Journal of Big Data

RESEARCH Open Access

Deep features fusion for KCF-based moving object tracking

Devira Anggi Maharani[1*], Carmadi Machbub[1], Lenni Yulianti[1] and Pranoto Hidaya Rusmin[1]

*Correspondence:
deviraanggi@students.itb.ac.id

[1] School of Electrical Engineering and Informatics, Institut Teknologi Bandung, Bandung, Indonesia

Abstract

Real-time object tracking and occlusion handling are critical research areas in computer vision and machine learning. Developing an efficient and accurate object-tracking method that can operate in real-time while handling occlusion is essential for various applications, including surveillance, autonomous driving, and robotics. However, relying solely on a single hand-crafted feature results in less robust tracking. As a hand-crafted feature extraction technique, HOG effectively detects edges and contours, which is essential in localizing objects in images. However, it does not capture fine details in object appearance and is sensitive to changes in lighting conditions. On the other hand, the grayscale feature has computational efficiency and robustness to changes in lighting conditions. The deep feature can extract features that express the image in more detail and discriminate between different objects. By fusing different features, the tracking method can overcome the limitations of individual features and capture a complete representation of the object. The deep features can be generated with transfer learning networks. However, selecting the right network is difficult, even in real-time applications. This study integrated the deep feature architecture and hand-crafted features HOG and grayscale in the KCF method to solve this problem. The object images were obtained through at least three convolution blocks of transfer learning architecture, such as Xception, DenseNet, VGG16, and MobileNet. Once the deep feature was extracted, the HOG and grayscale features were computed and combined into a single stack. In the KCF method, the stacked features acquired the actual object location by conveying a maximum response. The result shows that this proposed method, especially in the combination of Xception, grayscale, and HOG features, can be implemented in real-time applications with a small center location error.

Keywords: HOG, Deep features, KCF, Xception, Real-time, Transfer learning

Introduction

In recent years, significant progress has been made in visual tracking for robotics applications [1, 2] and surveillance systems [3, 4]. The main focus has been on real-time implementation and addressing challenges related to occlusion. Despite numerous attempts to develop reliable tracking mechanisms, this area still has ongoing work.

Since the seminal work by Bolme et al. [5], correlation-based filters have become highly popular within the visual tracking community. These filter-based tracking methods are

well-known for their computational efficiency, making them particularly valuable for real-time applications [6, 7]. Bolme et al. [5] introduced the minimum output sum of squared error (MOSSE), the first object-tracking method to use correlation filtering. To determine the target's position, the Fast Fourier Transform (FFT) technique, and grayscale features represent the target image. Through FFT analysis, the tracker can identify the target's position by locating the maximum response value, which is considered the target position. However, the grayscale may not capture all the object appearance data for tracking [8]. Numerous methods have been developed since then to enhance it. Henriques [9] suggested utilizing HOG features instead of grayscale features in the KCF method, where this algorithm extracts features from the observed object and constructs a set of feature samples with cyclic shifts. It then trains a classifier to predict the target position using ridge regression with a kernel method. The KCF tracking method employs a circulant matrix for sampling, leading to decreased complexity and improved tracking speed [9–12]. Although the outstanding tracking performance of KCF under normal circumstances, it cannot produce a reliable performance when faced with obstacles such as occlusion [13].

Even though these methods have made significant progress, computer vision researchers are still working on addressing issues related to real-time implementation and occlusion handling. Creating a reliable and effective system to imitate the human visual process is crucial in computer vision. Inspired by previous studies, one factor in effectively tracking objects is extracting useful information to identify appropriate target characteristics from images. It helps avoid tracking errors that can accumulate in the KCF method and result in inaccuracies when determining the tracked object position. Several hand-crafted features, including HOG [10, 11], Harris corner detection [14], and Scale-Invariant Feature Transform (SIFT) [15], have been suggested in earlier studies. As a hand-crafted feature extraction technique, the HOG feature is extracted by calculating and counting the gradient direction histogram of the image's local region, which helps capture the target's contour and edges [16, 17]. It is essential in localizing objects in images. However, in practical application, it does not capture fine details in object appearance and is sensitive to changes in lighting conditions. On the other hand, the grayscale feature has computational efficiency and robustness to changes in lighting conditions. However, it may not capture all the discriminative information needed for accurate tracking.

Deep features can extract detailed information from images and distinguish between different objects. As of now, deep learning has been used for visual object tracking, yielding excellent results [11, 18–22]. In particular, transfer learning, which leverages the knowledge gained from a pre-trained model on a large dataset, enables the generation of deep features encompassing richer and more discriminative representations [18]. The deep features continuously learn more abstract deep characteristics. It can find the best feature representation from the target dataset and associate it with the original image, allowing for a more precise expression of the image and better distinguishing among various objects [19], and getting the essential image information [10]. The Hierarchical Convolutional Features tracker derives hierarchical convolutional features, which can extract deep features and use the multi-level correlation response maps to infer the target location. This tracker achieves 10–11 FPS, but when long-term occlusions occur, this

tracker fails to follow targets [23]. Furthermore, to obtain high tracking performance, Nam et al. proposed pre-train deep CNNs in multi-domains, with each domain corresponding to one training video sequence, and achieved good tracking performance with 1 FPS [24]. The Correlation Filter with anti-occlusion and multi-feature fusion has been proposed and achieves 5.89 FPS [6]. Even though deep model-based tracking methods can be applied to challenging situations, they still utilize a lot of hardware resources. Selecting a good network for real-time applications can be challenging as the depth of the network increases, resulting in higher-level feature abstractions that capture distinctive image characteristics. However, this limitation can be addressed by incorporating HOG and grayscale features from the extracted deep features. The use of such feature fusion has been widely applied in various domains, including feature fusion for improving image inpainting results [25, 26], high-resolution image reconstruction [27], and enhancing image quality [28]. It has also been employed with IoT devices for target-tracking purposes [29]. The fusion of features improves the overall performance and generates a more effective feature representation [30]. By leveraging the fusion of HOG and grayscale features from the extracted deep features, computational complexity, and resource requirements can be reduced while preserving crucial information. The tracking method can overcome the limitations of individual features and capture a complete representation of the object by fusing different features.

This paper proposed an approach to combine deep features with transfer learning architectures and hand-crafted features, namely HOG and grayscale. By doing so, this study aims to enhance the effectiveness of the KCF method and address concerns related to real-time single object tracking, as well as occlusion handling during specific durations. The primary contributions of this paper can be summarized as follows:

1. The present paper suggests an approach to improve the performance of the KCF tracking method in video tracking applications. Specifically, the proposed method combines deep feature architecture and hand-crafted features, including HOG and grayscale, to address two key challenges in tracking: real-time tracking and occlusion handling. The goal of this approach is to enhance the robustness of the KCF method under these conditions.
2. The present study employs transfer learning techniques to extract deep features from object images. Specifically, the transfer learning architectures Xception, DenseNet, VGG16, and MobileNet are utilized for this purpose. These techniques leverage the knowledge acquired from pre-trained models on a large dataset to enhance the feature extraction process. The object images are passed through a minimum of three convolution blocks belonging to the same transfer learning architecture to obtain deep features. Following the extraction of deep features, HOG and grayscale features are computed and integrated into a single stack. In the KCF method, the stacked features are intended to acquire the actual object location by conveying a maximum response. The intention of this combination is to create a more comprehensive and robust feature representation for the object being tracked. By incorporating deep and hand-crafted features, the proposed method seeks to improve the accuracy and robustness of the KCF tracking method under various tracking conditions, including real-time tracking and occlusion handling.

The remainder of this paper is organized as follows. "Materials and method" section provides an overview of transfer learning, a key aspect of the proposed method. "Our approach" section presents a detailed description of the proposed method, including integrating deep feature architecture and hand-crafted features such as HOG and grayscale. In "Results" section, three experimental datasets utilized in this study are introduced, and the experimental results obtained are analyzed and discussed. Finally, "Conclusion" section summarizes the major findings of this study and outlines future plans.

Materials and method

Xception

The Xception model was proposed in 2016 by François Chollet. This Xception architecture model outperforms VGG-16, ResNet50, ResNet101, ResNet152, and Inception-V3 on ImageNet. In its early version, the Xception model has its foundation, such as the original Inception [31] and the Inception-V3 [32]. The Inception method in a CNN is an intermediary step between ordinary convolution and depthwise separable convolution operations that are separated in-depth (followed by pointwise convolutions). The depthwise separable convolution is an Inception Module with many towers. Therefore, the observation findings suggest a deep CNN inspired by Inception. The inception module can be substituted with depthwise separable convolutions Xception, which performs slightly better than Inception V3 with ImageNet (a large image classification dataset). Since Xception architecture and Inception V3 have the same number of parameters, there is an improvement in the performance due to more effective model parameters. The convolution layer aims to analyze 3D filters using two spatial dimensions (width and height) and channel dimensions. Both the cross-channel and spatial correlations must be mapped by a single convolutional kernel. This Inception module tries to simplify and improve the process by breaking the procedure into a set of operations that will separately determine the cross-channel and spatial correlation. According to the core principle of Inception, it is not required to map cross-channel and spatial correlations concurrently. Exception's "Extreme Inception" architecture features a 36-layer convolution as the feature extraction network [33]. The Xception architecture has 14 modules and consists of a linear stack of depth-wise separable convolution layers with residual connections. These 14 modules are further divided into three groups, including entry flow (4 modules), middle flow (8 modules), and exit flow (optional and fully connected layers). The data is transferred through the entry flow, and the middle flow repeated eight times, and finally, the exit flow. Batch normalization is applied to all Convolution and SeparableConvolution layers.

In this research, we applied the third convolution module in the entry flow for this feature extraction. Figure 1 illustrates the multiple channel result on three Xception blocks. The results of the three Xception convolution blocks show a variation in each channel, which can provide a rich feature representation for the tracked object. By combining the channels from the three Xception convolution blocks, a diverse and rich feature representation can be generated for the tracked object. Figure 2 shows the input of the Xception module is (299, 299, 3) with the average layer in the last layer. The pre-trained model has already learned useful features from a large dataset, and these features can be

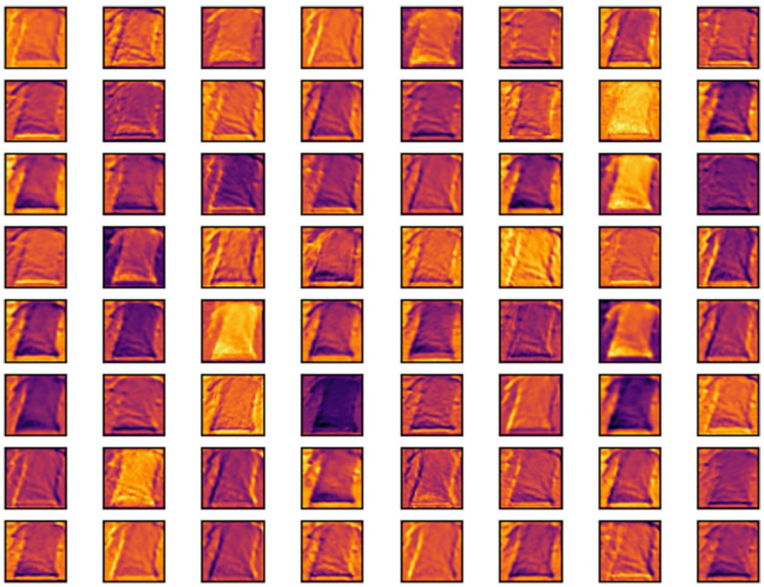

Fig. 1 Multiple channel results on three Xception convolution blocks

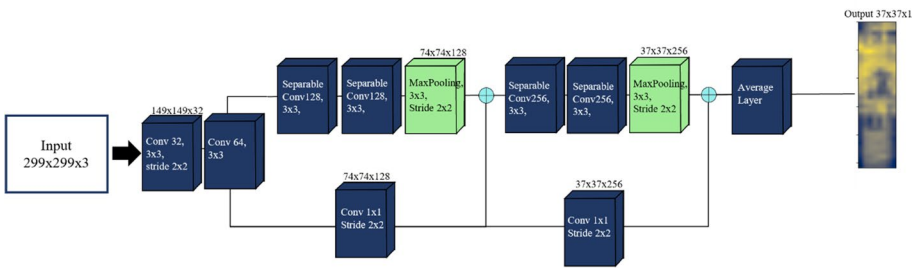

Fig. 2 Feature extraction with entry flow in Xception architecture

leveraged to improve the performance of the new model. As more deep feature layers are added, the capacity for effective feature extraction also improves gradually. Deep features are capable of acquiring diverse image feature descriptions through multiple convolution layers, each containing several convolutions that can generate distinct feature descriptions from the original image.

DenseNet

Dense Convolutional Network (DenseNet) network also aims to bring the issue of the vanishing gradient caused by the depth of the network [33, 34]. DenseNet establishes connections between every layer and all the other layers in a feed-forward model. Each layer receives inputs from all the previous layers and transmits its feature maps to all the following layers. Connections are established directly from any layer to all the following layers. To transfer information from the previous layers to the next layers, the feature maps are merged through concatenation at each layer. The need to learn repetitive information is eliminated, resulting in a significant reduction in the number of parameters required for the model. To enable down-sampling in DenseNet architecture, the

network is divided into multiple dense blocks that are densely connected. The layers located between these blocks are referred to as transition layers, which perform convolution and pooling operations. These transition layers contain a batch normalization layer and a 1×1 convolutional layer, followed by a 2×2 average pooling layer. DenseNet has the capability to have narrow layers. Every layer within its block has access to all the previous feature-maps, resulting in the utilization of the network's "collective knowledge". Following the last dense block, a global average pooling is carried out, after which a softmax classifier is appended.

In this research, we pruned the network of at least three convolutional blocks in the first dense block of DenseNet, as illustrated in Fig. 3. The input size of the image was set at (224, 224, 3). The first dense block in DenseNet comprised six layers that functioned together to process the input image. These layers included a batch normalization layer, a ReLU activation layer, two 3×3 convolutional layers with 64 filters, and additional batch normalization and ReLU activation layers. The output feature maps of each layer were concatenated with the input feature maps of the next layer, rather than being added together. This ensured that each layer had access to all of the features learned by the previous layers, resulting in a more comprehensive representation of the input data. Additionally, since the outgoing feature maps did not modify the incoming feature maps, this concatenation approach helped to reduce the number of parameters required in the network. In the final layer, we added an average layer and produced three channels.

MobileNet

Previous research [35] has successfully detected an object effectively using this MobileNet method. MobileNet is a neural network structure intended for use on mobile and embedded devices that have limited computing resources. Its primary feature is the use of depthwise separable convolutions, which involves dividing the convolution into two steps: depthwise convolution and pointwise convolution. In the depthwise convolution step, each input channel is filtered separately, creating a set of feature maps. The pointwise convolution step then combines these feature maps using a 1×1 convolution to produce the final output. This approach reduces the number of parameters and computations required compared to traditional convolutions.

In this research, the MobileNet network was pruned as illustrated in Fig. 4. The network was designed to handle input images with a size of (224, 224, 3) and it comprised several layers including the convolutional layer, batch normalization layer, ReLU activation layer, and Depthwise convolution layer. Specifically, four convolutional layers were

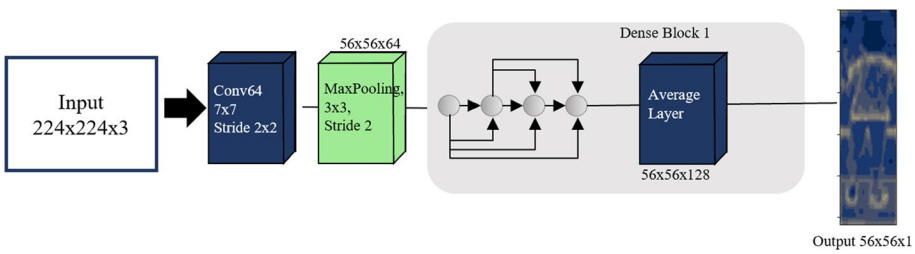

Fig. 3 Feature extraction with DenseNet

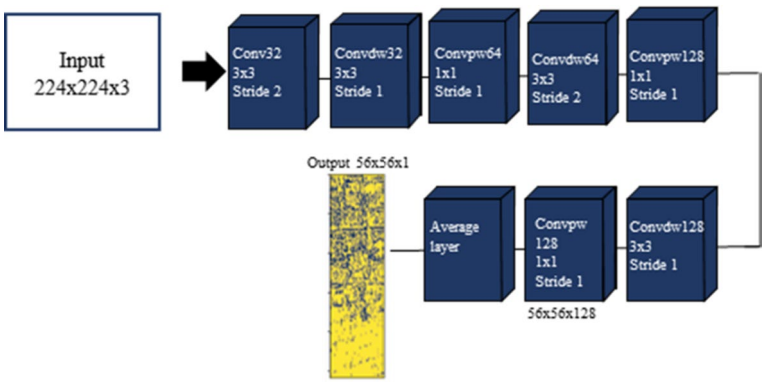

Fig. 4 Feature extraction with MobileNet

employed in the network. The last layer of the network was equipped with an average layer that calculates the average value of all channels in each pixel. This approach helped to capture a more representative feature of the input image, which can enhance the accuracy and reliability of the tracking process. The use of multiple convolutional layers helped to extract more complex and high-level features of the input image, while the average layer helped to obtain a single feature vector that summarized the entire input image. Furthermore, the pruning technique employed in this research helped to reduce the computational cost and the number of parameters required by the network, which was crucial for achieving real-time performance in video-tracking applications.

VGG16

The VGG16 was first proposed by [36], who won the ILSVRC-2014 localization and classification tracks competition. It has a distinctive character compared to AlexNet [37], through its deep network structure with a minimal convolutional filter of 3×3. In the competition, six deep CNNs were employed. The VGG16 consists of 13 convolution layers and three fully connected layers [33]. Both networks employ a stack of 3×3 small convolutional filters with stride 1, followed by multiple non-linearity layers.

The design of VGG16, which has 16 layers of convolution, is very similar to that of AlexNet. The convolution layer portion, which features a 3×3 convolution with several filters and is chosen for feature extraction, receives the 224×224 picture input. Figure 5 shows a convolution layer with 64,128, and 256 filters, followed by a Rectified linear unit (ReLU) activation function, MaxPooling (2×2) layer. The max pooling layer is added to reduce the spatial dimensions of the output. If it is completed up to the last layer in this VGG16 architecture, it will result in a vector of 1000 values. However, after conducting several experiments, we found that using only three blocks of convolutional layers in VGG16 produced good features and enabled real-time implementation.

Kernelized correlation filter (KCF) [9]

The KCF tracker is nominated as a fast tracker in the performance category due to its cyclic shift approach and simple principles [38]. KCF was first proposed by [9] as a classic conventional form of discriminating and a correlation filter framework. This set of methods learns to filter from a series of training samples. The KCF sample is created

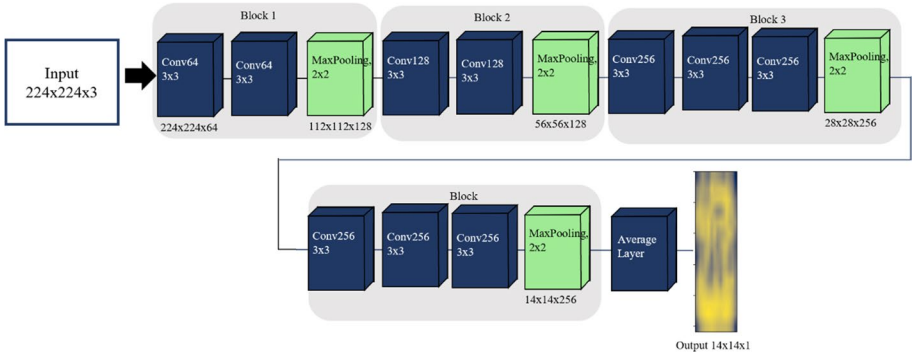

Fig. 5 Feature extraction with VGG16

using the cyclic shift technique, which allows for high frame rates [39]. The training and detection stages are the two basic KCF processes. In this case, the target is chosen as a binary classifier throughout the training phase. The traditional method of tracking involves trying to isolate a group of objects and solving linear regression issues. In order to get the data, linear regression attempts to describe the relationship between two variables using the appropriate linear equation. The equation form of linear regression:

$$y = a + bX, \tag{1}$$

where X is the explanatory variable and y is the dependent variable, b is the slope, and a is the intercept, which is the value of y when $x = 0$. This KCF method concerns more with ridge regression because it has a simple closed-form solution with a more sophisticated method [9]. Mathematically, the objective function of linear ridge regression is (2). The purpose of the training is to get the function $f(z) = w^T z$ that can minimize the square error of the sample x_i and target y_i.

$$\min_w \Sigma_i \left(f(x_i) - y_i\right)^2 + \lambda \|w\|^2, \tag{2}$$

Ridge regression has a closed-form solution by (3)

$$w = (X^T X + \lambda I)^{-1} X^T y. \tag{3}$$

The detection obtained a finding that denotes the target location coordinates. Based on the previous research project [40], it was assumed that the tag of the training sample as $\{(X_1, y_1), (X_2, y_2) \ldots (X_n, y_n)\}$ to locate a purpose $f(z) = w^T z$. Set of samples $X = [X_1, \ldots X_n]$ as an image patch that has one sample per row x_i, y is a regression target y_i, and I is an identity matrix. KCF in this case will play in the Fourier domain which usually uses complex values such as:

$$w = (X^H X + \lambda I)^{-1} X^H y, \tag{4}$$

where X^H as a Hermitian transpose, $X^H = (X^*)^T$ and X^* as a complex conjugate from X. The KCF employs cyclic shift from the base sample to make a circulant matrix thus the linear regression becomes easier. The cyclic shift of X denoted as $P = [X_0, X_1, \ldots X_{n-1}]^T$

where P is the permutation matrix. Cyclic shifts of all sample data matrix $X = P(x)$ or called circulant matrix.

$$X = P(x) = \begin{bmatrix} x_1 & x_2 & x_3 & \cdots & x_r \\ x_r & x_1 & x_2 & \cdots & x_{r-1} \\ x_{r-1} & x_r & x_1 & \cdots & x_{r-2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_2 & x_3 & x_4 & \cdots & x_1 \end{bmatrix}. \tag{5}$$

Discrete Fourier Transform (DFT) makes all circular matrices diagonal regardless of the vector x produced. It is presented as follows:

$$X = F\ diag(\hat{x})F^H, \tag{6}$$

where F is a Digital Fourier Transform (DFT), a constant matrix that does not depend on x and transforms the data into the Fourier domain, $\hat{x}$ is a DFT vector, while F^H is a Hermitian transpose of F. Equation (6) is the eigen decomposition of the circular matrix. Back to Eq. (4) where $X^H X$ as a noncentered covariance matrix so that if substituted with Eq. (6), it will become:

$$X^H X = F\ \mathrm{diag}(\hat{x}^*)F^H\ F\ \mathrm{diag}(\hat{x})F^H. \tag{7}$$

Since the diagonals of the matrix are symmetrical, calculating the Hermitian transpose leaves only the conjugate complex $\hat{x}^*$. Also, we can remove the factor from $F^H F = 1$ so that the equation becomes:

$$X^H X = F\ \mathrm{diag}(\hat{x}^*)\ \mathrm{diag}(\hat{x})F^H. \tag{8}$$

Since the operations on the diagonal matrix are element-wise, this can be defined as the element-wise product as $\odot$ and yielded:

$$X^H X = F\ \mathrm{diag}(\hat{x}^* \odot \hat{x})F^H. \tag{9}$$

By substituting Eq. (4), we get the solution:

$$w = \left(F\ \mathrm{diag}(\hat{x}^* \odot \hat{x})F^H + \lambda I\right)^{-1} X^H y, \tag{10}$$

$$w = \left(F\ \mathrm{diag}(\hat{x}^* \odot \hat{x})F^H + \lambda F^H I F\right)^{-1} X^H y, \tag{11}$$

$$w = \left(F\ \mathrm{diag}(\hat{x}^* \odot \hat{x} + \lambda)^{-1} F^H\right) X^H y, \tag{12}$$

$$w = F\ \mathrm{diag}(\hat{x}^* \odot \hat{x} + \lambda)^{-1} F^H F\ \mathrm{diag}(\hat{x})F^H y, \tag{13}$$

$$w = F\ \mathrm{diag}\left(\frac{\hat{x}}{\hat{x}^* \odot \hat{x} + \lambda}\right) F^H y, \tag{14}$$

$$F\hat{w} = \text{diag}\left(\frac{\hat{x}^*}{\hat{x}^* \odot \hat{x} + \lambda}\right) Fy. \tag{15}$$

Since $Fw = \hat{w}$, then:

$$\hat{w} = \text{diag}\left(\frac{\hat{x}^*}{\hat{x}^* \odot \hat{x} + \lambda}\right) \hat{y}. \tag{16}$$

Since the diagonal and vector matrices product is an element-wise only, then:

$$\hat{w} = \text{diag}\left(\frac{\hat{x}^* \odot \hat{y}}{\hat{x}^* \odot \hat{x} + \lambda}\right), \tag{17}$$

where $\hat{x} = Fx$ is the DFT of x and $\hat{x}^*$ s the complex conjugate of $\hat{x}$. In this case, the Inverse Fast Fourier Transform (IFFT) can be used to calculate the current w simply. The KCF tracker classifier is trained by minimizing the following features. Where λ, as in Eq. (2), represents the regularization parameter, the linear regression classifier does not show good results when the data is distributed in non-linearity. In this case, a classifier needs a more robust nonlinear regression function. The nonlinear problem's low-dimensional solution is mapped to the high-dimensional kernel space using a kernel technique that allows the KCF to extend the problem into nonlinear space. After mapping with the kernel function, the linear regression coefficient of linear problems in high-dimensional kernel space is:

$$w = \sum_i \alpha_i \varphi(X_i). \tag{18}$$

Under the optimal conditions, the parameter α represents the coefficients, and $\varphi(X)$ represents mapping with the Gaussian kernel. In a high-dimensional space, the linear regression function is:

$$f(z) = w^T z = w^T \varphi(z) = \sum_{i=1}^{n} \alpha_i \varphi(X_i)\varphi(z). \tag{19}$$

So, Eq. (19) can be expressed as:

$$f(z) = \sum_{i=1}^{n} \alpha_i \kappa(z_i, X_i), \tag{20}$$

κ as a Gaussian kernel function, and the kernel function calculation method is:

$$\begin{aligned} \kappa^{XX'} &= \kappa\left(X, X'\right) \\ &= \exp\left(-\frac{1}{\sigma^2}\left(||X||^2 + \left|\left|X'\right|\right|^2 - 2\mathcal{F}^{-1}\left(\hat{\mathrm{X}}^* \odot \hat{\mathrm{X}}'\right)\right)\right) \end{aligned} \tag{21}$$

where σ as a standard deviation, * represents complex conjugation. $\mathcal{F}^{-1}$ is an inverse Fourier transformation is a transformation that reverses the direction of the Fourier coefficients. As a result, the α equation is translated from the regression coefficient solution.

$$\alpha = (K + \lambda I)^{-1} y, \tag{22}$$

where K as a *mxm* kernel matrix as:

$$K_{ij} = \kappa\left(X_i, X_j\right). \tag{23}$$

A cyclic matrix can be proven to be the corresponding kernel matrix K.

$$\hat{\alpha} = \frac{\hat{y}}{\widehat{K^{XX'}} + \lambda}. \tag{24}$$

The cyclic shift samples form a cyclic matrix in Eq. (5). Where $\hat{}$ as a DFT. $\kappa^{XX'}$ is the first row of kernel matrix $K = P\left(\kappa^{XX}\right)$. The sample can be checked after training α using the above procedure. All the specimens to be the Z observed are generated by the cyclic shift of the base sample z, and the training sample X is caused by the cyclic shift of the base sample X. The kernel matrix is:

$$K^Z = P\left(K^{XZ}\right). \tag{25}$$

It is possible to measure the response output value of all input samples according to Eq. (20):

$$f(z) = \sum_{i=1}^{n} \alpha_i \kappa(z_i, X_i) = K^{ZT}\alpha. \tag{26}$$

For calculations efficiency, K^Z can be diagonalized with DFT:

$$\hat{f}(z) = \hat{\kappa}^{XZ} \odot \hat{\alpha}. \tag{27}$$

During the tracking process, particularly the updating process in the detection process, $f(z)$ is a match score for all cyclic shifts from the test image patch. In this case, the target position is estimated by calculating the highest value. The use of KCF through the implementation of goal tracking is considered an effective method since it achieves the fastest and highest efficiency among the recent top-performing methods. However, KCF cannot effectively track targets that vanish and appear again with varying scales [40]. The kernel tricks in KCF are used to transform our data into linear separable feature space with a higher dimension.

Our approach

This section presents the detailed methodology of the proposed approach, as illustrated in Fig. 6. In recent years, deep learning algorithms and their computer vision capabilities have undergone significant advancements. Specifically, deep learning architectures are widely utilized for object classification tasks. As a result, in this study, deep learning architectures are utilized as a feature extraction method to distinguish between different objects.

The tracker's first step is to choose a target and create a bounding box to obtain a template object. The object is then passed through a learning transfer block, where a pretrained model is used to extract relevant features learned by the network. These learned

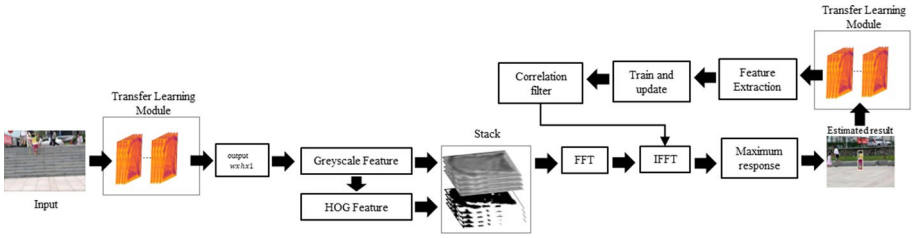

Fig. 6 The overall framework

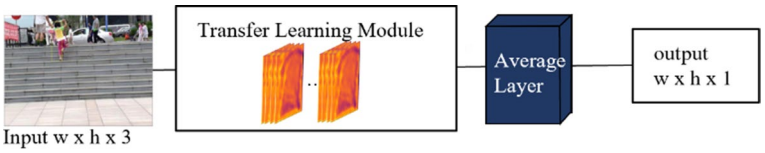

Fig. 7 Deep feature extraction with transfer learning

weights transform the input data into a new representation and initialize feature extraction layers. To learn to detect more complex features, at least three convolution blocks of each transfer learning method are used. By utilizing transfer learning with pre-trained models and selecting the appropriate network blocks, we can extract relevant high-level features for object tracking, improving the tracking algorithm's efficiency. Additionally, this approach can enable real-time implementation of the algorithm by reducing the computational load required for feature extraction. The convolution blocks become higher layers and are designed to average feature maps. The resulting grayscale image represents the most informative features and is used to calculate the HOG feature. By fusing the grayscale and HOG features, the tracking method can overcome the limitations of individual features and capture a complete representation of the object. Thus, the fused features are used for KCF tracking. This approach allows for the comprehensive tracking of an object by combining multiple features.

Deep features-based transfer learning

Figure 7 shows the input image *wxhx*3 referring to an image with width w, height h, and 3 color channels (red, green, and blue). In transfer learning, the image passes through several layers to extract its features, which are typically done through a deep network model that has been previously trained on a large dataset. The last layer of the pre-trained deep neural network model is usually a classification layer consisting of several neurons with an activation function, which provides accurate classification results. However, in this research, the classification layer may not be relevant to the task at hand. Therefore, the last layer of the pre-trained model can be replaced with a new layer, such as an average layer, which produces an output of *wxhx*1. This average layer calculates the average of features in each channel across the entire image and produces an output that represents the overall features of the image.

The grayscale and HOG features are used in KCF and make a good result in real-time implementation and accuracy. Obtaining grayscale features from the transfer learning network can be a useful method for reducing input data dimensionality,

improving computational efficiency, and increasing interpretability by showing the relative intensity of each pixel in all channels of the network, rather than the combination of red, green, and blue values. And also, the grayscale feature can be used for the HOG feature that effectively detects edges and contours, which is essential in localizing objects in images. By combining various features, a tracking method can overcome the limitations of using only one feature and capture a more comprehensive representation of the object being tracked. This approach can improve the accuracy and robustness of the tracking method, as it allows for a more complete and diverse set of information to be used to track the object over time.

We used these weights for feature extraction and obtained the grayscale image, and also, from the grayscale, we could calculate the HOG feature. Nevertheless, deep features depend on extensive datasets. We made use of HOG, while deep features were utilized to overcome the abstract feature extraction capability of HOG. When the multi-feature descriptions of deep learning are combined with HOG, it enhances the classical HOG by extracting features not only from the original image, resulting in an improved feature description ability.

Because of its quick characteristics as a tracker, the KCF tracker is currently being used. To test how it works, we have tried various settings. The tracker generally functions better if the network is deeper, which significantly lengthens the implementation time. Therefore, deep feature extraction is much enhanced by using the third block of each learning transfer.

$$\text{Feature map} = f\left(x_{c,d,e}\right), \tag{28}$$

where (c, d) is a pixel index, e indicates the channels and x represents the input image, and f is the activation function applied element-wise to the output. We utilize the weights of each transfer learning approach as feature extraction through fine-tuning phases in multiple channels. Thus, we can optimize the use of weights learned by the previous network to produce more relevant and representative features in the training and sample testing process. Fine-tuning stages are performed in multiple channels to ensure more accurate and effective feature extraction results. Equation (28) calculates the output of a single filter in a convolutional layer by taking a weighted sum of the input pixels in the corresponding region of the image, applying an activation function, and outputting the resulting feature map. This process is repeated for each filter in the layer to produce a set of feature maps.

We put the benchmark sequences often utilized in the literature to the test. In this instance, we created a video file from the sequences. Additionally, the findings demonstrate strong performance under challenging conditions. Objects in a scene can be obscured by other things. In such cases, the object may be completely hidden by others, disguised in certain areas, or located behind other objects. Occlusion can cause the object's visual model to briefly vary, which can pose a challenge for object-tracking techniques. Figure 8 illustrates the challenge faced by our proposed tracker when the object deals with partial and complete occlusion. Moreover, the results in Fig. 8 demonstrate that the use of a fused feature in KCF leads to peak performance when occlusion is present.

Fig. 8 The tracking result of the Gilr2 dataset when occlusion occurs

Tracking strategy

After the input image has gone through three convolution blocks of transfer learning architecture, the last layer is with the average layer, which produces a grayscale feature with *wxhx*1 dimension. This single-channel feature is used to extract the HOG feature. The feature extraction aims to discover the candidate region's characteristics that can specifically identify the target. The quality of the feature is the most significant direct influence on tracking results. A new kind of pattern detector can be found in each of the derived feature map channels. In this case, others are more sensitive to color information and primarily distinguish textural elements, while some are highly discriminative regarding edges and corners [23]. In complex video environments, most tracking methods rely on subtle features, which can make them less robust and more susceptible to environmental changes. The HOG feature [41] is a descriptor that is unaffected by changes in object color information and rapidly characterizes an object's local gradient characteristics. To take advantage of multiple feature extraction methods, both HOG and grayscale features are computed and combined into a single stack.

The grayscale feature as an image with a single channel computes the HOG features. The input image is first divided into small cells of size 4×4 pixels. Within each cell, the gradient orientations and magnitudes are computed, which are used to construct a histogram of gradient orientations. This histogram is divided into 9 bins, with each bin representing a range of gradient orientations. The HOG feature for each cell is then represented by the values in the corresponding 9 bins of its histogram. These cell-level HOG features are then concatenated to form a block. The block size determines how many cells are included in each block, and in this case, the block size is 8×8 pixels. To capture more information about the image, the blocks overlap with a stride of 4×4 pixels. This means that each cell is included in multiple blocks, and each block is partially overlapping with neighboring blocks. The final HOG feature vector is the concatenation of all the block-level HOG features. Afterwards, the two features are concatenated into one stack and used for KCF tracking.

The KCF method, which uses the surrounding images as a training sample to train the target detector, extracts the HOG feature from an image of a tracking target. Then we trained a kernelized correlation filter using the input from the stack. This method involved using a kernel to compute the correlation between an object template and the input image at each position in the image. First, an object template was selected which

feature had been extracted and a kernel was created with the same size as the template. Then, the kernel was applied to the input image at each position. A kernel is a function used to compute the correlation between the object template and the input image at each position. In this research, the kernel was created using a Gaussian kernel. The location with the highest correlation value is considered to be the location of the desired object by finding the maximum value in the response map (computed using Eq. 26). A tracking method is suggested below:

X: template image patch
Y: desired correlation response
Z: test image patch
$\hat{\boldsymbol{\alpha}}$ **:** coefficient

Output:
Detection score for each location

Pipeline for a tracker:
(1) Image through three blocks convolution of transfer learning architecture which makes single channel output
(2) Get grayscale and HOG feature
(3) Calculate Gaussian kernel correlation as in eq(21);
(4) Calculate $\hat{\boldsymbol{\alpha}}$ with (24)
(5) Extract the deep feature, HOG, and grayscale features and use the discrete Fourier transform to obtain $\boldsymbol{\kappa}^{\boldsymbol{xx}}$
(6) Get responses $\boldsymbol{f}(\boldsymbol{z})$ and find the target position by maximizing $\boldsymbol{f}(\boldsymbol{z})$
(7) Update the parameters

$$\begin{cases} \boldsymbol{\alpha} = (\mathbf{1} - \boldsymbol{\beta})\boldsymbol{\alpha}_{t-1} + \boldsymbol{\beta}\,\hat{\boldsymbol{\alpha}} \\ \boldsymbol{x} = (\mathbf{1} - \boldsymbol{\beta})\boldsymbol{x}_{t-1} + \boldsymbol{\beta}\,\hat{\boldsymbol{x}} \end{cases}$$

The initial goal position was used to train the model. In order to reduce boundary artifacts caused by circular shifts, the patch used was larger than the tracked object, and the patch input used a cosine window. The following frame was made as a test picture using the bounding box's current location. Utilizing the test image as the place of the highest score, the target was discovered. The bounding box was further modified. Meanwhile, in a new position, the new model was trained. In this case, in order to supply memory for the tracker, the convex combination of the current and prior states was updated between alpha and x.

When the number of features is increased, the KCF method speed will substantially decrease. For features fusion, the HOG and grayscale feature are extracted and obtained z from discrete Fourier transform. Furthermore, $\kappa(z_i, X_i)$ is calculated using the Eq. (26). Then, parameter α in KCF is also determined. The output of regression values of all possible response regions of each feature $= \hat{f}(z_i) = \hat{\kappa}^{Xz_i} \odot \hat{\alpha}$. We designed the response of each feature as presented in Fig. 9. Thus, we could get the target position with the

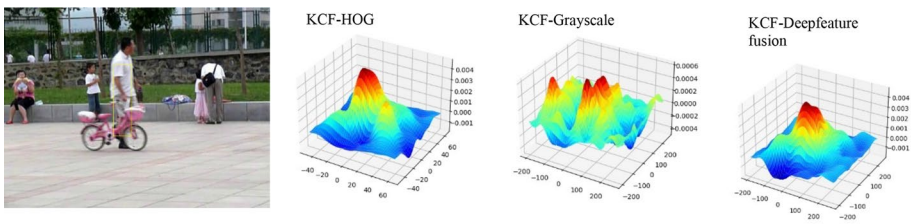

Fig. 9 HOG, grayscale, and feature fusion response

maximum response feature. The position corresponding to the maximum response is the target's location being tracked. However, when partial occlusion occurs, the peak value may be affected and lead to inaccurate tracking results. In Fig. 9, multiple response peaks are generated by the KCF method using grayscale features and two HOG vector peaks. To improve the accuracy of the results, feature fusion is performed between deep grayscale features and HOG. The result of this feature fusion produces a single response peak with the maximum value. This occurs because deep features can extract more complex features, making the object's appearance more distinct than grayscale and HOG features used separately. Therefore, the use of deep gray and HOG feature fusion in the KCF method can significantly improve the accuracy and precision of object detection.

Experimental setup

a. Hardware setup
   All the implementation software ran on Windows using Keras with TensorFlow. v1.15, which is a backend to Keras [42]. The hardware setup was CPU i7 AMD Ryzen 5 3500X 6-Core Processor and 64-bit operating system. The graphical processor unit was a single Nvidia Geforce GTX 1650.
b. Dataset
   Experiments on challenging videos dataset [43] show that the proposed approach has been successfully implemented. In this case, a sequence of images is converted to obtain a video file.
c. Evaluation metrics
   The evaluation of tracking algorithms is a critical aspect of object tracking. The Center Location Error (CLE) is a widely-used evaluation metric that calculates the Euclidean distance between the predicted object center location and the ground-truth object center location. Another important metric is the Overlap Success Rate (OS%), which measures the overlap between the predicted and ground-truth bounding boxes and compares it to the union of their areas. Additionally, precision and recall are commonly used to evaluate the performance of tracking algorithms. Precision measures the true positives about the sum of true positives and false positives, while recall measures the true positives concerning the sum of true positives and false negatives. These metrics are typically used to evaluate the performance of a tracking algorithm for each video with a threshold of 0.5.

Results

The goal of feature extraction is to discover features in the candidate region that get the target characteristics exclusively. The most significant direct influence on tracking outcomes in object tracking comes from the feature's quality. After passing through three block convolutional layers of transfer learning networks, we combine these HOG and grayscale features into a single stack to increase tracking and real-time implementation performance. When a complete occlusion occurs in the Girl2 dataset, the suggested approach effectively tracks the object.

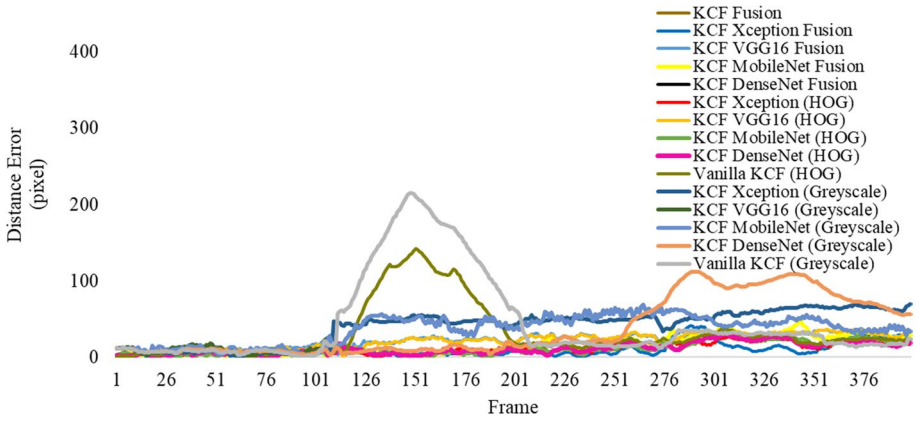

Fig. 10 Center location error curves of Girl2 dataset (1st frame to 400th frame)

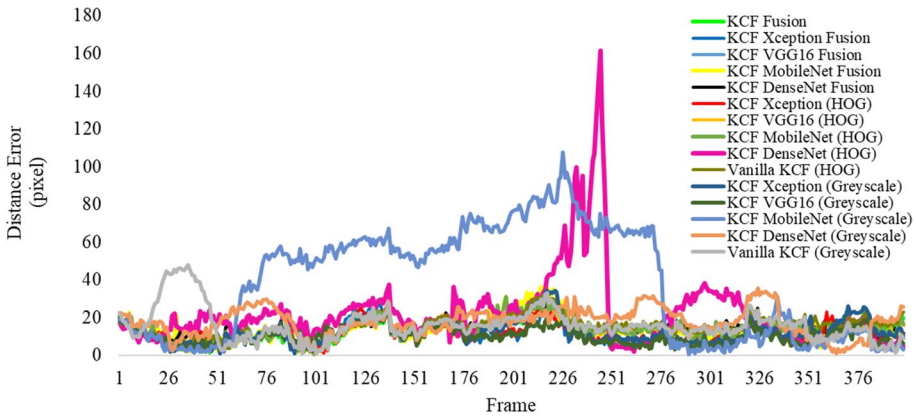

Fig. 11 Center location error curves of faceocc1 dataset (1st frame to 400th frame)

Our proposed method yields a significant increase compared to the vanilla KCF technique. The graph Fig. 10 shows that when a severe occlusion occurs at frame #109, the center location error significantly increases. This can be due to the fact that during a full occlusion, the object cannot be well detected using HOG and grayscale features in the KCF method. Both features heavily rely on image clarity and texture details, and thus, when a full occlusion occurs, these features may not provide sufficient information to track the object. However, when a full occlusion occurs at frame #109, the use of deep features with at least three convolutional layer blocks in transfer learning can overcome this issue. These features are generated from layers of convolutions and pooling that can extract information at increasingly complex levels.

Figure 11 shows the location error in using KCF MobileNet with the grayscale feature with CLE of 31.33 and KCF Densenet with HOG feature with CLE of 22.92, it shows that many points produce suddenly high error values. In addition, errors can also occur due to the weakness of these features in recognizing objects under certain conditions, such as when there is occlusion or poor lighting. However, location errors when using KCF Xception fusion and KCF Xception HOG indicate that both methods produce relatively small error values in partial and severe object occlusion conditions.

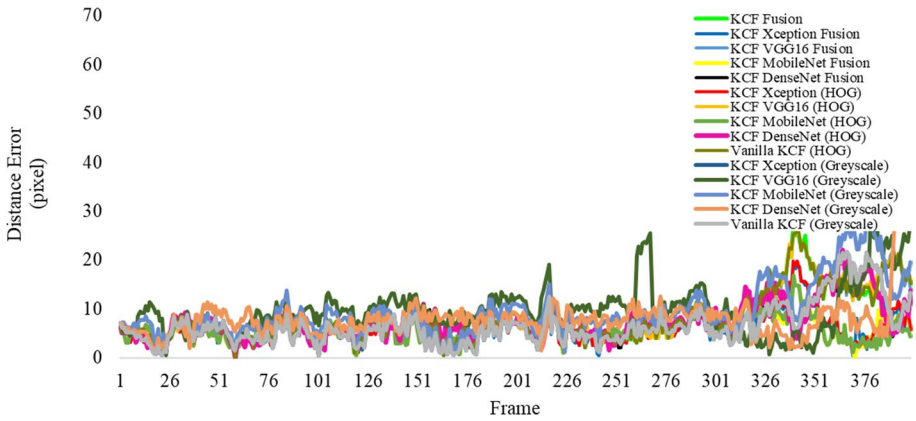

Fig. 12 Center location error curves of faceocc2 dataset (1st frame to 400th frame)

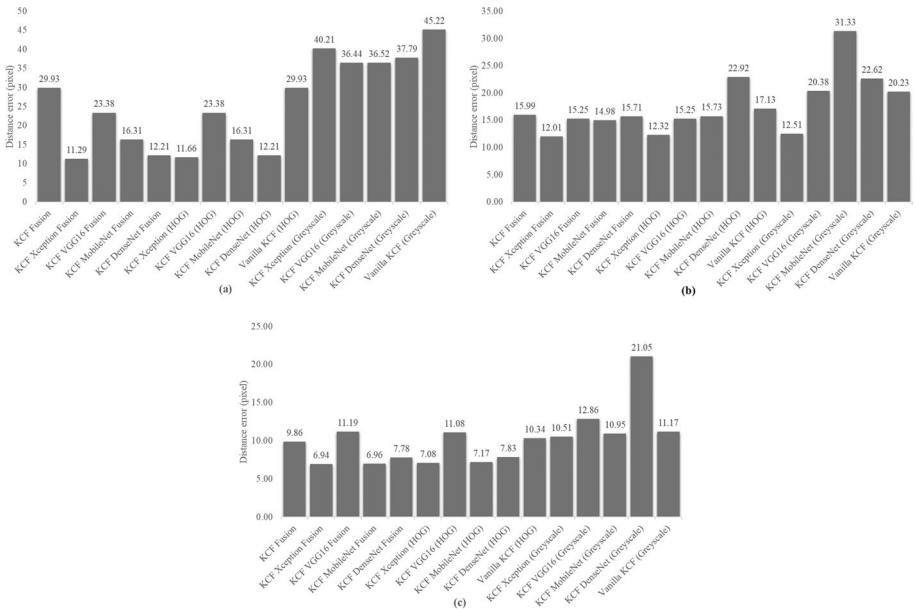

Fig. 13 CLE result of **a** girl2, **b** faceocc1, and **c** faceocc2 dataset

The details are shown in Fig. 13b. This can be attributed to the ability of both methods to extract complex object features in various conditions, including occlusion conditions.

Figure 12 shows the results of an experiment that compares the performance of different fusion methods on a faceocc2 dataset. The results show that the KCF-Xception fusion method produces the lowest CLE with a value of 6.94, which is significantly lower than the other methods. The MobileNet fusion method produces the second lowest CLE after KCF-Xception with a value of 6.96. On the other hand, the KCF DenseNet Grayscale method produces the largest CLE with a value of 21.63. The Grayscale feature, which transforms images into grayscale before processing, leads to the loss of valuable color information that could be beneficial for object tracking. The use of deep features with transfer learning for object feature extraction in images can

Table 1 The performance of different transfer learning deep features models in the OTB-100 dataset

Transfer learning	CLE AVG	HOG	Grayscale	Recall	Precision	OS	CLE	FPS
KCF fusion	18.59	✓	✓	0.82	0.63	0.59	18.59	64
KCF-Xception fusion	12.84	✓	✓	**0.93**	**0.81**	**0.80**	**10.08**	36
KCF VGG16 fusion		✓	✓	0.88	0.62	0.58	16.61	41
KCF MobileNet fusion		✓	✓	**0.90**	0.74	0.67	12.75	33
KCF DenseNet fusion		✓	✓	**0.90**	**0.76**	**0.68**	11.90	30
KCF Xception	13.58	✓	×	0.76	0.69	0.64	**10.35**	51
KCF VGG16		✓	×	0.88	0.61	0.58	16.57	44
KCF MobileNet		✓	×	0.86	0.71	0.66	13.07	57
KCF DenseNet		✓	×	0.87	0.75	**0.68**	14.32	48
Vanilla HOG KCF	19.13	✓	×	0.69	0.57	0.55	19.13	**68**
KCF Xception	24.43	×	✓	**0.93**	0.60	0.58	21.08	56
KCF VGG16		×	✓	0.78	0.70	0.64	23.23	49
KCF MobileNet		×	✓	0.68	0.62	0.56	26.27	60
KCF DenseNet		×	✓	0.73	0.58	0.56	27.15	53
Vanilla grayscale KCF	25.54	×	✓	0.81	0.58	0.57	25.54	**72**
KAZE [44]	–	–	–	–	–	–	18.50	
Unscented Rauch-Tung-Striebel smoother and KCF [45]	–	–	–	–	–	–	–	55.2
Yolov3+KCF [10]	–	–	–	–	–	–	–	16.1

result in lower CLE. This means that deep features can help improve the performance of object-tracking systems, especially in cases of object occlusion.

As in Fig. 13, the analysis of the error location plot on the girl2 dataset shows that the use of Baseline KCF with grayscale features still results in high error location values in some frames. The CLE value in using Baseline Vanilla KCF with grayscale features is 45.22 pixels. This can be attributed to the limitations of grayscale features in recognizing objects in various conditions. In addition, Baseline KCF only uses one feature, so it cannot extract features complexly on the tracked object. Furthermore, when the proposed deep grayscale feature with multiple layers was added, it was able to improve the results of the baseline KCF-grayscale feature on the girl2 dataset. The average location error produced by the deep grayscale feature on the girl2 dataset was 37.74 pixels, which is smaller compared to the baseline Vanilla KCF-grayscale feature. Furthermore, the duration of object occlusion has also been calculated, with an average of 56 frames for partial occlusion and 8 frames for severe occlusion.

From Table 1, the recall result, there is a significant variation in values, ranging from Vanilla HOG KCF with a recall of 0.69 to KCF-Xception Fusion with a recall of 0.93. Some values are relatively high, including KCF-Xception Fusion, KCF-DenseNet Fusion, and KCF Xception grayscale, indicating their ability could track the object. The method employing the green and red font indicates the highest and second-highest results. And also, Table 1 shows the deep feature HOG was able to produce an average CLE of 13.58 pixels, which was significantly smaller compared to the baseline KCF-grayscale feature and KCF-deep grayscale feature. The use of the deep feature HOG in the KCF method improved the tracking performance compared to the KCF-deep grayscale feature. This deep feature fusion was able to achieve an average CLE error of 12.84 pixels, which is smaller compared to using only deep feature HOG or grayscale. These results demonstrate that several transfer learning models, such as Xception, VGG16, MobileNet, and DenseNet with convolution blocks, can improve the recall, precision, OS, and CLE of

the KCF method for object tracking. Moreover, adding a transfer learning module, followed by the computation of either HOG or grayscale features or their fusion, has been shown to improve the performance of the KCF tracker.

Conclusion

This study integrated the deep feature architecture and hand-crafted features HOG and grayscale and allows for real-time implementation and effective occlusion handling, a common challenge in tracking methods. By fusing different features, the tracking method can overcome the limitations of individual features and capture a complete representation of the object. These methods improve the feature extraction process by leveraging the information gained from pre-trained models. Once the input image is obtained through convolution blocks transfer learning architecture (such as Xception, VGG16, MobileNet, or DenseNet), the HOG and grayscale features are computed and combined in KCF methods. The result shows that the fusion of Xception transfer learning with HOG and grayscale in the KCF method significantly improves the recall, precision, OS, and CLE, and real-time implementation achieves up to 36 FPS. The study is limited by the duration of occlusion on the object, which is not too long, with an average of 56 frames for partial occlusion and 8 frames for severe occlusion, and focuses solely on single object tracking. In the future, we can explore the possibility of using object target tracking techniques for multiple object tracking. By doing so, we can enhance the efficiency and speed of real-world applications of object-tracking technology.

Abbreviations

KCF	Kernelized correlation filter
HOG	Histogram oriented gradient
DFT	Digital Fourier Transform
FFT	Fast Fourier Transform
IFFT	Inverse Fast Fourier Transform
CLE	Center location error
CNN	Convolutional neural network
FPS	Frame per second

Acknowledgements
This work was supported in part by Ministry of Research and Technology/National Research and Innovation Agency (Penelitian Disertasi Doktor) and the School of Electrical Engineering and Informatics Institut Teknologi Bandung.

Author contributions
DA, CM, LY, and PH wrote the paper, designed the model and the computational framework, and analyzed the data. DA and LY developed the theoretical framework. CM and PH contributed to the interpretation of the results. All authors discussed the results, commented on the manuscript, reviewed drafts of the article, and approved the final draft.

Funding
This research received funding from Ministry of Research and Technology/National Research and Innovation Agency (Penelitian Disertasi Doktor) and the School of Electrical Engineering and Informatics Institut Teknologi Bandung.

Availability of data and materials
The original dataset used for this study is available in: http://cvlab.hanyang.ac.kr/tracker_benchmark/datasets.html.

Declarations

Ethics approval and consent to participate
Not applicable.

Consent for publication
Not applicable.

Competing interests
The author reports no potential competing interests.

Received: 27 June 2022 Accepted: 18 August 2023
Published online: 01 September 2023

References

1. Liu Y, Meng Z, Zou Y, Cao M. Visual object tracking and servoing control of a nano-scale quadrotor: system, algorithms, and experiments. IEEE/CAA J Autom Sin. 2021;8:344–60.
2. Cui S, Wang Y, Wang S, Wang R, Wang W, Tan M. Real-time perception and positioning for creature picking of an underwater vehicle. IEEE Trans Veh Technol. 2020;69:3783–92.
3. Padmaja B, Myneni MB, Krishna Rao Patro E. A comparison on visual prediction models for MAMO (multi activity-multi object) recognition using deep learning. J Big Data. 2020;7:1–15.
4. Sreenu G, Saleem Durai MA. Intelligent video surveillance: a review through deep learning techniques for crowd analysis. J Big Data. 2019;6:1–27.
5. Bolme DS, Beveridge JR, Draper BA, Lui YM. Visual object tracking using adaptive correlation filters. In: 2010 IEEE computer society conference on computer vision and pattern recognition. 2010. p. 2544–50.
6. Zhang J, Liu H, He Y, Kuang LD, Chen X. Adaptive response maps fusion of correlation filters with anti-occlusion mechanism for visual object tracking. EURASIP J Image Video Process. 2022;2022:4.
7. Khan B, Jalil A, Ali A, Alkhaledi K, Mehmood K, Cheema KM, et al. Multiple cues-based robust visual object tracking method. Electronics. 2022;11:345.
8. Zhao F, Hui K, Wang T, Zhang Z, Chen Y. A KCF-based incremental target tracking method with constant update speed. IEEE Access. 2021;9:73544–60.
9. Henriques JF, Caseiro R, Martins P, Batista J. High-speed tracking with kernelized correlation filters. IEEE Trans Pattern Anal Mach Intell. 2014;37:583–96.
10. Chen Y, Sheng R. Single-object tracking algorithm based on two-step spatiotemporal deep feature fusion in a complex surveillance scenario. Math Probl Eng. 2021;2021:1–11.
11. Maharani DA, Machbub C, Rusmin PH, Yulianti L. Feature fusion with deep neural network in kernelized correlation filters tracker. In: 2021 IEEE 11th international conference on system engineering and technology (ICSET). 2021. p. 363–7.
12. Kinasih F, Machbub C, Yulianti L, Rohman AS. Two-stage multiple object detection using CNN and correlative filter for accuracy improvement. Heliyon. 2023;9: e12716.
13. Ding M, Chen WH, Wei L, Cao YF, Zhang ZY. Visual tracking with online assessment and improved sampling strategy. IEEE Access. 2020;8:36948–62.
14. Harris C, Stephens M. A combined corner and edge detector. In: Alvey vision conference. 1988. p. 10–5244.
15. Lowe DG. Object recognition from local scale-invariant features. In: Proceedings of the seventh IEEE international conference on computer vision. 1999. p. 1150–7.
16. Bertinetto L, Valmadre J, Henriques JF, Vedaldi A, Torr PHS. Fully-convolutional siamese networks for object tracking. In: Computer vision–ECCV 2016 workshops: Amsterdam, the Netherlands, October 8–10 and 15–16, 2016, proceedings, Part II 14. 2016. p. 850–65.
17. Comaniciu D, Meer P. Mean shift: a robust approach toward feature space analysis. IEEE Trans Pattern Anal Mach Intell. 2002;24:603–19.
18. AlBasiouny ER, Attia AF, Abdelmunim HE, Abbas HM. Robust visual tracking using very deep generative model. J Big Data. 2023;10:1–26.
19. Xie Y, Shen J, Wu C. Affine geometrical region CNN for object tracking. IEEE Access. 2020;8:68638–48.
20. Li C, Yang B. Adaptive weighted CNN features integration for correlation filter tracking. IEEE Access. 2019;7:76416–27.
21. Rohan A, Rabah M, Kim SH. Convolutional neural network-based real-time object detection and tracking for parrot AR drone 2. IEEE Access. 2019;7:69575–84.
22. Ding J, Huang Y, Liu W, Huang K. Severely blurred object tracking by learning deep image representations. IEEE Trans Circuits Syst Video Technol. 2015;26:319–31.
23. Ma C, Huang J Bin, Yang X, Yang MH. Hierarchical convolutional features for visual tracking. In: Proceedings of the IEEE international conference on computer vision. 2015. p. 3074–82.
24. Nam H, Han B. Learning multi-domain convolutional neural networks for visual tracking. In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2016. p. 4293–302.
25. Chen Y, Xia R, Zou K, Yang K. FFTI: image inpainting algorithm via features fusion and two-steps inpainting. J Vis Commun Image Represent. 2023;91: 103776.
26. Chen Y, Xia R, Zou K, Yang K. RNON: image inpainting via repair network and optimization network. Int J Mach Learn Cybern. 2023;14:1–17.
27. Chen Y, Xia R, Yang K, Zou K. MFFN: image super-resolution via multi-level features fusion network. Vis Comput. 2023. https://doi.org/10.1007/s00371-023-02795-0.
28. Chen Y, Xia R, Yang K, Zou K. DGCA: high resolution image inpainting via DR-GAN and contextual attention. Multimed Tools Appl. 2023. https://doi.org/10.1007/s11042-023-15313-0.
29. Zhang J, Bhuiyan MZA, Yang X, Singh AK, Hsu DF, Luo E. Trustworthy target tracking with collaborative deep reinforcement learning in EdgeAI-aided IoT. IEEE Trans Ind Inform. 2021;18:1301–9.
30. Li H, Wang D, Zhang J, Li Z, Ma T. Image super-resolution reconstruction based on multi-scale dual-attention. Connect Sci. 2023. https://doi.org/10.1080/09540091.2023.2182487.
31. Szegedy C, Liu W, Jia Y, Sermanet P, Reed S, Anguelov D, et al. Going deeper with convolutions. Proc IEEE Comput Soc Conf Comput Vis Pattern Recognit. 2015. p. 1–9.
32. Szegedy C, Vanhoucke V, Ioffe S, Shlens J, Wojna Z. Rethinking the Inception Architecture for Computer Vision. Proc IEEE Comput Soc Conf Comput Vis Pattern Recognit. 2016;2818–26.

33. Mahdianpari M, Salehi B, Rezaee M, Mohammadimanesh F, Zhang Y. Very deep convolutional neural networks for complex land cover mapping using multispectral remote sensing imagery. Remote Sens. 2018;10:1119.
34. Huang G, Liu Z, Van Der Maaten L, Weinberger KQ. Densely connected convolutional networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2017. p. 4700–8.
35. Howard AG, Zhu M, Chen B, Kalenichenko D, Wang W, Weyand T, et al. MobileNets: efficient convolutional neural networks for mobile vision applications. arXiv Preprint. 2017. https://arxiv.org/abs/170404861.
36. Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition. arXiv Prepr. 2014. https://arxiv.org/abs/14091556.
37. Krizhevsky A, Sutskever I, Hinton GE. ImageNet classification with deep convolutional neural networks. Commun ACM. 2017;60:84–90.
38. Zhou T, Zhu M, Zeng D, Yang H. Scale adaptive kernelized correlation filter tracker with feature fusion. Math Probl Eng. 2017. https://doi.org/10.1155/2017/1605959.
39. Yue F, Li X. Improved kernelized correlation filter algorithm and application in the optoelectronic tracking system. Int J Adv Robot Syst. 2018;15:1729881418776582.
40. Wang X, Wang G, Zhao Z, Zhang Y, Duan B. An improved kernelized correlation filter algorithm for underwater target tracking. Appl Sci. 2018;8:2154.
41. Dalal N, Triggs B. Histograms of oriented gradients for human detection. In: 2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05), vol. 1. 2005. p. 886–93.
42. Abadi M, Agarwal A, Barham P, Brevdo E, Chen Z, Citro C, et al. TensorFlow: large-scale machine learning on heterogeneous distributed systems. arXiv Preprint. 2016. https://arxiv.org/abs/160304467.
43. Wu Y, Lim J, Yang MH. Online object tracking: a benchmark. In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2013. p. 2411–8.
44. Bhat PG, Subudhi BN, Veerakumar T, Laxmi V, Gaur MS. Multi-feature fusion in particle filter framework for visual tracking. IEEE Sens J IEEE. 2019;20:2405–15.
45. Xia R, Chen Y, Ren B. Improved anti-occlusion object tracking algorithm using unscented Rauch-Tung-Striebel smoother and kernel correlation filter. J King Saud Univ Comput Inf Sci. 2022;34:6008–18.

Publisher's Note