Management of crowdsourced first-person video: Street View Live

Steven Bohez
Department of Information Technology
Ghent University - iMinds
Gaston Crommenlaan 8/201
B-9050 Ghent, Belgium
steven.bohez@ugent.be

Jens Mostaert
Department of Information Technology
Ghent University - iMinds
Gaston Crommenlaan 8/201
B-9050 Ghent, Belgium
jens.mostaert@ugent.be

Tim Verbelen
Department of Information Technology
Ghent University - iMinds
Gaston Crommenlaan 8/201
B-9050 Ghent, Belgium
tim.verbelen@ugent.be

Pieter Simoens
Department of Industrial Technology and Construction
Ghent University
Valentin Vaerwyckweg 1
B-9000 Ghent, Belgium
pieter.simoens@ugent.be

Bart Dhoedt
Department of Information Technology
Ghent University - iMinds
Gaston Crommenlaan 8/201
B-9050 Ghent, Belgium
bart.dhoedt@ugent.be

ABSTRACT

We present a framework for large-scale crowdsourcing of first-person viewpoint videos recorded on mobile devices. Collecting videos at a massive scale poses a number of major issues in terms of network planning. To improve the scalability with regards to the number of users, videos and geographical area and better cope with restrictions on storage, bandwidth and processing power, the framework is distributed and based on the two-layer cloudlet architecture. To mitigate the limited bandwidth in the access network, a set of decision algorithms is constructed and evaluated that are able to filter out irrelevant videos based on their metadata and given selection criteria. To illustrate the crowdsourcing framework, we present Street View Live, an application for presenting videos based on location, similar to the popular Google Street View but with up-to-date videos covering the location instead of possibly outdated images. In order to have an up-to-date view of every location, the video collection is continuously extended and updated by crowdsourcing videos from mobile devices.

Categories and Subject Descriptors

C.2.4 [**Computer-Communication Networks**]: Distributed Systems—*Distributed applications*; E.1 [**Data**]: Data Structures—*Distributed data structures*; F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems—*Sequencing and scheduling*; H.3.4 [**Information Storage and Retrieval**]: Systems and Software—*Distributed systems, Performance evaluation (efficiency and effectiveness)*; I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Intelligent agents, Multiagent systems*

General Terms

Design, Algorithms, Performance, Experimentation

Keywords

Crowdsourcing, Mobile computing, Cloud computing, Cloudlets, Street view

1. INTRODUCTION

In recent years, billions of mobile devices such as smartphones and tablets have been sold [4]. As these devices are almost always online, it has become very easy for mobile users to share content on social network sites such as Facebook and Twitter. Mobile devices also posses an array of sensors, such as (possibly multiple) cameras and location sensors, which allow users to upload geotagged photos and videos to sites such as Flickr and YouTube. Over one hundred hours of video are uploaded to YouTube every minute [16]. With the emergence of smart glasses like Google Glass, it will require even less effort to record and share first-person viewpoint videos. It can be foreseen that in the near future, mobile users will generate and upload massive amounts of videos covering their daily lives.

All these videos and location data provide a treasure trove of information, with each video showing a glimpse of the world where and when it was recorded. This information enables a whole new range of applications. One such application is Street View Live, as proposed in [11], which allows users to virtually navigate roads on a map through first-person viewpoint video previously captured by another user walking the same road. Ideally, this video was recorded just moments before in order to provide an up-to-date view. By massively crowdsourcing videos from mobile users, it should be possible to have enough up-to-date material of sufficient quality to cover major cities and even entire countries.

Crowdsourcing videos on such a massive scale, however, poses a number of important challenges regarding network planning, storage capacity, energy consumption, etc. In this paper, we address the issues regarding the scalability of such a large-scale video crowdsourcing framework, applied to the Street View Live use case. This framework should not only be scalable in the number of users and video's being uploaded, but also in the geographic area for which videos are being crowdsourced. On top of that, we need to account for the restrictions in available bandwidth and storage capacity.

These challenges are tackled in two steps. First, Section 3 addresses the issues of having a centralized, cloud-based approach and proposes a better-suited solution in the form of a distributed 2-tier cloudlet architecture. Cloudlets [10, 14], also known as edge clouds, are trusted server infrastructure placed closely to mobile users at the edge of the network, e.g. co-located with the wireless access point. Distributing infrastructure close to the user allows the system to exploit dependencies on location, which in turn reduces the total amount of bandwidth consumed. Nonetheless, videos still need to be transmitted over the last hop of the network, namely the mobile wireless access network. Due to the wireless medium, there are tight bandwidth restrictions that need to be taken into account. While limited, we still want to get the most out of the available upload bandwidth by determining which videos add the most value to the existing collection. Based on this value, which is calculated using video metadata, intelligent decisions can be made on which videos to accept or reject from uploading. Such decision algorithms are discussed in Section 5.

The remainder of this paper is structured as follows. In Section 2, relevant related work is discussed. Sections 3 and 4 handle respectively the distributed architecture and implementation of the video crowdsourcing framework and Street View Live application. Section 5 goes into more detail of the decision algorithms used to accept or reject video uploads, which are then evaluated in Section 6. Finally we draw our conclusions and discuss future work in Section 7.

2. RELATED WORK

Although cameras are arguably the richest sensors currently present on mobile devices, not much work has been done in exploiting these sensors for crowdsourcing applications. We believe this is caused by the limited scalability when crowdsourcing videos on current systems. While several crowdsourcing frameworks have been introduced in recent years that incorporate scalability, such as Medusa [9] and CrowdLab [2], few of them are equipped to handle the requirements of massive-scale crowdsourcing of video [15].

In SignalGuru [7], videos are crowdsourced from windshield-mounted smartphones in order to detect and predict traffic light schedules. This allows drivers to optimize their driving speed which in turn reduces vehicle fuel consumption. The collaborative sensing platform developed in this work is entirely mobile based, requiring no fixed infrastructure. Video processing is done on the mobile devices themselves, which communicate in an ad hoc fashion to exchange traffic light prediction information. This application allows for a completely mobile solution. Firstly because power constraints are less of an issue as drivers tend to plug windshield-mounted devices into the vehicles' power supply. Secondly, the videos themselves do not have to be transmitted over the network and can be discarded after traffic light detection has been performed. Issues regarding storage or bandwidth limitations hence do not occur.

Our work is more similar to the work done by Simoens et al. [11]. They present GigaSight, a hybrid cloud architecture for scalable crowdsourcing of videos. Their approach is described as a Content Delivery Network (CDN) in reverse and is similar to the two-layer cloudlet architecture discussed in this paper. Our approach is different in two major ways however. First, GigaSight aims to crowdsource continuous, real-time video from heads-up displays whereas we opt to crowdsource pre-recorded fragments. This decision is motivated by the fact that it is not yet feasible to continuously capture and stream video from wearable devices due to battery and connectivity constraints. Users are more likely to record shorter, individual fragments, and may even choose to upload them at a later time, e.g. when connected to a power source. Second, this paper focuses on the issues of the actual uploading and gathering of videos whereas GigaSight is aimed at the processing and indexing of the video streams and how those services can be scaled out. Specifically, their work focuses on privacy and introduces the concept of "denaturing" as a way to preserve it. They conclude, however, that even in the near future the limited upload bandwidth of the access network may pose a major issue.

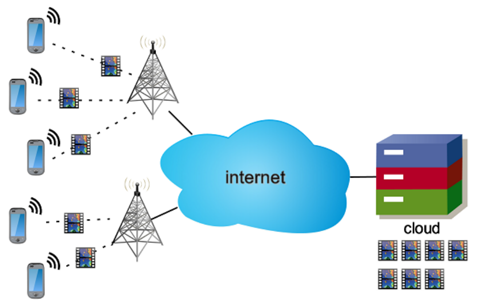
This idea of preserving privacy in the large-scale crowdsourcing of videos is further supported by the work in [12], where a technique is presented which not only efficiently blurs faces and other privacy-sensitive parts of a video, but also significantly reduces the bitrate of the filtered video. The authors specifically advocate their technique for Street View Live applications. The possibilities of crowdsourcing combined with Street View has recently gotten more attention. Google has opened up its Google Maps API to allow users to share panoramic shots of their favorite locations with the world [5]. Instead of Street View being the goal of crowdsourcing imagery, Street View has also succesfully been used as a data source for crowdsourcing. In [6], untrained crowd workers find, label, and assess sidewalk accessibility problems using Google Street View imagery.

3. FRAMEWORK ARCHITECTURE

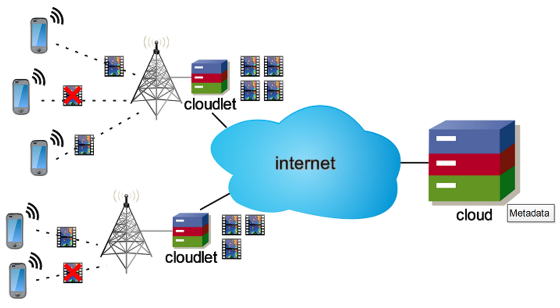
Massively crowdsourcing videos from both a large number of mobile users as well as a widespread geographical area is no easy task. For applications that have to scale out easily, cloud computing has become the golden standard, as it provides computing resources in an on-demand fashion. With the Street View backend deployed on the cloud, mobile clients push videos over the internet to the centralized datacenters where they are stored along with their metadata, as shown in Figure 1a.

3.1 Two-tier Cloudlet Architecture

A major issue with centralized cloud infrastructure is that all data transmitted between the mobile client and the cloud, video in this case, has to pass over the core internet. This causes bandwidth to be consumed along each network connection between the client and the cloud backend, which may add up to a significant amount if a large number of videos are uploaded simultaneously. Moreover, the internet and specifically the last-hop to the cloud datacenter may become an upload bottleneck in some cases. If we assume



(a) Cloud architecture



(b) Cloudlet architecture

Figure 1: In contrast to a classical cloud architecture, a 2-tier architecture is used where videos are stored on cloudlets at the edge of the network and only the metadata is stored centrally.

the recommended bitrate by YouTube of 8.5 Mbps and an available upload bandwidth of 10 Gbps to the datacenter, only about 1200 simultaneous real-time videostreams can be transmitted, which may not be enough to cover (multiple) major cities.

Recently, however, there has been a trend to push cloud infrastructure closer to the edge of the network and the end user, the so-called edge clouds or cloudlets. By deploying a crowdsourcing application on cloudlet infrastructure, the total amount of bandwidth consumption is reduced. Bottlenecks are also less likely to occur as at the most a few network connections are shared between simultaneous video uploads. Moreover, there is an additional argument in favor of the use of cloudlets for applications where data locality can be expected, such as Street View Live. It is plausible that data, in this case video, originating from a certain location will mostly be utilized by users in the same geographic area. For example, in the Street View Live case, videos covering the city of Ghent will most likely be watched by people visiting or living in and around Ghent. Hence, it makes sense to store those videos geographically close to the city.

Therefore, we propose a two-tiered architecture as shown in Figure 1b, with cloudlets at the network edge bridging the mobile users with a centralized cloud. Crowdsourced videos are uploaded over the wireless network to a cloudlet assigned to their geographic coordinates, where they are stored until users look them up. While we want to distribute the videos themselves across the network edge, we still want to maintain a global overview of the entire collection. Therefore a centralized cloud is still desired to store all the metadata of the crowdsourced videos.

However, while the two-tiered cloudlet architecture avoids any bottlenecks at the cloud backend, there still remains the issue of the limited wireless access bandwidth. Permanently storing the videos on the mobile devices themselves is not feasible due to their limited resources. Instead, in order to utilize the available bandwidth as efficiently as possible, an upload request is first sent to a cloudlet and the video itself is only uploaded if this request is accepted. The cloudlet can then make the decision on which videos to upload and which not, based on knowledge of the previous uploads.

3.2 Components

In order to easily deploy the the crowdsourcing framework on the two-tier infrastructure as well as the mobile clients, a component-based approach is chosen. The component diagram on Figure 2 shows a detailed view of the software components constituting the framework, as well as their deployment. The *Communication* component is the only component deployed on the mobile client, while the other components are distributed among among the available cloudlets and cloud infrastructure.

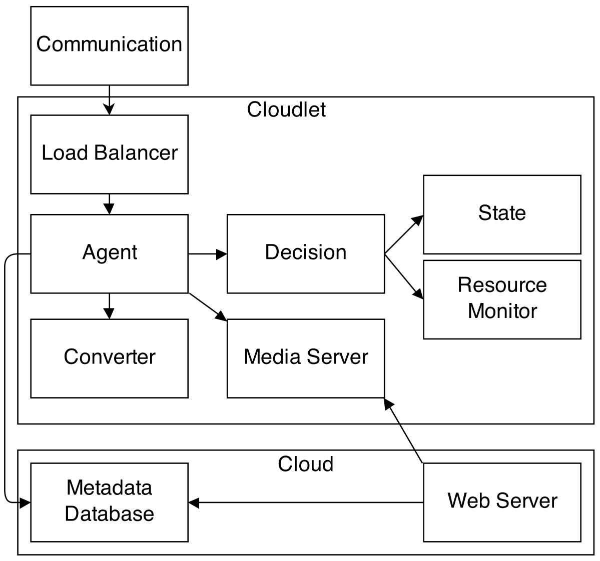
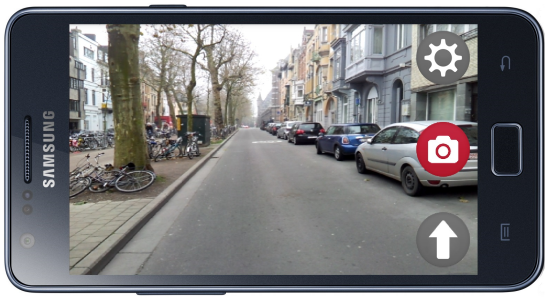


Figure 2: Architecture of the crowd-sourcing framework. The *Communication* at the client sends upload requests to the *Agent*, while the *Decision* decides whether to accept or reject a video.

Communication The *Communication* component is the only component deployed on the mobile device. When triggered by the user, it will make a request for a video upload at the nearest *Load Balancer*. This request contains the required metadata of the video to make the decision whether to accept it or not, e.g. its location and timestamp.

Load Balancer This component will forward the upload request to any of the available *Agent* components. It can do this using any standard load balancing algorithm, e.g. Round Robin.


(a) Mobile client

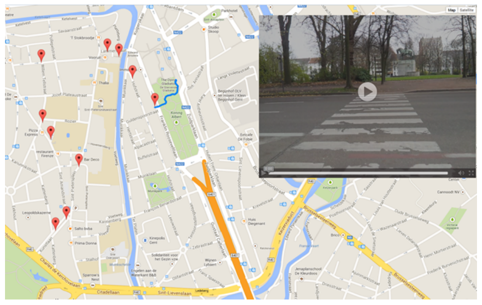

(b) Street View Live webpage

Figure 3: Screenshots of both the mobile application used to record and upload videos, as well as the Street View Live webpage used to view them.

Agent The *Agent* is the most important component of the system, as it actually handles the upload requests. When it receives a requests, it queries the *Decision* component with the metadata contained in the request. When the response is positive, the *Agent* will request the *Communication* component at the client to actually upload the accepted video. When fully received, the communication with the client is complete and terminated. Before sending the video to the *Media Server*, it is first sent to a *Converter* component if the source and required format differ. Multiple *Agent* components can be available per cloudlet, to distribute the load among the available servers.

Decision The *Decision* component will make the decision whether to accept a certain video based on the metadata in the upload request and data form the *State* and *Resource Monitor* components using the desired decision algorithm (see Section 5).

State The *State* contains a (compact) representation of the metadata of part of the already uploaded video collection. The exact structure of the representation will depend on the chosen decision algorithm.

Resource Monitor As the name implies, the *Resource Monitor* component will keep track of the available resources, specifically the available bandwidth for uploading videos and the remaining storage capacity.

Converter The *Converter* is responsible for converting the uploaded video to the format required by the *Media Server*, e.g. a format suitable for streaming over the web.

Media Server The *Media Server* provides the actual video storage facilities of the application, as well as the means of streaming a particular video.

Metadata Database The *Metadata Database* provides a searchable index of the metadata of the entire video collection as well as the location (*Media Server* on a specific cloudlet) where each video can be found. This component is deployed on centralized cloud infrastructure.

Web Server Finally, the *Web Server* provides the frontend for the Street View Live application in the form of a web page that allows users to browse a map with markers showing each of the available videos. When users select a specific marker, a video stream is set up and the video is shown in a pop-up window.

4. IMPLEMENTATION

A prototype of the video crowdsourcing platform was built using AIOLOS[1] [13]. AIOLOS is middleware for scalable mobile cloud computing built on top of OSGi [8], an open-standard service platform for Java. Applications consist of components (bundles) that can be started and stopped dynamically by the OSGi runtime. Bundles register Java objects as services and may themselves request other services. These service references are then resolved dynamically.

The goal of AIOLOS is to facilitate distributing OSGi-based applications in a mobile cloud environment. AIOLOS hides the actual location of the service implementation using proxies, allowing to transparently perform remote calls, but also to migrate services between devices at runtime, as well as scale out services and automatically perform load-balancing. The developer is further aided by provided functionality for service discovery, service and device monitoring, custom load balancing policies and Virtual Machine (VM) management when running on a Infrastructure-as-a-Service (IaaS) provider. Being Java-based, AIOLOS can run on any platform that has a Java Virtual Machine (JVM) available, as well as Android, a popular operating system for mobile devices. It is also compatible with multiple open-source OSGi implementations, such as Apache Felix [1] or the low-footprint Concierge [3], more suitable for mobile devices.

The architecture of the crowdsourcing framework is mapped to AIOLOS by defining the required service interfaces and providing an OSGi bundle for each of the compoments shown in 2. As AIOLOS already provides the necessary functionality for load balancing, however, the *Load Balancer* is omitted. Figure 3 shows screenshots of the prototype implementations of the mobile client and Street View Live webpage.

5. DECISION ALGORITHMS

If multiple videos are uploaded simultaneously, the access network to a specific cloudlet may become overloaded. To avoid this, the *Agent* will first process the metadata of the videos using a decision algorithm in order to decide which

[1]Available at `http://aiolos.intec.ugent.be`

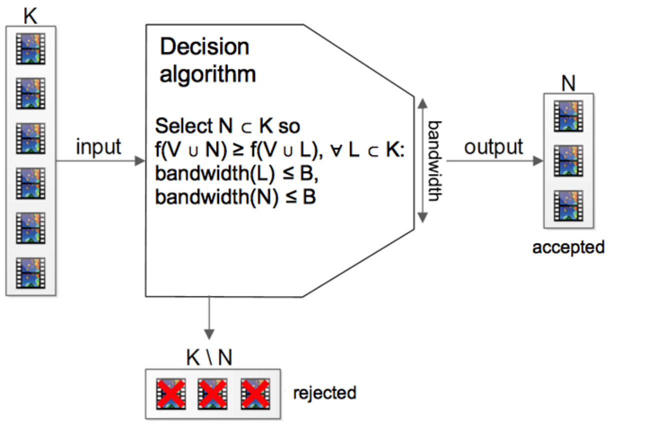
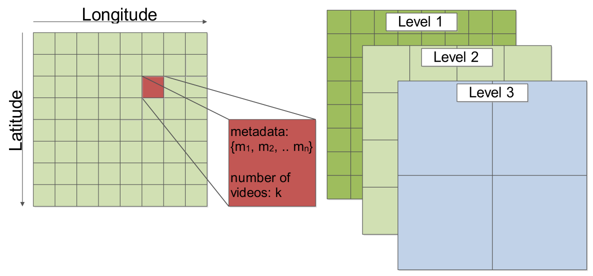


Figure 4: Schematic representation of the decision algorithms.



(a) Histogram (b) Hierarchical histogram

Figure 5: A histogram (a) divides the geographical area into a grid. A hierarchical histogram (b) consists of multiple levels, with each level composed of a histogram of different cell size.

videos should be uploaded and which not. The goal is to optimally use the available bandwidth and not waste network resources on redundant videos. However, it is not feasible to take the metadata of the entire collection into account for each new request as this does not scale well. Instead, we want to be able to only look at a subset of the already collected metadata while still being as close to optimal as possible.

5.1 Formal Description

This problem can be formally described as follows. Let V represent the metadata of all videos the framework already collected. Each time step i a collection of candidate videos K_i is offered to the platform, but there is only bandwidth B_i available, which might not be sufficient to upload all videos of K_i. We thus need a decision algorithm that can select a subset N_i of K_i in such a way that the required bandwidth to upload N_i is less than or equal to B_i. Furthermore, the videos of N_i are picked in such a way that $f(V \cup N_i) \geq f(V \cup L_i)$, with L_i each other possible subset of K_i which requires a bandwidth less than or equal to B_i to upload.

The function f represents a global scoring function which assigns a score $f(V)$ to the collection V. f needs to satisfy a number of properties to be a valid global scoring function. First of, it has to be stationary, i.e. it should not depend on the order in which the videos were added to the collection. Second, it should be increase monotonically when new videos are added, as we expect that each offered video will at least contribute in some (possibly very small) way to the collection. The exact form of f depends on our definition of a relevant video and thus the goal of the crowdsourced video. For the Street View Live application considered in this paper, the videos in the collection should be chosen in such a way that they uniformly cover the geographic area for which videos are collected as well as being as recent as possible. Figure 4 shows a schematic representation of the decision algorithms.

5.2 Algorithm structure

Each decision-algorithm presented here is built according to the same twofold structure. To keep the algorithm scalable with the size of the collection V, the information contained in V has to be compacted in some way, so only the essential subset of metadata is taken into account for the decision. The first part of the twofold structure is therefore formed by the way the algorithm keeps track of the information contained in V, which we call the local *state* of the algorithm.

The second part is formed by the selected local scoring function, which determines a score for each offered video which is calculated using both the metadata of the offered video as well as the local state. Note that this local scoring function, which values a single video with respect to an existing collection, is different form the global scoring function, which values a collection as a whole. The local scoring function is used in the decision to accept a video or not, while the global scoring function is used to benchmark the resulting collection. By calculating the local score for each of the offered videos in K_i, the subset with the highest total score that still fits within the available bandwidth is accepted, while the others are rejected.

Each algorithm can either be executed preemptively or non-preemptively. When using preemption, the set of video currently uploading are added to the set of offered videos K_i, i.e. it is possible to interrupt an uploading video in favor of another video with a higher score. If the already uploaded part is sufficiently large, it is still added to the collection, otherwise it is discarded. When no preemption is used, a video will upload completely once it has been selected, regardless of the videos being offered during the time it is being transmitted. The following sections describe in more detail the choices for the state and local scoring function, which combine to a total of 12 decision algorithms.

5.2.1 State representations

Two representations for the state are considered. The first option is to use a histogram. In this approach, the geographical area for which videos are collected is divided into a grid, as presented in Figure 5a. For simplicity, we assume this area is rectangular. For each cell in the grid, both the total number of videos that are located in that specific cell as well as the metadata of the n videos with the highest local score are tracked. These n videos are used as input for the local scoring function. By adjusting the size of n, we can make a trade-off between accuracy and computation time. Local scoring functions will always be applied on one cell of the histogram, this is the cell that the video is located in for which we are calculating the score. The number of cells in the histogram is a parameter that needs to be optimized, which is done in Section 6.

The second option is to use an extension of the histogram, i.e. the hierarchical histogram. The hierarchical histogram consists of multiple levels, in which each level is composed of a histogram with a different number of cells, as presented in Figure 5b. Again, only the metadata of the top n videos is stored for each cell on each level. The histogram is built bottom-up, with the lowest level consisting of cells of 1 unit of distance squared, until a certain number of levels. Local scoring functions are applied on each level in the same manner as for the histogram, after which a weighted sum of the scores is made with the weight for level i being $w_i = 2^{i-1}$.

These compact state representations are useful because the local scoring function of an offered video only needs to be calculated in a single cell for the histogram or one stack of cells for the hierarchical histogram. This allows the state to be easily partitioned in independent parts, which can then be assigned to the available cloudlets. In other words, each *State* component only has to maintain a limited number of cells, ensuring scalability.

5.2.2 Local Scoring Functions

We propose three different local scoring functions. These are chosen in order to match the criteria of the Street View Live application: aim to have an uniformly spread collection of recent videos. The first local scoring function calculates the score of an offered video as the average Euclidean distance between its location $\{x_a, y_a\}$ and the location $\{x_i, y_i\}$ of the n videos that are tracked for the corresponding cell (i.e. the videos with the highest local scoring function). This distance is divided by the age l_a of the offered video to favor recent videos, as shown in Equation 1. This function has complexity $\mathcal{O}(n)$ in the number of tracked videos n.

$$s_a = \frac{\sum_{i=1..n} \sqrt{(x_i - x_a)^2 + (y_i - y_a)^2}}{l_a} \tag{1}$$

The second local scoring function calculates the score of an offered video using the average number of videos that are located in all cells N (of a level for the hierarchical histogram) and the number of videos in the corresponding cell n_a, as represented in Equation 2. The score is again divided by the age l_a. This scoring function has complexity $\mathcal{O}(1)$.

$$s_a = \frac{N}{l_a\ n_a} \tag{2}$$

The third and final scoring function studied calculates the score of an offered video using the age of this video and other videos that were tracked for the corresponding cell. This scoring function calculates a value before and after adding the offered video. The difference between these values is then used as the final score. s_v represents the value before adding the offered video and V is a list of the ages of all videos tracked for the corresponding cell, sorted from youngest to oldest.

$$s_v = \sum_{0 \le i < n} \frac{1}{2^i\ V_i} \tag{3}$$

s_n represents the value after adding the offered video and N represents V after adding the offered video to this list.

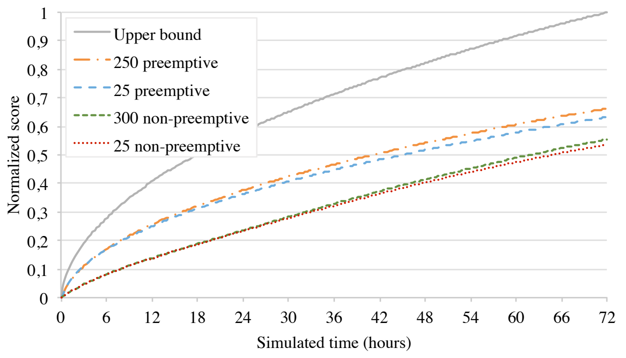


Figure 6: Normalized score with respect to the simulation time. Both the use of preemption as well as the number of cells in the histogram affect the score.

$$s_n = \sum_{0 \le i < n+1} \frac{1}{2^i\ N_i} \tag{4}$$

Finally, Equation 5 shows the final score for the offered video.

$$s_a = s_v - s_n \tag{5}$$

6. EVALUATION

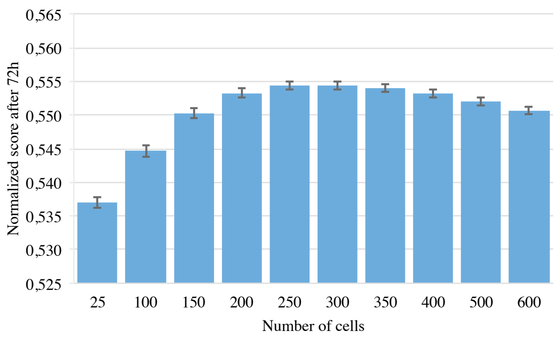
Each combination of a local scoring function and a state representation will result in a new decision-algorithm. The goal of the evaluation is first to optimize the parameters of the different decision algorithms. Second, the different algorithms are compared with each other and with other relevant algorithms both in terms of performance (how well do they approach the optimum) as well their required execution time (which is important at runtime).

6.1 Global Scoring Function

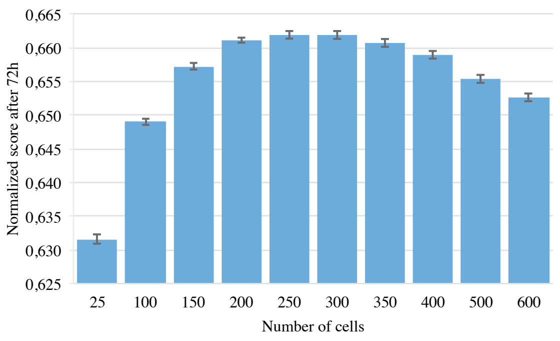
In order to be able to compare these algorithms, we first need to define the global scoring function f. For the Street View Live application, it is desired to collect videos that are as recent and as far from each other as possible. The algorithm to calculate this score is constructed the same way as the hierarchical histogram in Section 5.2.1, but keeps track of all videos in each cell instead of just a few. To calculate the score of the entire collection, Equation 6 is used to calculate a score for each cell, which are then summed to obtain the global score.

$$s = \sum_{0 \le i < n} \frac{\exp(-V_i/\alpha)}{2^i} \tag{6}$$

The aging factor α allows us to adjust the importance that is given to the age of videos against their position. Choosing α smaller, will attach more importance to the age of videos, while bigger values of α will focus more on the position of videos. In our results, *alpha* is set to 50 days. Note that this scoring function satisfies the properties that adding a video to the collection should always lead to an increased score and that the specific order in which they are added is not important. Also note that the total score of the collection decreases when no new videos are added. Decision



(a) Non-preemptively



(b) Preemptively

Figure 7: Simulation results for the decision algorithm consisting of the histogram state combined with average distance local scoring function.

algorithms can now be compared by looking at the global scores obtained after the same set of videos are offered.

6.2 Simulations

The parameters of each decision algorithm were optimized using simulations after which the best results were compared. The results may however partially depend on the parameters that were used in the simulation. The simulated geographic area is a square with length 2^{16} meters, witch matches the size of an average city. The location of videos is Gaussian distributed around the square center, with a standard deviation of 1.3 km. For simplicity, bandwidth is discretized in 200 upload slots, meaning a maximum of 200 videos can be uploaded simultaneously. Videos arrive according to a Poisson process at a rate of 5 per second. Simulations with non-stationary, non-Poisson arrival processes with the same average arrival rate were also performed, but yielded very similar results and are furthermore not discussed for clarity. The average upload time was set to 10 minutes. Upload requests are buffered for a periodic interval of 5 seconds after which the decision algorithms processes them. After the decision is made, the global scoring function is re-evaluated. A total of 72 hours is simulated.

An upper bound for the global scoring function can be found by simulating an infinite amount of upload slots, so no offered video ever has to be rejected. In the results shown, the value of upper bound at the end of the simulation is used to normalize the global score throughout the simulation. Note that this is a very spacious upper bound, as it can never be achieved in reality.

6.3 Parameter Optimization

Figure 6 shows the global scoring function varying through time for the decision algorithm consisting of the histogram state representation combined with the average distance local scoring function, both preemptive and non-preemptive versions for two different granularities of the histogram. The granularity refers to the number of cells in the histogram both length- and width-wise. The upper bound is also shown. Two observations can be made. First, the preemptive version clearly outperforms the non-preemptive one, which matches our intuition. Second, the number of cells in the histogram clearly has an impact on performance, so an optimal number needs to be found.

The optimal number of cells is determined experimentally. Figure 7a shows the global scoring function for the non-preemptive version at the end of the simulation, for a varying number of cells. Values shown are averages over 4 runs along with standard deviation. An optimum is achieved for 300 cells. Figure 7b shows the same plot for the preemptive version, where the optimum is reached for 250 cells. Note that for any number of cells, the preemptive version outperforms the non-preemptive version by about 10%.

This same optimization process can be performed for the remaining 10 decision algorithms. For each combination of state representation and local scoring function, the preemptive version performs significantly better than the non-preemptive one. Therefore, in the remaining evaluations, only the preemptive algorithms are considered.

6.4 Comparison

A comparison between the 6 different preemptive decision algorithms is shown in Figure 8. The preemptive algorithm combining a hierarchical histogram with 10 levels combined with the local scoring function based on video age renders the best result for these simulation settings.

Next, the performance of this algorithm is compared with a few obvious, rather naive algorithms and the upper bound in Figure 9. These algorithms are *First Come First Served (FCFS)*, *Youngest First (YF)* and *Average Distance (AD)*. The FCFS algorithm simply accepts videos in the order they arrive as long as there is sufficient bandwidth available. The YF algorithm accepts the youngest videos first, if there is no sufficient bandwidth available, older videos that are still uploading are interrupted. The AD algorithm selects videos with a higher average distance to all videos in the entire collection first. The hierarchical histogram in combination with the age local scoring function achieved the best score, but the difference with the AD algorithm is relatively small.

Performance in terms of global score is however not the only metric important here. These algorithms need to be executed at runtime and hence their execution time needs to scale well with the size of the video collection. Figures 10a and 10b show that, while the AD algorithm achieves nearly the same scores as the hierarchical histogram, it scales linearly with an increasing video collection. The hierarchical histogram, on the other hand, scales logarithmic. Moreover, the calculation time after 40K videos is about three orders of magnitude lower. This makes the decision algorithm based on the hierarchical histogram clearly the better choice.

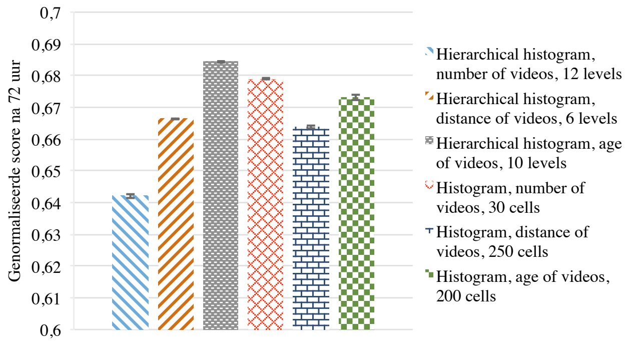


Figure 8: Comparison of the preemptive decision algorithms.

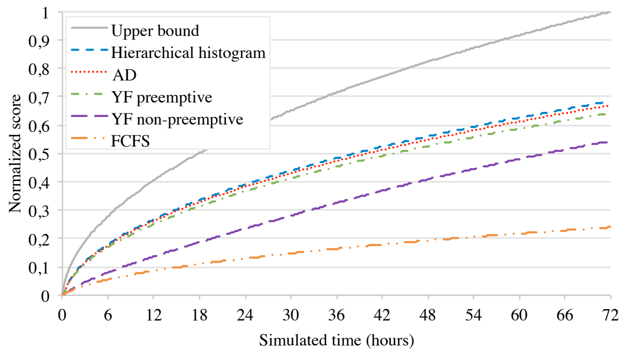


Figure 9: Normalized score with respect to the simulation time. The hierarchical histogram combined with the age scoring function achieves the best score, shortly followed by the Average Distance algorithm.

7. CONCLUSIONS AND FUTURE WORK

In this paper we presented a framework for large-scale crowdsourcing of first-person viewpoint videos recorded on mobile devices. To illustrate the possibilities of this framework, we presented the Street View Live application. This application offers a service similar to Google Street View, but with up-to-date videos instead of still images, collected through large-scale crowdsourcing.

Collecting videos on such a massive scale poses a number of challenges in terms of network planning. Two challenges were tackled in this work. The first challenge is to obtain a scalable distributed architecture for the crowdsourcing framework. This architecture was obtained by extending the classical cloud architecture to a two-layer cloudlet architecture. Uploading videos to cloudlets close to the user rather than a distant cloud consumes less total bandwidth and makes the framework more scalable.

The second challenge is posed by the access network of the cloudlets, which only has limited upload bandwidth available. We presented and evaluated several decision-algorithms that are used to filter out less relevant videos when the access network to a cloudlet becomes overloaded. These algorithms are all based on the same twofold structure, namely a compact way to represent the collection of videos and a local scoring function that is applied to a local segment of this representation. Through simulation, algorithm parameters were optimized and a comparison between the different optimized algorithms was made. The best performing algorithm using a local representation performed at least as well in selecting videos as a more obvious algorithm which required the entire representation. Moreover, execution times were three order of magnitude lower and scaled better with video collection size.

A number of extensions to this work are possible. In the simulations of the decision algorithms, more realistic load patterns could be used. The upload behavior of large groups of users can for example be simulated by using messages on social network sites such as Flickr, Twitter and Facebook, as a large number of these messages contain geotags. Furthermore, the decision algorithms don't take into account the restrictions on the available storage. These should be taken into account by extending the algorithm to select a set of stored videos to be removed to make way for videos with a higher score.

8. ACKNOWLEDGMENTS

Steven Bohez is funded by Ph.D. grant of the Agency for Innovation by Science and Technology in Flanders (IWT). Tim Verbelen is funded by Ph.D. grant of the Fund for Scientific Research, Flanders (FWO-V). This project was partly funded by the UGent BOF-GOA project "Autonomic Networked Multimedia Systems".

9. REFERENCES

[1] Apache. Felix. [Online]. Available: http://felix.apache.org. [Accessed: 30-Jul-2014].

[2] E. Cuervo, P. Gilbert, B. Wu, and L. Cox. Crowdlab: An architecture for volunteer mobile testbeds. In *Communication Systems and Networks (COMSNETS), 2011 Third International Conference on*, pages 1–10, Jan 2011.

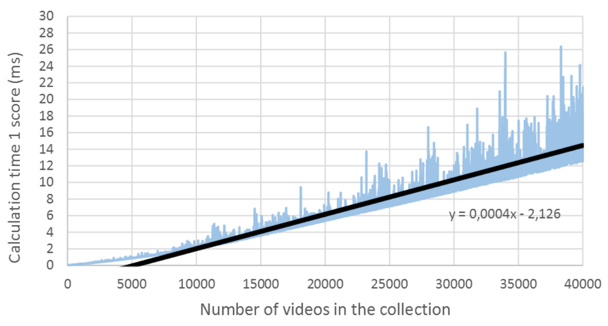
[3] Eclipse. Concierge. [Online]. Available: http://projects.eclipse.org/projects/rt.concierge. [Accessed: 20-Jul-2014].

[4] Gartner. Gartner says worldwide traditional pc, tablet, ultramobile and mobile phone shipments are on pace to grow 6.9 percent in 2014. [Online]. Available: http://www.gartner.com/newsroom/id/2692318. [Accessed: 15-Jul-14].

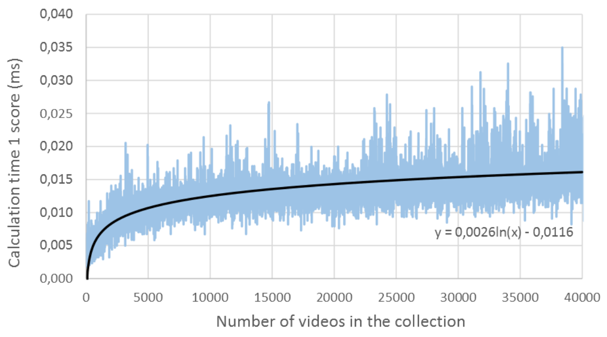
[5] Google. Create your own street view. [Online]. Available: http://google-latlong.blogspot.fr/2013/12/create-your-own-street-view.html. [Accessed: 13-Aug-2014].

[6] K. Hara, V. Le, and J. Froehlich. Combining crowdsourcing and google street view to identify street-level accessibility problems. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '13, pages 631–640, New York, NY, USA, 2013. ACM.

[7] 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.



(a) Average Distance



(b) Hierarchical histogram with age-based local score

Figure 10: While the Average Distance algorithm scales linearly with an increasing video collection size, the hierarchical histogram algorithm combined with the age local scoring function scales logarithmically.

[8] OSGi Alliance. Open services gateway initiative. [Online]. Available: http://www.osgi.org. [Accessed: 15-Jul-2014].

[9] M.-R. Ra, B. Liu, T. F. La Porta, and R. Govindan. Medusa: A programming framework for crowd-sensing applications. In *Proceedings of the 10th International Conference on Mobile Systems, Applications, and Services*, MobiSys '12, pages 337–350, New York, NY, USA, 2012. ACM.

[10] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies. The case for VM-based cloudlets in mobile computing. *IEEE Pervasive Computing*, 8(4):14–23, Oct. 2009.

[11] P. Simoens, Y. Xiao, P. Pillai, Z. Chen, K. Ha, and M. Satyanarayanan. Scalable crowd-sourcing of video from mobile devices. In *Proceeding of the 11th annual international conference on Mobile systems, applications, and services (MobiSys '13)*, page 139, New York, New York, USA, June 2013. ACM Press.

[12] U. Soderstrom and H. Li. Anonymous video processing for live street view. In *Internet of Things (iThings/CPSCom), 2011 International Conference on and 4th International Conference on Cyber, Physical and Social Computing*, pages 109–113, Oct 2011.

[13] T. Verbelen, P. Simoens, F. De Turck, and B. Dhoedt. AIOLOS: middleware for improving mobile application performance through cyber foraging. *Journal Of Systems And Software*, 85(11):2629–2639, 2012.

[14] T. Verbelen, P. Simoens, F. De Turck, and B. Dhoedt. Cloudlets: bringing the cloud to the mobile user. In *Proceedings of the 3rd ACM Workshop on Mobile Cloud Computing and Services*, pages 29–35. ACM Press, 2012.

[15] Y. Xiao, P. Simoens, P. Pillai, K. Ha, and M. Satyanarayanan. Lowering the barriers to large-scale mobile crowdsensing. In *Proceedings of the 14th Workshop on Mobile Computing Systems and Applications*, HotMobile '13, pages 9:1–9:6, New York, NY, USA, 2013. ACM.

[16] YouTube. Statistics. [Online]. Available: http://www.youtube.com/yt/press/statistics.html. [Accessed: 29-Jul-14].