Abstract

Face pixelation in TV shows or videos is manually realized and not well studied to date. As the prevailing of online video streaming, we develop a new tool called face pixelation in live-streaming (FPLV) to generate automatic personal privacy filtering during unconstrained streaming. FPLV is organized in a frame-to-video structure for fast and accurate face pixelation of irrelevant people. Leveraging image-based face detection and recognition networks on individual frames, we propose a positioned incremental affinity propagation (PIAP) clustering algorithm to associate faces across frames. Through deep feature and position aggregated affinities, PIAP handles the cluster number generation, new cluster discovering, and faces’ raw trajectories forming simultaneously. Affected by various factors, raw trajectories might be intermittent and unreliable. Hence, we introduce a proposal net for loosed face detection with an empirical likelihood test to compensate the deep network insufficiency and refine the raw trajectories. A Gaussian filter is laid on refined trajectories for final pixelation. FPLV obtains satisfying accuracy and real-time performances under streaming video data we collected.

1 Introduction

Live-stream platforms and streamers rapidly become part of our daily life. Benefit from the popularity of smart-phones and 4G mobile networks, online live video streaming is amazingly pervasive today. Contradict to indoor-broadcast that involves with the streamers alone at the most time; outdoor live-streaming always happens in public. During such streaming activities, many other people like passersby and pedestrians are filmed through streams’ cameras unconsciously and then broadcast to potentially everyone without permission. In such cases, streamers, live-stream platforms, and apps have raised a host of legal issues, primarily in the areas of privacy infringement. Despite different laws of privacy in different regions, respect privacy rights are universally correct. To prevent disturbing disclosure and privacy violation from live-streaming, we build a privacy security mechanism which at least filters out the irrelevant peoples’ face from live-streaming.

Referring to the practice of TV shows, people who are unwilling to appear in front of the camera will be facially pixelated to protect their privacy. However, the placement of mosaics on faces is almost a handwork even today in videos. To release the burden of manual censorship, in this paper, we develop a new tool called FPLV to automate the face pixelation process in live-streaming for personal privacy protection. Although there does not exist any standards for face pixelation, but as a rule of thumb, face pixelation should prevent personal privacy from leaking as well as avoid affecting the audiences’ experience.

Owing to the tremendous success of deep convolutional neural network (CNN), [Yu et al., 2017] already can handle the image level face pixelation through object detection and multi-task learning. Leveraging well-established face detection and recognition algorithms frame by frame, a naive face pixelation method could be built by allocating Gaussian filters to blur the identified non-steamer’s faces on individual frames. Limited by learning samples and available contextual information, the above naive face pixelation algorithm heavily violates the requirement of accuracy. But from another aspect, it provides us a way to implement FPLV through assembling frame-wise results of deep face detection and recognition networks and fixing errors according to video context. Following such a thought, our FPLV introduces two stages of raw face trajectories forming and a flexible parametric trajectory refinement stage with a Gaussian kernel empirical likelihood.

At the beginning of a live-stream, we demand a \((2 \times N)\) frames buffering section. With FPLV’s real-time efficiency on live video streaming, this buffering allows us stacking every \(N\) frames to form a short video segment and significantly improves the performance of our trajectory refinement stage without affecting the audience’s experience. Obtaining discrete results from face detection and recognition on each \(N\) frames, we build a novel positioned incremental affinity propagation (PIAP) clustering algorithm to connect the discrete faces across frames. Because of the ill-initialized cluster number, affinity propagation (AP) [Frey and Dueck, 2007] is the most suitable choice among clustering algorithms. Due to
the requirements of efficiency, we extended the AP clustering into an incremental (IAP) way to process the incoming segments dynamically. Providing pair-wise similarities for faces regarding deep feature expression and motion information, positioned IAP (PIAP) clusters the same faces into a particular group. An inner-group sequential link generates the raw trajectories. With the created raw trajectories, a proposal net is set up as a supplement. To reach a high rate of accuracy, we manage another shallow CNN to propose suspicious face areas at a high recall rate to compensate the detection lost within each segment. Combing a Gaussian kernel empirical likelihood with the raw trajectories, we deny the inappropriate bounding boxes and receive the proper ones for refinements. Overlapping Gaussian Filters realize the final face pixelation on target refined trajectories. Figure 1 shows the working pipeline of FPLV.

Specifically, this paper consists of following contributions:

- We are the first one dedicate to build face pixelation for personal privacy protection in live video streaming.
- We develop a frame-to-video framework for real-time implementation of FPLV.
- We build PIAP clustering algorithm to connect the same faces across frames and create robust raw face trajectories.
- We propose a non-parametric imposing method according to empirical likelihood based on Gaussian kernels. This error rejection algorithm can be widely applied to other face detection algorithms.
- We collect and build a video streaming dataset from the Internet for testing our proposed method.

The remaining of the paper is organized as follows. Section 2 reviews the related work; Section 3 introduces the detail methods of FPLV; experiments and results could be found in Section 4; Section 5 concludes and discusses future work.

2 Related Work

To our best knowledge, face pixelation in live video streaming is not be studied before. Despite a few shot changes, live-streaming videos always contain frequent camera shakes and motion blur. Limited by conditions, streaming videos commonly involve with a single shot. But the video quality can only maintain at a relatively low level (720p, 30FPS with frequent camera movements). All these harsh realities cause more false positives and negatives in detection and recognition. Such results will directly cause the naively generated facial mosaics either blinking or incorrectly placed during the entire streaming. To meet the requirement of accuracy, we first try replacing the 2D deep CNN with its extension 3D CNN [Ji et al., 2013]. Taking an entire video as input, 3D CNN handles spatial-temporal information simultaneously and archives great success in video segmentation [Yue-Hei Ng et al., 2015; Karpathy et al., 2014] and action recognition areas [Ji et al., 2013; Tran et al., 2015]. Tested on the live-stream dataset we collected, 3D CNN models are quite time-costing and very sensitive to the video quality and length. Furthermore, as designed to deal relatively high-level abstractions of video contents, such models cannot competently handle the multi-task regression on frame-level face pixelation task and aggravate facial mosaics’ blinking.

Another possible way to realize FPLV is multi-object tracking (MOT) or multi-face tracking (MFT). YouTube Studio and Microsoft Azure published their off-line video face pixelation tool based on MFT. Although MOT and MFT are claimed to be thoroughly studied [Zhang and Gomes, 2014], the tracking accuracy is still challenging in unconstrained videos. Common online MOT or MFT algorithms are implemented through continuously motion prediction and corrections by Kalman-Filter [Babenko et al., 2011; Comaschi, 2016]. Because of the frequent camera movements in real-time streaming, such tracking algorithms struggle to handle the tracklets linking and drifting problems. Hence, tests of such tracking algorithms always build on high-resolution music video dataset [Zhang et al., 2016a] without massive camera moves and motion blur. Besides, to work properly, widely accepted solutions [Zhang et al., 2016b; Insafutdinov et al., 2017] of MOT or MFT in unconstrained videos assume some known priors including numbers of people showed up in videos, initial positions of the people, no fast motions, no camera movements, and high recording resolution. The state-of-the-art studies on MFT [Lin and Hung, 2018] manage to exclude as many priors as possible but require the full-length video in advance for analysis. Another realization of MOT is tracking by detection. Since focusing on motion prediction, likewise, the tracking by detection algorithms also generate tracklets even when targets are entirely blocked by others. This strategy is unacceptable in face pixelation because pixelation could only apply to faces that are recognizable to the audience. Otherwise, the inexplicable mosaics will be annoying and affect views’ experience. Therefore, the MOT
and MFT algorithms cannot be directly adapted to face the pixelation problem. In short, all the above techniques are far from satisfyingly realizing face pixelation mechanism in live video streaming.

3 Proposed Method

3.1 Detection and Recognition within Video Segments

We start by deploying the face detection and recognition network. In this paper, MTCNN proposed by [Zhang et al., 2016b], and Facenet proposed by [Schroff et al., 2015] process every frame in turn. Note that the detection and recognition network could be substituted by other recent published state-of-the-art work like PyrimaidBox [Tang et al., 2018] & Deepface [Taigman et al., 2014]. Instead of comparing or tuning developed face detection and recognition algorithms, we intentionally avoid using pioneer algorithms to demonstrate the performances of FPLV under relatively poor priors. MTCNN accepts arbitrary size inputs and detects face larger than 12×12 pixel. Facenet reads the face detection results as input and generates 512 dimension feature vector for each face. One more thing to mention here is the face alignment right after detection. Every detected face will be elaborately cropped and aligned to frontal pose through the affine transformation before feeding to the recognition network for embedding.

3.2 Raw Face Trajectories Forming

Clustering builds the connection of the same face across frames since its result could be facely corrected with some intervention. With the results of face detection and recognition, PIAP process in a segment-wise way to group the faces within and across segments simultaneously. Apparently, the longer segments offer more information but also incur more noise and reduce efficiency. In this paper, FPLV cuts every 150-frames into a segment and leverages the whole existing context to reach accurate and fast face pixelation. Define a data stream $Z = \{Z_1, Z_2, ..., Z_n\}$ is sequentially collected face feature vectors of the current video segment. i.e. $Z$ is an $512\times n$ matrix with 512 dimension feature vector. Starting with classical AP, the objective is to maximize:

$$Y = \sum_{i=1}^{n} s(i, c_i)$$

where $s(i, c_i)$ denotes the similarity between $Z_i$ and its nearest exemplar $Z_{c_i}$. For notation simplicity, we use $i, j \in R^{d}$ stands for the item $Z_i, Z_j \in Z$ in below.

According belief propagation algorithm, the responsibility $r(i, j)$ and availability $a(i, j)$ could be computed as:

$$r(i, j) = s(i, j) - \max_{j, j \neq j} \{a(i, j) + s(i, j)\}$$

(2)

$$a(i, j) = \min\{0, r(j, j) + \sum_{i,j \in \{i,j}\}}$$

(3)

Update responsibilities and availabilities according to (2) and (3) till convergence, then the clustering result of exemplars $r(i, j)$ and $a(i, j)$ in (2) and (3) are set to be zeros while initialization and (1)-(4) describe the widely used AP algorithm [Frey and Dueck, 2007]. The key challenge of building incremental AP is that the previously computed data nodes have already connected through nonzero responsibilities and availabilities. However, the newly arrived ones remain zero (detached). Therefore, the data nodes come at different timestamps are also stay at varying statuses with disproportionate relationships. Moreover, considering the time efficiency of message passing, simple continuously rolling affinity propagation at each time step cannot solve the incremental AP problem.

In our case, video frames are strongly correlated data. We could assign a proper value to newly arrived faces in adjacent frames according to the previous ones without affecting the clustering purity. ‘Comparing with others’, the feature vectors of a particular person’s faces across different frames are supposed to stay close with each other. That is, when these faces are consecutively detected and embedded, they should gather within a small area in the feature space. Moreover, since no massive shot change in our streaming videos, the motion of one person’s face shall remain tiny between adjacent frames. Thus, our incremental AP algorithm is proposed based on the facts that if two consecutively detected faces refer to one person, they should not only be clustered into the same group but also have same responsibilities and availabilities as well as stay close in physical positions. Both of these two facts are not considered in past studies of incremental affinity propagation [Sun and Guo, 2014; Yang et al., 2013].

Following the commonly used notations in the AP algorithm, the similarity matrix is denoted as $S_{t-v}$ at $t-v$ time with $(M_{t-v} * M_{t-v})$ dimension where $(t-v) = \frac{N}{FPS}$. And the responsibility matrix and availability matrix at time $t-v$ are $R_{t-v}$ and $A_{t-v}$ with a same dimension as $S_{t-v}$. Then, the update rule of $R_t$ and $A_t$ respect to $R_{t-v}$ and $A_{t-v}$ can be written as:

$$r_t(i, j) = \begin{cases} r_{t-v}(i, j), & i \leq M_{t-v}, j \leq M_{t-v} \\ r_{t-v}(i', j), & i > M_{t-v}, j \leq M_{t-v} \\ r_{t-v}(i, j'), & i \leq M_{t-v}, j > M_{t-v} \\ 0, & i > M_{t-v}, j > M_{t-v} \end{cases}$$

(5)

Note that the dimension of three matrices is increasing with time. $r_t(i, j)$ is the newly arrived face vectors of a segment at time $t$. $M_{t-v}$ stands for the amount of faces at time $t-v$.

$$i' = \arg \max_{i', i' \leq M_{t-v}} s(i, i')$$

(6)
Algorithm 1 Positioned Incremental Affinity Propagation

Input: \( R_{t-\varepsilon'} , A_{t-\varepsilon'} , \varepsilon' , \varepsilon , Z \)

Output: \( R_t , A_t , \varepsilon_t \)

1: while not end of a live-streaming do
2: if the first video segment of a live-stream then
3: Assign zeros to all responsibilities and availabilities.
4: else
5: Compute responsibilities and availabilities for \( Z \) according to equation (5) and (7).
6: Extend responsibilities matrix \( R_{t-\varepsilon'} \) to \( R_t \), and availabilities \( A_{t-\varepsilon'} \) to \( A_t \).
7: end if
8: Message-passing according to equation (2), (3) and (8) until convergence.
9: Compute \( \varepsilon_t \) as equation (4).
10: end while

Easily, the \( A_t \) could be updated through

\[
a_t(i, j) = \begin{cases} 
    a_{t-\varepsilon'}(i, j), & i \leq M_{t-\varepsilon'}, j \leq M_{t-\varepsilon'} \\
    a_{t-\varepsilon'}(i', j), & i > M_{t-\varepsilon'}, j \leq M_{t-\varepsilon'} \\
    a_{t-\varepsilon'}(i, j'), & i \leq M_{t-\varepsilon'}, j > M_{t-\varepsilon'} \\
    0, & i > M_{t-\varepsilon'}, j > M_{t-\varepsilon'} 
\end{cases}
\]  (7)

The entries of similarity matrix \( S \) with cosine similarities is defined as:

\[
s(i, j) = -\text{shrk} \frac{i \cdot j}{\|i\|\|j\|} \]  (8)

Where shrk is a shrinkage factor which estimates the shrinkage on temporal connections of two faces’ vectors between frames. Decided by the streaming quality we mentioned, in this paper we use such a dynamical shrinkage:

\[
\text{shrk} = \gamma \frac{\text{Resolution}}{\text{FPS}} \|P(i) - P(j)\|^2 \]  (9)

\( P(i), P(j) \in R^2 \) are the central pixel location for face vector \( i(Z_i) \) and \( j(Z_j) \) generated by corresponding bounding boxes. \( \gamma \) is the normalization. FPS and Resolution are the pair of variables for adjusting. In most cases, FPS remains as a constant like 30 and resolution may vary from 480-1080. This variable pair decides the motion and distortion allowed across frames for a particular face.

Handled by our proposed positioned incremental affinity propagation (PIAP) algorithm, we can quickly cluster faces in new segments, exclude the outliers as well as discovering newly appeared faces. Unlike newly spotted faces, the occasionally happens false negatives in the detection stage cannot form any steady trajectories and quickly isolated by other real faces while clustering. As a result, the scattered faces referring to the same person could be linked together sequentially to form our raw face trajectories.

3.3 Trajectory Refinement

Although well tuned MTCNN could reach 99.7% accuracy in the benchmark test, we cannot fully rely on it while dealing with streaming videos. In Figure 1, a well-trained face detector could offer us a raw trajectory with little breaks. Such kind of jumps within a trajectory could be quickly covered by interpolation. To fix gaps accumulated by false negatives in a trajectory, we build a proposal net structured in Figure 2. The proposal net resizes frames as MTCNN does, and proposes suspicious face areas in such gap frames. If some of the detection lost is captured through the proposal net, with interpolation and the captured detection lost, we can recursively refine the trajectories. As we do not require the proposal net to fully cover the detection lost, we also control the threshold to suppress the false positives under a high recall rate.

An observation of a raw trajectory is marked as a sequence \( z = \{ z_1, z_2, \ldots, z_m \} \). Then, a possible combination based on compensating detection result is a sequence \( z' = \{ z'_1, z'_2, \ldots, z'_m \} \) \((m' < m)\). Illustrated in Figure 2, \( z_f \) and \( z'_f \) are \( d \) \((d = 32)\) as we use the output of last layer before regression) dimensional feature vector of faces in \( z \) and \( z' \). There exists more than one \( z' \) to a single \( z \) as the characteristic of the proposal net, and we need to decide which \( z' \) is a propitiate compensation of \( z \) or none of them are appropriate.

Under ideal conditions, \( z' \) is expected to be captured directly by the deep face detection network. But some noise \( \varepsilon z' \) stops deep CNN from functioning and requires the help the proposal net. The expense of high recall rate is the massively generated false positives. A direct way to exclude these false positives is embedding them all into high-level features and separating the face or partial face detections for the wrong ones. The face recognition networks may realize such a process but the time cost is unaffordable since the number of proposed faces could be substantial. Therefore, we concentrate on finding a numerical way for high-level feature extraction.

According to Figure 3 that reveals the relation of \( z \) and \( z' \), a Gaussian Process (GP) model is applied to reflect such an inference correlation of \( z \) and \( z' \):

\[
z' = f(z, \theta) + \varepsilon z'
\]  (10)
where \( f \) is Gaussian Process with hyperparameter \( \theta \), and independent noise term \( \varepsilon_{z'} \sim N(0, \sigma^2 I) \). \( I \) is the identity matrix. Some studies in MFT [Lin and Hung, 2018] fields applied a similar concept for prediction optimization.

The Central Limit Theorem (CLT) implies that the outputs of \( f(\theta, z) + \varepsilon_{z'} \) converge to a multivariate Gaussian distribution. A regular way to derive high-level parametric abstractions of \( z \) and \( z' \) is finding the maximum likelihood estimates (MLE) of \( \theta \) denote as \( \hat{\theta} \):

\[
\hat{\theta} = \arg \max_{\theta} \{ \ln p(\theta | z, z') \} \tag{11}
\]

and check the range of \( \hat{\theta} \) to decide whether \( z \) and \( z' \) stay close enough such that they could be regarded as samples from a same Gaussian distribution.

This GP model is sufficient for false positives rejection in most cases [Lin and Hung, 2018]. However, in live-streaming, we manipulate on short video segments. The small amount of frames with high dimensional feature vectors violates the initial assumption of CLT. That is, the probability of \( \hat{\theta} \) is not a Gaussian RKHS [Fukumizu, 2004].

We measure the distribution of face vectors is a compact set defined at a Gaussian kernel.

The goal of two-sample test problem is to determine whether two distributions are different on basis of samples. A hypothesis \( \mathcal{H}_0 : P_z = P_{z'} \) stands if and only if \( z \) and \( z' \) are from the same distribution. If two-sample test for \( z \) and \( z' \) stands, we should reject \( \mathcal{H}_0 \) [Ciuperca and Salloum, 2016]. As described in (10), a fully Bayesian treatment of neural networks is also a Gaussian distribution. Then we assume the domain of face vectors is a compact set defined at a universal Gaussian reproducing kernel Hilbert space (RKHS).

And the kernel function is a Gaussian kernel \( k_G \). Following the widely used maximum mean discrepancy (MMD), we measure the distribution \( z \) and \( z' \) by embedding them into such a Gaussian RKHS [Fukumizu et al., 2004]. Therefore, the squared MMD between two observations \( z \) and \( z' \) is the squared RKHS distance between their respective mean embedding [Chwialkowski et al., 2016]. Referring [Ciuperca and Salloum, 2016], \( \mathcal{H}_0 \) can be rewritten as:

\[
\mathcal{H}_0 : P_z = P_{z'} \quad \text{Where} \quad z = (x, y), z' = (x', y')
\]

\[
\Leftrightarrow MMD^2[z, z'] = E[h(z, z')] = 0 \tag{12}
\]

Where

\[
h(z, z') = ||x - x'|| + ||y - y'|| - ||x' - y'|| - ||y' - x'|| = k_G(x, x') + k_G(y, y') - k_G(x', y') - k_G(x, y') = 0 \tag{13}
\]

The solution of (12) under large sampled points of \( z \) and \( z' \) approximates the above MLE of a multi-variant Gaussian Process. i.e. \( \hat{\theta} = 0 \) where \( \hat{\theta} \in (N * d) \) is the mean value. Due to CTL is not a appropriate assumption in short video segments, and Figure 3 shows \( z \) and \( z' \) are i.i.d which implies \( h_i, i = 1, ..., N \) are also i.i.d observations, we introduce the empirical likelihood ratio (ELR)\(^1\) for solving the two-sample test of \( \mathcal{H}_0 : \hat{\theta} = 0 \) as:

\[
L(p) = \sup_{p_i} \{ \prod_{i=1}^N p_i, \sum_{i=1}^N p_i = 1, \sum_{i=1}^N p_i h_i = 0 \} \tag{14}
\]

Where \( p_i \) is the empirical likelihood ratio, and \( p_i \) is subject to the normalization and \( \hat{\theta} = 0 \) constrains. Therefore, an explicit solution of (14) is attained from a Lagrange multiplier argument:

\[
p_i = \frac{1}{N 1 + \lambda h_i} \sum_{i=1}^N \frac{h_i}{1 + \lambda h_i} = 0 \tag{15}
\]

According to (14) and the Wilk’s Theorem, the supremum is transferred to the ELR test

\[
T_{ELR} = -2 \log L(p) \xrightarrow{d} X^2 \tag{16}
\]

under \( \mathcal{H}_0 : P_z = P_{z'}, \text{if } E_{z,z'}[K_G^2(z, z')] < \infty \).

Where \( X^2 \) is the chi-square distribution with 1 degree of freedom according to Wilk’s Theorem, and \( T_{ELR} \) converge in that distribution. Thus, we can reject the null hypothesis \( \mathcal{H}_0 \) when \( T_{ELR} \geq X^2_\alpha \), and \( X^2_\alpha \) is the probability of the confidence interval:

\[
Pr(X^2 \geq X^2_\alpha) = \alpha \tag{17}
\]

The rejection threshold of (17) could be computed through off-the-shelf python package. Therefore, leveraging \( p_i \) as the new parametric feature to eliminate the false positives raised by proposal net is extremely time-saving without any parametric assumptions. The cost is solely related to \( T_{ELR} \) computation according to (15) which is linear to the sample size. Our two-sample test algorithm via empirical likelihood with Gaussian kernel achieves impressing results in real tests.

With the efforts of (16) and (17), the received \( z' \) is added to \( z \)’s trajectory. Fast interpolation once again applies to fill up the tiny jumps still exist and then generate the refined trajectory. Final face pixelation overlaps Gaussian Filters on non-steamer’s trajectories to blur the irrelevant people’s face. In this way, the automatic personal privacy protection in live video streaming is implemented through FPLV.

4 Experiments

4.1 Details

Dataset: As face pixelation problem in live video streaming is not studied before, there are no available dataset and benchmark tests for reference. We collected and built a live-streaming video dataset from YouTube and Facebook platforms and manually pixelated the faces for comparison. Limited by the current source, we select four short and relatively not complicated live-streaming fragments from our

\(^1\)More detailed derivations on: fplv.github.io
### Table 1: Details of the live-stream dataset

| Property | Frames | Resolution | Number of people |
|----------|--------|------------|------------------|
| HS       | 6753   | 720p       | 4                |
| LS       | 6604   | 480p       | 3                |
| HN       | 7721   | 1080p      | 2                |
| LN       | 9401   | 480p       | 1                |

Table 2: Precision on live-stream video dataset

| Method               | HS  | LS  | HN  | LN  |
|----------------------|-----|-----|-----|-----|
| YouTube              | 0.85| 0.78| 0.84| 0.89|
| Azure                | 0.77| 0.53| 0.77| 0.75|
| TLD                  | 0.79| 0.72| 0.76| 0.72|
| PyramidBox+Deepface  | 0.92| 0.84| 0.98| 0.82|
| FPLV                 | 0.98| 0.96| 0.97| 0.97|

Table 3: Accuracy on live-stream video dataset

| Method               | HS  | LS  | HN  | LN  |
|----------------------|-----|-----|-----|-----|
| YouTube              | 0.87| 0.82| 0.88| 0.80|
| Azure                | 0.83| 0.67| 0.84| 0.76|
| TLD                  | 0.75| 0.72| 0.78| 0.71|
| PyramidBox+Deepface  | 0.61| 0.44| 0.66| 0.47|
| FPLV                 | 0.93| 0.83| 0.91| 0.83|

### 4.2 Results on live video streaming data
Self-evidently, FPLV generates best face pixelation results as listed in Table 3. Note that although YouTube studio performs relatively close to our FPLV than other algorithms, the face pixelation tool of YouTube and Azure cannot process the video in real-time. Figure 4 demonstrates a partial result of FPLV comparing to YouTube Studio. In a live-stream of famous singer Jam Hsiao on Facebook, the upper line results of FPLV well handled the pixelation task when two faces overlap each other. The bottom line generates by YouTube through offline face tracking algorithms fails such a pixelation as other person’s mosaics are incorrectly overlaid on streamer’s face.

### 4.3 Efficiency
The main cost of FPLV is on the face embedding algorithm. The embedding for one face takes 10-15ms on our i7-7800X, GTX1080, 32G RAM machine, and the face detection including compensation for a frame take 3ms. Another time-costing part is the initialization of IPAP; the initialization process takes 10-30ms depending on the case. And each incremental propagation loop takes 3-5ms. So FPLV can satisfy the real-time efficiency requirement, and under extreme circumstances that contain a lot of faces, we can trick the detection and recognition to jump among frames for efficiency.

### 5 Conclusions
According to our knowledge, we are the first to address the face pixelation problem in live video streaming by building the proposed FPLV. FPLV is already surpassing offline tools offered by YouTube and Microsoft and becomes applicable in real life scenarios. With Apple A12 chips, FPLV could be potentially transferred to smartphones without much computation burden. So far, FPLV can achieve high accuracy and real-time performances on the dataset we collected and we will further extend FPLV to behave better on low-resolution scenarios.

---

2 More video results on: fplv.github.io
References

[Babenko et al., 2011] Boris Babenko, Ming-Hsuan Yang, and Serge Belongie. Robust object tracking with online multiple instance learning. *IEEE transactions on pattern analysis and machine intelligence*, 33(8):1619–1632, 2011.

[Chwialkowski et al., 2016] Kacper Chwialkowski, Heiko Strathmann, and Arthur Gretton. A kernel test of goodness of fit. *JMLR: Workshop and Conference Proceedings*, 2016.

[Ciuperca and Salloum, 2016] Gabriela Ciuperca and Zahraa Salloum. Empirical likelihood test for high-dimensional two-sample model. *Journal of Statistical Planning and Inference*, 178:37–60, 2016.

[Comaschi, 2016] F Comaschi. *Robust online face detection and tracking*. PhD thesis, Technische Universiteit Eindhoven, 2016.

[Frey and Dueck, 2007] Brendan J Frey and Delbert Dueck. Clustering by passing messages between data points. *science*, 315(5814):972–976, 2007.

[Fukumizu et al., 2004] Kenji Fukumizu, Francis R Bach, and Michael I Jordan. Dimensionality reduction for supervised learning with reproducing kernel hilbert spaces. *Journal of Machine Learning Research*, 5-Jan:73–99, 2004.

[Insafutdinov et al., 2017] Eldar Insafutdinov, Mykhaylo Andriluka, Leonid Pishchulin, Siyu Tang, Evgeny Levinkov, Bjorn Andres, and Bernt Schiele. Arttrack: Articulated multi-person tracking in the wild. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6457–6465, 2017.

[Ji et al., 2013] Shuiwang Ji, Wei Xu, Ming Yang, and Kai Yu. 3d convolutional neural networks for human action recognition. *IEEE transactions on pattern analysis and machine intelligence*, 35(1):221–231, 2013.

[Karpathy et al., 2014] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. Large-scale video classification with convolutional neural networks. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 1725–1732, 2014.

[Lin and Hung, 2018] Chung-Ching Lin and Ying Hung. A prior-less method for multi-face tracking in unconstrained videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 538–547, 2018.

[Schroff et al., 2015] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 815–823, 2015.

[Sun and Guo, 2014] Leilei Sun and Chonghui Guo. Incremental affinity propagation clustering based on message passing. *IEEE Transactions on Knowledge and Data Engineering*, 26(11):2731–2744, 2014.

[Taigman et al., 2014] Yaniv Taigman, Ming Yang, Marc’Aurelio Ranzato, and Lior Wolf. Deepface: Closing the gap to human-level performance in face verification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1701–1708, 2014.

[Tang et al., 2018] Xu Tang, Daniel K Du, Zeqiang He, and Jingtuo Liu. Pyramidbox: A context-assisted single shot face detector. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 797–813, 2018.

[Tran et al., 2015] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning spatiotemporal features with 3d convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 4489–4497, 2015.

[Yang et al., 2013] Chen Yang, Lorenzo Bruzzone, Renchu Guan, Laijun Lu, and Yanchun Liang. Incremental and decremental affinity propagation for semisupervised clustering in multispectral images. *IEEE Transactions on Geoscience and Remote Sensing*, 51(3):1666–1679, 2013.

[Yu et al., 2017] Jun Yu, Baopeng Zhang, Zhengzhong Kuang, Dan Lin, and Jianping Fan. iprivacy: image privacy protection by identifying sensitive objects via deep multi-task learning. *IEEE Transactions on Information Forensics and Security*, 12(5):1005–1016, 2017.

[Yue-Hei Ng et al., 2015] Joe Yue-Hei Ng, Matthew Hausknecht, Sudheendra Vijayanarasimhan, Oriol Vinyals, Rajat Monga, and George Toderici. Beyond short snippets: Deep networks for video classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4694–4702, 2015.

[Zhang and Gomes, 2014] Tong Zhang and Herman Martins Gomes. Technology survey on video face tracking. In *Imaging and Multimedia Analytics in a Web and Mobile World 2014*, volume 9027, page 90270F. International Society for Optics and Photonics, 2014.

[Zhang et al., 2016a] Shun Zhang, Yihong Gong, Jia-Bin Huang, Jongwoo Lim, Jinjun Wang, Narendra Ahuja, and Ming-Hsuan Yang. Tracking persons-of-interest via deep multi-task learning. *IEEE Transactions on Information Forensics and Security*, 12(5):1005–1016, 2016.

[Zhang et al., 2016b] Zhanpeng Zhang, Ping Luo, Chen Change Loy, and Xiaoou Tang. Joint face representation adaptation and clustering in videos. In *European conference on computer vision*, pages 236–251. Springer, 2016.