IMPROVEMENT ON MOTION-GUIDED SIAMESE OBJECT TRACKING NETWORKS USING PRIORITIZED WINDOWS

A Thesis Submitted to
the Graduate School of Engineering and Science of
İzmir Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of

MASTER OF SCIENCE

in Computer Engineering

by
Ünver Can ÜNLÜ

July 2021
İZMİR

ACKNOWLEDGEMENTS

There were people who supported and helped me during this research. First, I would like to acknowledge my thesis advisor Assoc. Prof. Yalın BAŞTANLAR, especially for guiding me and his efforts throughout this research. In addition, my thesis jury members, Assoc. Prof. Mustafa ÖZUYSAL and Assoc. Prof. Fatih NAR helped me present better research thanks to their valuable feedbacks. Finally, I would like to thank my family and life-long friends for beside me always. This accomplishment would not have been possible without them.

ABSTRACT

IMPROVEMENT ON MOTION-GUIDED SIAMESE OBJECT TRACKING NETWORKS USING PRIORITIZED WINDOWS

In recent years, there has been significant progress in Visual Object Tracking with evolutions of both computers and learning algorithms, especially in Neural Networks. Therefore, we obtain better results by combining Neural Networks and traditional tracking methods such as Kalman Filter and Correlation Filters. SiamFC is an example of such algorithms because SiamFC combines Siamese Neural Networks and Correlation Filters. SiamFC is open to development because it does not have an online learning process. An example of the improved SiamFC is Kalman-Siam that combines Kalman Filter and Multi-feature SiamFC. Kalman-Siam uses Kalman-Filter to solve the occlusion situation problem by processing the target's previous motion trajectory. Therefore, the tracking can fail in other complex scenarios for Kalman-Siam. One of the methods for solving such problems is detecting this situation and starting the re-tracking process as we used in this research. Also, we used a parameter calculated on the response map after the correlation operation in SiamFC to detect these situations. First, our algorithm generates possible prioritized search windows. Then, it runs in a specific order of priority for these generated search windows surrounding the target's last known location. We named this process Adaptive Window Search that starts from the highest priority search windows and continues until the lowest search windows do not exist. Therefore, we named our algorithm Adaptive-Kalman-Siam. We demonstrated more successful results on commonly used datasets. Adaptive-Kalman-Siam tracks an object better than SiamFC and Kalman-Siam in Background Clutters, Fast Motion, Motion Blur, and Occlusion complex tracking scenarios.

ÖZET

NESNE TAKİBİNDE KULLANILAN HAREKET-YÖNLENDİRMELİ SİYAM AĞLARINDA ÖNCELİKLENDİRİLMİŞ PENCERELER İLE İYİLEŞTİRME

Son yıllarda hem Yapay Sinir Ağlarının hem öğrenme algoritmalarının gelişimi ile Görsel Nesne İzlemede önemli ilerlemeler kaydedilmiştir. Bu sayede Kalman Filtresi ve Korelasyon Filtreleri gibi geleneksel obje takip yöntemleri ile Yapay Sinir Ağları birleştirerek daha iyi sonuçlar elde ediyoruz. SiamFC, bu tür algoritmalara bir örnektir. Çünkü SiamFC, Siyam Yapay Sinir Ağlarını ve Korelasyon Filtrelerini birleştirir. Bununla birlikte, SiamFC geliştirilmeye oldukça açıktır. Bunun nedeni, SiamFC'nin obje takibi sırasında bir öğrenme sürecine sahip olmamasıdır. Kalman-Siam bu alanda iyileştirilmiş SiamFC'ye bir örnektir. Kalman-Siam, Kalman Filtresi ile Çok Katmanlı SiamFC'yi birleştirir. Kalman-Siam, takip edilen objenin önceki hareketini işleyerek objelerin üst üste gelme durumu problemini çözmek için Kalman Filtresini kullanır. Kalman-Siam için diğer zorlu senaryolarda takip işlemi başarısız olabilir. Bu tür sorunları çözmenin yollarından biri de bu durumu tespit etmek ve yeniden takip sürecinin başlatılmasıdır. Bu araştırmada bu yöntemi kullandık. Ayrıca bu durumları tespit etmek için algoritmamızda SiamFC'de korelasyon işlemi sonrası sonuç üzerinde hesaplanan bir parametre kullandık. İlk önce, algoritmamız önceliklendirilmiş arama pencereleri oluşturur. Ardından algoritma, yeniden takip için hedefin bilinen son konumunu çevreleyerek oluşturulan bu arama pencereleri için belirli bir öncelik sırasına göre çalışır. Yüksek öncelikli arama pencerelerinden başlayıp arama pencereleri kalmayana kadar bu yeniden takip sürecini Adaptive Window Search olarak adlandırdık. Bu nedenle algoritmamıza Adaptive-Kalman-Siam adını verdik. Yaygın olarak kullanılan veri setleri üzerindeki sonuçlarla daha başarılı bir yeniden takip süreci gözlemledik. Bu araştırmada önerilen Adaptive-Kalman-Siam, bir nesneyi Arka Plan Karmaşaları, Hızlı Hareket, Hareket Bulanıklığı ve Üst Üste Gelme gibi karmaşık takip senaryolarında SiamFC ve Kalman-Siam'dan daha iyi obje takibi yapmaktadır.

TABLE OF CONTENTS

LIST OF FIGURES

Figure **Page**

Figure **Page**

LIST OF TABLES

Table **Page**

CHAPTER 1

INTRODUCTION

This chapter will provide a brief and general introduction to Visual Object Tracking and give the aim of the thesis. First, we describe the steps of the object tracking process. Then, we classify the tracking algorithms fundamentally according to three different characteristics. Finally, we describe complex scenarios, which researchers try to solve, in Visual Object Tracking.

1.1. Visual Object Tracking

Visual Object Tracking is a task identifying a region of interest within an image sequence or a video. It is a sub-field of Computer Vision, and there are many critical real-world applications. For example, robotics, video surveillance, and autonomous vehicles. Researchers pay attention to Visual Object Tracking because it is open to new inventions. The performance and efficiency of tracking algorithms depend on many factors. Visual Object Tracking has some difficulties in complex tracking scenarios. Researchers try to solve them using especially Machine Learning, Image Processing, and Deep Learning.

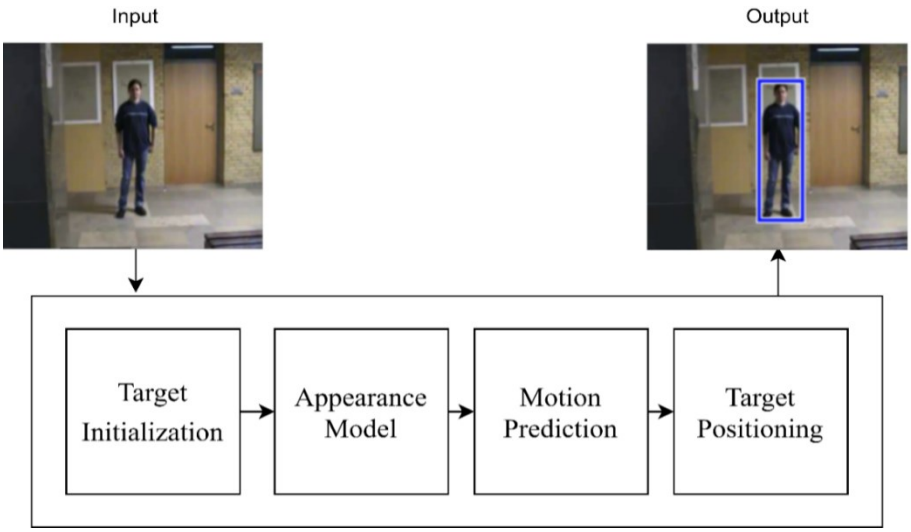


Figure 1.1 The blocks of traditional Visual Object Tracker design

According to Fiaz et al. [1], there are four sequential steps in the Visual Object Tracking process: target initialization, appearance model, motion prediction, and target positioning. The first step is target initialization which is the task of annotating the region of interest. This annotation is the target representation of the region of interest. In many ways, it can be bounding box, ellipse, centroid, skeleton, contour, or silhouette in Figure 1.2. The second step is appearance modeling, representing the region of interest as features and building a mathematical model to detect targets in the image sequence and learn how to detect them. The third step is motion prediction, and it is calculating the target's position changes in the current image. The final step is target positioning is determining the target position in the current image. Figure 1.1 shows the visualization of the processes of object tracking step by step in order.

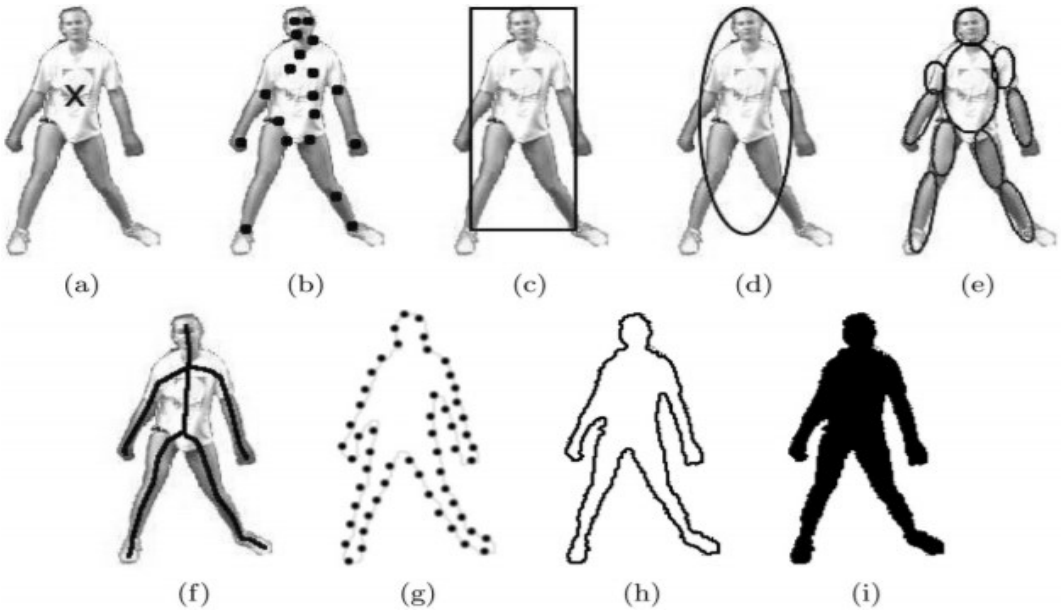


Figure 1.2 Object representations [2] (a) centroid, (b) multiple points, (c) rectangular patch, (d) elliptical patch, (e) part-based multiple patches, (f) object skeleton, (g) complete object contour, (h) control points on object contour, (i) object silhouette

1.2. Classification Of Tracking Approaches

We can classify Visual Object Tracking algorithms in three main differences: features, principles, and the number of tracked objects in this section. This chapter describes each of these classifications in detail.

1.2.1. Classification By Features

According to Fiaz et al. [1], we can divide Visual Object Tracking algorithms into two groups which feature the algorithm uses. The first is Handcrafted (HC), and the other is Deep-Learning-based features. For example, Histogram of Oriented Gradients (HOG), Scale-Invariant Feature Transform (SIFT), and color could be Handcrafted features. These are the most common ways to represent target views after achieving successful results in various fields, including Image Classification, Object Detection, and Image Segmentation. However, researchers have started using Deep-Learning-based features in recent years. Generally, Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN), and Residual Neural Networks obtain Deep-Learning-based Features. However, more data is needed to train an object tracker whose model uses one of these neural networks. Algorithms using Handcrafted features also give successful results, although Deep-Learning-based features achieve successful results.

1.2.2. Classification By Learning Approaches

According to learning approaches, we can divide Visual Object Tracking algorithms into Online Tracking and Offline Tracking. In Online Tracking, the object tracking algorithm has an online learning process that adapts itself to the target's situation changes while tracking. In Offline Tracking, there is not any model updating while tracking. Depending on the number of the calculation of the online learning process, Online Tracking can have a time constraint in the application.

1.2.3. Classification By The Number Of Tracked Objects

According to the number of tracked objects, we can divide Visual Object Tracking algorithms into two groups: Single-Object Tracking (SOT) and Multiple-Object Tracking (MOT). Although MOT is like the multiplication of SOT as a tracking process, specific algorithms can have lower execution times and achieve better results.

Table 1.1 Complex Visual Object Tracking scenarios from OTB dataset [3]

Name	Abbreviation	Definition
Illumination Variation	IV	The illumination in the target region is significantly changed.
Scale Variation	SV	The ratio of the bounding boxes of the first frame and the current frame is out of the range ts, ts > 1 (ts=2).
Occlusion	OCC	The target is partially or fully occluded.
Deformation	DEF	Non-rigid object deformation.
Motion Blur	MB	The target region is blurred due to the motion of the target or camera.
Fast Motion	FM	The motion of the ground-truth is larger than tm pixels (tm=20).
In-Plane Rotation	IPR	The target rotates in the image plane.
Out-of-Plane Rotation	OPR	The target rotates out of the image plane.
Out-of-View	OV	Some portion of the target leaves the view.
Background Clutters	BC	The background near the target has a similar color or texture as the target.
Low Resolution	LR	The number of pixels inside the ground-truth bounding box is less than tr (tr =400).

1.3. Challenges Of Visual Object Tracking

Although Visual Object Tracking algorithms have remarkable results, there are some difficulties in real-time applications such as loss of information in image projection, low image quality, the target's uncertainties, movement, illumination variation, and object occlusion. These difficulties make complex tracking scenarios, and they affect the performance and accuracy of the object tracker. Nevertheless, tracking algorithms try to handle these complex scenarios. Table 1.1 shows these complex scenarios provided by the OTB dataset [3] for their sequences.

1.4. Motivation And Aim Of The Study

We can use Siamese Neural Networks in Visual Object Tracking algorithms. Although SiamFC [4] gives us a good solution as an object tracking algorithm based on Siamese Neural Networks, SiamFC is not an online tracking algorithm because SiamFC does not have any learning process during tracking. Also, there are many updated versions of SiamFC. For example, Kalman-Siam [5] proposes a combination of Kalman Filter [6], Multi-feature fusion [7], and SiamFC. Kalman-Siam uses the target's previous trajectory information thanks to Kalman Filter and *P* parameter calculation on the response map to solve Occlusion tracking scenarios. However, using the target's previous trajectory information is not enough for other complex tracking scenarios such as Background Clutters, Fast Motion, Motion Blur. Also, there is not a re-tracking mechanism for neither SiamFC nor Kalman-Siam. Therefore, we develop a re-tracking mechanism for Kalman-Siam. This mechanism generates possible prioritized search windows surrounding the target's last know location first, then finds a suitable search window from the generated ones in this search windows' priority order. We call this process Adaptive Window Search. To determine tracking failure, start Adaptive Window Search, and find the suitable search window, we use the *APCE* parameter on the response map. Also, we named our tracker as Adaptive-Kalman-Siam. This thesis proposes that the Adaptive-Kalman-Siam tracker solves other complex tracking scenarios by adding a re-tracking mechanism to Kalman-Siam alongside Kalman Filter. In summary, we make the following contributions:

- We developed a re-tracking mechanism for Kalman-Siam. First, this mechanism generates possible prioritized search windows surrounding the target's last know location, then finds a suitable search window by lowering the priority as it cannot find. We named this re-tracking mechanism as Adaptive Window Search.
- We use the *APCE* parameter on the response map to detect tracking failure situations, start the Adaptive Window Search process and find a suitable search window from the generated possible search window.

Our tracker performance achieved better results than Kalman-Siam in OTB-100 [3], TC-128 [8] datasets, and specially selected sequences from these datasets. Furthermore, the tracker can be run in real like Kalman-Siam.

1.5. Organization Of The Thesis

The organization of the rest of this thesis is as follows: Chapter 2 provides literature survey about object tracking, including Kalman Filter, Correlation Filters, and Siamese Neural Networks based approaches. Chapter 3 explains the proposed tracking method. Benchmarking on datasets is described in Chapter 4. Chapter 5 contains the benchmarking results and the analysis of the results. Finally, Chapter 6 gives conclusions.

CHAPTER 2

LITERATURE SURVEY

Many object tracking algorithms have successful results. We need to research Visual Object Tracking literature to understand these algorithms also before providing a solution. This chapter investigates Kalman Filter and common architectures of Correlation Filters and Siamese Neural Networks with applications such as SiamFC and Kalman-Siam tracking algorithms.

2.1. Kalman Filter

Rudolf Kalman invented the Kalman Filter in 1960. Kalman Filter is a recursive algorithm and uses previous state measurements of a system over time and predicts unknown state estimations using Linear Equation and assuming noise as Gaussian Distribution. Also, Kalman Filter has many variations in handling different scenarios and usage areas due to its simplicity and real-time operation. Some of the applications are navigation vehicles for radar tracking, finance for risk forecasting, signal processing. [9]

2.1.1. Kalman Filter Steps

Kalman Filter has two steps: Prediction and Update steps. First, the Prediction Step is also called Time Update Step. Briefly, the Prediction Step is responsible for forwarding the current state and error covariance using previous states. In the Prediction Step Equations formulas, x and P represent the current state and the error covariance. Second, the Update Step is also called the Corrector Step. The Update Step is responsible for improving estimates using Kalman Gain with newly obtained state measurements. In the Update Step Equations formulas, K and z represent The Kalman Gain and the Measured State, respectively. Also, this kind of algorithm is classified as Predictor-Corrector or Prediction-Update. [9]

We use terms in priori for the Prediction Step and posteriori for the Update Step estimates in the equations. The Prediction Step Equations are both Equation 2.1 and Equation 2.2. The Update Step Equations are both Equation 2.3, Equation 2.4, and Equation 2.5. In addition to these equations, Equation 2.6 is the Measurement model of the system.

$$\hat{x}_k^- = A\hat{x}_{k-1}^- + Bu_{k-1} \tag{2.1}$$

$$P_k^- = AP_{k-1}^- A^T + Q \tag{2.2}$$

$$K_k = P_k^- H^T (HP_k^- H^T + R)^{-1} \tag{2.3}$$

$$\hat{x}_k = \hat{x}_k^- + K_k(z_k - H\hat{x}_k^-) \tag{2.4}$$

$$P_k = (I - K_k H)P_k^- \tag{2.5}$$

$$z_k = Hx_k + v_k \tag{2.6}$$

2.1.2. Object Tracking Using Kalman Filter In 2D Space

We can use Kalman Filter implementation based on the Taylor Series estimation of Position, Velocity, and Acceleration formulas for Object Tracking in two-dimensional space. We assume that Acceleration is constant, so state variables are Position and Velocity on the x-axis and the y-axis, as shown in Newton Dot Notation in Equation 2.7. Briefly, the number of dots over a variable means its derivative respect to its dependent variables at the same level as the number of dots over itself in Newton Dot Notation.

$$\overrightarrow{x_k} = [x_k \quad y_k \quad \dot{x}_k \quad \dot{y}_k]^T \tag{2.7}$$

Both Equation 2.8 and Equation 2.9 are the Position formulas on the x-axis and y-axis. Both Equation 2.10 and Equation 2.11 are the Velocity formulas on the x-axis and y-axis as the Kinematic Equation depends on time.

$$X_k = x_{k-1} + \dot{x}_{k-1}\Delta t + \frac{1}{2}\ddot{x}_{k-1}(\Delta t)^2 \quad (2.8)$$

$$y_k = y_{k-1} + \dot{y}_{k-1}\Delta t + \frac{1}{2}\ddot{y}_{k-1}(\Delta t)^2 \quad (2.9)$$

$$\dot{x}_k = \dot{x}_{k-1} + \ddot{x}_{k-1}\Delta t \quad (2.10)$$

$$\dot{y}_k = \dot{y}_{k-1} + \ddot{y}_{k-1}\Delta t \quad (2.11)$$

We can write the Kinematic Equations as Linear Equation form in Equation 2.12. Later, we can simplify these equations as matrix multiplication form in Equation 2.13.

$$\overrightarrow{x_k} = \begin{bmatrix} x_{k-1} + \dot{x}_{k-1}\Delta t + \frac{1}{2}\ddot{x}_{k-1}(\Delta t)^2 \\ y_{k-1} + \dot{y}_{k-1}\Delta t + \frac{1}{2}\ddot{y}_{k-1}(\Delta t)^2 \\ \dot{x}_{k-1} + \ddot{x}_{k-1}\Delta t \\ \dot{y}_{k-1} + \ddot{y}_{k-1}\Delta t \end{bmatrix} \quad (2.12)$$

$$\overrightarrow{x_k} = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{k-1} \\ y_{k-1} \\ \dot{x}_{k-1} \\ \dot{y}_{k-1} \end{bmatrix} + \begin{bmatrix} \frac{1}{2}(\Delta t)^2 & 0 \\ 0 & \frac{1}{2}(\Delta t)^2 \\ \Delta t & 0 \\ 0 & \Delta t \end{bmatrix} \begin{bmatrix} \ddot{x}_{k-1} \\ \ddot{y}_{k-1} \end{bmatrix} \quad (2.13)$$

We can simplify this matrix multiplication form in the Prediction Step, as shown in Equation 2.1. And then, we get the State Transition Matrix *A* in Equation 2.14 and the Control Matrix *B* in Equation 2.15.

$$A = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.14)$$

$$B = \begin{bmatrix} \frac{1}{2}(\Delta t)^2 & 0 \\ 0 & \frac{1}{2}(\Delta t)^2 \\ \Delta t & 0 \\ 0 & \Delta t \end{bmatrix} \quad (2.15)$$

In the Measurement Equation shown in Equation 2.6, we can assume that we can measure only the Position on the x-axis and y-axis. So, the Process Noise Vector *V* is equal to zero, and we can write the Measurement Equation as Equation 2.16. Then we get the Transformation Matrix *H* in Equation 2.17.

$$\overrightarrow{z_k} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_k \\ y_k \\ \dot{x}_k \\ \dot{y}_k \end{bmatrix} + v^k \tag{2.16}$$

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \tag{2.17}$$

Table 2.1 Process Noise Covariance Q

	x	y	$\dot{x}$	$\dot{y}$
x	σ_x^2	0	$\sigma_x\sigma_{\dot{x}}$	0
y	0	σ_y^2	0	$\sigma_y\sigma_{\dot{y}}$
$\dot{x}$	$\sigma_{\dot{x}}\sigma_x$	0	$\sigma_{\dot{x}}^2$	0
$\dot{y}$	0	$\sigma_{\dot{y}}\sigma_y$	0	$\sigma_{\dot{y}}^2$

The Position and the Velocity on different axes are independent of each other. However, the Position and the Velocity on the same axes are dependent. Thus, we can generate the Process Noise Covariance *Q* according to Table 2.1 using only the Standard Deviations of Position and Velocity on the x-axis and y-axis. And then, we can write the Process Noise Covariance *Q* as matrix form as shown in Equation 2.18.

$$Q = \begin{bmatrix} \sigma_x^2 & 0 & \sigma_x\sigma_{\dot{x}} & 0 \\ 0 & \sigma_y^2 & 0 & \sigma_y\sigma_{\dot{y}} \\ \sigma_{\dot{x}}\sigma_x & 0 & \sigma_{\dot{x}}^2 & 0 \\ 0 & \sigma_{\dot{y}}\sigma_y & 0 & \sigma_{\dot{y}}^2 \end{bmatrix} \tag{2.18}$$

We can simplify the Process Noise Covariance Matrix *Q* using just the Standard Deviations of the Position and the Velocity replacing shown in Equation 2.19, Equation 2.20, Equation 2.21, and Equation 2.22. Then, we get Equation 2.23 after all replacing operations.

$$\sigma_x = \frac{1}{2}(\Delta t)^2 \sigma_a \tag{2.19}$$

$$\sigma_y = \frac{1}{2}(\Delta t)^2 \sigma_a \tag{2.20}$$

$$\sigma_{\dot{x}} = (\Delta t)\sigma_a \tag{2.21}$$

$$\sigma_{\dot{y}} = (\Delta t)\sigma_a \tag{2.22}$$

$$Q = \begin{bmatrix} \frac{1}{4}(\Delta t)^4 & 0 & \frac{1}{2}(\Delta t)^3 & 0 \\ 0 & \frac{1}{4}(\Delta t)^4 & 0 & \frac{1}{2}(\Delta t)^3 \\ \frac{1}{2}(\Delta t)^3 & 0 & (\Delta t)^2 & 0 \\ 0 & \frac{1}{2}(\Delta t)^3 & 0 & (\Delta t)^2 \end{bmatrix} \tag{2.23}$$

Table 2.2 Measurement Noise R

	x	y
x	σ_x^2	0
y	0	σ_y^2

We can create the Measurement Noise *R* according to Table 2.2 using the only Position Variance on the x-axis and y-axis. And then, we can write the Measurement Noise *R* as matrix form as shown in Equation 2.24.

$$R = \begin{bmatrix} \sigma_x^2 & 0 \\ 0 & \sigma_y^2 \end{bmatrix} \tag{2.24}$$

2.2. Correlation Filter Based Tracking

Correlation measures that how variables are related to each other. In Visual Object Tracking, correlation is the operation that is element-wise multiplication between the target's actual and predicted features. After a correlation operation, we get a response map. Generally, we apply a cosine window on the response map to get the peak values and smooth the other values. We define the location of the peak value as the target's new location. We can do correlation operations also in the Fourier domain. However, using Correlation Filters in Visual Object Tracking algorithms has drawbacks. The change of target scale and appearance (orientation and shape) can make the response value low. A better feature extraction method is needed to get better correlation results. [10]

The most significant examples of Correlation Filters based tracking algorithms are Minimum Output Sum of Squared Error (MOSSE) [11], Circulant Structure Kernel (CSK) [12], and Kernelized Correlation Filters (KCF) [13]. First, MOSSE introduced Correlation Filters into the Visual Object Tracking area. In MOSSE, the grayscale images of adjacent frames are transformed into the Fourier Domain. Then, MOSSE applies a correlation operation on these frames. MOSSE runs very fast, but MOSSE is not successful enough because MOSSE uses the grayscale features. Second, CSK uses a gaussian kernel to calculate the correlation operation on the grayscale images. The accuracy of CSK is higher than MOSSE, but CSK is not successful enough, like MOSSE. Third, KCF algorithm uses HOG features and generates samples using cyclic shifts surrounding the target area. KCF filters the response of the cyclic matrices. The accuracy of KCF is higher than others, and KCF runs fast. However, KCF has limitations because the search area is fixed. Nowadays, Correlation Filters are used more as an auxiliary element in Visual Object Tracking algorithms.

Deep Learning breakthrough positively affects Visual Object Tracking like Image Classification, Object Detection, and Instance Segmentation. Different neural networks such as CNN, RNN, and Siamese Neural Networks are used in Visual Object Tracking. Generic Object Tracking Using Regression Networks (GOTURN) [14], Recurrent YOLO (ROLO) [15], Simple Online and Realtime Tracking with a Deep Association Metric (DeepSORT) [16], and SiamFC [4] are the essential Deep Learning based Visual Object Tracking algorithms. First, GOTURN uses the architecture of the CaffeNet neural

network [17]. In GOTURN, the previous frame and the current frame pass through convolutional layers with the same weight. Later, the outputs of these convolutional layers are processed in fully connected layers. Then, GOTURN predicts the target's location within a search region. Second, ROLO combines a YOLO (You Only Look Once) [18] object detection neural network and Long-Short-Term Memory (LSTM) [19]. ROLO uses YOLO to detect objects and then predicts the target's location using LSTM recurrently. Third, DeepSORT extends Simple Online Real-Time Tracking (SORT) [20] tracking algorithms with Deep Learning. DeepSORT uses the Fast R-CNN (Region Based Convolutional Neural Networks) neural network [21] to detect objects. Then, the detected objects are matched with the previous frame information using Kalman Filter [6] and Hungarian Algorithm [22]. In recent works, Deep Learning is used to extract the target features or detect the target in Visual Object Tracking algorithms.

2.3. Siamese Neural Networks Based Tracking

Siamese Neural Networks get multiple inputs and generate a single output. The CNN part of the network has shared weights and applies the same weight on the inputs. The output is the similarity measurement of the inputs. Generally, Siamese Neural Networks take two inputs, and they are called Two-Channel Siamese Neural Networks. According to the learning aim, convolutional branches of the network could be changed, such as the AlexNet neural network [23] and the VGG neural network [24]. Siamese Neural Networks can be used in areas like duplicate detection, finding anomalies, and face recognition. In addition, there are successful Visual Object Tracking algorithms such as GOTURN [14], Siamese Instance Search for Tracking (SINT) [25], and SiamFC [4]. Section 2.3 includes the explanation of GOTURN, and Section 2.5 describes SiamFC. SINT processes the original target and the search region in Siamese Neural Networks. Then, a specific matching function removes the background from the search region and decides the target's new location. In recent years, Siamese Neural Networks have had good quantitative results in Visual Object Tracking [26].

2.4. Siamese Fully Convolutional Networks (SiamFC)

Bertinetto et al. [4] introduced Siamese Fully Convolutional Networks in 2016. SiamFC combines Siamese Neural Networks and Correlation Filters tracking algorithms. Following sections, the Siamese Neural Networks architecture of SiamFC, cross-correlation step in SiamFC, Hanning Window, scale update in SiamFC, and the limitations of SiamFC are explained.

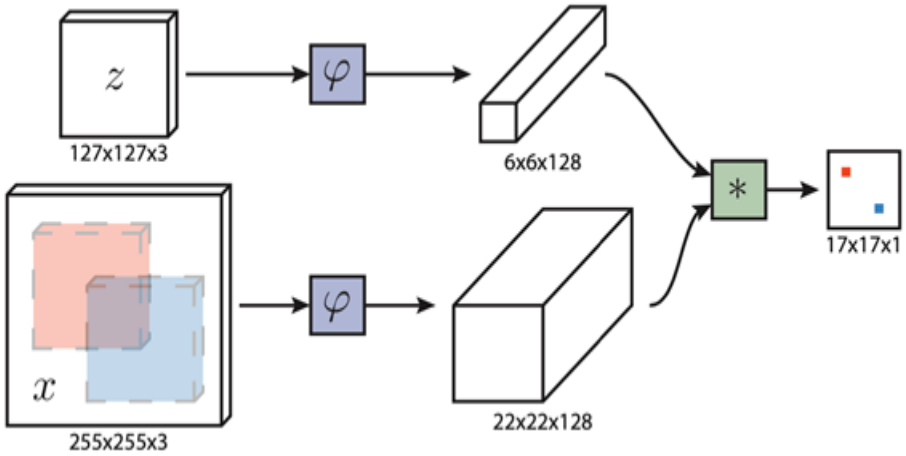


Figure 2.1 The architecture of SiamFC tracker [4]

2.4.1. The Architecture Of SiamFC

Figure 2.1 shows the architecture of SiamFC. The Siamese Neural Networks part of SiamFC is based on the AlexNet neural network [23] except for its convolution stage layers as a backbone. Table 2.3 shows these convolution stage layers in detail.

Table 2.3 The convolution stage of the AlexNet neural network [4]

Layer	Support	Chan. map	Stride	Activation size for exemplar	Activation size for search	chans.
				127 × 127	255 × 255	×3
conv1	11 × 11	96 × 3	2	59 × 59	123 × 123	×96
pool1	3 × 3		2	29 × 29	61 × 61	×96
conv2	5 × 5	256 × 48	1	25 × 25	57 × 57	×256
pool2	3 × 3		2	12 × 12	28 × 28	×256
conv3	3 × 3	384 × 256	1	10 × 10	26 × 26	×192
conv4	3 × 3	384 × 192	1	8 × 8	24 × 24	×192
conv5	3 × 3	256 × 192	1	6 × 6	22 × 22	×128

SiamFC takes two inputs which are Exemplar and Instance images. The difference between them is that the Instance image has 224 × 224 resolution, and the Exemplar image has 127 × 127 resolution. The convolution stage of the AlexNet neural network [23] extracts Deep-Learning-based features from the Exemplar and the Instance image by sharing the same weights. The Exemplar image is a target template, and it never changes during the tracking. However, the Instance image is a search image whose center is the target's center location. The Instance image area is always more extensive than the Exemplar image area.

Bertinetto et al. [4] employ a discriminative loss function approach. That means the pairs are divided into positive and negative pairs. Then the logistic loss of that pairs is calculated while training the neural network. Equation 2.25 calculates a pair's logistic loss where *y* is the ground-truth label, *v* is the real-value score. Equation 2.26 is the loss function, the mean of the individual losses where *D* is the score map. Exemplar and Instance images that are from the same sequence are ignored. Finally, the weights are updated using Stochastic Gradient Descent (SGD).

$$l(\mathrm{y},\mathrm{v}) = \log(1 + \exp(-\mathrm{yv})) \tag{2.25}$$

$$\mathrm{L}(\mathrm{y},\mathrm{v}) = \left|\frac{1}{\mathrm{D}}\right| \sum l(\mathrm{y}[\mathrm{u}],\mathrm{v}[\mathrm{u}]) \tag{2.26}$$

2.4.2. Cross-Correlation In SiamFC

After features are extracted from the inputs, SiamFC has the Correlation Filter head, which measures the similarity of two feature data thanks to cross-correlation operation. Equation 2.27 is the formula of the cross-correlation operation. *X* is the Search Image as the current frame, and *Z* is the Exemplar image as the target template, the ground-truth representation in dataset sequence in SiamFC. $\phi(\cdot)$ is the function for the feature representation, and $Corr(\cdot)$ is the correlation operation. $R(Z,X)$ denotes the similarity between the images. In SiamFC, the response map has 17 × 17 resolutions.

$$\mathrm{R}(\mathrm{Z},\mathrm{X}) = \mathrm{Corr}(\phi(\mathrm{Z}),\phi(\mathrm{X})) \tag{2.27}$$

2.4.3. Hanning Window

Hanning Window is applied on the response map to get the maximum value near the center and sharpen the maximum value after generating a response map. Hanning Window is a window Function, and window functions are generally used for signal processing and statistics. Hanning Window, also called the Hann Filter or the Raised Cosine Window, was invented by Julius von Hann. The formula of the Hanning Window is shown in Equation 2.28.

$$w(n) = 0.5 - 0.5\cos\left(\frac{2\pi n}{M-1}\right)\ 0 \le n \le M - 1 \qquad (2.28)$$

2.4.4. Locating The Target's New Location

After the Hanning Window is applied on the response map, we get a non-negative, smooth, and bell-shaped curve. There could be more than one peak on the result. However, the peak location whose maximum value is considered the target location in the current frame. Finally, an update processing step, including updating the target's location, shape, Instance, and Exemplar images' size, exists. Figure 2.2 shows the successful example of SiamFC.

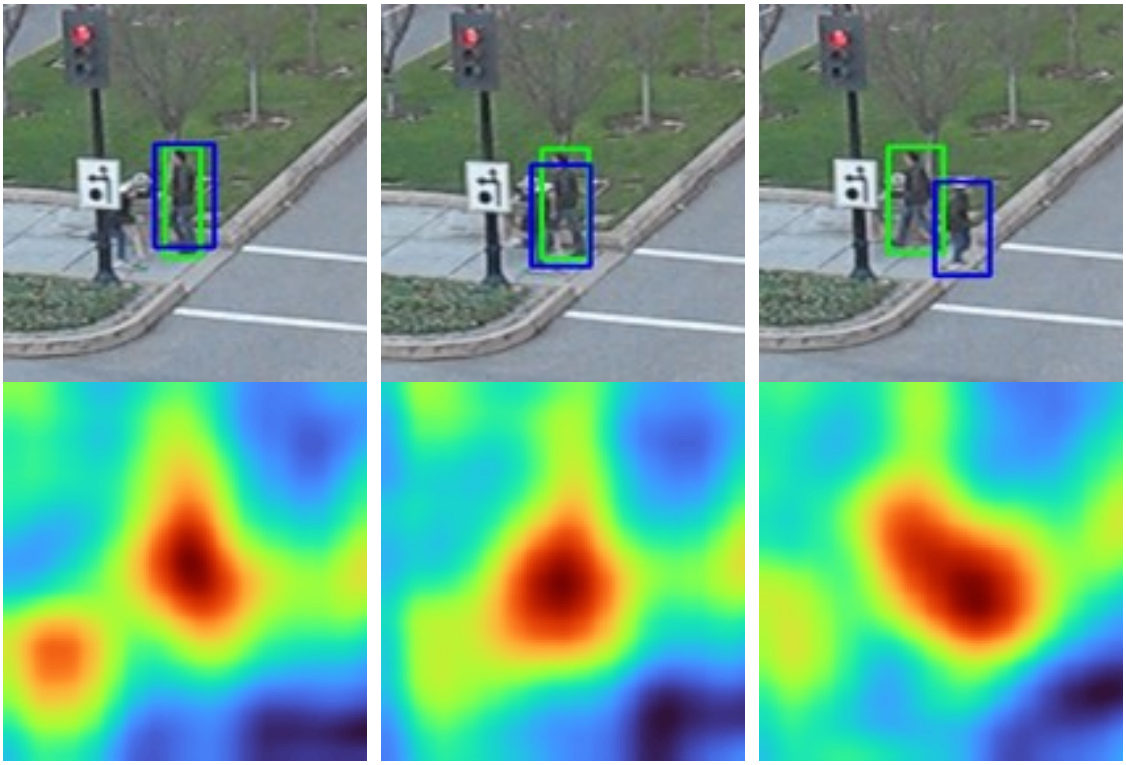
SiamFC has a scale updating process to handle the target's scale changes. Three different sizes of Instance images are processed to detect the scale change. It means there are three response maps after the cross-correlation operation. The scale of the response map, which has the highest peak value, is selected as the target's new scale.

Figure 2.2 Successful example of SiamFC tracker (a) exemplar image, (b) instance image, (c) result, (d) response map (green: ground-truth, blue: tracker result for the frame; red: high, blue: low for the heat map)

2.4.5. The Limitations Of SiamFC

Let us discuss the limitations of SiamFC. First, SiamFC does not have an online learning process. There is no control mechanism and learning a model according to the tracking situation during the object tracking process in SiamFC. Therefore, SiamFC always uses the same initial ground-truth at the first frame in the sequence as the target template. Second, Hanning Window creates a response map with the highest value near the center as the peak location when there are multiple peaks. Third, the target's trajectory is not considered while tracking. Thus, the target can be lost when an occlusion situation occurs, as shown in Figure 2.3.

In Figure 2.3, there are two objects which are a man, an actual target, and a boy riding a scooter in the Human3 sequence from the OTB-100 dataset [3]. An occlusion situation occurs at frame 28, and peaks of all objects are combined at that frame. Also, each object has a similar response after the cross-correlation operation, making the target lost in ongoing frames.



Frame 25 Frame 28 Frame 33

Figure 2.3 Failed example of SiamFC tracker. Results and response maps on *Human3* sequence for frames 25, 28, and 33 from the OTB-100 dataset using SiamFC tracker (green: ground-truth, blue: tracker result for frames; red: high, blue: low for heat maps)

2.5. Kalman-Siam

In recent years, because of the performance and speed of SiamFC, many follow-up studies based on SiamFC are proposed. Zhou et al. [5] introduced Kalman-Siam, an improved version of SiamFC, and Kalman-Siam solves some of the limitations of SiamFC. The following sections explain the improvements of Kalman-Siam, the architecture of Kalman-Siam, the occlusion detection mechanism in Kalman-Siam, and the limitations of Kalman-Siam.

2.5.1. The Improvements Of Kalman-Siam

There are three improvements in Kalman-Siam according to SiamFC. First, Kalman-Siam obtains the target's motion information, and Kalman Filter predicts the target's location in the next frame. Second, Kalman-Siam combines the different layers from the base network. This improvement increases the success rate because each network layer has a different response to the same input. Third, Kalman-Siam has an occlusion detection mechanism by using Kalman Filter and calculating the *P* parameter on the response map. Thanks to Kalman Filter, Kalman-Siam has a learning process based on the target's movement using the target's previous trajectory during tracking. This improvement makes Kalman-Siam an Online Tracking algorithm. According to the calculation, the box which Kalman Filter predicts can be selected.

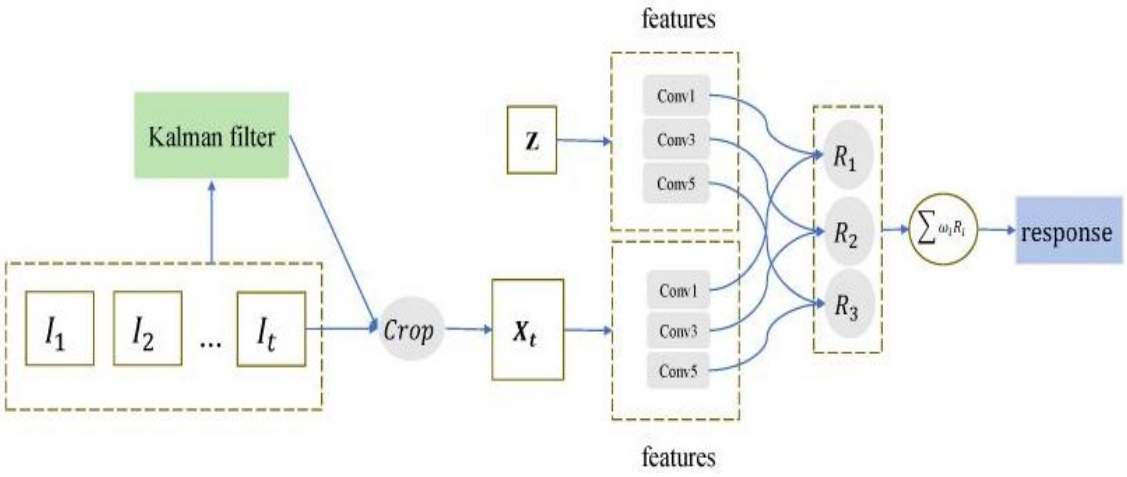


Figure 2.4 The architecture of the Kalman-Siam tracker [5]

2.5.2. The Architecture Of Kalman-Siam

The base neural network of Kalman-Siam and SiamFC are the same, and this is the AlexNet neural network [23]. Figure 2.4 shows the architecture of Kalman-Siam. However, there are two main differences in Kalman-Siam. First, although SiamFC crops the searched image according to the target's last know location, Kalman-Siam crops the searched image according to the prediction of Kalman Filter. Second, SiamFC applies cross-correlation operation on only the neural network outputs of Search and Exemplar image channels. However, Kalman-Siam applies cross-correlation operation on the

outputs from *conv1*, *conv3*, and *conv5* layer of the Search and Exemplar image channels with the same layer. And then, the weighted sum of these three response maps makes a single response map using Equation 2.29. *W* and *R* represent the layer's weight and the layer's response map in the equation. High-level features can adapt to the target's transformation. On the other hand, low-level features have importance on locating the target. Therefore, the weights of these layer outputs are 0.2, 0.2, and 0.6, respectively. [5]

$$\mathrm{R} = \sum_{\mathrm{i}}(\mathrm{W_i}\,\mathrm{R_i}) \tag{2.29}$$

2.5.3. The Occlusion Detection Mechanism In Kalman-Siam

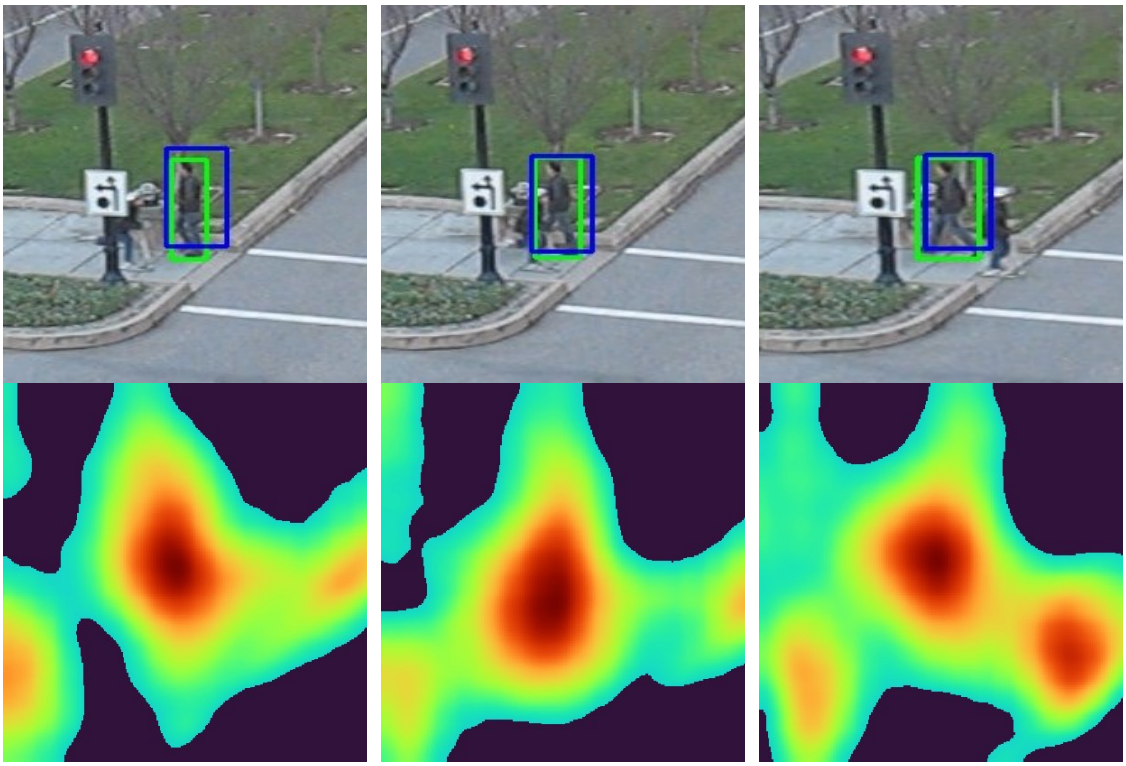
Kalman-Siam has a solution when the target encounters an occlusion situation. To solve the situation, Kalman-Siam firstly tries to detect whether this situation occurs or not. For the detection, Kalman-Siam uses some unique parameters, P, $P_{average}$, and P_{ratio}, shown in both Equation 2.30, Equation 2.31, and Equation 2.32.

$$\mathrm{P} = \frac{\mathrm{R_{peak}} - \mathrm{R_{low}}}{\mathrm{mean}\left(\frac{\mathrm{R}}{\mathrm{R_{mean}}}\right)} \tag{2.30}$$

$$\mathrm{P_{average}} = \frac{\mathrm{P(t-1)+P(t-2)+P(t-3)+P(t-4)}}{4} \tag{2.31}$$

$$\mathrm{P_{ratio}} = \frac{\mathrm{P}}{\mathrm{P_{average}}} \tag{2.32}$$

P measures how distinctive the peak is in the response map. $P_{average}$ takes the average for the last four frames. P_{ratio} normalizes the magnitude of P dividing by $P_{average}$. R represents the response map. The value of the P_{ratio} threshold is equal to 0.6. For better calculation, the value of the response map is normalized to between 0 and 1. Also, Kalman-Siam applies a threshold on the value of the response map with 0.3 to filter out the noise in the response map. Kalman-Siam calculates Equation 2.30, Equation 2.31, and Equation 2.32 and predicts the target's location using Kalman Filter. When complex tracking scenarios occur, the value of both P and P_{ratio} are decreased. If the value of the P_{ratio} is lower than the threshold value, Kalman-Siam selects the target's location as the prediction of Kalman Filter in the current frame in the dataset sequence.



Frame 25 Frame 28 Frame 34

Figure 2.5 Successful example of Kalman-Siam tracker. Results and response maps on *Human3* sequence for frames 25, 28, and 34 from the OTB-100 dataset using Kalman-Siam tracker (green: ground-truth, blue: tracker result for frames; red: high, blue: low for heat maps)

2.5.4. The Limitations Of Kalman-Siam

Even if Kalman-Siam has better results using the target's previous trajectory information thanks to Kalman Filter, some tracking processes can be challenging, such as rapid moves that change their direction to the opposite. For example, in Figure 2.6, there is a deer as the target is jumping. This sequence is tagged as the Fast Motion (FM) attribute, and the target moves fast along the y-axis. Kalman-Siam cannot handle this kind of movement. So, this target's movement makes its previous trajectory information insignificant. Also, there is another example in Figure 2.7. There is a man jumping rope in Figure 2.7, the target moves like the previous example. Moreover, the sequence is tagged as the Motion Blur (MB). Some frames are blurred, and the peak in their response map is not high enough. Therefore, the *P* parameter's value is low in these frames. Both motion blur and targets movement makes the P parameter insufficient to locate the target in the response map.

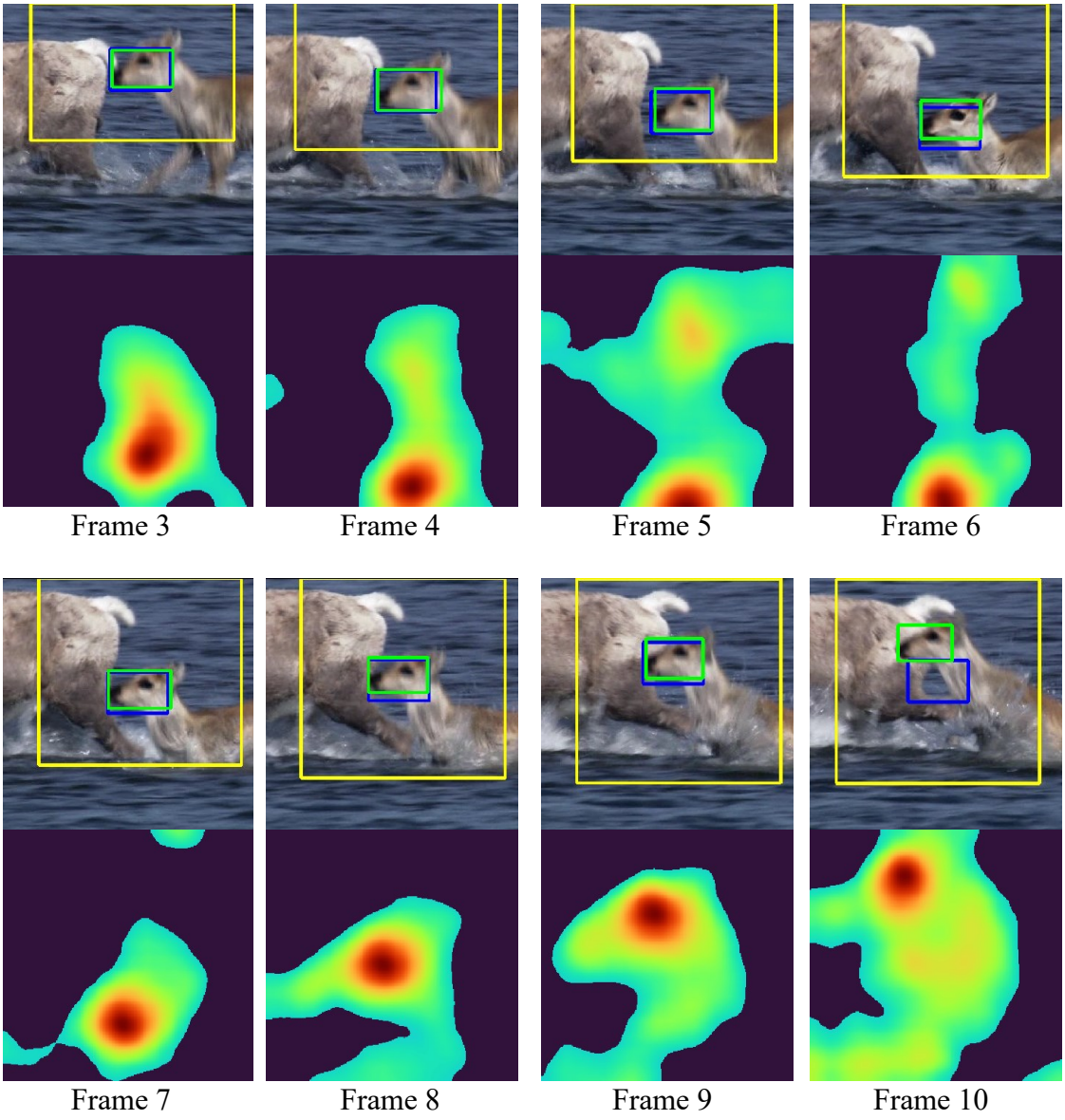


Figure 2.6 Failed example of Kalman-Siam tracker for Fast Motion labeled sequence. Results and response maps on *Deer* sequence for frames 3, 4, 5, 6, 7, 8, 9, and 10 from OTB-100 dataset using Kalman-Siam tracker (yellow: search window, green: ground-truth, blue: tracker result for frames; red: high, blue: low for heat maps)

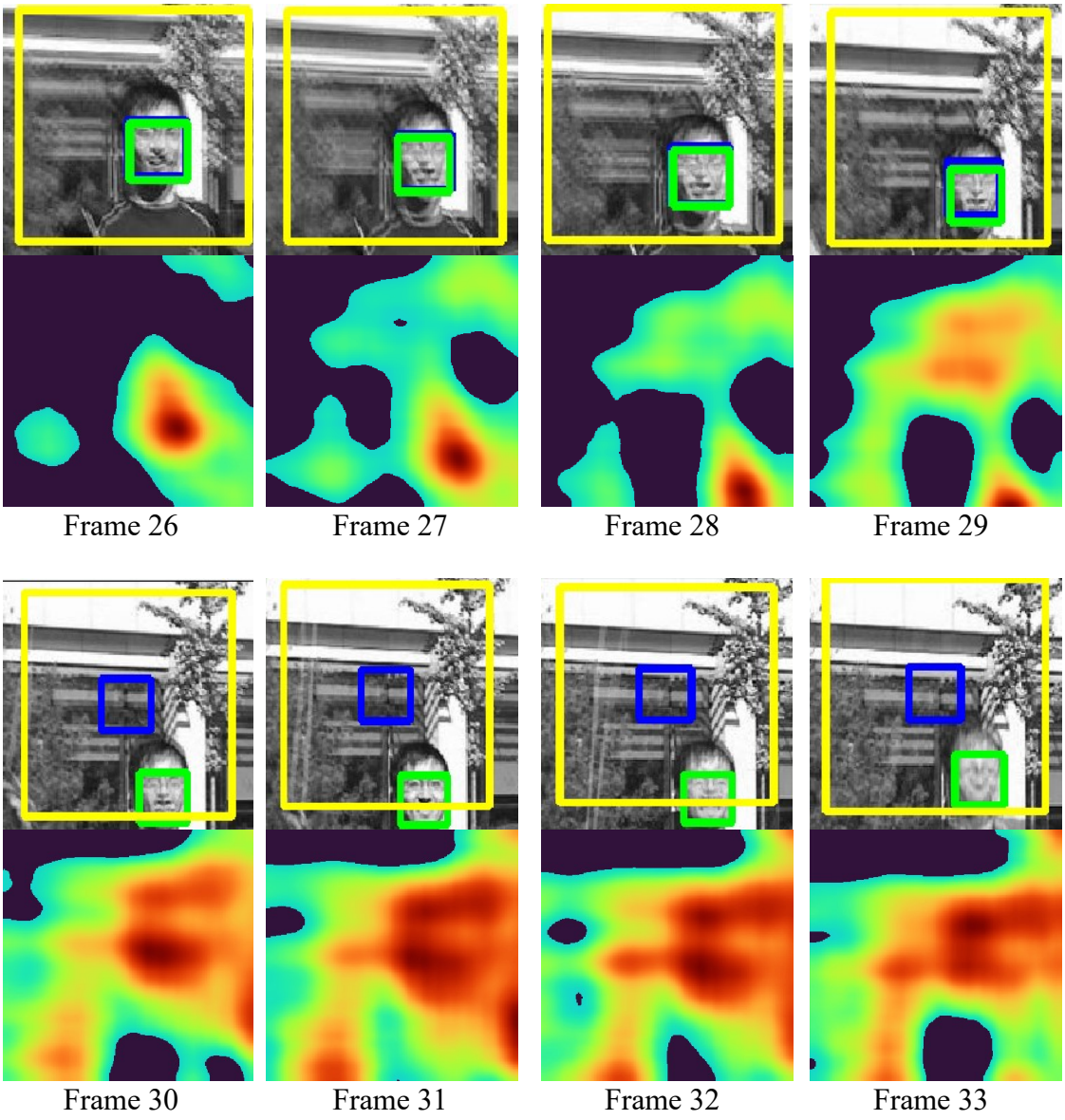


Figure 2.7 Failed example of Kalman-Siam tracker for Motion Blur labeled sequence. Results and response maps on *Jumping* sequence for frames 26, 27, 28, 29, 30, 31, 32, and 33 from OTB-100 dataset using Kalman-Siam tracker (yellow: search window, green: ground-truth, blue: tracker result for frames; red: high, blue: low for heat maps)

CHAPTER 3

PROPOSED METHOD

Bertinetto et al. [4] introduced SiamFC, and Zhou et al. [5] proposed Kalman-Siam, an updated version of SiamFC with Kalman Filter and multi-feature fusion. Although we can get better results using Kalman-Siam, there are limitations of Kalman-Siam, as mentioned in previous sections. Thus, we proposed an updated version of Kalman-Siam, including Adaptive Window Search and Kalman Filter, and replaced the *P* parameter with another parameter. We named this tracker as Adaptive-Kalman-Siam, and this chapter describes Adaptive-Kalman-Siam.

3.1. The Architecture Of Adaptive-Kalman-Siam

We developed Adaptive-Kalman-Siam based on Kalman-Siam. We kept the improvements of Kalman-Siam added to SiamFC. The first is estimating the target's location with its previous movement using Kalman Filter and determining the Search-Window based on this prediction. The second is obtaining a better response map by combining the features in different correlated layers from the neural network. The third is determining whether there exists an occlusion situation by calculating on the obtained response map. If the calculated value is below a certain threshold, Kalman-Siam selects the prediction of Kalman Filter as the target's new location.

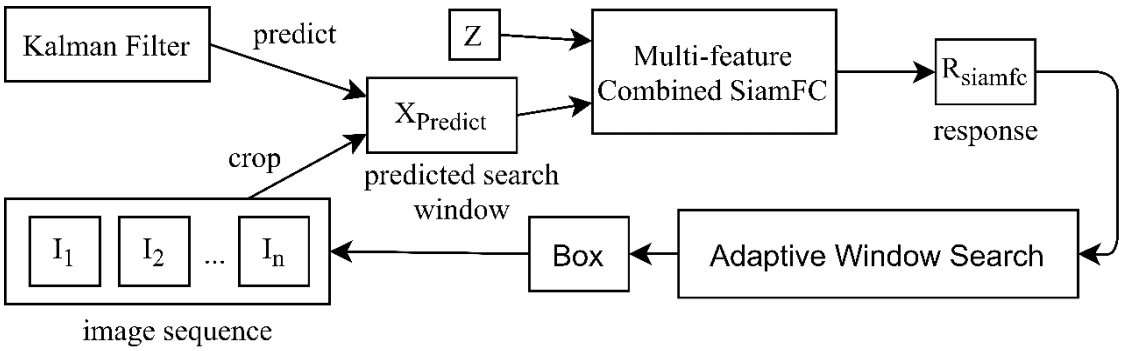


Figure 3.1 The architecture of the proposed Adaptive-Kalman-Siam tracker

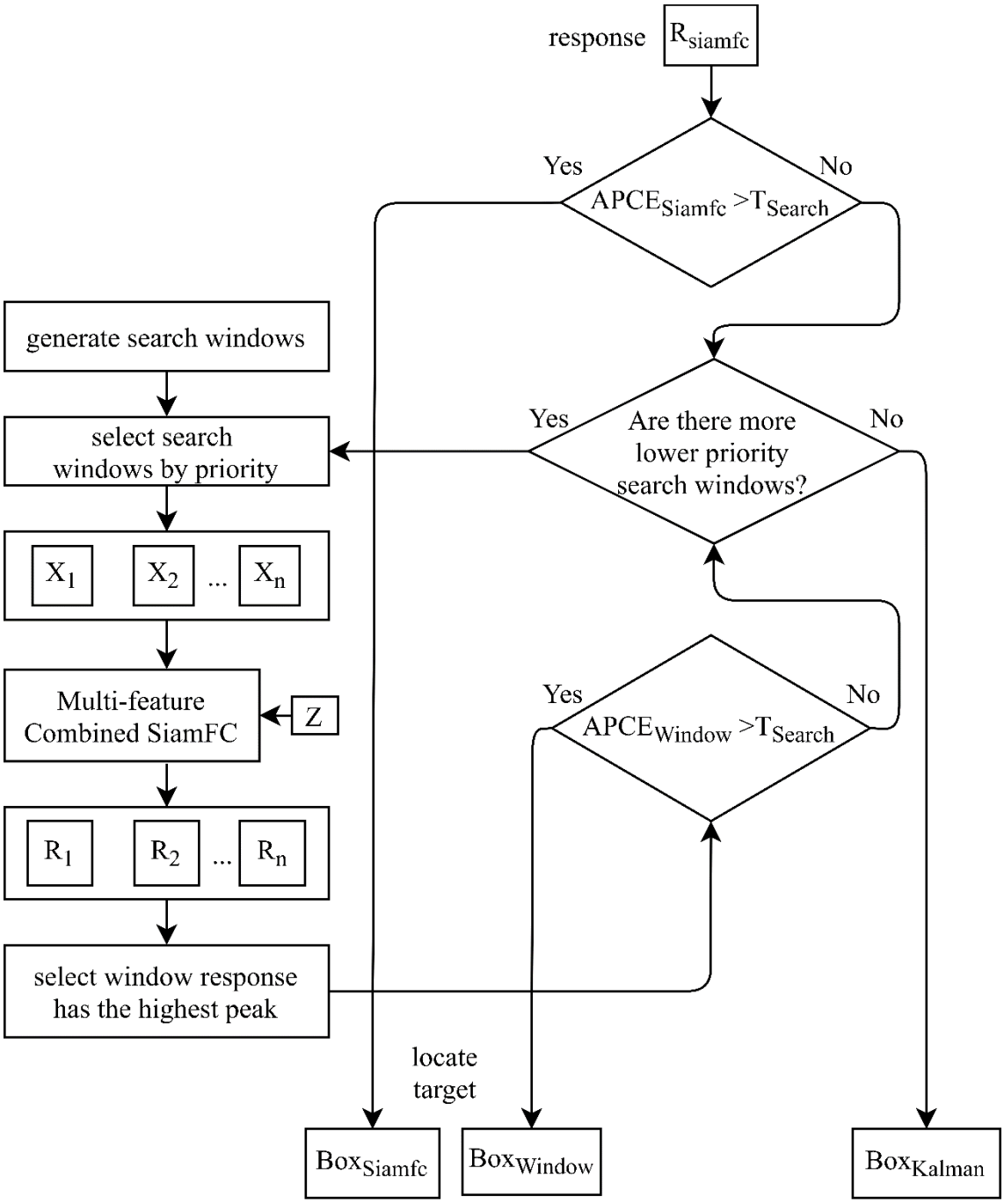


Figure 3.2 The flowchart of Adaptive Window Search algorithm

Figure 3.1 shows the architecture of Adaptive-Kalman-Siam. The updated part is demonstrated in the box named Adaptive Window Search. Figure 3.2 shows the flowchart of the Adaptive Window Search algorithm. Adaptive-Kalman-Siam uses T_{Search} threshold, and we set 0.7. We determined this threshold value after the experiment of parameter searching. Adaptive-Kalman-Siam decides to start or continue Adaptive Window Search using T_{Search} threshold. Suppose Adaptive-Kalman-Siam cannot obtain the processed search window successfully enough. In that case, it decides whether the estimated location of the Kalman Filter will be selected as the object's new location. The values to be used for comparison with this threshold are calculated as the *APCE* parameter. Also, we add another threshold which is T_{Kalman} and its value is set at 0.5 for storing the last *APCE* value and updating the Kalman Filter. This threshold is valid when Adaptive Window Search cannot find a suitable search window. Following Section 3.2 and Section 3.3 describe the *APCE* parameter and Adaptive Window Search.

3.2. Average Peak To Correlation Energy (APCE)

Wang et al. [27] introduced the Average Peak Correlation Energy (APCE) parameter. Equation 3.1 shows the formula of *APCE*. We used the *APCE* parameter similar to the *P* parameter of Kalman-Siam, so the tracker calculates $APCE_{average}$ and $APCE_{ratio}$ using the same formula for $P_{average}$ and P_{ratio}, shown in both Equation 3.2 and Equation 3.3.

$$\mathrm{APCE} = \frac{|\mathrm{F_{max}} - \mathrm{F_{min}}|^2}{\mathrm{mean}\left(\sum_{\mathrm{w,h}} (\mathrm{F_{w,h}} - \mathrm{F_{min}})^2\right)} \tag{3.1}$$

$$\mathrm{APCE_{average}} = \frac{\mathrm{APCE(t-1)} + \mathrm{APCE(t-2)} + \mathrm{APCE(t-3)} + \mathrm{APCE(t-4)}}{4} \tag{3.2}$$

$$\mathrm{APCE_{ratio}} = \frac{\mathrm{APCE}}{\mathrm{APCE_{average}}} \tag{3.3}$$

APCE indicates the fluctuated level of response maps and the confidence level of the detected target. $APCE_{average}$ takes the average for the last four frames. $\mathrm{APCE_{ratio}}$, normalizes the magnitude of *APCE* dividing by $APCE_{average}$. *F* represents the response map.

There are two main reasons why we prefer to use the *APCE* parameter rather than the *P* parameter. First, the value range of the *APCE* parameter is wider than the value range of the *P* parameter. Second, the *APCE* parameter has more usage than the *P* parameter in tracking algorithms based on Correlation Filters.

3.3. Adaptive Window Search

Shin et al. [28] implemented a re-tracking mechanism on the KCF [13] tracking algorithm. This algorithm detects tracking failure and then re-tracks the target with new search windows using the eight different neighboring windows. Adaptive-Kalman-Siam has a similar process, but there exist 25 search windows that can be distinct and intersected. Thus, there are too many search windows. To prevent performance loss, the processing of these search windows is a specific priority. We define that the window centered on the target's last location is the base search window. Table 3.1 shows these search windows with their priority and distance to the base search window. The search windows that are closer to the center of the base search window have higher priority. We grouped these search windows that have the same priority. All search windows have the same height and width based on the target's last location.

Table 3.1 The priority table of the search windows

Search Window Number	**Distance to the Last Location According to Search Window Size (w=Search Window width, h=Search Window height)**	**Priority**
5	(0,0)	Highest
10, 11, 12, 13, 14, 15, 16, 17	(±0.25 w, ±0.25 h)	High
18, 19, 20, 21, 22, 23, 24, 25	(±0.5 w, ±0.5 h)	Medium
1, 2, 3, 4, 6, 7, 8, 9	(±1 w, ±1 h)	Low

The peak value of a response map could exist near the border of a search window. However, optimally the peak value should be around the center of the search window. Applying the Hanning Window filter to a response map makes the high value near the peak value. Figure 3.3 shows the high-priority search windows. Therefore, intersected search windows in Adaptive-Kalman-Siam reduces the risk of smoothing the peak of the actual tracked target near the border of a search window. We defined points far from the center of the base search window at 25 percent and 50 percent of the dimensions of the base search window. We made these points the center point of the intersecting search windows. Both Figure 3.4 and Figure 3.5 show these center points of intersected search windows.

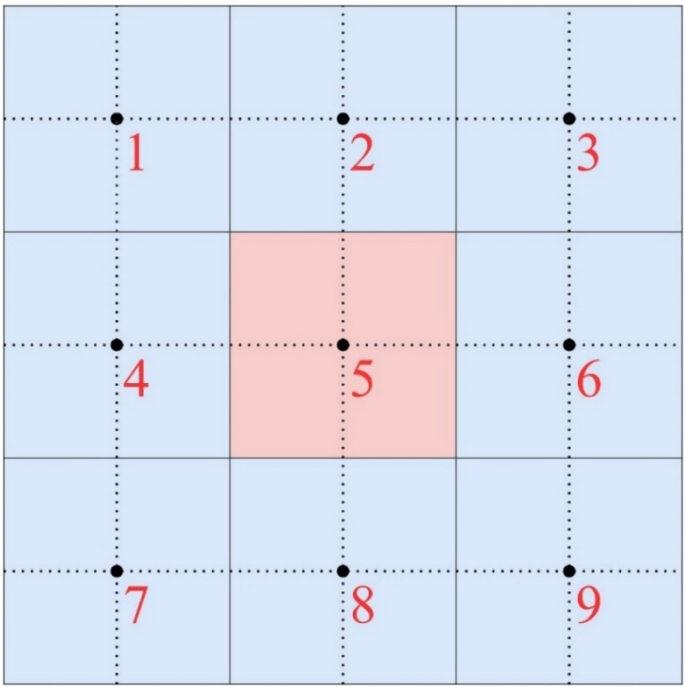


Figure 3.3 The distinct search windows. The center of the base search window (5) and the other search windows as the distinct neighbor of the base search window (1, 2, 3, 4, 6, 7, 8, 9)

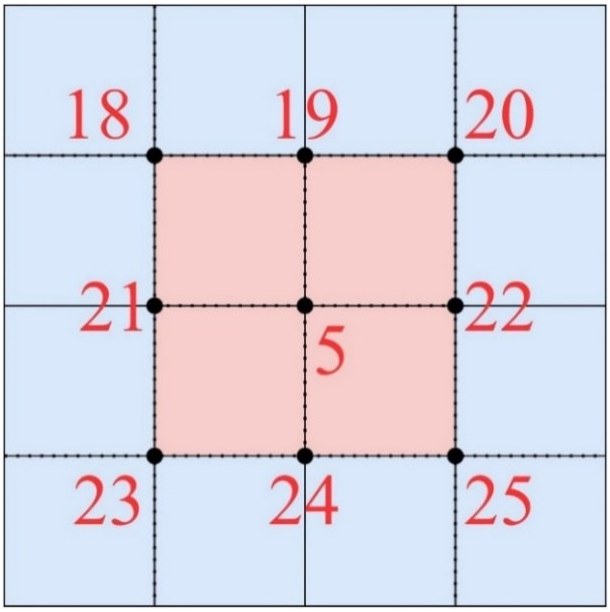


Figure 3.4 The center of the base search window (5), the center points of intersected search windows (18, 19, 20, 21, 22, 23, 24, 25) are away from the center of the base search window by 50 % of the height/width.

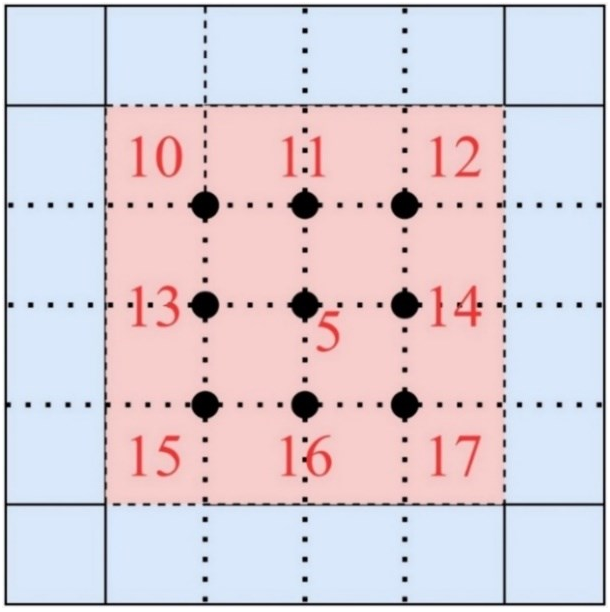


Figure 3.5 The center of the base search window (5), The center points of intersected search windows (10, 11, 12, 13, 14, 15, 16, 17) are away from the center of the base search window by 25% of the height/width.

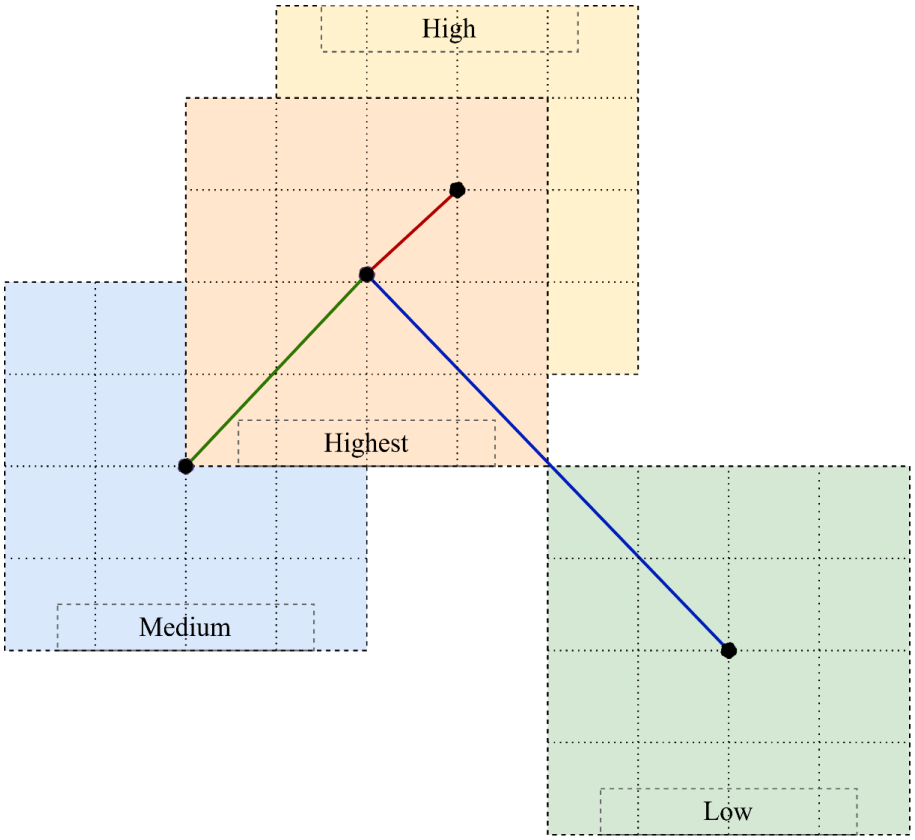


Figure 3.6 Distance from the target's last known location to each prioritized search window groups

Figure 3.6 shows how much the groups of prioritized search windows are far away from the target's last know location. The distances of each group are proportional to search window dimensions. In addition, Figure 3.6 allows us to see the ratios of distances between each other. As we can see, the re-tracking possibility of the proposed Adaptive Kalman-Siam increases when the distance increases. However, using more possible search windows decrease the general execution time performance of the re-tracking algorithm. Therefore, we limit the size of possible search windows.

Figure 3.7 shows that Adaptive-Kalman-Siam started the Adaptive Window Search process at frame 10, then choosing search window 5. At that frame, although the $APCE_{ratio}$ of the Kalman-Siam is 0.66, the $APCE_{ratio}$ of the search window with number 5 is 0.86. Therefore, the Adaptive Window Search process is finished, and the target was not lost.

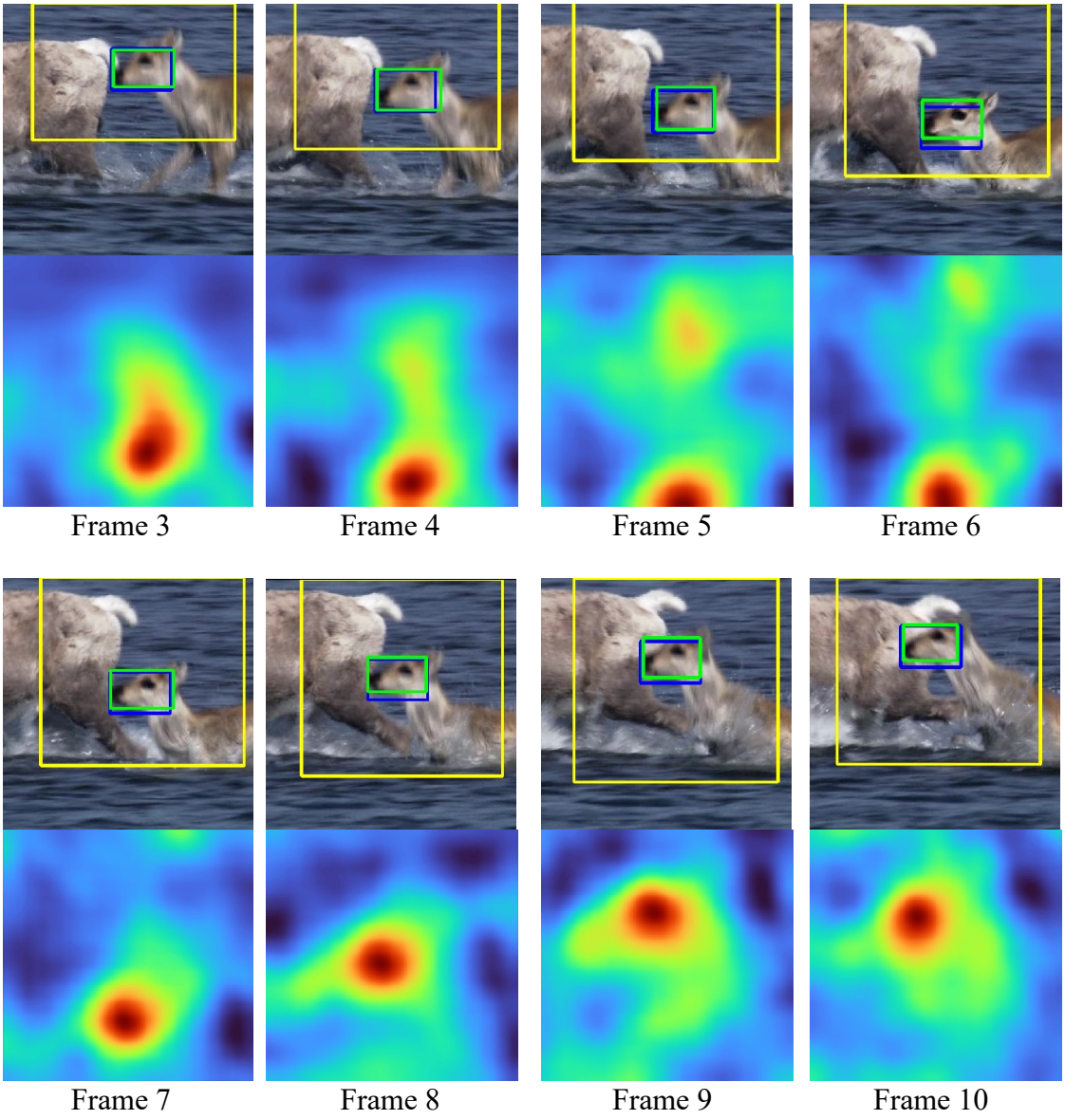


Figure 3.7 Example of Adaptive-Kalman-Siam tracker for Fast Motion labeled sequence. Results and response maps on *Deer* sequence for frames 3, 4, 5, 6, 7, 8, 9, and 10 from OTB-100 dataset using Adaptive-Kalman-Siam tracker (yellow: search window, green: ground-truth, blue: tracker result for frames; red: high, blue: low for heat maps)

The Adaptive Window Search process is started at frame 30 in Figure 3.8, then search window 5 is selected. When the $APCE_{ratio}$ of the Kalman-Siam is 0.51, the $APCE_{ratio}$ of the search window with number 5 is 0.90. Thus, the Adaptive Window Search process is finished, and tracking is continuing.

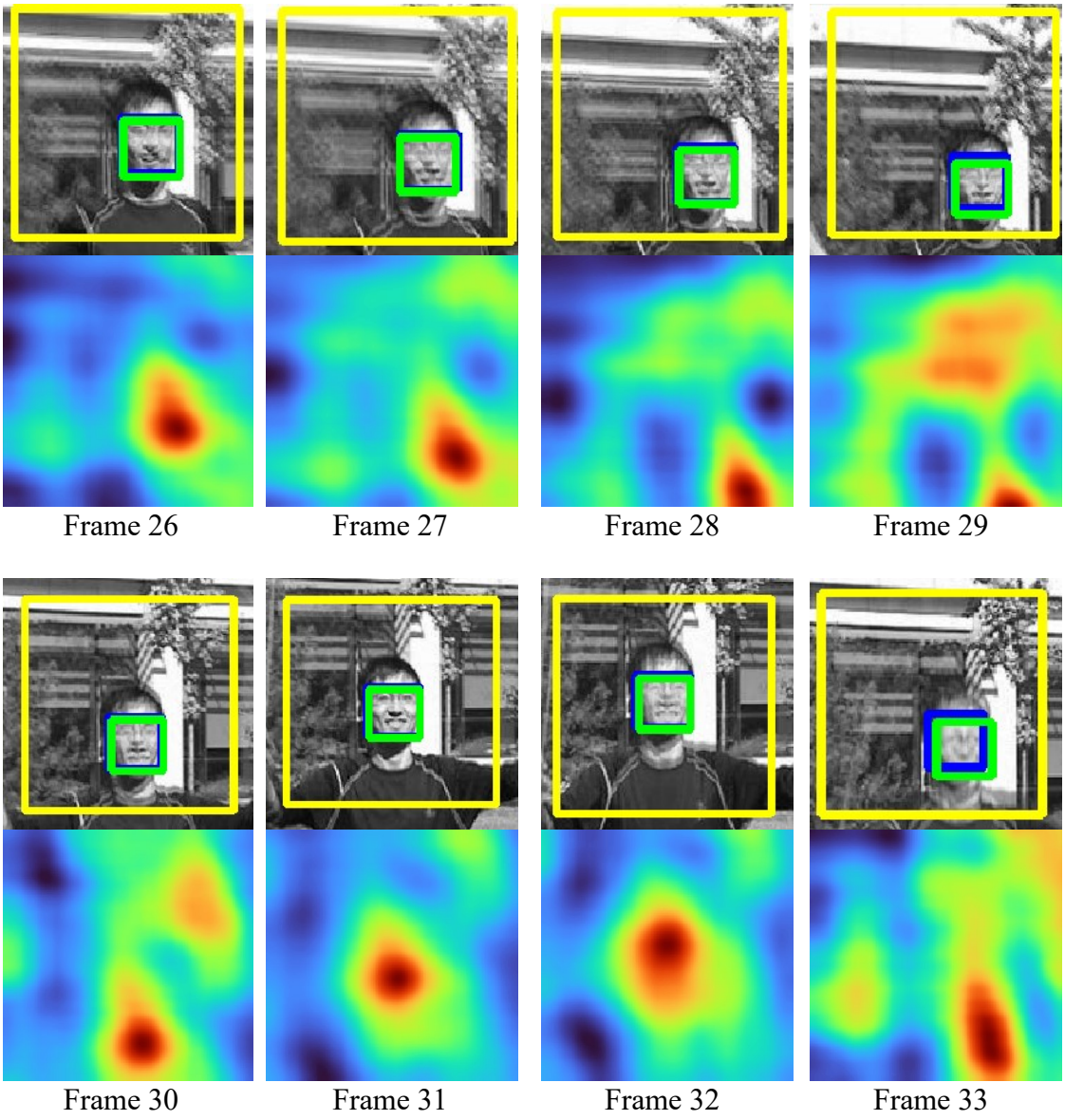


Figure 3.8 Example of Adaptive-Kalman-Siam tracker for Motion Blur labeled sequence. Results and response maps on *Jumping* sequence for frames 26, 27, 28, 29, 30, 31, 32, and 33 from OTB-100 using Adaptive-Kalman-Siam tracker (yellow: search window, green: ground-truth, blue: tracker result for frames; red: high, blue: low for heat maps)

CHAPTER 4

BENCHMARKING VISUAL OBJECT TRACKING

There exist a lot of datasets, evaluation metrics, and strategies for benchmarking Visual Object Tracking algorithms. This chapter examines how to measure the performance of these algorithms. In addition, this chapter describes which evaluation metrics we can use to compare the performances of these algorithms. Moreover, this chapter introduces which datasets we can use while evaluating these algorithms.

4.1. Evaluation Metrics

There exist three different evaluation strategies. These evaluation strategies are One Path Evaluation (OPE), Temporal Robustness Evaluation (TRE), and Spatial Robustness Evaluation (SRE) in Visual Object Tracking. We select OPE evaluation strategy to evaluate tracking algorithms. In OPE, we initialize these algorithms with only the first ground-truth for each dataset sequence. We do not manipulate the process of these algorithms until the sequence ends during tracking. [3]

This section describes standard evaluation metrics for Visual Object Tracking algorithms: Intersection over Union (IoU), Center Location Error, Success, and Precision.

4.1.1. Intersection Over Union (IoU)

Intersection over Union (IoU) can be measured as target area overlap ratio between the ground-truth and the predicted target, as shown in Equation 4.1 and demonstrated in Figure 4.1. The IOU compares only the similarity of target representations, not depending on the target scale. However, there exists a drawback in using IOU because it does not measure the distance from the ground-truth and the predicted target. For example, the values of the IOU of two shapes are the same and equal to zero in Figure 4.2, although these shapes do not intersect with each other.

$$\text{IoU} = \frac{|A \cap B|}{|A \cup B|}$$

Equation 4.1 Rectangular A and B for IoU calculation

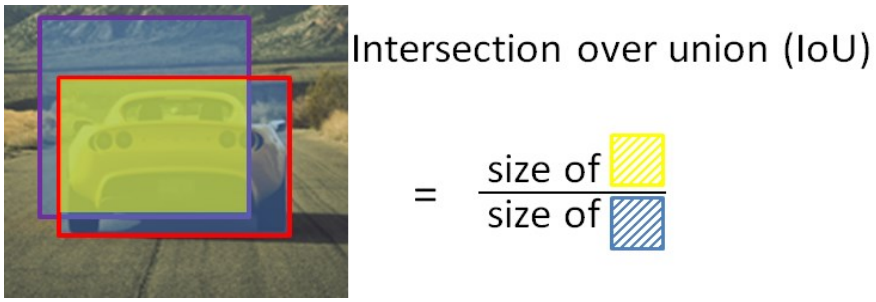


Figure 4.1 IoU of two rectangular areas

(a) (b)

Figure 4.2 The problem of IoU (a) the shapes are close to each other,
(b) the shapes are far from each other

4.1.2. Success

We can calculate Success as the ratio of the number of frames that the target's IoU value is higher than or equal to the overlap ratio thresholds. These thresholds are ranging from 0 to 1 increasingly. As usual, we select 0.5 as the overlap ratio threshold to rank object tracking algorithms. Later, we can plot the calculated values of Success with all overlap ratio thresholds in a graph as a Success plot. In Figure 4.3, we can see an example of the Success Plot of SiamFC tracker on OTB-100 dataset [3] using OPE strategy.

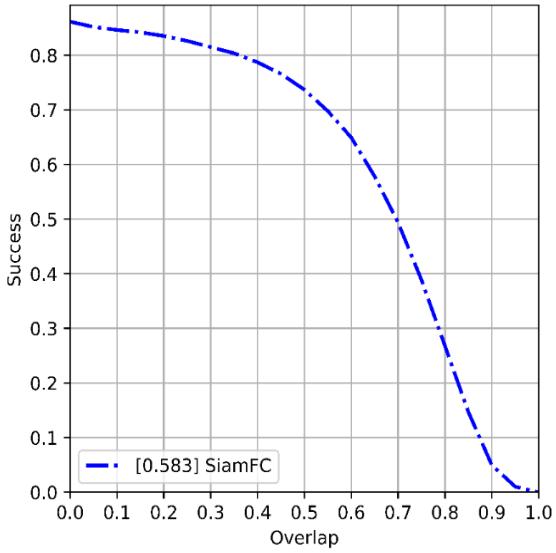


Figure 4.3 Success Plot of SiamFC tracker on the OTB-100 dataset using OPE strategy

4.1.3. Center Location Error

We can calculate Center Location Error as the average Euclidean distance between the center locations of the ground-truth and the predicted, as demonstrated in Equation 4.2. In the equation, there are p_1 and p_2 points and the coordinate values of these points on the x-axis and the y-axis. The value of Center Location Error increases if we tend to lose the tracking target. Thus, we grouped center position distances by different thresholds. Center Location Error is the base of the Precision evaluation metric.

$$d(p_1, p_2) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \tag{4.2}$$

4.1.4. Precision

We can calculate Precision as the ratio of the number of frames that the Center Location Error is lower than the threshold. These thresholds are ranging from 0 to 51 increasingly. As usual, we select 20 as the pixel distance threshold to rank object tracking algorithms. Thus, when the value of Precision is high, the distance, which is from the target estimated center and the ground-truth center, ranges from 20 pixels in most frames. Later, we can plot the calculated values of Precision with all center location error thresholds in a graph as a Precision plot. In Figure 4.4, we can see an example of the Precision Plot of the SiamFC tracker on the OTB-100 dataset [3] using OPE strategy.

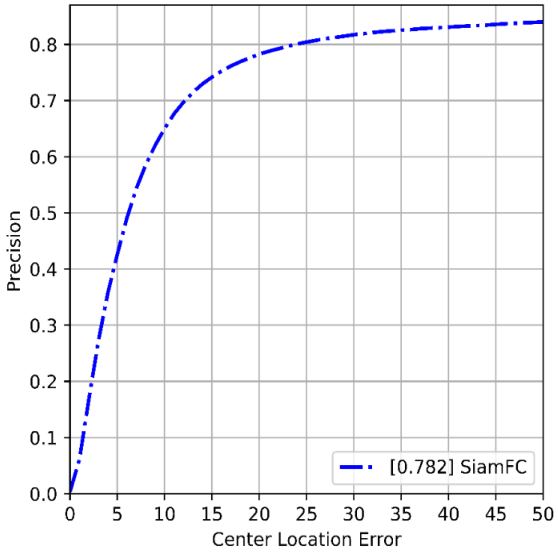


Figure 4.4 Precision Plot of SiamFC tracker on the OTB-100 dataset using OPE strategy

4.2. Datasets

We can evaluate object tracking algorithms on several datasets. The OTB-100 [3] and TC-128 [8] datasets are the most used ones, so we selected these datasets to evaluate our proposed tracking algorithm and analysis the result. Also, we prepared two new datasets whose sequences we selected from OTB-100 and TC-128 standard datasets especially. Later, we run our proposed algorithm on these datasets and analysis the results.

4.2.1. Object Tracking Benchmark (OTB)

Wu et al. [3] proposed the Object Tracking Benchmark (OTB) dataset in CVPR2013. The purpose of the dataset is to benchmark online object tracking algorithms. OTB dataset has a hundred sequences and OTB dataset has 4 different versions: OTB-2013, OTB-2015, OTB-50, and OTB-100. Researchers actively use OTB-50 and OTB-100. The target representation of the dataset is a box, and the ground-truths of each sequence have (x, y, box-width, box-height) format. Except for *Jogging* and *Skating2* sequences, the dataset has only one ground-truth box for each sequence. The sequences of the OTB dataset are tagged with eleven attributes, as described in Table 1.1. Also, Figure 4.5 shows the example images from different sequences in the OTB-100 dataset.

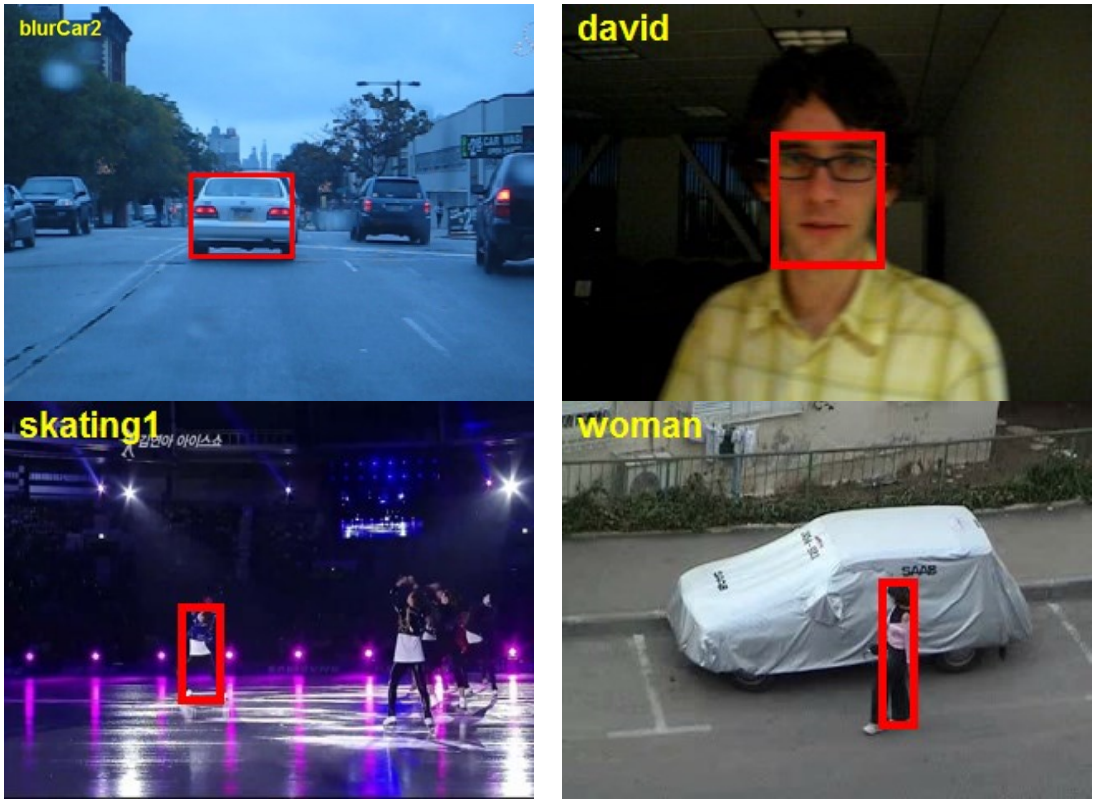


Figure 4.5 Example images from *BlurCar2*, *David*, *Skating1*, and *Woman* sequences in OTB-100 dataset (red: ground-truth)

4.2.2. Temple Color (TC-128)

Liang et al. [8] proposed the TC-128 dataset to evaluate object tracking algorithms, especially color-enhanced ones. TC-128 dataset consists of 128 sequences. Like the OTB dataset, the target representation of the TC-128 dataset is a box. The ground-truths of each sequence have (x, y, box-width, box-height) format, and the dataset has only one ground-truth box for each sequence except the *Jogging* sequence. Also, the TC-128 dataset tags its sequences with the same eleven attributes like the OTB dataset. These attributes are shown and described in Table 1.1. Also, Figure 4.6 shows the example images from different sequences in the TC-128 dataset.

Figure 4.6 Example images from *Airport_ce*, *Bolt*, *Coke*, and *Pool_ce1* sequences in TC-128 dataset (red: ground-truth)

4.2.3. Special Selected Dataset

We selected fifteen sequences from each of the OTB-100 [3] and TC-128 [8] datasets. Our sequence selection criterion was selecting sequences that are tagged with at least one of the following attributes. These attributes were BC (Background Clutters), FM (Fast Motion), MB (Motion Blur), OCC (Occlusion) attributes. We chose these attributes especially because these complex tracking scenarios are relatively open to improvement. Both Table 4.1 and Table 4.2 show these sequences. We named these selected sequences from the OTB-100 and the TC-128 datasets as IZTECH15-OTB and IZTECH15-TC datasets.

Table 4.1 Sequences in IZTECH15-OTB dataset

Sequence	**Attributes**
Human3	SV, OCC, DEF, OPR, BC
Surfer	SV, FM, IPR, OPR, LR
Boy	SV, MB, FM, IPR, OPR
Deer	MB, FM, IPR, BC, LR
Jumping	MB, FM
Tiger1	IV, OCC, DEF, MB, FM, IPR, OPR
Basketball	IV, OPR, OCC, DEF, BC
KiteSurf	IV, OCC, IPR, OPR
Coke	IV, OCC, FM, IPR, OPR, BC
Suv	OCC, IPR, OV
DragonBaby	SV, OCC, MB, FM, IPR, OPR, OV
Human4	IV, SV,OCC,DEF
Trellis	IV, SV, IPR, OPR, BC
CarDark	IV, BC
Human7	IV, SV, OCC, DEF, MB, FM

IV: Illumination Variation
SV: Scale Variation
OCC: Occlusion
DEF: Deformation
MB: Motion Blur
FM: Fast Motion
IPR: In-Plane Rotation
OPR: Out-of-Plane Rotation
OV: Out-of-View
BC: Background Clutters
LR: Low Resolution

Table 4.2 Sequences in IZTECH15-TC dataset

Sequence	**Attributes**
Railwaystation_ce	OCC, IPR, BC
Messi_ce	SV, OCC, DEF, MB, IPR, BC
Face_ce2	IV, OCC, MB, IPR, OPR, FM
Busstation_ce1	OCC, BC
Boy	OPR, SV, MB, FM, IPR
Bicycle	IV, SV, BC
Deer	MB, FM, IPR, BC
Tiger1	IV, OPR, OCC, DEF, MB, FM, IPR
Basketball	IV, OPR, OCC, DEF, BC
Busstation_ce2	OCC, IPR, OPR, BC
Hurdle_ce2	DEF, FM, BC
Michaeljackson_ce	IV, DEF, FM, IPR, OPR
Badminton_ce2	DEF, MB, OPR
Skyjumping_ce	IV, SV, DEF, FM, OPR
Trellis	IV, OPR, SV, IPR, BC

IV: Illumination Variation
SV: Scale Variation
OCC: Occlusion
DEF: Deformation
MB: Motion Blur
FM: Fast Motion
IPR: In-Plane Rotation
OPR: Out-of-Plane Rotation
OV: Out-of-View
BC: Background Clutters
LR: Low Resolution

CHAPTER 5

RESULTS AND ANALYSIS

This chapter shows the overall Precision and Success results of SiamFC [4], Kalman-Siam [5], and Adaptive-Kalman-Siam on OTB-100 [3], TC-128 [8], IZTECH15-OTB, and IZTECH15-TC datasets using OPE strategy. In addition to Precision and Success, we will compare the execution time performance of these trackers. However, we will explain how we obtain the results before the results and their analysis.

5.1. Implementation

We implemented SiamFC, Kalman-Siam, and Adaptive-Kalman-Siam tracking algorithms using the PyTorch Deep Learning library. We did not train any neural networks and only used the pre-trained weights of the SiamFC. We ran these tracking algorithms in the GPU-powered cloud using Google Collab to get the overall results on OTB-100, TC-128, IZTECH15-OTB, and IZTECH15-TC datasets

5.2. Overall Results

In our experiments, the results of SiamFC are higher than the other tracking algorithms because we do not have the actual implementation and pre-trained weights for the Kalman-Siam. Thus, we implemented Kalman-Siam according to its article, and we used pre-trained weights for SiamFC. Our proposed tracking method gets better results than our Kalman-Siam implementation. Both Figure 5.1, Figure 5.2 show the results of SiamFC, Kalman-Siam, and Adaptive-Kalman-Siam on OTB-100 and TC-128 datasets. Adaptive-Kalman-Siam is approximately 5 and 10 percent more successful than the Kalman-Siam, respectively.

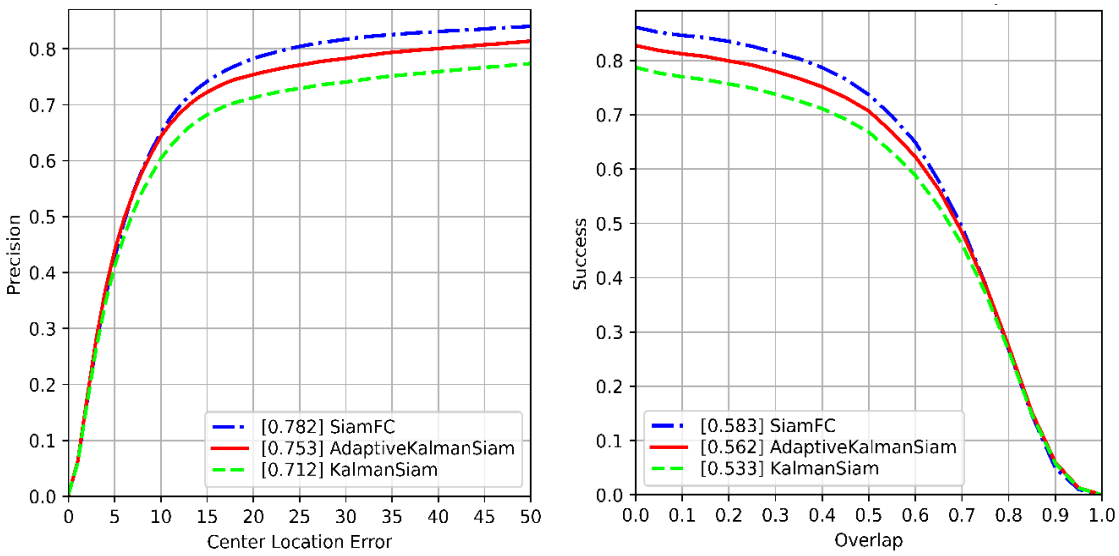


Figure 5.1 The precision and success plot of SiamFC, Kalman-Siam, and Adaptive Kalman-Siam on the OTB-100 dataset using OPE strategy

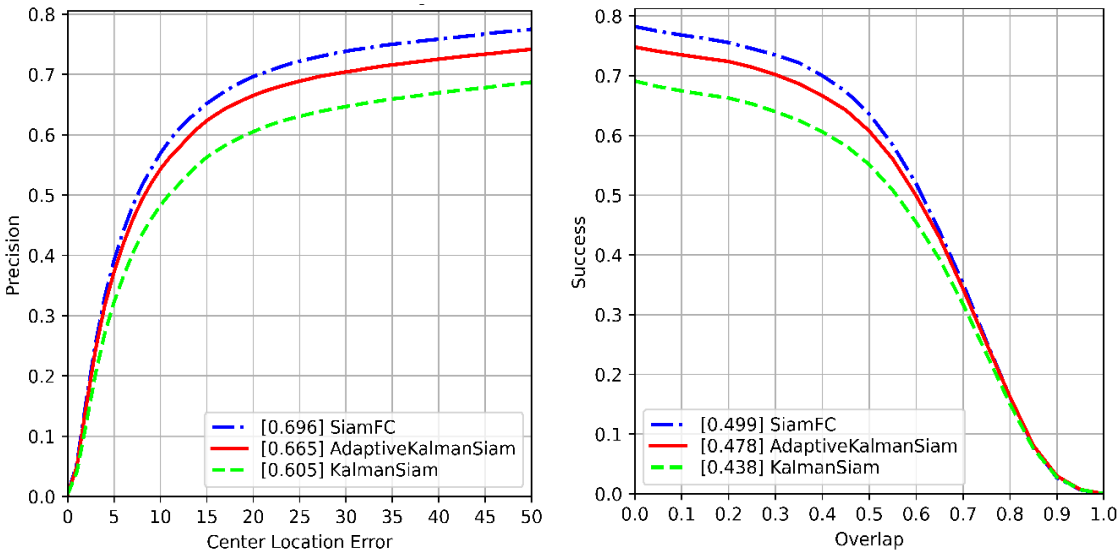


Figure 5.2 The precision and success plots of SiamFC, Kalman-Siam, and Adaptive Kalman-Siam on the TC-128 dataset using OPE strategy

5.3. Attribute-Specific Results

The Adaptive-Kalman-Siam achieves better results than the SiamFC and the Kalman-Siam in IZTECH15-OTB and IZTECH15-TC datasets with specially selected sequences by approximately 30 percent. Both Figure 5.3 and Table 5.1 show the overall results of three trackers on the IZTECH15-OTB dataset. Both Figure 5.4 and Table 5.2 show the overall results of three trackers on the IZTECH15- TC dataset.

Adaptive-Kalman-Siam achieves even better results, especially in sequences tagged with BC (Background Clutters), FM (Fast Motion), MB (Motion Blur), and OCC (Occlusion) attributes. Both Table 5.3 and Table 5.4 show the results in IZTECH15-OTB and IZTECH15-TC datasets consisting of sequences labeled with these attributes.

Thanks to Kalman-Siam's improvements on SiamFC, Kalman-Siam performs better than SiamFC in sequences tagged with BC (Background Clutters) and OCC (Occlusion) attributes tagged sequences. However, the proposed Adaptive-Kalman-Siam is more successful in these attributes than Kalman-Siam. Moreover, the Adaptive Window Search algorithm solves the occlusion problem better than Kalman-Siam. However, Kalman-Siam already has an occlusion detection mechanism. Also, Adaptive Kalman-Siam's *APCE* parameter is better than Kalman-Siam's *P* parameter for locating the target in a response map in sequences tagged with BC (Background Clutters) and tracking small-sized targets.

Kalman-Siam's contribution to SiamFC has a low effect on challenging sequences tagged with FM (Fast Motion) and MB (Motion Blur) attributes. Also, Kalman-Siam achieves a little less success than SiamFC in these sequences. However, the proposed Adaptive-Kalman-Siam holds the least SiamFC success rate in these sequences and is even more successful. When the target's acceleration is not constant or the target's movement direction changes suddenly, Kalman-Siam's Kalman Filter estimation can have a negative effect. Also, when the search window is blurred because of the target's motion, the *P* parameter is not high enough to locate the target in the response map. The proposed Adaptive-Kalman-Siam's Adaptive Window Search solves these problems.

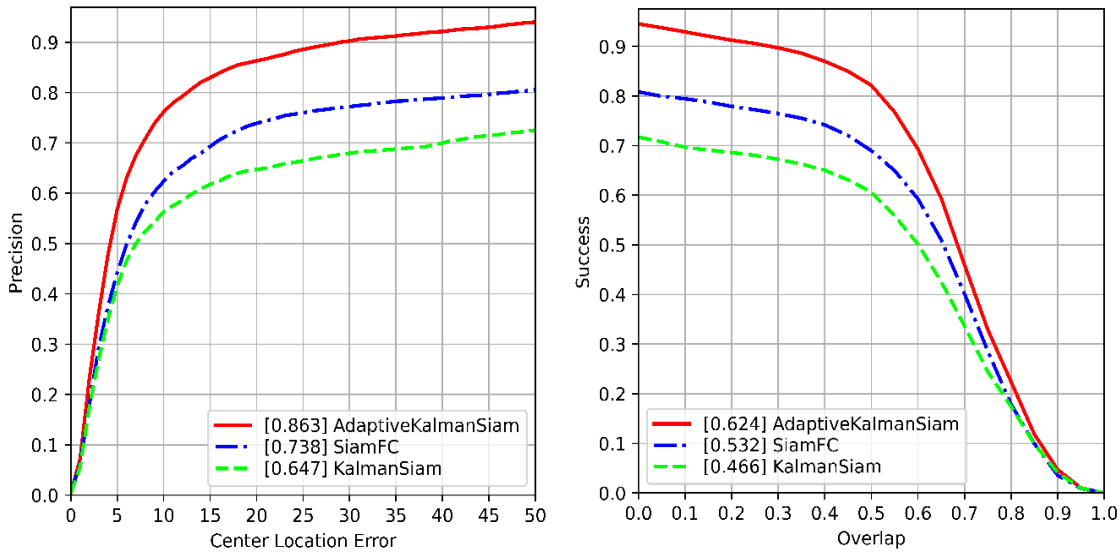


Figure 5.3 The precision and success plots of SiamFC, Kalman-Siam, and Adaptive Kalman-Siam on the IZTECH15-OTB dataset using OPE strategy

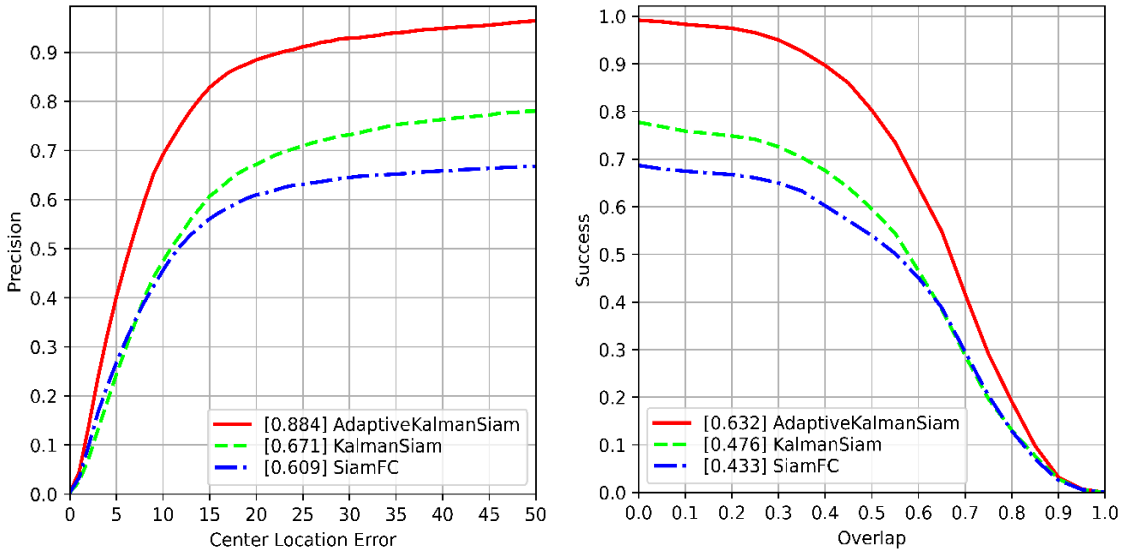


Figure 5.4 The precision and success plots of SiamFC, Kalman-Siam, and Adaptive Kalman-Siam on the IZTECH15-TC dataset using OPE strategy

Table 5.1 The evaluation results of SiamFC, Kalman-Siam,
and Adaptive Kalman-Siam on the IZTECH15-OTB dataset using OPE strategy

Tracker	**Precision (t=21)**	**Success (t<=0.5)**	**FPS**
SiamFC	0.647	0.532	**85.020**
Kalman-Siam	0.738	0.466	35.310
Adaptive-Kalman-Siam	**0.863**	**0.624**	36.010

Table 5.2 The evaluation results of SiamFC, Kalman-Siam,
and Adaptive Kalman-Siam on the IZTECH15-TC dataset using OPE strategy

Tracker	**Precision (t=21)**	**Success (t<=0.5)**	**FPS**
SiamFC	0.609	0.433	**82.350**
Kalman-Siam	0.671	0.476	37.300
Adaptive-Kalman-Siam	**0.884**	**0.632**	34.590

Table 5.3 The precision and success plots of SiamFC, Kalman-Siam, and Adaptive Kalman-Siam on the IZTECH15-OTB dataset using OPE strategy for specific attributes

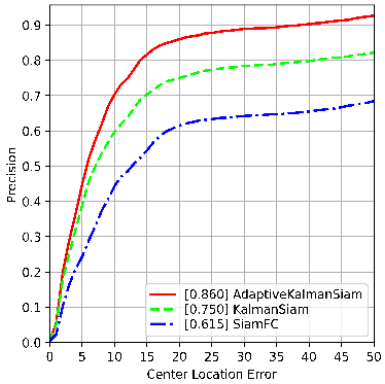
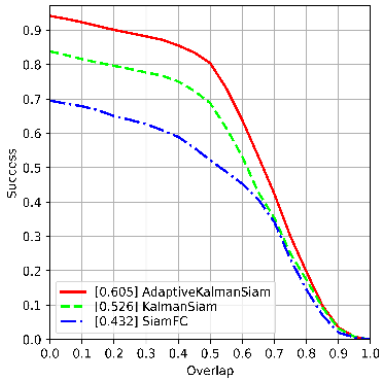
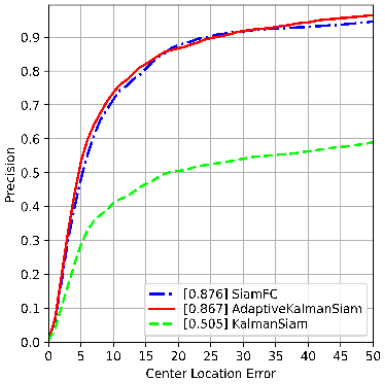
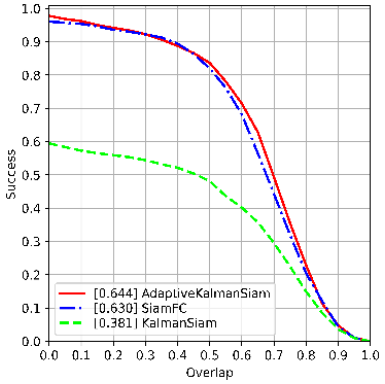
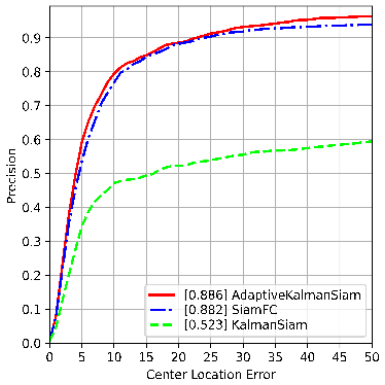
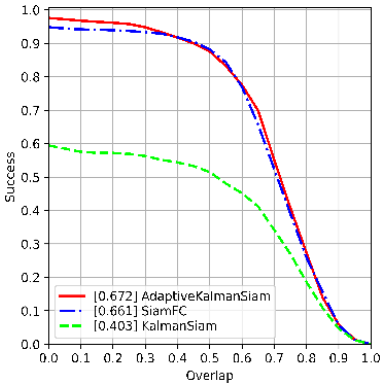
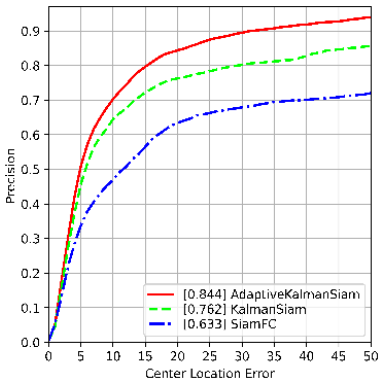
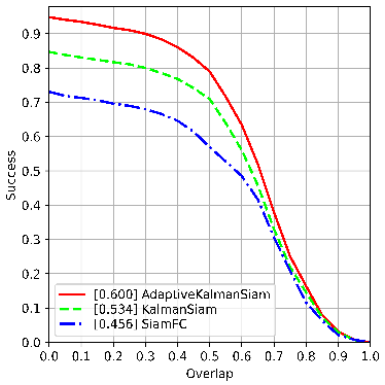
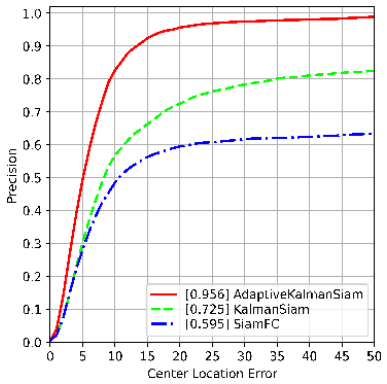
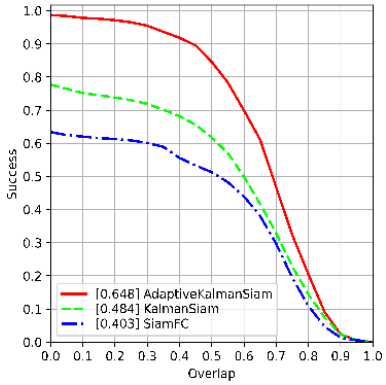
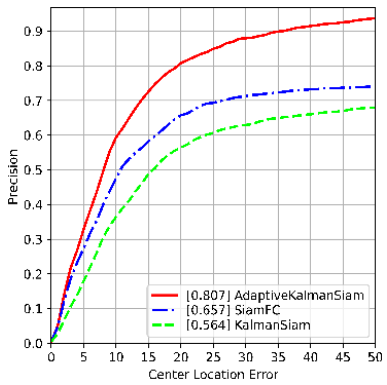
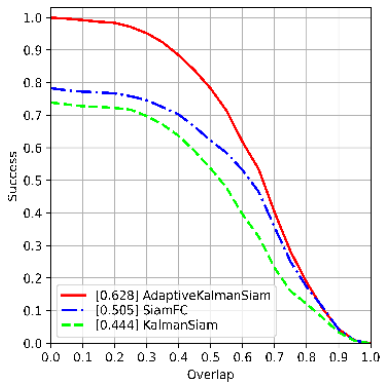
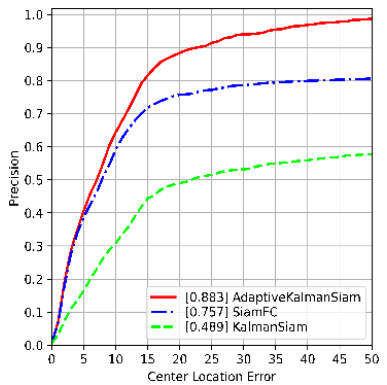
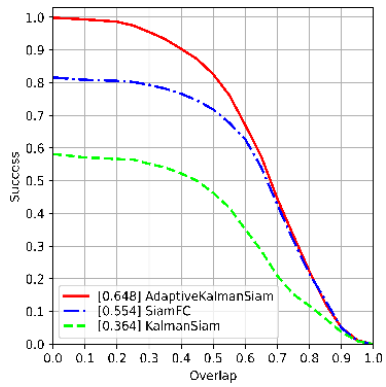
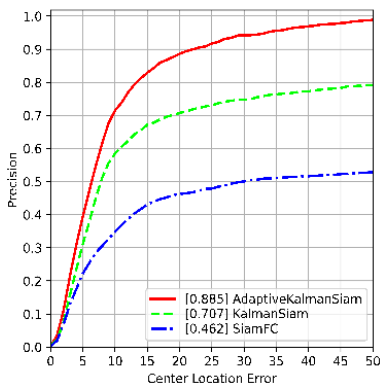
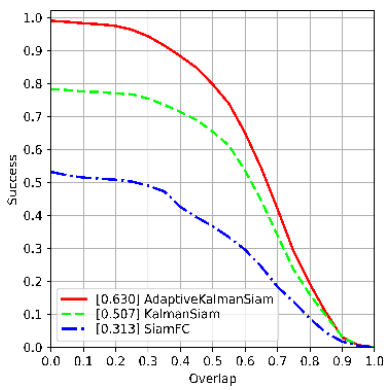
Name	Precision	Success
Background Clutters (BC)		
Fast Motion (FM)		
Motion Blur (MB)		
Occlusion (OCC)		

Table 5.4 The precision and success plots of SiamFC, Kalman-Siam, and Adaptive Kalman-Siam on the IZTECH15-TC dataset using OPE strategy for specific attributes

Name	Precision	Success
Background Clutters (BC)		
Fast Motion (FM)		
Motion Blur (MB)		
Occlusion (OCC)		

5.4. Performance Comparison

We calculated our performance metrics as Frame per Second (FPS). According to our results, Adaptive-Kalman-Siam works at approximately the same speed as Kalman-Siam. On the other hand, SiamFC runs at approximately twice the speed of both Kalman-Siam and Adaptive-Kalman-Siam. However, Kalman-Siam and Adaptive-Kalman-Siam work at about 35 FPS, and it means real-time.

5.5 Ablation Study

We did an ablation study with three different versions of the proposed tracker during the development of the proposed Adaptive-Kalman-Siam tracker. These are Adaptive-Kalman-Siam with P parameter, Adaptive-Kalman-Siam with only distinct search windows, and Adaptive-Kalman-Siam with a single threshold which means T_{Kalman} is now used in the algorithm. Both Figure 5.5 and Figure 5.6 show the results on the IZTECH15-OTB and IZTECH15-TC datasets. In addition, both Table 5.5 and Table 5.6 show the comparison with SiamFC and Kalman-Siam. Each of the improvements of the proposed Adaptive Kalman-Siam affects the tracker's accuracy positively.

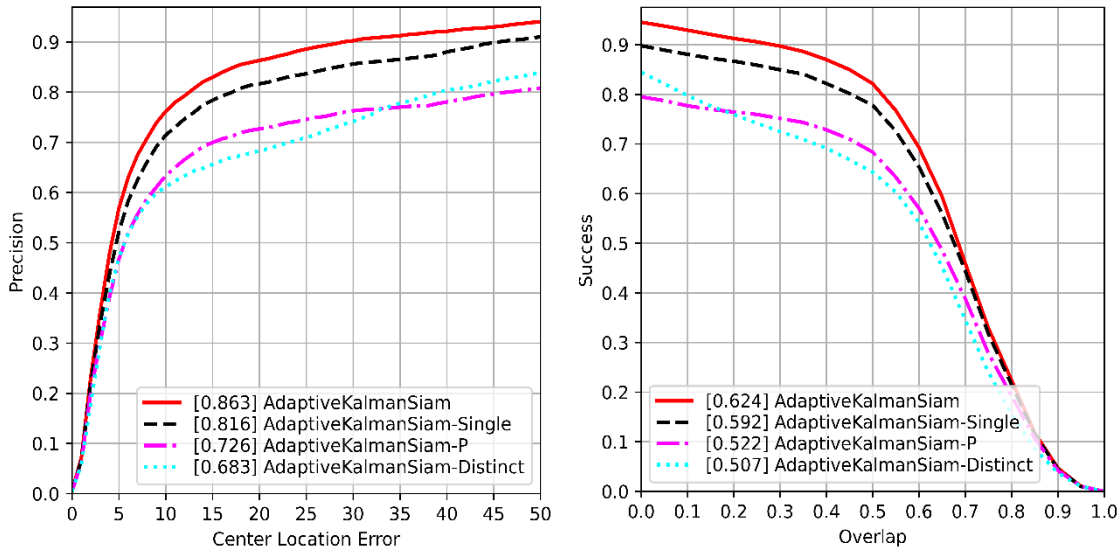


Figure 5.5 The precision and success plots of the ablation study on the IZTECH15-OTB dataset using OPE strategy

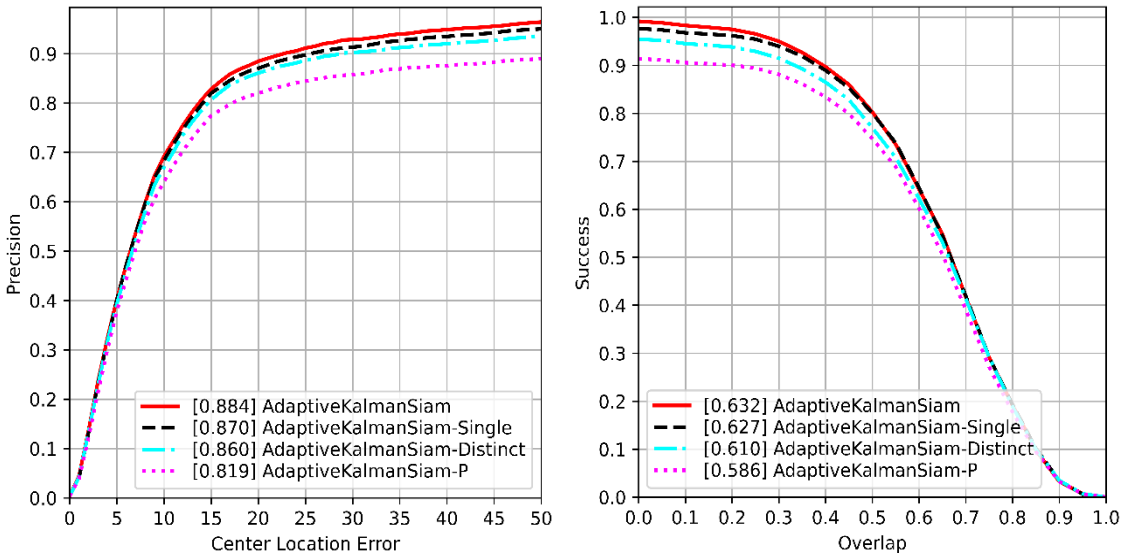


Figure 5.6 The precision and success plots of the ablation study on the IZTECH15-TC dataset using OPE strategy

Table 5.5 The ablation study results on the IZTECH15-OTB dataset using OPE strategy

Tracker	**Precision (t=21)**	**Success (t<=0.5)**
SiamFC	0.647	0.532
Kalman-Siam	0.738	0.466
Adaptive-Kalman-Siam	**0.863**	**0.624**
Adaptive-Kalman-Siam-P	0.726	0.522
Adaptive-Kalman-Siam-Distinct	0.683	0.507
Adaptive-Kalman-Siam-Single	0.816	0.592

Table 5.6 The ablation study results on the IZTECH15-TC dataset using OPE strategy

Tracker	**Precision (t=21)**	**Success (t<=0.5)**
SiamFC	0.609	0.433
Kalman-Siam	0.671	0.476
Adaptive-Kalman-Siam	**0.884**	**0.632**
Adaptive-Kalman-Siam-P	0.819	0.586
Adaptive-Kalman-Siam-Distinct	0.860	0.610
Adaptive-Kalman-Siam-Single	0.870	0.627

CHAPTER 6

CONCLUSION

Although SiamFC is a successful Visual Object Tracker based on Siamese Neural Networks, it is not successful enough in complex tracking scenarios such as Occlusion. The main reason is that SiamFC does not have an online learning process while tracking. Therefore, for complex tracking scenarios, the tracker should detect the target's situations change during tracking. However, thanks to Kalman-Siam, the previous object trajectory is used with the Kalman Filter. Also, Kalman-Siam defines the occlusion state with the *P* parameter calculation made on the response map. In addition, Kalman-Siam uses a combined neural network output from different feature levels. These improvements make Kalman-Siam have better accuracy than SiamFC in complex tracking scenarios, especially Occlusion. Also, we developed an adaptive search window algorithm on Kalman-Siam for complex tracking scenarios, and it took one step ahead.

The main improvement of our proposed tracker is obtaining search windows according to the target's last location when re-tracking is required. Then, our proposed tracker uses these search windows according to the distance to the target's last location. Furthermore, the proposed tracker uses a threshold for the re-tracking process. We selected the *APCE* calculation as the parameter rather than the *P* calculation from Kalman-Siam for the threshold. Then, we got more efficient results by using the *APCE* parameter instead of the *P* parameter. Finally, we proved the success score of the improvements of our proposed tracking method by conducting experiments. As a result, we achieved the same running time performance with Kalman-Siam. Furthermore, we achieved more successful results than Kalman-Siam for complex tracking scenarios such as Occlusion, Fast Motion, Background Clutters, and Motion Blur.

REFERENCES

[1] M. Fiaz, A. Mahmood, S. Javed, and S. K. Jung, "Handcrafted and deep trackers: Recent visual object tracking approaches and trends," *ACM Computing Surveys*, vol. 52, no. 2. 2019, DOI: 10.1145/3309665.

[2] A. Yilmaz, O. Javed, and M. Shah, "Object tracking: A survey," *ACM Computing Surveys*, vol. 38, no. 4. 2006, DOI: 10.1145/1177352.1177355.

[3] Y. Wu, J. Lim, and M. H. Yang, "Online object tracking: A benchmark," 2013, DOI: 10.1109/CVPR.2013.312.

[4] L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, and P. H. S. Torr, "Fully-convolutional siamese networks for object tracking," 2016, vol. 9914 LNCS, DOI: 10.1007/978-3-319-48881-3_56.

[5] L. Zhou and J. Zhang, "Combined Kalman Filter and Multifeature Fusion Siamese Network for Real-Time Visual Tracking," *Sensors (Basel).*, vol. 19, no. 9, p. 2201, May 2019, DOI: 10.3390/s19092201.

[6] R. E. Kalman, "A new approach to linear filtering and prediction problems," *J. Fluids Eng. Trans. ASME*, vol. 82, no. 1, 1960, DOI: 10.1115/1.3662552.

[7] L. Zhou, H. Li, and J. Zhang, "Multi-feature fusion Siamese Network for Real-Time Object Tracking," 2018, DOI: 10.1145/3297156.3297259.

[8] P. Liang, E. Blasch, and H. Ling, "Encoding Color Information for Visual Tracking: Algorithms and Benchmark," *IEEE Trans. Image Process.*, vol. 24, no. 12, 2015, DOI: 10.1109/TIP.2015.2482905.

[9] G. Welch and G. Bishop, "An Introduction to the Kalman Filter," *In Pract.*, vol. 7, no. 1, 2006, DOI: 10.1.1.117.6808.

[10] S. Liu, D. Liu, G. Srivastava, D. Połap, and M. Woźniak, "Overview and methods of correlation filter algorithms in object tracking," *Complex Intell. Syst.*, 2020, DOI: 10.1007/s40747-020-00161-4.

[11] D. S. Bolme, J. R. Beveridge, B. A. Draper, and Y. M. Lui, "Visual object tracking using adaptive correlation filters," 2010, DOI: 10.1109/CVPR.2010.5539960.

[12] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista, "Exploiting the circulant structure of tracking-by-detection with kernels," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2012, vol. 7575 LNCS, no. PART 4, DOI: 10.1007/978-3-642-33765-9_50.

[13] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista, "High-speed tracking with kernelized correlation filters," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 3, 2015, DOI: 10.1109/TPAMI.2014.2345390.

[14] D. Held, S. Thrun, and S. Savarese, "Learning to track at 100 FPS with deep regression networks," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2016, vol. 9905 LNCS, DOI: 10.1007/978-3-319-46448-0_45.

[15] G. Ning *et al.*, "Spatially supervised recurrent convolutional neural networks for visual object tracking," 2017, DOI: 10.1109/ISCAS.2017.8050867.

[16] N. Wojke, A. Bewley, and D. Paulus, "Simple online and realtime tracking with a deep association metric," in *Proceedings - International Conference on Image Processing, ICIP*, 2018, vol. 2017-September, DOI: 10.1109/ICIP.2017.8296962.

[17] Y. Jia *et al.*, "Caffe: Convolutional architecture for fast feature embedding," 2014, DOI: 10.1145/2647868.2654889.

[18] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016, vol. 2016-December, DOI: 10.1109/CVPR.2016.91.

[19] S. Hochreiter and J. Schmidhuber, “Long Short-Term Memory,” *Neural Comput.*, vol. 9, no. 8, 1997, doi: 10.1162/neco.1997.9.8.1735.

[20] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft, "Simple online and realtime tracking," in *2016 IEEE International Conference on Image Processing (ICIP)*, 2016, pp. 3464–3468, DOI: 10.1109/ICIP.2016.7533003.

[21] R. Girshick, "Fast R-CNN," in *2015 IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 1440–1448, DOI: 10.1109/ICCV.2015.169.

[22] H. W. Kuhn, "The Hungarian method for the assignment problem," *Nav. Res. Logist. Q.*, vol. 2, no. 1–2, 1955, DOI: 10.1002/nav.3800020109.

[23] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Commun. ACM*, vol. 60, no. 6, 2017, DOI: 10.1145/3065386.

[24] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2015.

[25] R. Tao, E. Gavves, and A. W. M. Smeulders, "Siamese instance search for tracking," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016, vol. 2016-December, DOI: 10.1109/CVPR.2016.158.

[26] R. Pflugfelder, "An In-Depth Analysis of Visual Tracking with Siamese Neural Networks," *arXiv*. 2017.

[27] M. Wang, Y. Liu, and Z. Huang, "Large margin object tracking with circulant feature maps," in *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, 2017, vol. 2017-January, DOI: 10.1109/CVPR.2017.510.

[28] J. Shin, H. Kim, D. Kim, and J. Paik, "Fast and robust object tracking using tracking failure detection in kernelized correlation filter," *Appl. Sci.*, vol. 10, no. 2, 2020, DOI: 10.3390/app10020713.