A Scalable and Privacy-Aware IoT Service for Live Video Analytics

Junjue Wang
Carnegie Mellon University
junjuew@cs.cmu.edu

Brandon Amos
Carnegie Mellon University
bamos@cs.cmu.edu

Anupam Das
Carnegie Mellon University
anupamd@cs.cmu.edu

Padmanabhan Pillai
Intel Labs
padmanabhan.s.pillai@intel.com

Norman Sadeh
Carnegie Mellon University
sadeh@cs.cmu.edu

Mahadev Satyanarayanan
Carnegie Mellon University
satya@cs.cmu.edu

ABSTRACT

We present OpenFace, our new open-source face recognition system that approaches state-of-the-art accuracy. Integrating OpenFace with inter-frame tracking, we build RTFace, a mechanism for denaturing video streams that selectively blurs faces according to specified policies at full frame rates. This enables privacy management for live video analytics while providing a secure approach for handling retrospective policy exceptions. Finally, we present a scalable, privacy-aware architecture for large camera networks using RTFace.

CCS CONCEPTS

•Security and privacy → Domain-specific security and privacy architectures; *Privacy protections;* **•Computing methodologies** → *Computer vision tasks;* **•Computer systems organization** → *Cloud computing;*

GENERAL TERMS

Design, Experimentation, Measurement, Performance

KEYWORDS

privacy protection, face recognition, cloud computing, cloudlet

1 Introduction

Video cameras are ubiquitous today. A recent survey in the U.K. estimated one surveillance camera in a public space for every 11 people [3]. Despite privacy concerns, there was a 62% positive response on the *Debate.org* public opinion web site to the question "Are video surveillance cameras in public places a good idea?" [10].

If privacy concerns can be satisfactorily addressed, enabling real-time analytics on video streams from public spaces can be valuable. Examples abound. In a shopping mall, an alert triggered by detection of a liquid spill or broken glass can be promptly addressed. An "amber alert" for a lost or abducted child can trigger real-time face recognition in parallel on many video streams from public spaces and help to rapidly locate the child. Real-time activity inferencing on outdoor camera feeds can detect pedestrians slipping on an icy sidewalk and trigger corrective measures. Real-time video analytics can also offer business value, e.g., prompt detection and remedy of lengthening checkout lines to improve customer satisfaction. Prompt discovery that shoppers are not lingering at a new window display may suggest that it is not as appealing as expected, and needs to be changed soon. Early detection of icy sidewalks benefits the bottom line of insurance companies, since they underwrite risk.

Live video offers several advantages relative to other sensing modalities. Most important is its flexibility and open-endedness: new image and video processing algorithms can be developed to enhance the information extracted from an existing video stream. Additionally, video offers high resolution, wide coverage, and low cost relative to other sensing modalities. The passive nature of video sensing is especially attractive for public spaces. A participant does not have to wear a special device, install an app, or do anything special. He or she merely has to be visible to a camera.

In this paper, we describe a system that offers privacy-aware live video as an IoT service. Following the lead of Simoens et al. [30], we use the term *denaturing* to refer to the process of making video "safe" from a privacy viewpoint. A denatured video stream is one whose content has been analyzed to identify potential privacy leaks, and has been modified to eliminate those leaks. Section 2.1 discusses the denaturing concept in more detail. A denatured video stream may be safely released to untrusted video analytics software.

Our prototype focuses on *face recognition* as the basis of denaturing for two reasons. First, it leads to a compelling proof of concept because accurate real-time face recognition is a difficult technical challenge even today. Second, recognition of specific individuals lies at the heart of many important real-world policies.

MMSys'17, Taipei, Taiwan
© 2017 Copyright held by the owner/author(s). 978-1-4503-5002-0/17/06...$15.00
DOI: http://dx.doi.org/10.1145/3083187.3083192

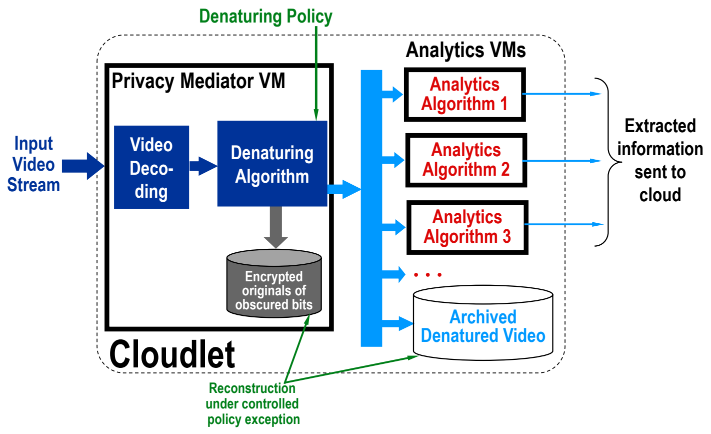


Figure 1: Architecture for Live Video Denaturing and Analytics

Figure 1 illustrates the overall architecture of our system. We use *edge computing* rather than cloud computing to perform denaturing and video analytics. Edge computing is a nascent model of computing in which small multi-tenant data centers called *cloudlets* are located close to IoT devices such as video cameras. Davies et al. [9] have explained why denaturing on a cloudlet offers greater confidence in privacy enforcement than denaturing in the cloud. They refer to the software module performing denaturing as a *privacy mediator.* Scalability is also improved by performing denaturing and video analytics on cloudlets: by avoiding transmission of video into the cloud, ingress bandwidth demand per camera is greatly reduced.

This paper makes the following contributions:

(1) We introduce *OpenFace,* a new open-source face recognition algorithm whose accuracy approaches that of the best proprietary algorithms. OpenFace can be trained to recognize a new face in just a few tenths of a second.
(2) We show how OpenFace can be combined with face tracking across video frames to implement fast, policy-guided denaturing. The resulting denaturing system, called *RTFace* has a very low rate of privacy leaks.
(3) We show how RTFace can safely preserve obscured bits to allow reconstruction under policy-specified conditions such as a search warrant or an insurance claim.
(4) We discuss extensions and performance considerations in applying RTFace at enterprise scale.
(5) We present experimental results that validate the speed and scalability of RTFace.

A video demo of RTFace is at https://youtu.be/gQa8oScFS94.

2 Background and Related Work

2.1 Denaturing

Denaturing involves content-based modification of images or video frames, guided by a privacy policy. In general, this policy has to strike a balance between privacy and value. One extreme of denaturing is a blank video: perfect privacy, but zero value. At the other extreme is the original video stream. This has the highest value for video analytics, but also incurs the highest exposure of privacy. Where to strike this balance is a policy issue, and the answer will likely vary across individuals and contexts. Consider the shopping mall example mentioned earlier. In the large area covered by the mall, there may be tens or hundreds of video cameras deployed. Since they are all part of a single system that is controlled by the mall management, mall-wide privacy policies are meaningful. For example, the mall may implement a default "opt-in" privacy policy that blurs the faces of all individuals unless they explicitly request otherwise. While she is in a hair salon, a mall customer might temporarily opt-in her child so that she can keep an eye on him. Her smartphone app continuously displays a live video stream with the child's face visible, dynamically switching to a newly-optimal camera as the child wanders about exploring a large toy store.

Although policy and mechanism are ideally disjoint, the mechanism determines the range of enforceable policies. The sophistication, accuracy, and speed of image processing algorithms is a key determinant of the mechanism that can be supported for live denaturing. Figure 2 shows three examples of denatured video frames.

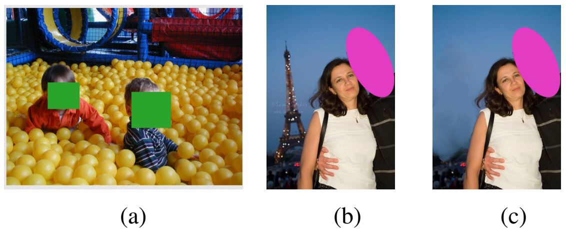


Figure 2: Examples of Denatured Video Frames

Figure 2(a) only requires face detection. Once the bounding boxes of faces have been determined, blurring those pixels is trivial. Figure 2(b) shows selective, policy-guided blurring of faces. Perhaps the person whose face is visible is a well-known public figure, with a lower expectation of privacy. Implementing this policy requires face recognition, after face detection. This paper focuses on scenarios such as Figure 2(b). Figure 2(c) is a more aggressively denatured version of Figure 2(b) because even the landmark which gives the location of image capture has been obscured. This requires object recognition of well-known buildings. With the implementation of appropriate landmark detection algorithms (already supported by some augmented reality applications today), it would be straightforward to extend our system to support scenarios such as Figure 2(c).

2.2 Content-based Image Privacy

Our work focuses on live video analytics. This forces us to solve the problem of real-time video denaturing at full frame rate. Although implementation is less challenging if video analytics is done offline and/or on a sample of the input data, the examples given earlier caution against oversimplification. For instance, activity inferencing to detect people slipping on ice is reliable at full frame rate, but is less reliable on a low-frequency sample. Doing the analytics offline will not help to avoid impending accidents.

Image privacy based on offline or sampled data has been the subject of many recent papers. Simoens et al. [30] describe an approach in which denaturing is performed on a low-frequency sample of video frames. Aditya et al. [1] perform selective face denaturing using DNN-based person recognition that combines facial features with hairstyle and context information. However, even with GPUs on mobile devices, they still require many seconds per image. Bo et al. [5] describe a reversible obfuscation scheme for denaturing faces. However, the work requires users to wear clothes with a printed barcode and does not run in the real-time. Raval et al. [27] use tracking to selectively block objects physically or virtually tagged by users at near real time on mobile devices. Automated recognition and policy-based denaturing, as enabled by our work, are not supported. Jana et al. [15] investigate video transformations that preserve privacy, yet allow many vision processing algorithms to still work.

Previous works [11] [24] on de-identification study effective image modifications to preserve the anonymity of people from face recognition software. They assume the locations of faces in an image are known. Different from de-identification studies, our work explores how to automatically identify areas of interest in an accurate and timely fashion. We adopt a conservative scheme of de-identification by changing the color of all sensitive pixels to black. Advanced de-identification methods can be combined with our work.

2.3 Face Recognition

Face recognition has been an active research topic since the early 1970s [18]. Although recognizing a human face can be viewed as a special case of object recognition, challenges arise because the face is not a rigid object and the recognition process needs to be robust with respect to variations such as aging, facial expression, makeup, and hair styling [16]. For many decades, progress in face recognition was slow. Jafri and Arabnia [14] provide a comprehensive survey of this evolution, identifying landmark systems such as Eigenfaces (1991) [34] and Fisherfaces (1997) [4].

Only since 2012, with the adoption of techniques based on *deep convolutional neural networks (DNN)*, has progress on face recognition accelerated. Facebook's DeepFace (2014) [32] and Google's FaceNet (2015) [29] achieve significant improvements over traditional approaches and yield near-human accuracy today. Unfortunately, these DNN-based systems are trained with private datasets containing millions of proprietary images from social media products. These datasets are one to two orders of magnitude larger than publicly-available datasets of labeled human faces. Further, the face recognition implementations are not directly accessible to the public. Hence, we have been forced to develop our own DNN-based face recognition system as described in the next section.

3 OpenFace

Humans are exquisitely accurate and fast in recognizing the faces of other humans. In a typical workday involving encounters with hundreds of individuals, we may not make a single mistake in face recognition. We can also learn new faces from very brief encounters. Ramon et al. [26] found that it takes a normal human about 370 ms to recognize a face as familiar, and around 620 ms to realize a face is unfamiliar. For full recognition of the identity of a face, Kampf et al. [17] found that normal humans take about 1000 ms. OpenFace is inspired by human accuracy and speed, and approaches the state-of-art accuracy of DeepFace and FaceNet.

3.1 Training and Classification

To obtain high accuracy and speed, OpenFace uses DNN-based feature extraction followed by classification using an SVM (support vector machine). Figure 3 shows the steps involved in recognizing faces in a video frame. The first step is to identify the bounding boxes of faces in the frame using the standard `Dlib` face detector [20]. Each detected face is then processed independently. Since the face may be oriented in an arbitrary direction, the 2D affine transform shown in Figure 4 is used to normalize its pose. The normalized face is fed to a pre-trained DNN, whose output is a 128-element feature vector that uniquely characterizes each face. We then feed this feature vector to a linear SVM classifier of known faces to produce a single identity and a confidence estimate.

We train the DNN for feature extraction with roughly 500,000 labeled images obtained by combining the publicly-available CASIA-WebFace [35] and FaceScrub [25] datasets. This DNN uses a modified version of FaceNet's `nn4` network, which is itself based on the GoogLeNet [31] architecture. Our modifications produce a variant with fewer parameters for our smaller dataset. Table 1 details OpenFace's network structure. The network in total has 3.7 million parameters. Each row is a layer in the neural network and the last six

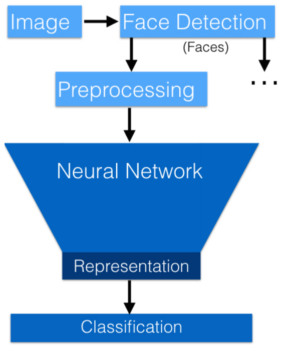


Figure 3: Face Recognition Steps

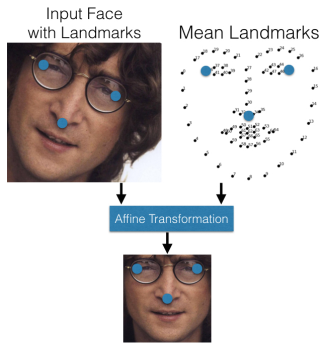


Figure 4: Normalization

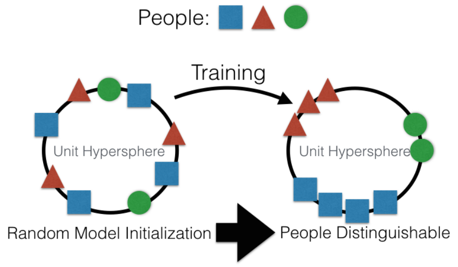


Figure 5: Embedding on Unit Hypersphere

columns indicate the parameters of pooling or the inception layers from [31]. Dimensionality reductions to N dimensions after pooling is denoted with "Np". The OpenFace DNN can be implemented on any deep learning framework. We used Torch [7] because it had most of the features we needed, for instance the inception layer. Torch also enables us to run the DNN on both CPUs and GPUs. The training procedure uses a triplet loss function that results in an embedding of images on the unit hypersphere producing 128-element normalized feature vectors, with Euclidean distance representing similarity (Figure 5). We map unique images into triplets, and the gradient of the triplet loss is backpropagated back through the mapping to the unique images (Figure 6).

In each mini-batch, we sample at most P images per person from Q people in the dataset and send all $M \approx PQ$ images through the network in a single forward pass on a GPU to get M feature vectors. We currently use $P = 20$ and $Q = 15$, for an NVDIA Tesla K40 GPU with 12 GB of memory. We take all anchor-positive pairs to obtain $Q\binom{P}{2}$ triplets, compute their triplet losses, and map the derivatives through a backwards network pass to the original images. Training on our dataset takes roughly 48 hours on a system with an 8-core Intel® Xeon® E5-1630 v3 processor and the above GPU.

3.2 Accuracy and Performance

The simplest recognition task is to say whether a pair of images are of the same person. Figure 7 compares the ROC curve of OpenFace on this task to previously-published DeepFace results on the restricted protocol of the LFW benchmark dataset [12]. The ROC curve of FaceNet has not been published, but is inferred to be similar to DeepFace. The labels give the area under the curve (AUC) figure for each classifier. The OpenFace AUC of 0.973 approaches that of humans (0.995) and DeepFace (0.997). It is a major improvement on

Layer	Type	Output Size	#1×1	#3×3 reduce	#3×3	#5×5 reduce	#5×5	pool proj
1	conv1 $(7\times7\times3,2)$	$48\times48\times64$						
2	max pool + norm	$24\times24\times64$						m 3×3, 2
3	inception (2)	$24\times24\times192$		64	192			
4	norm + max pool	$12\times12\times192$						m 3×3, 2
5	inception (3a)	$12\times12\times256$	64	96	128	16	32	m, 32p
6	inception (3b)	$12\times12\times320$	64	96	128	32	64	ℓ_2, 64p
7	inception (3c)	$6\times6\times640$		128	256,2	32	64,2	m 3×3, 2
8	inception (4a)	$6\times6\times640$	256	96	192	32	64	ℓ_2, 128p
9	inception (4e)	$3\times3\times1024$		160	256,2	64	128,2	m 3×3, 2
10	inception (5a)	$3\times3\times736$	256	96	384			ℓ_2, 96p
11	inception (5b)	$3\times3\times736$	256	96	384			m, 96p
12	avg pool	736						
13	linear	128						
14	ℓ_2 normalization	128						

Table 1: The OpenFace `nn4.small2` Deep Neural Network Definition

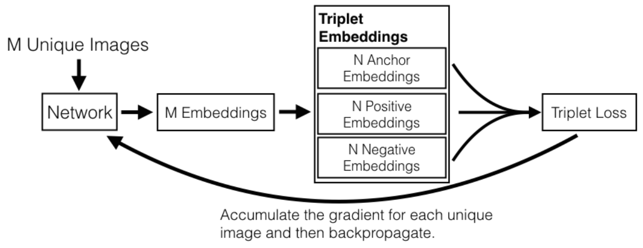


Figure 6: Training Procedure

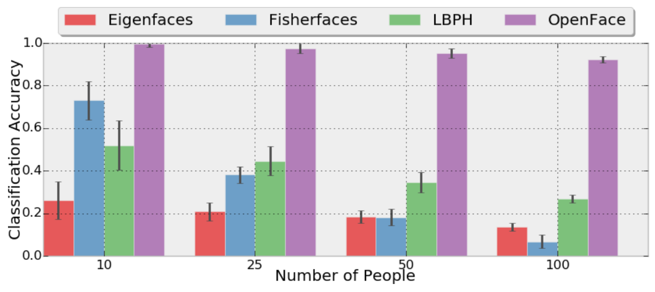


Figure 8: Identification Accuracy on LFW Benchmark

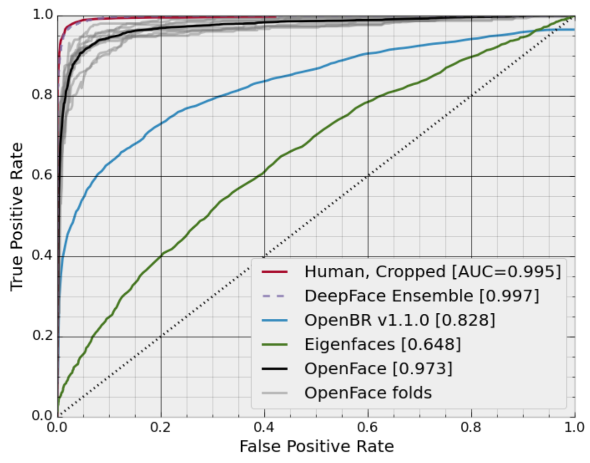


Figure 7: Face Recognition Accuracy (ROC) on LFW Benchmark

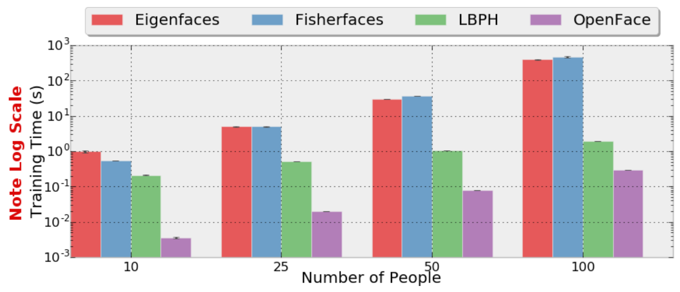


Figure 9: Training Time for New Faces

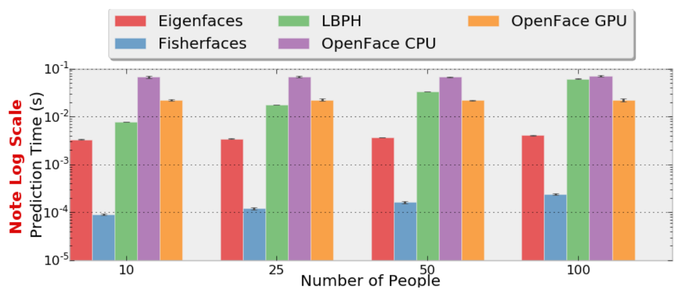


Figure 10: Recognition Speed

OpenBR (0.828) and Eigenfaces (0.648), which are two widely-used open source face recognizers. The curves labeled "OpenFace folds" shows the observed variation when different random subsets of the LFW dataset are used for training and testing.

On specific tasks below, we compare OpenFace to Eigenfaces [34], Fisherfaces [4], and LBPH [2]. Comparison on these tasks with DeepFace and FaceNet are not possible because those algorithms are not publicly available for experiments.

Identification of Faces: Implementing a face-based denaturing policy requires a face to be identified as a specific member of a collection of known faces. This is a different classification task from above, where the goal was only to state if two faces are the same. The task becomes harder as the number of faces in the collection increases. Figure 8 shows that OpenFace has the highest accuracy by a large margin, and its accuracy falls off slowly as the number of faces increases from 10 to 100. While Openface can be used to recognize more faces, we are unable to report a larger scale evaluation due to the limitations of the LFW dataset.

Incremental Training for New Faces: Once the DNN has been trained as described in Section 3.1, adding a new identity only requires the SVM classifier (last step of Figure 3) to be retrained. The time for this depends on the total number of known identities. As shown in Figure 9, this takes well below one second even for a collection size of 100 known faces. The figure also shows that OpenFace is much faster and more scalable than the alternatives.

Recognition Speed for a Single Face: Figure 10 compares the speed of OpenFace in a recognition task to the alternatives, using the same hardware mentioned in Section 3.1. The figure shows that recognition speed is roughly 80 ms without a GPU, dropping to roughly 20 ms with a high-end GPU. The recognition speed does not depend on the size of face collection. Relative to the alternatives, the superior accuracy of OpenFace comes at a significant price in terms of speed. Relative to the human speeds (many hundreds of milliseconds) [17] [26], OpenFace is clearly competitive.

4 Denaturing at Full Frame Rate

Our goal is to match human accuracy while meeting the speed requirements for live video. In contrast to human speed, denaturing a sequence of video frames at 30 fps only allows about 33 ms per frame. Hence, face recognition for denaturing has to be one to two orders of magnitude *faster* than human performance, while preserving near-human accuracy. The challenge increases if there are many faces to be recognized in each frame.

4.1 Denaturing without Tracking

Figure 11 illustrates a simple pipeline for denaturing. After video decoding, a face detection algorithm is run to find a bounding box around each face in the frame. These faces may be at any scale, from a tiny face in the distance to a large face that fills most of the frame. The best face detectors available today are scale invariant, but they are sensitive to the orientation. Detection of frontal faces is most reliable, while detection of faces in profile is significantly less reliable. After face detection, each bounding box is presented to OpenFace for recognition. Within the limits of available hardware resources, OpenFace can use coarse-grain parallelism to concurrently work on multiple bounding boxes in the recognition step. Finally, the bits within the bounding boxes are selectively modified according to an identity-specific denaturing policy. If an opt-in policy for revealing faces is used, any unrecognized face is obscured. If an opt-out policy is used, those faces remain visible. Before modifying a bounding box, its original bits are encrypted and then saved on a virtual disk within the privacy mediator VM. The purpose and details of this step are discussed in Section 6. The denatured frame is then released for video analytics.

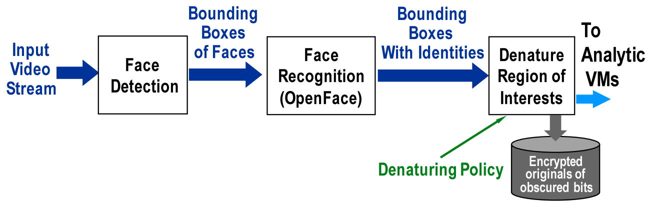

Input Video Stream
Face Detection
Bounding Boxes of Faces
Face Recognition (OpenFace)
Bounding Boxes With Identities
Denature Region of Interests
To Analytic VMs
Denaturing Policy
Encrypted originals of obscured bits

Figure 11: Simple Denaturing Pipeline

Operation	Speed
OpenFace w/o GPU	30 (0.1) ms per face
Dlib face detection	127 (1) ms per frame
Dlib face tracking	7.6 (0.1) ms per bounding box
Perceptual hashing	0.3 (0.1) ms per bounding box

Processing speed in a VM running on a 4-core Intel® Core™ i7-4790 processor @3.6 GHz with Hyper-Threading, and no GPU. The reported performance is averaged over 3 runs across 100 images with resolution 1280x720. Numbers in parentheses are standard deviations. Note that the time for OpenFace here cannot be directly compared to the value in Figure 10 because the hardware and linked shared libraries are different.

Table 2: Speeds of Critical Operations

Unfortunately, the simple denaturing pipeline in Figure 11 is unable to achieve adequate speed. Assuming that the time for the third step (interpreting policy and obscuring bits) is negligible, the best-case latency of the pipeline is the sum of the first two steps: face detection time on a frame, plus the slowest face recognition in a set of parallel recognitions. Table 2 shows the measured speed of face detection and recognition on our denaturing hardware. With these speeds, the pipeline of Figure 11 is clearly too slow. The sum of Dlib face detection and one face recognition is over 150 ms, far exceeding our budget of 33 ms. Using a GPU does not help to speed up face detection — it only speeds up face recognition.

4.2 Combining Recognition with Tracking

To speed up the denaturing pipeline, we build on the insight that face motion between two successive frames is likely to be small. Even at athletic speeds, human motion is slow enough that translations and rotations of a face in successive frames are likely to involve changes only to a proximal region of pixels. We leverage this insight by combining initial detection and recognition of faces, followed by *tracking* of those faces. Fast object tracking algorithms that are an order of magnitude faster than face detection and recognition have been extensively discussed in the computer vision literature. We use tracking to eliminate face detection and recognition from the timing-critical path of denaturing. Note that this is a different goal and usage context from the use of object tracking described by Chen et al. [6]. There, the goal is to reduce computational load on a mobile device by offloading the expensive object recognition step to a cloudlet and only performing the cheaper object tracking on the mobile device.

Object tracking initializes with an image and a bounding box from that image. It outputs an updated bounding box for the subsequent frame. We use a tracking-by-detection algorithm from Dannelljan

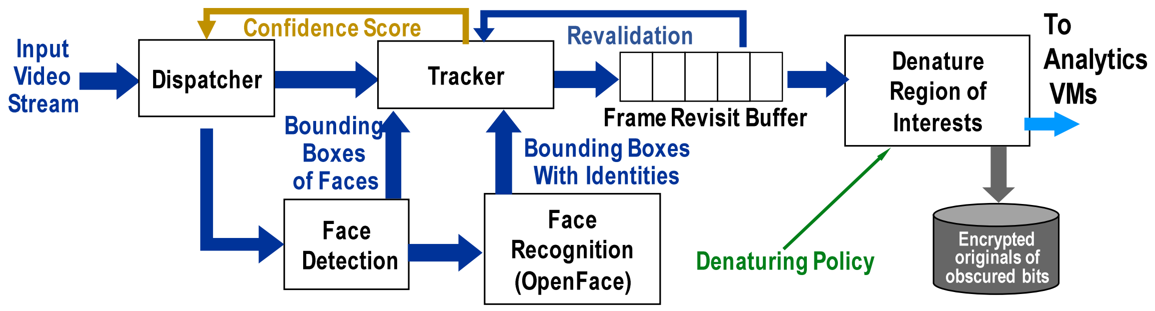


Figure 12: RTFace: Denaturing Pipeline with Recognition and Tracking

et al. [8] known for its robustness. The Dlib implementation takes 7.6 ms per frame on the denaturing hardware of Table 2.

Unfortunately, tracking is imperfect and significant drift can accumulate across many frames between the predicted and real locations of a face. In addition, a new face may only become detectable for the very first time in a particular frame even though it was present in earlier frames. These effects can lead to *privacy leaks,* in which a face that should be obscured is, instead, visible in the denatured video stream. As discussed in Section 5, we reduce privacy leaks by revalidating frames before release and conservatively biasing denaturing towards obscuring too much rather than too little.

4.3 RTFace: Revised Denaturing Pipeline

Figure 12 shows a denaturing pipeline called *RTFace* that combines face recognition and face tracking. When a video frame arrives, the dispatcher module decides whether it needs to be sent to the face detection module. Regardless of that decision, the frame is passed on to the tracker module. If there are previously-detected faces in the frame, the tracker updates their locations. The frame is then placed in a *frame revisit buffer (FRB)* that adds a minimum delay before denaturing. Since the FRB is passive, its presence only adds latency to the pipeline without affecting its throughput.

During the wait in the FRB, the tracker has an opportunity to correct tracking and detection errors by using new information received asynchronously from the detection module. This information may apply to any of the frames in the FRB. The tracking algorithm is applied to correct frames in the FRB using both forward tracking and backward tracking. The reason for backward tracking is that the initial detection of a face in a frame may be presaged by many previous frames in which a superior detector (such a human observer) may be able to detect that face. This would correspond to a privacy leak. The finite size of the FRB limits the amount of correction possible. In our experience, a size of 30 frames has proved adequate. In other words, the denatured stream is delayed by one second to allow for correction. Merging updates from asynchronous face detection and recognition into tracking requires careful bookkeeping. Since the location of a face in a frame may change due to physical motion, we use a unique id to label each bounding box and to track its progress.

The speed of the tracker and dispatcher determine the achievable throughput of the pipeline. The speed of the detection module does not affect throughput — it only affects the ability to eliminate privacy leaks. Very slow detection increases the chances of an undetected privacy leak on a frame that is released from the FRB.

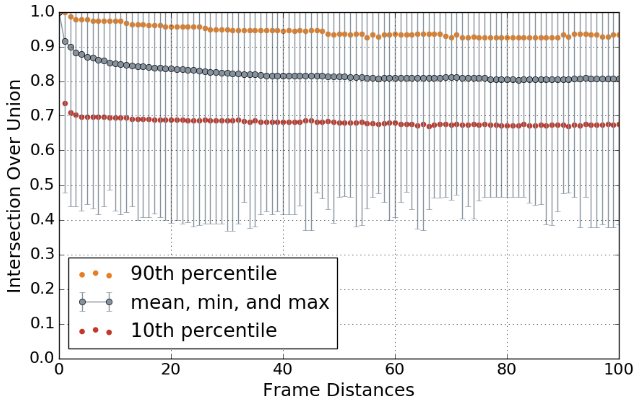


Figure 13: Tracking Accuracy

4.4 RTFace: Measured Accuracy and Speed

The YouTube Celebrities Face Tracking and Recognition Dataset [19] has been used by the computer vision community for evaluating tracking accuracy. From this dataset, we use a 10-video subset of distinct celebrities as the benchmark in our evaluation. The subset chosen corresponds to those videos on which the Dlib face detector is most accurate (i.e., misses the least number of faces).

We treat the bounding boxes of faces discovered by Dlib on a frame as the "ground truth" for that frame. The metric used by the computer vision community to quantify tracking accuracy is Intersection over Union (IOU) of a predicted bounding box and its ground truth bounding box [28]. A face is deemed to have been tracked accurately if its IOU value is 0.5 or better.

For each frame with faces detected, we create trackers for all faces and let them run until the end of the video. Figure 13 presents the tracking accuracy of our prototype as frame distance increases. These results confirm that the mean IOU remains well above 0.5 even after 100 frames of tracking. Even the 10-percentile level remains well above 0.5. These results confirm that tracking is a robust and safe technique for improving the speed of denaturing.

We use the same benchmark to test overall accuracy and speed, relative to a detection-only implementation. Using the average of three runs, our prototype correctly recalls and classifies 92% of detected faces at a speed of 89 fps. Without tracking, the maximum sustainable frame rate drops to 12 fps. Since the resolution of the YouTube Celebrities dataset is only 320x240 or lower, this difference is understated. With higher resolutions, the speedup of using a tracking-based pipeline would be higher.

5 Reducing Privacy Leaks

Even today, face detection by humans is so much better than software, that a few frames with a human-recognizable face can slip by before RTFace detects a face. Note that this is a problem even with the simple denaturing pipeline of Figure 11.

5.1 Mitigation Strategies

Tracking and the use of an FRB together help to reduce leaks. A face detected in a later frame is tracked backwards through the frames in the FRB. This reduces privacy leaks at the cost of increased false positives (i.e., a non-existent face is obscured). We use the following additional techniques to reduce privacy leaks.

Dispatch criteria: The dispatcher in Figure 12 uses two criteria to decide whether to send a frame for asynchronous detection and recognition. These are the tracker's confidence score c, and the number of frames t since last detection. When the confidence score c is below a predetermined threshold i, the tracker has drifted too much. Hence, detection and recognition need to be run in order to correct the tracker. Independently, running face detection periodically reduces tracking error and also reduces the latency for discovering new faces that enter camera view or old faces that disapper. Our experiments also indicate that image-dependent factors such as the blurriness of an image can affect tracking performance.

Detection bias: Denaturing errors should be biased towards too much privacy rather than too little. False positives (detection of a non-existent face, or non-recognition of a face that may be left unobscured according to current policy) are more benign than false negatives (missing a face and hence failing to obscure it, assuming an opt-in policy). We therefore tune the face detector in Figure 12 to a relatively low threshold for face detection. This may result in many non-faces being presented to OpenFace for recognition, but this is harmless because the result is typically "unknown person."

Rapid update tracking: Though detection and recognition no longer reduce throughput, their speed does influence the incidence of privacy leaks. We can further reduce privacy leaks by decoupling detection from recognition, and by obscuring a new face as soon as it is detected. After recognition, it may become apparent that the face should have been left unobscured. This correction can be applied to all the frames in the FRB, but it may be too late for some earlier frames that have already been released. Although this error is conservative (i.e., no faces are revealed that should not have been), we would like reduce its incidence. To this end, we use *perceptual hashing* [21, 22] as a fast but weak approximation to face recognition. If the perceptual hash of the new bounding box is very similar to that of an earlier one, we temporarily use the identity of the latter face. If there are no matches, the identity is assumed to be an unknown face. Once recognition completes, the identity of the face in the new bounding box is known with much higher confidence. Comparison using a perceptual hash is safe: it may indicate that two faces are different when they are really the same, but the reverse is extremely unlikely. We are opportunistically exploiting a form of temporal locality here: over a short duration, a person's image may appear identical across several frames (same lighting conditions, same pose, same makeup, same emotional state, etc.) – perceptual hashing is a computationally cheap mechanism to catch these instances.

5.2 Non-Frontal Faces

In contrast to humans, who excel at detecting and recognizing faces at all angles, software face detection and recognition perform best in frontal view. As a result, it is possible for profile or near-profile faces to be missed by the denaturing pipeline. The 2D affine transform shown in Figure 4 helps with recognition, but not with the prior step of detection. The use of an FRB with backward tracking, helps in some instances of this problem. However, if a face remains in profile for many frames before turning frontal, its detection may come too late to correct the earliest frames. Improving the accuracy of profile face detectors would be valuable.

6 Controlled Reversal of Denaturing

There are occasions when public safety or other societally important considerations override privacy considerations. For example, the crucial clues in identifying the perpetrators of the 2013 Boston Marathon terrorist attack were obtained from surveillance cameras. If a denaturing system such as RTFace had been used in that setting, it would have been necessary to reverse the denaturing of the archived video. Most democratic societies have well-established procedures for obtaining policy exceptions (e.g., search warrants) in such situations. In the context of denaturing, there are many important policy questions pertaining to who can request reversal of denaturing, who can authorize it, how broad the authorization should be, how freely the original data can be released after reversal, etc. A reversal mechanism is needed that can support a plausible range of policies.

Figures 14 illustrates such a mechanism, with the numbered arrows referring to calls in the API shown in Figure 15. All policy-related aspects are encapsulated into the system-wide entity labeled "Trusted External Authority" (TEA). All network communication takes place over secure, authenticated TCP connections using standard TLS technology. Encryption of obscured bits in the mediator is based on private key encryption (AES-128) using a random key generated and escrowed by the TEA. Each key has a finite lifetime, whose duration is determined by TEA policy. In practice, we expect lifetimes of a few hours to be likely. As a key's expiry time approaches, the mediator requests the TEA for a new key using the first call in Figure 15. A mediator does not retain old keys. If cloudlet security is compromised, only the most recently denatured video is at risk of exposure. The keys needed to reverse denaturing of older archived video is only available from the TEA.

If a need arises to reverse the denaturing on old video, a *policy exception credential* is first obtained for the desired video stream (identified by the ID of its mediator) and time period. We leave unspecified the format of this credential and the procedure to obtain it — these are clearly important specifications that will need to be completed in any deployment of this system. Using the second call in Figure 15, an external requestor can present the policy exception credential to the TEA. Using the video stream id and timestamp information authorized by this credential, the TEA retrieves relevant escrowed keys. It then makes a series of ReverseDenature calls (third call in Figure 15) to the specified mediator. The mediator uses timestamp information to locate the relevant encrypted bounding boxes, decrypts them using the key provided to obtain bit-exact copies of the removed regions, and then applies them to the relevant segment

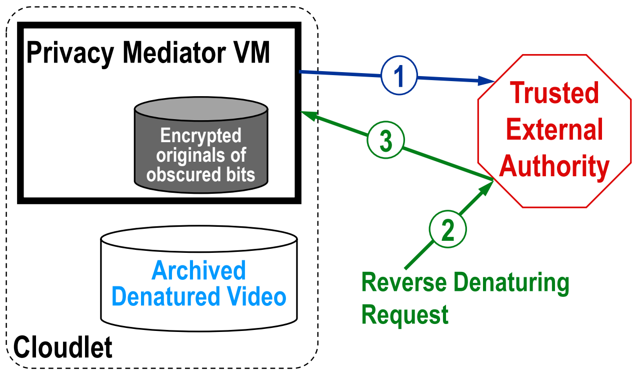

Privacy Mediator VM
Encrypted originals of obscured bits
Archived Denatured Video
Cloudlet
1
3
2
Trusted External Authority
Reverse Denaturing Request

Figure 14: **Policy-Directed Reversal of Denaturing**

TEA = Trusted External Authority

From mediator to TEA:
① $GetNewKey\ (StartTime) \rightarrow (NewKey, ExpiryTime)$

From external requestor to TEA:
② $GetOriginalVideo\ (MediatorID, StartTime, EndTime,$
$PolicyExceptionCredential) \rightarrow (OriginalVideo)$

From TEA to mediator:
③ $ReverseDenature\ (Key, StartTime, EndTime) \rightarrow$
$(OriginalVideo)$

Figure 15: **API to Enable Reversal of Denaturing**

of archived denatured video. This lets the mediator reconstruct the original video segment modulo any compression artifacts. If the archived video segment or encrypted bounding boxes have been purged by the storage retention policy of the cloudlet, the reverse denaturing request will fail. The TEA reassembles the segments returned by the mediator into the original video for the specified time period, and returns it to the requestor.

7 IoT Service Deployment at Enterprise Scale

The decreasing costs of cameras and computation devices have enabled large-scale deployments of IoT cameras in places such as schools, company workspaces, and public streets. Although live analysis of video streams from such always-on cameras can provide many societal benefits, legitimate concerns over privacy and misuse of video data pose a significant obstacle to acceptance of such systems. Based on OpenFace and RTFace, we propose an architecture for a large-scale, privacy-aware IoT deployment that allows real-time analytics while incorporating privacy mediation. We have implemented a prototype of our proposed design.

Figure 16 demonstrates our design with four example cameras. It shows all components and communications for the leftmost camera. Solid arrows in the figure represent the data (blue solid arrows) and control (gold solid arrows) flows when a user comes into the range of a camera. Dashed arrows represent the offline registration steps. We assume RGB static and moving cameras. Techniques including image stabilization and deblurring can be used to mitigate the negative influences of a moving camera. The main components and operations are detailed below.

Privacy Mediator VM: In our system, there is one privacy mediator VM associated with each camera. This VM's primary task is to perform selective denaturing of the video before it is stored or made available for analytics, and implements the architecture as shown in Figure 1 using RTFace as the denaturing algorithm. Privacy mediator VMs run on cloudlets located near the cameras, typically connected by high-bandwidth, wired networks. A cloudlet may host multiple privacy mediator VMs. Furthermore, depending on the size of the deployment, there can be multiple cloudlets each serving multiple cameras in an area.

Centralized Information Repository: Our system implements a centralized database, called the IoT Resource Registry (IRR) [33], to track the state of the system components. It contains the locations and status of each camera and associated privacy mediator VM in the system. The privacy mediator VMs register themselves to the IRR when they are deployed, and report regularly on the status of the associated camera streams. This information can be used for maintenance and to restart VMs or cameras in case of failures. Furthermore, the IRR keeps a directory of the various analytic services available on each stream, as well as the applicable privacy policies and details about any available privacy settings associated with each stream. In the context of this paper, such a policy specification would for instance indicate whether a video stream supports a setting that enables an individual to opt-out of face denaturing, and information on how to adjust the settings at the privacy mediator VM. Note that the IRR itself does not handle, process, or store any video data, nor the users' privacy settings; rather it stores only the metadata used to describe and control the distributed set of components in the system. In our prototype, the IRR also plays the role of the TEA in Figure 14. In a full-scale deployment, this role would likely be separated out into a separate component.

Camera Setup: Cameras in this system are assumed to connect through a high-bandwidth, wired network. Each is configured to continuously generate and send a compressed video stream to only its associated privacy mediator VM. A bluetooth low energy (BLE) beacon is co-located with each camera to broadcast camera-specific information. This serves two purposes. First, it acts as an electronic sign informing individuals in the vicinity that video is being captured in the area. This beacon can be received by any modern smartphone or wearable device that supports BLE. Secondly, the beacon includes information about the camera and the IRR responsible for the particular camera network. This acts as a simple discovery mechanism independent of any external coordination or accurate localization, and allows many different camera networks to operate in overlapping regions. Finally, the range of the BLE beacon is typically a few tens of meters, so it is similar to the viewing range of typical outdoor surveillance cameras, and can serve as a means to determine which cameras may observe the user.

IoT Assistant (IoTA): Individuals who wish to set and manage their privacy settings need to run a client-side agent, called the IoT Assistant (IoTA) [33], on a personal device that they carry. Our current implementation is Android-based, and can run on many smartphones as well as wearable devices. The IoTA receives the camera BLE beacons and communicates to the appropriate IRR to discover available privacy settings and the corresponding mediator VMs. It can then automatically set the user's privacy settings at

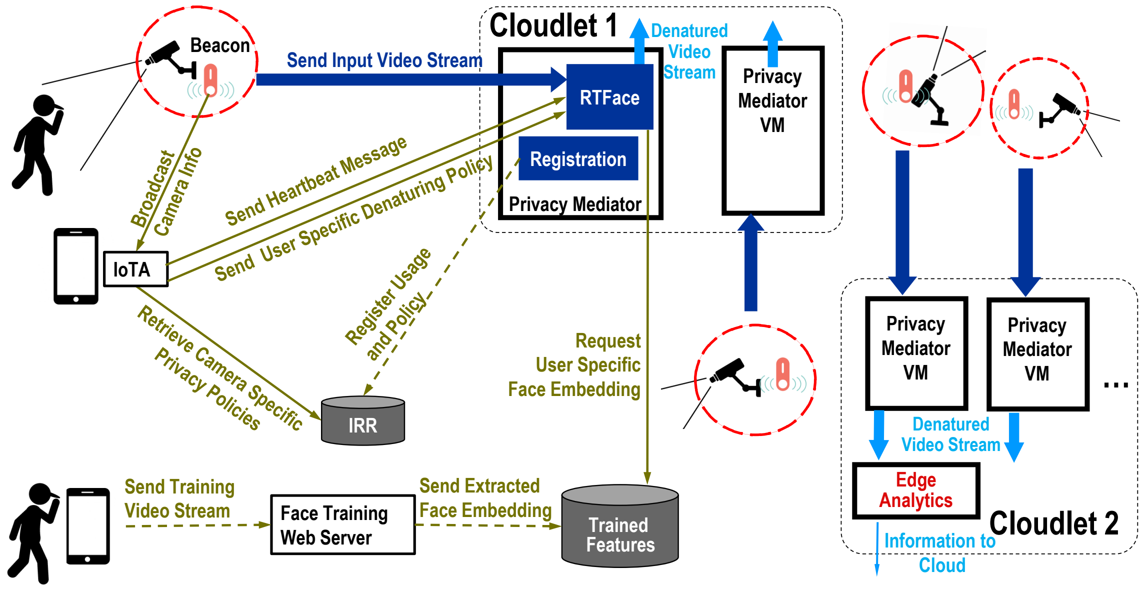
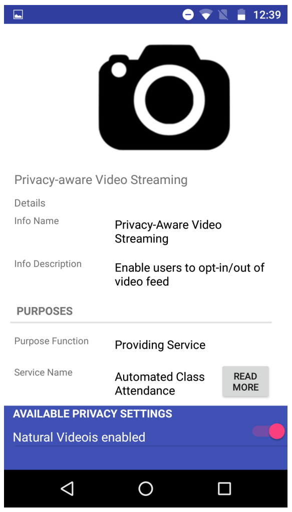


Figure 16: Scalable Privacy Mediation Architecture


12:39

Privacy-aware Video Streaming

Details

Info Name	Privacy-Aware Video Streaming
Info Description	Enable users to opt-in/out of video feed

PURPOSES

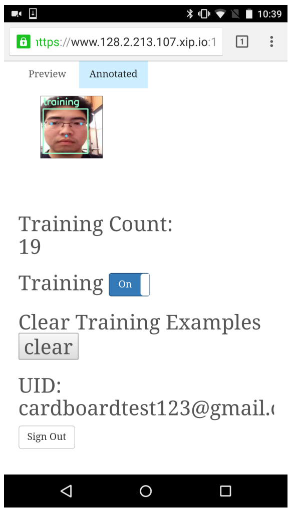
Purpose Function	Providing Service
Service Name	Automated Class Attendance

READ MORE

AVAILABLE PRIVACY SETTINGS

Natural Videois enabled

Figure 17: IoTA App Screenshot


10:39

ttps://www.128.2.213.107.xip.io:1

Preview Annotated

Training

Training Count:
19

Training On

Clear Training Examples
clear

UID:
cardboardtest123@gmail.

Sign Out

Figure 18: Training Website

the privacy mediator for the camera. Users can selectively allow their faces to be visible in the particular camera stream or camera network to benefit from live video analytics. In addition, if desired, the IoTA can also alert the user when near active cameras. Thus, it serves to inform the user of video capture and provide a means of control. Note that the use of the IoTA is optional – the default policy of the privacy mediator VMs is to blur all faces, thereby preserving privacy of all individuals. So by using the IoTA, users "opt-in" to the selective denaturing of their faces based on their specified settings. A screenshot of IoTA is shown in Figure 17.

Runtime Workflow: When a user comes into the proximity of a video camera, the IoTA receives the camera BLE beacon that identifies the camera, camera network, and IRR. The IoTA retrieves camera- and network-specific privacy policies from IRR. It then selectively alerts the user if desired, or if the user has not set privacy preferences that cover the particular camera. User-specified privacy policies are then sent to the privacy mediator associated with the camera. In addition, the IoTA periodically sends heartbeat messages to indicate the continued presence of the user near the camera. At the privacy mediator, these heartbeat messages are used to narrow down the search space of OpenFace face recognition. Upon receiving the first heartbeat message of a new user, the privacy mediator VM retrieves and caches the 128-dimensional feature vectors for the user's face from a global database. It then quickly trains a linear SVM using the feature vectors of all nearby users, as indicated by recent heartbeat messages. As discussed in Figure 9, training a linear SVM takes less than 0.1 seconds for 50 people. After a sufficient timeout interval, feature vectors that have not had a heartbeat message are removed from the set used for training, thereby keeping the SVM small and manageable. The privacy mediator VM performs denaturing on the video stream using RTFace as described in Section 4. The denatured video is then presented to video analytics VMs on the cloudlet.

Training: Before a user can specify a privacy preference, his face needs to be recognizable by the system. OpenFace achieves the accuracy shown in Figure 8 by using just 20 training images per person. At a frame rate of 30 FPS, a few seconds of video at different face angles can provide more than enough training data.

We have implemented a prototype face training website to which users can stream a short video for training, as shown in Figure 18. The website front-end accesses the device camera using HTML5 support in the browser, and transmits captured frames through the secure Websocket protocol. For each face detected in training images, the training web server extracts a 128-dimensional feature vector using OpenFace and stores them into a global database. At runtime, these feature vectors are retrieved by the privacy mediator VM when a user enters into the range of a camera.

Note that a complete deployed system for training and storing feature vectors will need to address a few additional security concerns. Most importantly, requests for a feature vector from an IoTA need to be properly authenticated and authorized. This can be implemented using standard SSL/TLS based user authentication and access control lists. For example, this could allow a user to set his own privacy settings and those of his children as well. In addition, at training time, there needs to be some external means of ensuring that the video clip uploaded is really that of the user. This is a standard bootstrapping requirement of biometric authentication mechanisms.

8 Scalability and Design Choices

Scalability is critical for handling a large number of IoT cameras. Under the constraints of network and computational resources, the system needs to scale well with respect to the number of users, as well as the quantity and quality of cameras.

The use of cloudlets and the design of RTFace both contribute to system scalability. Cloudlets, sitting at the edge of the Internet, reduce the amount of data that needs to be sent to the cloud for analysis. RTFace significantly reduces the computation needed for denaturing a video stream by combining face recognition and face tracking. In this section, we first explore the scalability with respect to users. Then, based on experimental data, we offer design recommendations for the computational and networking requirements of cloudlets as a function of the quantity and quality of cameras.

8.1 Number of Users

Denaturing faces from a large pool of users poses challenges to both the accuracy and speed of the system. Figure 8 demonstrates that OpenFace performs reasonably well for a search space of 100 people. For a large scale camera deployment, such as a city-wide deployment, the entire set of individuals to be recognized from can be much larger. However, despite the large user population in total, the number of faces a single camera would capture at a particular time is usually small. Therefore, as described in Section 7, we use a beaconing mechanism to limit the search space for each camera. Only the face feature vectors of individuals within the beacon radius are included in the camera's SVM classifier. Furthermore, OpenFace uses linear SVMs of low complexity. Although execution time is typically quadratic to the number of classes, as shown in Figure 10, the increase in prediction time from 10 people to 100 people is small relative to the execution time of the neural network. This beaconing mechanism does require the SVMs to be retrained as users enter and leave the range of the camera. However, as we see in Figure 9, this requires only a few tenths of a second even for 100 users.

First, we explore the maximum processing speed when allocating one core per camera. We use 2300 frames from a 1920x1080 (HD) video that contains multiple faces as test input. There are twenty people in the system. At most 5 people are present in a single frame. To focus on processing speed, we load all frames into memory first, and then record the compute time to process them. Figure 19 shows the average processing time different methods can achieve when running on a 1-core 3 GB RAM VM. The figure compares RTFace

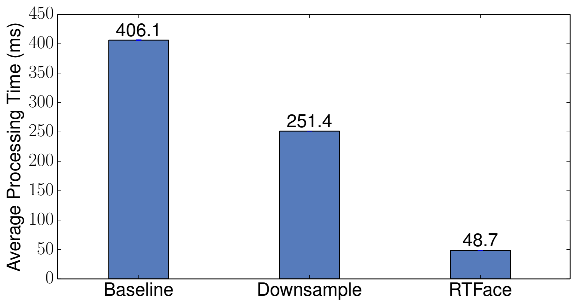


Figure 19: Single-core Processing Time on Cloudlet

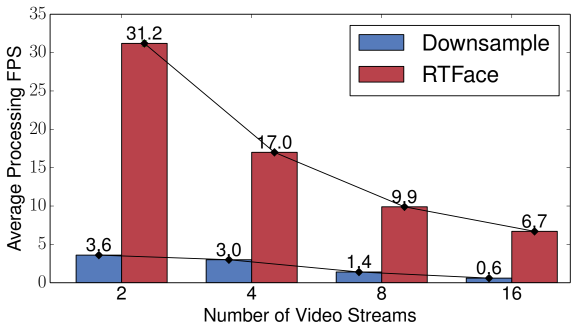


Figure 20: Multi-stream Frame Rate on 4-core Cloudlet

Resolution (pixel)	RTFace Processing Time (ms)
1920x1080 (HD)	48.7 (0.01)
3840x2160 (4K)	64.6 (0.01)

Figures in parentheses are standard deviations across 3 runs.

Table 3: Impact of Frame Resolution on 1-core Processing Speed

Resolution (pixel)	Decoding Time (ms)
1920x1080 (HD)	20.4 (1.8)
3840x2160 (4K)	85.4 (1.9)

Figures in parentheses are standard deviations across 3 runs.

Table 4: Impact of Frame Resolution on 1-core Decoding Speed

with Baseline (running recognition on every frame) and Downsample (downsampling frames before detection). Downsampling high resolution frames is often used to improve speed at the cost of accuracy. In our Downsample experiments, we downsample frames to 480x360, a low resolution at which the face detector still detects most faces. Figure 19 shows that RTFace can run at 5 times the speed of Downsample, and 8 times that of Baseline. The 48.7 ms processing time for RTFace indicates it can sustain a speed of slightly over 20 FPS at HD resolution, if we allocate only one core per camera.

8.2 Quantity and Quality of Cameras

Next, we investigate how RTFace performs in a multi-core, multi-stream scenario. Using the same 4-core cloudlet described in Section 4.4, we concurrently process 2, 4, 8, and 16 video streams. The input video stream is the same as that described above. Figure 20 compares the average frame rate per stream as the number of video

	Video Resolution			
Target	1080p (HD)		2160p (4K)	
Network Utilization	Streams	Cores	Streams	Cores
10%	20	42	4	18
30%	60	126	12	54
70%	140	294	28	126
100%	200	420	40	180

(a) Software Decoding

	Video Resolution			
Target	1080p (HD)		2160p (4K)	
Network Utilization	Streams	Cores	Streams	Cores
10%	20	32	4	10
30%	60	96	12	29
70%	140	224	28	68
100%	200	320	40	96

(b) Hardware Decoding

Note that a dense deployment of video cameras could result in 100 cameras or more in a large multi-storey office building. In such a building, these numbers suggest a deployment scenario with multiple LAN segments and a cloudlet per segment.

Table 5: Network and Cloudlet Sizing for 1 Gbps LAN

streams grows. RTFace consistently performs an order of magnitude better than running detection and recognition on every frame, even if the frames are downsampled. Other than the two-stream case, which did not fully utilize all cores, the frame rate achieved are consistent with a linear extrapolation from the single core data.

Video resolution can also significantly impact compute time. Table 3 compares RTFace's processing speed on a single core for 4K and HD images. Higher resolutions affect computation times for face detection in particular. On the other hand, since recognition is applied to faces scaled to a standard resolution, it does not affect recognition. Thus, RTFace takes longer on higher resolutions but not as much as one might exepct.

As cameras produce compressed video streams, the complete video pipeline involves both decoding and denaturing. Table 4 presents the software decoding times for different resolutions on a single core. Together with data shown in Table 3, a complete pipeline for HD images would take 69.1 ms. Using the approximate linear scaling inferred from Figure 20, an HD video stream needs 2.1 cores to achieve 30 FPS. A 4K video stream needs 4.5 cores. Hardware decoders can be used to reduce the software workload. Data from hardware decoders in standard desktop platforms [13], show that up to 25 HD streams or 4 4K streams at 25 FPS can be decoded on a quad-core Intel® Core™ i7 processor at 40% CPU utilization. Therefore, for 30 FPS video streams, with hardware decoders, an HD stream needs 0.1 core and a 4K stream needs 0.5 cores for decoding. The complete denaturing pipeline would require 1.6 cores per HD stream, or 2.4 cores per 4K stream.

8.3 Balanced LAN and Cloudlet Design

In addition to computation as a bottleneck, continuously streaming videos from many cameras also requires high bandwidth in the communication network. With a cloud-based centralized design, both the ingress bandwidth into the cloud infrastracture and the metro-area aggregation network can become bottlenecks. By July 2015, 6.7 hours of video were being uploaded to YouTube each second [36]. The cumulative upload bandwidth of these videos is equivalent to only 24,120 cameras simultaneously streaming HD videos. This is two orders of magnitude smaller than the estimated 5.9 million security cameras in the UK in 2013 [3]. Netflix recommends 25 Mbps bandwidth for streaming 4K videos [23]. At that data rate, 100 buildings with 100 cameras each will need a 250 Gbps aggregation network to a metro-wide cloudlet. More decentralization, using a cloudlet in each building or cluster of buildings, can significantly reduce the demand on a metropolitan area network. With denaturing and video analytics occurring close to cameras, only a small amount of distilled information needs to be transmitted further.

In designing the LAN segments for a single building or small cluster of buildings, a key parameter is the level of network utilization. On a contention-based network such as Ethernet, high planned utilization is not recommended. The performance of non-video traffic (such as Web traffic, interactive traffic, and VoIP) are likely to suffer as network utilization rises due to continuous video transmission.

Table 5 illustrates the relationship between targeted network utilization, camera resolution, and the number of cameras per 1 Gbps LAN segment. The number of cameras, the resolution, and whether hardware decoding is used, together determine the number of cores needed in the cloudlet associated with that LAN segment. For example, a LAN segment with a target utilization of 10% can support up to 20 HD cameras or 4 4K cameras. The cloudlet on that LAN segment would need 42 cores for the HD cameras, or 18 cores for the 4K cameras. As the target LAN utilization rises, so does the number of cameras and cores per cloudlet. The figures for 100% utilization are not achievable in practice, and are only shown for reference. Of course, the dispersal of cloudlets complicates system management. This is an important problem that will have to be addressed through the development of appropriate system management tools and techniques.

9 Conclusion

This is the first work to show that enforcement of privacy policies based on recognition of individuals is feasible in real time on streamed video. We present OpenFace, an open-source face recognizer whose accuracy approaches that of the best available proprietary recognizers. We show that we can combine OpenFace with face tracking to maintain high accuracy yet achieve full frame rate speeds. We describe a privacy mediator that is fast, accurate, and can reverse denaturing in a principled and secure way to accommodate policy exceptions. Finally, we show how to put these elements together into a scalable privacy-aware IoT architecture that enables live video analytics across a large number of cameras.

Real-time information from cameras has the potential to greatly benefit society, but the rapid proliferation of video cameras in public spaces raises very real privacy concerns. Implementing systems that empower individuals to take charge of their privacy will facilitate the acceptance of widespread cameras and real-time video analytics. The denaturing mechanism presented here takes a first step towards this goal. Although our focus has been on face recognition, our work can be easily generalized to other objects. Beyond video, the broad principles and implementation strategies developed here are also relevant to any IoT sensing modality with a high data rate.

Acknowledgements

We thank Nigel Davies and Nina Taft for many illuminating discussions on the concept of privacy mediators in IoT. This research was partly funded through the National Science Foundation (NSF) under grant number CNS-1518865 and SBE-1513957, as well as by DARPA and the Air Force Research Laboratory, under agreement number FA8750-15-2-0277. Additional support was provided by Intel, Google, Vodafone, Deutsche Telekom, Verizon, Crown Castle, NVIDIA, and the Conklin Kistler family fund. Brandon Amos is supported by the NSF Graduate Research Fellowship Program under Grant No. DGE1252522. Any opinions, findings, conclusions or recommendations expressed in this material are those of the authors and should not be attributed to their employers or funding sources.

REFERENCES

[1] Paarijaat Aditya, Rijurekha Sen, Peter Druschel, Seong Joon Oh, Rodrigo Benenson, Mario Fritz, Bernt Schiele, Bobby Bhattacharjee, and Tong Tong Wu. 2016. I-pic: A platform for privacy-compliant image capture. In *Proc. of ACM MobiSys*.

[2] Timo Ahonen, Abdenour Hadid, and Matti Pietikäinen. 2006. Face Description with Local Binary Patterns: Application to Face Recognition. *IEEE Trans. on PAMI* 28, 12 (December 2006).

[3] David Barrett. 2013. One surveillance camera for every 11 people in Britain, says CCTV survey. *Daily Telegraph* (July 10, 2013).

[4] Peter N Belhumeur, João P Hespanha, and David J Kriegman. 1997. Eigenfaces vs. fisherfaces: Recognition using class specific linear projection. *IEEE Trans. on PAMI* 19, 7 (July 1997).

[5] Cheng Bo, Guobin Shen, Jie Liu, Xiang-Yang Li, YongGuang Zhang, and Feng Zhao. 2014. Privacy. tag: Privacy concern expressed and respected. In *Proc. of ACM SenSys*.

[6] Tiffany Yu-Han Chen, Lenin Ravindranath, Shuo Deng, Paramvir Bahl, and Hari Balakrishnan. 2015. Glimpse: Continuous, Real-Time Object Recognition on Mobile Devices. In *Proc. of ACM SenSys*.

[7] Ronan Collobert, Koray Kavukcuoglu, and Clément Farabet. 2011. Torch7: A matlab-like environment for machine learning. In *BigLearn, NIPS Workshop*.

[8] Martin Danelljan, Gustav Häger, Fahad Khan, and Michael Felsberg. 2014. Accurate scale estimation for robust visual tracking. In *Proc. of the British Machine Vision Conference*.

[9] Nigel Davies, Nina Taft, Mahadev Satyanarayanan, Sarah Clinch, and Brandon Amos. 2016. Privacy Mediators: Helping IoT Cross the Chasm. In *Proc. of ACM HotMobile 2016*.

[10] Debate.org. 2017. Are video surveillance cameras in public places a good idea? http://debate.org. (2017).

[11] Ralph Gross, Latanya Sweeney, Fernando De la Torre, and Simon Baker. 2006. Model-based face de-identification. In *Proc. of IEEE CVPR Workshop*.

[12] Gary B Huang, Manu Ramesh, Tamara Berg, and Erik Learned-Miller. 2007. *Labeled faces in the wild: A database for studying face recognition in unconstrained environments*. Technical Report 07-49. University of Massachusetts, Amherst.

[13] Intel. 2016. Milestone Leverages Intel Processors with Intel Quick Sync Video to Create Breakthrough Capabilities for Video Surveillance and Monitoring. http://www.intel.com/content/dam/www/public/us/en/documents/white-papers/milestone-software-quick-sync-video-surveillance-monitoring-white-paper.pdf. (2016).

[14] Rabia Jafri and Hamid Arabnia. 2009. A Survey of Face Recognition Techniques. *Journal of Information Processing Systems* 5, 2 (2009), 41–68.

[15] Suman Jana, Arvind Narayanan, and Vitaly Shmatikov. 2013. A Scanner Darkly: Protecting user privacy from perceptual applications. In *IEEE Symposium on Security and Privacy*.

[16] Tony S Jebara. 1995. *3D pose estimation and normalization for face recognition*. Ph.D. Dissertation. McGill University.

[17] Michal Kampf, Israel Nachson, and Harvey Babkoff. 2002. A serial test of the laterality of familiar face recognition. *Brain and Cognition* 50, 1 (2002), 35–50.

[18] Takeo Kanade. 1973. *Picture processing system by computer complex and recognition of human faces*. Ph.D. Dissertation. Kyoto University.

[19] Minyoung Kim, Sanjiv Kumar, Vladimir Pavlovic, and Henry Rowley. 2008. Face tracking and recognition with visual constraints in real-world videos. In *Proc. of IEEE CVPR*.

[20] Davis E King. 2009. Dlib-ml: A machine learning toolkit. *The Journal of Machine Learning Research* 10 (2009), 1755–1758.

[21] Suleyman Serdar Kozat, Ramarathnam Venkatesan, and Mehmet Kivanç Mihçak. 2004. Robust perceptual image hashing via matrix invariants. In *Proc. of IEEE ICIP*. 3443–3446.

[22] Vishal Monga and Brian L Evans. 2006. Perceptual image hashing via feature points: performance evaluation and tradeoffs. *IEEE Trans. on Image Processing* 15, 11 (2006), 3452–3465.

[23] Netflix. 2017. Internet Connection Speed Recommendations. https://help.netflix.com/en/node/306. (2017).

[24] Elaine Newton, Latanya Sweeney, and Bradley Malin. 2005. Preserving privacy by de-identifying face images. *IEEE transactions on Knowledge and Data Engineering* 17, 2 (2005), 232–243.

[25] Hong-Wei Ng and Stefan Winkler. 2014. A data-driven approach to cleaning large face datasets. In *Proc. of IEEE ICIP*. 343–347.

[26] Meike Ramon, Stephanie Caharel, and Bruno Rossion. 2011. The speed of recognition of personally familiar faces. *Perception* 40, 4 (2011), 437–449.

[27] Nisarg Raval, Animesh Srivastava, Ali Razeen, Kiron Lebeck, Ashwin Machanavajjhala, and Landon P. Cox. 2016. What you mark is what apps see. In *Proc. of ACM MobiSys*.

[28] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, AndrejKarpathy, Aditya Khosla, Michael Bernstein, and others. 2015. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision* 115, 3 (2015), 211–252.

[29] Florian Schroff, Dmitry Kalenichenko, and James Philbin. 2015. Facenet: A unified embedding for face recognition and clustering. In *Proc. of IEEE CVPR*.

[30] Pieter Simoens, Yu Xiao, Padmanablan Pillai, Zhuo Chen, Kiryong Ha, and Mahadev Satyanarayanan. 2013. Scalable Crowd-Sourcing of Video from Mobile Devices. In *Proc. of ACM MobiSys*.

[31] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2015. Going deeper with convolutions. In *Proc. of IEEE CVPR*.

[32] Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf. 2014. Deepface: Closing the gap to human-level performance in face verification. In *Proc. of IEEE CVPR*.

[33] Carnegie Mellon University Personalized Privacy Assistant Team. 2017. Personalized Privacy Assistant Project. http://privacyassistant.org/. (2017).

[34] Matthew Turk and Alex Pentland. 1991. Eigenfaces for recognition. *Journal of cognitive neuroscience* 3, 1 (1991), 71–86.

[35] Dong Yi, Zhen Lei, Shengcai Liao, and Stan Z Li. 2014. Learning Face Representation from Scratch. *arXiv preprint arXiv:1411.7923* (2014).

[36] YouTube and Tubefilter. 2017. Hours of video uploaded to YouTube every minute as of July 2015. https://www.statista.com/statistics/259477/hours-of-video-uploaded-to-youtube-every-minute. (2017).