Scalable Crowd-Sourcing of Video from Mobile Devices

Pieter Simoens[†], Yu Xiao[‡], Padmanabhan Pillai[•], Zhuo Chen,
Kiryong Ha, Mahadev Satyanarayanan

December 2012
CMU-CS-12-147

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

[†]Carnegie Mellon University and Ghent University
[‡]Carnegie Mellon University and Aalto University
[•]Intel Labs

Abstract

We propose a scalable Internet system for continuous collection of crowd-sourced video from devices such as Google Glass. Our hybrid cloud architecture for this system is effectively a CDN in reverse. It achieves scalability by decentralizing the cloud computing infrastructure using VM-based cloudlets. Based on time, location and content, privacy sensitive information is automatically removed from the video. This process, which we refer to as denaturing, is executed in a user-specific Virtual Machine (VM) on the cloudlet. Users can perform content-based searches on the total catalog of denatured videos. Our experiments reveal the bottlenecks for video upload, denaturing, indexing and content-based search and provide valuable insight on how parameters such as frame rate and resolution impact the system scalability.

This research was supported by the National Science Foundation (NSF) under grant numbers CNS-0833882 and IIS-1065336, by an Intel Science and Technology Center grant, by the Department of Defense (DoD) under Contract No. FA8721-05-C-0003 for the operation of the Software Engineering Institute (SEI), a federally funded research and development center, and by the Academy of Finland under grant number 253860. Any opinions, findings, conclusions or recommendations expressed in this material are those of the authors and do not necessarily represent the views of the NSF, Intel, DoD, SEI, Academy of Finland, University of Ghent, Aalto University, or Carnegie Mellon University. This material has been approved for public release and unlimited distribution except as restricted by copyright.

Keywords: mobile computing, cloud computing, Google Glass, headup display, HUD, first-person vision system, FPV, sensing, smartphones, virtual machines, system architecture, cloudlets, denaturing, face detection, face recognition, content-based search, image search, early discard, Diamond

1 Introduction

Head-up displays (HUDs) have been used for more than a decade in military and maintenance tasks. In these challenging domains, the primary design goal was to provide hands-free user assistance. Comfort and style were only second order design considerations, hence limiting mainstream adoption by the public. Today, we are on the verge of a disruptive generation of HUDs that target daily use by the general public. Google Glass is the most well-known example, but others are also being developed. For example, the NSF Quality of Life Technology Center has created a first-person vision system for disabled veterans [19].

When equiped with a front-end camera, HUDs enable *near-effortless capture of first-person viewpoint video*. Recording a video will merely require you to press a button on the shank of your glasses, instead of taking your smartphone out of your pocket and performing a number of touch screen interactions. Easy video capture will greatly increase the number of videos shared with the world (e.g. by uploading to YouTube). A good example of the videos we might expect was recently provided by designer Diane von Furstenberg, who wore Google Glass to capture and share her experience at the Spring 2013 Fashion Show in New York City [8]. As of today, YouTube already contains thousands of first-person videos captured with less elegant commercial products such as the Contour+2 [6]. Integrating video capture with correlated sensor information such as gaze tracking, audio, geolocation, acceleration, and biodata (e.g., heartrate) is only a matter of time.

In this paper, we focus on the gathering, cataloging and access of first-person video from many contributors. By automatic tagging of this rich data collection, and by enabling deep content-based search of any subset of that data, we create a valuable public resource much like the Web itself. Even simple uses of this resource can transform today's applications in imaginative ways:

Imagine Google Street View Live, allowing you to see live video that was recently captured by someone walking down that street. You see the same things that the user saw, and linger over the scenes that caught the user's attention. You are effectively reliving his walk down the street. Parts of some scenes are blurred because the user did not want to share them with you. You can click on "Winter", and the video stream changes to one captured by a user walking down that street last winter. You can click on "multiview" and see the street from the viewpoints of multiple users simultaneously.

Figure 1 speculates on many other use cases that could be enabled by a large-scale searchable video repository.

Enabling this futuristic vision poses technical challenges at many levels. It also creates many policy-related challenges, such as those pertaining to privacy and user incentives. Our focus is on technical issues, but policy issues are relevant to the extent that they require supporting mechanisms in the implementation. This paper makes the following contributions:

- It describes the potential benefits of crowd-sourced first-person video collection, offers an incentive model for such a system, identifies the key challenges in realizing such a capability at global scale, and derives the technical requirements for a viable implementation.
- It presents *GigaSight,* a hybrid cloud architecture for scalable crowd-sourcing of video from mobile devices. This architecture is effectively a Content Distribution Network (CDN) in reverse, and achieves scalability by decentralizing the cloud computing infrastructure using VM-based cloudlets at the edges of the Internet.
- It introduces the concept of *denaturing* for privacy, and shows how contributor-specific privacy policies can be automatically enforced on captured video through contributor-specific algorithms.
- It shows how captured video content can be automatically tagged for future searches, both using meta-data such as location and time as well as image content tags extracted by computer vision algorithms. For image content that is not tagged, we also show how to perform interactive content search of video segments.
- It evaluates a proof-of-concept prototype implementation of the GigaSight architecture to expose its scalability bottlenecks. Based on these measurements, the paper proposes mechanisms to improve scalability in future implementations.

Marketing and Advertising: Crowd-sourced videos can provide observational data for questions that are difficult to answer today. For example, which are the billboards that attract the most user attention? How successful is a new store window display in attracting interest? Which are the clothing colors and patterns that attract most interest in viewers? How regional are these preferences?

Locating people, pets and things: A missing child was last seen walking home from school. A search of crowd-sourced videos from the area shows that the child was near a particular spot an hour ago. The parent remembers that the child has a friend close to that location. She is able to call the friend's home and locates the child there.

Public safety: Which are the most dangerous intersections, where an accident is waiting to happen? Although no accident has happened yet, it is only a matter of time before a tragedy occurs. Crowd-sourced videos can reveal dangerous intersections, for timely installation of traffic lights.

Fraud detection: A driver reports that his car was hit while it was parked at a restaurant. However, his insurance claims adjuster finds a crowd-sourced video in which the car is intact when leaving the restaurant.

Figure 1: Example Use Cases

Figure 2: Theft captured in photo (Source: CNN [1])

2 Incentive Model and Privacy

2.1 Continuous Capture and Sharing

Today, users manually select and tag the video fragments they upload to YouTube. Their videos are typically of personal achievements (e.g. first person viewpoint sports), or special occasions that they want to share. The content thus has enough personal value to serve as an incentive for the user to spend time on editing, selection and tagging. This self-interest is absent in continuously captured scenes of everyday life. The value of sharing such scenes lies in the fact that they may have high value to someone else. For example, a recent CNN news article [1] reported the arrest of a thief whose act of stealing appeared in the background of the photo shown in Figure 2. The tiny figure in the background would normally be ignored by anyone who views this image. It is only the context of a theft that makes it relevant. This is only one of many real-world examples in which an image captured for one reason proves valuable to someone else in a completely different context. Stieg Larsson's fictional work "Girl with the Dragon Tattoo" embodies exactly this theme: a clue to the murderer's identity is embedded in the backgrounds of old crowd-sourced photographs [13]. While individual frames from continuous video capture can substitute for photographs in these examples, use of video segments can lead to even richer insights. In the use cases of Figure 1, the public safety example requires video: humans find it easy to detect a near-miss in the motion captured by a video segment, but hard to detect it from a single photograph or frame. More generally, content-based video search algorithms from the computer vision community have greatly increased in sophistication and capability in the past decade [11, 21]. They are now capable of reliably detecting human-meaningful actions such as clapping, hugging, bending down to pick up an object, tripping and falling, slipping and sliding, and so on.

These examples reveal two different requirements. First, in the absence of personal motivation, there needs to be an explicit incentive model for sharing continously captured videos. We believe that any realistic incentive

model must be accompanied by a privacy preserving mechanism that is as effortless as the capturing of the video itself. No economically viable incentive will be high enough to convince large numbers of users to go through their continously-captured videos and manually remove sensitive scenes. Second, tags added by the user who captured a video are unlikely to be sufficient as the basis of search. Content-based search using computer vision algorithms that embody a search query will be needed. This processing can done in advance for frequently-searched objects, thus producing an index for the video collection. However, some searches may involve queries about objects or scenes that were not anticipated during indexing. For these, a search mechanism that is better than brute force viewing of video segments is required.

2.2 Author-Publisher Model

Our goal here is only to describe a *plausible* incentive model. We recognize that a pilot deployment of the proposed system will be necessary to validate the proposed model and to explore other incentive models. We propose that crowd-sourced video be treated as *authored content*. What you see and capture effortlessly through your Google Glass is yours. By looking at some people rather than others, by gazing longer at some window displays rather than others, by averting your gaze from a street scene, and so on, you are implicitly exercising taste and judgement. A video capture embodies that taste and judgement. Good will or peer recognition may suffice to encourage sharing in some use cases (e.g. Google Streetview Live), but they are fragile incentives. A more robust and scalable approach is to create a business relationship with the service provider that invests in the video capture infrastructure. Like a book publisher, the service provider monetizes your authored content and shares the revenue with you. This leads to a simple incentive model: *you reap financial rewards for capturing and sharing scenes that others find to be of value.*

2.3 Denaturing

Unfortunately, always-on video capture is much less deliberate and controlled than authoring text. You can't help capturing scenes, but specific objects/people in them may not be what you (or they) want published. It is therefore crucial to edit out frames and/or blur individual objects in scenes. What needs to be removed is highly user-specific, but no user can afford the time to go through and edit video captured on a continuous basis. One therefore needs a process that continously performs this editing as video is submitted for sharing. If a user is confident that the editing process accurately reflects his personal preferences, he is likely to share his captured video without further review. We refer to this user-specific lowering of fidelity as *denaturing*.

Denaturing has to strike a balance between privacy and value. At one extreme of denaturing is a blank video: perfect privacy, but zero value. At the other extreme is the original video at its capture resolution and frame rate. This has the highest value for potential customers, but also incurs the highest exposure of privacy. Where to strike the balance is a difficult question that is best answered individually, by each user. This decision will most probably be context-sensitive.

Denaturing is a complex process that requires careful analysis of the captured frames. From a technical viewpoint, the state-of-the-art computer vision algorithms enable face detection, face recognition, and object recognition in individual frames. In addition, activity recognition in video sequences is also possible [21].

However, preserving privacy involves more than blurring (or completely removing) frames with specific faces, objects or scenes. From other objects in the scene, or by comparing with videos taken at the same place and/or time from other users with different privacy settings, one might still deduce which object was blurred and is hence of value to the person who captured the video. In its full generality, denaturing may not only involve content modification but may also involve meta-data modification. For example, the accuracy of location meta-data associated with a sequence of video frames may be lowered to meet the needs of *k-anonymity* in location privacy [9, 22, 30]. Whether the contents of the video sequence will also have to be blurred depends on the visual distinctiveness of that location — a scene with the Eiffel Tower in the background is obviously locatable even without explicit location meta-data.

Clearly, denaturing is a very deep concept that will need time, effort and deployment experience to fully un-

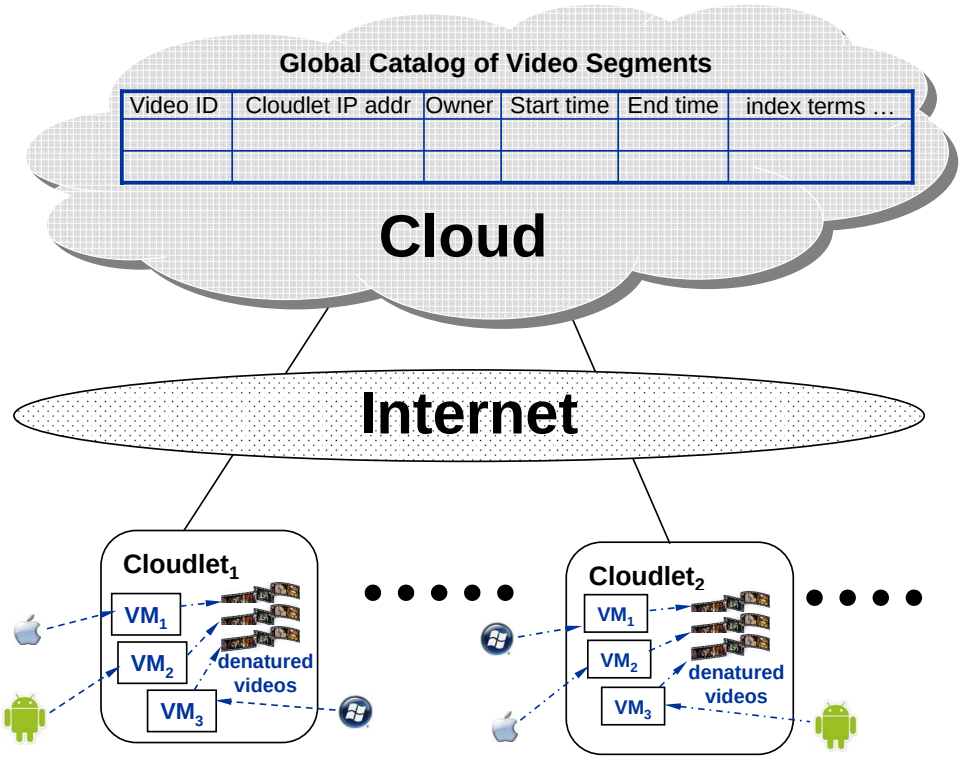


Figure 3: GigaSight Architecture.

derstand. We restrict the scope of this paper to understanding the architectural and performance implications of denaturing, and leave the development of widely-accepted denaturing algorithms as future work. For evaluating the impact of denaturing on scalability in Section 5, we use conceptually simple but compute-intensive denaturing algorithms such as blurring all faces or blurring only a subset of faces from a user-specific list.

3 Architecture: a CDN in Reverse

A key challenge for GigaSight is the high cumulative data rate of incoming videos from many users. Without careful design, this could easily overwhelm the capacity of metro area networks or the ingress Internet paths into centralized cloud infrastructure such as Google's large data centers or Amazon's EC2 sites. Today, 1 hour of video is uploaded to YouTube each second [29], which is the equivalent of only 3 600 users simultaneously streaming. When the usage of HUD becomes mainstream, this number will rapidly increase. Verizon recently announced an upgrade to 100 Gbps links in their metro area networks [18], yet one such link is capable of supporting 1080p streams from only 12 000 users at YouTube's recommended upload rate of 8.5 Mbps. Supporting a million users will require 8.5 Tbps.

To solve this problem, we propose a *hybrid cloud architecture* that is effectively a CDN in reverse. This architecture, shown in Figure 3, uses decentralized cloud computing infrastructure in the form of VM-based *cloudlets* [23]. A cloudlet is a new architectural element that arises from the convergence of mobile computing and cloud computing. It represents the middle tier of a 3-tier hierarchy: mobile device – cloudlet – cloud. A cloudlet can be viewed as a "data center in a box" that "brings the cloud closer." While cloudlets were originally motivated by reasons of end-to-end latency for interactive applications, our use of cloudlets here is based solely on bandwidth considerations. Today, "micro data centers" from companies such as Myoonet [17] and AOL [15] are already available for repurposing as cloudlets. To achieve good performance and extend battery life, tasks such as image processing for denaturing are offloaded from mobile devices to cloudlets. Zhu et al. [31] propose a media-edge cloud architecture in which storage, CPUs and GPU clusters are presented at the edge of the network to provide sufficient QoS and QoE. This architecture aligns with our cloudlet-based 3-tier approach.

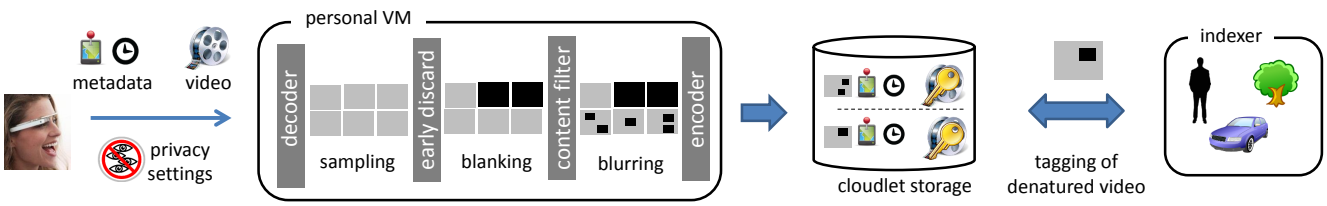


Figure 4: Overview of the GigaSight implementation.

It is important to note in Figure 3 that cloudlets are not just temporary staging points for denatured video data en route to the cloud. With a large enough user base and continuous video capture, the constant influx of data at the edges will be a permanent stress on the ingress paths to the cloud. Just buffering data at cloudlets for later transmission to the cloud won't do — because users will be streaming 24/7, there will never be a "later" when ingress paths are unloaded. Hence, cloudlets are the true home of denatured video segments. Only metadata about these video segments (such as owner (anonymized), location of capture, start and end time of capture, cloudlet where stored, and index terms) is stored in a global catalog in the cloud. In a small number of cases, based on popularity or other metrics of importance, some video segments may be copied to the cloud for archiving or replicated in the cloud and other cloudlets for scalable access. But most video segments reside only at a single cloudlet. How long they are kept around depends on the storage reclamation and replication policy.

Denaturing is performed on a cloudlet as video is being streamed into it. For each mobile user who is currently associated with a cloudlet, there is a "personal" VM that performs the customized denaturing for that user. When a user moves a significant distance, this VM migrates to another cloudlet that is now closer. This process is analogous to Wi-Fi handoff. Although the use of VMs incurs additional overhead relative to denaturing by a cloudlet-wide process, our approach supports user-specific denaturing and is likely to give the user greater confidence in the process.

Other cloudlet VMs (not shown in Figure 3) encapsulate image processing code to perform background indexing of recently-captured video segments. Users can also manually add tags to their video segments. To handle searches that are time-sensitive (such as locating a lost child) or to search for content that is not indexed, custom search code encapsulated in a VM can directly examine denatured video segments.

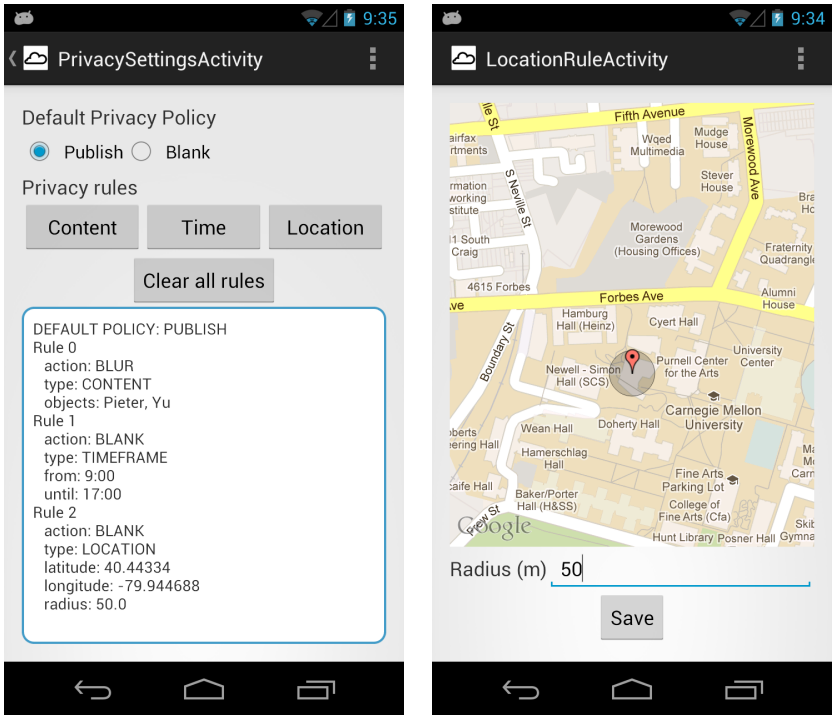
4 Prototype Implementation

We have implemented a complete prototype of the GigaSight architecture presented in Figure 3. The prototype comprises the complete chain of functionality from uploading video, over denaturing and indexing to the final step of searching through the available video segments. We structure the flow of the discussion according to the subsequent components a captured video segment will pass through before becoming searchable.

4.1 Mobile client

Given the stringent limitations on weight and size to keep HUDs comfortable, it can be expected that their storage and processing power will be limited. We argue that these devices will primarily be marketed as a new type of user interface and that most users will still carry smartphones. We envision a smartphone as a proxy to the GigaSight framework where video segments are buffered during periods without network connectivity. Our GigaSight mobile app was developed for Android Ice Cream Sandwich (4.0.4). Its functionality includes the configuration of user-specific privacy rules and the buffering of video segments. Figure 5 shows two screenshots of our app. In our current prototype, the mobile device camera acts as a complete surrogate for the HUD, including the actual video capture. When comfortable HUDs become available and the technical specifications become better known, we can revise the distribution of functionality between the wearable display and the smartphone.

The user defines a default policy to publish or to blank all his video segments. He can specify rules to deviate from this policy. Each rule is a chain of filters that scope time, location or objects. We apply a logical OR between



(a) Example privacy rule (b) Example location filter

Figure 5: Screenshots of the GigaSight app.

rules, and a logical AND between the filters of a single rule, given the user sufficient semantic possibilities to specify privacy settings. Time filters allow the user e.g. to specify that no video is allowed to be shared during 9 AM and 5 PM. Location conditions allow the user to specify geographical regions by simply tapping on a map. The object-based filters are currently limited to the faces present in the training set of our face recognition algorithms. In principle, this would allow the user for example to blur all his Facebook friends in his video segments. Although we recognize that extensive user studies are required to find the most intuitive way for users to specify their privacy settings, we believe that the categories of time, location and visible objects reflect basic criteria that will be present in each privacy policy.

Captured videos are cached and uploaded as soon as a Wi-Fi connection becomes available. Through a REST-based API, the mobile device registers new segments and updates the associated metadata information. At present, we capture audio and video in MP4 format, as well as time-stamped GPS information. In future versions, more information streams can be associated with the video when additional sensors become available on HUDs.

4.2 Personal VM

The personal VM denatures the video uploaded by the mobile client following the user-defined privacy rules discussed in the previous section. It is the only component in the GigaSight entity, apart from the mobile device itself, that accesses the original, non-denatured video. As such, it forms the cloud-based counterpart of the mobile device: an entity that the user trusts to store personal content, but with much more computational and storage resources. When a mobile client signals that it has a segment ready for upload, the personal VM requests the Data Manager to allocate storage space for the denatured video. The Data Manager, running as a cloudlet-wide service in a separate VM, organizes the cloudlet storage and metadata database. More details on the Data Manager are provided in section 4.3.

Figure 4 illustrates how uploaded video is denatured inside the personal VM before being stored on the cloudlet. For the sake of clarity, the components interfacing with the mobile client and the Data Manager are not shown in

Figure 6: Example of a denatured video frame.

this figure. The denaturing process is implemented using C++ and OpenCV 2.4.2. OpenCV is the de facto standard open source library for computer vision and provides support for image processing.

Denaturing is implemented as a multi-step pipeline. In the first step, a subset of the video frames is selected for actual denaturing. As the results presented in section 5.2.2 will clearly indicate, video denaturing is too computationally complex to perform at the native video frame rate. Then, metadata-based filters with low computational complexity are applied. This early-discard step is a binary process: based on the time and location, the frame is either completely blanked or passed through unmodified. In the third step, we apply more complex content-based filters. Currently, our prototype only supports face detection: any detected face will be blurred, as shown in Figure 6. The video frames are first decoded to raw RGB frames and converted to gray scale. Histogram equalization is applied to improve the contrast of the frame in order to increase the face detection accuracy. For each frame, the algorithm gives an array with the bounding boxes of the detected faces.

We combined a profile face detector with two variants of frontal face detection. The profile face detector and one of the frontal face detectors are based on Haar-like features [28, 14]. The classifiers for these algorithms are both included in the OpenCV distribution. The second frontal face detector uses the Local Binary Pattern (LBP) [2]. The LBP classifier[1] we use was trained from the PUT Vein Pattern Database[2].

The output of the denaturing process is a low-framerate video file that is stored on the cloudlet storage system. We argue that a low frame rate is still sufficient to provide a representative overview of the video content for the indexing and search operations. The same observation has been previously exploited in video codecs, whereby only a few frames are encoded independently, and the compression of all other frames depends on the difference to these key frames. The frames of the denatured video act as thumbnails for the full fidelity video.

Along with the denatured version, we also store an encrypted version of the original video. In the current version of our prototype, the personal VM encrypts video based on the Advanced Encryption Standard (AES) implementation that is included in the OpenSSL library. Each encrypted video can only be decrypted with a unique private key that is generated during encryption. If a user, based on his viewing of the low-framerate denatured version, wants to see the complete video, the original must be decrypted and denatured inside the personal VM of the user that originally captured the video. Of course, if a video has been completely denatured as the result of a previous search, the ad-hoc denaturing will not be repeated.

[1] `http://www.visionopen.com/cv/cascadePUT.xml`

[2] `https://biometrics.cie.put.poznan.pl/`

4.3 Data Manager

The Data Manager runs in an individual VM on the cloudlet. It manages the storage of the video segments and the database with the associated metadata. The data is logically organized as a collection of segments. We define a segment as a record of the reality during a continuous period in time and space. Each segment contains one or more streams, each representing a single informational dimension of the reality. Streams can be audio, video, GPS coordinates or any other sensor information captured.

All video and metadata are stored on local disk storage in the cloudlet, that is made accessible to the local VMs as a Network File System-mounted (NFS) volume. NFS permits multiple VMs to concurrently access this storage. When a personal VM or the indexer wants to write or read a stream, the DataManager will provide a NFS-mounted path. Access to the DataManager is mediated through a REST-interface implemented using TastyPie, Django and Python.

The metadata describing the different streams is stored in a MySQL database and initially includes the segment ID of the stream, capture time, duration, access control rights as well as a geographical bounding box of the GPS coordinates, delineating the area in which the video has been captured. The GPS fixes are stored as a separate stream of the same segment. This metadata will allow the user to restrict the scope of (and hence speed up) his search, e.g. by specifying in his query to return only videos captured in the city centre at night. When the thumbnails of a video stream are processed by a specific indexer, the results are stored in a separate table of the database. These tags describe the detected objects and their positions in the video.

We deliberately choose to maintain the master copy of a video on the cloudlet itself. Our main motivation stems from bandwidth considerations, as explained earlier. However, since cloudlets are located at the network edge, all users connected to a specific cloudlet will upload videos that are mostly captured in the same geographic area of the cloudlet. This characteristic may be used as first-order heuristic to determine which cloudlets to be searched when a user performs a query on the video search engine.

4.4 Video Content Indexer

The indexing of denatured video content is a background activity that is performed by a separate VM on the cloudlet. In our current implementation, each denatured frame is analyzed individually by computer vision code to obtain tags for that frame. In a next version of our prototype, we will include the analysis of sequences of denatured frames so that we can tag human-meaningful actions such as clapping, falling, etc. For each indexed frame, an entry is created in a dedicated tag table of the cloudlet database. Each entry contains the tag, the ID of the video segment and a confidence score. For example, an entry "dog, `zoo.mp4`, 328, 95" indicates that our indexer detected with 95% confidence a dog in frame 328 of the video `zoo.mp4`. After extraction, these tags are also propagated to the catalog of video segments in the cloud. The global search workflow using these tags is described in Section 4.5.

As a proof of concept, we use a Python-based implementation of Shotton et al's image categorization and segmentation algorithm [24] with classifiers trained on the MSRC21 data set mentioned in that work. This enables us to identify, with acceptable levels of false positives and false negatives, the 21 classes of common objects such as aeroplanes, bicycles, birds, etc. that are listed in the first column of Table 2.

An additional source of information about image content can come from the denaturing step. Since denaturing is performed before indexing, some important features of an image may be not be available to downstream image processing algorithms. For example, a face detection algorithm may not correctly tag an image with faces because they are blurred. The obvious solution of indexing before denaturing is not acceptable because users expect their videos to be denatured first, before they are exposed to the cloudlet's image processing code. We therefore allow the denaturing step to export tags that are discovered in the course of denaturing. In the example above, the tag "face" would have been exported as an attribute of a frame with faces. Thus, the tags for a frame are the union of those obtained during denaturing by the user's image processing code within his personal VM, and those obtained during indexing by cloudlet's image processing code.

4.5 Search Workflow

GigaSight uses a two-step hierarchical workflow to help a user find video segments relevant to a specific context. First, the user performs a conventional SQL search on the cloud-wide catalog. His query may involve metadata such as time and location, as well as tags extracted by indexing. Our prototype uses a Django-based webserver that supports these queries on a MySQL database. A production version of GigaSight would likely use a distributed cloud database such as BigTable to ensure a scalable global service. The result of this step is a list of video segments and their denatured thumbnails. The identity (i.e., the IP addresses) of the cloudlets on which those video segments are located can also obtained from the catalog.

Viewing all the video segments identified by the first step may overwhelm the user. We therefore perform a second search step that filters on actual content to reduce the returned results to a more relevant set. This step is very computationally intensive but can be run in parallel on the cloudlets. This step uses *early discard,* as described by Huston et al [10], to increase the selectivity of a result stream. Using a plugin interface, image processing code fragments called *filters* can be inserted into the result stream. These code fragments allow user-defined classifiers to examine video segments and to discard irrelevant parts of them, thus reducing the volume of data presented to the user. We provide a suite of filters for common search attributes such as color patches and texture patches. For more complex image content (such as the 21 classes shown in Table 2) the user can train his own STF filters offline and insert them into the result stream.

To illustrate this two-step workflow, consider a search for *"any images taken yesterday between 2pm and 4pm during a school outing to the Carnegie Science Center in Pittsburgh, showing two children in a room full of yellow balls and one of the children wearing his favorite blue plaid shirt."* The first step of the search would use the time and location information and the "face" tag to narrow the search. The result is a potentially large set of thumbnails from denatured videos that cover the specified location. From a multi-hour period of video capture by all visitors, this may only narrow the search to a few hundred or few thousand thumbnails. Using a color filter tuned to yellow, followed by a composite color/texture filter tuned to blue and plaid, most of these thumbnails may be discarded. Only the few thumbnails that pass this entire bank of filters are presented to the user. From this (hopefully) small set of thumbnails, it is easy for the user to pick the result shown in Figure 6.

5 Evaluation

In exploring the limits to scaling GigaSight, we seek to identify the key bottlenecks in the system through experiments on our prototype. After the description of our experimental testbed in section 5.1, we characterize separately the load incurred by video upload, denaturing and indexing in section 5.2. Based on the insights gathered, we study in section 5.3 how the available compute capacity on a cloudlet should be allocated between the personal VMs and the indexer in a set-up with multiple phone users.

5.1 Experiment set-up

To evaluate our prototype, we built a set-up consisting of smartphones and advanced quad-core desktop machines emulating cloudlet and cloud hardware. Table 1 lists the details of this hardware. The state-of-the-art dual-core smartphones were connected to the cloudlets over a private 5 GHz 802.11n access point (AP), to minimize interference and collisions caused by other devices not participating to the experiment. The AP was connected via a private 1 Gbps LAN network to the cloudlet. The high-end quad-core desktops may mimick the typical size of a cloudlet that would be deployed today close to a Wi-Fi AP.

Lacking HUDs representative of devices such as Google Glass, we manually captured 10 minute videos with a smartphone while walking in downtown Pittsburgh. The videos were captured with resolutions set to 480p@30fps and 1080p@24fps, the representative formats available on action cameras such as the Contour+2, and encoded through the H.264 codec in a MP4 container.

To expand and diversify our video database, especially for the denaturing and indexing performance evaluation, we crawled 36 videos from YouTube, in the resolutions 360p (including 8 for 480x360 and 4 for 640x360), 720p

Table 1: Hardware used for the experiments

smartphone	Samsung Google Galaxy Nexus I9250 Android v4.0.4 Dual-core 1.2 GHz Cortex-A9 Wi-Fi 802.11 a/g/n 1 GB RAM, 16 GB internal memory
cloudlet	Intel® Core™ i7-3770 CPU @ 3.40 GHz Linux 3.2.0 x86_64 CPU MHz 1600, 4 CPU cores 32 GB RAM, 900 GB hard disk
Wi-Fi AP	Belkin N750 DB Wi-Fi Dual-Band N+ 2.4 GHz / 5 GHz (link rate up to 450 Mbps)

(1080x720) and 1080p (1920x1080). When selecting the videos on YouTube, we ensured that they represented first-person viewpoints and therefore we excluded music clips and other professionally edited content. We classified the selected YouTube videos into 4 categories: first-person viewpoint sport, in-vehicle viewpoint, public events (e.g. street parade) and public places (including holiday reports).

5.2 Scaling and Throughput

The scalability of this architecture, in terms of users that can be served relative to cloudlet compute capacity and wireless network bandwidth, is the most important question that we will explore. The load on our system is incurred by video collection, denaturing and indexing. It is essential to figure out which of these steps poses the scalability bottleneck with our current hardware.

5.2.1 Video Upload: The Cost of Capture

The goal of our first experiment is to quantify the capacity of the Wi-Fi channel in terms of the number of concurrent video upload from multiple phones to GigaSight. Furthermore, we evaluate the individual throughput and energy consumption of each phone.

We first validated the maximum TCP throughput of our Wi-Fi channel using iPerf, a bandwidth measurement tool. When running the iPerf client on our laptop, the average throughput was around 160 Mbps, in line with what can be realistically expected from a dedicated 802.11n channel. However, when running the iPerf client on our Android phone, the measured throughput dropped to 25 Mbps. Clearly, the maximum throughput for a single device is limited by the processing capacity of the device and/or the implementation of the TCP stack and Wi-Fi driver.

We then moved on to experiments with actual video upload, thereby gradually increasing the number of phones uploading to the cloudlet. To avoid any potential overhead of virtualization, denaturing or indexing, we only ran a lean Python server on the cloudlet that measures the total throughput received and the throughput per phone. This experiment allows us to set the baseline of how many users can be supported by our current cloudlet and Wi-Fi AP.

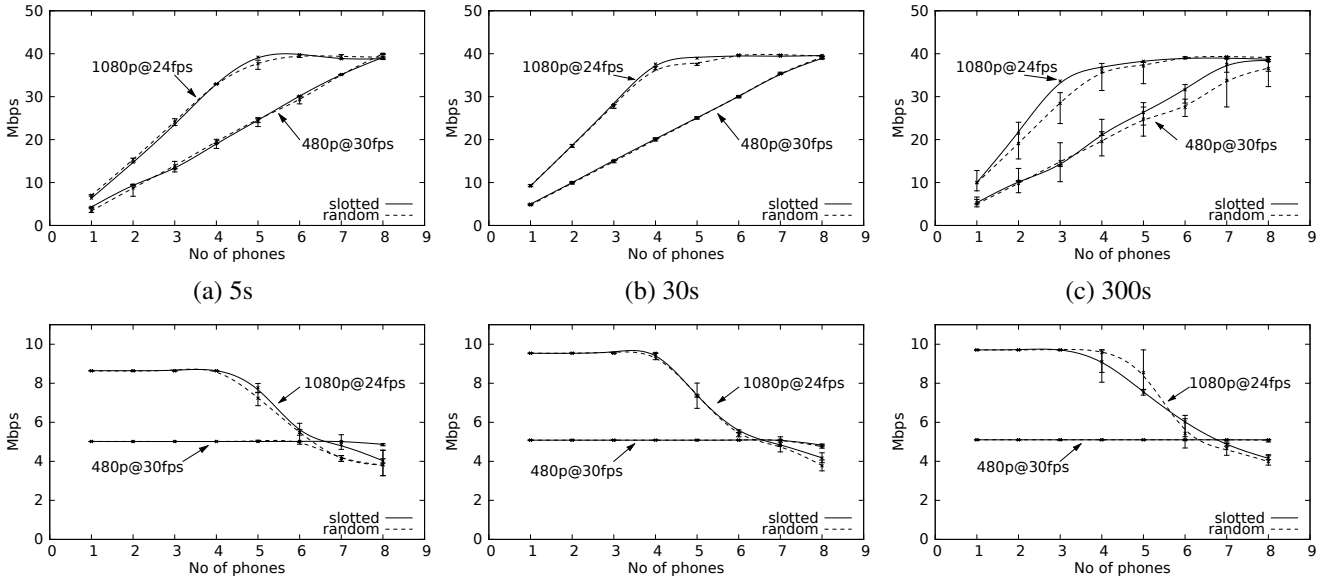
We envision the smartphone as a cache-in-your-pocket, temporarily buffering the captured video before sending it at once to the cloudlet, instead of continuously streaming to the cloudlet. To allow for repeatability of the experiments, we divided videos captured by the smartphone in fragments of 5 s, 30 s or 300 s by means of the FFMPEG tool [4]. These fragments were uploaded to the devices beforehand and reused during the experiments.

We compare two types of upload pattern: random and slotted. In the random case, the smartphones will start the upload of the first segment after a delay randomly selected in the interval [0, n], with n equal to the length of the video segments used (5 s, 30 s or 300 s). The subsequent segments become available for upload one by one every n seconds.

In the slotted case, the interval of n seconds is divided into a number of slots equal to the number of participating phones. For example, with 5 s segments and 2 phones, one phone will upload its first segment at t = 0 s, 5 s, and

so on, whereas the other device will upload at t = 2.5 s, 7.5 s, and so on. This mechanism tries to avoid as much as possible collisions with other phones. With more phones, the individual slots become shorter and it becomes more likely that the upload of a video segment is still in progress when the next phone starts transmitting. At this point, the individual throughput of each phone will start degrading.

We conducted experiments for the 480p and 1080p videos. The results, averaged over 5 iterations, are shown in Figure 7. The figures in the top row show the cumulative throughput received by the cloudlet and are indicative for the total load on the system. The bottom row shows average throughput per phone.



(a) 5s (b) 30s (c) 300s

(d) 5s (e) 30s (f) 300s

Figure 7: Throughput [Mbps] for different resolutions and video framerates. The figures on the top row show the cumulative throughput of the AP, the bottom row shows the net throughput achieved by each phone.

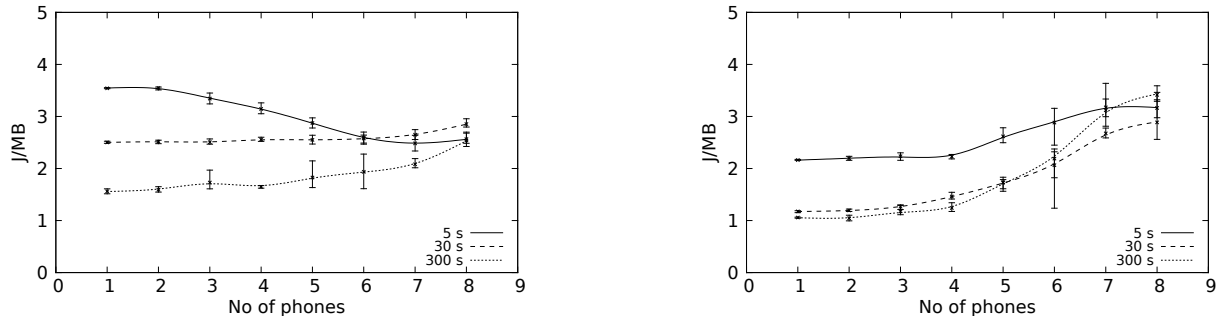
For both resolutions, the overall throughput to the cloudlet increases linearly with the number of phones, until the maximum channel throughput (approx. 40 Mbps) is reached. The bitrate of video upload is around 5 Mbps for 480p@30fps and 10 Mbps for 1080p@24fps. We observed that the throughput per phone starts to drop when there were more than 8 phones uploading 480p videos or 5 phones uploading 1080p videos at the same time.

The overall throughput does not increase significantly with the use of slotted transmission mechanism. We attribute this to the fact that the transmission speed of a smartphone is limited. Even when a phone has the full channel capacity at its disposal, it will not reach the maximum theoretical bitrate. As a result, the duration of the upload of a single segment overspills the length of the slot even with few other phones and relatively long segment sizes. We expect that the performance of the slotted transmission will increase with better hardware and software implementations. Notably, the Android 4.2 update to the Google Nexus phone might bring improvements.

Another observation is that uploading shorter segments results in a lower average individual throughput, due to some deviation in the FFMPEG processing when dividing the video in short segments. The cumulative size of 6 segments of 5 s of video is smaller than when the same 30 s are encoded as a single segment.

Figure 8 shows how much energy it takes for a phone to send one unit of data. We measured the energy by means of the Monsoon Power Monitor [16]. The presented results are based on the random upload scenario; the results for slotted upload are largely similar and omitted for clarity and space considerations. In general, the energy consumption increases drastically when the Wi-Fi channel is satured.

In general, it is more energy efficient to upload larger video segments. The device has to wake up less frequently from the sleep state, while the total number of bytes transmitted remains constant. Interestingly, the energy



(a) 480p@30fps (b) 1080p@24fps

Figure 8: Average energy consumption of the smartphone

consumption per byte for 5 s 480p video upload decreases with the number of nodes until the Wi-Fi channel is saturated. Careful analysis of the power measurement traces indicated that the smartphone applies two power states for transmission. In the experiments with one and two smartphones, the transmission starts in the highest power state before switching to the lower power state. When more smartphones are involved in the experiment, the smartphone immediately starts in the lower power state, presumably because it senses more collisions on the network. Transmitting in a lower power state results in longer transmission durations. In the 5 s 480p case, the increased energy cost of the longer transmission duration does not take over the energy savings realized by sending at a lower rate because there is not enough data to send. In all other cases, the longer transmission duration results in an increase of the average power consumption.

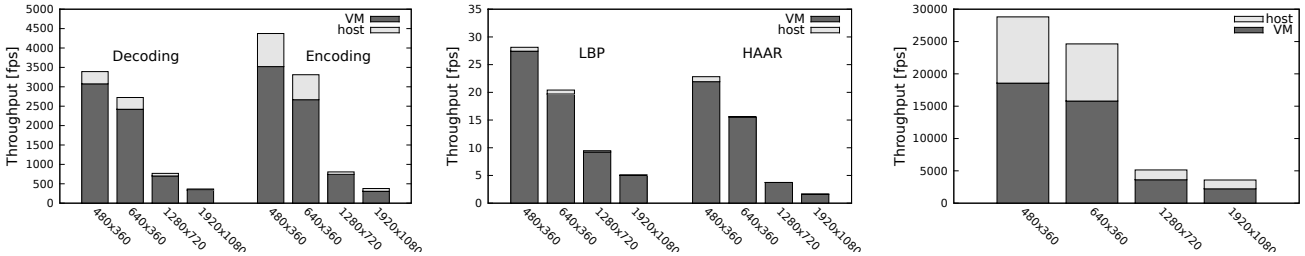
5.2.2 Denaturing: The Cost of Preserving Privacy

In this section, we provide answers to two questions with respect to the scalability of denaturing. First, what is the throughput of each step of the denaturing pipeline presented in Figure 4, expressed in terms of video frames processed per second? Second, what is the overhead of virtualization in terms of the degradation of the throughput of the denaturing process?

As listed in Table 1, each cloudlet machine has 4 cores. By enabling hyperthreading, each machine can provide up to 8 virtual CPUs (vCPUs) that must be distributed between the VMs running on the host. We used KVM as hypervisor. The video denaturing pipeline consists of 4 stages: video decoding, early-discard of frames based on metadata and sampling rate, content-based blurring and video encoding. Along with a denatured, low-framerate version, the entire original video is stored in encrypted form on the cloudlet. To measure the individual throughput of each step, we ran each stage individually, either on the host or inside a VM with 8 vCPUs. The results are shown in Figure 9. We omitted the results of the early-discard, since this is a computationally trivial step. Note that the content-based filtering is the only step in the pipeline that can be parallelized: decoding, encoding and encryption rely on the sequential order of the frames. Hence, to obtain the results we ran 8 threads for face detection and blurring, while we used one thread for each of the other steps. The tests were conducted with the 36 videos crawled from YouTube, representing a wide diversity in content.

As shown in Figure 9, the throughput decreases with the video resolution for each step in the denaturing pipeline. Even for the highest resolution (1080p), a personal VM with 8 virtual CPUs can decode/encode over 300 fps and encrypt over 2200 fps. This is much higher than the load that can be expected on a single personal VM: users will typically upload not more than 30 fps. However, the throughput of face detection is much lower. In our experiments, we combined the HAAR-based profile detector with both the LBP and the HAAR frontal face detector. Clearly, the HAAR frontal face detector is more computationally intensive: the throughput drops from 21.93 fps to 1.61 fps when the resolution increases from 360p to 1080p. The LBP-based frontal face detector achieves relatively higher throughput: it processes up to 27.42 fps for 360p videos and 5.04 fps for 1080p videos. Obviously, the major bottleneck is the computer vision algorithms used for face detection.

In practice, the decoding, encoding and denaturing routines are running in parallel. Upon the arrival of a video



(a) Video Decoding and Encoding (b) Face Detection and Blurring (c) AES Encryption

Figure 9: Performance of Video Denaturing

segment, the decoding thread converts the video into raw frames and pushes them into a shared queue. Meanwhile, a number of parallel denaturing threads fetch the raw frames and apply the meta-data and content-based filtering. The processed frames are forwarded to an encoding thread to be written into a new MP4 video file. This new video file contains only denatured frames and is indexable. Its framerate depends on the number of frames selected per second of video for denaturing. Typically, this selection rate will be much lower than the original frame rate because of computational constraints. As explained in section 4, the video at original frame rate can be denatured on an ad-hoc basis.

We measured the overall throughput of the personal VM in three scenarios: running as a native process, inside a personal VM with 8 vCPUs or running in 2 personal VMs with 4 vCPUs each. The latter scenario reflects the case with two users. The number of parallel denaturing threads is equal to 8 in the native scenario, and equal to the number of vCPUs in the virtualized scenarios.

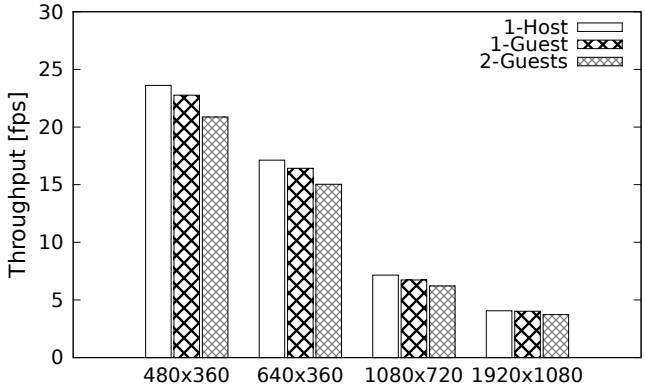
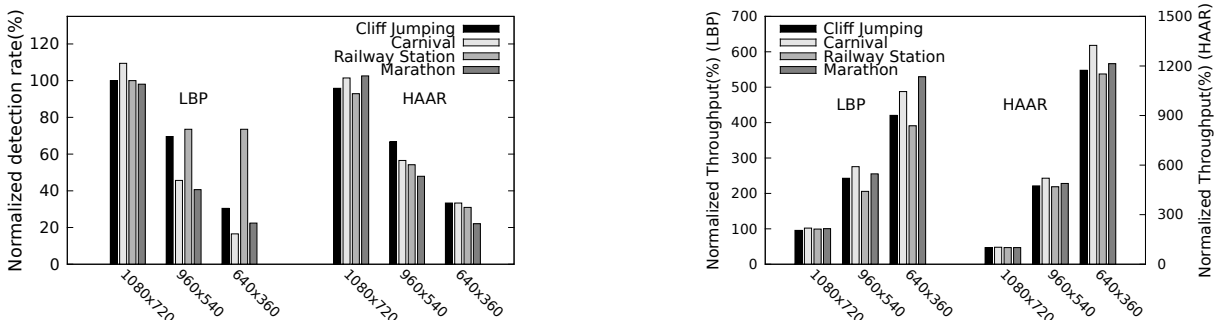


Figure 10: Overall throughput of the personal VM

The results are shown in Figure 10, including the cumulative throughput of both VMs in the 2-Guests scenario. Denaturing the video in a native process on the host results in an overall throughput of 4.06 fps for 1080p videos, 7.16 fps for 720p and up to 23.61 fps for 360p. This processing throughput drops by 1 % to 6 % when using virtualization. Due to scheduling overhead, the cumulative throughput of the cloudlet drops by an additional 7 % to 8 % in the scenario with 2 VMs. We can conclude that given the computational requirements of the denaturing, the penalty of virtualization on the total throughput is limited.

Reducing the resolution is a clear path to increase the throughput of the personal VM, but might negatively impact the detection accuracy of the denaturing process. We are convinced that the privacy preserving algorithms must be close to perfect for users to trust the system. To investigate this trade-off between throughput and accuracy, we selected one 1080p video from each of the 4 categories in our YouTube video database and created lower-resolution copies. We randomly chose 50 frames of the complete video and manually counted the number of



(a) Accuracy vs. Resolution

(b) Throughput vs. Resolution

Figure 11: Accuracy and Throughput, normalized to the original 1080p resolution.

detected faces in each of the resolutions. The results shown in Figure 11(a) are relative to the number of correctly detected faces we counted in the original resolution, while the results shown in Figure 11(b) are relative to the throughput when processing at the original resolution.

Scaling down the resolution prior to denaturing results in a higher throughput but has a clear impact on the accuracy. Interestingly, scaling down from 1080p to 720p does not result in a significant decrease of the detection accuracy, but neither does it significantly improve the throughput. For some videos, the accuracy is even higher than in the original, 1080p resolution. Further lowering the resolution is however detrimental to the accuracy, mainly because relatively small faces in a video are no longer detectable. Note that the classifiers we used were trained with images of faces not larger than 20x20 pixels and hence should in principle be able to detect such small faces.

Our results suggest that denaturing should be performed at resolutions of at least 720p, otherwise users may loose their trust in the denaturing process. On the other hand, one might argue that small faces are already unrecognizable at lower resolutions even without denaturing. We recognize that a more in-depth study of this trade-off is required.

5.2.3 Indexing: The Cost of Content Search

The process of content-based indexing is conceptually very similar to denaturing: it involves running computer vision algorithms inside a VM on the cloudlet. Hence, the main research questions involving the indexer are similar to the ones posed in the previous section: what is the throughput of the indexer in frames per second, and what is the penalty of virtualization? To evaluate the impact of the resolution on the accuracy, we indexed 12 YouTube videos in 1080p resolution and their scaled-down versions.

We compare the throughput of the indexing process for the same scenarios: running as a native process, or inside a VM with either 8 or 4 vCPUs. The frames are indexed using the STF filter for the MSRC21 dataset, generating a confidence value for each of the 21 tags.

Figure 12 shows how the indexing throughput decreases from 4 fps for the lowest resolution to less than 1 fps for the highest resolution. Table 2 shows the trade-off between resolution and indexing accuracy for each of the 21 classes. We considered the frames detected by the filter in the original 1080p video as the ground truth (perfect accuracy) and compared how many of those frames are found in the same video at lower resolution. When compared to the denaturing, the detection accuracy of the STF filters is more robust to lowering the resolution. Overall, we can conclude that in our current prototype we can perform the indexing on the lowest (360p) resolution without sacrificing too much accuracy.

5.3 Cloudlet Capacity Allocation

The previous sections indicated how denaturing and indexing are extremely computationally intensive. In this section, we study how the available cloudlet hardware should be allocated between the indexer and the personal VMs of multiple users. Allocating more vCPUs to the personal VMs in favor to the indexer allows to increase the

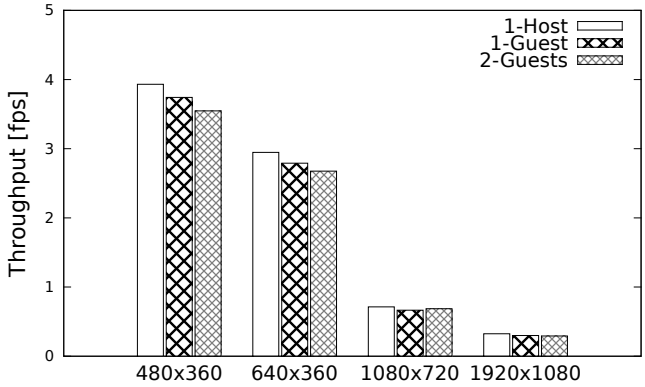


Figure 12: Throughput of the indexing process

number of denatured frames per second of video for each user, but the number of non-indexed denatured videos will keep increasing if the indexer has not sufficient processing power. Conversely, overdimensioning the indexer might result in idle CPU cycles because it is waiting for new denatured videos from the personal VMs.

We combined 4 quad-core machines to a larger cloudlet and dedicated two of them to the indexer. The resources on the other two machines were distributed over a number of personal VMs for the denaturing process. The Data Manager was running on one of the machines used for denaturing. Videos were uploaded in chunks of 30 s at 1080p resolution and processed through the complete GigaSight chain: registering in the database, decoding, denaturing, encryption and indexing. We followed our conclusions of the previous sections: denaturing is performed at the original 1080p resolution for maximum accuracy, while the indexer will first transcode the denatured video to 360p before running the actual content detection algorithms.

We varied the number of vCPUs of the personal VM, the number of users and the selection rate: the number of frames per second of video that is forwarded to the early-discard and content-based filtering steps of Figure 4. These two parameters jointly determine the load on the GigaSight system. For example, if there are two users uploading at 24fps and each personal VM selects 1 out of 24 frames for processing, the personal VMs are expected to produce 2 denatured frames every second. The throughput of the indexer must operate at the same rate or higher.

Table 3 provides an overview of workable system configurations and the throughput of the denaturing and indexing VMs. The bottleneck on the overall system throughput is indicated in bold. When the indexer is indicated as the bottleneck, it means we had to reduce the selection rate of the personal VMs and hence not fully utilized the resources allocated to the personal VM. Conversely, when the denaturing is the bottleneck, it means that the indexer has spare resources and that the configured selection rate is the maximum that can be achieved with the given size of the personal VMs. Note that the number of users and the number of video segments increases the overhead of virtualization and the overhead of database transactions, respectively. All these factors must be taken into account when allocating the cloudlet capacity.

6 Related Work

Privacy Computer vision algorithms have been widely used for preserving the user's privacy in video sharing. PrivacyCam [5] is a framework developed for CCTV channels for public safety. Introducing different layers of privacy protection, users with higher clearance levels can see more information than daily CCTV operators. Using the hardware contemporary at the time of writing, the authors report a throughput of approx. 4 fps of 320x240 JPEG frames. Hardware has noticeably improved since the publication of the paper, and we achieve approximately the same throughput for higher resolution frames. PrivacyCam is complementary to our work, as it aims to protect the privacy of the persons being filmed, whereas we are primarily concerned with protecting the privacy of the video creators.The creators are given full control about which scenes are made public.

The privacy risks emerging from innocuous wearable sensors have been extensively studied by [20]. The

Table 2: Accuracy of the indexing process for different resolutions

Classes	Total Tags	720p		480p		360p	
		TP (rate)	FP	TP (rate)	FP	TP (rate)	FP
aeroplane	4	4(100.0%)	0	4(100.0%)	0	4(100.0%)	0
bicycle	0	0 (N/A)	2	0 (N/A)	2	0 (N/A)	11
bird	68	63 (92.6%)	4	59 (86.8%)	7	57 (83.8%)	15
boat	0	0 (N/A)	0	0 (N/A)	0	0 (N/A)	0
body	2408	2352 (97.7%)	19	2284 (94.9%)	24	2185 (90.7%)	23
book	74	74(100.0%)	20	74(100.0%)	60	74(100.0%)	92
building	875	870 (99.4%)	137	856 (97.8%)	219	836 (95.5%)	273
car	122	118 (96.7%)	52	111 (91.0%)	136	108 (88.5%)	176
cat	573	560 (97.7%)	10	552 (96.3%)	15	546 (95.3%)	24
chair	22	22 (100.0%)	1	20 (90.9%)	2	19 (86.4%)	5
cow	0	0 (N/A)	0	0 (N/A)	0	0 (N/A)	0
dog	1004	965 (96.1%)	14	921 (91.7%)	25	882 (87.8%)	35
face	523	504 (96.4%)	32	468 (89.5%)	59	464 (88.7%)	65
flower	12	11 (91.7%)	0	10 (83.3%)	0	10 (83.3%)	0
grass	131	126 (96.2%)	0	120 (91.6%)	0	116 (88.5%)	0
road	370	351 (94.9%)	24	317 (85.7%)	41	291 (78.6%)	50
sheep	2	2(100.0%)	0	2(100.0%)	0	1 (50.0%)	0
sign	125	124 (99.2%)	10	120 (96.0%)	12	115 (92.0%)	17
sky	1409	1338 (95.0%)	1	1210 (85.9%)	1	1074 (76.2%)	3
tree	1080	1066 (98.7%)	97	1040 (96.3%)	189	984 (91.1%)	234
water	1290	1208 (93.6%)	4	1145 (88.8%)	6	1102 (85.4%)	8
total	10092	9758 (96.7%)	427	9313 (92.3%)	798	8868 (87.9%)	1031

Results are based on the indexing of 3583 frames obtained by sampling 1 frame per 2 seconds from 12 video segments scaled down to different resolutions. "Total Tags" refers to the number of frames classified to have certain tags by the indexing algorithm at 1080p, which we assume to be ground truth. The number of True Positives (TP) represents how many of these frames were also found in the same video at lower resolution. The rate in parentheses is TP divided by total tags. False Positives (FP) are frames not found in the 1080p resolution. Non-denatured frames were used, as is evident from the results of the face detection.

results highlight an overall lack of awareness of the actual information contained in personal data sets and the privacy threats associated with them. On the other hand, the study reveals that participants have significantly more concerns with sharing sensed data with the general public, especially when released with their identities. We aim to tackle this concern by our concept of denaturing in personal VMs.

Visual content tagging A substantial amount of work has been performed on the use of metadata for automated tagging and highlight selection in video segments. This metadata is typically derived either directly from other smartphone sensors, or from social network activity. In [27], video highlights in live broadcasts of sports events are automatically selected by analyzing sudden bursts in Twitter activity. Bao et al. [3] correlate sensor information of nearby smartphones to detect event highlights, such as acoustic information (cheering) or common orientation (all people looking at the same person making a toast).

We have used purely vision-based algorithms to categorize and tag video segments. Our prototype uses Semantic Texton Forests, but other detectors have been created, typically in specific areas where the number of common concepts is more or less fixed. For example, Snoek et al. [25] proposed visual detectors for analyzing videos of rock concerts. They created detectors for 12 common concert concepts such as a guitar and the logo of the festival. By comparing an expert's opinion with the automatically suggested labels, they report an accuracy of approx 85 %.

Table 3: Multi-user system performance

vCPUs per personal VM	# users	selection rate [fps]	denaturing throughput [fps]	indexer throughput [fps]
2	4	0.5	**1.93**	1.91
	6	0.5	2.83	**2.18**
	8	0.5	3.67	**2.38**
4	2	1.5	**2.73**	2.67
	3	1.5	4.22	**3.67**
	3	1	**2.92**	2.90
	4	1	3.84	**3.47**
8	1	3	**2.93**	2.92
	2	2	3.90	**3.53**
	2	3	5.79	**4.05**

Crowdsourcing of video An interesting use case of video crowdsourcing was presented in [12]. The video captured by smartphones mounted on the windshield is analyzed to detect traffic signals ahead. The information is shared with other drivers in the vicinity. This allows to advise the driver on the optimal speed they should maintain so that the signal is green when they arrive at the next intersection.

7 Future Work

The work described here is only a first step towards scalable content-based search on crowd-sourced, denatured videos. There are several important directions for future research.

Performance of Computer Vision Algorithms GigaSight heavily builds on computer vision algorithms for denaturing and content-based indexing. These algorithms are extremely resource intensive. As our results indicate, the overhead of virtualization is however limited. We believe that the throughput might drastically improve when GPU virtualization matures and advanced, highly optimized GPU routines become available inside the VM.

Besides the resource requirements, another challenge of today's computer vision algorithms is their detection accuracy. Today's algorithms are typically focused on individual photographs and often benchmarked under well controlled conditions of e.g. frontal full-screen faces. Our experience shows that the individual frames of a video captured by a smartphone are often not as tack sharp as true photographs. Lighting variations, semi-occlusion of faces and objects, as well as the detection of even small faces are other challenges.

Snoek et al. [26] have evaluated the progress of the accuracy of visual-search engines between 2006 and 2009. They conclude that the progress is substantial: the performance doubled in just three years.

Storage management and reclamation We have currently left open for further research how much storage capacity a cloudlet needs to have. This capacity will be a function of the average retention period of a video segment and of the observed mobility patterns of typical users from historical traces. Many algorithms published in the context of CDN can be reused in GigaSight, such as the distribution and duplication of popular video segments.

In contrast to YouTube, where videos are never deleted, the major goal of GigaSight is to provide a global view on the world. Hence, another interesting concept we plan to focus on is *semantic deduplication*: how can "largely similar" video segments, e.g. captured by two users walking next to each other, be reduced to just one representative? What are the storage savings that can be realized from semantic deduplication, and what is the impact of its computational cost on the scalability?

Apart from storing the videos, the related metadata must also be handled properly. In our current implementation, we used off-the-shelf MySQL databases on the cloudlet and the cloud. Metadata is synchronized over a REST-interface. Although this provided enough performance for our experiments, to truly scale up the system we will look into more advanced distributed databases such as Spanner [7].

8 Conclusion

Having a searchable catalog of crowd-sourced videos provides a unique hindsight view on the world that potentially leverages many existing and future applications. We have presented GigaSight, a framework integrating the complete chain of processing from the initial capture of video to the visual content search. Uploaded videos are denatured inside personal VMs running on cloudlets: sensitive scenes are automatically removed based on time, location and content. As denaturing is a near effortless way for users to protect their privacy, this effectively lowers the barrier to share first-person video. The personal VMs establish a clear demarcation line where the original videos can be processed.

We have investigated both frame rate and resolution as parameters to scale up the system. Denaturing only a subset of the video frames results in a linear reduction of the system load, and still allows the user to guess the content of a video from a limited set of frames. This measure can however not be applied to an infinite extent, especially for *inter*-frame action detection. Our experiments indicate that the effect of scaling down the resolution depends on the specific computer vision algorithm used. In our prototype, lowering the resolution does not reduce the accuracy of object detection in our indexer, but dramatically reduces the accuracy of the face detection algorithm in the denaturing process.

As HUDs such as Google Glass enter the mainstream, people will have the ability to effortlessly and continuously capture video. GigaSight offers an approach to transfering this capability into a truly disruptive force.

Acknowledgements

This research was supported by the National Science Foundation (NSF) under grant numbers CNS-0833882 and IIS-1065336, by an Intel Science and Technology Center grant, by the Department of Defense (DoD) under Contract No. FA8721-05-C-0003 for the operation of the Software Engineering Institute (SEI), a federally funded research and development center, and by the Academy of Finland under grant number 253860. Any opinions, findings, conclusions or recommendations expressed in this material are those of the authors and do not necessarily represent the views of the NSF, Intel, DoD, SEI, Academy of Finland, University of Ghent, Aalto University, or Carnegie Mellon University. This material has been approved for public release and unlimited distribution except as restricted by copyright.

References

[1] CNN report: New Jersey family's picture catches theft in the making. `http://www.cnn.com/2010/CRIME/08/24/new.jersey.theft.photo/index.html?hpt=C1`, August 2010.

[2] T. Ahonen, A. Hadid, and M. Pietikinen. Face recognition with local binary patterns. In T. Pajdla and J. Matas, editors, *Computer Vision - ECCV 2004*, volume 3021 of *LNCS*, pages 469–481. Springer Berlin / Heidelberg, 2004.

[3] X. Bao and R. Roy Choudhury. Movi: mobile phone based video highlights via collaborative sensing. In *Proc. of 8th intl conf on Mobile systems, applications, and services*, MobiSys '10, pages 357–370, New York, NY, USA, 2010. ACM.

[4] F. Bellard et al. Ffmpeg. `http://www.ffmpeg.org`, 2012.

[5] A. Chattopadhyay and T. Boult. Privacycam: a privacy preserving camera using uclinux on the blackfin dsp. In *Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2007.

[6] Contour. `http://contour.com/products/contour-plus-2`, 2012.

[7] J. Corbett, J. Dean, et al. Spanner: Googles globally-distributed database. *To appear in Proceedings of OSDI*, page 1, 2012.

[8] Google+. `https://plus.google.com/s/%23DVFthroughGlass`, September 2012.

[9] M. Gruteser and D. Grunwald. Anonymous Usage of Location-Based Services Through Spatial and Temporal Cloaking. In *Proc. of the 1st Intl. Conf. on Mobile systems, applications and services*, San Francisco, California, 2003.

[10] Huston, L., Sukthankar, R., Wickremesinghe, R., Satyanarayanan, M., Ganger, G.R., Riedel, E., Ailamaki, A. Diamond: A Storage Architecture for Early Discard in Interactive Search. In *Proc. of the 3rd USENIX Conf. on File and Storage Technologies*, April 2004.

[11] Y. Ke, R. Sukthankar, and M. Hebert. Event Detection in Crowded Videos. In *IEEE International Conf. on Computer Vision*, Oct 2007.

[12] E. Koukoumidis, M. Martonosi, and L.-S. Peh. Leveraging smartphone cameras for collaborative road advisories. *Mobile Computing, IEEE Transactions on*, 11(5):707 –723, may 2012.

[13] S. Larsson. *Girl with the Dragon Tattoo*. Afred A. Knopf, 2010.

[14] R. Lienhart and J. Maydt. An extended set of haar-like features for rapid object detection. In *Intl. Conf. on Image Processing*, volume 1, pages I–900 – I–903, 2002.

[15] R. Miller. AOL Brings Micro Data Center Indoors, Adds Wheels. `http://www.datacenterknowledge.com/archives/2012/08/13/aol-brings-micro-data-center-indoors-adds-wheels`, August 2012.

[16] Monsoon Solutions Inc. Power monitor. `http://www.msoon.com`, 2012.

[17] Myoonet. Unique Scalable Data Centers, December 2011. `http://www.myoonet.com/unique.html`.

[18] PC World. `http://www.pcworld.com/article/255519/verizon_to_offer_100g_links_resilient_mesh_on_optical_networks.html`, 2012.

[19] Quality of Life Technology Center. `http://www.cmu.edu/qolt/AboutQoLTCenter/PressRoom/ces-2012/first-person-vision.html`, 2012.

[20] A. Raij, A. Ghosh, S. Kumar, and M. Srivastava. Privacy risks emerging from the adoption of innocuous wearable sensors in the mobile environment. In *Proceedings of the 2011 annual conference on Human factors in computing systems*, pages 11–20. ACM, 2011.

[21] S. Sadanand and J. Corso. Action bank: A high-level representation of activity in video. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 1234 –1241, june 2012.

[22] P. Samarati and L. Sweeney. Protecting Privacy when Disclosing Information: k-Anonymity and Its Enforcement through Generalization and Suppression. Technical Report SRI-CSL-98-04, SRI International, 1998.

[23] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies. The Case for VM-Based Cloudlets in Mobile Computing. *IEEE Pervasive Computing*, 8(4), Oct. 2009.

[24] J. Shotton, M. Johnson, and R. Cipolla. Semantic Texton Forests for Image Categorization and Segmentation. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, June 2008.

[25] C. G. Snoek, B. Freiburg, J. Oomen, and R. Ordelman. Crowdsourcing rock n' roll multimedia retrieval. In *Proc. of the Intl. Conf. on Multimedia*, pages 1535–1538. ACM, 2010.

[26] C. G. Snoek and A. W. Smeulders. Visual-concept search solved? *Computer*, 43(6):76 –78, june 2010.

[27] A. Tang and S. Boring. # epicplay: crowd-sourcing sports video highlights. In *Proc. of 2012 ACM conf on Human Factors in Computing Systems*, pages 1569–1572. ACM, 2012.

[28] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Proc. of the IEEE CS Conf. on Computer Vision and Pattern Recognition*, pages I–511 – I–518, 2001.

[29] YouTube. `http://www.youtube.com/t/press_statistics`, 2012.

[30] G. Zhong and U. Hengartner. A Distributed k-Anonymity Protocol for Location Privacy. In *Proc. of 7th IEEE International Conference on Pervasive Computing and Communication*, March 2009.

[31] W. Zhu, C. Luo, J. Wang, and S. Li. Multimedia cloud computing. *Signal Processing Magazine, IEEE*, 28(3):59 –69, may 2011.