applied sciences

MDPI

Article

Vision-Based Hand Detection and Tracking Using Fusion of Kernelized Correlation Filter and Single-Shot Detection

Mohd Norzali Haji Mohd [1], Mohd Shahrimie Mohd Asaari [2,*], Ong Lay Ping [2] and Bakhtiar Affendi Rosdi [2]

[1] Faculty of Electrical and Electronics Engineering, Universiti Tun Hussein Onn Malaysia, Parit Raja, Batu Pahat 86400, Johor, Malaysia; norzali@uthm.edu.my
[2] School of Electrical and Electronic Engineering, Universiti Sains Malaysia, Engineering Campus, Nibong Tebal 14300, Penang, Malaysia
* Correspondence: mohdshahrimie@usm.my; Tel.: +60-45996086

Abstract: Hand detection and tracking are key components in many computer vision applications, including hand pose estimation and gesture recognition for human–computer interaction systems, virtual reality, and augmented reality. Despite their importance, reliable hand detection in cluttered scenes remains a challenge. This study explores the use of deep learning techniques for fast and robust hand detection and tracking. A novel algorithm is proposed by combining the Kernelized Correlation Filter (KCF) tracker with the Single-Shot Detection (SSD) method. This integration enables the detection and tracking of hands in challenging environments, such as cluttered backgrounds and occlusions. The SSD algorithm helps reinitialize the KCF tracker when it fails or encounters drift issues due to sudden changes in hand gestures or fast movements. Testing in challenging scenes showed that the proposed tracker achieved a tracking rate of over 90% and a speed of 17 frames per second (FPS). Comparison with the KCF tracker on 17 video sequences revealed an average improvement of 13.31% in tracking detection rate (TRDR) and 27.04% in object detection error (OTE). Additional comparison with MediaPipe hand tracker on 10 hand gesture videos taken from the Intelligent Biometric Group Hand Tracking (IBGHT) dataset showed that the proposed method outperformed the MediaPipe hand tracker in terms of overall TRDR and tracking speed. The results demonstrate the promising potential of the proposed method for long-sequence tracking stability, reducing drift issues, and improving tracking performance during occlusions.

Keywords: human–computer interaction; hand detection; hand tracking; single-shot detection; kernelized correlation filter

Citation: Haji Mohd, M.N.; Mohd Asaari, M.S.; Lay Ping, O.; Rosdi, B.A. Vision-Based Hand Detection and Tracking Using Fusion of Kernelized Correlation Filter and Single-Shot Detection. *Appl. Sci.* **2023**, *13*, 7433. https://doi.org/10.3390/app13137433

Academic Editors: Yudong Zhang and Haidi Ibrahim

Received: 15 February 2023
Revised: 29 May 2023
Accepted: 12 June 2023
Published: 23 June 2023

1. Introduction

Hand detection and tracking are crucial front-end procedures for many human hand-gesture tasks and remain challenging topics for researchers. They play a vital role in various computer vision applications, including hand pose estimation, human–computer interaction (HCI) systems, virtual reality (VR), augmented reality (AR), and more. These tasks are essential in numerous human–machine applications, such as providing navigational assistance for the visually impaired [1], enabling contactless navigation for surgeons to minimize contamination during surgery [2], recognizing and interpreting sign language through hand gestures [3], and facilitating interactive remote control for VR video games [4].

Hand gestures can be classified into two categories: static gestures and dynamic gestures. Static hand gestures refer to hand postures where the position remains unchanged for a certain period of time, while dynamic hand gestures involve hand postures where the hand position changes over time. Static gestures are characterized by factors such as orientations, shape, and contextual environment, whereas dynamic gestures, also known as trajectory-based gestures, are characterized by factors such as trajectory and motion speed [5]. In the context of trajectory-based gestures, the accurate detection and tracking of the hand play a vital role. Recognizing and interpreting dynamic hand gestures requires

capturing the hand's motion trajectory and accurately tracking its movement. Detection ensures that the hand is identified and localized within the frame, while tracking enables the continuous monitoring of the hand's position, orientation, and motion throughout a sequence of frames. The detection and tracking of the hand are critical for trajectory-based gesture recognition. By extracting the trajectory pattern from landmarks such as the center of the palm, user can analyze the hand's motion and obtain meaningful features for gesture recognition algorithms. The integration of detection and tracking techniques allows the user to capture and analyze the temporal dynamics of hand gestures, providing more accurate and reliable recognition results.

In general, two main approaches can be implemented for hand detection and tracking: marker-aided and vision-based marker-less methods. The former approach requires the user to wear a glove-like device, which provides more accurate results. However, wearing such a device inconveniences the user and restricts the naturalness of HCI [6]. The latter technique is undoubtedly a better approach as it allows for natural HCI interaction without the need for wearable devices. However, its accuracy may vary when the vision system is exposed to external factors such as extreme lighting conditions and cluttered backgrounds.

In addition to accurate hand detection, tracking hand movement is essential for many hand-based computer vision applications. The hand tracking must be smooth, fast, and accurate to extract and decode information correctly. However, reliably detecting hands from cluttered scenes remains a challenging task. This is due to the complex appearance diversities of dexterous human hands in color images, including variations in hand shapes, skin colors, illuminations, orientations, scales, etc. [7]. Human hands are dynamic and tend to move quickly, making tracking difficult (e.g., clenching and releasing a fist, scratching the nose, waving hands). Furthermore, occlusion is a common issue encountered in hand detection systems [8]. Occlusion occurs when the hand is obstructed by other objects during real-time detection, such as covering the left hand with the right hand or placing the hand in front of the face. This introduces higher complexity to the system as the extracted information is limited for achieving the goal of hand detection.

Apart from the occlusion with external objects, the human hand can also be self-occluded; for example, when clenching a fist, the fingertips are occluded by the palm. Moreover, real-time object-detection mechanisms require heavy I/O operations and accurate processing [9], making them computationally expensive. The system needs to continuously capture image frames from the camera feed, detect the object, draw the predicted bounding box, and track the target object in every frame. This heavy and complex computation can slow down the system, making it undesirable even with high accuracy. Therefore, a wise trade-off between speed and accuracy also needs to be considered.

2. Related Works and Motivation

Vision-based marker-less hand tracking holds great promise for HCI applications, including sign language recognition, augmented reality, telesurgery, home automation, and gaming. However, its implementation faces several challenges, including tracking inaccuracy caused by complex articulated hand motion, high appearance variability, and demanding computational and real-time requirements [10–14]. Consequently, research in this field continues to be a challenging problem that has gained significant attention from the computer vision community.

Sharp et al. [11] presented a real-time hand tracking system based on a single depth camera. The system tracked hands in various poses and environments and demonstrated fast and accurate recovery from tracking failures. They utilized the analysis by synthesis technique, generating candidate poses and scoring them against the input image to determine the most likely hand pose. Machine learning and temporal propagation contributed to a large set of candidate hand-pose hypotheses. The evaluation showed accurate tracking of different hand poses, including open, closed, and gesturing hands. However, the system had limitations regarding occlusions and changes in lighting conditions.

Recent studies have shown that the accuracy of vision-based tracking can be improved by the adoption of deep learning methods through the Convolutional Neural Network (CNN) technique. Mueller et al. [15] proposed an innovative real-time 3D hand tracking method using monocular RGB images. Their approach combined a CNN with a kinematic 3D hand model to achieve accurate tracking. They predicted 2D joint heatmaps, enabling the 3D hand model to infer corresponding 3D joint positions. A noteworthy contribution was the introduction of a novel synthetic data generation method using a Generative Adversarial Network (GAN). This GAN was trained to generate hand images that closely resemble real images, enhancing the CNN's robustness to variations in lighting, pose, and occlusion. However, the authors acknowledged certain limitations in their approach. They addressed scenarios where the hand and background had similar appearances, which posed challenges for their network model, leading to unstable tracking and inaccurate predictions. Furthermore, tracking became problematic when multiple hands were in close proximity within the input image, resulting in unreliable detections and posing a challenge for accurate tracking.

Zhang et al. [16] proposed MediaPipe hands, a real-time hand detection and tracking system based on the BlazePalm CNN architecture. This method accurately predicted hand poses using image inputs and detected hand bounding boxes and landmarks. The system followed a "tracking by detection" approach, with the palm detector locating hand regions and the hand landmark model estimating precise 2.5D landmarks. These landmarks accurately depicted specific key points on the hand, such as fingertips, knuckles, and the center of the palm. However, challenges arose when hands were occluded, in extreme poses, or appeared very small in the image due to large distances. Additionally, factors such as lighting conditions and cluttered backgrounds could affect system performance. MediaPipe hands offered robust hand tracking with accurate pose estimation, but limitations had to be considered.

The CNN sub-libraries such as Regional-based Convolutional Neural Network (R-CNN), Faster R-CNN, You Only Look Once (YOLO), and Single-Shot Detection (SSD) algorithms have also gained attention in this research field. In Wu et al. [8], hand pose estimation was achieved by performing skeleton-difference network (SDNet) analysis to predict the locations of hand joints. To train the network for detecting hand location, a deep learning model based on CNN was implemented. For model training, the dataset was pre-processed by cropping the hand region and labelling the hand joints. The employed CNN network model had another layer added to it known as a position-sensitive region of interest (RoI) pooling layer, based on ZFnet, to further improve its accuracy. From their experiments, the proposed model was able to detect hands and three other classes accurately, achieving a mAP of 91.6%.

On the other hand, Liu et al. [17] implemented two popular CNN backbones, the VGG-16 and ResNet 50 to compare their performance in a hand detection system. Both backbones were trained on the same dataset, ImageNet, to ensure a fair comparison. Additional features such as rotational map were introduced to the CNN network to enable the detector to draw bounding boxes near rotated hands in images. Implementation results showed a significant improvement in mAP, with the VGG-16 network achieving 92.3% mAP and a frame rate of 13.10 FPS and ResNet 50 attaining 94.8% mAP and frame rate of 19.80 FPS, outperforming the other existing detection benchmarks such as YOLO (76.4% mAP and frame rate of 35 FPS). In Gao et al. [18], the existing SSD network was improved by replacing the original backbone of SSD, VGG-16 net with ResNet 101, and fusing the feature maps' three Conv4 layers with two Conv6 layers. These improvements are aimed at improving the accuracy of the existing SSD network in detecting smaller objects, such as distanced hand gestures to a space robot in a space station. Results from their proposed design showed significant enhancement of SSD model in terms of mAP, as their model was able to achieve mAP of 89.4% on public hand datasets such as the Oxford hands dataset [19], EgoHands dataset [20], and self-developed Space Robot Simple Sign Language (SRSSL) dataset [18].

In another recent work, Mukherjee et al. [21] proposed the Faster R-CNN network in their fingertip detection and tracking system for air-writing recognition. The system was decomposed into several sections: detecting writing hand pose, keeping track of fingertips in every successive frame, and recognition of the air-writing characters. For hand pose detection, the Faster R-CNN network was trained on EgoFinger [22] and EgoHands datasets with four losses: RPN regression loss, RPN classification loss, R-CNN regression loss, and R-CNN classification loss. Other than focusing on detecting the hand pose in video streams, they also proposed implementing hand centroid localization to locate the position of the hands accurately. Visual trackers such as Kernelized Correlation Filter (KCF) tracker [23], Tracking-Learning-Detection (TLD) [24] tracker, and Multiple Instance Learning (MIL) [25] tracker were implemented as the hand tracking mechanism to assist the Faster R-CNN detector in keeping track with the fingertips movement during air-writing process. Their experimental results showed that the proposed method outperformed the rest of the visual trackers, achieving a mAP of 73.1% as compared with TLD tracker (mAP of 66.7%), KCF tracker (mAP of 55.4%), and MIL tracker (mAP of 42.4%).

The focus of this work is to develop a fast and robust hand detection and tracking system that is reliable to support a real-time application. To achieve this objective, a tracking-by-detection algorithm is proposed by integrating the well-known correlation filter-based trackers, the KCF tracker [23], with the state-of-the-art deep learning object-detection algorithm, Single-Shot Detection (SSD) [26]. The KCF tracker is considered in this work because of its widespread acceptance due to its competitive performance in terms of speed and accuracy. For instance, the KCF tracker has been implemented in face tracking systems alongside a Continuously Adaptive Mean-Shift (CamShift) algorithm to optimize the tracking performance and to recover the tracker when the CamShift algorithm failed to track faces midway during inference [27]. Despite its promising characteristics, the KCF tracker still struggles to maintain seamless tracking in the occurrence of occlusion and objects falling out-of-view, and it is unable to correct errors during the tracking process [28]. This limitation is due to the principle of the KCF tracker that traces the target based on the correlation of the appearance and position of the target in the previous frames. Therefore, the presence of challenging scenes that result in a significant mismatch between the appearance of the target and the reference object will lead to tracking failure [10].

In this paper, a robust hand tracking method is proposed which integrates the correlation filter with a correction strategy using the fast object-detection model, the SSD algorithm. With this integration, the tracker can be reinitialized when hand movement is not tracked properly, ensuring consistent and accurate tracking. By detecting the tracked object only during the first frame and when it is lost by the tracker, heavy computational costs of the detector are minimized, leading to an improvement in real-time performance. To assess the performance of the proposed tracker, it is compared with the state-of-the-art KCF tracker in terms of mean average precision (mAP) and frame per second (FPS).

3. Methodology

3.1. SSD Hand Detection Model

In order to support ideal real-time application, the deep learning object-detection model needs to be fast, accurate, and as light as possible to avoid resource exhaustion issues. However, training a robust deep learning model from scratch is very challenging and time-consuming. Thus, the transfer learning approach is preferred to reduce model training time. In this work, SSD MobileNet V2 with Feature Pyramid Network (FPN) Lite 320×320 is chosen as the pre-trained model due to its light weight, fast detection speed, and good mAP results of the Common Objects in Context (COCO) dataset [29].

Unlike the conventional SSD models which employed the default VGG-16 backbone, the selected version of the SSD model employed an FPN feature extractor that consists of three main elements: a bottom-up pathway, top-down pathway, and lateral connections. The top-down pathway and lateral connections are interconnected by addition, whereas the bottom-up pathway is a feedforward pathway with its spatial resolution gradually

decreasing, and semantic values of each layer increasing as the pathway goes up. This results in relatively low semantic values in the bottom-most layer, which is the main reason for the poor performance of SSD in detecting smaller objects. On the other hand, the top-down pathway is responsible for reconstructing higher resolution features by up-sampling the feature maps from higher pyramid levels [30]. The higher resolution features generated by the top-down pathway and the features generated by the bottom-up pathway are then interconnected via the lateral connections, which help to improve the ability of the SSD model in predicting the locations of the detected objects.

To initiate transfer learning for the training process, the pre-trained SSD MobileNet V2 FPN Lite model is first downloaded to the workspace. The base model is loaded with its pre-trained weights and the classification layers of the model (i.e., the top layers in the FPN bottom-up pathway) are frozen to avoid destroying the pre trained information stored in the model. Besides freezing the top classification layers, the batch normalization layer, which contained the weights of mean and variance of the pre-trained model, is also frozen to avoid updating the stored weights of the model during training. With the rest of the layers are frozen, only four layers from the feature extraction layers will be re-trained.

For the re-training procedure, the public hand detection dataset EgoHands (http://vision.soic.indiana.edu/projects/egohands/, accessed on 29 September 2021) [20], developed by Indiana University, is chosen. The EgoHands dataset was mainly used for egocentric hand detection and segmentation tasks. It consists of video sequences captured from a head-mounted camera. The dataset contains 48 video sequences captured in diverse indoor and outdoor environments. The videos feature various activities such as cooking, playing instruments, painting, typing, and more. Each frame of the videos is annotated with hand bounding box annotations and pixel-level segmentation masks. The annotations mark the location and extent of the hands in the frames, allowing for both detection and segmentation tasks. This dataset provides high-quality pixel-wise segmentation of hands in an egocentric view and full annotations of hands. The dataset is randomly split into two sets: 80% of images train the dataset and 20% test the dataset. As we target to detect only one hand from the input image, the number of the class is set to 1, while the batch size is set to 4 to speed up the model training process. The training iteration is set to 20K iterations and the cosine learning rate base is set to 0.08 to train a large batch of datasets. Model training is performed by executing the training script provided by the Tensorflow Object Detection API. Once the training loss has dropped to the optimistic value range (around 0.15–0.20), the model training process will be halted to prevent the model from overfitting.

3.2. Evaluation of Hand Detection Model

Before integrating the hand detection model with the KCF tracker, the trained hand detection model is evaluated based on the average recall (AR) and mean average precision (mAP) for different intersection over union (IoU) threshold. Mean average precision is the mean of average precision evaluated in overall detection classes and/or all IoU thresholds, while average precision is the area under the curve of the precision–recall graph. On the other hand, average recall is the measure of assertiveness or the confidence level of the detector model in detecting a given class [31]. In this work, COCO detection evaluation metrics are used to evaluate the model. In the context of COCO evaluation metrics, the average precision equals mean average precision, standardly evaluated for 10 IoU thresholds from 0.5 to 0.95 with a step size of 0.05 (denoted as IoU $\in$ (0.5:0.05:0.95)) [32]. Generally, mAP evaluates the accuracy of the detector model by comparing the predicted bounding boxes over the ground truth bounding boxes based on a given IoU threshold. For example, when evaluated at IoU = 0.50, only the predicted bounding boxes with the overlapped area with respect to the ground truth bounding boxes exceeding the threshold value is considered as true positive (TP). The more TP obtained at the given IoU threshold, the greater the mAP value, and thus the more accurate the model.

3.3. Integration of SSD and KCF Tracker

Running hand detection on every frame of camera inference or video feed is computationally expensive and requires heavy computation. Thus, the SSD detector is proposed to combine with KCF visual tracker to reduce the heavy computation of the system and as a backup when the visual tracker fails to track hand movement by reinitializing the detector. Figure 1 shows the workflow of the tracking-by-detection algorithm by combining the SSD detector with the KCF visual tracker. The system consists of two phases, namely, the detection phase and the tracking phase. The detection phase is activated on the first frame and when the tracking state fails to recover the correct position. For this purpose, the tracking mode is controlled by a Boolean state "Trackable" as "True or False" as an indicator to activate or deactivate the tracking process on every successive frame. In the detection phase, first, the system will check for tracking status. If no tracking process is running, i.e., "Trackable = False", the SSD detector will be activated.

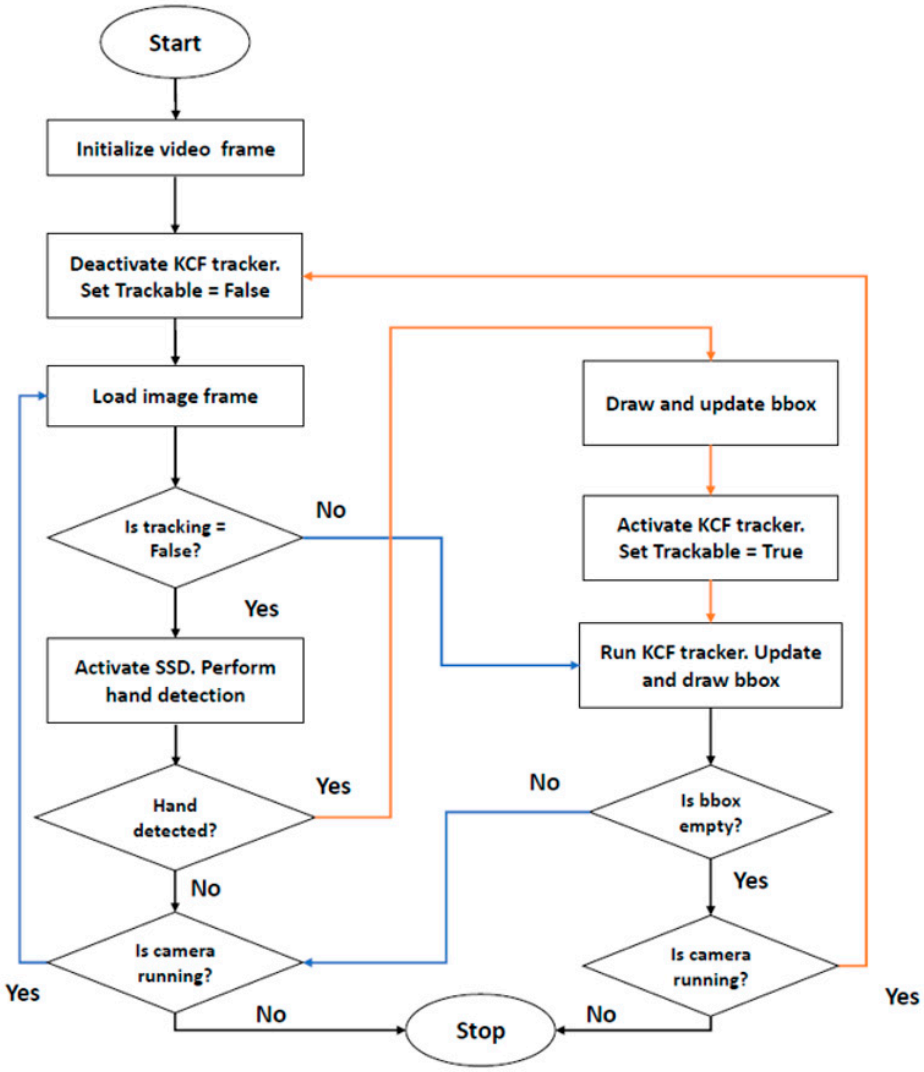

Figure 1. Flowchart of the integration of SSD and KCF Tracker.

In this phase, if a hand is detected, a set of bounding boxes will be drawn on the inference window, and the tuple (data variable) which is used to store the bounding box coordinates will be updated with the new set of bounding box coordinates. The KCF tracker is then initialized and activated, and the system will proceed for the tracking phase. Moving forward in the tracking phase, the system will always check the bounding box tuples to prevent passing the empty bounding box coordinates to the tracker, which may

cause the KCF tracker to crash due to an initialization error. If the bounding box tuple is not empty, the Boolean state "Trackable" is set to "True" to indicate that the tracking process is in progress. The tracker will be updated with the bounding box coordinates obtained from the SSD detector. If the update process is successful, a new set of bounding boxes will be drawn on the inference window on every successive frame, and the bounding box tuples will be updated with the new set of bounding box coordinates. However, if the KCF tracker failed midway during the hand tracking process, the SSD detector will be reinitialized by setting the "Trackable" parameter to "False". The system will then be looped back to the detection phase for reinitialization purposes. The performance of tracking rate and accuracy of this tracking-by-detection algorithm is evaluated by computing its Tracker Detection Rate (TRDR) and Object Tracking Error (OTE) [33], as obtained by Equations (1) and (2), respectively:

$$\text{TRDR} = \frac{\Sigma \text{ True Positives}}{\Sigma \text{ Ground Truth Points}} \times 100 \quad (1)$$

$$\text{OTE} = \frac{\sum_{i=1}^{N} \sqrt{\left(Gx_i - Px_i\right)^2 + \left(Gy_i - Py_i\right)^2}}{N} \quad (2)$$

where True Positives indicates the predicted bounding box position overlapped with respect to the Ground truth position at IoU threshold of 0.5, N is the number of tracked frames, (Gx_i, Gy_i) represents the ground truth of the hand's centroid position, and (Px_i, Py_i) represents the estimated hand's centroid position for the i-th frame.

4. Result and Discussion

This section presents the performance analysis of the proposed hand detection and tracking framework. The evaluation primarily focuses on the performance of the SSD hand detection model on the EgoHands dataset, specifically the testing dataset portion. The evaluation utilizes COCO's detection evaluation metrics, including mean Average Precision (mAP) at different Intersection over Union (IoU) thresholds, average recall (AR), and detection speed, to assess the model's ability to detect hand objects from input images. The evaluation results are then compared with a benchmark model, the Faster R-CNN algorithm, which is widely recognized as one of the most popular and accurate object-detection models.

In the latter part of the experiments, the evaluations focused on the performance analysis of the proposed hand tracking algorithm, as well as the integration of the SSD model with the KCF tracker. The evaluation was performed by running the model inference on sequences of video containing hand movements forming trajectory-based hand gestures and some free hand motions without describing any specific gesture meaning. All videos were recorded using Microsoft LifeCam NX-3000 (Microsoft, China), and the relative distance from the camera to the subject (the hand object) was about 0.5 m. For the trajectory-based hand gestures, there were ten trajectories used to simulate the tracking of hand motion, which described gesture numbers zero to nine. For the free hand motion, there were seven video sequences used to simulate different scenes such as slow movement, fast movement, occlusion behind objects, outdoor lighting conditions, less background contrast, and deformable hand shapes. The tracking rate and accuracy of estimating hand position were evaluated using the TRDR (tracking detection rate) and OTE (object detection error) measures [33]. To observe any improvements in tracking performance, the proposed algorithm was compared to the original KCF tracker. All experiments were conducted on a local computer equipped with an Intel Core i5 CPU 2.50 GHz processor, 8.00 GB RAM, and running the Windows 10 operating system.

Table 1 presents the performance of the SSD and Faster R-CNN hand detection models on the testing data of the EgoHands dataset. The computed metrics reveal that the overall detection accuracy of the SSD model is slightly lower than that of the Faster R-CNN model.

However, in terms of detection speed, the Faster R-CNN model performs significantly slower, taking approximately 15 times longer for inference compared to the SSD model. The slower computation speed of the Faster R-CNN model can be attributed to the heavy workload of the selective search method utilized in the model. This method requires computing four similarity measures, including color similarity, texture similarity, size similarity, and shape similarity, for every detecting frame. Additionally, the Faster R-CNN model necessitates two shots of the image—one for region proposal generation and another for object detection—further contributing to slower real-time computation speed. Considering the trade-off between speed and accuracy, this evaluation justifies the selection of the SSD model for implementation in a real-time hand tracking system.

Table 1. Performance comparison between SSD and Faster-RCNN for hand detection on Ego-Hands dataset.

Mean Average Precision	SSD Model	Faster R-CNN
mAP (IoU = 0.5:0.95)	67.74%	72.29%
mAP (IoU = 0.5)	92.22%	98.64%
mAP (IoU = 0.75)	81.71%	87.57%
AR (IoU = 0.5:0.95)	72.7%	77.9%
Detection time per images	0.0667 s	1.00 s

In the following experiments, the performance of the hand tracking framework is evaluated using image frames from the webcam feed. The SSD detection model is configured with an IoU threshold of 0.5 to suppress false positives during the inference process. Table 2 presents the results of the proposed tracking-by-detection algorithm on video sequences captured from the webcam feed, specifically for ten trajectory-based gestures ranging from '0' to '9'. The comparison with the original KCF tracker and the improvement achieved by the proposed algorithm are also provided.

Table 2. Performance comparison between the proposed algorithm and KCF tracker on ten trajectory-based gestures.

Trajectory Gesture	Frame Num.	KCF + SSD (Proposed)			KCF			Improvement (%)	
		TRDR (%)	OTE	FPS	TRDR (%)	OTE	FPS	TRDR	OTE
0	307	96.1	32.550	17.0	87.8	79.989	31.0	9.4077	59.3112
1	229	96.7	11.920	15.0	86.1	10.316	38.0	12.2634	−15.5461
2	266	88.9	18.072	14.0	85.2	25.434	34.0	4.3432	28.9425
3	281	95.1	38.137	17.0	94.6	36.743	38.0	0.47559	−3.7939
4	318	98.4	9.688	17.0	90.9	12.246	39.0	8.1839	20.8834
5	328	96.2	10.368	18.0	91.7	9.943	39.0	4.95255	−4.2776
6	282	95.2	17.633	16.0	89.1	19.254	35.0	6.8327	8.4199
7	244	90.9	10.535	15.0	88.2	15.642	35.0	3.0258	32.6538
8	283	95.2	9.020	18.0	88.7	32.390	38.0	7.2709	72.1535
9	301	90.0	8.027	17.0	81.9	62.087	36.0	9.8633	87.0709
Average		94.3	16.595	16.4	88.4	30.405	36.3	6.6546	28.5818

It can be observed that the integration of the SSD algorithm (i.e., the proposed algorithm) has significantly enhanced the overall accuracy and tracking rate of the KCF tracker. The proposed algorithm achieves a lower OTE value of 16.595 and a higher tracking rate of 94.3%. However, in terms of average frame rate, the performance of the KCF tracker surpasses that of the proposed algorithm, with an average speed of 36.3 FPS compared to 16.40 FPS achieved by the proposed algorithm. Although the KCF tracker exhibits faster processing speed, its overall accuracy is comparatively lower. The KCF tracker encounters challenges in accurately tracking hand gestures "0", "3", "8", and "9" due to its inflexibility

in adapting to scale variations and its susceptibility to drifting issues, especially during rapid hand movements.

To provide a deeper understanding of these issues, we direct readers to Figure 2, which presents an example showcasing the tracking results for gesture "8". In this case, the proposed algorithm has demonstrated superior performance compared to the KCF tracker, achieving a promising tracking rate of 95.2% and a lower tracking error of 9.02. Conversely, despite the KCF tracker's higher overall frame rate, it exhibits a higher tracking error of 32.3901, which is three times greater than that of the proposed algorithm.

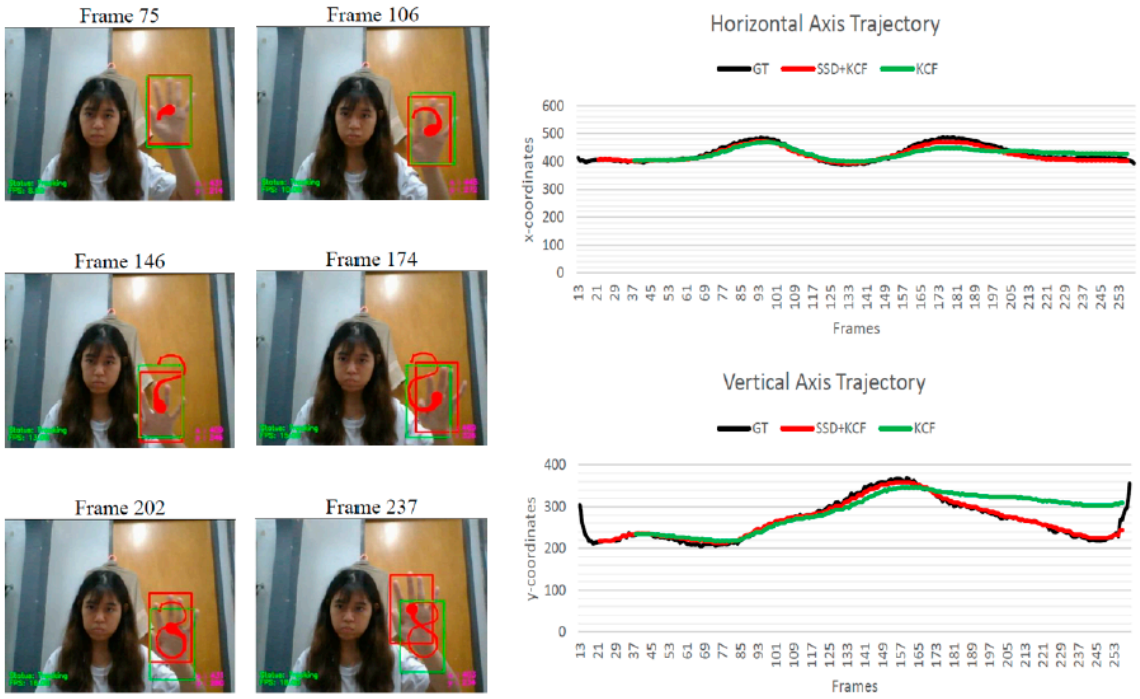

Figure 2. Tracking results of proposed algorithm (red) and KCF tracker (green) for gesture "8".

Analyzing the tracking results depicted in Figure 2, it becomes evident that the KCF tracker encountered drifting issues, as its bounding box drifted towards the arm when the hand continuously moved upwards, as shown in Frame 202 and Frame 237. On the other hand, the proposed algorithm exhibits excellent tracking results without any drifting issues. Moreover, both the horizontal and vertical axis trajectories of the proposed algorithm closely align with the ground truth trajectories. In contrast, the vertical axis trajectory of the KCF tracker deviates significantly from the ground truth trajectory.

The proposed tracking algorithm was further evaluated using several challenging video sequences, including scenarios involving slow and fast hand movements, occlusion behind other objects, outdoor environments, cluttered backgrounds with low contrast to skin tone, and deformable hand shapes while drawing alphabetical trajectory gestures. The evaluation results, along with a comparison to the original KCF tracker and the improvement achieved by the proposed algorithm, are summarized in Table 3. Based on the obtained results, it is evident that the overall performance of the proposed algorithm surpasses that of the KCF tracker. The proposed algorithm achieves a higher average tracking rate of 93.406% and a lower tracking error of 26.199. It is worth noting that the integration of the SSD algorithm has also improved the overall performance of the KCF tracker, resulting in a 22.802% improvement in TRDR and a 24.841% improvement in OTE value.

Table 3. Performance comparison between the proposed algorithm and KCF tracker on seven video sequences with challenging condition.

Tracking Scenes	Frame Num.	KCF + SSD (Proposed)			KCF			Improvement (%)	
		TRDR (%)	OTE	FPS	TRDR (%)	OTE	FPS	TRDR	OTE
Slow movement	546	98.26	26.609	19.0	96.32	27.607	25.0	2.014	3.613
Fast movement	664	84.01	30.341	16.0	42.97	23.893	33.0	95.509	−26.987
Occlusion behind object	708	82.50	21.213	18.0	89.44	73.907	32.0	−7.759	71.298
Outdoor scene	310	98.37	35.802	26.0	81.71	76.753	21.0	20.389	53.354
Clothes with less contrast to skin tone	597	94.95	30.729	18.0	65.16	26.383	27.0	45.718	−16.472
Deformable hand while drawing gesture “Q”	384	98.30	29.123	16.0	95.45	59.681	22.0	2.986	51.201
Deformable hand while drawing gesture “X”	310	97.45	9.575	16.0	96.72	15.415	30.0	0.757	37.882
Average		93.406	26.199	18.4	81.11	43.375	27.1	22.802	24.841

To gain a deeper understanding of the evaluation results, a qualitative analysis was conducted on specific tracking scenes. In the case of tracking slow hand movements, both the proposed algorithm and the KCF tracker exhibit similar performance. However, when it comes to handling fast and abrupt hand movements, the proposed algorithm outperforms the KCF tracker. The proposed algorithm achieves a higher tracking rate of 84.01%, which is twice that achieved by the KCF tracker. This indicates that the proposed algorithm is more effective at accurately tracking the hand during fast movements. While the proposed algorithm does have a relatively higher tracking error (OTE) of 30.341, it demonstrates a lower tendency of tracking failure compared to the KCF tracker. This can be observed in Figure 3, specifically in Frame 209 and Frame 309, where the proposed algorithm successfully keeps up with the fast-moving hand while the KCF tracker encounters tracking failure issues and fails to reinitialize its system. Furthermore, it was noted that the trajectory plots obtained from the proposed tracker are more complete compared to the implementation of the KCF tracker alone. It is important to acknowledge that, although the proposed algorithm tracks the hand object throughout a larger number of frames, offset errors between the estimated position and the ground truth tend to propagate. On the other hand, the KCF tracker fails to track the hand much earlier, resulting in lesser propagation of offset errors.

In the case of an occlusion-handling scenario, although the tracking rate of the proposed algorithm is relatively lower, its accuracy surpasses that of the KCF tracker, resulting in a lower tracking error. By examining the tracking results in Figure 4, it becomes apparent that both the horizontal and vertical axis trajectories of the KCF tracker deviate significantly from the ground truth due to tracking failures and severe drifting issues. Despite the KCF tracker managing to track the heavily occluded hand, its bounding box drifts towards the occluded object and fails to recover even when the hand reappears in that region. In contrast, the proposed algorithm demonstrates the ability to recover from drifting issues by reinitializing the system, as illustrated in Frame 142 to Frame 160 in Figure 4. Although tracking a heavily occluded hand remains challenging, the overall performance of the proposed tracker in handling occlusion has shown significant improvement.

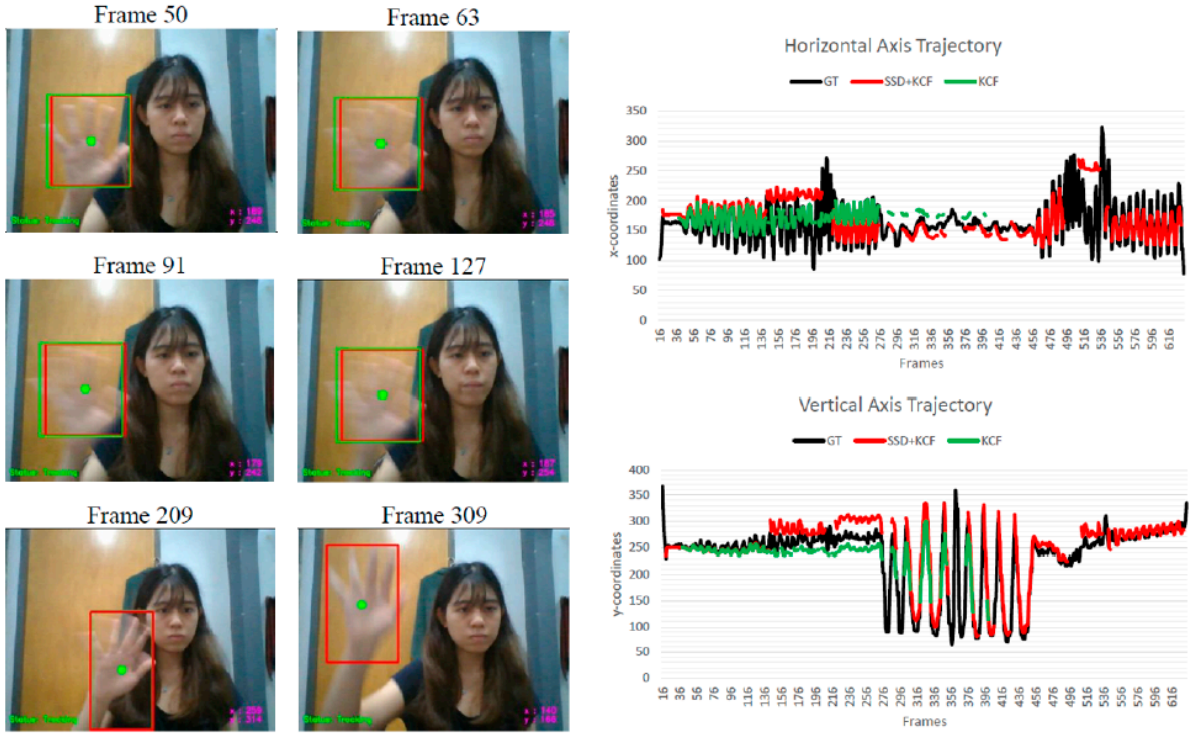

Figure 3. Tracking results of proposed algorithm (red) and KCF tracker (green) for tracking fast hand movement.

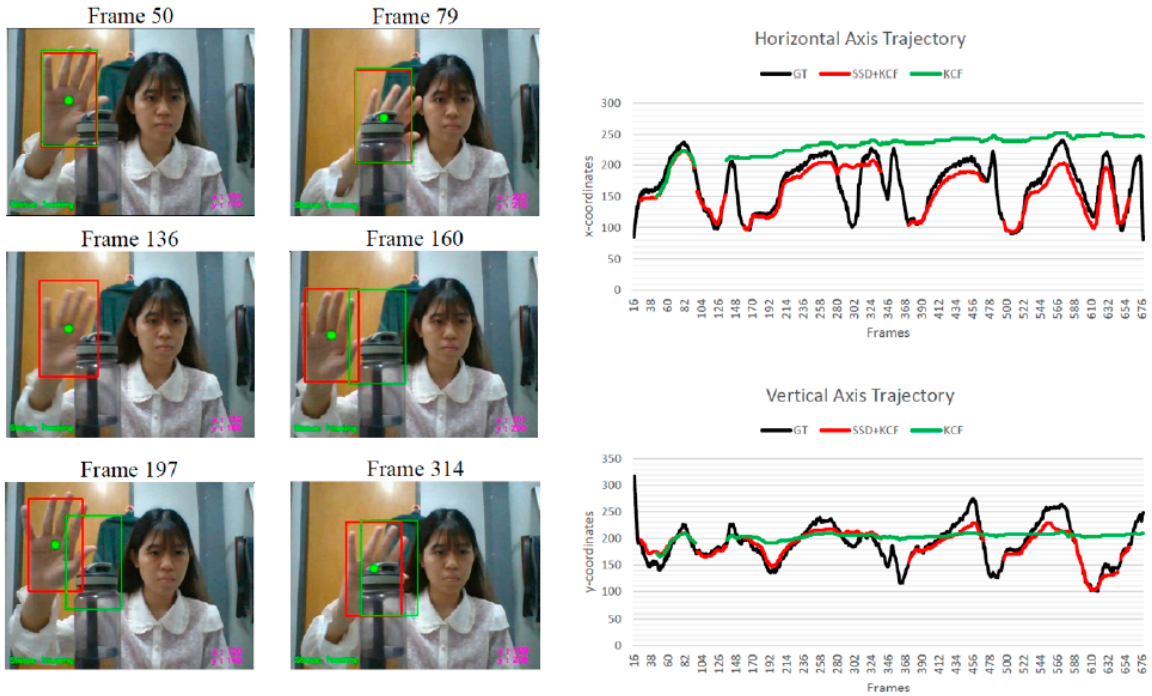

Figure 4. Tracking results of proposed algorithm (red) and KCF tracker (green) on occluded hand behind other object.

In the case of outdoor conditions with exposed lighting, the proposed algorithm demonstrates significantly better performance compared to the KCF tracker. However, as shown in Figure 5, it can be observed that the tracking bounding box of the proposed algorithm gets stuck in a region with visual characteristics similar to the skin features (Frame 155). When the hand rapidly moves away from this confusing region, the tracker

loses its tracking position. Nevertheless, the proposed algorithm quickly recovers from this loss by reinitializing the tracking position with the assistance of the SSD algorithm. In contrast, the KCF tracker recovers from the tracking loss much later in the sequence when the hand approaches the previously lost location, relying on the correlation filter mechanism.

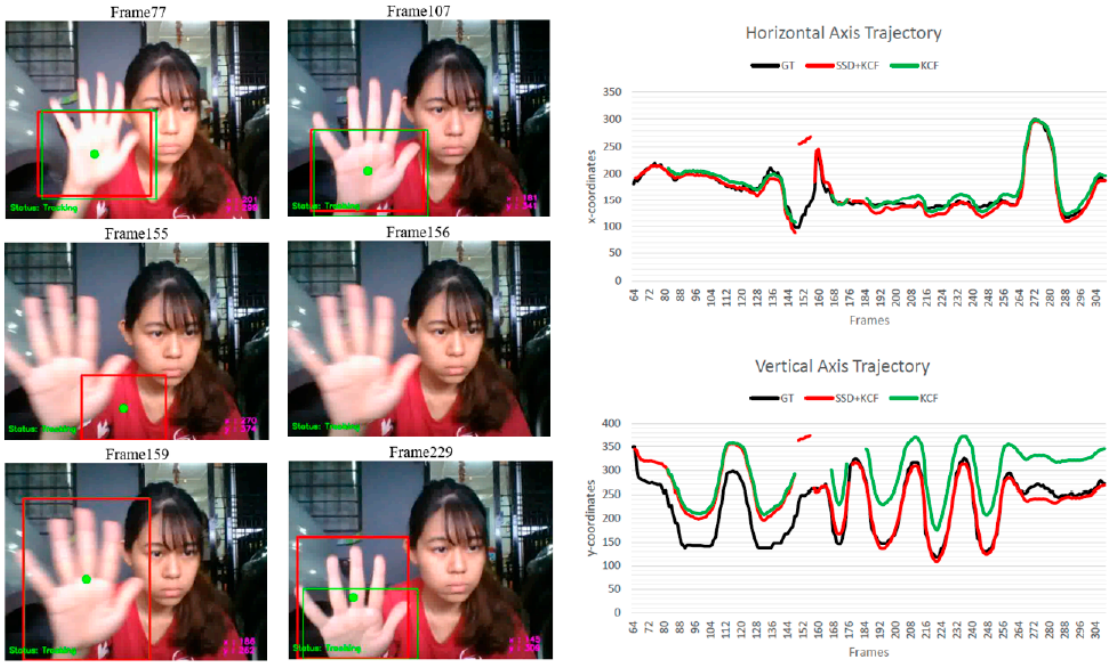

Figure 5. Tracking results of proposed algorithm (red) and KCF tracker (green) on outdoor scene.

For the case of tracking deformable hand shapes while drawing gesture "Q", the proposed algorithm outperforms the KCF tracker, achieving a higher tracking rate of 98.30% and a relatively lower tracking error of 29.1236. Although both algorithms are able to track the hand accurately with minimal failures, the KCF tracker experiences significant drifting issues, as observed in its tracking results depicted in Figure 6 (Frame 308 and Frame 344). When the hand continuously moves downwards, the bounding box of the KCF tracker drifts and becomes stuck in the background, which shares a similar visual feature with the hand. The KCF tracker fails to recover from this issue throughout the video sequence, resulting in heavy deviations in both the horizontal and vertical axis trajectories and nearly doubling the obtained OTE value compared to the proposed algorithm. In contrast, the proposed algorithm greatly reduces the drifting issue through the error recovery framework facilitated by the integration of the SSD algorithm.

To evaluate the generalizability of the proposed approach, a comparison was made with a state-of-the-art tracker, the MediaPipe hand [16]. For this comparison, ten video sequences from the Intelligent Biometric Group Hand Tracking (IBGHT) dataset [34] were used. This dataset was chosen for the experiment as it provides dynamic hand trajectories along with ground truth data. Table 4 provides detailed information about each sequence.

Table 5 presents a summary of the tracking results obtained by the proposed method and the MediaPipe hand tracker on ten trajectory-based hand gestures from the IBGHT dataset. The proposed method achieved an average target detection and tracking rate (TRDR) of 94.38% across all gestures, surpassing the performance of the MediaPipe hand tracker, which achieved 93.06%. However, it is worth noting that the MediaPipe hand tracker exhibited a lower object tracking error (OTE) with an average value of 7.59, compared to 10.59 for the proposed method. This indicates that the MediaPipe hand tracker excelled in accurately localizing the hand objects, resulting in fewer errors in object boundaries and position estimation. Despite the higher OTE, the proposed method demonstrated superior tracking speed, achieving an average frames per second (FPS) of 17.8, compared to the MediaPipe hand tracker's 12.7 FPS. The higher TRDR achieved by the proposed

method highlights its effectiveness in detecting and tracking hands in the given video sequences, even in challenging environments with cluttered backgrounds and occlusions. In conclusion, the proposed KCF + SSD method outperforms the MediaPipe hand tracker in terms of overall tracking accuracy (TRDR) and tracking speed (FPS). However, it is important to consider that the MediaPipe hand tracker excels in precise localization of hand objects, leading to lower object tracking errors (OTE). Therefore, the choice between the two methods depends on the specific requirements of the application, considering the trade-off between accuracy and speed. Further optimizations can be explored to improve the proposed method's OTE while maintaining its superior tracking performance.

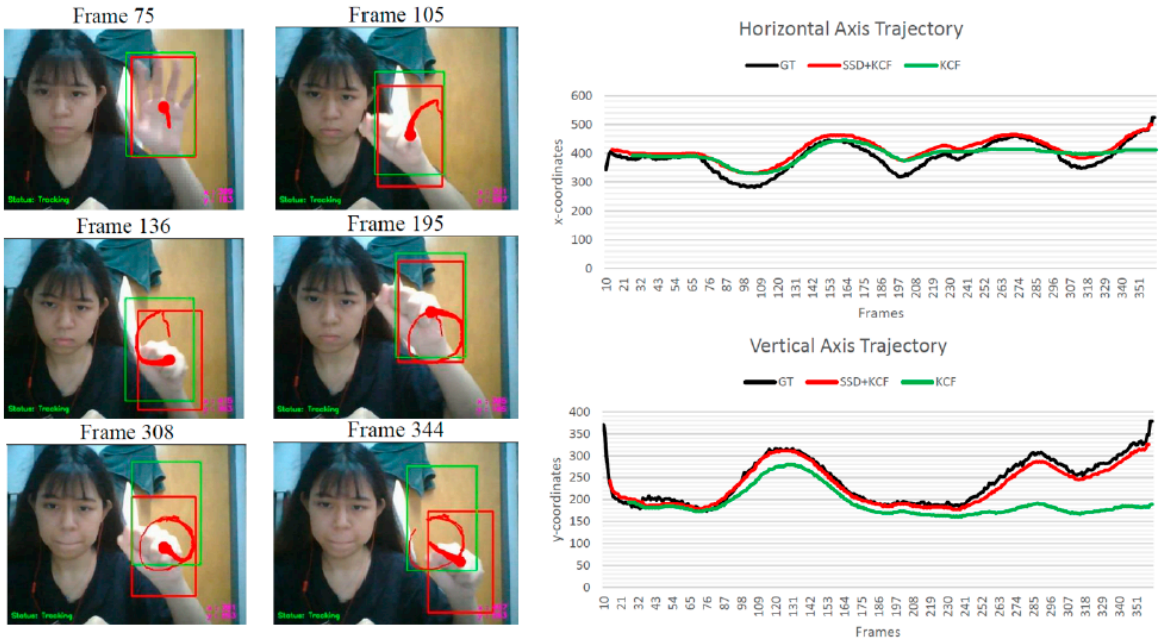

Figure 6. Tracking results of proposed algorithm (red) and KCF tracker (green) on deformable hand shape.

Table 4. Detailed properties of trajectory-based gestures from IBGHT dataset [34].

Trajectory Gesture	Num. of Frames	Hand Object Include Arm Region	Cluttered Background	Camera Distance to Object (Meter)
0	244	Yes	Cluttered	2
1	238	No	Partially	1
2	237	Yes	Partially	1.5
3	229	No	Partially	1
4	210	No	Partially	1
5	222	Yes	Partially	1
6	242	No	Cluttered	1.5
7	221	Yes	Cluttered	2
8	242	Yes	Cluttered	2
9	242	No	Partially	1

It should also be acknowledged that the proposed algorithm may encounter tracking failures, particularly when dealing with a fast-moving hand or a heavily occluded hand. Although the integrated SSD algorithm helps improve the tracking rate by reinitializing the system, it does not provide full support to the tracker during tracking. The absence of a motion-handling algorithm in the visual tracker poses challenges in accurately tracking fast-moving hands, where the target object may appear deformed or blurry to the tracker. Furthermore, the proposed algorithm has limitations in detecting and tracking small or distant hands due to the constraints of the SSD algorithm. The FPN Lite feature-extractor

utilized in the SSD algorithm consists of a bottom-up pathway, top-down pathway, and lateral connections. However, the semantic values in the lowest layer of the bottom-up pathway are relatively low, resulting in poor performance when detecting small or distant hands. Additionally, the proposed system is designed to detect and track only one hand to reduce system complexity. Consequently, it may not be suitable for implementation in systems that require the detection and tracking of multiple hands, such as hand gesture recognition systems where interpreting gestures often requires the presence of a pair of hands. It is essential to consider these limitations when evaluating the applicability of the proposed algorithm in various contexts and to explore potential enhancements or alternative approaches to address these challenges effectively.

Table 5. Performance comparison between the proposed algorithm and MediaPipe hand tracker on ten trajectory-based hand gestures from IBGHT hand tracking dataset [34].

Gestures	Frame Num.	KCF + SSD (Proposed)			MediaPipe Hand [16]		
		TRDR (%)	OTE	FPS	TRDR (%)	OTE	FPS
0	244	94.26	9.463	18.0	93.9	5.465	11.0
1	238	98.3	10.781	19.0	97.9	6.871	12.0
2	237	96.6	12.389	16.0	93.3	5.878	15.0
3	229	90.3	12.516	20.0	89.52	6.875	13.0
4	210	95.2	11.032	17.0	93.8	8.456	12.0
5	222	95.5	10.725	18.0	91.4	7.678	14.0
6	242	97.5	9.950	17.0	93.4	9.447	14.0
7	221	96.8	11.102	16.0	96.4	8.890	11.0
8	242	88.8	7.388	18.0	90.1	6.475	11.0
9	242	90.5	10.599	19.0	90.9	9.864	14.0
Average		94.38	10.59	17.8	93.06	7.59	12.7

5. Conclusions

A tracking-by-detection algorithm, constructed by integrating the SSD algorithm with the KCF visual tracker, is developed to detect and track hands from color images sequences with cluttered background and exerting minimal constraints on the subject. The proposed algorithm was tested on 17 video sequences and the experimental results show that it is fast and robust for real-time applications, achieving a promising tracking rate of over 90% and overall frame rate of around 17 FPS.

Based on the hand detection analysis, it can be justified that the chosen SSD object-detection algorithm is more suitable to be employed in the proposed algorithm due to its higher frame rate and accuracy achieved, as compared to that of the Faster R-CNN algorithm. For hand tracking performance analysis, it can be concluded that the proposed algorithm is able to keep track of the hand seamlessly with a promising tracking rate and lower tracking error. Integration of the SSD algorithm has improved the KCF visual tracker in many aspects, including its long-duration tracking performance, tracking performance during occlusion, and reduced drifting tendency. However, the limitations of KCF tracker such as being unable to effectively track fast-moving hands and heavily occluded hands still remain, due to its intrinsic weakness which is that it solely relies on the single Histogram of Oriented Gradients (HOG) feature extracted during its initialization. In the future, SSD integration with different classes of visual trackers can be explored, such as implementing a Channel and Spatial Reliability Tracking (CSRT) tracker which is well-known for its higher accuracy at the cost of slower computation speed.

Although the proposed tracking-by-detection algorithms can reduce local resource computation for hand detection and tracking, the task remains challenging for researchers. Accuracy is often impacted by factors such as occlusion, fast-moving hands, and abrupt gesture changes. Integrating motion-handling features such as color can improve algorithm accuracy and reduce tracking loss. Retraining the SSD algorithm by adding new

layers for fine-tuning can improve hand detection accuracy and expand the dataset with accurate images.

Author Contributions: Conceptualization, M.N.H.M., M.S.M.A. and B.A.R.; Methodology, O.L.P.; Formal analysis, M.S.M.A. and O.L.P.; Investigation, O.L.P.; Resources, B.A.R.; Writing—original draft, O.L.P.; Writing—review & editing, M.N.H.M. and M.S.M.A.; Supervision, M.S.M.A.; Funding acquisition, M.N.H.M. All authors have read and agreed to the published version of the manuscript.

Funding: This research is supported by the Ministry of Higher Education Malaysia (MOHE) Fundamental Research Grant Scheme FRGS/1/2019/ICT04/UTHM/02/2, (K187) Universiti Tun Hussein Onn Malaysia and FRGS grant FRGS/1/2020/TK0/USM/02/13, Universiti Sains Malaysia.

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

References

1. Bhandari, A.; Prasad, P.; Alsadoon, A.; Maag, A. Object detection and recognition: Using deep learning to assist the visually impaired. *Disabil. Rehabil. Assist. Technol.* **2021**, *16*, 280–288. [CrossRef]
2. Skarga-Bandurova, I.; Siriak, R.; Biloborodova, T.; Cuzzolin, F.; Bawa, V.; Mohamed, M.; Samuel, R. Surgical Hand Gesture Prediction for the Operating Room. *Stud. Health Technol. Inform.* **2020**, *273*, 97–103.
3. Gangrade, J.; Bharti, J. Vision-based Hand Gesture Recognition for Indian Sign Language Using Convolution Neural Network. *IETE J. Res.* **2020**, *39*, 723–732. [CrossRef]
4. Huang, Y.; Liu, K.; Lee, S.; Yeh, I. Evaluation of a Hybrid of Hand Gesture and Controller Inputs in Virtual Reality. *Int. J. Hum. –Comput. Interact.* **2021**, *37*, 169–180. [CrossRef]
5. Pisharady, P.K.; Saerbeck, M. Recent methods and databases in vision-based hand gesture recognition: A review. *Comput. Vis. Image Underst.* **2015**, *141*, 152–165. [CrossRef]
6. Erol, A.; Bebis, G.; Nicolescu, M.; Boyle, R.; Twombly, X. Vision-based hand pose estimation: A review. *Comput. Vis. Image Underst.* **2007**, *108*, 52–73. [CrossRef]
7. Xu, C.; Cai, W.; Li, Y.; Zhou, J.; Wei, L. Accurate hand detection from single-color images by reconstructing hand appearances. *Sensors* **2020**, *20*, 192. [CrossRef]
8. Wu, M.; Ting, P.; Tang, Y.; Chou, E.; Fu, L. Hand pose estimation in object-interaction based on deep learning for virtual reality applications. *J. Vis. Commun. Image Represent.* **2020**, *70*, 102802. [CrossRef]
9. Ahmad, A.; Migniot, C.; Dipanda, A. Hand pose estimation and tracking in real and virtual interaction: A review. *Image Vis. Comput.* **2019**, *89*, 35–49. [CrossRef]
10. Shin, J.; Kim, H.; Kim, D.; Paik, J. Fast and robust object tracking using tracking failure detection in kernelized correlation filter. *Appl. Sci.* **2020**, *10*, 713. [CrossRef]
11. Sharp, T.; Keskin, C.; Robertson, D.; Taylor, J.; Shotton, J.; Kim, D.; Rhemann, C.; Leichter, I.; Vinnikov, A.; Wei, Y. Accurate, robust, and flexible real-time hand tracking. In Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems, Seoul, Republic of Korea, 18–23 April 2015; pp. 3633–3642.
12. Khan, F.S.; Mohd, M.N.H.; Soomro, D.M.; Bagchi, S.; Khan, M.D. 3D hand gestures segmentation and optimized classification using deep learning. *IEEE Access* **2021**, *9*, 131614–131624. [CrossRef]
13. Sridhar, S.; Mueller, F.; Oulasvirta, A.; Theobalt, C. Fast and Robust Hand Tracking Using Detection-Guided Optimization. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 7–12 June 2015; pp. 3213–3221.
14. Mohd Asaari, M.S.; Rosdi, B.A.; Suandi, S.A. Adaptive Kalman Filter Incorporated Eigenhand (AKFIE) for real-time hand tracking system. *Multimed. Tools Appl.* **2015**, *74*, 9231–9257. [CrossRef]
15. Mueller, F.; Bernard, F.; Sotnychenko, O.; Mehta, D.; Sridhar, S.; Casas, D.; Theobalt, C. GANerated hands for real-time 3D hand tracking from monocular RGB. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–22 June 2018; pp. 49–59.
16. Zhang, F.; Bazarevsky, V.; Vakunov, A.; Tkachenka, A.; Sung, G.; Chang, C.L.; Grundmann, M. Mediapipe hands: On-device real-time hand tracking. *arXiv* **2020**, arXiv:2006.10214.
17. Liu, D.; Zhang, L.; Luo, T.; Wu, Y. Towards interpretable and robust hand detection via pixel-wise prediction. *Pattern Recognit.* **2020**, *105*, 107202. [CrossRef]
18. Gao, Q.; Liu, J.; Ju, Z. Robust real-time hand detection and localization for space human-robot interaction based on deep learning. *Neurocomputing* **2020**, *390*, 198–206. [CrossRef]
19. Arpit, M.; Andrew, Z.; Philip, T. Hand detection using multiple proposals. In Proceedings of the British Machine Vision Conference, Dundee, UK, 29 August–2 September 2011; Jesse, H., Stephen, M., Emanuele, T., Eds.; pp. 75.1–75.11.
20. Bambach, S.; Lee, S.; Crandall, D.; Yu, C. Lending A Hand: Detecting Hands and Recognizing Activities in Complex Egocentric Interactions. In Proceedings of the IEEE International Conference on Computer Vision, Santiago, Chile, 7–13 December 2015; pp. 1949–1957.

21. Mukherjee, S.; Ahmed, S.; Dogra, D.; Kar, S.; Roy, P. Fingertip detection and tracking for recognition of air-writing in videos. *Expert Syst. Appl.* **2019**, *136*, 217–229. [CrossRef]
22. Huang, Y.; Liu, X.; Zhang, X.; Jin, L. A pointing gesture based egocentric interaction system: Dataset, approach and application. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, Las Vegas, NV, USA, 26 June–1 July 2016; pp. 16–23.
23. Henriques, J.; Caseiro, R.; Martins, P.; Batista, J. High-speed tracking with kernelized correlation filters. *IEEE Trans. Pattern Anal. Mach. Intell.* **2015**, *37*, 583–596. [CrossRef]
24. Kalal, Z.; Mikolajczyk, K.; Matas, J. Tracking-learning-detection. *IEEE Trans. Pattern Anal. Mach. Intell.* **2011**, *34*, 1409–1422. [CrossRef]
25. Wang, Z.; Yoon, S.; Xie, S.J.; Lu, Y.; Park, D.S. Visual tracking with semi-supervised online weighted multiple instance learning. *Vis. Comput.* **2016**, *32*, 307–320. [CrossRef]
26. Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S.; Fu, C.; Berg, A. SSD: Single shot multibox detector. In Proceedings of the Computer Vision—European Conference on Computer Vision 2016, Amsterdam, The Netherlands, 11–14 October 2016; Lecture Notes in Computer Science. Leibe, B., Matas, J., Sebe, N., Welling, M., Eds.; Springer International Publishing: Berlin/Heidelberg, Germany, 2016; pp. 21–37.
27. Zhang, N.; Zhang, J. Optimization of face tracking based on KCF and Camshift. *Procedia Comput. Sci.* **2018**, *131*, 158–166.
28. Liu, C.; Yao, X.; Zhu, Z.; Peng, S.; Zheng, W. A robust tracking method based on the correlation filter and correcting strategy. In Proceedings of the 2017 International Conference on Image, Vision and Computing (ICIVC), Chengdu, China, 2–4 June 2017; pp. 698–702.
29. Lin, T.; Goyal, P.; Girshick, R.; He, K.; Doll'ar, P. Focal loss for dense object detection. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017; pp. 2980–2988.
30. Lin, T.; Doll´ar, P.; Girshick, R.; He, K.; Hariharan, B.; Belongie, S. Feature pyramid networks for object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 2117–2125.
31. Padilla, R.; Passos, W.; Dias, T.; Netto, S.; da Silva, E. A comparative analysis of object detection metrics with a companion open-source toolkit. *Electronics* **2021**, *10*, 279. [CrossRef]
32. Ren, S.; He, K.; Girshick, R.; Sun, J. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Trans. Pattern Anal. Mach. Intell.* **2016**, *39*, 1137–1149. [CrossRef] [PubMed]
33. Black, J.; Ellis, T.; Rosin, P. A novel method for video tracking performance evaluation. In Proceedings of the IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance, Nice, France, 11–12 October 2003; pp. 125–132.
34. Asaari, M.S.M.; Rosdi, B.A.; Suandi, S.A. Intelligent biometric group hand tracking (IBGHT) database for visual hand tracking research and development. *Multimed. Tools Appl.* **2014**, *70*, 1869–1898. [CrossRef]