Face Retrieval in Large-Scale News Video Datasets

Thành Đức Ngo†††, Hung Thanh Vu††, Nonmembers, Duy-Dinh Le†††, and Shin’ichi Satoh†††, Members

SUMMARY Face retrieval in news video has been identified as a challenging task due to the huge variations in the visual appearance of the human face. Although several approaches have been proposed to deal with this problem, their extremely high computational cost limits their scalability to large-scale video datasets that may contain millions of faces of hundreds of characters. In this paper, we introduce approaches for face retrieval that are scalable to such datasets while maintaining competitive performances with state-of-the-art approaches. To utilize the variability of face appearances in video, we use a set of face images called face-track to represent the appearance of a character in a video shot. Our first proposal is an approach for extracting face-tracks. We use a point tracker to explore the connections between detected faces belonging to the same character and then group them into one face-track. We present techniques to make the approach robust against common problems caused by flash lights, partial occlusions, and scattered appearances of characters in news videos. In the second proposal, we introduce an efficient approach to match face-tracks for retrieval. Instead of using all the faces in the face-tracks to compute their similarity, our approach obtains a representative face for each face-track. The representative face is computed from faces that are sampled from the original face-track. As a result, we significantly reduce the computational cost of face-track matching while taking into account the variability of faces in face-tracks to achieve high matching accuracy. Experiments are conducted on two face-track datasets extracted from real-world news videos, of such scales that have never been considered in the literature. One dataset contains 1,497 face-tracks of 41 characters extracted from 370 hours of TRECVID videos. The other dataset provides 5,567 face-tracks of 111 characters observed from a television news program (NHK News 7) over 11 years. We make both datasets publicly accessible by the research community. The experimental results show that our proposed approaches achieved a remarkable balance between accuracy and efficiency.

key words: face-track extraction, face-track matching, large-scale, news video

1. Introduction

News videos play an important role as a source of information nowadays because of their rich and relevant contents. With the advances in modern technology, a huge number of news videos can be obtained easily. Accordingly, this creates an urgent demand to retrieve useful information from such news video datasets. Because most news is related to people, human face retrieval, which is defined as the task of extracting and returning faces relevant to a given query, ob-
Face-track extraction. Face-track extraction is a key step in video-based face retrieval systems. Existing studies on automatic face-track extraction follow a standard paradigm that consists of two basic steps, detecting faces in frames and grouping faces of the same character into face-tracks. In the first step, the Viola-Jones detector is usually used to detect near frontal faces in frames of videos. In the second step, the detected faces of the same character are grouped by using either clustering [13] or tracking approaches [10],[11],[23]. In [13], Ramanan et al. built a color histogram for the hair, face, and torso associated with each detected face in a frame. A concatenated vector of the normalized color histogram represented the face. They then clustered all vectors to obtain groups of similar faces, using agglomerative clustering. The limitations of this approach include its high computational cost for constructing and clustering high-dimensional representation feature vectors and its dependence on determining a reasonable threshold for the clustering algorithm to ensure that no group contains faces of multiple characters and the groups are not over-fragmented.

On the other hand, Everingham et al. in [10] and Sivic et al. [11] proposed the use of tracking approaches to associate the detected faces of the same character. In [11], an affine covariance region tracker of [14] is used. This tracker can develop tracks on deforming objects, where the between-frame region deformation can be modelled by an affine geometric transformation plus perturbations. The outcome is that a face can be tracked (by the collection of regions on it) through significant variations in poses and changes in expressions, allowing distant detected faces to be associated. One of the main drawbacks of this approach is its high computational cost for locating and tracking affine covariant regions. In contrast, Everingham et al. in [10] used a more efficient tracker, which is Kanade-Lucas-Tomasi (KLT) tracker [30], to create a set of track points starting at some frames in a shot and continuing until some later frames. Grouping faces in different frames for one character is based on enumerating the track points shared between faces. However, because the KLT tracker is sensitive to illumination changes and partial occlusions, additional techniques are required to obtain accurate face-track extraction results. Another way of using a tracker for face-track extraction was recently introduced by Merler et al. in [23]. Instead of using a tracker to find the face, and torso associated with each detected face in a frame.
of using tracking results to connect detection results, they combine both of these to estimate the optimal positions of faces. Their online multiple instance learning tracker is expensive and the linear combination is sensitive to parameter changes.

**Face-track matching.** There are two major categories of approaches to using multiple-exemplars of faces in face-tracks (i.e. sets of face images) for robust face matching and recognition. The approaches in the first category [15]–[18] make use of both face images and the temporal order of their appearances. The face dynamics within the video sequence are modeled and exploited to improve recognition accuracy. For instance, Li et al. [19], [20] introduced an approach to modeling facial dynamics by constructing facial identity structures across views and overtime in the Kernel Discriminant Analysis feature space. Edwards et al. [21] proposed learning the mutation of individual faces through video sequences by decoupling sources of image variations, such as poses, facial expressions and illumination. They then used the trained statistical face model to incorporate identity evidence over a sequence. In [17], Liu and Chen used an adaptive Hidden Markov Model (HMM) for this face recognition problem. In the training phase, they created a HMM for each character to learn the statistics and temporal dynamics using the eigen-face image sequence. The implicit constraint of these approaches is that the dynamics of faces should be temporally consecutive. In general, this constraint is not always satisfied.

Without relying on temporal coherence between consecutive images, the approaches in the second category use multiple face images only and treat the problem as a set-matching problem. These approaches are differentiated based on the ways in which the sets are modeled and the similarity between sets is computed. Shakhnarovich et al. [4] modeled a face sequence using a probability distribution. However, to make the computation tractable, they made the assumption that faces are normally distributed, which may not be true [22]. Cevikalp and Triggs [8] claimed that a face sequence is a set of points and they discovered a convex geometric region expanded by these points. The min-min approach [10], [11], [23] considered a face sequence as a cluster of points and measured the distance between these clusters. Subspace approaches [5]–[7] viewed a face sequence as points spread over a subspace. Although these methods can be highly accurate, a lot of computation is needed to represent the distribution of the face sequence, such as computing the convex hulls in [8], the probability models in [5], and the eigenvectors in [5]–[7]. For this reason, they are not scalable to large-scale video datasets.

**Face datasets.** In evaluating the performance of face matching approaches, most of the previous works on face retrieval in video use two benchmark datasets: MoBo (Motion of Body) [24] and Honda/UCSD [25]. The scales of these datasets are limited, varying from hundreds to thousands of face images of tens of individual characters. Particularly, Honda/UCSD consists of 75 videos involving 20 individual characters. Each video contains approximately 300-500 frames. Meanwhile, Mobo provides 96 image sets of 24 individual characters. Hence, there are only 4 image sets for each character. One of the largest face datasets recently available is the YouTube Faces dataset [26], which provides 3,425 videos of 1,595 individual characters. However, each character has only around 2.15 videos. Such a small number of samples for each character is insufficient to stably evaluate a face matching or recognition approach, which is an important part of a face retrieval system. In addition, there are no face datasets related to real-world news videos, which is our targeted domain. In view of all the above mention considerations, we prepare new datasets for evaluating the approaches.
3. Framework Overview

Figure 2 illustrates the overview of our framework. In the off-line stage, the face-tracks in all video shots are extracted using our face-track extraction approach (described in Sect. 4). Each extracted face-track contains multiple face images of one individual character, varied under different viewpoints, illumination conditions, and expressions within a shot. Each single face image in a face-track is represented by a feature vector. The process consisting of face-track extraction and face image representation is performed once for the entire video dataset. Our main contribution here is making the face-track extraction approach robust against flash lights, scattered appearances of characters, and occlusions.

Given a face-track as an input retrieval query, the online stage of our system starts by using our proposed face-track matching algorithm (described in Sect. 5) to estimate the similarity between a query face-track and each face-track in the retrieved set containing all face-tracks extracted from the dataset in the offline stage. A ranked list of the evaluated face-tracks is returned as the retrieval result of the online stage. Because the retrieved set is huge, our approach targets an extremely efficient face-track matching strategy while maintaining a competitive performance with state-of-the-art approaches.

4. Face-Track Extraction

A common strategy in the existing approaches for face-track extraction consists of detecting faces in frames and grouping the detected faces of the same character. While detecting faces is done by using a standard face detector (e.g. Viola-Jones face detector) [10], [11], [13], grouping detected faces requires comprehensive techniques to identify faces of the same character.

4.1 Face-Track Extraction Based on Tracking Points

To group detected faces into face-tracks, connections should be established between faces belonging to the same character in different frames. A point tracker can be used for this purpose.

Assuming some points are generated and tracked through frames of a shot, we have the output of the tracking process as a set of tracking trajectories. One trajectory is for one generated point. We call such trajectories point tracks. Given two faces A and B in different frames and the set of point tracks, there are four types of point tracks regarding their intersection with the faces: (a) point tracks that pass through both A and B, (b) point tracks that pass through A but not B, (c) point tracks that pass through B but not A, and finally, (d) point tracks that do not pass through either A or B. A point track passes through a face if its point lies within the face bounding box in the corresponding frame.

A confidence grouping measure (CGM) that the two faces A and B belong to the same character can then be defined as:

\[
CGM(A, B) = \frac{N_a}{N_b + N_c}
\]

where \(N_a\), \(N_b\), and \(N_c\) are the number of tracks of types (a), (b), and (c). If \(CGM(A, B)\) is larger or equal to a certain threshold, the two faces, A and B, are grouped into one face-track. Figure 3 presents an overview of a face-track extraction approach based on tracking points.

4.2 Removal of Frames Containing Flash Lights

Although grouping faces based on a point tracker is efficient, applying the point tracker to news videos results in poor accuracy due to the occurrences of flash lights. The reason is that point trackers usually rely on intensity information to compute the image motion to find the correspondence between points in different frames. When flash lights occur in a frame, they significantly change the intensity of the frame. Thus, the tracker cannot track points properly (there is an example in Fig. 4). To handle such situations that happen frequently in news videos, frames containing flash lights should be removed. We call such frames flash-frames.

To identify flash-frames, we measure the luminosity of the frames in the video shot. If the luminosity of a frame is significantly increased compared with its neighbors, the
frame is declared to be a flash-frame. Particularly, given a set of consecutive frames $S$:

$$S = \{fr_s : s = t, t + W\}$$

(2)

where $t$ is a frame index and $W$ is the potential length of a flash light (i.e. the number of consecutive frames affected by a flash light). The frames in $S$ are determined to be flash-frames if $\forall fr_s \in S$, we have:

$$\begin{align*}
L(fr_s) &> \gamma L(fr_{s-1}) \\
L(fr_s) &> \gamma L(fr_{s-W+1})
\end{align*}$$

(3)

given $L(fr_s)$ is the computed luminosity of frame $fr_s$, and $\gamma$ is a predefined luminosity sensitive threshold. In our experiments, we found that $\gamma = 1.25$ and $W = \{1, 2, 3\}$ are optimal to detect all flash-frames with a low false alarm rate.

By removing flash-frames, faces in these frames are also eliminated in grouping into face-tracks. However, such faces can not enrich information on their corresponding face-tracks but only add noise since their visual identity characteristics are often lost due to overlighting. And, two over-lightened faces of two characters may look very similar each other. Hence, eliminating them brings benefit to face-track matching.

4.3 Point Generation and Tracking

There are two main processing steps of a point tracker, which are point generation and point tracking. Once points are generated, they can be tracked through a sequence of frames. With a state-of-the-art point tracker such as the KLT tracker, points are generated by using an approach introduced by Shi and Tomasi [29]. The approach selects optimal points for tracking without any constraints on the positions of points. Then, points are tracked by computing optical flow between frames.

The requirement to use tracking results (i.e. point tracks) for grouping faces of the same character is that the faces must have some point tracks passing through all of them. There are two cases where the requirement is not met:

(i) Faces of new characters are detected in frames in which points are not generated or they are generated but not inside the faces (there is an illustration in Fig. 5). To bypass this shortcoming, we generate track points for faces that are considered to be the faces of new characters. Faces of new characters in a certain frame are faces that cannot be grouped into any existing face-tracks.

In particular, we process a given shot frame-by-frame. Each face in the current frame of processing is checked against all existing face-tracks formed in the previous frames to find which face-track the face belongs to. Checking between a face and each face-track is based on computing the confidence grouping measure ($CGM$, presented in Sect.4.1) between the face and the last face of the face-track. The face will then be grouped into a face-track whose $CGM$ is largest and larger or equal to a certain threshold (0.5 in our experiments). A face having no $CGM$ larger than the threshold is set as the initial face of a new face-track (i.e. a new face for a new character). Track points are then only generated inside the face bounding boxes.

(ii) Points are incorrectly tracked due to occlusions. Track points move from inside the face to outside it after occlusion. Furthermore, track points from background regions move inside the face. Thus, the number of point tracks passing through a face before occlusion and other faces after occlusion significantly declines, resulting in failed face grouping. Figure 6 shows an example of such problems.

We handle this problem by detecting and replacing incorrectly tracked points as well as not generating track points in background regions. When a face in the current frame is grouped into an existing face-track, we investigate all track points passing through the face or the last face of the face-track. All points whose tracks only pass through one of the two faces are removed. Because such points are likely to have been tracked inaccurately, removing them prevents us from transferring tracking errors to later frames. Then, additional points are generated to replace those that have been re-
moved and to provide points updated with a new visual appearance of the face. We apply the approach of Shi and Tomasi [29] to the face bounding box to generate additional points. Points whose tracks pass through both the faces are kept.

4.4 Our Proposed Approach for Face-Track Extraction

Given a video shot with all flash-frames removed, our approach starts by finding the first frame in which faces are detected. All point tracking and face grouping processes are initialized from this frame. This helps us to save computational cost and avoid tracking errors caused by transitional effects between shots. Initial track points will be generated inside all detected faces in the frame. Each face now becomes the first face of a corresponding newly formed face-track.

After initialization, we sequentially process each frame afterwards until the end of the shot is reached. The pseudo-code is presented in Algorithm 1. Figure 7 has a step-by-step illustration of our approach.

**Algorithm 1. Our approach for face-track extraction**

**Require:** a video shot with identified flash-frames

**Require:** faces detected in frames

1. Move to the first frame $f_{r1}$ in which faces are detected.
2. Initialization
   - Create a new face-track with each detected face in $f_{r1}$.
   - Generate track points inside the detected faces.
3. for each frame $f_{ri}$ from $f_{r1}$ to the end of the shot do
   4. if $f_{ri}$ is not a flash-frame then
   5. Use the KLT tracker to track existing points and update their positions in $f_{ri}$.
   6. for each detected face $f_i$ in $f_{ri}$ do
      7. for each existing face-track $ft_j$ do
         8. Compute CGM between the face $f_i$ and the last face $f_{tj}$ of $ft_j$, $CGM(f_i, f_{tj})$.
      9. end for
   10. Find face-track $ft_X$ so that $CGM(f_i, f_{tX})$ ≥ $CGM(f_j, f_{tj}), \forall ft_j$ and $CGM(f_i, f_{tX})$ ≥ 0.5.
   11. if $ft_X$ found then
      12. Group $f_i$ to $ft_X$.
      13. Remove incorrectly tracked points and generate additional points for $f_i$.
   14. else
      15. Create a new face-track with $f_i$.
      16. Generate new points for $f_i$.
   17. end if
   18. end for
   19. end if
20. end for

Note that our approach does not compare all possible pairs of faces for face grouping. Such pairwise comparison rapidly becomes intractable as the number of faces in a shot increases. Instead, we group faces into face-tracks according to the temporal order of their appearance. A detected face in a frame is only compared to the last faces of existing face-tracks. By doing this, we avoid greedy pair-wise comparisons.

5. Matching Face-Tracks

Several approaches for matching face-tracks have been proposed (as presented in Sect. 2). However, although these approaches have shown high accuracy in benchmark datasets, their high computational costs limit their practical applications in large-scale datasets. This motivates us to target a matching approach that provides a good balance between accuracy and computational cost. The approach should be extremely efficient while achieving a competitive performance with state-of-the-art approaches.

To maintain competitive accuracy, we still use the plentiful information from the multiple faces of a face-track to enrich the representation. However, instead of using all the faces in a face-track, we propose taking a subsample of the faces. In doing so, the required computational cost can be reduced while keeping the amount of information sufficient to improve accuracy. We call our approach $k$-Faces.

Based on an observation that faces in neighboring frames of a character are not dramatically changed, we propose to subsample faces of a face-track regarding their temporal order of appearance. The neighboring range is controlled by a variable, $k$. With a given value of $k$, our approach starts by temporally dividing the face-track into $k$ equal parts. The middle face for each part is selected to represent all faces within the part since we assume that faces within such a part are barely similar each other when $k$ is sufficiently large.

Given $k$ faces subsampled from the face-track and each face has been extracted its own facial feature, the face-track now is corresponding to a set of $k$ points distributed in the feature space. We employ the mean point to represent the
Fig. 8  An illustration of our proposed \( k \)-Faces approach. In (a), each face-track is first divided into \( k \) equal parts (\( k = 4 \), in this example). The middle face (with bright-colored outline) is selected in each part to represent the part. Then, \( k \) selected faces (marked with stars) are used to compute a mean face (circle or triangle) on the feature space. The mean face now represents the whole face-track. Finally, in (b), the similarity of face-tracks is estimated based on the distance between their mean faces.

set. The distance between two sets now relies on the distance between their mean points. In other words, if the mean point is called a mean face, the similarity between two face-tracks corresponds to the distance between their mean faces. Figure 8 illustrates our \( k \)-Faces approach.

We have two main advantages by using a single mean face of \( k \) subsampled faces to represent a face-track. First, the computational cost on estimating similarities between face-tracks is low, since only one mean face is used for one face-track. Second, if there are noisy faces in a face-track and they are not a majority, subsampling faces helps us to reduce the number of noisy faces actually involved in processing. Our approach is therefore less sensitive to noisy faces than other approaches such as those employing all faces of the face-track to compute the mean face or those estimating the similarity between two face-tracks based on the pair-wise distances of their faces.

Let \( m^A = \{m^A_1, m^A_2,...,m^A_N\} \) and \( m^B = \{m^B_1, m^B_2,...,m^B_N\} \) denote the mean faces of face-tracks \( A \) and \( B \), respectively, and \( N \) represents the number of dimensions of feature space. We use the following standard distance types to compute the distance between \( m^A \) and \( m^B \).

**Cosine:**
\[
\text{distance} = 1 - \frac{\sum_{i=1}^{N} m^A_i \times m^B_i}{\sqrt{\sum_{i=1}^{N} (m^A_i)^2} \times \sqrt{\sum_{i=1}^{N} (m^B_i)^2}}
\]  
(4)

**Euclidean:**
\[
\text{distance} = \sqrt{\sum_{i=1}^{N} (m^A_i - m^B_i)^2}
\]  
(5)

**L1:**
\[
\text{distance} = \sum_{i=1}^{N} |m^A_i - m^B_i|
\]  
(6)

The pseudo-code for our \( k \)-Faces is presented below.

**Algorithm 2. The proposed \( k \)-Faces approach**

**Require:** Two face-tracks \( A, B \) and a predefined value of \( k \)

1: for each face-track \( A, B \) do
2: Divide the face-track into \( k \) equal parts according to the temporal order.
3: In each divided part, select the middle face.
4: Compute the mean face (on the feature space) of the \( k \) selected faces.
5: end for
6: Compute the distance between the mean faces of \( A \) and \( B \).
7: return the computed distance.

Clearly, the higher the value of the \( k \) set, the more faces in each face-track selected to compute the mean face and the better the approximations, which may result in improved accuracies. However, note that the computational cost can greatly increase. By using \( k \) as a predefined parameter, \( k \)-Faces provides users with flexibility in balancing the accuracy that they expect and the cost that they can afford (or the time they can spend waiting for the result). The question of
selecting reasonable values for \( k \) on a dataset is presented in Sect. 6.2.4.

6. Experiments

In this section, we present our experiments to evaluate the proposed approaches. The experiments are divided into two parts: the first evaluates the performance of the proposed approach in face-track extraction, and the second in face-track matching.

6.1 Evaluation of Face-Track Extraction

We tested our proposed approach for face-track extraction on 8 video sequences from different video broadcasting stations including NHK News 7, ABC News, and CNN News. All shot boundaries are provided in advance. A face detector based on the Viola-Jones approach [12] is used to detect near frontal faces in every frame of the video sequences. A conservative threshold is used to reduce the number of false positives (i.e. a non-face classified as a face). In particular, we only keep detected faces which are larger than 60 × 60, and the number of neighbor rectangles that makes up a candidate face must be greater than 4.

Ground-truth annotation on the face-tracks in the videos is manually prepared. Each face-track of a character appearing in a video shot is annotated by indicating all detected faces of the character. An extracted face-track is regarded as correct if it contains all faces of the face-track compared to ground-truth annotation. If the face-track has more or less faces than in the annotation, it is said incorrectly extracted. Note that if a character moves out of the frame and then moves back into it again, annotators will divide the appearance of that character into two independent face-tracks in ground-truth annotation. Table 1 summarizes the number of frames, faces, and face tracks.

We directly compare our approach with the state-of-the-art approach proposed by Everingham et al. [10] in this experiment. Their approach generates track points in the first frame of the shot and tracks them throughout the shot based on local appearance matching. Points that cannot be tracked from one frame to the next are eliminated and replaced with new points. Their face grouping criteria is similar to that presented in Sect. 4.1. The threshold of CGM for grouping two faces is also 0.5.

As shown in Table 2, by detecting flash-frames, our approach successfully overcomes the problem of face-track fragmentation due to flash lights. Meanwhile, the approach by Everingham et al. almost completely fails to do that. In addition, the results also show that our approach is superior to that of Everingham et al. in handling problems caused by partial occlusion and the appearance of a character in the middle of a shot. The only face-tracks that we could not extract exactly are those fully occluded in some frames during their occurrences. In those cases, all points in the face regions are drifted to the background region. After such full occlusions, there is no clue to re-grouping the face of that character. Using only a tracker is not enough to handle this problem. One can apply visual information-based clustering to group the fragmented face-track, as in [13], but this obviously requires extra cost. Nevertheless, we observe that full occlusion rarely happens in news video because the characters featured in the news are recorded with care, especially the important and well-known ones. This is a special characteristic of news videos. The last column of the table shows the overall extraction performance of both approaches. These facts clearly indicate that our approach is robust and outperforms that of Everingham et al. in [10].

In terms of speed (see Table 3), our approach is approximately 2 times slower than that of Everingham et al. However, our complexity is somehow linear to the total number of faces because we consequently enlarge face-tracks according to the temporal order by checking new faces with only the last appeared face of each face-track. Meanwhile, Everingham et al. compared all pairs of faces in the shot. Their complexity is polynomial to the total number of faces. If the number of faces increases, the gap in speed between our approach that by Everingham et al. will narrow rapidly.

In this experiment, we show that our proposed techniques and solutions to the problems are robust and efficient enough for extracting face-tracks in real-world news videos by successfully extracting 94% of all face-tracks. Based on our observations, other complex techniques can be applied to handle the problems. However, the trade-off between obtaining the 6% remaining face-tracks and incurring an overly high computational cost should be considered with care.
6.2 Evaluation of Face-Track Matching

6.2.1 Datasets

Due to the limitations of existing public datasets, we prepared new datasets for the experiments. Face-tracks are extracted from videos of the datasets by using our proposed approach to face-track extraction (see Sect. 4.2). The identity of the character associated with each extracted face-track is given by annotators. Because our approach extracts face-tracks in each video shot, we used a robust shot boundary detector to obtain shot boundaries for videos. The whole process, including shot boundary detection and face-track extraction, is fully automated.

**TRECVID dataset.** We used TRECVID news videos from 2004 to 2006. This dataset contains 370 hours of videos in different languages, such as English, Chinese, and Arabic. The total number of frames that we processed was approximately 35 million. Among those, 20 million faces were grouped into 157,524 face-tracks. We filtered out short face-tracks that had less than 10 faces, which resulted in 35,836 face-tracks. Finally, we annotated 1,497 face-tracks containing 405,887 faces of 41 well-known individual characters.

**NHKNews7 dataset.** This dataset consists of observations from the NHK News 7 program over 11 years. After the annotation process, 1,259,320 faces of 111 individuals are provided. The total number of face-tracks is 5,567. Each character has from 4 to 550 face-tracks. In this dataset, we discard face-tracks with fewer than 100 faces and more than 500 faces. Compared to the TRECVID dataset, the NHKNews7 dataset is much more challenging.

Table 4 shows a comparison between our datasets and some public benchmark datasets. Based on the results, it is obvious that our datasets are superior over the other datasets, such as MoBo and Honda/UCSD, on all statistical terms, including number of videos, number of characters, and average face-track length. Buffy dataset [32], a yet another popular dataset, is also smaller than ours. Although they have more face-tracks for each character, their face-tracks are rather small. The number of face-tracks having less than 10 faces is 374, approximately 47% of the dataset. Compared to the YouTube Faces dataset, we provide much more face-tracks (or video shots) per character. Thus, our datasets are more relevant for stably evaluating a face retrieval system.

Figure 9 presents statistical information on our datasets. The datasets can be downloaded from http://satoh-lab.ex.nii.ac.jp/users/ndthanh/NIIFacetrackDatasets. However, due to copyright issues, the face images in face-tracks cannot be published. Instead, we provide a feature vector, used in [10], for each face image. The feature vector of a face is extracted by computing the descriptors of the local appearance of the face around each of the located facial features. Before extracting the descriptors, the face is geometrically normalized to reduce the effect of pose variation. An affine transformation is estimated, which transforms the located

![Fig. 9](image-url) Statistical information on our datasets. (a) shows the distribution of face-tracks over their lengths; (b) and (c) present the number of face-tracks for the top 20 individual characters in each dataset.

| Datasets    | #Face-tracks (Ftk.) | #Characters (Chr.) | #Faces per Ftk. | #Ftk. per Chr. |
|-------------|---------------------|-------------------|----------------|---------------|
| MoBo        | 96                  | 24                | 300            | 4.00          |
| Honda/UCSD  | 75                  | 20                | 300-500        | 3.75          |
| Buffy [32]  | 802                 | 11                | 1-392          | 72.9          |
| YoutubeFaces| 3,425               | 1,595             | 48-6,075       | 2.15          |
| TRECVID     | 1,497               | 41                | 10-3,781       | 36.51         |
| NHKNews7    | 5,567               | 111               | 100-500        | 50.15         |
facial feature points to a canonical set of feature positions. Then, the appearance descriptors around each facial feature are computed. The final feature representation of the face is formed by concatenating all the descriptors of its facial features. Regarding to our experiments in [31] and this paper with the TRECVID dataset, this feature is better than local binary pattern (LBP) feature for face-track matching.

6.2.2 Evaluated Approaches

We compared k-Faces with several approaches, including those based on pair-wise distances, MSM [6], and CMSM [7].

Given two face-tracks having multiple face images represented as feature vectors, pair-wise based approaches compute the distances between each possible pair of feature vectors in two face-tracks. The maximum distance, the minimum distance, or the mean distance of the computed pair-wise distances is then used as the similarity measurement between two face-tracks. We refer to the approaches as pair.max, pair.min, and pair.mean, respectively (see Fig. 10 for the illustration). The pair.min (sometimes called min-min) is one of the state-of-the-art approaches widely used in other studies [10], [23], [26], [27].

In addition, we also evaluate an yet another approach that is a hybrid of pair.min and our proposed approach. The approach starts by dividing each face-track into $k$ equal parts according to the temporal order and selecting the middle face of each part. Given $k$ selected faces for each face-track, the pair.min approach using only these selected faces is then applied to estimate the similarity between two face-tracks. We call this approach $k$-pair.min. With $k$ is not larger than the number of faces in the face-track, by using only $k$ faces in each face-track, this approach is more efficient than the original pair.min. When $k$ is getting larger, the performance of this approach definitely approximates the performance of pair.min. However, because $k$-pair.min still requires $k^2$ comparisons between the selected faces of two face-tracks, it is theoretically $k^2$ more expensive than our k-Faces approach. And, $k$-pair.min is also more sensitive to noisy faces than our k-Faces due to the fact that it relies on pair-wise distances between faces of the face-track.

Regarding [28], if the pair-wise based approaches are representative of non-parametric sample based approaches, MSM and CMSM are representative of approaches based on a parametric model. MSM, introduced by Yamaguchi et al. [6], represents an image set by a linear subspace spanned by the principal components of the images. The similarity between the sets is computed using the angle between subspaces. CMSM is an extension of MSM, in which subspaces of the sets are projected onto a constraint subspace. In doing so, the subspaces are expected to be more separable. All of these approaches have been shown their robustness in benchmark datasets, such as MoBo [24], HondaUCSD [25], and YouTube Faces [26]. Therefore, it is appealing to compare our k-Faces with them for a comprehensive evaluation.

Besides evaluating k-Faces with different values of $k$ and different types of distance (e.g. Euclidean, L1, and cosine), we try another criterion for selecting $k$ representative faces in a face-track. Instead of temporally dividing the face-track and choosing the middle face of each part, another criterion that is based on clustering can be applied in selecting these representative faces. In this new way, all the faces in a face-track will be clustered to $k$ groups by using a clustering algorithm. The centroid of each group is selected. Then, the mean of $k$ centroids is used as the representative face for the face-track. In this experiment, we use the standard K-Means for clustering. We refer to the former k-Faces as k-Faces.Temporal and to the latter k-Faces as k-Faces.KMeans.

We evaluate the performance of a face-track matching approach by computing the average precision of the rank list that it returned. In particular, in each dataset, a face-track is alternatively picked out as a query face-track while the remaining face-tracks are used as the retrieved database. Given a query, the average precision of the returned ranked list is computed. Finally, the mean of all average precision (MAP) values for all queries is reported as the overall evaluation metric for the approach with the given database.

Let $r$ denote a rank in the returned face-track list, $Pre(r)$ the precision at rank $r$ of the list, $N_l$ the length of the list, $N_{hit}$ the total number of face-tracks matched with the query face-track $q$, and $IsMatched(k)$ a binary function returning 1 if the face-track at rank $r$ is matched with $q$ (based on ground-truth annotations) and, zero otherwise. Then, the MAP of the evaluated approach can be computed as follows:

\[
AP(q) = \frac{N_l}{\sum_{r=1}^{N_l} (Pre(r) \times IsMatched(r))}
\]

(7)

\[
MAP = \frac{\sum_{q} AP(q)}{\text{number of queries}}
\]

(8)

MAP is a standard metric for evaluating retrieval and match-
Fig. 11  MAP(s) of the approaches in the TRECVID (left) and NHKNews7 (right) datasets.

ing systems. Besides the MAP, we record the processing times of the approaches in each dataset to compare their efficiency.

6.2.3 Results

Figure 11 presents the mean average precision (MAP) of the evaluated approaches in our two datasets, TRECVID and NHKNews7. With MSM and CMSM, their best performances over various sets of parameters are selected.

In general, all the MAPs vary from 64.61% to 76.82% in the TRECVID dataset. Meanwhile, in the NHKNews7 dataset, the best MAP is 60.99%, and the worst is 40.89%.

The difference in the MAPs between the two datasets can be explained by following reasons. First, the number of characters in NHKNews7 is larger than that in TRECVID. 111 characters in NHKNews7 compared to 41 characters in TRECVID. This clearly increases the probability of mismatching face-tracks. Second, the videos in NHKNews7 were recorded over a long time (i.e. 11 years). Thus, besides facial variations in each face-track caused by the environmental conditions at the time of recording (e.g., illumination, pose, viewpoint), the face-tracks of the character themselves also reflect the biological variations of the character over time: for instance, a character may look older after several years (see Fig. 12 for example). For these reasons, matching
Table 5  Mean Average Precision and processing times (in seconds) of the evaluated approaches.

Note that the preprocessing process is only performed once for a given dataset.

| Approaches | TRECVID | NHKNews7 |
|------------|---------|----------|
|            | MAP(%)  | Preprocessing Time | Matching Time | MAP(%)  | Preprocessing Time | Matching Time |
| pair.min + L1 | 76.54  | 0.00 | 2544.73 | 60.99  | 0.00 | 6678.00 |
| k = pair.min + L1 (k=20) | 76.50  | 0.00 | 418.24 | 57.11  | 0.00 | 972.36 |
| k-Faces.Temporal + L1 (k=20) | 73.65  | 26.54 | 1.63 | 53.68  | 41.4 | 3.23 |
| k-Faces.KMeans + L1 (k=20) | 73.07  | 6380.30 | 1.20 | 55.43  | 14949.72 | 3.45 |
| MSM       | 69.20  | 4454.10 | 347.39 | 38.92  | 4896.72 | 667.15 |
| CMSM      | 64.02  | 4991.02 | 95.36 | 53.08  | 5841.80 | 155.40 |

![Image](image.png)

Fig. 12  Face-tracks of President George W. Bush recorded in 2001 (top) and 2009 (bottom).

faces in NHKNews7 becomes more challenging, which resulted in decreased MAP(s) for all the evaluated approaches.

A clear and consistent observation from both datasets is that pair.min (i.e. min-min) is always among the two best approaches. It achieved 76.54% MAP and 60.99% MAP in the two datasets, respectively. Among the distance types, L1 is the optimal for use with pair.min. A reasonable replacement is the Euclidean distance. However, there is still a minor accuracy gap between pair.min using L1 and pair.min using Euclidean distance. In addition, computing the Euclidean distance between two feature vectors is more expensive than computing their L1 distance.

The results also show that pair.min is better than pair.mean. This is because pair.mean uses the mean of all pair-wise distances between two face-tracks as the similarity score. By computing the mean, pair.mean reduces the effect of noisy pairs. At the same time, it eliminates the influence of pairs containing identical faces, which can help to instantly determine that the faces are belong to the same character. Thus, the discriminative power of the the computed similarity score is reduced, compared that computed by pair.min. This causes the difference in MAPs between pair.min and pair.mean. More generally, this explains why such a gap between pair.min and pair.mean is larger in NHKNews7 than in TRECVID. Because the average length of face-tracks on NHKNews7 is longer (i.e. each face-track contains more faces of a character), there is a greater chance that two face-tracks of the same character contain identical faces. Among the approaches based on pair-wise distances, pair.max achieved the worst performance. Its best MAPs with L1 distance in the TRECVID and the NHKNews7 datasets are 28.16% and 14.99%, respectively. We do not include pair.max in the Fig. 11.

Regarding our k-Faces, its MAP increases when k increases. Between k-Faces.Temporal and k-Faces.KMeans, the impact of k on the MAP of k-Faces.KMeans is less significant. Because k-Faces.KMeans always uses all the faces in a face-track for clustering and selecting centroids for representative faces, the final mean face is less sensitive to k. In contrast, k plays an important role in k-Faces.Temporal. The higher the k set, the more representative faces of each face-track selected. Thus, the final mean face of each face-track becomes more reliable and accurate. The advantages of k-Faces.KMeans is that it can achieve high accuracy even when k is very small. However, its disadvantage is the high computational cost of clustering faces on a high-dimensional feature space (i.e. 1,937 dimensions). When k is large enough, there is no substantial difference in MAP between k-Faces.KMeans and k-Faces.Temporal.

In both datasets, when k increases from 2 to 20, the MAPs of k-Faces approaches grow rapidly. However, the MAPs become stable from k = 20 upward. Because further increasing k does not help improve accuracy but increases the computational cost, we select k = 20 for investigating the trade-off between the accuracy and computational cost of k-Faces approaches in comparison to others.

As we can see from Fig. 11, the hybrid approach k-pair.min provides more competitive performance with the pair.min approach than our proposed k-Faces. The performance of k-pair.min with most values of k is between those of pair.min and k-Faces. When k is sufficiently large (20 in TRECVID, and 150 in NHKNews7), k-pair.min approximates pair.min. In some cases, k-pair.min is even better than pair.min, e.g., when k is larger than 50 in the TRECVID dataset. However, the gap is very small varying from 0.2% to 0.25%. This is because there are noisy faces (e.g. blurred faces) in a few face-tracks, and k-pair.min is less sensitive to noise than pair.min due to its subsampling process. Although k-pair.min is more accurate than our k-Faces, we need to be reminded that it theoretically requires \(k^2\) times more comparisons than ours. Their practical efficiency is summarized in Table 5.

Table 5 shows MAP and processing time of each approach. Processing time is divided into two parts, preprocessing and matching. The preprocessing time refers to the time required to preprocess face-tracks in a given dataset before matching. In k-Faces approaches, the preprocessing of face-tracks includes selecting representative faces and computing their mean face. In MSM and CMSM, preprocessing includes computing subspaces for face-tracks. The matching time is averaged over one query run. The time unit used...
6.2.4 Discussion

As shown in Table 5, k-Faces.KMeans and k-Faces.Temporal achieve almost equal accuracy and consume the same amount of time for one query in both datasets. However, k-Faces.Temporal is hundreds of times (240 times in TRECVID and 360 times in NHKNews7) faster than k-Faces.KMeans in the preprocessing phase. This suggests that in terms of both accuracy and efficiency, selecting representative faces based on temporal sampling is better than that based on clustering.

Compared to state-of-the-art approaches, our k-Faces.Temporal is thousands of times faster than pair.min, and hundreds of times faster than MSM, CMSM and \( k - \text{pair.min} \) in both datasets. In terms of accuracy, k-Faces takes third place, with 73.65\% in the TRECVID dataset, after pair.min and k-pair.min. The difference in MAP between our approach and pair.min is only 2.89\%. Meanwhile, k-Faces.Temporal is significantly better than MSM and CMSM, which respectively achieved 69.20\% and 64.62\% accuracy. In NHKNews7 dataset, k-Faces.Temporal is better than CMSM, but worse than pair.min, k-pair.min, and MSM. One may question why MSM performed poorly in the TRECVID dataset, but was superior to k-Faces.Temporal in the NHKNews7. The reason for this is the fact that the face-tracks in the NHKNews7 dataset are larger than those in the TRECVID dataset. Therefore, more sample faces in each face-track can be used to obtain a reliable subspace.

The results obtained from this experiment generally indicate that our proposed approach is extremely efficient while achieving performance comparable with that of state-of-the-art approaches.

6.2.4 Discussion

This subsection discusses two main arguments on using k-Faces approaches:

- Why the k-Faces.Temporal has comparable accuracy to k-Faces.Kmeans and how to select a reasonable value for \( k \) given a dataset.
- How accurate the performance of k-Faces approaches (k-Faces.Temporal and k-Faces.KMeans) are compared to other approaches on benchmark and public video datasets.

The key idea behind both k-Faces.Temporal and k-Faces.Kmeans to achieve reasonable accuracy while maintaining efficient retrieval speed is to only use a subset of faces among all available faces in each face-track. To do that, k-Faces.Temporal is based on the assumption that the faces of a character appearing in neighboring frames are visually similar. Thus, one of them can be used as being representative. Meanwhile, k-Faces.KMeans relies on clustering to find clusters of similar faces directly in the feature space. Faces within a cluster are considered to be neighboring in the space. Hence, the center of each cluster can be used to represent the cluster.

In both approaches, the variable \( k \) controls the range of neighboring. The larger the value of \( k \) that is selected, the smaller the range is. In other words, the number of faces in a part temporally divided by k-Faces.Temporal and the number of faces in a cluster in k-Faces.KMeans are smaller. This means the representative face for each part (or cluster) selected by both approaches becomes less different with other faces within the part (or cluster). In addition, more faces of a face-track are involved in computing the single mean face (i.e. the mean of \( k \) selected faces, or the mean of \( k \) centroids) as \( k \) is increasing to represent the whole face-track. As the two sets of actual faces used by both approaches gradually overlap, the means of the sets become more similar. All of these reasons explain why k-Faces.Temporal has comparable accuracy to k-Faces.KMeans, given sufficiently large \( k \).

To evaluate the representativeness of the mean faces computed by both approaches, we compute the mean distance from each of the mean faces to all other faces of a face-track. Smaller distance indicates better representativeness since the mean face is more similar to the other faces within the face-track. The mean distance is used instead of the sum of distances since the numbers of faces in face-tracks are very different. We report the average of such mean distances over all face-tracks (i.e. average representativeness) with different values of \( k \) in Fig. 13.

When \( k \) is sufficiently large (e.g. by being larger or equal to 20) in Fig. 13, the mean faces computed by both k-Faces.Temporal and k-Faces.KMeans have equal representativeness. This observation is also consistent with what we learned from previous experiments in which we evaluate the accuracy of the approaches with different values of \( k \). In general, the larger \( k \) is, the better the representativeness.
tativeness of the mean faces and the higher the accuracy.

By computing the representativeness of the mean faces with different values of $k$ in a given dataset, we obtain insights into selecting a reasonable $k$ to balance computational costs and accuracy. At a value of $k$ that keeps increasing $k$ does not help to significantly improve the representativeness of the mean faces, that value of $k$ should be selected (e.g. $k = 20$ as in our datasets). This is because if the mean faces do not change, performance also may not change.

In order to investigate the performances of $k$-Faces approaches compared to other approaches on public and well-known video datasets, we carry out other experiments using two datasets, Honda/UCSD [25] and Buffy [32]. The Buffy dataset already provides face-tracks (i.e. sets of faces belonging to the same characters). Meanwhile, with Honda/UCSD, we download its videos and apply our face-track extraction approach to obtain face-tracks. All experimental settings are kept the same as those in our previous experiments. The performance of all evaluated approaches are presented in Fig. 14.

The experimental results shown in Fig. 14 once again demonstrate that the performance of our proposed $k$-Faces.Temporal approach is comparable to that of other state-of-the-art approaches.

7. Conclusion

In this paper, we investigate face retrieval in large-scale news video datasets. Our contributions are threefold. First, we present a face-track extraction approach that incorporates techniques to handle problems due to flash lights, partial occlusions, and scattered appearances of characters in real-world news videos. Second, we present an approach for face-track matching that significantly reduces the computational cost while achieving competitive performance compared with state-of-the-art approaches. Third, we prepare datasets, evaluate state-of-the-art face retrieval approaches, and make public two real-world face-track datasets of such scales that have never been considered in the literature.

References

[1] M. Turk and A. Pentland, “Face recognition using eigenfaces,” IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 1991.
[2] A. Pentland, B. Moghaddam, and T. Starner, “View-based and modular eigenspaces for face recognition,” IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), pp.84–91, 1994.
[3] P.N. Belhumeur, J.P. Hespanha, and D.J. Kriegman, “Eigenfaces vs. fisherfaces: Recognition using class specific linear projection,” IEEE Trans. Pattern Anal. Mach. Intell., pp.711–720, 1997.
[4] G. Shakhnarovich, J.W. Fisher, and T. Darrell, “Face recognition from long-term observations,” European Conference on Computer Vision (ECCV), pp.851–868, 2002.
[5] W. Fan and D.-Y. Yeung, “Locally linear models on face appearance manifolds with application to dual subspace based classification,” IEEE Conf. Comput. Vis. Pattern Recognit., pp.1384–1390, 2006.
[6] O. Yamaguchi, K. Fukui, and K. Maeda, “Face recognition using temporal image sequence,” IEEE Int. Conf. Auto. Face and Gesture Recog., pp.318–323, 1998.
[7] K. Fukui and O. Yamaguchi, “Face recognition using multi-viewpoint patterns for robot vision,” International Symposium of Robotics Research, pp.192–201, 2003.
[8] H. Cevikalp and B. Triggs, “Face recognition based on image sets,” IEEE Conf. Comput. Vis. Pattern Recognit., pp.2567–2573, 2010.
[9] R. Wang, S. Shan, X. Chen, and W. Gao, “Manifold-manifold distance with application to face recognition based on image set,” IEEE Conf. Comput. Vis. Pattern Recognit., pp.1–8, 2008.
[10] M. Everingham, J. Sivic, and A. Zisserman, “Taking the bite out of automated naming of characters in TV video,” Image Vis. Comput., pp.545–559, 2009.
[11] J. Sivic, M. Everingham, and A. Zisserman, “Person spotting: Video shot retrieval for face sets,” International Conference on Image and Video Retrieval (CIVR), pp.226–236, 2005.
[12] P. Viola and M. Jones, “Robust real-time face detection,” Int. J. Comput. Vis., vol.57, pp.137–154, 2004.
[13] D. Ramanan, S. Barker, and S. Kakade, “Leveraging archival video for building face datasets,” Int. Conference on Computer Vision (ICCV), pp.1–8, 2007.
[14] J. Sivic, F. Schaffalitzky, and A. Zisserman, “Object level grouping for video shots,” European Conference on Computer Vision (ECCV), pp.85–98, 2004.
[15] A. Hadid and M. Pietikainen, “From still image to video-based face recognition: An experimental analysis,” IEEE Int. Conf. Auto. Face and Gesture Recog., pp.813–818, 2004.
[16] K.C. Lee, J. Ho, M.H. Yang, and D. Kriegman, “Video-based face
recognition using probabilistic appearance manifolds,” IEEE Conf. Comput. Vis. Pattern Recognit., pp.313–320, 2003.

[17] X. Liu and T. Chen, “Video-based face recognition using adaptive hidden markov models,” IEEE Conf. Comput. Vis. Pattern Recognit., pp.340–345, 2003.

[18] S. Zhou, V. Krueger, and R. Chellappa, “Probabilistic recognition of human faces from video,” Comput. Image Understand., pp.214–245, 2003.

[19] Y. Li, S. Gong, and H. Liddell, “Video-based online face recognition using identity surfaces,” IEEE Int. Conf. Auto. Face and Gesture Recog., pp.40–46, 2001.

[20] Y. Li, S. Gong, and H. Liddell, “Constructing facial identity surfaces for recognition,” Int. J. Comput. Vis., pp.71–92, 2003.

[21] G. Edwards, C. Taylor, and T. Cootes, “Improving identification performance by integrating evidence from sequences,” IEEE Conf. Comput. Vis. Pattern Recognit., pp.486–491, 1999.

[22] L. Wolf and A. Shashua, “Learning over sets using kernel principal angles,” J. Machine Learning Research, vol.4, pp.913–931, 2003.

[23] M. Merler and J.R. Kender, “Selecting the best faces to index presentation videos,” ACM International Conference on Multimedia (ACM Multimedia), pp.1461–1464, 2011.

[24] R. Gross and J. Shi, The CMU Motion of Body (MoBo) Database, Carnegie Mellon University, 2001.

[25] K.C. Lee, J. Ho, M.H. Yang, and D. Kriegman, “Visual tracking and recognition using probabilistic appearance manifolds,” Comput. Vis. Image Understand., pp.303–331, 2005.

[26] T.D. Ngo, D.-D. Le, S. Satoh, and D.A. Duong, “Robust face track finding in video using tracked points,” Signal Image Technology and Internet Based Systems, pp.59–64, 2008.

[27] M. Everingham, T.D. Ngo, D.-D. Le, S. Satoh, and D.A. Duong, “An efficient method for face retrieval from large video datasets,” Int. Conference on Image and Video Retrieval (CIVR), pp.382–389, 2010.

[28] M. Everingham, J. Sivic, and A. Zisserman, “Hello! My name is… Buffy — Automatic naming of characters in TV video,” British Machine Vision Conference, 2006.