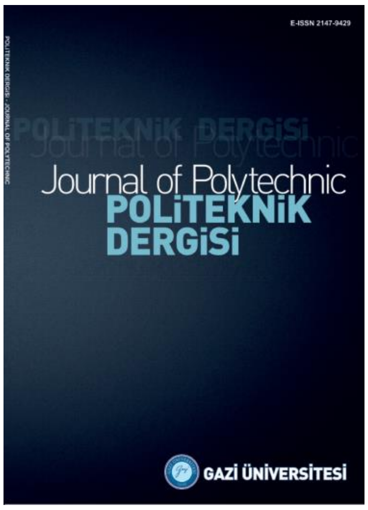


POLİTEKNİK DERGİSİ

JOURNAL of POLYTECHNIC

ISSN: 1302-0900 (PRINT), ISSN: 2147-9429 (ONLINE)

URL: http://dergipark.org.tr/politeknik

Real-time scalable system for face tracking in multi-camera

Çoklu kameralarda gerçek zamanlı ölçeklenebilir yüz tanıma sistemi

***Yazar(lar) (Author(s))**: Mehmet F. OZDEMIR[1], Davut HANBAY[2]*

ORCID[1] 0000-0003-3563-054X

ORCID[2]: 0000-0003-2271-7865

To cite to this article: Özdemir M. F. and Hanbay D., "Real-time scalable system for face tracking in multi-camera", ***Journal of Polytechnic***, *(*): *, (*).

Bu makaleye şu şekilde atıfta bulunabilirsiniz: Özdemir M. F. ve Hanbay D., "Real-time scalable system for face tracking in multi-camera", ***Politeknik Dergisi***, *(*): *, (*).

Erişim linki (To link to this article): http://dergipark.org.tr/politeknik/archive

DOI: 10.2339/politeknik.1332952

Real-Time Scalable System For Face Tracking In Multi-Camera

Highlights

- *The proposed system is a real-time face tracking system in a multi-camera environment.*
- *DeepSORT-based new design is recommended to make face tracking more stable and faster.*
- *A cost-oriented, effective fault-tolerant and scalable system is proposed.*

Graphical Abstract

In this study, a real-time, multi-camera, deep learning-based face tracking system was developed.

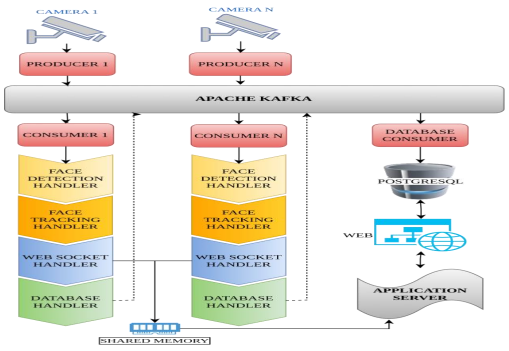


Figure. System architecture

Aim

The aim of the proposed system is to realize a real-time facial recognition system using images from multiple cameras.

Design & Methodology

The system was designed with image processing algorithms and deep learning models.

Originality

A real-time facial recognition system has been designed. In addition, the deep learning model has been replaced with the face recognition model ArcFace in DeepSORT.

Findings

In the proposed system, when an image is input into the system, it can be displayed on the web page after approximately 127 ms.

Conclusion

The more stable face tracking was achieved and, the proposed system was shown to be stable, efficient, and cost-effective.

Declaration of Ethical Standards

The author(s) of this article declare that the materials and methods used in this study do not require ethical committee permission and/or legal-special permission.

Real-Time Scalable System For Face Tracking In Multi-Camera

Araştırma Makalesi / Research Article

Mehmet F. OZDEMIR[1*], Davut HANBAY[1]

[1]Dept. of Computer Engineering, Inonu University, Malatya, 44280, Türkiye

(Geliş/Received : 26.07.2023 ; Kabul/Accepted : 01.03.2024 ; Erken Görünüm/Early View : 14.03.2024)

ABSTRACT

Face detection and tracking have become increasingly popular in recent years. It has critical importance in security, defense, and robotics applications uses encountered in everyday life. For this purpose, many decision support or expert systems have been developed using artificial intelligence and machine learning. Thanks to the developments in the field of deep learning and hardware many effective and reliable face tracking systems have been realized. However there are still very few real-time scalable end-to-end systems. Also, the realization of this system on multiple cameras is a real challenge. In this study, a real-time, multi-camera, deep learning-based face tracking system has been developed. In the realized system, SCRFD model is used for face detection, ArcFace model is used for face recognition, and an updated DeepSORT algorithm is used for more stable face tracking. In addition, Apache Kafka stream processing system and Socket.IO bidirectional communication library were used to process multi-camera data in real-time and scalable. In the proposed system, when an image is input into the system, it can be displayed on the web page after approximately 127 ms.

Keywords: Face recognition, face tracking, deep learning, multi-camera.

Çoklu Kameralarda Gerçek Zamanlı Ölçeklenebilir Yüz Tanıma Sistemi

ÖZ

Yüz tespiti ve takibi son yıllarda giderek daha popüler bir başlık hâline gelimiştir. Günlük yaşamda karşılaşılan güvenlik, savunma ve robotik uygulamaları kullanımlarında kritik öneme sahiptir. Bu amaçla yapay zeka ve makine öğrenmesi kullanılarak birçok karar destek ve uzman sistem geliştirilmiştir. Derin öğrenme ve donanım alanında yaşanan gelişmeler sayesinde birçok etkili ve güvenilir yüz takip sistemi hayata geçirilmiştir. Ancak hala çok az sayıda gerçek zamanlı ölçeklenebilir uçtan uca sistem bulunmaktadır. Ayrıca bu sistemin birden fazla kamerada gerçekleştirilmesi gerçek bir zorluktur. Bu çalışmada gerçek zamanlı, çoklu kameralı, derin öğrenme tabanlı bir yüz takip sistemi geliştirilmiştir. Gerçekleştirilen sistemde yüz tespiti için SCRFD modeli, yüz tanıma için ArcFace modeli, daha stabil yüz takibi için güncellenmiş DeepSORT algoritması kullanılmıştır. Ayrıca çoklu kamera verilerinin gerçek zamanlı ve ölçeklenebilir şekilde işlenmesi için Apache Kafka akış işleme sistemi ve Socket.IO çift yönlü iletişim kütüphanesi kullanılmıştır. Önerilen yaklaşımda sisteme bir görüntü girdi olarak verildiğinde yaklaşık 127 ms sonra web sayfasında görüntülenebilmektedir.

Anahtar Kelimeler: Yüz tanıma, yüz takibi, derin öğrenme, çoklu-kamera.

1. INTRODUCTION

Computers and other electronic devices that we use in our daily lives allow us to develop human-machine interaction systems. One of the most valuable areas for the development of these systems is computer vision. There are many studies on computer vision in the literatüre [1-7]. Face detection and face recognition are only two of the studied subfields of computer vision. There are many real-time face recognition approaches in the world today. While most of them provide single-camera solutions, some have extended their success to multi-camera. Moreover, it is not an easy process to implement real-time face recognition in multi-camera systems. Because many difficulties are encountered when face recognition is desired in multi-camera systems.

The first is that the image streams from each camera must be processed independently and simultaneously. At the same time, the system should be designed to be distributed and scalable. In other words, as the number of cameras increases then the system load should be distributed accross more than one server. Apache Kafka [8] provides a real-time platform, best known as open-source distributed event streaming. It is widely used for real-time processing, high-performance data pipelines, stream processing, data integration, and mission-critical applications. It also provides a scalable platform and

*Sorumlu Yazar (Corresponding Author)
e-posta : mfatih.ozdemir@inonu.edu.tr

forms backbone for independent and simultaneous processing of each camera.

The second is to prevent data loss by building a fault-tolerant system. Fault tolerance is essential when even a single piece of data matters. Apache Kafka is a fault-tolerant platform in terms of being a distributed system. Topics can be divided into Partitions and copies can be kept in different brokers. Thus, if one of the brokers breaks down or becomes inoperable, it can be continued by other brokers without data loss.

The other is that it is difficult to combine face detection and face recognition in a stable, efficient, and low-cost system. First of all, face detection and recognition algorithms should be handled separately. In face detection, there are deep learning models [1,9-11] as well as many traditional methods [12-14]. It can be said that deep learning models give more successful results than traditional models. Among the popular deep learning models for face detection, MTCNN [9], Tinaface [10], and SCRFD [11] are given below.

MTCNN [9] has proposed a deep cascading multitasking framework that takes advantage of the natural correlation between detection and alignment to improve their performance. It uses a cascading architecture with three stages of deep convolutional networks, which are designed for face and landmark position prediction. In the MTCNN model, a three-stage stepped image pyramid is created. In the first stage, it obtains regression vectors of the candidate facial windows and their bounding boxes with the proposal network (P-Net). After the candidates are calibrated, non-maximum suppression (NMS) is used to combine highly overlapping candidates. In the second stage, another CNN called the Refined Network (R-Net) is used to eliminate the false candidates from all the candidates from the first stage. In the last stage, the output of five facial landmark positions is obtained using the output network (O-Net).

TinaFace [10] is one of the successful models that has taken its place in literature. It uses ResNet50 [15] as a backbone. It is also stated that all modules and techniques in TinaFace are easily applicable based on general object detection. TinaFace achieved 92.4% AP in WIDER FACE's [16] hard dataset. Tinaface model adopts several approaches. First, it has been stated that face detection is actually a one-class generic object detection. Therefore, it deals with face detection with techniques in generic object detection. Secondly, TinaFace provides a powerful, simple basic method based on generic object detection. It achieves an average accuracy (AP) of 92.1% on hard settings on the test subset of WIDER FACE with single scale and single model. The final version of the model achieves 92.4% AP on hard settings on the test subset with test time augmentation (TTA).

In addition to success, models that can be applied in real-time are newly added to the literature. SCRFD [11] is one of the models that is successful and also suitable for real-time. By making some improvements to TinaFace, they presented a more efficient and successful model. First, they state that face detection is more efficient under VGA resolution. In addition, they obtained more training examples for shallow stages. Secondly, they designed a simplified search field between the different components of a face detector. As a result, their model is more than 3 times faster and 3.86% more successful than TinaFace.

In face recognition, although there are traditional methods such as Eigenface [17], deep learning approaches are more popular. Face recognition is actually nothing but trying to correctly determine the identity of a person. Therefore, face recognition process needs to correctly extract the real features of face. Deep learning models such as SphereFace [18], CosFace [19], ArcFace [20], ElasticFace-Arc [21] can be shown among successful models in literature for face recognition. These models deal with the open set deep face recognition problem, where ideal face features are expected to have a maximum within-class distance smaller than minimum inter-class distance in a given metric space. In the open set approach, the identities in train and test dataset are expected to be completely different from each other. In this approach, after the features of face are extracted in the face recognition system, comparison is made using the nearest neighbor metric. On the other hand, in the closed set approach to face recognition, the identities of the training data set must be included in the test data set. In the next step, prediction is performed on the closed model. However, obtaining learning characteristics in the open set system is often difficult due to the large intra-class variation and high inter-class similarity. Therefore, these models have tried to optimize Softmax function, which allows learning angular discriminative features. It was observed that ArcFace was similar to ElasticFace-Arc and more successful than SphereFace and CosFace. The accuracy of these models on various datasets are as follows:

- SphereFace, face recognition model trained using A-Softmax, achieved face recognition accuracy of 99.42% in LFW [22] 95% in YTF [23] and 75.76% in MegaFace Rank-1 [24].
- CosFace achieved face recognition accuracy of 99.73% in LFW, 97.6% in YTF, 84.26% in MegaFace Challenge 1 Rank-1 and 77.06% in MegaFace Challenge 2 Rank-1.
- ElasticFace-Arc, trained on the MS1MV2 dataset, reached an accuracy of 99.82% in LFW and 98.81% in MegaFace Challenge 2 Rank-1.
- ArcFace, trained on the MS1MV2 dataset, which is the semi-automatic refined version of the MS-Celeb-1M dataset, reached an accuracy of 99.83% in LFW dataset and 98.02% in YTF dataset.

Another challenge is the near real-time viewing of processed images by the end user, which is essential where security is critical. Therefore, a communication line between end user and system should be established and data transmission should be made in real time. There are many communication approaches in this field.

However, WebSocket API is one of the most suitable approach for real-time bi-directional data transfer. This communication protocol establishes a bidirectional connection over TCP and is well suited for real-time applications. On the other hand, Socket.IO library provides some additional features to WebSocket API. A more efficient and stable communication can be established with these improvements. In some systems, the current frame from the camera stream is considered irrelevant to the previous frame. This prevents information from previous frames from being used. Therefore, the previously detected object must be associated with the current frame. There are many models for this in the literature. Among these algorithms, SORT (Bewley et al. 2016), DeepSORT (Wojke, Bewley, and Paulus 2018), Bytetrack (Y. Zhang et al. 2022) and OCSort (Cao et al. 2023) algorithms are only interested in tracking the object, regardless of object detection. Since DeepSORT is a more successful model and has re-identification, it is used in the proposed system, which avoids repeating the face recognition process in each frame, and realizes a more stable and lower cost system.

2. RELATED WORKS

There are many studies in the field of multi-camera face recognition in the literature. Each of these studies tries to realize multi-camera face recognition with different perspectives. However, the common goal of all of them is to build a multi-camera face recognition system.

Jason et al. proposed a model for common face recognition from video sequences in a multi-camera environment [29]. In this study, they benefited from inter-camera collaboration. This collaboration resulted in high recognition performance in common and non-common fields of view. In the non-common field-of-view approach, it is stated that an object predicts the appearance using last viewed location with inter-camera collaboration. Performance data of the proposed model were obtained in the experiment using four cameras. This study aimed to predict the direction of a target leaving the field of view of a camera, as well as time-of- arrival model between the appearances of targets in the cameras.

Z. Lian et al. proposed a multiple fusion-based real-time face tracking system [30]. In this system, after the face was detected by MTCNN, they presented a feature fusion-based method for face tracking. They used shape, motion and appearance features to measure object similarity. A convolutional neural network was used to extract appearance features. Motion and shape features were extracted using the Kalman filter [31]. It has been stated that a more stable tracking process was achieved thanks to its features combined with adjustable weights.

H. Badave et al. proposed a new approach to face recognition using multi-camera head pose estimation [32]. The purpose of the approach was to identify the person more efficiently by estimating the head pose. The study was about choosing the most ideal camera according to direction of the human head using a multi-camera system. The system uses facial landmark estimation-based face recognizer. It was tested on more than one person.

In this study, a face detecting and tracking system using deep learning with multi-camera was proposed. The study was tested on multi-camera images of 4 people. In addition, the system components were executed on the same computer. As a result, after a camera image was detected and applied to the proposed system, the processed image in the web interface was obtained in approximately 127 ms.

The organization of the paper is as follow materail and methods used are briefly explained in third section. application steps are explained in fourth section. In fifth section, conclusions are shared.

3. MATERIAL AND METHOD

In this section, we introduced briefly the basics of used methods.

3.1. SCRFD

Although great advances have been made in face detection, an efficient face detection with low computational cost and high precision has not been fully achieved. SCRFD [11] is a successful model for solving these problems. In TinaFace-based SCRFD model, multi-scale features are obtained by passing images through the FPN network in the feature extractor. Then, in the neck part, the multi-scale features on the backbone are combined. Finally, in the head section, face boxes and scores are predicted. The model combines two simple but effective approaches. These are,

1. Sample Redistribution (SR) approach is used, which increases the number of images in the training dataset.
2. Based on a carefully defined search methodology, Computation Redistribution (CR) approach is used, which reallocates computation between different components of the model (backbone, neck, and head).

Efficiency-Accuracy balance has been given great importance for detailed experiments performed on the WIDER FACE [16] dataset and SCRFD family proposed in a wide variety of computational forms. The proposed submodel SCRFD-34GF [11] outperforms the TinaFace [10] model by 3.86%. It also offers more than 3 times faster performance on VGA resolution images. Although TinaFace achieves impressive results in face detection, it has a high computational cost. In the SCRFD model, efficient face detection is performed under a fixed VGA resolution (640×480) instead of using a large resolution to reduce computational cost. Also, most of the faces in WIDER FACE are smaller than 32×32 pixels, so the prediction takes place in shallow stages. It was thought that it would be useful to obtain more training samples to further improve the estimation. Sample redistribution method with a large image cropping strategy is used to increase the number of images in the training dataset.

In Table 1, Accuracy and efficiency of SCRFD-34GF in the WIDER FACE validation set is compared with other approaches. As seen in the comparison, with the updates, the accuracy of the SCRFD-34 has increased and the inference time has reduced.

Table 1. Accuracy and efficiency of different methods on the WIDER FACE validation set [11]

Method	Easy	Medium	Hard	Infer(ms)
DSFD [4]	94.29	91.47	71.39	55.6
RetinaFace [1]	94.92	91.90	64.17	21.7
HAMBox [3]	95.27	93.76	76.75	25.9
TinaFace [10]	95.61	94.25	81.43	38.9
SCRFD-34GF [11]	96.06	94.92	85.29	11.7

The structure of a face detector has a computational distribution. It is important in determining its accuracy and efficiency. Therefore, the model was revised with CR in the study. In this approach, the search area in the model was reduced by controlling the degrees of freedom. In addition, random samples were taken from different components of the model for architectures with different configurations on the backbone, neck, and head. Then, it was calculated bootstrap based on statistics of the models and predicted the probable range in which the best models fall. As a result, a simplified search field was designed by CR among different components of the model (backbone, neck and head).

3.1. ArcFace

In many face recognition studies, it is emphasized that one of the main difficulties in training in deep convolutional neural networks is design of loss functions that can increase discriminative power. ArcFace [20] model also uses the same approach and uses a new loss function. ArcFace, the proposed face recognition model to obtain highly distinctive features, has a clear geometric interpretation as it fits the geodetic distance on a hypersphere precisely with its new loss function. The proposed face recognition model ArcFace uses a new loss function to obtain highly distinctive features. The model has a clear geometric interpretation as it fits geodetic distance on hypersphere exactly with the help of the new loss function.

$$L_1 = -\frac{1}{N}\sum_{i=1}^{N} log \frac{e^{W_{y_i}^T x_i + b_{y_i}}}{\sum_{j=1}^{n} e^{W_j^T x_i + b_j}} \qquad (1)$$

The Softmax loss function shown in Equation 1 is reinterpreted for Arcface. In Equation 1, N represents batch size and n represents class number. x_i represents deep attributes of i sample belonging to y_i class. W_j refers to j column of W weight. b_j represents bias value.

$$L_2 = -\frac{1}{N}\sum_{i=1}^{N} l\,og \frac{e^{s\,cos\theta_{y_i}}}{e^{s\,cos\theta_{y_i}} + \sum_{j=1,\, j\neq y_i}^{n} e^{s\,cos\theta_j}} \qquad (2)$$

In the loss function proposed in Equation 2, which was created by adding new improvements, bias value is accepted as zero. In addition, $cos\theta$ value is included in the loss function by using θ angle between W_j and x_i in $W_j^T x_i = |W_j|.|x_i|\,.cos\theta_j.cos\theta$ is included in the loss function by using angle θ between W_j and x_i. W_j value is normalized with $l2$ form to obtain $|W_j| = 1$ and also embedding property value $|x_i|$ is scaled with s value after normalizing with $l2$ form in the same way. The normalization step ensures that the predictions depend only on angle between feature and weight, and thus embedding features are distributed over a hypersphere of radius s.

$$L_3 = -\frac{1}{N}\sum_{i=1}^{N} l\,og \frac{e^{s\,cos(\theta_{y_i}+m)}}{e^{s cos(\theta_{y_i}+m)} + \sum_{j=1,\, j\neq y_i}^{n} e^{s\,cos\theta_j}} \qquad (3)$$

The embedding features extracted in model are distributed around each feature center in hypersphere. An additional angular margin value m is added between x_i and actual reference value weight W_{y_i} in equation 3 to simultaneously improve within-class density and inter-class mismatch of embedding features. Finally, the obtained embedding features can be compared using the cosine similarity metric.

ArcFace has been compared to facial recognition models that train a large-scale image database containing many face pairs and a large-scale video dataset. As a result of extensive experimental evaluations against other advanced face recognition models, it has been stated that ArcFace performs consistently well and can be easily implemented with negligible computational overhead. ArcFace trained on MS1MV2 dataset which is the semi-automatic refined version of MS-Celeb-1M dataset. Its verification performance is an accuracy of 99.83% in LFW [22] dataset and 98.02% in YTF [23] dataset as shown in Table 2.

Table 2. Verification performance of different approaches on LFW and YTF Datesets [20]

Method	LFW	YTF
FaceNet [33]	99.63	95.10
Marginal Loss [34]	99.48	95.98
SphereFace [18]	99.42	95.0
SphereFace++ [35]	99.47	-
CosFace [19]	99.73	97.6
MSIMV2, R100, ArcFace [20]	99.83	98.02

3.2. DeepSORT

DeepSORT [26] is an algorithm that takes only tracking objects as a task. It differs from both object detection and object tracking approaches. DeepSORT has introduced a new approach by integrating appearance information into new algorithm to improve the performance of the SORT

[25] algorithm. As a result of the experimental evaluations, it was stated that the new improvements reduced the number of ID switches by 45% and a competitive performance was achieved at high FPS speeds. While SORT performs well overall in terms of tracking accuracy, it causes a relatively high number of ID switches. The reason is that the association metric with state estimation uncertainty is only correct. Therefore, SORT algorithm falls short in handling occlusions. DeepSORT overcomes this problem by replacing the association metric with one that combines movement and appearance information. In particular, a trained convolutional neural network is implemented to distinguish pedestrians in a large-scale person dataset. Thanks to integration of this network, it makes the implementation of the system easy, efficient and applicable to online scenarios. it also increases durability against misses and occlusions.

First, the features are extracted from the convolutional neural network using detected objects. The extracted features are passed through a sequential series of distance measurement algorithms. These algorithms are Mahalanobis and Cosine distance measurement calculations. Mahalanobis distance is an association metric that works better when motion uncertainty is low. However, unexplained camera movements can cause rapid displacements in the image plane. This makes the Mahalanobis distance a rather useless metric during occlusions. Therefore, it was necessary to add the Cosine distance measurement metric as a second metric. Mahalanobis distance provides information specifically for short-term predictions, while the Cosine distance is useful in re-identification after long periods of occlusions where movement is less distinctive. In addition, Kalman filter [31] is used with a constant velocity motion and linear observation model. It tries to predict the next state of objects based on their previous state.

3.3. Apache Kafka

Apache Kafka [8] is an open-source distributed event streaming system. This platform is commonly used for high-performance data pipelines, data integration, flow analytics, and mission-critical applications. Apache Kafka is a real-time data capture application from event sources such as videos, sensors, databases, mobile devices, cloud services, and software applications. In addition, processing event streams retrospectively and routing event streams to different target technologies are among the most basic tasks. Kafka combines the following three key features so that it can be implemented with a single end-to-end tested solution for event streaming scenarios:

- Publishing (writing) and subscribing (reading) event streams of data is its most important feature.
- Durable and reliable storage of event streams is another feature.
- Another feature is that it allows event streams to be processed as they occur or backward.

All these functions are delivered as a flexible, highly scalable, fault-tolerant, and secure platform. If any server fails in a distributed architecture in Kafka, the remaining servers can continue to stream without any data loss. It has support for Java, Scala, Go, Python, C/C++, and many other programming languages for Kafka clients. In Kafka, producers are client applications that write events to Kafka. Consumers are applications that read and process these events by subscribing to them. In Kafka, producers and consumers work separately and independently.

Events are organized in such a way that topics are stored permanently. In other words, a topic can be likened to a folder in a file system and events to files in a folder. Unlike traditional queue messaging systems, events are not deleted for a period of time after consumption based on the setting made. How long events should be retained via the per-topic configuration setting is defined on the platform. According to the settings, old events are deleted.

3.4. Socket.IO

Socket.IO [36] is a library that provides bidirectional, real-time and event-based communication between server and client. The library is built as a lightweight wrapper around WebSocket API. In addition, Socket.IO provides the additional features listed below in addition to a plain WebSocket API.

- Enables the HTTP long polling option in case the WebSocket connection with the reliability cannot be established.
- It supports auto reconnection.
- It buffers the packets.
- It has the information whether the package has arrived or not.
- It can broadcast to all clients.
- It can split application logic on a single shared connection.

Socket.IO client connects to the HTTP long polling transport by default. The main reason for this is that it may not always be possible to establish a WebSocket connection due to corporate proxies, personal firewalls, and antivirus software. WebSocket connection failure means that the real-time application waits for a certain amount of time to start exchanging data. This harms user experience in real-time applications. Because of HTTP long polling step works pretty much anywhere, so it’s used in the first place so there’s an immediate connection. Then, an attempt is made to upgrade HTTP long polling connection to a WebSocket connection. If the upgrade is successful, HTTP long-polling stops, and the session switches to the WebSocket connection. If unsuccessful, HTTP long-polling connection remains open and continues to be used.

4. APPLICATION

In this study, real-time face tracking is performed over a multi-camera system using deep learning techniques. One of the biggest problems in video streaming systems is that they can not respond to requests instantly. In addition, it is important that the images taken from the camera are processed quickly without any loss. Therefore, the Apache Kafka tool is used as the backbone of the system. It serves as a high tolerance buffer for a real-time system. Images taken from Apache Kafka were processed in the fastest way with the constructed chain structure. Then, Apache Kafka was used again for database saving operations that would create a bottleneck. With the designed system, it is aimed to take quick action by detecting many problems that may cause alarms by interpreting the image with minimum delay in security or defense.

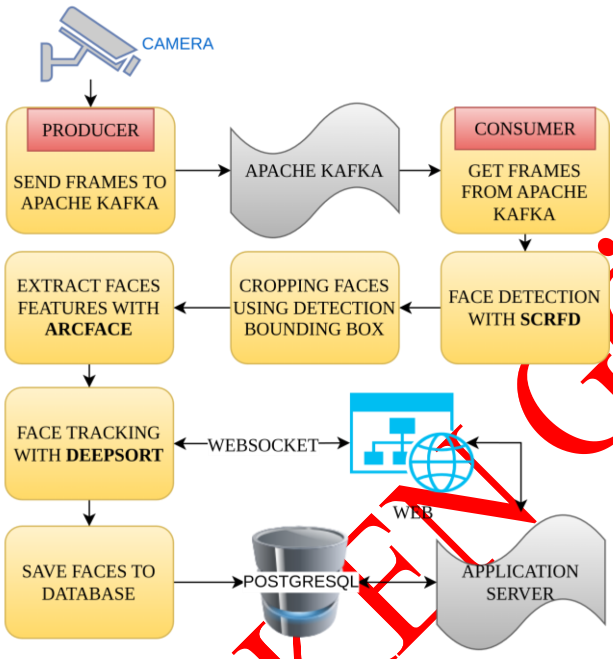

CAMERA
PRODUCER
SEND FRAMES TO APACHE KAFKA
APACHE KAFKA
CONSUMER
GET FRAMES FROM APACHE KAFKA
EXTRACT FACES FEATURES WITH ARCFACE
CROPPING FACES USING DETECTION BOUNDING BOX
FACE DETECTION WITH SCRFD
FACE TRACKING WITH DEEPSORT
WEBSOCKET
WEB
SAVE FACES TO DATABASE
POSTGRESQL
APPLICATION SERVER

Figure 1. Flowchart for a camera

As seen in Figure 1, the flowchart is given for a camera, which is the smallest functioning part of the system. The frames reading from the camera is inserted to Apache Kafka via a producer process to avoid any loss due to image processing bottleneck. Then, a consumer process is run to read from Apache Kafka to process the image. The incoming image is first sent to the face detection model. The faces are cropped from the image with the resulting face bounding boxes and their embedding features are extracted. Finally, the embedding features are compared with cosine similarity to find out who it belongs to. Thus, the results found are sent back to Apache Kafka to be inserted in the database.

Apache Kafka is used as the backbone of the system. Using Apache Kafka as the backbone of the system provides many advantages. Essentially, it provides a unified, high-throughput, low-latency, and scalable platform for processing real-time data streams. It provides a fault-tolerant structure for places where data is critical. The data stream in the system is completely managed by Apache Kafka. The data stream in the multi-camera environment is provided simultaneously via Apache Kafka. A control program is used to manage the simultaneous operations. This code runs as a service on the Linux server to manage Apache Kafka processes, producers and consumers at specified time intervals. This program is responsible for restarting Apache Kafka producers or consumers when they are stopped. In addition, the images streaming to Apache Kafka are processed using a chain of responsibility structure. The first step in the chain of responsibility is face detection. Then there is the face recognition step, which includes face tracking. A new approach has been obtained by adding a face recognition model to the DeepSORT tracking algorithm in face tracking. Thus, stable, effective and low-cost face tracking is provided in the system.

After the face recognition step, the processed images are displayed on the web using the Socket.IO library. In the last step, a log record is added to the PostgreSQL database using the person, camera, and timestamp information. PostgreSQL [37] database is used for all persistent data in the system.

The operation of the system, whose working summary is given above, can be explained in detail in four steps:

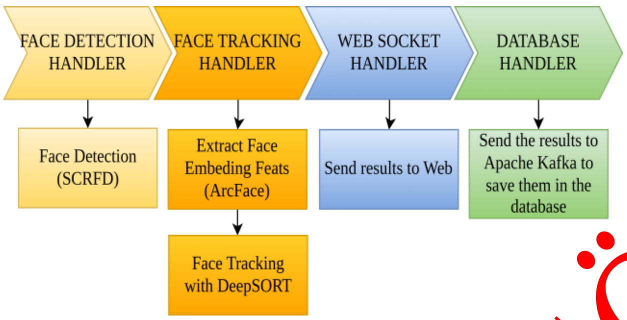
- *Stage 1*: the images streamed from the camera are saved in Apache Kafka.
- *Stage 2*, the images read from Apache Kafka are given to a chain of responsibility structure for processing.
- *Stage 3*, log records are created in the database by using the person identified, timestamp, and camera information.
- *Stage 4*, the processed images are displayed on the web.

At Stage 1 is to run a separate producer process for each camera. Each producer sends the images streamed from the cameras to the corresponding Apache Kafka topic. Each topic is created with the primary key value of the camera. Images are read from the camera using the *read* function in the OpenCV library. The images are converted to textitjpeg image format using the OpenCV *imencode* function. Finally, a byte array of the converted textitjpeg image is sent to Apache Kafka.

At Stage 2, the images stored in Apache Kafka are read for processing by the consumer process. Consumers run as separate processes for each camera. After the consumer connects to the Apache Kafka server, it subscribes to the corresponding topic. It then sends the image to the next step for processing. The consumer is constantly listening for new data on the topic. After the jpeg bytes of the image are read from the topic, they are converted to their original format using the OpenCV *imencode* function. Finally, the images are sent to the chain of responsibility structure

Images read by consumers from Apache Kafka topics are processed sequentially in the chain of responsibility structure. The main purpose of choosing the chain of responsibility design pattern at this step is to provide flexibility for adding new models. Thus, age detection, gender detection etc. models can be easily integrated into the system whenever required.

The chain of responsibility in the system consists of four handlers. As shown in Figure 2, chain of responsibility is used sequentially. First, "Face Detection Handler" is used for face detection and "Face Tracking Handler" is used for face tracking. "Web Socket Handler" is used to send the image to the website and store it in the shared memory. "Database Handler" is used to insert the log record into the database.



FACE DETECTION HANDLER
FACE TRACKING HANDLER
WEB SOCKET HANDLER
DATABASE HANDLER
Face Detection (SCRFD)
Extract Face Embeding Feats (ArcFace)
Send results to Web
Send the results to Apache Kafka to save them in the database
Face Tracking with DeepSORT

Figure 2. Handler chain

The "Face Detection Handler" uses a highly accurate and highly efficient SCRFD face detection model. After the image is given to the model, face bounding boxes are obtained as output. The images to be given to the model input are resized to 640x640. The resulting face bounding boxes are passed to the next step, the "Face Tracking Handler”.

In the "Face Tracking Handler", ArcFace face recognition model is integrated into the DeepSORT tracking algorithm with a new approach as shown in Figure 3.

Face features are extracted by the convolutional neural network (CNN) in the DeepSORT algorithm. 128 feature maps are obtained at the output of the CNN. These features are used to measure whether the newly discovered faces match the tracking faces. To do this, DeepSORT has a two-step process. First, nearest neighbor distance metrics were used. The distances were computed using Cosine and Mahalanobis. Matching was then performed using the Hungarian algorithm. In the second step, intersection over union (IoU) is used to measure overlaps for remaining mismatch detections and tracks. After this step, unmatched tracks that have reached the maximum age are deleted. Otherwise, tracking is started or continued.

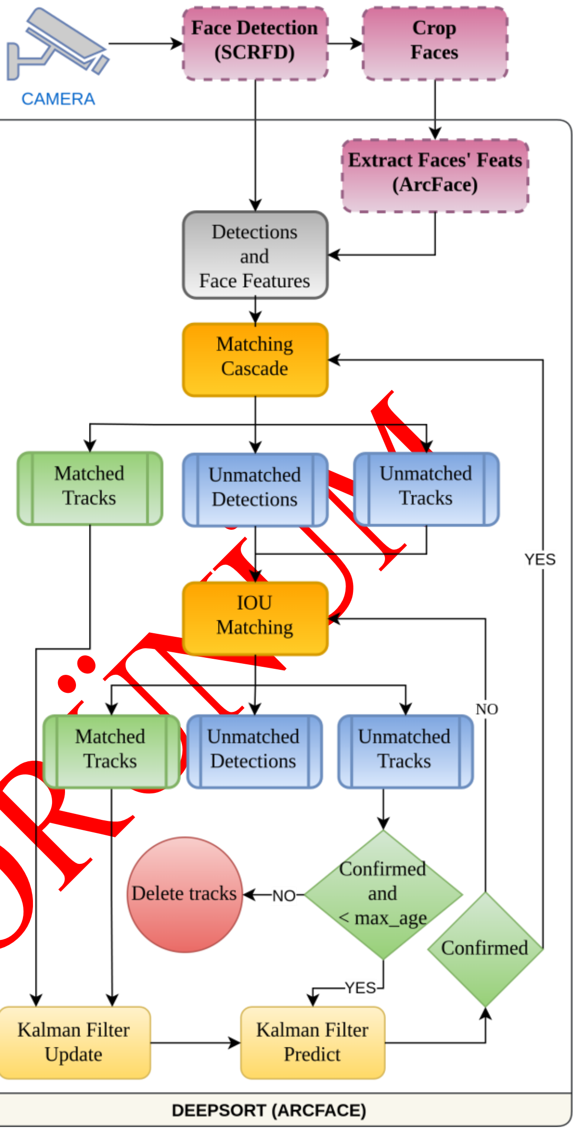


Figure 3. Face tracking model

Face recognition was performed on those whose face tracking was started or continued in DeepSORT. This is done for faces that are detected for the first time or for faces that previously failed face recognition. ArcFace was used as the face recognition model. ArcFace has a proven track record and is preferred because of its high accuracy rate. In this model, each face is fed into the network with an input size of 112x112. Face embedding features of size 512 are obtained in the model output. Then, cosine distances are calculated between the face embedding features in the model output and the face embedding features of the people inserted in the database. As a result, face recognition is performed using cosine distances. Face recognition is retried every 10 frames if it is unsuccessful. To speed up the process, face embedding features of people are loaded from the database into memory. The in-memory data is stored in a key-value structure. As a result, a stable and fast face tracking system has been established in this study.

The processed image in the "Face Tracking Handler" is transferred to the "Web Socket Handler". The purpose of this handler is to send the processed image to the web application at any time. Therefore, streaming images are transferred to shared memory. After the images are converted to *base64* format, they are stored in the shared memory with a key-value pair. The primary key value of the camera is used as the key. The primary key value is the unique value created for the camera in the database. When the camera control page is opened in the web application, a bi-directional communication channel is established with the Socket.IO library and the image stream is provided. The primary key value of the camera is used as the address path information in the communication channel.

"Database Handler is used to insert log records to the database after Web Socket Handler operation is completed. The log record consists of the person's identity, the camera they appeared in, and the timestamp. To prevent the real-time system from being affected by the bottleneck that may occur in the database, the log record is first sent to the "database" Apache Kafka topic. Then, the log record on the "database" topic is inserted into the database by the "DatabaseConsumer" consumer. Log records for all cameras are sent to the database Apache Kafka topic. In other words, all log records to be inserted to the database are collected in the "database" topic.

The system architecture is shown in Figure 4. The system uses the Python programming language. Thanks to Apache Kafka, a horizontally scalable system can be built as the number of cameras increases. In addition, possible data loss can be prevented by storing images in different Apache Kafka partitions for the same Topic.

A real-time face tracking system that can operate in a multi-camera environment is proposed. To make this system to work more stable, a new DeepSORT algorithm including face recognition algorithm is used. As a result of the study, an end-to-end system was designed and the images were presented on the web with low latency.

The study was conducted on a computer with an AMD Ryzen 5 5600X processor, NVIDIA RTX 3060 graphics card, and 16 GB of memory using the PyTorch machine learning library. The study was performed on multiple camera images. During the measurements, the average of the results obtained from 1800 frames was taken. According to the measurements, after the image was entered into the system, it could be displayed on the web page with a delay of approximately 127 ms. Approximately 59 ms of this time is spent on face detection and 34 ms of this time is spent on face tracking. On the other hand, when the DeepSORT algorithm is not used, it was displayed on the web page with a delay of 143 ms. The significance of not using the DeepSORT algorithm is that the face detection model outputs directly to the face tracking model in every frame.

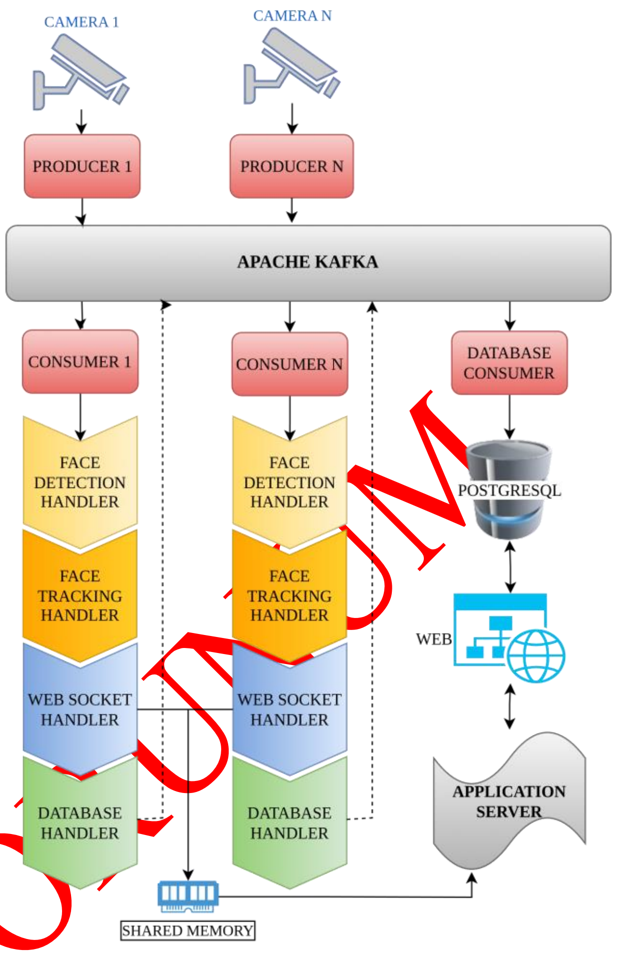


Figure 4. System architecture

5. CONCLUSION

The system used SCRFD for face detection and ArcFace for face recognition. Cost-effective face detection was achieved with SCRFD. ArcFace provided an efficient face recognition model. Face tracking was achieved by combining face detection, face recognition, and DeepSORT algorithms. The use of DeepSORT in face tracking contributed to the efficiency of the system by using the previous image data. In addition, DeepSORT continues face tracking even if the face is not detected or recognized. Thus, more stable face tracking was achieved. As a result, the proposed system was shown to be stable, efficient, and cost-effective.

Apache Kafka event streaming platform and Socket.IO bidirectional communication library are used for the proposed system. Thanks to Apache Kafka, input/output (IO) operations are realized in real time without interruption. In addition, the fault-tolerant structure of Apache Kafka prevents possible data loss. At the operating system level, separate processes are created for each camera. This results in a scalable system and more efficient use of resources.

As a result, one frame in the camera given as input to the proposed system for the multi-camera environment could be displayed at the output with a delay of 127 ms. In addition, most of the time was used for face recognition in the system.

In future studies, a less costly and more successful study can be conducted by using different face detection or face recognition algorithms. In addition, the effects and success of the distributed system can be tested in multi-camera environments. In addition, a stability and efficiency study can be done with different object tracking approaches instead of DeepSORT. A more stable system can be created by optimizing the matching algorithms in DeepSORT. A more efficient system can be created by using different deep learning models in DeepSORT.

ACKNOWLEDGEMENT

This study was supported by Inonu University Scientific Research Projects Coordination Unit (BAP) with the project coded FYL-2021-2449.

DECLARATION OF ETHICAL STANDARDS

The author(s) of this article declare that the materials and methods used in their studies do not require ethics committee approval and/or legal-specific permission.

AUTHORS' CONTRIBUTIONS

Mehmet Fatih ÖZDEMİR: Developed the system architecture. Performed the experiments.

Davut HANBAY: Analyse the results.

CONFLICT OF INTEREST

There is no conflict of interest in this study.

REFERENCES

[1] Deng J., Guo J., Ververas E., Kotsia I., Zafeiriou S., "Retinaface: Single-shot multi-level face localisation in the wild", ***Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition***, 5202-5211, (2020).

[2] Hanbay K., Alpaslan N., Talu M., Hanbay D., Karci A., Kocamaz A., "Continuous rotation invariant features for gradient-based texture classification", ***Computer Vision and Image Understanding*** , 132: 87-101, (2015).

[3] Liu Y., Tang X., Han J., Liu J., Rui D., Wu X., "HAMBox: Delving Into Mining High-Quality Anchors on Face Detection", ***Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)***, (2020).

[4] Li J., Wang Y., Wang C., Tai Y., Qian J., Yang J., Wang C., Li J., Huang F., "DSFD: Dual Shot Face Detector", ***Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)***, (2019).

[5] Üzen H., Hanbay K., "Yaya Özellik Tanıma için LM Filtre Temelli Derin Evrişimsel Sinir Ağı", ***Politeknik Dergisi*** , 23: 605–613, (2020).

[6] AKYEL C., ARICI N., "U-Net-RCB7: Image Segmentation Algorithm", ***Politeknik Dergisi*** , 26: 1555–1562, (2023).

[7] KARADAĞ B., ARI A., "Akıllı Mobil Cihazlarda YOLOv7 Modeli ile Nesne Tespiti", ***Politeknik Dergisi*** , 26: 1207–1214, (2023).

[8] Apache ., "Apache Kafka" , https://kafka.apache.org

[9] Zhang K., Zhang Z., Li Z., Qiao Y., "Joint Face Detection and Alignment Using Multitask Cascaded Convolutional Networks", ***IEEE Signal Processing Letters*** , 23: 1499-1503, (2016).

[10] Zhu Y., Cai H., Zhang S., Wang C., Xiong Y., "Tinaface: Strong but simple baseline for face detection", ***arXiv preprint arXiv:2011.13183*** , (2020).

[11] Guo J., Deng J., Lattas A., Zafeiriou S., "Sample and Computation Redistribution for Efficient Face Detection", (2021).

[12] Viola P., Jones M., "Rapid object detection using a boosted cascade of simple features", ***Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001***, (2001).

[13] Mita T., Kaneko T., Hori O., "Joint haar-like features for face detection", ***Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1***, 1619-1626, (2005).

[14] Zhang L., Chu R., Xiang S., Liao S., Li S., "Face Detection Based on Multi-Block LBP Representation", ***Advances in Biometrics***, Berlin, Heidelberg, 11-18, (2007).

[15] He K., Zhang X., Ren S., Sun J., "Deep residual learning for image recognition", ***Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*** , 2016-Decem: 770-778, (2016).

[16] Yang S., Luo P., Loy C., Tang X., "WIDER FACE: A Face Detection Benchmark", ***Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)***, 5525-5533, (2016).

[17] Turk M., Pentland A., "Eigenfaces for recognition", ***Journal of cognitive neuroscience*** , 3: 71-86, (1991).

[18] Liu W., Wen Y., Yu Z., Li M., Raj B., Song L., "SphereFace: Deep Hypersphere Embedding for Face Recognition", ***2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)***, 6738-6746, (2017).

[19] Wang H., Wang Y., Zhou Z., Ji X., Gong D., Zhou J., Li Z., Liu W., "CosFace: Large Margin Cosine Loss for Deep Face Recognition", ***2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition***, 5265-5274, (2018).

[20] Deng J., Guo J., Xue N., Zafeiriou S., "ArcFace: Additive angular margin loss for deep face recognition", ***Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*** , 2019-June: 4685-4694, (2019).

[21] Boutros F., Damer N., Kirchbuchner F., Kuijper A., "ElasticFace: Elastic Margin Loss for Deep Face Recognition", ***Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops***, 1578-1587, (2022).

[22] Huang G., Mattar M., Berg T., Learned-Miller E., "Labeled faces in the wild: A database forstudying face

recognition in unconstrained environments", ***Workshop on faces in'Real-Life'Images: detection, alignment, and recognition***, (2008).

[23] Wolf L., Hassner T., Maoz I., "Face recognition in unconstrained videos with matched background similarity", ***CVPR 2011***, 529-534, (2011).

[24] Kemelmacher-Shlizerman I., Seitz S., Miller D., Brossard E., "The MegaFace Benchmark: 1 Million Faces for Recognition at Scale", ***2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)***, 4873-4882, (2016).

[25] Bewley A., Ge Z., Ott L., Ramos F., Upcroft B., "Simple online and realtime tracking", ***Proceedings - International Conference on Image Processing, ICIP*** , 2016-Augus: 3464-3468, (2016).

[26] Wojke N., Bewley A., Paulus D., "Simple online and realtime tracking with a deep association metric", ***Proceedings - International Conference on Image Processing, ICIP*** , 2017-Septe: 3645-3649, (2018).

[27] Zhang Y., Sun P., Jiang Y., Yu D., Weng F., Yuan Z., Luo P., Liu W., Wang X., "ByteTrack: Multi-Object Tracking by Associating Every Detection Box", , (2022).

[28] Cao J., Pang J., Weng X., Khirodkar R., Kitani K., "Observation-centric sort: Rethinking sort for robust multi-object tracking", ***Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition***, 9686-9696, (2023).

[29] Rambach J., Huber M., Balthasar M., Zoubir A., "Collaborative multi-camera face recognition and tracking", ***2015 12th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)***, 1-6, (2015).

[30] Lian Z., Shao S., Huang C., "A Real Time Face Tracking System based on Multiple Information Fusion", ***Multimedia Tools and Applications*** , 79: 16751-16769, (2020).

[31] Welch G., Bishop G., Others ., "An introduction to the Kalman filter", (1995).

[32] Badave H., Kuber M., "Head Pose Estimation Based Robust Multicamera Face Recognition", ***2021 International Conference on Artificial Intelligence and Smart Systems (ICAIS)***, 492-495, (2021).

[33] Schroff F., Kalenichenko D., Philbin J., "FaceNet: A Unified Embedding for Face Recognition and Clustering", ***Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)***, (2015).

[34] Deng J., Zhou Y., Zafeiriou S., "Marginal loss for deep face recognition", ***Proceedings of the IEEE conference on computer vision and pattern recognition workshops***, 60--68, (2017).

[35] Liu W., Lin R., Liu Z., Liu L., Yu Z., Dai B., Song L., "Learning towards minimum hyperspherical energy", ***Advances in neural information processing systems*** , 31:, (2018).

[36] Rauch G., "Socket.IO" , https://socket.io

[37] Stonebraker M., Rowe L., "The Design of POSTGRES", ***ACM SIGMOD Record*** , 15:, (1986).