discriminative trackers, which aim at building a binary classifier by maximizing the separability between the object and the background, take visual object tracking as a binary classification problem. Different from generative models which only use the object appearance information, discriminative trackers model both of the object and its surroundings. Kalal et al. proposed a semi-supervised P-N learning method for object tracking to build binary classifiers with structural constraints. In recent years, correlation filters have been widely applied in object tracking, and good performance has been achieved. Ma et al. designed three correlation filters to estimate the translation, scale variations, and the confidence of tracking results. In addition, they trained a random fern classifier to redetect the lost object. Bertinetto et al. combined correlation filters and histogram scores to deal with motion blur, illumination changes, and deformation. A spatiotemporal context learning method with multichannel features is proposed for object tracking, and good performance was achieved. Lukežič et al. proposed deformable parts models with correlation filters for object tracking.

Recently, deep object tracking approach has been widely applied. Held et al. trained off-line neural networks that can track novel objects at 100 frames per second (FPS). In particular, to improve tracking performance, the fully convolutional Siamese approach has been applied in object tracking by making use of region proposals, ensembling, segmentation, and memory networks.

**Object detection**

Object detection is defined as identifying the presence of objects and locating identified objects in an image. Object detection can be divided into two phases: the learning phase and the recognition phase. In general, detectors firstly train classifiers using prelabeled samples off-line. Then the detectors recognize and locate objects online in an image using the trained classifiers. The traditional object detection is not appropriate for object tracking due to several reasons. Firstly, the learning phases of these methods cannot be run in real time due to the consumption of large computational resources. Secondly, the tracked object needs to be manually selected in the first frame in practice, which is difficult to obtain enough prelabeled samples to train highly discriminative classifiers. Finally, these detectors are trained for a generic category of objects (such as any car) rather than a single specific object (such as a particular car). As a result, these detectors are hard to classify similar objects from the same category in object tracking. Essentially, during object tracking, detectors can obtain a small number of labeled samples from the first image and a large number of unlabeled samples from subsequent images. Therefore, the learning methods for object tracking should be competent to exploit both labeled and unlabeled samples for online incremental learning and training. As a result, online semi-supervised learning methods have been proposed to solve these problems.

Online semi-supervised learning can be mainly categorized into self-learning and co-learning. Self-learning is a widely used semi-supervised learning framework, which firstly trains an initial classifier with the labeled data. The labels of the unlabeled data are predicted in advance according to the classifier, and the most confident data are added into the training set. Then the classifier is retrained using the new training set. This procedure is run iteratively and repetitively. In the work of Grabner et al., online boosting learning algorithms based on self-learning were proposed for selecting discriminative features to train classifiers. The discriminative capability of each feature can be ranked in a feature pool, and top-ranked features are selected to construct classifiers for object detection and tracking. The self-learning support vector machines (SVM) can be used to train SVM classifiers with strong discriminative competence to classify the object and the background. Ding et al. proposed a novel manifold regularized model in a reproducing kernel Hilbert space to solve the online semi-supervised learning. Chen et al. proposed a novel online semi-supervised learning framework which exploited multiple regularization terms based on the notion of ascending the dual function in constrained optimization. Liu et al. proposed an adaptive and online semi-supervised least square SVM with a manifold regularization for the online classification of streaming data. Self-learning is a kind of wrapper algorithms, whose performance is affected greatly by the chosen supervised learning method and the collected labeled and unlabeled data. For example, when unlabeled training samples are misclassified or polluted, tracking errors will be accumulated during object tracking, which will result in great drifts and even failures.

To deal with these problems, researchers developed co-learning frameworks for object detection by building multiple and independent classifiers. In co-training, it is
assumed that features can be divided into several conditionally independent subspaces. Classifiers are trained using the prelabeled data in the feature subspaces. Then each classifier classifies the unlabeled data, and most confident data (data and its predicted label) are recorded. The data recorded from one classifier are regarded as training samples to train other classifiers. The co-learning strategy has been widely used for object detection and object tracking. Zeisl et al.\textsuperscript{28} proposed an online semi-supervised multiple instance boosting algorithm, which can incorporate more priors and deal with uncertainties to update positive model for occlusions. Li et al.\textsuperscript{29} developed a novel semi-supervised CovBoost method which utilizes the information from auxiliary samples, target samples, and unlabeled samples to train a strong classifier. The CovBoost method adapts to appearance variations of the object and background successfully in experiments. Co-learning strategy can also be introduced into the semi-supervised SVM learning framework, especially multiple kernel SVM for object tracking and detection.\textsuperscript{30} Meshgi et al.\textsuperscript{31} proposed a novel co-learning framework which combined two different detectors: a rapid adaptive exemplar-based detector and another more sophisticated but slower detector with a long-term memory.

After training classifiers, detectors can identify and locate objects in an image. Similar to visual appearance representations mentioned above, the detected object can also be characterized using either local visual features or global visual features. For local visual features, detectors search objects in the whole image using feature matching methods. However, when the area of the object in the image is small or the object has less texture, the number of the features extracted from the object region may be not enough to realize robust object detection using matching methods. Thus, the applications of feature matching methods are restricted in practice. The sliding window-based methods are widely used, where the whole image is divided into a number of subimage patches. Then a classifier is applied to each patch to decide whether it contains the object or not. The size of an object (objects) may not be fixed in an image sequence. To detect objects at any scale and position, researchers made use of a pyramid of images to generate sliding windows with a fixed size. However, it usually requires lots of computing resources to construct a multiscale pyramid of images. Considering the abovementioned reasons, researchers designed sliding windows with different sizes in original images to improve the real-time performance.\textsuperscript{1} However, the number of the sliding windows may still be too large to process in real time, especially for high-resolution images. Thus cascaded classifiers concatenated with several different classifiers were introduced to reduce the computing time.\textsuperscript{32} Each classifier in the cascade architecture either passes a sample to the next classifier when its predicted label is positive or rejects the sample and processes a new sample from the beginning again. A sample that passes through all classifiers can be regarded as a positive sample. In cascaded classifiers, the first few classifiers are dedicatedly designed to classify samples using simple and efficient features, which can be processed at high speed. Thus a large number of samples are rejected with less time consumption using the first few classifiers.

As a subset of machine learning, deep learning based on artificial neural networks has been applied to solve the semi-supervised learning problem, such as pixel recurrent neural networks,\textsuperscript{33} deep convolution generative adversarial networks,\textsuperscript{34} generative adversarial networks,\textsuperscript{35} ladder models,\textsuperscript{36} and auxiliary deep generative networks.\textsuperscript{37} Currently, real-time performance has been achieved in deep learning applications.\textsuperscript{15,38} but training deep neural networks is complicated by the fact that a large number of parameters are updated during training. This slows down the training by requiring lower learning rates, careful parameter initialization and more computing resources, and makes it notoriously hard to realize online, semi-supervised deep learning.

### The framework of PTAD for object tracking

As our work is closely allied with TLD, we make a brief introduction here. TLD was proposed for long-term single object tracking. As shown in Figure 1(b), it mainly consists of three components: a tracker, a detector, and a learning module. The tracker used to estimate the motion state of an object in each frame is named the median flow (MF) tracker that is based on the Lucas–Kanade (LK) optical flow method.\textsuperscript{1} The detector, which is cascaded by a variance filter, a random forest classifier, and a nearest-neighbor classifier (NN classifier), scans each sliding window and makes a judgment that whether the object exists or not. Then the outputs of the tracker and the detector are integrated into a single bounding box as the output result of TLD. The P-N learning module exploits the output result to generate training samples and update the cascaded classifiers. From Figure 1, it can be seen that TLD is different from general tracking-by-detection methods.\textsuperscript{39} In TLD, the tracking and detection are performed separately, and fulfill information exchange via a learning component. This framework has several advantages. On the one hand, an independent detector can recover or correct a tracker when it fails, or drifts are large. On the other hand, the combination of the tracking trajectory generated by the tracker and the object’s location obtained by the detector can provide unlabeled training data with structure constraints. The data can be used to train more discriminative binary classifiers.

However, the TLD framework has some demerits. Firstly, the appearance of the object usually changes slowly during long-term object tracking. For example, as shown in Figure 2, the visual appearance of the panda changes slowly and gradually during several consecutive frames. In most periods of object tracking, visual
appearance always changes gradually and slowly. So it is unnecessary for the learning module to update the classifiers in every frame. Secondly, with the development of tracking techniques, many proposed trackers do not often encounter failures. Thus a detector does not have to be processed in each frame for recovering or correcting a tracker. Finally, in the TLD framework, tracking, learning, and detection modules are operated in the same thread, and each frame is processed by the three modules successively. Currently, there are many tracking, learning, and detection algorithms with good performance which can be used to improve the performance of object tracking, but their computation costs may be quite high. Hence it is difficult to integrate these algorithms into the TLD framework for real-time applications.

Due to the drawbacks mentioned above, we propose a PTAD framework which consists of two independent and parallel components: the tracking component and the detection component which are processed in two different threads, respectively. A tracker similar with the tracking component of TLD is used to localize the object from frame to frame. The tracker is regarded as the front end of PTAD, which should have low computation burden for real-time implementations. In PTAD, if the tracker loses the object, it cannot be self-recovered or reinitialized. The detector including classification and learning modules is defined as the back end of PTAD. For realizing multiscale object detection, a detector must scan lots of sliding windows in the whole image to decide whether a tracked object exists or not. Due to lack of enough prelabeled samples, it is impossible to train a robust and accurate binary classifier off-line for practical applications. Therefore, an online semi-supervised learning algorithm is employed to update the classifier for object detection. The detector commonly requires more computing resources and computing time. In object tracking, the tracker is sometimes inevitable to lose its object or suffer from a large drift, while it can be recovered and reinitialized from failures as expected. In PTAD, the detector can help to reinitialize and correct the tracker.

The architecture of PTAD is shown in Figure 3. In the first frame, an initial object is manually defined, which is used to initialize a tracker and train a detector via semi-supervised learning algorithms. After initialization, the entire process of the PTAD framework is divided into consecutive segments which are shown as cyan boxes. Red boxes in the segments represent that these frames are

Figure 1. General tracking-by-detection paradigms and TLD method for object tracking. For general tracking-by-detection paradigms, a single process deals with tracking and detection simultaneously, and tracking results are brought from the detector’s output directly. For TLD, tracking and detection are processed separately, and their outputs are integrated into learning modules with structure constrains. A detector helps to recover a tracker when drifts are large or the tracker fails. (a) General tracking-by-detection paradigms. (b) TLD method for object tracking. TLD: Tracking-Learning-Detection.
Figure 2. Example frames from benchmarking tracking sequences, which can be found at http://cmp.felk.cvut.cz/tld. We can find that the appearance of the target object (panda) usually changes slowly between several continuous frames.

Figure 3. The architecture of the proposed PTAD. The PTAD framework is split into tracking and detection (including classification and learning modules), which are two separate and independent tasks and processed in two different threads respectively. A tracker regarded as the front end of PTAD can process the image sequence at high speed. A detector is the back end of PTAD. The outputs of the tracker and the detector are integrated into the learning modules. At the same time, the outputs of the detector are used to reinitialize and correct the tracker. Red boxes represent that these frames are processed only by the tracker. The number of these frames in the red box of each segment is adjustable in practical applications, which is represented as $\Delta K_i, i = 1, 2, 3, ..., n$. Similar to PTAM, the first frame in each segment is called a keyframe, and the frames in every red box are called TFs. The entire process of object tracking using PTAD can be decomposed into consecutive segments which are represented by cyan boxes. PTAD: parallel tracking and detection; PTAM: parallel tracking and mapping; TF: tracking frame.
processed only by the tracker. The number of these frames in the red boxes is adjustable in real applications, which has been noted as $\Delta K_i, i = 1, 2, 3, \ldots, n$. Similar to PTAM, the first frame in each segment is called keyframe, and the frames in every red box are called tracking frames (TFs). Both of the tracker and the detector deal with the keyframe in each segment. All the outputs are integrated into a single output to update the classifiers for the detector and correct or reinitialize the tracker.

In PTAD, the precision of the tracker is important in the process of generating accurate training samples for training robust cascaded classifiers with a low misclassification rate. Knowing whether the tracker fails or not matters a lot in order to improve the precision of the tracker. A tracking failure detection module has already been integrated in existing trackers such as the MF tracker. If such trackers are integrated into PTAD, their own tracking failure detection module will be directly used. For other trackers without tracking failure detection modules, the normalized correlation coefficient (NCC) method will be employed to identify their tracking states. In this case, several last correct tracking results before the current frame are stored as templates with which the current tracking result is compared to obtain the pairwise similarities. If the maximum similarity is greater than a threshold, the current tracking will be considered to be successful, or failed on the contrary. If the tracking succeeds, the current tracking result will be regarded as a new template to replace the earliest one. If fails, the detector will reinitialize the tracker, delete all the previous templates, and add a new template cropped from the new initial location. The threshold is a parameter only for the NCC method. The tracking failure detection can help to improve the precision performance of a tracker, at the cost of spontaneous reducing the recall rate. However, robust detectors can help to reinitialize and correct the tracker to improve the recall performance of the tracker. Therefore, it is possible to achieve both high precision and recall rate at the same time for tracking algorithms when they are embedded in PTAD.

When the tracker loses the object, the output bounding box of the detector is used to reinitialize the tracker. Since the two modules are performed in parallel, the tracker may handle with the next several frames during the reinitialization. Based on the assumption that the object appearance varies slowly and gradually among several successive frames, a template matching method has been utilized to locate the object in the current TF. As shown in Figure 3, the detector should reinitialize the tracker in the last frame of each segment. If the detector fails finding the object, it continues to reinitialize the tracker in the following segments until it succeeds. If the tracker succeeds, the detector will process the image patch cropped from the tracking output and obtain a confidence. At the same time, the detector deals with the sliding windows. If the detector fails locating the object in the image and the confidence of the tracking result is high, the tracking result will be added into the learning module to update the classifiers. The two outputs will be merged into a single bounding box to correct the tracker under the circumstance that the detector locates the object with a high overlap rate. Furthermore, if the confidences of the two results are high, the merged result will be also fed into the learning module. On the contrary, we regard the bounding box with the maximal confidence as the final result. The tracker is reinitialized when the confidence of the tracker result is lower than that of the detection result. The affine transformation is operated on the object fed into the learning module to generate enough training samples.

As a significant advantage compared with TLD, various tracking and detection algorithms can be integrated into our PTAD framework under a real-time constraint. In the following sections, several tracking and detection algorithms will be integrated into the framework, and experiments will be performed on benchmarking datasets.

**Trackers**

Here we only focus on single object tracking algorithms. Trackers, which are regarded as the front end of PTAD, should be able to realize real-time operations and accurate object tracking. We introduce two famous tracking algorithms into PTAD.

**Correlation filter-based trackers.** Conventionally, correlation filters are used to measure the similarity between two signals, which can produce high correlation responses for two similar signals and low responses for two distinctive signals. Thus correlation filters can provide a reliable distance metric between the object and the background. Here we give a brief introduction of the core workflow of general correlation filter-based trackers. Firstly, the initial position of an object is manually determined in the first frame and an image patch is cropped from the location. Visual features are extracted from the image patch, which are used to train an initial correlation filter. For each frame after initialization, the estimated position and size in the previous frame is used to crop an image patch from the current frame, where visual features are extracted. A cosine window is applied for smoothing artifacts introduced by wrapped-around edges. Subsequently, correlation operations between current visual features and the previously trained correlation filters are performed based on convolution. Then inverse fast Fourier transform is performed on the correlation results to obtain a spatial confidence map where the object’s position has high peak responses. Finally, trackers can obtain new training samples from the estimated position in the current image to learn and update the correlation filter for other images. Henriques et al.\(^\text{40}\) proposed the kernelized correlation filter (KCF) tracker. Correlation filters combined with kernel methods are more robust to appearance variations than simple linear classifiers. In KCF, the histogram of oriented gradient (HOG)
feature and multichannel color features are fused together to facilitate robust object tracking. In actual experiments from Henriques et al., KCF can be run at over 100 FPS, which actualizes a high accuracy. Here the KCF is employed to evaluate the PTAD framework.

Optical flow trackers. Optical flow is commonly used to capture the apparent motion of an object from spatiotemporal image brightness variations, which is usually estimated by the LK differential method. It has been widely applied in object tracking, and good tracking performance has been achieved. In object tracking, object is represented as a bounding box, and a number of points in the box are selected to estimate the translational vectors between consecutive frames, which yields the object’s motion information. There are two key assumptions for the LK tracking method to be considered. Firstly, a constraint exists on the local brightness constancy, which shows the object in a certain image region has the same apparent brightness during a short time. Secondly, it assumes that the magnitude of the flow is small, and the object in an image region only moves to a local neighborhood of the object. The LK tracker can be processed at high speed. Thus the LK tracker is not only appropriate to deal with small illumination fluctuation, low noises and slight local deformation, but also proper for real-time implementations. In TLD, the MF tracker is based on the LK tracking method and has realized tracking failure detection using Forward-Backward error. The MF tracker is integrated into PTAD for testing our framework.

Detectors

Detectors identify the existence of an object in the whole image by a sliding-window method and locate its position if existed. In this article, object detection is considered as a binary classification problem for generated sliding windows. In order to improve detection efficiency, cascaded classifiers are used to classify these sliding windows.

The sliding window-based method is used to resolve the whole image into a large number of subimage patches in PTAD. To detect an object at any scale and position, two methods are proposed to generate sliding windows with all possible scales and shifts. One is to construct an image pyramid by down-sampling or up-sampling the original image. Then sliding windows are generated in each level of the pyramid with fixed scale. The other is to generate sliding windows with all possible scales and shifts in the original image, where the image pyramid is not required. Every sliding window inherits a scaling factor should be considered. This method is mainly applied in object detection with scale-invariant features.

Variance Filter can help to eliminate redundant sliding windows with less time consumption. The variance of any sliding window \( p \) in an image \( I \) can be represented as

\[
E(I^2(p)) - E^2(I(p))
\]

The sum of pixels is computed in constant time using the integral image. In PTAD, the variance threshold is set to be 50% variance acquired from the image patch cropped from the initial bounding box in the first frame.

The screened sliding windows should be classified again to determine whether they contain objects or not. Several popular detectors will be introduced into PTAD for dealing with the remaining sliding windows. The performance of each detector mainly depends on two key factors: visual features and trained classifiers. The combination of different classifiers and visual features can be used to construct multiple types of detectors. We will give a detailed introduction about these classifiers in next subsection.

Haar-like feature is calculated only using color information, which is very simple. The adjacent sub-windows at random locations and scales within a sliding window are generated, and the difference between the sum of pixels of sub-windows is calculated as the Haar-like feature. Due to the use of integral images, the calculation speed of a Haar-like feature at any size is fast in constant time. It is a preferable selection for real-time tracking and detection systems. For Haar-like features, multiscale object detection can be realized using all possible scales and shifts of an initial bounding box in the original image without constructing a multiscale pyramid of images.

Compressive feature proposed by Zhang et al. has been applied in object tracking successfully. Here we use the compressive feature for object detection. The compressive feature is similar with the Haar-like feature. Actually, it can be regarded as the sparse Haar-like features, which are generated by compressively sensing in a large set of Haar-like features with a sparse measurement matrix. The compressive features preserve almost all the information of the original image. During object detection, the sparse measurement matrix can be computed only once off-line. The integral image can also be applied. Thus the computational speed of the compressive feature realizes a distinctive rise.

HOG is a feature descriptor which is used to represent the gradient orientation and intensity of an image. The basic idea behind HOG is that the object’s local appearance and shape within an image can be well described by the distribution of local intensity gradients or the oriented intensity of the edge. To construct HOG descriptors, the sliding windows are firstly divided into small spatial connected regions (cells). For each cell, HOG directions or edge orientations are calculated using all pixels within the cell. These cells are merged together into some larger connected regions (blocks). The local histograms can be contrast-normalized by accumulating a measure of local histograms of all cells within the block. Finally, these normalized histograms are combined as the descriptor.

Pixel comparisons are very simple visual features for object tracking. The pixel comparisons are randomly determined only once off-line and stay fixed online. For better robustness to shift and image noise, the image patch is firstly smoothed. The predefined pixel comparisons are
performed in the smoothed image patch. The results can be concatenated into a binary feature. The pixel comparisons are very similar with BRIEF, ORB, and BRISK features. In object detection, pixel comparisons are predesigned, which are permuted and split into some large set of comparisons. Every set can be linked together into a feature.

After passing through above classifiers, some negative samples may be misclassified as positive. The further processing based on the online model and the nearest neighbor can help to reduce the error rate of the detector.

NN classifier based on the online model can be used to classify the sliding window. The online model is a collection of positive and negative samples, which represents the object and its surrounding. These samples in the online model are some normalized image patches cropped from previous images. Image patches cropped from the remaining sliding windows are firstly normalized into the same size with the samples in the online model. Then the NCC is used to measure the similarities between the samples in the online model and these image patches. For each image patch, the positive nearest neighbor for the first % of the positive patches in the online model \( M \) is defined as:

\[
S_{\text{pos}} = S_{\text{pos}}(p, M) = \max_{p_i \in M} S(p, p_i^+) \]

Similarly, the negative nearest neighbor is defined as:

\[
S_{\text{neg}} = S_{\text{neg}}(p, M) \]

The relative similarity \( S' = \frac{S_{\text{pos}}}{S_{\text{pos}} + S_{\text{neg}}} \) and the conservative similarity \( S^c = \frac{S_{\text{pos}}}{S_{\text{pos}} + S_{\text{neg}}} \) are used to represent the resembling degrees between an image patch and samples in the online model and classify the image patch.

**Learning modules**

Learning modules are used to initialize the detectors in the first frame and update the classifiers online. The output results of the tracker and the detector are integrated by the learning modules to generate the training samples and to update the classifiers. The spatiotemporal properties of a video can help to generate more accurate training samples. The image patches cropped from the sliding windows in an image represent unlabeled samples, containing the spatial structure. The tracker can generate a tracking trajectory which has the temporal structure. In general, the image patches which are close to the tracked trajectory are positive samples (temporal constraints), and the patches in surrounding of a trajectory are negative samples (spatial constraints), where the P-N learning strategy can exploit the spatiotemporal structure information of the data to train more robust and discriminative detectors. The P-constraints (P-experts) and the N-constraints (N-experts) represent the temporal constraints and the spatial constraints, respectively. The number of negative samples is much larger than positive samples. To equalize the ratio of positive and negative samples, all positive samples are added to the training set without any restriction, while only partial negative samples, which is difficult to be distinguished from the tracked object (i.e. image patches cropped from the same class of the tracked object), are added to the training set. Here we introduce several popular learning methods to train binary classifiers that can combine with the features introduced in the last subsection to construct different detectors.

**Boosting learning** as a general learning framework is widely applied in detection, recognition, and tracking tasks, obtaining impressive performance and efficiently improving the accuracy of the given learning algorithm. Here we briefly introduce the boosting approach that converts weak classifiers to a strong classifier by selecting discriminative features. A weak classifier only needs to label samples better than random speculation. Each weak classifier, corresponding to a feature and being trained by a given learning algorithm, generates a hypothesis \( h^{\text{weak}} \). The set of \( M \) weak classifiers with hypothesis is represented by \( H^{\text{weak}} = \{h^{\text{weak}}_1, \ldots, h^{\text{weak}}_M\} \). In general, the number of weak classifiers is large. In order to cut down computing time, \( N (N < M) \) selectors are performed to choose a subset of weak classifiers to construct a strong classifier. Because each weak classifier corresponds to a feature, the selectors can select a subset of \( M \) features from the whole feature set. A strong classifier is computed by a linear combination of \( N \) selected weak classifiers.

**Multiple instance learning (MIL)** is a discriminative learning paradigm which can handle problems with incomplete comprehension on labels of the training samples. Most of them require a training data set \( \{(x_1, y_1), \ldots, (x_n, y_n)\} \) where \( x_i \) is an instance (such as a feature vector), and \( y_i \) represents a label. The training data of MIL are a set of labeled bags, and each contains multiple instances. The form of the training data is \( \{(X_1, y_1), \ldots, (X_m, y_m)\} \), where \( X_i \) a bag of instances \( \{x_{i1}, \ldots, x_{im}\} \) and \( y_i \) is a bag of labels. \( y_i \) is defined as \( \max(y_j), j = 1, \ldots, m \). Thus a bag \( y_i \) is a positive label if at least one instance is positive, while a negative one if all instances are negative. In PTAD, we make use of the MIL learning algorithm proposed by Babenko et al. to train a classifier for object detection.

**Random Forests (Ferns) (RFs)** is a strong classifier which is constructed using an ensemble randomized decision trees (weak classifiers). These trees are constructed and tested independently from each other, and their training and testing procedures can be performed in parallel. Each internal node of a tree defines a test which can split the input data into the left child tree or the right one. The input data finally reaches a leaf node of a tree. The leaf nodes of each tree contain the posterior distribution of class labels estimated based on the training set. In a forest, a tree \( i \) can be represented as \( f_i(x; \theta_i) \), where \( x \) is a sample to be classified, and \( \theta_i \) represents the stochastic parameters of the tree. A forest can be shown as \( \{f_1, \ldots, f_N\} \), where \( N \) is the number of the trees. The probability of predicting class \( k \) for a sample can be defined as \( p(k|x) = \frac{1}{N} \sum_{i=1}^{N} p_i(k|x) \), where
$p_r(k|x)$ is the posterior distribution of class labels of the leaf node of the $i$th tree where $x$ reached. Random Ferns can be regarded as special random forests, which are proposed to increase the speed of the general random forests. Random Ferns are non-hierarchical structures, and each fern defines an ordered set of binary tests. For random ferns, the results of the binary tests of all nodes in a fern can be combined to produce a binary code as a leaf node. All the possible leaf nodes contain the estimated posterior probabilities of class labels. Different from random forests, each node in the fern is involved in generating these binary codes (leaf nodes). The probability of predicting the class of each sample is combined over ferns in the same way as random forests.

**Experiments**

The proposed PTAD framework can be used to realize fast and robust long-term object tracking by integrating different tracking and detection algorithms. We integrate several well-known tracking and detection algorithms into the PTAD framework, in order to validate its effectiveness. Tracking experiments are performed on the TLD data set. The KCF tracker and the MF tracker are selected as the trackers. The cascade classifiers are designed for object detection, where the variance filter and the NN classifier are the first node classifier and the last node classifier respectively. We design several middle node classifiers including MIL + HOG (MIL-HOG), Random Ferns + Pixel comparisons (RFs-PCs), Boosting + Haar-like feature (Boosting-HL), and MIL + Compressive feature (MIL-CT). As a result, eight different object tracking algorithms are constructed by combining four different detectors and two trackers. We tested all the tracking algorithms on the TLD data set, and all the experiments were performed on a PC with 2.4 GHz i7-4700MQ CPU and 8 GB RAM.

**Experimental setup**

The TLD data set contains 10 image sequences: David, Jumping, Pedestrian1, Pedestrian2, Pedestrian3, Car, Motocross, Volkswagen, Carchase, and Panda, which confront most of the challenges in object tracking (e.g., camera moving, partial or full occlusion, pose change, illumination change, scale change, and similar objects). Some of the sequences are long enough to be appropriate to test our algorithms, which have been manually annotated. Readers can refer to Kalal et al. for the detailed introduction of the property of each sequence.

PTAD is implemented in C++. In the experiments, the tracked object is manually chosen in the first frame to initialize the tracker and the detector. PTAD then tracks the defined object up to the end and record the tracked trajectory. The performance of different tracking algorithms is evaluated based on the recorded trajectories and the annotated ground truth using the predefined performance criterions including precision $P$, recall rate $R$, and F-measure $F$. A tracking result is considered to be a true positive if the overlap between the estimated bounding box and the ground truth is larger than 25%. $P$ is defined as that the number of true positives divided by the total number of responses that contain both true positives and false positives. $R$ is represented as the number of true positives divided by the number of frames that contain the tracked object. $F$ is defined as $2PR/(P + R)$.

In the experiments, all parameters of the tracking modules are fixed. The parameters of the KCF tracker and the MF tracker are the same as those in Kalal et al. and Henriques et al. In the work of Kalal et al., the tracking failure detection has been designed, which is directly used in our experiments. For the KCF tracker, the NCC method is employed to judge whether the tracker fails or not. The threshold for the NCC tracking failure detection method is set to be 0.65.

The detection modules consist of a set of the cascade classifiers. In the beginning, we make use of the two aforementioned approaches to build a number of sliding windows. For the HOG feature, an image pyramid with six levels is built by successively down-sampling and upsampling the original image using scales step as 1.2. Then the sliding windows are generated with shifts in all pyramid images: horizontal step = 10% of the initial width, vertical step = 10% of the initial height. For the Haar-like feature, the pixel comparisons and the compressive feature, the sliding windows are generated with different scales and shifts in the original image using the same construction method and parameters in TLD. The number of scales is also set to be 6. For the variance filter, the variance threshold is set to be 50% variance of the image patch cropped from the initial bounding box in the first frame. The NN classifier for online model uses the same parameters as those in TLD.

We exploit four different classifiers as the middle node classifiers to test our framework. In our experiments, few parameters for Boosting, MIL, Random Ferns learning modules needs to be set in particular. The initial parameters of the Boosting and MIL trackers in OpenCV are used as the parameters of Boosting and MIL learning algorithms. The parameters of Random Ferns are the same as those in Kalal et al. A fast HOG feature extraction method proposed in Felzenszwalb et al. is used in the classifiers of HOG + MIL. The Haar-like feature is extracted using OpenCV, and the feature dimension is set to be 650. The dimension of the compressive feature proposed in Zhang et al. is set to be 100. For Random Ferns + Pixel comparisons, 10 ferns locate in random ferns, and each one consists of 13 pixel comparisons with $2^{13} = 8192$ possible codes that contain the posterior probabilities of class labels. Here we do not give a detailed description of these parameters, and please refer to the papers mentioned above for more details.
Table 1. Performance evaluation of the selected trackers (KCF, KCF with tracking failure detection, and MF) measured by Precision/Recall/F-measure.a

| Sequences | Frames | KCF | KCF (failure detection) | MF |
|-----------|--------|-----|-------------------------|----|
| 1. David  | 761    | 1.00/1.00/1.00 | 1.00/0.03/0.06 | 1.00/1.00/1.00 |
| 2. Jumping| 313    | 0.38/0.38/0.38 | 1.00/0.11/0.20 | 1.00/0.11/0.20 |
| 3. Pedestrian1 | 140   | 0.68/0.68/0.68 | 1.00/0.27/0.43 | 0.48/0.09/0.16 |
| 4. Pedestrian2 | 338  | 1.00/0.21/0.35 | 1.00/0.12/0.21 | 1.00/0.12/0.21 |
| 5. Pedestrian3 | 184   | 0.73/0.33/0.45 | 0.91/0.33/0.49 | 1.00/0.33/0.50 |
| 6. Car    | 945    | 0.91/0.97/0.94 | 0.99/0.59/0.74 | 0.96/0.56/0.71 |
| 7. Motocross | 2665  | 0.00/0.01/0.00 | 1.00/0.00/0.00 | 1.00/0.00/0.00 |
| 8. Volkswagen | 8576  | 0.94/0.53/0.67 | 0.99/0.06/0.11 | 1.00/0.05/0.10 |
| 9. Carchase | 9928  | 0.24/0.04/0.07 | 1.00/0.02/0.04 | 1.00/0.01/0.03 |
| 10. Panda  | 3000   | 0.17/0.18/0.17 | 1.00/0.04/0.08 | 1.00/0.00/0.00 |
| Mean value | 0.49/0.28/0.33 | 0.99/0.09/0.13 | 0.99/0.07/0.11 |

KCF: kernelized correlation filter; MF: median flow; PTAD: parallel tracking and detection.

aThe KCF tracker with tracking failure detection and the MF tracker will be integrated into PTAD, and bold numbers indicate the best scores.

Experimental results and analysis

In the beginning, we test the performance of the selected trackers (KCF, KCF with tracking failure detection based on the NCC method and MF) on the TLD data set. The trackers are designed for short-term object tracking without any method to recover and reinitialize from a failure. In the experiments, the tracker would be terminated if it lost the object during the tracking period, and the remaining images would not be processed. Table 1 shows their performance evaluated by the criterions of Precision/Recall/F-measure, and bold numbers indicate the best scores between the KCF tracker with tracking failure detection and the MF tracker.

The last row represents the weighted average performance for all the image sequences (weighted by the number of frames in the sequence). The original KCF tracker can achieve a higher recall rate but a lower precision than that of the KCF tracker with tracking failure detection. The tracking failure detection based on the NCC method can help to improve the precision performance for the KCF tracker, and the performance of the tracker is close to the MF tracker. Neither of the two trackers achieves the best performance for all the image sequences. Some of the proposed trackers could only adapt to some specific scenarios and failed achieving the best performance in practical situations. So it is significant to provide an open interface to integrate different trackers into PTAD for long-term object tracking. Next, we will integrate the two trackers with several detectors into PTAD and validate the effectiveness of our framework using the TLD data set.

For PTAD, the assumption that the appearance of the object varies slowly among several frames is important for long-term object tracking. Thus the tracker would not often lose the object in a short period. And the detector does not need to reinitialize and recover the tracker and update the classifiers in each frame of an image sequence. Thus we propose the PTAD framework in which the tracker and the detector can be processed independently and separately respectively. However, the appearance of the object and the scene may change relatively quickly in some practical applications. What’s more, there are a few TFs in each segment being unprocessed by the detector. It is necessary to make several experiments to evaluate the impact of the assumption and the number of the TFs ($\Delta K$) on the performance of our PTAD framework.

In the experiments, Motocross, Volkswagen, and Carchase are selected as the tracking datasets. The three sequences in the TLD dataset are long enough and contain all the typical challenges for long-term tracking. The tracked object in Motocross moves very fast, and its appearance changes relatively quickly. The appearance of the tracked object in Volkswagen changes slowly and gradually. In Carchase, the appearance of the object changes more quickly than that of the object in Volkswagen and more slowly than that of the object in Motocross. We select the KCF tracker as the tracker of PTAD, and the detector consists of three cascaded classifiers: the patch variance filter, RFs-PCs and the NN classifier. In order to better explain the experimental results, the number of the TFs ($\Delta K$) in each segment is fixed in each experiment. We set $\Delta K$ to be 0, 5, 10, 15, and 30, respectively. $\Delta K = 0$ represents that the detection and tracking modules are processed sequentially like the TLD framework.

The performance of the tracking algorithms, which is evaluated by Precision/Recall/F-measure, is shown in Table 2. Comparing with Table 1, we can find that all the

Table 2. Performance evaluation of the tracking algorithm based on PTAD measured by Precision/Recall/F-measure.a

| $\Delta K$ | 7. Motocross | 8. Volkswagen | 9. Carchase |
|-----------|--------------|--------------|------------|
| 0         | 0.94/0.63/0.75 | 1.00/0.94/0.97 | 0.82/0.57/0.67 |
| 5         | 0.92/0.42/0.57 | 1.00/0.93/0.96 | 0.96/0.53/0.69 |
| 10        | 0.77/0.36/0.49 | 1.00/0.93/0.96 | 0.87/0.53/0.66 |
| 15        | 0.71/0.15/0.25 | 0.99/0.92/0.96 | 0.79/0.48/0.59 |
| 30        | 0.24/0.03/0.05 | 0.85/0.57/0.68 | 0.72/0.45/0.55 |

PTAD: parallel tracking and detection; TF: tracking frame.

a$\Delta K$ represents the number of the TFs in each segment, which is fixed in each experiment. Bold numbers indicate the best scores.
performance using PTAD is better for the three image sequences. The combination of the detection and the tracking can improve the performance of long-term object tracking. Table 2 shows the recall rate of the tracking algorithm declines with the increasing of the number of the TFs in each segment. The recall rate has the fastest and the slowest declining for Motocross and Volkswagen respectively. The more the image sequence satisfies the assumption, the slower decline with the increasing of the number of the TFs the recall rate will have. From Table 2, it can also be found that the precision of the tracker remains relatively stable when the number of the TFs is relatively smaller. When the assumption is well met, the tracking algorithm based on PTAD can obtain the approximate performance as that achieved by combining tracking with detection sequentially like the TLD algorithm. Moreover, in order to achieve good tracking performance, we should improve the detector’s effectiveness and reduce the number of the TFs appropriate number of the TFs seems to be trivial and may be changed, which would affect the performance of the tracking algorithms. In the experiments, confirming the appropriate number of the TFs may be important in order to guarantee the eight trackers to achieve the best performance. We only set the frame rates of the trackers to be about 30 FPS, and the detectors are adjusted as fast as possible. Therefore, the number of the TFs in each segment is elastic. In practical applications, users can adjust the frame rates of the tracker and the detector to meet their requirements. Furthermore, users can exploit the thread synchronization mechanism to perform the tracking and detection tasks sequentially like the TLD framework. In the repeated experiments with the same experimental setup, the computational resources of the tracker and the detector may be different. At the same time, the tracking and detection algorithms are performed in parallel. The keyframes and the number of the TFs in each segment are also different in the repeated experiments, thus the experimental results are not exactly the same. In order to perform a fair comparison with other tracking algorithms, each experiment is repeated five times and the final experimental results come from the average of the five results.

The experimental results are shown in Table 3, and bold numbers indicate the best scores. The last row represents

| Sequences      | KCF + RFs-PCs | MF + RFs-PCs | KCF + Boosting-HL | MF + Boosting-HL |
|----------------|---------------|--------------|-------------------|-----------------|
| 1. David       | 0.95/0.74/0.83| 1.00/0.99/0.99| 0.98/0.46/0.63    | 0.98/0.95/0.96  |
| 2. Jumping     | 0.98/0.81/0.89| 0.98/0.86/0.92| 0.98/0.65/0.79    | 0.97/0.38/0.55  |
| 3. Pedestrian1 | 0.98/0.49/0.65| 0.75/0.57/0.65| 0.97/0.41/0.58    | 0.90/0.62/0.73  |
| 4. Pedestrian2 | 0.93/0.79/0.85| 0.95/0.80/0.87| 0.98/0.72/0.83    | 0.95/0.57/0.71  |
| 5. Pedestrian3 | 0.89/0.98/0.93| 0.85/0.95/0.90| 0.91/0.90/0.91    | 0.94/0.88/0.91  |
| 6. Car         | 0.96/0.99/0.98| 0.96/0.99/0.97| 0.98/0.91/0.94    | 0.93/0.92/0.92  |
| 7. Motocross   | 0.81/0.61/0.69| 0.82/0.61/0.70| 0.88/0.39/0.54    | 0.78/0.40/0.52  |
| 8. Volkswagen  | 0.98/0.95/0.96| 0.95/0.92/0.93| 0.97/0.72/0.83    | 0.85/0.59/0.70  |
| 9. Carchase    | 0.96/0.60/0.73| 0.93/0.49/0.64| 0.94/0.49/0.64    | 0.96/0.23/0.37  |
| 10. Panda      | 0.86/0.72/0.79| 0.86/0.45/0.58| 0.87/0.63/0.73    | 0.90/0.28/0.43  |
| Mean value     | 0.94/0.75/0.82| 0.92/0.68/0.76| 0.94/0.59/0.72    | 0.90/0.42/0.55  |

| Sequences      | KCF + MIL-CT  | MF + MIL-CT  | KCF + MIL-HOG     | MF + MIL-HOG     |
|----------------|---------------|--------------|-------------------|-----------------|
| 1. David       | 0.98/0.60/0.79| 0.96/0.95/0.95| 0.58/0.18/0.27    | 0.97/0.95/0.96  |
| 2. Jumping     | 0.95/0.65/0.77| 0.97/0.55/0.70| 0.96/0.36/0.52    | 0.95/0.22/0.36  |
| 3. Pedestrian1 | 0.91/0.32/0.47| 0.73/0.53/0.61| 0.85/0.28/0.42    | 0.54/0.34/0.42  |
| 4. Pedestrian2 | 0.92/0.58/0.71| 0.92/0.79/0.85| 0.76/0.58/0.66    | 0.70/0.51/0.59  |
| 5. Pedestrian3 | 0.88/0.55/0.68| 0.97/0.60/0.74| 0.72/0.50/0.59    | 0.51/0.54/0.53  |
| 6. Car         | 0.96/0.98/0.97| 0.94/0.97/0.95| 0.95/0.96/0.95    | 0.95/0.96/0.96  |
| 7. Motocross   | 0.60/0.38/0.46| 0.59/0.35/0.44| 0.59/0.34/0.43    | 0.70/0.45/0.54  |
| 8. Volkswagen  | 0.93/0.85/0.89| 0.65/0.55/0.59| 0.93/0.75/0.87    | 0.76/0.46/0.57  |
| 9. Carchase    | 0.90/0.51/0.64| 0.90/0.26/0.40| 0.93/0.54/0.68    | 0.95/0.30/0.45  |
| 10. Panda      | 0.84/0.60/0.70| 0.87/0.25/0.39| 0.72/0.58/0.64    | 0.82/0.21/0.33  |
| Mean value     | 0.88/0.63/0.72| 0.79/0.42/0.51| 0.86/0.59/0.70    | 0.84/0.40/0.52  |

RFs-PCs: Random Ferns; +: Pixel comparisons; Boosting-HL: Boosting + Haar-like feature; MIL-CT: Multiple instance learning + Compressive feature; KCF: Kernelized correlation filter; MF: median flow; PTAD: Parallel tracking and detection; MIL-HOG: Multiple instance learning + histogram of oriented gradient.

Bold numbers indicate the best scores.
weighted average results for all the image sequences. Obviously, the performance of the eight trackers in Table 3 is better than that of the three trackers in Table 1. Meanwhile, the trackers in Table 3 also achieve better performance compared with OnlineBoost, SemiBoost, BeyondSemiBoost, MIL, and coGD in Table 4 which comes from Kalal et al.1 The TLD tracker achieves the best performance in Table 4. This shows that the combination of the tracking and the detection can help to improve the performance for object tracking, especially for long-term object tracking. Furthermore, comparing Table 3 with Table 4, we observe that the trackers based on PTAD can achieve similar performance as the TLD tracker, such as KCF + RFs-PCs, MF + RFs-PCs, KCF + Boosting-HL, KCF + MIL-CT, and KCF + MIL-HOG. Especially for KCF + RFs-PCs, its performance is slightly better than that of TLD. In object tracking, the assumption conditions are usually satisfied. Thus the tracker and the detector can be performed in parallel, and the tracking algorithms based on PTAD can achieve comparable performance as TLD where the tracking, detection and learning are run sequentially.

Comparing with TLD, PTAD has its unique advantages: providing an open interface to integrate different object tracking and detection algorithms and ensuring the real-time performance. On the one hand, PTAD provides an open interface. As is known to us, none of the trackers and the detectors can achieve the best performance for all the image sequences. From Table 3, the MF + RFs-PCs tracker achieves good performance for David, Jumping, Pedestrian2 and Motocross, and the KCF + RFs-PCs tracker achieves good performance for other image sequences. Therefore, it is significant to provide an open interface to integrate different object tracking and detection algorithms to deal with different tracking scenarios. Users can integrate more trackers and detectors into our framework according to their practical requirements. The source code of PTAD framework can be downloaded from: https://github.com/nubot-nudt/ptad.

On the other hand, real-time performance can be obtained. The four detection components are run at

---

**Table 4. Performance evaluation of several trackers on TLD data set evaluated by Precision/Recall/F-measure.**

| Sequence | Frames   | OnlineBoost | SemiBoost | BeyondSemiBoost | MIL       | coGD    | TLD1.0 |
|----------|----------|-------------|-----------|----------------|-----------|---------|--------|
| 1. David | 761      | 0.41/0.29/0.34 | 0.35/0.35/0.35 | 0.32/0.24/0.28 | 0.15/0.15/0.15 | 1.00/1.00/1.00 | 1.00/1.00/1.00 |
| 2. Jumping | 313      | 0.47/0.05/0.09 | 0.25/0.13/0.17 | 0.17/0.14/0.15 | 1.00/1.00/1.00 | 1.00/0.99/1.00 | 1.00/1.00/1.00 |
| 3. Pedestrian1 | 140   | 0.61/0.14/0.23 | 0.48/0.33/0.39 | 0.29/0.10/0.15 | 0.69/0.69/0.69 | 1.00/1.00/1.00 | 1.00/1.00/1.00 |
| 4. Pedestrian2 | 338   | 0.77/0.12/0.21 | 0.85/0.71/0.77 | 1.00/0.02/0.04 | 0.10/0.12/0.11 | 0.72/0.92/0.81 | 0.89/0.92/0.91 |
| 5. Pedestrian3 | 184   | 1.00/0.33/0.49 | 0.41/0.33/0.36 | 0.92/0.46/0.62 | 0.69/0.81/0.75 | 0.85/1.00/0.92 | 0.99/1.00/0.99 |
| 6. Car | 945      | 0.94/0.59/0.73 | 1.00/0.67/0.80 | 0.99/0.56/0.72 | 0.23/0.25/0.24 | 0.95/0.96/0.96 | 0.92/0.97/0.94 |
| 7. Motocross | 2665 | 0.33/0.00/0.01 | 0.13/0.03/0.05 | 0.14/0.00/0.00 | 0.05/0.02/0.03 | 0.93/0.30/0.45 | 0.89/0.77/0.83 |
| 8. Volkswagen | 8576 | 0.39/0.02/0.04 | 0.04/0.04/0.04 | 0.02/0.01/0.01 | 0.42/0.04/0.07 | 0.79/0.06/0.11 | 0.80/0.96/0.87 |
| 9. Carchase | 9928   | 0.79/0.03/0.06 | 0.80/0.04/0.09 | 0.52/0.12/0.19 | 0.62/0.04/0.07 | 0.95/0.04/0.08 | 0.86/0.70/0.77 |
| 10. Panda | 3000    | 0.95/0.35/0.51 | 1.00/0.17/0.29 | 0.99/0.17/0.30 | 0.36/0.40/0.38 | 0.12/0.12/0.12 | 0.58/0.63/0.60 |
| Mean value | 0.62/0.09/0.13 | 0.50/0.10/0.14 | 0.39/0.10/0.15 | 0.44/0.11/0.13 | 0.80/0.18/0.22 | 0.82/0.81/0.81 |

*The results come from Kalal et al.1 Bold numbers indicate the best scores.

**Table 5. The number of the keyframes, the average number of the TFs in each segment (\(D_K\)), and the average processing time of each keyframe for the four detectors integrated into PTAD.**

| Sequences | Frames | RFs-PCs | Boosting-HL | MIL-CT | MIL-HOG |
|-----------|--------|---------|-------------|--------|---------|
| 1. David  | 761    | 361     | 120         | 223    | 64      |
| 2. Jumping | 313    | 269     | 32          | 36     | 12      |
| 3. Pedestrian1 | 140  | 111     | 15          | 20     | 7       |
| 4. Pedestrian2 | 338   | 205     | 90          | 142    | 48      |
| 5. Pedestrian3 | 184   | 79      | 22          | 38     | 12      |
| 6. Car   | 945    | 480     | 177         | 172    | 102     |
| 7. Motocross | 2665 | 1821    | 476         | 720    | 492     |
| 8. Volkswagen | 8576 | 4480    | 311         | 385    | 90      |
| 9. Carchase | 9928 | 693     | 822         | 952    | 628     |
| 10. Panda | 3000   | 2013    | 450         | 416    | 358     |
| Keyframe number | 26850 | 16341   | 2421        | 3104   | 1813    |
| Mean \(D_K\) | 0.64  | 10.11   | 7.65        | 13.81  |         |
| Mean time (ms) | 49.2  | 333     | 259         | 444    |         |

RFs-PCs: Random Ferns + Pixel comparisons; Boosting-HL: Boosting + Haar-like feature; MIL-CT: Multiple instance learning + Compressive feature; PTAD: parallel tracking and detection; MIL-HOG: Multiple instance learning + Histogram of oriented gradient.
different speeds. When the same detector combines with the KCF tracker or the MF tracker, the number of keyframes processed by the detector is approximate. The numbers of the keyframes ($k_f$) for the four detectors are recorded in the experiments of KCF + RFs-PCs, KCF + Boosting-HL, KCF + MIL-CT, and KCF + MIL-HOG. The average number of the TFs in each segment (mean $\Delta K$) can be represented as $\frac{frames\_nums - k_f\_nums}{k_f\_nums}$, where $frames\_nums$ is the total number of images of all the sequences. The average processing time of each keyframe is obtained as $\frac{frames\_nums \times 30}{k_f\_nums}$ (ms). Table 5 shows the number of keyframes, the average number of the TFs in each segment (mean $\Delta K$), and the average processing time of each keyframe. The processing time of the RFs-PCs detector is about 49 ms per frame. The processing time of other detectors are more than 250 ms per frame, and it is about 444 ms for the MIL-HOG detector. If these detectors are used in the TLD framework, the frame rate of the tracking algorithm would be low. In PTAD, the tracker is performed as the front end, and the detector is run as the back end. Thus the frame rate of the tracking algorithm is the same as that of the front end. Here the frame rates of the eight trackers are 30 FPS. In practical applications, users can adjust the frame rate of the front end to satisfy real-time requirements. Finally, we find that KCF + RFs-PCs and MF + RFs-PCs achieve the best performance. The quantity of the TFs ($\Delta K$) has an impact on the trackers’ performance. Therefore, we should improve the efficiency of the detectors as high as possible in object tracking.

Choosing appropriate trackers and detectors

Trackers may be just for some specific scenarios. It’s almost impossible for a tracker to achieve the best performance for all image sequences. However, in order to deal with unknown tracking objects, we should select real-time, accurate, and robust trackers which can adapt to different objects and more tracking scenarios. Similarly, we should also choose more robust and accurate detectors. In addition, the real-time performance of object detectors should be considered according to the appearance variations of the tracking object. When the object appearance changes fast, the assumption that the appearance of the object varies slowly among several frames is not met. In this situation, we should select the detector with better real-time performance. On the contrary, we can choose some slower detectors. According to the above principles, we can choose some appropriate trackers and detectors from state-of-the-art object tracking and detection algorithms for the PTAD framework.

What is more, we can train deep learning detectors for known objects off-line, and these models do not need to be updated online. Thus, the PTAD usually contains two modules: tracking and detection. We make use of a pretrained deep learning Single Shot Detector (SSD) model\textsuperscript{45} to detect the tracked object. About the real-time performance, it takes about 77 ms to process a frame of perspective image with GeForce GTX960M GPU. We perform the tracking experiment on the UAV123 data set.\textsuperscript{46} The tracked object is a car. In the Table 6, it can be seen that the detector trained off-line can be used to track the known object.

Table 6. Performance evaluation of several trackers on UAV123 data set evaluated by Precision/Recall/F-measure.\textsuperscript{a}

| Sequences   | KCF + SSD | KCF + RFs-PCs |
|-------------|-----------|---------------|
| Car5 (UAV123) | 1.0/1.0/1.0 | 0.93/0.93/0.93 |
| Car6 (UAV123) | 0.89/0.89/0.89 | 0.81/0.81/0.81 |
| Mean value  | 0.95/0.95/0.95 | 0.87/0.87/0.87 |

SSD: Single Shot Detector; KCF: kernelized correlation filter; RFs-PCs: Random Ferns + Pixel comparisons.

\textsuperscript{a}Bold numbers indicate the best scores.

Ground moving object tracking

In recent years, vision-based ground moving object tracking has become a very active field of research for macro aerial vehicles (MAVs). We employ the PTAD tracker (KCF + RFs-PCs) to estimate the location of the ground

\textsuperscript{14}International Journal of Advanced Robotic Systems

Figure 4. The MAV platform. MAV: macro aerial vehicle.
moving object between the consecutive frames. As shown in Figure 4, the DJI Matrice 100 is selected as the experimental platform. All experiments are performed on an Intel’s NUC computers with 3.1 GHz i7-5557U CPU and 8 GB RAM. A downward looking USB camera is mounted on the bottom of the MAV.

We perform object tracking experiments in a basketball court and a garden. The tracking results are demonstrated in Figure 5. It can be clearly seen that the PTAD tracker is robust to illumination variations, scale changes, out-of-view, and cluttered environments. The tracked object almost goes out of view of the camera in the penultimate frame. When the object appears again in the last frame, the detection module can detect the object and recover the failed tracker. In practical applications, the processing speed of the PTAD tracker can be higher than 30 FPS, so real-time requirements can be met.

Conclusions
In this article, we propose a framework of PTAD for unknown object tracking, especially for long-term unknown object tracking. The proposed framework has several advantages. Firstly, PTAD can combine tracking and detection and make up their own deficiencies. A tracker can provide weakly labeled training samples for a detector, and a detector can recover a tracker when drifts are large or the tracker fails. Secondly, the tracking and detection are run in parallel like PTAM. The tracker requires little computing resources and is performed as the front end. The detector is run as the back end. Thus our framework can achieve real-time performance in practical applications. Finally, PTAD provides an open interface to integrate and compare various tracking and detection algorithms. The source code of PTAD framework can be downloaded from: https://github.com/nubot-nudt/ptad. In this article, two trackers and four detectors are integrated into PTAD, and their performance is evaluated on the TLD data set. The experimental results demonstrate that our framework is effective for long-term real-time object tracking. Our future work will focus on two aspects. Firstly, we will extend our framework to GPU and improve the real-time performance of the detection component. Secondly, we will develop more robust trackers and detectors for PTAD.

Acknowledgement
The authors would like to thank the anonymous reviewers, whose insightful comments can greatly improve the quality of this article.

Declaration of conflicting interests
The author(s) declared no potential conflict of interest with respect to the research, authorship, and/or publication of this article.

Funding
The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This research is supported by the National Natural Science Foundation of China (grant nos U1813205 and 61773393).

ORCID iD
Dan Xiong  https://orcid.org/0000-0001-7441-8336

References
1. Kalal Z, Mikolajczyk K, and Matas J. Tracking-learning-detection. IEEE Trans Pattern Anal Mach Intell 2012; 34(7): 1409–1422.
2. Klein G and Murray D. Parallel tracking and mapping on a camera phone. In: Proceedings of the 8th IEEE international symposium on mixed and augmented reality, Orlando, Florida, USA, 19–22 October 2009, pp. 83–86.
3. Li X, Hu W, Shen C, et al. A survey of appearance models in visual object tracking. ACM Trans Intell Syst Technol 2013; 4(4): 58:1–58:48.
4. Wu Y, Lim J, and Yang MH. Online object tracking: a benchmark. In: Proceedings of the IEEE conference on computer vision and pattern recognition, Portland, Oregon, USA, 23–28 June 2013, pp. 2411–2418.
5. Liu S and Feng Y. Real-time fast moving object tracking in severely degraded videos captured by unmanned aerial vehicles. *Int J Adv Robot Syst* 2018; 15(1): 1729881418759108.

6. Yang Y and Cao Q. A fast feature points-based object tracking method for robot grasp. *Int J Adv Robot Syst* 2013; 10(3): 5595174.

7. Ma C, Huang J, Yang X, et al. Hierarchical convolutional features for visual tracking. In: *Proceedings of the IEEE international conference on computer vision*, Santiago, Chile, 7–13 December 2015, pp. 3074–3082.

8. Valmadre J, Bertinetto L, Henriques J, et al. End-to-end representation learning for correlation filter based tracking. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, Honolulu, HI, USA, 21–26 July 2017, pp. 5000–5008.

9. Wang H, Suter D, Schindler K, et al. Adaptive object tracking based on an effective appearance filter. *IEEE Trans Pattern Anal Mach Intell* 2007; 29(9): 1661–1667.

10. Mei X and Ling H. Robust visual tracking and vehicle classification via sparse representation. *IEEE Trans Pattern Anal Mach Intell* 2011; 33(11): 2259–2272.

11. Ma C, Yang X, Zhang C, et al. Long-term correlation tracking. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, Boston, MA, USA, 7–12 June 2015, pp. 5388–5396.

12. Bertinetto L, Valmadre J, Golodetz S, et al. Staple: complementary learners for real-time tracking. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, Las Vegas, NV, USA, 27–30 June 2016, pp. 1401–1409.

13. Zhou X, Liu X, Yang C, et al. Multi-channel features spatio-temporal context learning for visual tracking. *IEEE Access* 2017; 5: 12856–12864.

14. Lukežič A, Čehovin Zajc L, and Kristan M. Deformable parts correlation filters for robust visual tracking. *IEEE Trans Cybern* 2018; 48(6): 1849–1861.

15. Held D, Thrun S, and Savarese S. Learning to track at 100 fps with deep regression networks. In: *Proceedings of the 14th European conference computer vision*, Amsterdam, The Netherlands, 11–14 October 2016, pp. 749–765.

16. Bertinetto L, Valmadre J, Henriques JF, et al. Fully-convolutional Siamese networks for object tracking. In: *Proceedings of the 14th European conference computer vision*, Amsterdam, The Netherlands, 11–14 October 2016, pp. 850–865.

17. Li B, Yan J, Wu W, et al. High performance visual tracking with Siamese region proposal network. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, Salt Lake City, UT, USA, 18–22 June 2018, pp. 8971–8980.

18. He A, Luo C, Tian X, et al. Towards a better match in Siamese network based visual object tracker. In: *Proceedings of the 15th European conference computer vision*, Munich, Germany, 8–14 September 2018, pp. 132–147.

19. Wang Q, Zhang L, Bertinetto L, et al. Fast online object tracking and segmentation: a unifying approach. *CoRR* 2018.

20. Yang T and Chan AB. Learning dynamic memory networks for object tracking. In: *Proceedings of the 15th European conference computer vision*, Munich, Germany, 8–14 September 2018, pp. 153–169.

21. Prasad DK. Survey of the problem of object detection in real images. *Int J Image Proc* 2012; 6(6): 441–466.

22. Zhu X and Goldberg A. Introduction to Semi-Supervised Learning. *Synthesis lectures on artificial intelligence and machine learning*, 2009; 3(1): 1–130.

23. Grabner H, Leistner C, and Bischof H. Semi-supervised online boosting for robust tracking. In: *Proceedings of the 10th European conference computer vision*, Marseille, France, 12–18 October 2008, pp. 234–247.

24. Tian M, Zhang W, and Liu F. On-line ensemble SVM for robust object tracking. In: *Proceedings of the 8th Asian conference on computer vision*, Tokyo, Japan, 18–22 November 2007, pp. 355–364.

25. Ding S, Xi X, Liu Z, et al. A novel manifold regularized online semi-supervised learning model. *Cognit Comput* 2018; 10(1): 49–61.

26. Chen C, Sun B, Hu X, et al. Online semi-supervised learning with multiple regularization terms. *IEEE Access* 2019; 7: 68479–68494.

27. Liu Y, Xu Z, and Li C. Online semi-supervised support vector machine. *Inform Sci* 2018; 439–440: 125–141.

28. Zeisl B, Leistner C, Saffari A, et al. On-line semi-supervised multiple-instance boosting. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, San Francisco, CA, USA, 13–18 June 2010, pp. 1879–1879.

29. Li G, Huang Q, Qin L, et al. SSOCBT: a robust semisupervised online CovBoost tracker that uses samples differently. *IEEE Trans Circuit Syst Video Technol* 2013; 23(4): 695–709.

30. Yang F, Lu H, and Yang MH. Robust visual tracking via multiple kernel boosting with robustness constraints. *IEEE Trans Circuit Syst Video Technol* 2014; 24(2): 242–254.

31. Mesghi K, Mirzaei MS, Oba S, et al. Efficient asymmetric co-tracking using uncertainty sampling. In: *Proceedings of the IEEE international conference on signal and image processing applications*, Kuching, Malaysia, 12–14 September 2017, pp. 241–246.

32. Raykar VC, Krishnapuram B, and Yu S. Designing efficient cascaded classifiers: tradeoff between accuracy and cost. In: *Proceedings of the 16th ACM SIGKDD international conference on knowledge discovery and data mining*, Washington, DC, USA, 25–28 July 2010, pp. 853–860.

33. Van Den Oord A, Kalchbrenner N, and Kavukcuoglu K. Pixel recurrent neural networks. In: *Proceedings of the 33rd international conference on international conference on machine learning*, New York City, NY, USA, 19–24 June 2016, pp. 1747–1756.

34. Mao X, Li Q, Xie H, et al. Least squares generative adversarial networks. In: *Proceedings of the IEEE international conference on computer vision*, Venice, Italy, 22–29 October 2017, pp. 2813–2821.
35. Chang C, Chen T, and Chung P. Semi-supervised learning using generative adversarial networks. In: *Proceedings of the IEEE symposium series on computational intelligence*, Bangalore, India, 18–21 November 2018, pp. 892–896.

36. Rasmus A, Valpola H, Honkala M, et al. Semi-supervised learning with ladder networks. In: *Proceedings of the 28th international conference on neural information processing systems*, Montreal, Canada, 7–12 December 2015, pp. 3546–3554.

37. Maaløe L, Sønderby CK, Sønderby SK, et al. Auxiliary deep generative models. In: *Proceedings of the 33rd international conference on international conference on machine learning*, New York, NY, USA, 19–24 June 2016, pp. 1445–1453.

38. Redmon J and Farhadi A. Yolov3: An incremental improvement. *CoRR* 2018.

39. Hare S, Golodetz S, Saffari A, et al. Struck: structured output tracking with kernels. *IEEE Trans Pattern Anal Mach Intell* 2016; 38(10): 2096–2109.

40. Henriques JF, Caseiro R, Martins P, et al. High-speed tracking with kernelized correlation filters. *IEEE Trans Pattern Anal Mach Intell* 2015; 37(3): 583–596.

41. Babenko B, Yang MH, and Belongie S. Robust object tracking with online multiple instance learning. *IEEE Trans Pattern Anal Mach Intell* 2011; 33(8): 1619–1632.

42. Zhang K, Zhang L, and Yang MH. Fast compressive tracking. *IEEE Trans Pattern Anal Mach Intell* 2014; 36(10): 2002–2015.

43. Felzenszwalb PF, Girshick RB, McAllester D, et al. Object detection with discriminatively trained part-based models. *IEEE Trans Pattern Anal Mach Intell* 2010; 32(9): 1627–1645.

44. Bosch A, Zisserman A, and Munoz X. Image classification using random forests and ferns. In: *Proceedings of the 11th IEEE international conference on computer vision*, Rio de Janeiro, Brazil, 14–20 October 2007.

45. Liu W, Anguelov D, Erhan D, et al. Ssd: Single shot multibox detector. In: *Proceedings of the 14th European conference computer vision*, Amsterdam, The Netherlands, 11–14 October 2016, pp. 21–37.

46. Mueller M, Smith N, and Ghanem B. A benchmark and simulator for UAV tracking. *Far East J Math Sci* 2016; 2(2): 445–461.