Computational Intelligence and Neuroscience

The New High-Performance Face Tracking System based on Detection-Tracking and Tracklet-Tracklet Association in Semi-Online mode

NGOC Q. LY,[1] Tan T. Nguyen,[1,2] Tai C. Vong,[1] and Cuong V. Than [2]

[1] Faculty of Information Technology, VNUHCM-University of Science, 700000, Vietnam.
[2] AI Department, Axon, Ho Chi Minh, 700000, Vietnam.

Correspondence should be addressed to Tan T. Nguyen; ttnguyen@axon.com

Abstract

Despite recent advances in multiple object tracking and pedestrian tracking, multiple-face tracking remains a challenging problem. In this work, we propose a framework to solve the problem in semi-online manner (the framework runs in real-time speed with two-second delay). Our framework consists of two stages: detection-tracking and tracklet-tracklet association. Detection-tracking stage is for creating short tracklets. Tracklet-tracklet association is for merging and assigning identifications to those tracklets. To the best of our knowledge, we make contributions in three aspects: 1) we adopt a principle often used in online approaches as a part of our framework and introduce a tracklet-tracklet association stage to leverage future information; 2) we propose a motion affinity metric to compare trajectories of two tracklets; 3) we propose an efficient way to employ deep features in comparing tracklets of faces. We achieved 78.7% precision plot AUC, 68.1% success plot AUC on MobiFace dataset (test set). On OTB dataset, we achieved 78.2% and 72.5% precision plot AUC, 51.9% and 43.9% success plot AUC on normal and difficult face subsets respectively. The average speed was maintained at around 44 FPS. In comparison to the state-of-the-art methods, our performance maintains high rankings in top 3 on two datasets while keeping the processing speed higher than the other methods in top 3.

Introduction

While multiple object tracking has been receiving much attention from researchers all over the world, multiple-face tracking has received much less attention due to two main reasons: face tracking is a sub-problem of object tracking thus many works focus on the general problem, and there is a lack of encompassing multiple-face tracking datasets. Therefore, multiple-face tracking remains a challenging problem. Recent advances in the field of multiple pedestrian tracking can be used to solve the problem of multiple-face tracking. There are two main research directions for the problem: online and offline.

Offline approaches [1]–[6] treat the problem as a global optimization one and solve it once having received all the information of all frames of a video. These approaches basically revolve in three stages:

Stage 1: Apply detection algorithms over all frames of the video to get detected bounding boxes of individuals, which is treated as nodes of a graph.

Stage 2: Define a meaningful metric to measure the relationship between two nodes of the graph by employing visual, spatial and temporal information.

Stage 3: Optimize an objective function globally to get clustered the bounding boxes of individuals.

These approaches tend to use commonly known detectors to generate all detection boxes (stage 1). However, these methods are different from each other in defining relations between nodes (stage 2) and objective functions (stage 3). [1] proposes to model all potential locations over time, find trajectories that produce the minimum cost and track interacting objects simultaneously by using intertwined flow and imposing linear flow constraints. [2] employs an energy function that considers physical constraints such as target dynamics, mutual exclusion, and track persistence. [4] proposes to jointly cluster detections over space and time. The optimal number of people as well as the cluster of each person are obtained by partitioning the graph with attractive and repulsive terms. [6] introduces two types of edges (regular and lifted edges) for the tracking graph: the regular edges define the set of feasible solutions in the graph and the lifted edges add additional long-range information to the objective. The authors of [6] also employ human pose features extracted from a deep network for the detection-detection association. Solving the problem with no constraints of speed while having all the information beforehand, offline approaches often produce higher accuracy than online approaches summarized as follows.

Online approaches mainly focus on tracking by detection [7]–[15]. Basically, they employ three models: a state-of-the-art detection model to produce face detection bounding boxes, a standalone tracker [16]–[19] to produce face track bounding boxes, and a deep feature model [20]–[26] to extract representative features for matching. Combining detection and tracking methods help alleviate challenges when using stand-alone trackers such as sudden movements, blurring, pose variation. Specifically, because detection boxes are often neater (close to faces) than track boxes while track boxes contain spatial and motion information of the objects, fusing detection boxes and track boxes help mitigate the problem of accumulated error in trackers. By adopting the detection-tracking framework, the problem of face tracking is then reduced to data association [27], [28] problem, that is to assign detection boxes to track boxes. Data association [27], [28] between detection boxes and track boxes is then can be reduced to the bipartite matching problem (we assume no two detection boxes in one frame belong to one individual, and so for track boxes) and can be efficiently solved by Hungarian algorithm [29]. Because bipartite matching algorithms find 1-1 matches, it is crucial to define a meaningful affinity metric, representing the relationship between two nodes, for good performance.

These online approaches can be simplified as follows:

Step 1: For each frame, run a detection model to get possible positions of faces in that frame (we will refer these results as detections). Then we apply a deep feature model to extract features of these detections.

Step 2: Also, for that frame, run a tracker for each tracklet to get new possible positions from the previous position of each tracklet (we will refer these results as predictions). Then we apply a deep feature model to extract features of these predictions.

Step 3: A defined metric is employed to relate detections with predictions. The metric consists of two parts: motion affinity and appearance affinity. Motion affinity is measured by the intersection over union (or Mahalanobis distance) of detections and predictions. Appearance affinity is measured by Euclidean (or cosine) distance between features of detections and features of predictions (or possibly of tracklets).

Step 4: After three steps above, we now have an affinity matrix (N detections x M predictions). We apply a bipartite matching algorithm to associate new detections with predictions. Unassigned detections are treated as new individuals while assigned detections are used to update tracklets.

Step 5: Repeat steps 1-4 consecutively for frames of a video.

There are some disadvantages to these online approaches.

Disadvantage 1: At the ith frame we must assign identities to new detections at that frame. This means we cannot take advantage of the information in the future.

Disadvantage 2: To decide whether a new detection belongs to a known identity or is a new identity, we rely on the similarity matrix (computed by motion and appearance affinity) thresholded by a scalar value. To have as few the number of tracklets for one individual as possible, we must lower the threshold. However, doing that way, the possibility of one track containing many individuals is high.

Disadvantage 3: Because we must run detection model and tracking algorithm for each frame to get new detections and new predictions, then run deep feature model (models used for feature extraction are computationally expensive) for new detections and new predictions, these models must be light to run in real-time. This can lead to low accuracy in these models and causes errors for the whole framework.

Disadvantage 4: Because these approaches compare detections with predictions, they fail to employ very potential information that we can take advantage when we compare tracks with tracks. That is the fact that two temporal-overlapped tracks cannot belong to the same individual.

To resolve the issues stated above, we propose a semi-online framework for the multi-face tracking problem. The framework consists of two stages: detection-tracking stage and tracklet-tracklet association stage. For the detection-tracking stage, we employ the same principle as in online approaches with a modification: we use two complementary trackers (Kalman filter as a motion tracker and KCF (Kernelized Correlation Filter) as a visual tracker) to improve accuracy. For the tracklet-tracklet association, inspired by offline approaches, we treat each tracklet as a node of a graph and optimize the problem of assigning identifications globally. In this stage, we also introduce an efficient metric to compare two tracklets so that the framework can run with high speed.

Materials and Methods

Related works

Offline tracking

State-of-the-art methods for multi-face offline tracking are [30]–[32]. These approaches can be reduced to two main stages: tracklet creation (tracking-by-detection) and tracklet association. In [30], the authors first divide the video into many non-overlapping shots – music or film videos often contain many shots in different scenes. For each shot, the framework employs the tracking-by-detection paradigm to generate tracklets and merge those tracklets into groups by temporal, kinematic (motion, size) and appearance (deep feature) information. Then, the authors link tracklets across shots/scenes by treating each tracklet as a point, the appearance similarity between two tracklets as edge and applying the Hierarchical clustering algorithm to assign tracklets into groups. To increase the accuracy of the tracklet linking step, a discriminative feature extractor is needed. The authors introduce Learning Adaptive Discriminative Features whereby a deep extractor will be finetuned online based on samples from the video. [31] improve the performance of the mentioned method by using a more powerful detector (Faster R-CNN) in the tracking-by-detection stage and a more sophisticated tracklet association schedule. [32] pushes it further by applying body parts detector and introduce a co-occurrence model to generate longer tracklets when faces are out of camera (but body not) or detector cannot capture faces. Besides, the authors also introduce a refinement scheme for tracklet association based on Gaussian Process.

Online tracking

Hand-crafted features

One of the attempts to solve the multi-face online tracking problem that yield good results is [33]. In this work, the authors adopt the tracking-by-detection mechanism for the pipeline (Figure 1). Because of the frontal characteristics of the dataset being used, the authors employ a Haar-like cascade face detector [34] to attain computational efficiency. In any tracking problem, the ability to learn appearance change and predict future states of objects is crucial for the model. Thus, the authors introduce a structured SVM tracker that store previous patterns and positions of an object and can predict the new state of an object based on current spatial and visual information. The tracker is updated online based on both track prediction and detection. In the data association step, this work applies Hungarian algorithm for the cost matrix computed by the intersection over union of detection boxes and track boxes.

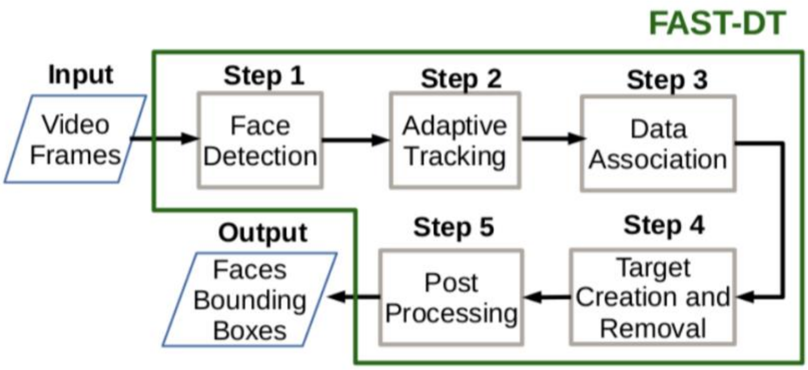


Figure 1. Multi-face detection and tracking framework [33]

Similar to the above work, [35] also adopt tracking-by-detection mechanism but with a more sophisticated tracker update routine. [36] try to decrease the false negative rate (miss detection caused by a simple detector) of the previous pipeline without reducing speed. In this work, the authors adopt an advancement of [34] and a color-assisted tracker as detect and track components respectively (Figure 2). The novelty of this work lies in the combined framework. Instead of running a detector for every frame like previous work, the authors propose a trigger mechanism so that the detector only need to run on some specific frames. Specifically, the detector is only triggered after a fixed interval (N frames) or earlier, when there is any tracking fail. The authors compare the histogram of the new track box with histograms of previous track boxes. If there is any large discrepancy, the track fail will trigger detection.

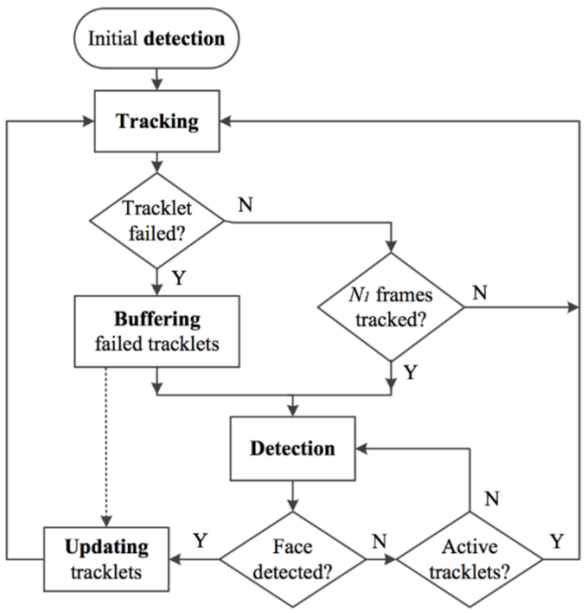


Figure 2. Multi-face tracking detection and tracking flow[36]

Similarly, [37] adopts the idea of sparse detection, modifies Viola-Jones detector in conjunction with a variant of optical flow to create a combined detection-tracking model.

Deep features

Recently, many works [38]–[42] integrate deep feature extractors into the tracking framework. Of those works, [38] adopts the sparse detection mechanism as described above and use KLT tracker [43] for the tracking-by-detection stage. In the data association step between detection boxes and track boxes, deep feature vectors are used as visual information in addition to spatial information.

Our approach overviews

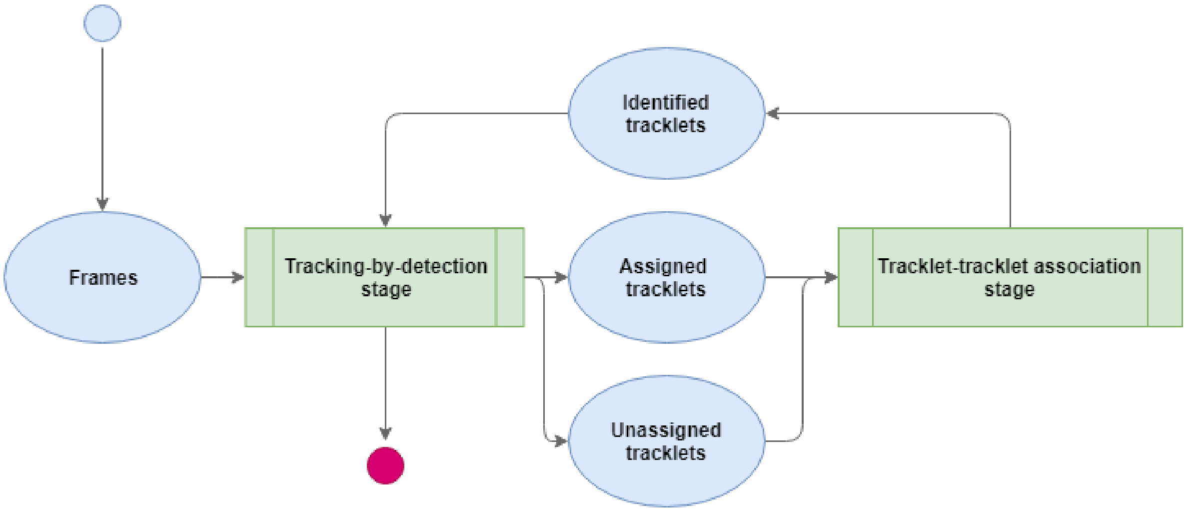


Figure 3. Our proposed method. The extra tracklet-tracklet association is introduced to improve accuracy by using more information and lighten the process before.

Semi-online tracking

Aiming for practical usage and from the analysis of the online detection-tracking approaches, we propose a new approach in semi-online manner by introducing the tracklet-tracklet association stage (Figure 3).

After getting the detections of a frame, we should match it with tracklets up until the previous frame to determine the identification of new detections. To achieve this criterion, using a deep feature extractor is a heavy waste. We propose a way to lighten the process while keeping the accuracy as high as possible. First, we use a light feature LBPH (Local Binary Pattern Histogram) extractor in **the detection-tracking stage** (Figure 5) for efficient computation and combine it with information from a tracking method (Kalman filter) to reduce the errors as much as possible in creating short tracklets (we have not yet assigned identifications for those tracklets). Then we observe that consecutive face boxes of one tracklet are nearly the same, thus in the **tracklet-tracklet association stage** (Figure 8) **,** we introduce a compression method to get representatives of a tracklet and apply a deep feature extractor on these representatives instead of all boxes. We then link short tracklets into long tracklets by using those features as appearance information. In the linking step, we also introduce a new method for motion similarity between two tracklets. The tracklet-tracklet association stage resolved much problems stated above: the future information of frames sequences is well manipulated; the computational complexity is cut off from deep feature comparison by applying the new compression method.

The end-to-end framework consists of two stages:

Detection-Tracking stage: The main role of this stage is to extract the track information of targets in a frame using detecting and tracking methods. Technically, the detection-tracking stage processes frame-by-frame for every mini-batch interval (i.e. 60 frames) and yield a list of tracklets. The process is illustrated in Figure 4.

Tracklet-tracklet association stage: At the end of each mini-batch process, the list of tracklets is passed to this stage. The main role of this stage is to correct false positives of the previous stage and connect related tracklet to create long tracklets and then assign identifications to these new tracklets. The process is showed in Figure 7.

Computational complexity

Our framework can process video streaming in real-time. The speed can reach around 60fps, which is greater or equal the frequency of common videos (from 30 to 60fps).

Detection-Tracking stage

We leverage known detection-tracking approaches with some modification to speed up the stage without sacrificing much performance and introduce a new stage to improve the performance. We also implemented a framework: the detection-tracking stage combining S3FD face detector to produce detection boxes, LHBPs feature extractor to extract the global features,

Kalman Filter tracker to produce tracking boxes, then Hungarian algorithms for matching the corresponding boxes to create tracklets.

After getting the detection information of a frame, we should match it with detection information of the previous frame to determine if they are the same identification. To achieve this criterion, using a deep feature extractor is a heavy waste. We proposed a way to lighten the process while keeping the accuracy as high as possible. First, we use a light feature extractor for efficient computation and combine it with information from another tracking method like Kalman to reduce the errors as much as possible. Then we can extract information from a mini-batch of frames to correct the process later while gaining more useful information, trade of some delays.

Tracklet-Tracklet association stage

The tracklet - tracklet association stage uses the motion information simulated by the spline interpolation and appearance information from FaceNet deep feature extractor to drop the false positives and match the suitable tracklets to accurately assign the ids for targets.

Detection – tracking stage

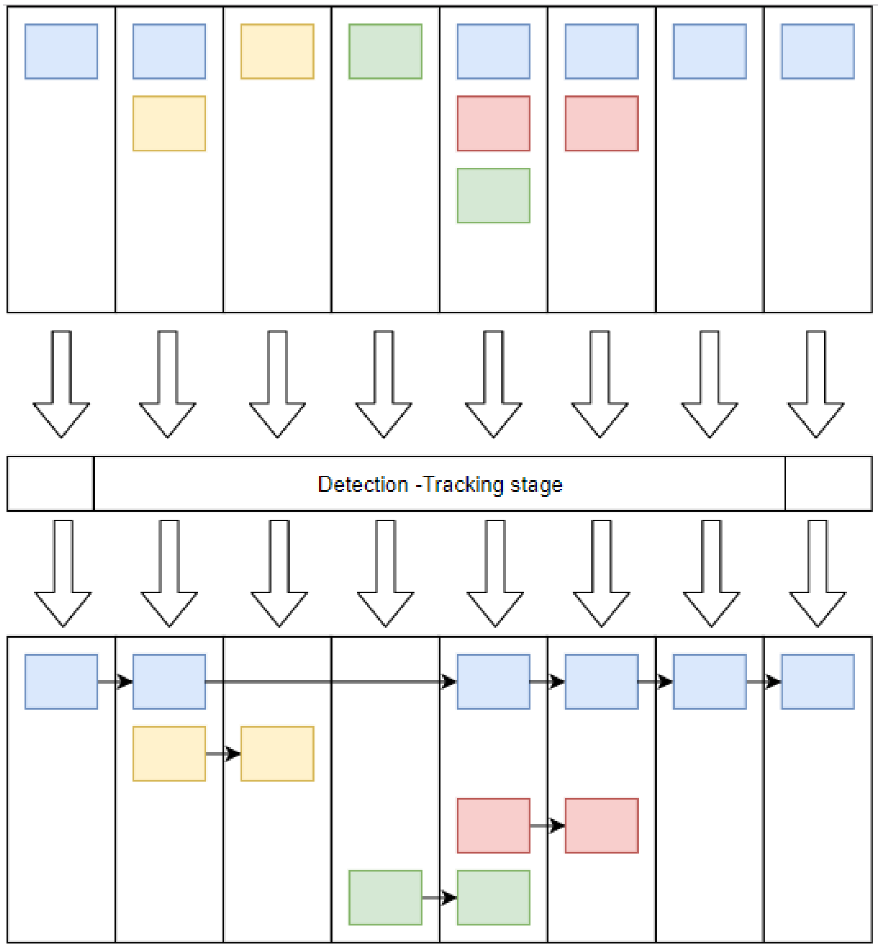


Figure 4. Detection-tracking stage (frame by frame). Columns are consecutive frames; each box is a tracked box in each frame; the arrows show how a tracklet is formed; Each identity is marked by different colors in each box.

Goal

In this stage, all the detection boxes of all frames in a batch will be grouped into short tracklets with the help of a single object tracking method.

Principle

Combining a single tracker and a detector helps a lot in overcoming the limitation of each single method. Using single trackers [16]–[19] to track faces in the wild situation is hard due to occlusion, illumination change, pose variation, sudden movement, etc. These issues can lead to track losses, inaccurate boxes (boxes that capture part of the face), incorrect boxes (boxes that capture the face of another individual). Moreover, using only a detector faces the appearance feature confusion if there are faces of different individuals with high appearance similarity.

We observe that detection models yield neater boxes than single trackers so using detection boxes as new information for updating single trackers is reasonable.

Method

In this stage, a detection model is used to generate possible bounding boxes of faces in a frame. During that time, a tracker is also used to predict a new possible bounding boxes positions from previous frames. Our detection-tracking algorithm will try to fuse these detection results with track results in order to better enhance the output, create more reliable tracklets

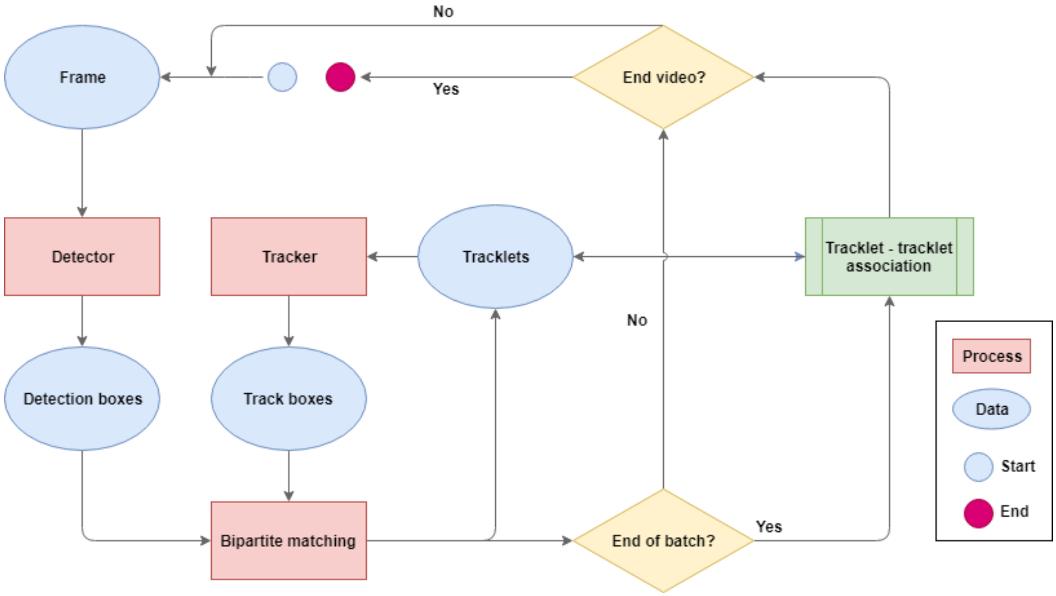


Figure 5. Our detection – tracking flow diagram.

At each frame, after running the detection and tracking process, we then get a list of (N) detection boxes and (M) track boxes. The track boxes are the spatial predictions of bounding boxes from previous tracklets, while the detection boxes are the bounding boxes of faces that existed in that frame. Those faces may be the old faces from the previous frames, but they may also be the new faces that only exist from that frame. The main purpose of the detection-tracking algorithm is to define a meaningful affinity matrix (N x M) so that it can reflect the relationships between those detection boxes and track boxes.

Two features that are commonly leveraged are motion and appearance:

Motion affinity between a detection box and a track box is defined by the intersection over union (IoU) of them.

Appearance affinity between a detection box and a track box is defined by cosine affinity between LBPHs features of them.

Those two features are used because for a pair of detection box and track box to be matched, two boxes should be close to each other with similar size and visual feature.

We define a gating unit for each affinity in order to filter out less likely matches. Because of our intention that if a detection box and a track box are considered a possible match, they must satisfy motion affinity alone and appearance affinity alone first.

As explained, we want both metrics to be high to treat a pair of detection box and track box a likely match; thus, if both affinity metrics pass the threshold then the final affinity is the multiplicative result of motion and appearance affinity, otherwise is zero.

$$Match(i,j) = \begin{cases} s_m(i,j).s_{\mathrm{a}}(i,j) & if\ s_m(i,j) > \gamma_{\mathrm{M}}\ and\ s_{\mathrm{a}}(i,j) > \gamma_{\mathrm{A}} \\ 0 & else \end{cases} \quad (1)$$

where,

$s_a(i, j)$ describes the appearance similarity distance between bounding boxes i and j, its range is from 0 to 1.

$s_m(i, j)$ describes the space similarity distance between bounding boxes i and j, its range is from 0 to 1.

γ_M is the threshold for space similarity distance determined by heuristic (we reason that detection box and track box should be near to be of one individual, so we set this value to 0.3).

γ_A is the threshold for appearance similarity distance determined by heuristic (the purpose of this stage is to create short tracklets, we use a high threshold to prevent wrong matches, specifically 0.9).

$Match(i, j)$ will be used to determine if a detection box and a track box is a possible match. It only has value if both motion and appearance metrics are over their thresholds. Otherwise, its value is zero, its range is from 0 to 1. The thresholds for $Match(i, j)$ are determined through experiments (value search).

Algorithm Tracklet-Tracklet Association

We present the algorithm as pseudo-code in Figure 6.

There are some variables we must notice:

- *E(i)*: the i-th frame.
- *affinity_matrix*: an affinity matrix between detection boxes and track boxes.
- *not_match_threshold*: number of unassigned times for a tracklet to be considered inactive.
- *motion_threshold*: minimum IoU to consider a pair of detection box and track box a possible match, stated as γ^M in equation (1).
- *appearance_threshold*: minimum cosine affinity threshold to consider a pair of detection box and track box a possible match, stated as γ^A in equation (1).
- *affinity_threshold*: minimum threshold to consider a pair of detection box and track box a match.

Algorithm 1: Detection-tracking tracklet assignment algorithm

Function Detection-tracking tracklet assignment

Input:

Ta: a list of active tracklets (a tracklet is considered active if temporal difference between the current frame and the last frame it has matched a detection box is less than *not_match_threshold*)

Ti: a list of inactive tracklets (a tracklet is considered inactive if temporal difference between the current frame and the last frame it has matched a detection box is greater than or equal *not_match_threshold*)

Output:

Ta: a list of active tracklets (updated)

Ti: a list of inactive tracklets (updated)

```
Begin
for frame in frames_of_batch do
    // Run detector on frame F(i)
    detections = detect_func(frame)
    track_predictions = []
    just_stopped_boxes = []

    // Run tracker for each active tracklet on frame F(i)
    for tracklet in Ta do
        // run tracker on frame i-th to find track box
        found, prediction = track_func(tracklet, F(i))
        // if found prediction, add to a list to match with detections
        if found:
            track_predictions.append(prediction)
            // if not found, lower its priority in association step
        else:
            just_stopped_boxes.append(tracklet.get_box_at_previous_frame)
        end if
        Ta.remove(tracklet)
    end //for tracklet
    // Construct distance matrix for association
    affinity_matrix = matrix(len(track_predictions), len(detections))
    for prediction in track_predictions do
        for detection in detections do
```

```
        affinity_matrix[prediction][detection] = get_affinity(detection, prediction)
    end //for detection
end //for prediction
// Hungarian function return:
// a list of unassigned tracklets
// a list of unassigned detections
// a list of pairs of detection-tracklet
unassigned_tracklets, unassigned_detections, matches = Hungarian(affinity_matrix)
for detection, tracklet in matches do
    tracklet.update(detection)
end //for detection, tracklet
for tracklet in unassigned_tracklets do
    tracklet.non_detection_streak += 1
    // if tracklet has not been assigned for any detection in streak_threshold times
    // assign tracklet to stopped state
    if tracklet.non_detection_streak > streak_threshold:
        tracklet.state = inactive
        Ta.remove(tracklet)
        Ti.append(tracklet)
    endif
end //for tracklet
affinity_matrix = matrix(len(just_stopped_tracklets), len(unassigned_detections))
for prediction in just_stopped_boxes do
    for detection in unassigned_detections do
        affinity_matrix[prediction][detection] = get_affinity(detection, prediction)
    end //for prediction
end //for detection
unassigned_tracklets, unassigned_detections, matches = Hungarian(affinity_matrix)
for tracklet in unassigned_tracklets do
```

```
        Ta.remove(tracklet)
        Ti.append(tracklet)
    end //for tracklet
    // initialize new tracklet by unassigned detections
    for detection in unassigned_detections do
        Ta.append(init_tracklet(detection))
    end //for detection
end //for frame
End //Function Detection-tracking tracklet assignment
```

Figure 6. Detection – tracklet assignment

Tracklet stores an internal tracker, a list of boxes (these boxes can be detection boxes or track boxes) capturing one individual across frames and other information used in the algorithm (state, non_detection_streak, etc.). The tracklet assignment process can be summarized as follow:

Step 1. For frame i-th, we run a face detector on that frame to get detection boxes (referred in the pseudo-code as detections).

Step 2. For each tracklet that is active, we run an internal tracker of that tracklet on frame i-th to get a track box, if the tracker of the tracklet die (i.e. cannot find a box) in this frame, we put the last box (either detection or track) of that tracklet in *just_stopped_boxes*, otherwise we put the new track box in *track_predictions*.

Step 3. From the above lists of detection boxes and tracking boxes, we construct an affinity matrix between track predictions and detections. We then apply Hungarian algorithm to assign detections to *track_predictions* based on this *affinity_matrix*. The result of this algorithm is a list of pairs of detection boxes and tracklets being assigned, a list of unassigned detection boxes, a list of unassigned tracklets.

Step 4. Based on the results of Hungarian algorithm, we update matched tracklets and unmatched tracklets.

Step 5. Repeat step 3 and step 4 for *just_stopped_boxes* and *unassigned_detections*.

Step 6. Repeat the above steps for the whole batch (i. e. 60 frames).

We then use the output of this algorithm for the tracklet-tracklet association stage. All the tracklets taken from this stage is regarded as unknown id tracklets in the next stage.

Tracklet-tracklet association

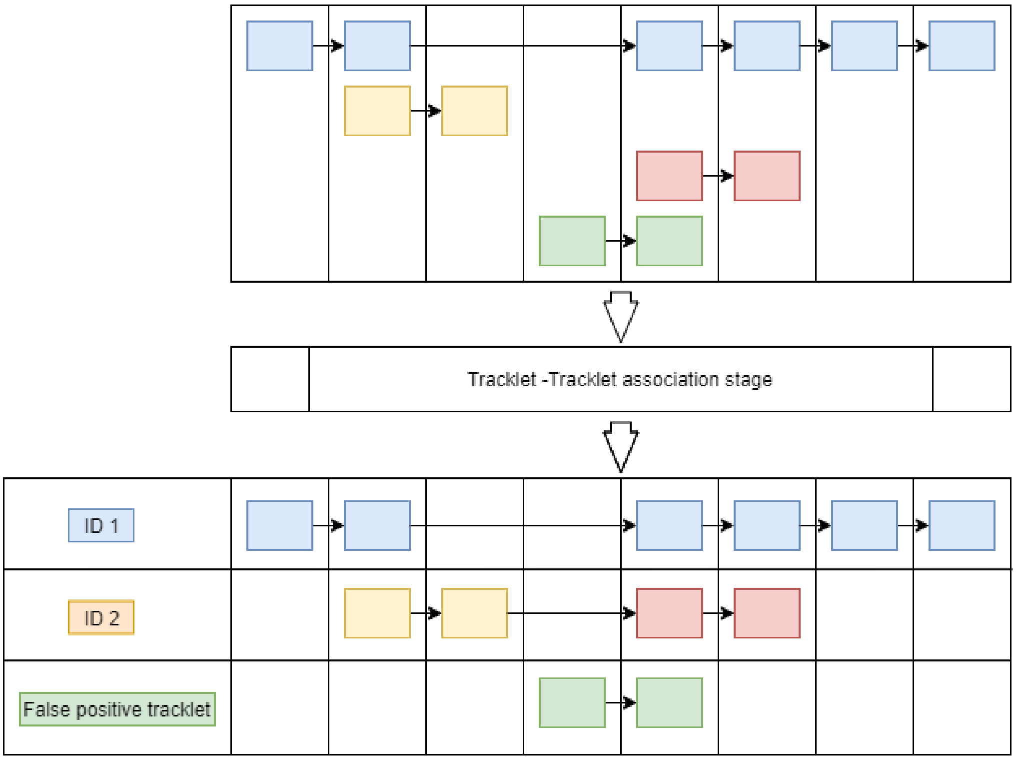


Figure 7. Tracklet-tracklet association stage. From tracklets formed before, the identities will be determined in this stage.

Goal

Short tracklets from the detection-tracking stage are passed to this stage. We will group short tracklets into long tracklets and assign identifications for them. After this stage, the boxes in each frame will be marked with identifications and ready to deliver to the result stream.

Principle

The objective of face tracking is that for everyone existed in a video, the framework should output as few as possible the number of tracklets for that individual without wrongly including other faces of other individuals. This leads to the tradeoff mentioned in section 1. We tackle this with two principles:

- Make sure the possibility of wrongly matching is as low as possible by using tight constraints (high affinity thresholds).
- Adopt efficient motion and appearance affinity metrics between tracklets (different from track-detection) to group tracklets into identities based on a community discovery algorithm in this stage.

Method

After each batch processing the detection-tracking stage, we have a list of unknown-id tracklets that are needed to be assigned identifications in this stage. We also have a list of known-id tracklets in the past (previous batches). Our job is now trying to assign identifications to unknown-id tracklets.

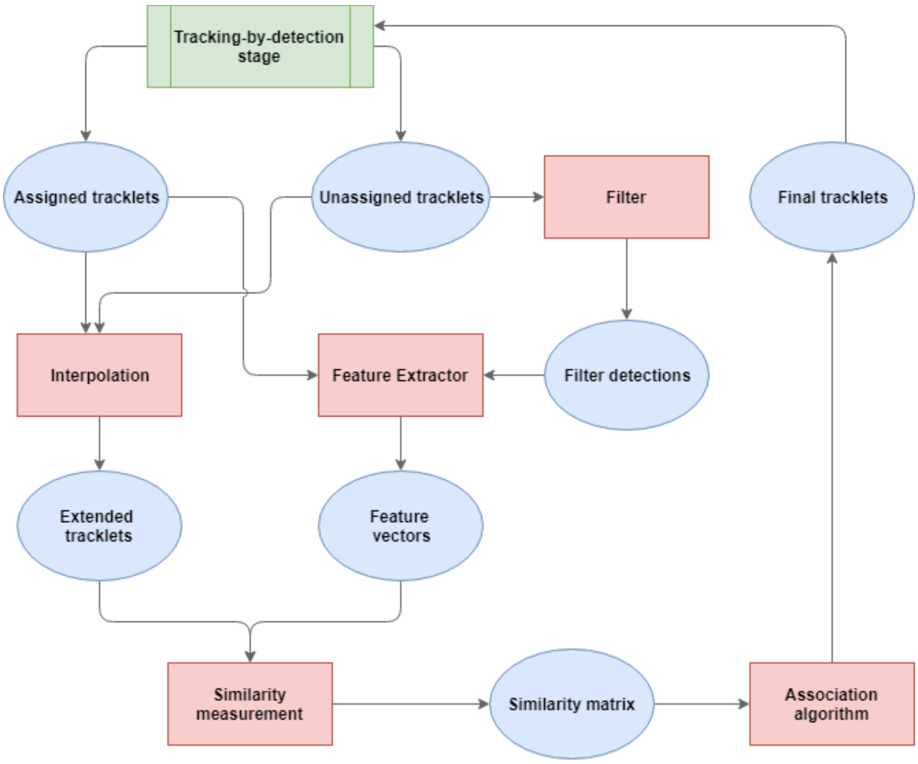


Figure 8. Our tracklet-tracklet association flow diagram.

We formulate the assignment puzzle as an optimization problem. Each tracklet is treated as a node of a graph. The edge of two nodes indicates the affinity between the two. We then apply a clustering algorithm, in this situation, Leiden algorithm[28] on this graph in order to partition it into subgraphs – groups, each containing tracklets - nodes of the same individual. We put constraints so that each subgraph will not contain two known-id tracklets or two temporally overlapped tracklets. One of the essential parts of this stage is defining a meaningful metric representing the edge of two nodes. To do that, we adopt the complementary nature of motion and appearance.

Motion distance

For motion, we introduce a trajectory difference metric. Given two tracklets (t(i), t(j)), it is safe to assume that t(i) predate t(j) and there is no temporal overlap between two tracklets. From the boxes of t(i), we extrapolate forward to get the possible boxes in the future relative to t(i). From the boxes of t(j), we extrapolate backward to get the possible boxes in the past relative to (t(j). For extrapolation, we assume that face movement can be modeled as a polynomial function and apply spline extrapolation. We ran model selection to determine the degree of movement and found that 1-degree spline performs best. Now the extrapolated parts of the two overlap temporally, we have a pair of overlapped extrapolated boxes in the same frame f(k). We now calculate a spatial distance between two boxes using two centers and a diagonal distance between two boxes according to their diagonals. We introduce a weight parameter to fuse the two distances into one unified box-box distance.

The box-box distance at frame k can be formulated in the following equation:

$$d_{M,k} = \lambda.\, d_{S,k} + (1 - \lambda).d_{D,k} \quad (2)$$

In that,

$d_{S,k}$ is the Euclidean distance between two centers of two boxes.

$d_{D,k}$ is the diagonal distance between two boxes calculated by the difference in length between two diagonals.

λ is the weight parameter to fuse above distances into one unified distance (we search from 0 to 1 with 0.1 interval and choose 0.4 to maximize area under the curve of success plot).

$d_{M,k}$ is the box-box distance at frame k we are going to obtain.

Then the trajectory distance is the average of pair distances:

$$d_M = \frac{1}{n-m+1} \sum_{k=m}^{n} d_{M,k} \quad (3)$$

where,

$k = m \rightarrow n$ are overlapped frame indices.

$d_{M,k}$ is the box-box distance at frame k.

d_M is the trajectory distance, the average box-box distance over $m - n + 1$ frames.

Appearance distance

For appearance, we use average Euclidean distance between two feature sets of two tracklets. For each box of a tracklet, we have a respective LBPHs feature (referred to as light feature) extracted from the detection-tracking stage. Assume t(i) have N light feature vectors and t(j) have M light feature vectors, one straight forward method is to compute N*M Euclidean distances and use the average as the distance between two tracklets. From experiments, we observe that LBPHs feature is not representative enough for this task. Thus, we adopt a deep feature extractor [20] for this task.

However, deep feature extractors are computationally expensive and if we compute deep features for all boxes of a tracklet the framework would not run in real-time. Moreover, temporally adjacent boxes often contain similar information, so it would be redundant to compute all the deep features. We introduce a mechanism to lower the number of boxes needed to be passed through a deep feature extractor using already computed light features.

Given a list of light feature vectors of a tracklet, we apply a clustering algorithm on these light feature vectors and pick out centroids, i.e. $N_{compressed}$ boxes, for deep feature extraction. This way we save a lot of time computing deep features while keeping the diversity of a tracklet. We then use average Euclidean distance between two deep feature sets of two tracklets as tracklet - tracklet appearance distance:

$$d_A = \frac{1}{N_{compressed}} \cdot \frac{1}{M_{compressed}} \sum_{n1}^{N_{compressed}} \sum_{m1}^{M_{compressed}} Euclid\big(f(n), f(m)\big) \quad (4)$$

In that,

$M_{compressed}$ is the number of filtered boxes of the first track for deep feature extraction.

$N_{compressed}$ is the number of filtered boxes of the second track for deep feature extraction.

d_A is our tracklet – tracklet appearance distance, calculated as the average Euclidean distance between two deep feature sets of two tracklets.

$f(n)$is the feature extracted from the n-th box of $N_{compressed}$ boxes.

$f(m)$ is the feature extracted from the m-th box of $M_{compressed}$ boxes.

Fusing results

A weighted sum of appearance and motion affinities is the affinity between two tracklets (used as the weight of the edge between two nodes). We fuse two affinities by taking the addition rather than multiplication as used in the detection-tracking stage because motion affinity is not reliable enough in case of long-term occlusion or camera shake. Thus, we set the weight for motion affinity low so that it plays as extra information.

$$d_{AM}(i,j) = \lambda . d_M(i,j) + (1 - \lambda) . d_A(i,j) \quad (5)$$

Where

$d_M(i,j)$ is the motion dissimilarity distance, calculated as explained.

$d_A(i,j)$is the appearance dissimilarity distance, calculated as explained.

λ is the weight parameter to adjust the importance of each distance. This value is determined through experiments (we search from 0 to 1 with 0.1 interval and choose 0.3 to maximize area under the curve for success plot).

$d_{AM}(i,j)$ is the dissimilarity distance of tracklet i and j.

Algorithm Tracklet-Tracklet Association

We present the algorithm as pseudo-code, Figure 9 presents preprocess, Figure 11 presents utility functions and Figure 10 presents tracklet-tracklet association.

Preprocess before the tracklet-tracklet association

Function Preprocess before the Tracklet-Tracklet Association

Input:

C: a list of continuous tracklets

H: a list of head-interrupted tracklets

T: a list of tail-interrupted tracklets

Output:

U: a list of unknown-id tracklets

```
for th in H do
    H.remove(th)
    if its tail-counterparted has an id:
        assign that id for th
    else:
        tracklet = merge(th, tail(th))
        H.append(tracklet)
    end if
end //for th

for tt in T do
    if is_face_test(tt) and num_detections(tt) < 5:
        T.remove(tt)
        reserve tt for next mini-batch
    end if
end //for tt

U = H + T + C
for u in U do
    if not is_face_test(u):
        U.remove(u)
    endif
end //for u
```

Figure 9. Preprocess before the tracklet-tracklet association

After each batch processing (the detection-tracking stage), we have a list of unknown-id tracklets that are needed to be assigned identifications. There are two types of tracklet after the batch processing: interrupted tracklets (ranging across batch intervals) and continuous tracklets (ranging within a batch interval). Among interrupted tracklets, we have head-interrupted and tail-interrupted ones. For example, assume our batch size is 64 and there is a tracklet starting from frame 55-th to frame 77-th. This batch approach will divide the original tracklet into two tracklets: the first part (tail-interrupted) starts from frame 55-th to 64-th and the second part (head-interrupted) starts from frame 65-th to 75-th. For each head-interrupted tracklet, if its respective tail-interrupted part (have the same track_id) has been assigned an identification, we merge it to the tail-interrupted part and treat the new tracklet as a known-id tracklet, otherwise, we merge both into one tracklet and treat it as an unknown-id tracklet waiting to be assigned in this stage. For each tail-interrupted tracklet, if it passes the *is_face_test* (at least half of the number of boxes are from detection boxes) and the number of detection boxes is smaller than 5, we delay the identification assignment for this tracklet and wait for the information from next batch (wait for its head-interrupted part).

Before passing unknown-id tracklets to the assignment process, we filter out false positive tracklets by the *is_face_test* function.

Algorithm 2: Tracklet-tracklet association

Function Tracklet-tracklet association

Input:

U: a list of unknown-id tracklets

A: a list of known-id tracklets

Output:

A: a list of known-id tracklets (updated)

Begin
// *compress tracklet for speed efficiency in extracting features*
for tracklet **in** U **do**
    ids = compress(tracklet)
    tracklet.features = deep_feature_extractor(tracklet, ids)
end //for tracklet

// *Construct the graph*
graph = construct_graph(U, A)

// *Leiden algorithm returns a list of groups*
groups = Leiden(graph)

// *assign id for matched tracklet*
for group in groups **do**
    update_id_tracklet(group)
end //for group

```
End //Function Tracklet-tracklet association
```

Figure 10. Tracklet - Tracklet assignment

After preprocessing, we now have a list of unknown-id tracklets. We also have a list of known-id tracklets. The two lists are input for the tracklet-tracklet association algorithm.

- First, we extract deep features for unknown-id tracklets as detailed above.
- We construct a graph with nodes being tracklets and edges being respective affinity (computation process is detailed above)
- Apply Leiden algorithm on this graph, the output of this algorithm contains groups of tracklets.
- If a group contains a known-id tracklet, we set that identification for others tracklet, otherwise, we create a new identity for that group.

Following is some utility functions for calculating the weight for the graph.

Utility procedures used in Algorithm 2

```
lambda: weight between motion and visual

procedure is_face_test(tracklet):
Begin
    nd = num_detections(tracklet)
    nt = num_tracks(tracklet)
    if nd > nt:
        return True
    end if
    return False
End

procedure compress(tracklet):
Begin
    light_features = tracklet.light_features
    clusters = Agglomerative_clustering(light_features, distance=L2)
    ids_choosen = []
    for cluster in clusters do
```

```
        ids_choosen.append(max_detection_score(cluster))
    end //for cluster
    return ids_choosen
End

procedure motion_affinity(t1, t2):
Begin
    et1 = forward_extrapolate(t1)
    et2 = backward_extrapolate(t2)
    distances = []
    get_ratio_and_spatial_distance(et1, et2)
    return mean(distances)
End

procedure visual_affinity(t1, t2):
Begin
    distances = []
    for f1 in t1.features do
        for f2 in t2.features do
            distances.append(Euclidean_distance(f1, f2))
        end //for f2
    end //for f1
    return mean(distances)
End
procedure affinity(t1, t2):
Begin
    motion = motion_affinity(t1, t2)
    visual = visual_affinity(t1, t2)
    return visual * lambda + (1 - lambda) * motion
End
```

Figure 11. Utility functions used in algorithm 2

Contributions

This proposed approach tackles challenges related to online approach above:

- Instead of computing deep features for all faces of one tracklet as online approaches do, we leverage light features (LBPHs) in the context of tracklet to efficiently compute deep features (extracted by deep network) without compromising representative power. In fact, the compressing method produces a more accurate representation for a tracklet thanks to diversity and high detection quality (high-score detected boxes).
- Using this framework, we can tighten the constraints in the tracking-by-detection stage so that the possibility of wrongly matching is low. Though having many tracklets after the tracking-by-detection stage, these tracklets will be grouped in the tracklet-tracklet association stage.
- We do not have to assign identifications to new detections right away in the detection-tracking stage but leave it to the tracklet-tracklet association stage. This way we can filter out false positives efficiently in the pre-processing step.
- The identification assignment step is tracklet-based; thus, we can take advantage of temporal information of tracklets (co-extant tracklets belong to different individuals)
- We also propose the trajectory difference metric to account for motion in tracklet-tracklet comparison.

In application, we often have limited data so using a pre-trained model and finetuning on our data is a reasonable choice. In this work, we show that simply adopting deep features (extracted by Facenet) and employ Euclidean (or cosine) metric is not discriminative enough in reference to real-life data. Therefore, we propose to apply Logistic discriminant metric learning so that the new embedding space for real-life data is more discriminative.

We speculate that other regions of person, besides the face, also contain discriminating features. We tried to employ some color-based feature (color name) and texture-based feature (LOMO) but the results were not comparable, thus leaving this part for future work.

Results and Discussion

Our experiments are conducted by python on the hardware GTX 1080 GPU, Intel(R) Xeon(R) CPU E5-2620 v4 @ 2.10GHz, 16GB RAM, while the MobiFace paper used a desktop machine with Intel i9-7900X CPU (3.30GHz) and one GTX 1080 Ti GPU. Therefore, it’s fair to compare the speed of our method versus other methods on MobiFace. For OTB, RFTD used a setup with Intel Core i7 with 3.07GHz clock with no GPU and CXT and SCM used similar computational power, so we only compare the performance of our method versus other methods in terms of accuracy.

The purpose of experiments on MobiFace and OTB datasets

In order to prove the efficiency of our tracking framework, we conducted two comparisons:

Comparing single trackers with tracking-by-detection approaches through results from MobiFace Dataset. The purpose is to prove that integrate the detection method will enhance the result more than using a single tracker.

Comparing tracking-by-detection approaches with our approach through results from OTB Dataset. The purpose is to prove that using the light feature to process in the tracking-by-detection stage and using the deep feature in the tracklet - tracklet association stage in conjunction with motion affinity is a significant improvement.

Experiments on MobiFace dataset

The MobiFace dataset

MobiFace dataset [44], as mentioned from the original paper, is the first dataset for single face tracking in mobile situations. Due to the lack of engrossing face tracking datasets before MobiFace, the performance of pioneer face trackers was reported on a few videos or on small subsets of the OTB dataset, and the comparison between approaches was limited. The introduced dataset provides a unified benchmark with different attributes for future development in this field. Some samples of the dataset are illustrated in Figure 12.

The authors collected 80 unedited live-streaming mobile videos captured by 70 different smartphone users in fully unconstrained environments and manually labeled over 95.000 bounding boxes on all frames. In order to cover typical usage of mobile device camera, the authors fetched videos from YouTube mobile live-streaming channels. Most of the videos are captured and uploaded under fully unconstrained environments without any extra video editing or visual effects. 6021 videos were collected and discarded under strict criteria that the target faces should appear at least in 10% of the video frames, and the target faces should not always stay still to serve the purpose of visual tracking. Besides the common 8 attributes in object tracking datasets, the authors proposed 6 additional attributes commonly seen in mobile situations.

The authors also fine-tuned and improved a handful of state-of-the-art trackers and perform evaluations on the dataset. Through comparing with those results, we can evaluate the efficiency of our method.

Setup the experiments

Note that MobiFace dataset is designed for supervised trackers - an initial box of a targeted face is specified in the first frame. However, our method is designed to work in an unsupervised way (we do not need initial boxes) and can track multiple targets at a time. In order to adapt to the dataset, we must reduce the system to fit with the protocol of the dataset. Specifically, in the first frame of each video, we compare the detected result of our system with the initial box provided by the dataset to specify the targeted face and then return track results of that target only.

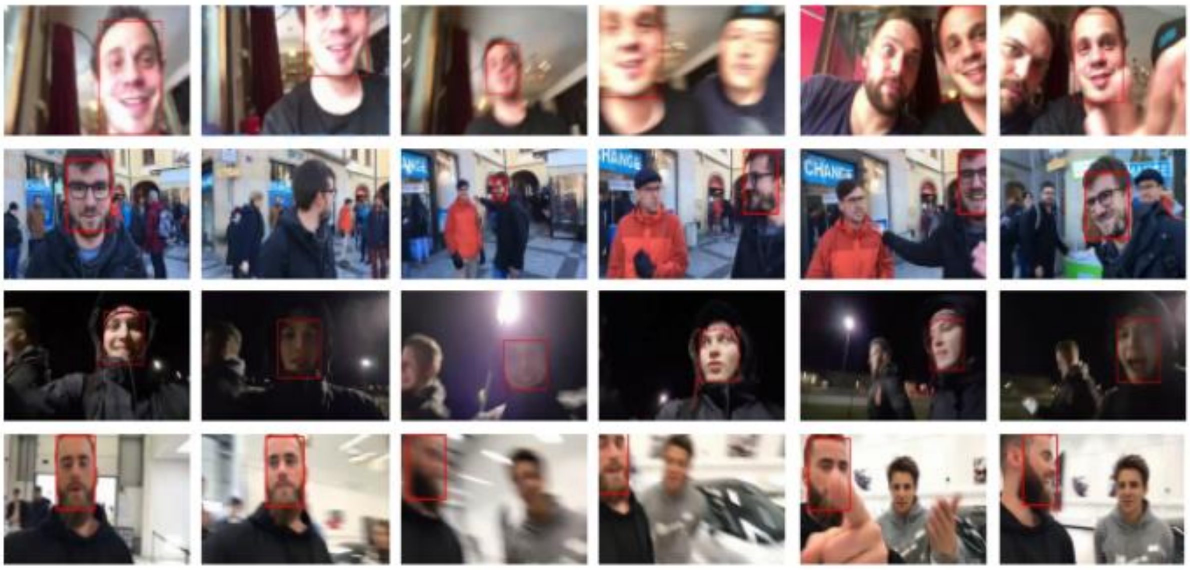


Figure 12. Some example frame from the MobiFace dataset [44]. Red ground truth bounding boxes are annotated by the authors.

The video is only stored in YouTube so from the time we access it, we are unable to collect all videos from the dataset because some has been deleted by the owners.

We consider the three metrics proposed in the dataset. As most of the metrics are in plot form, we will explain the way to extract an important metric from the plot, the area under the curve (AUC). With N is the number of thresholds used to draw the plot, we have $n = 1, 2, 3, \dots, N$. The curve was drawn from points with coordinate (t_n, f_n), t_n is the threshold value at that point and f_n is the evaluated value of our algorithm at that threshold, i.e. location error of precision plot, overlap score of success plot. The AUC is then calculated by

$$AUC = \sum_{n} (t_n - t_{n-1})\, f_n \tag{6}$$

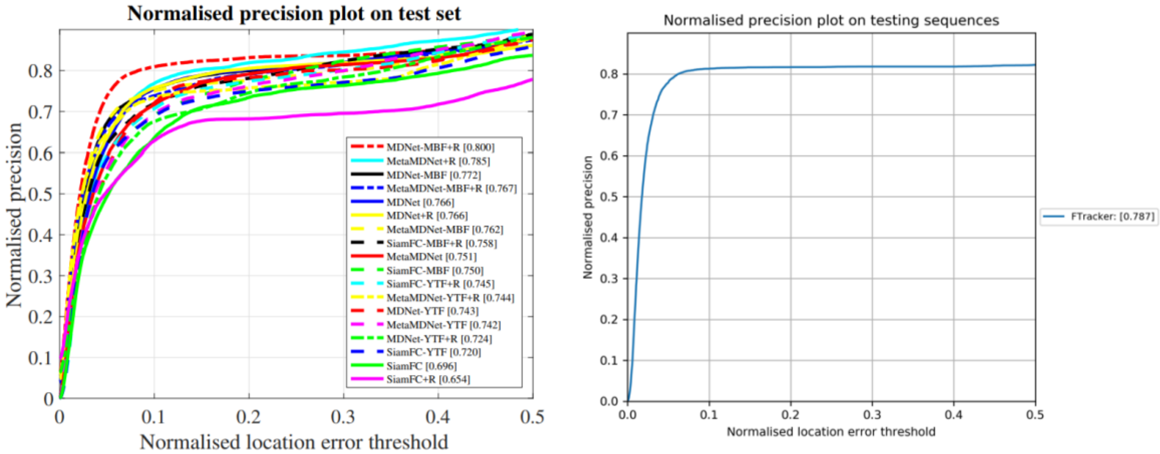
Normalised precision plot: Precision plot is a widely used evaluation metric for the tracking field. The precision is described as the location error, which is the Euclidean distance between the center location of the tracked face and the ground truth bounding box. This metric reflects how far the tracker has drifted from the targeted face. However, as the videos differ greatly in resolution, the authors adopt the recently proposed normalised precision value. The size of the frame is used for the normalisation, and the authors rank the trackers based on the area under the curve (AUC) for normalised precision value between 0 and 0.5.

Success plot: Overlap score is also another commonly used metric in the tracking field. Given a ground truth bounding box r_{gt} of the target, the predicted bounding box of our algorithm is r_p. Then we can compute the overlap score by the intersection over union (IoU) of those two boxes as S = $\frac{r_{gt} \cap r_p}{r_{gt} \cup r_p}$, where the $\cap$ and $\cup$ represent the intersection and union of two rectangles, respectively. The success plot reflects the percentage of frames in which the intersection over union (IoU) of the predicted and ground truth bounding box is greater than a given threshold. Usually, the average success rate at 0.5 threshold is enough for evaluation. In addition, the area under the curve (AUC), which is the accumulated success rate can also be used for measurement. We can use those metrics interchangeably to summarize the performance.

FPS: the average speed of the evaluated tracker running across all the sequences. The initialization time is not considered. Because of the applicability concern, a mobile face tracker must be able to run at high speed (either on CPU or GPU) to allow maximum potential migration to actual mobile devices. Due to the lack of implementation of competitive trackers on mobile platforms, we can only use the FPS measured on the desktop environment, which indicate the relative efficiency of the trackers for evaluating and comparing.

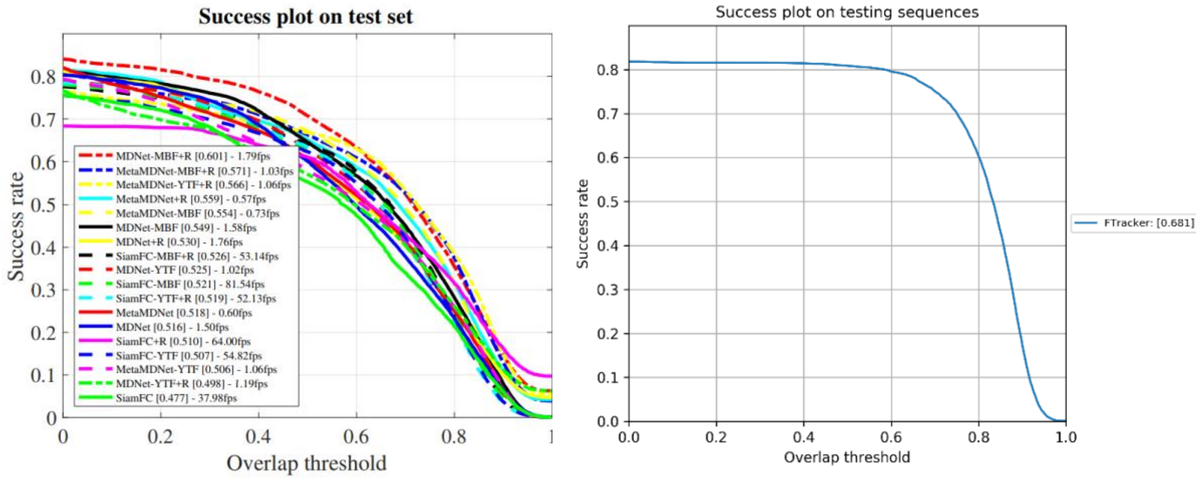
Experiment results

Evaluation metrics of our method and state-of-the-art methods are illustrated in Figure 13, Figure 14 and a detailed comparison is showed in Table 1:



(**a**) (**b**)

Figure 13. Evaluation results of trackers on MobiFace test set: **(a)** results from MobiFace paper [44], **(b)** results on our method



(**b**)

(**a**)

Figure 14. Evaluation results of trackers on MobiFace test set: **(a)** results from MobiFace paper [44], **(b)** results on our method

Table 1. A detailed comparison between our method and MobiFace evaluated results

Tracker	**Normalised Precision plot (AUC)**	**Success plot (AUC)**	**FPS**
MDNet-MBF+R	**0.800**	**0.601**	1.79
MetaMDNet-MBF+R	0.767	**0.571**	1.03
MetaMDNet-YTF+R	0.744	0.566	1.06
MDNet-MBF	**0.772**	0.549	1.58
SiamFC-MBF+R	0.758	0.526	**53.14**
SiamFC-MBF	0.750	0.521	**81.54**
Our framework	**0.787**	**0.681**	**44.38**[1]

Discussion

Because our approach is targeted for the multi-face tracking field. In order to make it work with the dataset, we run the framework over the dataset and get all tracks of targets in the video, then according to the initialized ground truth box, we define the target and return the target track results only. Because the dataset is from unconstrained environments with many existing faces, it is a noticeable effort of our tracker to avoid mistakes between tracklets and output the correct results.

As shown in the above plot, our method has an advantage in the success plot, but not the precision plot. The precision plot affected by the Euclidean distance of centers of ground truth bounding boxes and our tracking boxes. When the predicted box is drifted from the face, we terminate the tracklet instantly; therefore, with high normalised error, our tracker performs the same as with low normalised error while other trackers yield noticeably different results with different normalised errors.

The success plot may have more practical usages because the IoU decide how we can make use of the information we extracted. The success plots of trackers evaluated in MobiFace dataset are started from very high, but the slope is very steep. Starting from above 0.8 success rate for threshold 0, to threshold 0.5, they drop to below 0.7 success rate. The steep slope indicates predicted boxes of those trackers are not always aligned with ground truth boxes. Our starting point is somewhere below 0.8 success rate but maintains the success rate over the overlap threshold change. At threshold 0.5, our approach still has a high success rate, above 0.7, indicating our boxes is closely aligned with ground truth boxes. At 0.5 threshold, the predicted boxes cover most of the track target and can be well used in application.

[1] We profile the program and exclude reading image from disk time and writing image to disk time before calculating speed (details are in test.profile file in our source code).

Besides, as the main target of ours is for practical usages, a good success plot and success rate at 0.5 threshold - while keeping the speed - are acceptable.

Experiments on OTB (Object Tracking Benchmark) Dataset

About the dataset

OTB Dataset [45] is one of the most famous datasets specifically used for benchmarking the object trackers since its appearance. The authors worked to collect and annotate most of the common tracking sequences from different datasets. They also classified those sequences into multiple categories by challenges as in Table 2 and selected 50 difficult and representative ones in the TB-50 dataset for an in-depth analysis. The full dataset contains more sequences of human (36 body and 26 face/head videos) than other categories because human target objects have the most practical usages, some samples of the dataset is illustrated in Figure 15.

Table 2. Annotated Sequence Attributes with the Threshold Values in the Performance Evaluation from OTB Dataset [45]

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

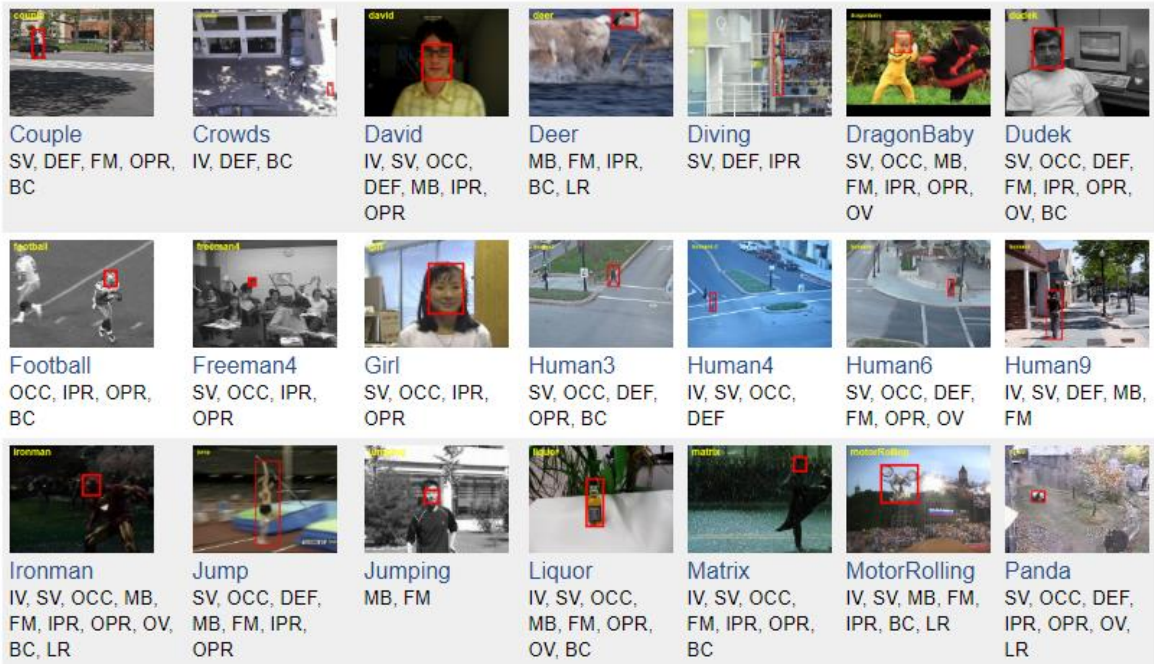


Figure 15. Some example sequences from the OTB Dataset [45]

Before the introduction of MobiFace dataset, face tracking methods can only be evaluated on small self-collected datasets or a subset of OTB dataset. The whole dataset is designed for the object tracking algorithms, but we selectively pick out the sequences with faces to conduct experiments and compare with those methods mentioned before. The chosen face subset is described in Table 3, the top 10 sequences are referred to as the difficult set and top 15 is the normal set [46]:

Table 3. Chosen sequences and their attributes

#	Sequence	Challenge
1	**Soccer**	IV, SV, OCC, MB, FM, IPR, OPR, BC
2	**Freeman4**	SV, OCC, IPR, OPR
3	**Freeman1**	SV, IPR, OPR
4	**FleetFace**	SV, DEF, MB, FM, IPR, OPR
5	**Freeman3**	SV, IPR, OPR
6	**Girl**	SV, OCC, IPR, OPR
7	**Jumping**	MB, FM
8	**Trellis**	IV, SV, IPR, OPR, BC
9	**David**	IV, SV, OCC, DEF, MB, IPR, OPR
10	**Boy**	SV, MB, FM, IPR, OPR
11	FaceOcc2	IV, OCC, IPR, OPR

12	Dudek	SV, OCC, DEF, FM, IPR, OPR, OV, BC
13	David2	IPR, OPR
14	Mhyang	IV, DEF, OPR, BC
15	FaceOcc1	OCC

However, the dataset is also designed for the single object tracker. So, evaluation on this dataset also cannot reflect all the potential power of our system, but we can use that result to relatively compare with previous trackers in order to verify the power of our framework.

Set up the experiments

Because the authors of MobiFace dataset inherit a lot of legacy from OTB dataset, in general, the setup stage and evaluation stage for OTB Dataset are the same as the MobiFace dataset.

Experimental results

Evaluation metrics of our method and state-of-the-art methods are illustrated in Figure 16, Figure 17, and a detailed comparison is showed in Table 4 and Table 5.

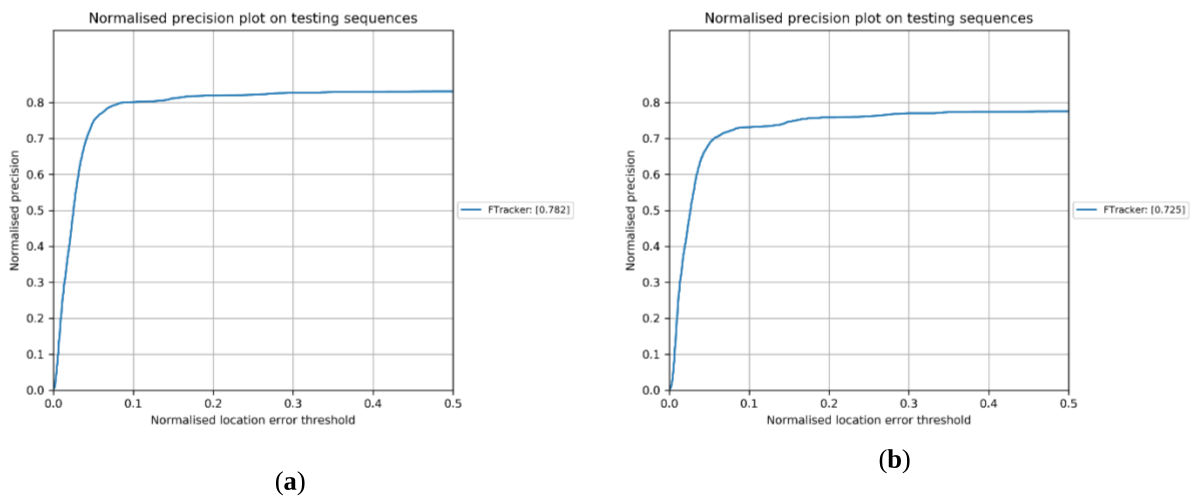


Figure 16. Our normalised precision plot on OTB Dataset face subsets **(a)** normal set**(b)** difficult set

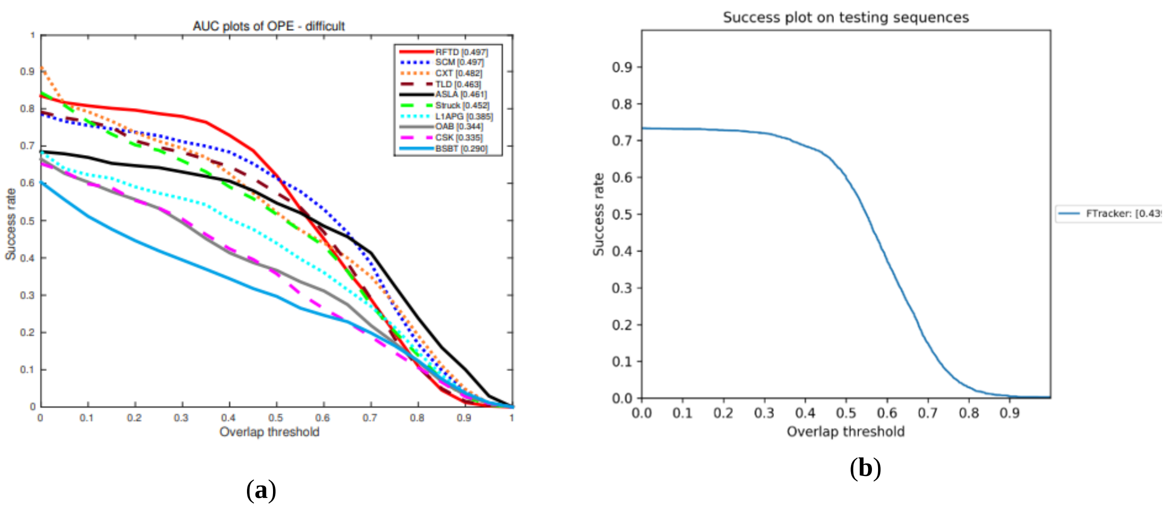


Figure 17. Success plots of trackers on OTB Dataset face subset (difficult set): **(a)** results from RFTD paper[46] **(b)** results on our method

Table 4. Top tracker comparison on OTB Dataset face subset (normal set). Evaluated results are from RFTD paper [46].

Face Tracker	**Success Plot AUC**	**Success plot Threshold (0.5)**
RFTD	55.2	**71.3**
Struck	**55.9**	67.6
SCM	**58.3**	**72.6**
ASLA	53.8	62.9
CSK	48.0	56.8
L1APG	50.7	59.7
OAB	42.6	48.9
TLD	51.8	67.3
CXT	**57.3**	65.7
BSBT	40.6	47.0
Our framework	51.9	**68.3**

Table 5. Top tracker comparison on OTB Dataset face subsets (difficult set). Evaluated results are from RFTD paper [46].

Face Tracker	**Success Plot AUC**	**Success plot Threshold (0.5)**
RFTD	**49.7**	**62.0**
Struck	45.2	51.7
SCM	**49.7**	**61.3**
ASLA	46.1	54.7
CSK	33.5	52.2
L1APG	38.5	43.9
OAB	34.4	36.6
TLD	46.3	57.4
CXT	**48.2**	52.2
BSBT	29.0	29.7
Our framework	43.9	**59.7**

Discussion

The precision plots in Figure 16 are good. The overall results are quite good, and the slope is shallow as predicted after witnessing above experiments. However, we have no data from other works to have an in-depth comparison.

As first sight from the metric Table 4 and Table 5, our framework has average AUC while the slope of our framework is also shallow as predicted. The main reason here is because when the predicted box is drifted from the face, we terminate the tracklet instantly; therefore, with high normalised error, our tracker performs the same as with low normalised error while other trackers yield noticeably different results with different normalised errors. The initial modest success rate leads to a modest average value. The success rate at threshold 0.5 is still good, ranking third in that section in both subsets.

Conclusions

In this work, we proposed a method for face tracking problem in semi-online manner - the online process with some minor delay. The comparing experiments are conducted on two datasets: MobiFace dataset and OTB dataset with many state-of-the-arts works in the field. The results show that our method can produce robust accuracy while keeping a good speed. With that, the effectiveness of adding the tracklet-tracklet association stage after detection stage in semi-online manner is proven. The manipulation of appearance affinity and motion affinity have brought us the accuracy of the framework, while the workload division and information sharing of the two main stages make our process lighter and achieve better speed. With the improvements, all the disadvantages pointed out in section 1 are solved.

The demonstrated framework has many advantages that can be manipulated to the production environment. First, the process of a whole was cut off to achieve a value suitable for continuous streaming with a little delay. Second, the accuracy maintains at an acceptable value, which makes our framework robust in many unconstraint environments. Finally, the framework can work without supervision, and is a high-performance multi-face tracking system.

Future works following this work can dig in many ways for the better. First, try other combinations of related techniques (detector, tracker, feature extractor) to achieve better results. Second, exploit the concept of semi-online manner (use some delay for better results) is also a good point. Third, the work only targets the continuous video stream, but in other fields, many other attributes can be used to improve the results. Finally, a public multi-face dataset will be a major contribution to this field.

Data Availability

The OTB and MobiFace dataset supporting this study are from previously reported studies and datasets, which have been cited. The processed data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The author(s) declare(s) that there is no conflict of interest regarding the publication of this paper.

Funding Statement

This research is funded by Viet Nam National University Ho Chi Minh City (VNUHCM) under grant no. B2018-18-01.

Acknowledgments

This research is funded by Viet Nam National University Ho Chi Minh City (VNUHCM) under grant no. B2018-18-01.

Thank to Axon company for the valuable support on cooperation.

Supplementary Materials

References

[1] J. Berclaz, F. Fleuret, E. Türetken, and P. Fua, “Multiple Object Tracking Using K-Shortest Paths Optimization,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, pp. 1806–1819, 2011.

[2] A. Milan, S. Roth, and K. Schindler, “Continuous Energy Minimization for Multitarget Tracking,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 1, pp. 58–72, Jan. 2014.

[3] C. Kim, F. Li, A. Ciptadi, and J. M. Rehg, “Multiple Hypothesis Tracking Revisited,” *2015 IEEE Int. Conf. Comput. Vis. ICCV*, pp. 4696–4704, 2015.

[4] S. Tang, B. Andres, M. Andriluka, and B. Schiele, “Multi-Person Tracking by Multicut and Deep Matching,” *ArXiv E-Prints*, p. arXiv:1608.05404, Aug. 2016.

[5] L. Leal-Taixé, C. Canton Ferrer, and K. Schindler, “Learning by tracking: Siamese CNN for robust target association,” *ArXiv E-Prints*, p. arXiv:1604.07866, Apr. 2016.

[6] C. Cruz, L. Sucar, and E. Morales, “Real-Time face recognition for human-robot interaction,” in *Proceedings of the 8th IEEE International Conference on Automatic Face and Gesture Recognition*, 2008, pp. 1–6.

[7] A. V. Segal and I. D. Reid, “Latent Data Association: Bayesian Model Selection for Multi-target Tracking,” *2013 IEEE Int. Conf. Comput. Vis.*, pp. 2904–2911, 2013.

[8] A. Sadeghian, A. Alahi, and S. Savarese, “Tracking The Untrackable: Learning To Track Multiple Cues with Long-Term Dependencies,” *ArXiv E-Prints*, p. arXiv:1701.01909, Jan. 2017.

[9] Q. Chu, W. Ouyang, H. Li, X. Wang, B. Liu, and N. Yu, “Online Multi-Object Tracking Using CNN-based Single Object Tracker with Spatial-Temporal Attention Mechanism,” *ArXiv E-Prints*, p. arXiv:1708.02843, Aug. 2017.

[10] L. Chen, H. Ai, Z. Zhuang, and C. Shang, “Real-time Multiple People Tracking with Deeply Learned Candidate Selection and Person Re-Identification,” *ArXiv E-Prints*, p. arXiv:1809.04427, Sep. 2018.

[11] C. Kim, F. Li, and J. M. Rehg, “Multi-object Tracking with Neural Gating Using Bilinear LSTM,” in *ECCV*, 2018.

[12] M. Thoreau and N. Kottege, “Improving Online Multiple Object tracking with Deep Metric Learning,” *ArXiv E-Prints*, p. arXiv:1806.07592, Jun. 2018.

[13] Y. Yoon, A. Boragule, Y. Song, K. Yoon, and M. Jeon, “Online Multi-Object Tracking with Historical Appearance Matching and Scene Adaptive Detection Filtering,” *ArXiv E-Prints*, p. arXiv:1805.10916, May 2018.

[14] N. Narayan, N. Sankaran, S. Setlur, and V. Govindaraju, “Re-identification for Online Person Tracking by Modeling Space-Time Continuum,” *2018 IEEECVF Conf. Comput. Vis. Pattern Recognit. Workshop CVPRW*, pp. 1519–151909, 2018.

[15] S. Zhang *et al.*, “Improved Selective Refinement Network for Face Detection,” *ArXiv E-Prints*, p. arXiv:1901.06651, Jan. 2019.

[16] R. Kalman, “A New Approach to Linear Filtering and Prediction Problems,” *Trans. ASME - J. Basic Eng.*, vol. 82, pp. 35–45, 1960.

[17] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista, “High-Speed Tracking with Kernelized Correlation Filters,” *ArXiv E-Prints*, p. arXiv:1404.7584, Apr. 2014.

[18] D. Held, S. Thrun, and S. Savarese, “Learning to Track at 100 FPS with Deep Regression Networks,” in *Unknown*, 2016, vol. 9905, pp. 749–765.

[19] L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, and P. H. S. Torr, “Fully-Convolutional Siamese Networks for Object Tracking,” *ArXiv E-Prints*, p. arXiv:1606.09549, Jun. 2016.

[20] F. Schroff, D. Kalenichenko, and J. Philbin, “FaceNet: A unified embedding for face recognition and clustering,” in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 815–823.

[21] Z. Zhong, L. Zheng, Z. Zheng, S. Li, and Y. Yang, “Camera Style Adaptation for Person Re-identification,” *ArXiv E-Prints*, p. arXiv:1711.10295, Nov. 2017.

[22] J. Zhuo, Z. Chen, J. Lai, and G. Wang, “Occluded Person Re-identification,” *ArXiv E-Prints*, p. arXiv:1804.02792, Apr. 2018.

[23] Y. Suh, J. Wang, S. Tang, T. Mei, and K. M. Lee, “Part-Aligned Bilinear Representations for Person Re-identification,” *ArXiv E-Prints*, p. arXiv:1804.07094, Apr. 2018.

[24] M. M. Kalayeh, E. Basaran, M. Gokmen, M. E. Kamasak, and M. Shah, “Human Semantic Parsing for Person Re-identification,” *ArXiv E-Prints*, p. arXiv:1804.00216, Mar. 2018.

[25] J. Almazán, B. Gajic, N. Murray, and D. Larlus, “Re-ID done right: towards good practices for person re-identification.,” *CoRR*, vol. abs/1801.05339, 2018.

[26] H. Wang *et al.*, “CosFace: Large Margin Cosine Loss for Deep Face Recognition,” *ArXiv E-Prints*, p. arXiv:1801.09414, Jan. 2018.

[27] M. Keuper, E. Levinkov, N. Bonneel, G. Lavoué, T. Brox, and B. Andres, “Efficient Decomposition of Image and Mesh Graphs by Lifted Multicuts,” *ArXiv E-Prints*, p. arXiv:1505.06973, May 2015.

[28] V. Traag, L. Waltman, and N. J. van Eck, “From Louvain to Leiden: guaranteeing well-connected communities,” *ArXiv E-Prints*, p. arXiv:1810.08473, Oct. 2018.

[29] H. W. Kuhn, “The Hungarian Method for the Assignment Problem,” in *50 Years of Integer Programming*, 2010.

[30] S. Zhang *et al.*, “Tracking Persons-of-Interest via Adaptive Discriminative Features,” in *Computer Vision – ECCV 2016*, Cham, 2016, pp. 415–433.

[31] S. Jin, H. Su, C. Stauffer, and E. Learned-Miller, “End-to-End Face Detection and Cast Grouping in Movies Using Erdös-Rényi Clustering,” in *arXiv e-prints*, 2017, pp. 5286–5295.

[32] C. Lin and Y. Hung, “A Prior-Less Method for Multi-face Tracking in Unconstrained Videos,” in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 538–547.

[33] F. Comaschi, S. Stuijk, T. Basten, and H. Corporaal, “Online multi-face detection and tracking using detector confidence and structured SVMs,” in *2015 12th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, 2015, pp. 1–6.

[34] P. Viola and M. J. Jones, “Robust Real-Time Face Detection,” *Int. J. Comput. Vis.*, vol. 57, no. 2, pp. 137–154, May 2004.

[35] M. Naiel, M. O. Ahmad, M. N. s Swamy, J. Lim, and M.-H. Yang, “Online Multi-Object Tracking via Robust Collaborative Model and Sample Selection,” *Comput. Vis. Image Underst.*, vol. 154, 2016.

[36] X. Lan, Z. Xiong, W. Zhang, S. Li, H. Chang, and W. Zeng, “A super-fast online face tracking system for video surveillance,” in *2016 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2016, pp. 1998–2001.

[37] A. Ranftl, F. Alonso-Fernandez, S. Karlsson, and J. Bigun, “Real-time AdaBoost cascade face tracker based on likelihood map and optical flow,” *IET Biom.*, vol. 6, no. 6, pp. 468–477, 2017.

[38] J. Chen, R. Ranjan, A. Kumar, C. Chen, V. M. Patel, and R. Chellappa, “An End-to-End System for Unconstrained Face Verification with Deep Convolutional Neural Networks,” in *2015 IEEE International Conference on Computer Vision Workshop (ICCVW)*, 2015, pp. 360–368.

[39] M. Hayat, S. H. Khan, N. Werghi, and R. Goecke, “Joint Registration and Representation Learning for Unconstrained Face Identification,” in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 1551–1560.

[40] N. Crosswhite, J. Byrne, C. Stauffer, O. Parkhi, Q. Cao, and A. Zisserman, “Template Adaptation for Face Verification and Identification,” in *2017 12th IEEE International Conference on Automatic Face Gesture Recognition (FG 2017)*, 2017, pp. 1–8.

[41] R. Ranjan *et al.*, “A Fast and Accurate System for Face Detection, Identification, and Verification,” *IEEE Trans. Biom. Behav. Identity Sci.*, vol. 1, pp. 82–96, 2018.

[42] Y. Wang, J. Shen, S. Petridis, and M. Pantic, “A real-time and unsupervised face Re-Identification system for Human-Robot Interaction,” *Pattern Recognit. Lett.*, 2018.

[43] Jianbo Shi and Tomasi, “Good features to track,” in *1994 Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 1994, pp. 593–600.

[44] Y. Lin, S. Cheng, J. Shen, and M. Pantic, “MobiFace: A Novel Dataset for Mobile Face Tracking in the Wild,” *ArXiv E-Prints*, p. arXiv:1805.09749, May 2018.

[45] Y. Wu, J. Lim, and M.-H. Yang, “Object Tracking Benchmark,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, pp. 1–1, 2015.

[46] F. Comaschi, S. Stuijk, T. Basten, and H. Corporaal, “Robust online face tracking-by-detection,” in *2016 IEEE International Conference on Multimedia and Expo (ICME)*, 2016, pp. 1–6.