Energy-Efficient Mobile Video Management using Smartphones

Jia Hao[‡], Seon Ho Kim[§], Sakire Arslan Ay[†], Roger Zimmermann[‡]
[‡]School of Computing, National University of Singapore, Singapore 117417
[§]Integrated Media Systems Center, University of Southern California, Los Angeles, CA 90089
[†]Department of Computer Science, University of Southern California, Los Angeles, CA 90089
haojia@comp.nus.edu.sg, seonkim@usc.edu, arslan@usc.edu, rogerz@comp.nus.edu.sg

ABSTRACT

Mobile devices are increasingly popular for the versatile capture and delivery of video content. However, the acquisition and transmission of large amounts of video data on mobile devices face fundamental challenges such as power and wireless bandwidth constraints. To support diverse mobile video applications, it is critical to overcome these challenges. We present a design framework that brings together several key ideas to enable energy-efficient mobile video management applications. First, we leverage off-the-shelf smartphones as mobile video sensors. Second, concurrently with video recordings we acquire geospatial sensor meta-data to describe the videos. Third, we immediately upload the meta-data to a server to enable low latency video search. This last step allows for very energy-efficient transmissions, as the sensor data sets are small and the bulky video data can be uploaded on demand, if and when needed. We present the design, a simulation study, and a preliminary prototype of the proposed system. Experimental results show that our approach substantially prolongs the battery life of mobile devices while only slightly increasing the search latency.

Categories and Subject Descriptors

H.3.4 [**Information Storage and Retrieval**]: Systems and Software—*Distributed Systems*; I.4.8 [**Image Processing and Computer Vision**]: Scene Analysis—*Sensor Fusion*

General Terms

Algorithms, Measurement, Performance

Keywords

Mobile video, geotagging, video search, energy efficiency

1. INTRODUCTION

The influx of affordable, portable, and networked video cameras has made various video applications feasible and practical. Furthermore, the combination of mobile cameras with other sensors has extended plain video sensor networks to wireless multimedia sensor networks. These are expected to manage far more and diverse information from the real world because videos with associated scalar sensor data can be collected, transmitted, and searched to more effectively support a wide range of multimedia applications. These include both conventional and emerging applications such as multimedia surveillance, environmental monitoring, industrial process control, and location based multimedia services [1]. As a result, various mobile devices, sensors, networks, and multimedia search schemes have been designed and tested to implement such systems.

Traditionally, any extensive sensor networks that have been constructed with expensive, custom hardware and network architecture work for specific applications only, leading to limited use. However, with rapid advances in communication and cellular phone technologies, smartphones have emerged as a possible off-the-shelf choice of mobile devices since they can satisfy most technical requirements of multimedia sensor networks, such as video capturing with high resolution, meta-data collection from various sensors, communication capabilities with widely available WiFi networks, and true handheld mobility. For example, smartphones such as Apple's iPhone 3GS and 4 and Motorola's Droid have a quality camera, a GPS receiver, a digital compass, an accelerometer, and considerable computing power.

Mobile multimedia applications have inherited the typical challenges of mobile computing such as capacity constraints of the battery and wireless bandwidth bottlenecks. Considering that both the video capture and wireless transmission of large amounts of video data with mobile devices are highly power intensive, it is fundamental to efficiently manage the battery power. Furthermore, mobile video applications introduce new challenges such as the searchability of online videos, especially in large scale applications, because open-domain video content is very difficult to be efficiently and accurately searched.

There are currently two prevalent approaches to make video content searchable. First, there is a significant body of research on content-based video retrieval, which employs techniques that extract features based on the visual signals of a video. While progress has been very significant in this area, the semantic gap between identifying the low-level features and recognizing important semantic themes in the videos is still wide [14]. Achieving high accuracy with these techniques is often limited to specific domains such as sports or news content, and applying it to large-scale video

MMSys'11, February 23–25, 2011, San Jose, California, USA.

Hardware	Description	Parameter	Coefficient (C_j)	Range (of β_j)
CPU	Qualcomm® MSM7201A™, 528 MHz	CPU_hi	$C_{CPU\_hi} = 3.97$ mW/%	$\beta_{CPU\_hi} : 0 - 100\%$
		CPU_lo	$C_{CPU\_lo} = 2.79$ mW/%	$\beta_{CPU\_lo} : 0 - 100\%$
Screen	3.2-inch TFT-LCD flat, touch-sensitive screen with 320×480 (HVGA) resolution	LCD	$C_{LCD} = 150$ mW	$\beta_{LCD} : 0, 1$
		$Brightness$	$C_{br} = 2.07$ mW/step	$\beta_{br} : 0 - 255$ steps
WiFi	Texas Instruments WL 1251B network chipset	$WiFi\_on$	$C_{WiFi\_on} = 39$ mW	$\beta_{WiFi\_on} : 0, 1$
		$WiFi\_trf$	$C_{WiFi\_trf} = 658.93$ mW	$\beta_{WiFi\_trf} : 0, 1$
		$WiFi\_bytes$	$C_{WiFi\_bytes} = 0.518$ mW/byte	$\beta_{WiFi\_bytes} : \geq 0$
Storage	MicroSD memory flash card	SD	$C_{SD} = 0.0324$ mW/sector	$\beta_{SD} : \geq 0$
GPS	GPS receiver	GPS	$C_{GPS} = 430$ mW	$\beta_{GPS} : 0, 1$
System	Residual system power consumption in addition to the above components	$System$	$C_{System} = 169.08$ mW	$\beta_{System} : 0, 1$

Table 1: Parameters of the HTC G1 smartphone used in the power model.

repositories creates significant scalability problems. The second approach utilizes searchable text annotations associated with the video content; however high-level concepts must often be added manually, rendering this method ineffective for large video repositories. Furthermore, these text annotations can be ambiguous and subjective.

Recent technological trends have opened another avenue that fuses much more accurate, relevant contextual information with videos: the concurrent collection of sensor-generated geospatial meta-data. The aggregation of multi-sourced geospatial data into a standalone meta-data tag allows video content to be identified by a number of precise, objective geospatial characteristics. For example, current-generation smartphones have GPS receivers, compasses, and accelerometers all embedded into a small, portable, energy-efficient package. When aggregated, the resulting meta-data can provide a comprehensive and easily identifiable model of a video's viewable scene, which can support a scalable organization, search, and streaming of large scale video repositories.

In the presence of such meta-data, there are two conventional ways to transmit both meta-data and video jointly from a mobile device: (1) immediate transmission after capturing through wireless network, and (2) delayed transmission when a faster network is available. The former can provide immediate availability of the data to users while consuming lots of battery energy and scarce wireless bandwidth. The latter consumes the minimum power while sacrificing real-time access to the captured videos. Thus, both approaches are not very appealing.

Employing smartphones as the choice of mobile devices, we propose a new framework to support an efficient mobile video capture and their transmission as shown in Figure 3. Based on the important observation that not all collected videos have high priority (i.e., many of them will not be requested and viewed immediately), the core of our approach is to separate the small amount of text-based geospatial meta-data of concurrently captured video content from the large binary-based video content. This small amount of meta-data is then transmitted to a server in real-time, while the video content will remain on the recording device, creating an extensive, resource efficient catalogue of video content, searchable by viewable scene properties established from meta-data attached to each video. Should a particular video be requested, only then will it be transmitted from the camera to the server in an on-demand manner (preferably, only the relevant segments, not the entire videos). The delivery of unrequested video content to a server can be delayed until a faster connection is available.

This paper presents the design, a simulation study, and a mobile device prototype implementation of an energy efficient mobile video management system. Our simulation results show that the proposed approach can significantly reduce the energy consumption of a smartphone while still providing a satisfactory service latency when videos are requested. Overall, the system achieves a balance between resource demands and quality of service.

The remainder of this paper is organized as follows. In Section 2, we present a mobile device power model. Section 3 describes the system design. In Section 4, a simulator is introduced and we evaluate our system in terms of energy and bandwidth efficiency, query response latency, and result completeness. Section 5 outlines the implementation of a device acquisition prototype. In Section 6, we summarize the related prior work. Finally, Section 7 concludes the paper.

2. POWER MODEL

We define an estimation model to describe the power levels of a mobile device operating under different modes. Our target device is the HTC G1, a smartphone that is based on the open source Google Android mobile device platform [8].

2.1 Modeled Hardware Components

We adapted the power estimation model introduced by Shye *et al.* [18]. They proposed a linear-regression-based power estimation model, which uses high-level measurements of each hardware component on the mobile device, to estimate the total system power consumption. In our work, we used this model at the device level to understand and evaluate the efficiency and feasibility of our proposed video search technique.

We next describe the relevant details of each hardware component on the target HTC G1 mobile phone. Table 1 lists the G1 hardware components that were considered in the power model and their corresponding parameters. In Table 1, C_j coefficients are the final regression coefficients obtained for the chosen G1 hardware. Our search system incorporates an additional GPS receiver unit to obtain location meta-data. Therefore, we modified the original model and included the power consumption for the GPS receiver. For simplicity, we excluded the power consumption for the Call, EDGE and DSP units.

CPU: The processor supports dynamic frequency scaling (DFS) and it is rated at 528 MHz, but is scaled down in the

platform to run at 124 MHz, 246 MHz, and 384 MHz. The highest frequency of 528 MHz is not used. The lowest frequency is never used on consumer versions of the phone, and is too slow to perform basic tasks. Thus, only the high (384 MHz) and medium (246 MHz) frequencies are considered in the model. CPU power consumption is strongly correlated with the CPU utilization and frequency. In Table 1, the CPU_hi and CPU_lo parameters represent the average CPU utilization while operating at 384 MHz and 246 MHz, respectively.

SCREEN: The display is described by two parameters: a boolean parameter LCD indicating whether the screen is on or off and a $Brightness$ parameter which models the effect of the screen brightness with 256 uniformly spaced levels.

WIFI: The boolean parameter $WiFi\_on$ describes whether the WiFi network interface is turned on or off; additionally $WiFi\_trf$ and $WiFi\_bytes$ indicate network traffic and the number of bytes transmitted during a particular time interval.

STORAGE: The number of sectors transferred to or from the MicroSD flash memory card per time interval are represented by the parameter SD.

GPS: The boolean parameter GPS denotes the power consumption coefficient when the GPS receiver is on.

SYSTEM: There exists also a residual power consumption parameter $System$. This parameter subsumes all power that is not accounted for the hardware components listed above. We refer to this as the baseline $System$ power in Table 1.

2.2 Analytical Power Model

The described modeling parameters are incorporated into the analytical power model that is utilized in our simulation experiments. The power model determines the relationship between the system statistics (e.g., the value for screen brightness) and the power consumption for each relevant hardware component. The inputs to the model are the statistics collected from the device (β_j values), and the output represents the total power consumption. The overall system power consumption as a function of time t is determined as follows:

$$\begin{aligned} P(t) = (C_{CPU\_hi} \times \beta_{CPU\_hi}(t)) + (C_{CPU\_lo} \times \beta_{CPU\_lo}(t)) + \\ (C_{LCD} \times \beta_{LCD}(t)) + (C_{Brightness} \times \beta_{br}(t)) + \\ (C_{WiFi\_on} \times \beta_{WiFi\_on}(t)) + (C_{WiFi\_trf} \times \beta_{WiFi\_trf}(t)) + \\ (C_{WiFi\_bytes} \times \beta_{WiFi\_bytes}(t)) + (C_{SD} \times \beta_{SD}(t)) + \\ (C_{GPS} \times \beta_{GPS}(t)) + (C_{System} \times \beta_{system}(t)) \end{aligned}$$

The ranges for the β_j values are listed in Table 1. The overall power consumption is calculated by substituting the statistics collected at time t for the selected hardware components into $P(t)$.

2.3 Validation of the Power Model

To evaluate the accuracy of our power model, we measured the power consumption of an HTC G1 with *PowerTutor* [19], an application for Android-based phones that displays the power consumed by major system components such as CPU, network interface, display, and GPS receiver (see Figure 1). According to the authors, PowerTutor was developed on the HTC G1 in collaboration with Google, and its accuracy should be within 5% of actual values for the G1.

With PowerTutor we obtained the various β-statistics for different hardware units. Specifically, we collected logs on

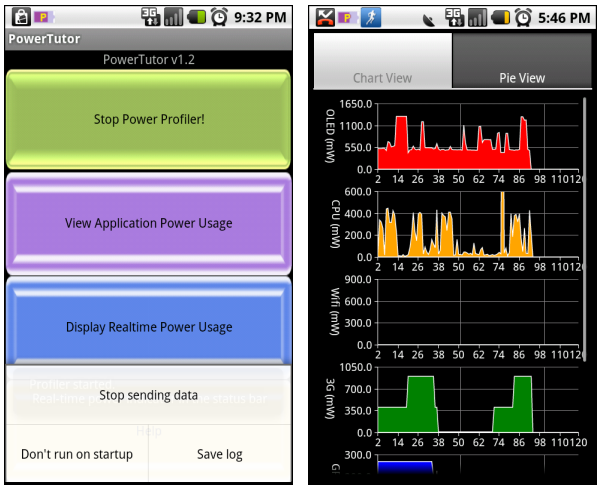


Figure 1: Screenshot of the PowerTutor.

Hardware	Parameter	Video Capture	WiFi Transmission
CPU	β_{CPU_hi}	77.45	77.33
	β_{CPU_lo}	0	0
LCD	β_{LCD}	1	0
	β_{br}	102	0

Table 2: β-parameters under different operational modes.

a G1 phone for different usage scenarios. For instance, we captured video for one minute, or uploaded video for one minute, and so on. During these tests, all non-essential processes were disabled. After multiple experiments, we determined the values shown in Table 2.

In the next step, the measured parameters were substituted into our power model. We then performed the same usage scenarios with the Android G1 phone for about two minutes, and collected the trace logs from PowerTutor. We measured the power consumption for various phone usage scenarios such as capture+GPS (capturing video and using GPS to obtain location information), capture+WiFi (capturing video and using WiFi to obtain the location), capture+GSM (capturing video and using GSM to obtain the location), and transmission+WiFi (transmitting data via WiFi). Grouped by usage scenario, the average power consumption obtained from the power model was compared to the power values reported by PowerTutor. The results are shown in Figure 2.

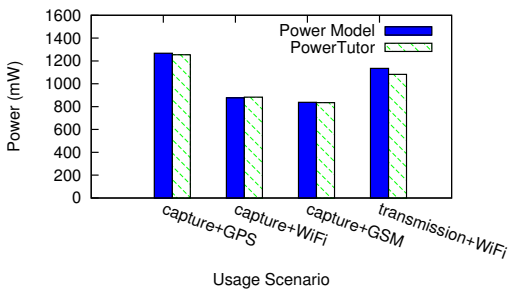


Figure 2: Comparison of the results from the power model with logs from PowerTutor.

The modeled and measured power consumptions match very well for each of the usage scenario. To calculate the accuracy of the model, we used the following error metric e:

$$e = \left| \frac{P_{measured} - P_{modeled}}{P_{measured}} \right| \quad (1)$$

The results indicate that the power estimation model accurately predicts the system-level power consumption. The error e for each scenario is less than 4.9%, and the average error across all the scenarios is 1.7%.

An important point to note is that capturing video and then transmitting it through WiFi are both very energy-consuming activities. With its standard 1,150 mAh-capacity battery, the G1 phone would last less than three hours in the worst case, when continuously capturing and transmitting video. Our proposal is to extend battery life through more selective transmissions.

3. SYSTEM DESIGN

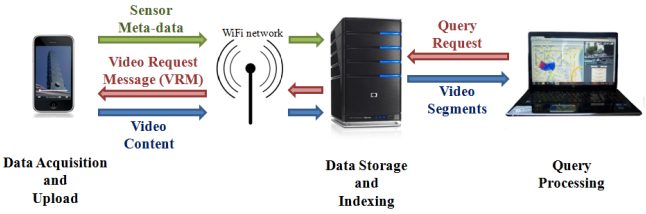


Figure 3: System environment for mobile video management.

Figure 3 shows an overview of the proposed system. Mobile nodes collect the videos and the sensor-associated meta-data such as GPS location, compass direction, capture time and other camera-related information. The video files remain locally on the device until requested while the meta-data are immediately uploaded to the server in real-time where they are stored and indexed in a database. In a typical search scenario, other users (e.g., observers) can query the videos that are being captured from many devices in real-time or near real-time. We assume that a user provides a query as a geographical region of interest. The video meta-data stored on the server are searched to identify and retrieve the video clips that show the requested query region and the search results are presented to the user. During query processing, the video content already available on the server is immediately sent to the user for viewing while the missing video segments are requested *on demand* from the mobile devices that captured the videos. Note that only the precisely delimited parts (i.e., only the video segments that actually overlap with the query region) are retrieved. The complete video content may be uploaded later when the device is in contact with a faster network connection.

The key idea of this approach is to save considerable battery energy by delaying the costly transmission of the large binary video data that have not been requested, especially when the transmission speed is low. We will describe the components of the proposed system next.

3.1 Data Acquisition and Upload

A camera positioned at a given point P in geo-space captures a scene whose covered area is referred to as the camera field-of-view (FOV, also called the *viewable scene*). We adapt the FOV model introduced in our prior work [2], which describes a camera's viewable scene in 2D space with four parameters: camera location P, camera orientation α, viewable angle θ and visible distance R (see Eqn. (2)).

$$FOV \equiv \langle P, \alpha, \theta, R \rangle \quad (2)$$

The camera position P consists of the latitude and longitude coordinates read from a positioning device (e.g., GPS) and the camera direction α is obtained based on the orientation angle provided by a digital compass. R is the maximum visible distance from P at which a large object within the camera's field-of-view can be recognized. The angle θ is calculated based on the camera and lens properties for the current zoom level [9]. The collected meta-data streams are analogous to sequences of $\langle nid, vid, t_{FOV}, t_f, P, \alpha, \theta, R \rangle$ tuples, where nid represents the ID of the mobile device, vid is the ID of the video file and t_{FOV} indicates the time instant at which the FOV is recorded. The timecode associated with each video frame is denoted by t_f.

In 2D space, the field-of-view of the camera at time t_{FOV} forms a pie-slice-shaped area as illustrated in Figure 4.

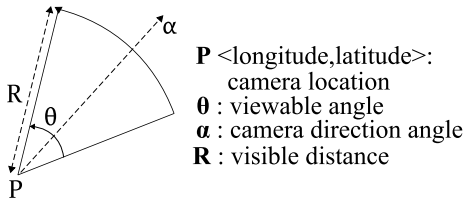


Figure 4: Illustration of FOV in 2D space.

When a mobile device begins video capture, the GPS and compass sensors are turned on to record the location and orientation of the camera. Our custom-written data-acquisition software fetches such sensor values as soon as new values are available. Video data are processed in real time to extract frame timecodes (t_f). The visible distance R is calculated based on the camera specifications. All collected meta-data (i.e., location, direction, viewable distance, frame timecode and video ID) are combined as a tuple and uploaded to the server.

An appropriate meta-data upload rate should be determined such that the server is updated immediately for real-time video search while the energy consumption for meta-data uploads is minimized. Two policies are possible. First, the system may send the meta-data whenever it is generated. Second, it may buffer the meta-data locally and then send the accumulated data periodically. Such meta-data aggregation and delivery may utilize available network bandwidth more efficiently. For the first policy, since meta-data is always ready to be uploaded, we assume that the WiFi interface is always on when recording. Whereas for the second policy, WiFi will be turned on and off periodically. Some startup energy is consumed when WiFi is turned on. As Cheung *et al.* measured [5], we set the startup energy as 6.47 J. We will further discuss this aspect in Section 4.

Another issue we would like to explore is energy-efficient collection of location meta-data. GPS, WiFi and GSM pose a challenging tradeoff between localization accuracy and energy consumption. While GPS offers good location accuracy of around 10 m, it incurs a serious energy cost that can drain a fully charged phone battery very fast. WiFi and GSM-

based schemes are less energy-hungry, however, they incur higher localization errors (approximately 40 m and 400 m, respectively). In our work we employ the GPS-based and GPS-save strategies. GPS-based scheme refers to sampling GPS data periodically, while GPS-save uses a more complicated strategy. When the device orientation change is within a limited range, we assume that the device user does not change his/her moving direction, and the GPS receiver is turned off to save energy. Once the direction changes, the GPS receiver is turned on, reporting the current location. When meta-data with two consecutive GPS data points is uploaded, we can interpolate the device location between the two GPS locations on the server. With this method considerable energy can be saved. More details can be found in Section 4.

3.2 Data Storage and Indexing

This module implements a storage server that manages the video files and the associated meta-data streams. It separately stores the video content and the meta-data. The video files are linked to the the meta-data streams by device ID (*nid*) and video ID (*vid*). Each FOV tuple in a meta-data stream includes a frame timecode t_f that points to a particular frame within the video content. This ensures a tight synchronization between the two streams.

The server keeps a data structure $nodeInfo$ for each mobile node, which includes the device MAC address, the unique device ID, and the IP address. While the storage server receives the meta-data from mobile devices, *nid* is added automatically to each FOV tuple. An additional binary tag (*inServer*) is maintained for each FOV tuple indicating whether the corresponding binary data of the video frame exists or not on the server. Spatial indices are built and maintained to facilitate the efficient search of FOVs.

3.3 Query Processing

When a user issues a query, the video meta-data in the server is searched to retrieve the video segments whose viewable scenes overlap with the geographical region specified in the query. The query region can be a point, a line (e.g., a road), a poly-line (e.g., a trajectory between two points), a circular area (e.g., neighborhood of a point of interest), a rectangular area (e.g., the space delimited with roads) or a polygon area (e.g., the space delimited by certain buildings, roads and other structures). In our initial prototype we only support rectangular queries.

Given a query Q, the query processing module returns a list of the video segments whose corresponding FOVs overlap with the query Q. Each video segment is identified with a tuple $\langle nid, vid, t_{start}, t_{end} \rangle$, where t_{start} and t_{end} are the timecodes for the first and last FOVs.

For each video segment in the query results, the query processor checks for the availability of the corresponding video content on the server. Recall that, the storage server keeps track of which video files are uploaded to the server and what parts of the meta-data they do belong to. For the FOVs with the *inServer* field set to 1, the corresponding video content is available on the server. And conversely, for those with the *inServer* field equal to 0 the video content is not available and therefore needs to be requested from the capturing mobile device. To acquire a missing video segment, a Video Request Message (VRM) is sent to the mobile device. A VRM message specifies the IP address of the target mobile device as well as the corresponding video ID and the beginning and ending timecodes for the requested video segment.

If the requested video with video ID *vid* is still available on the mobile device, the video segment from t_{start} to t_{end} is uploaded to the storage server. The *inServer* tags for the corresponding FOVs are set to 1. However, if the requested video cannot be located, the mobile device notifies the query processor by sending a Video does not Exist Message ($VNEM$). If no response is received from the device after n trials, the device is assumed to be turned off and the VRM message is dismissed. If the query processor can locate the videos for the search results on the server, it immediately sends the video data to the user. The video segments requested from the mobile devices are sent as soon as they arrive at the server.

4. EXPERIMENTAL EVALUATION

To evaluate our framework we implemented an extensive simulator and executed it on a server with two 4-core Intel(R) Xeon(R) X5450 3.0 GHz CPUs and 16 GB of memory, running Linux 2.6.18.

4.1 Simulator Operation

We first provide an overview of the operation of the simulator before describing its internal details. We assume an urban wireless communication infrastructure where mobile users are moving on the road network of the city of San Francisco. The users capture and transmit videos with predefined simulation models. Similarly, some other users launch queries to retrieve the collected videos from the same region.

The simulated space is approximately 14.3 km × 13.6 km in size. Within this area, N_{node} mobile users and N_{AP} WiFi network access points are distributed. The simulation proceeds in discrete time steps ts (5 s each) for a total duration of T. During T, two types of events occur: video *capture events* and *query events*. In a capture event, each mobile node independently starts to record video and the recording duration follows a log-normal distribution. The capture event arrival rate is λ_c per timestamp ts. Queries are issued by observers and sent to the server with a query event arrival rate of λ_q per timestamp ts. Queries are assumed to be distributed within the simulation space either uniformly random, or skewed with a clustering parameter h ($0 \leq h \leq 1$). When an area is frequently queried, it is regarded as a "popular area". The h parameter represents the popularity of the area. A higher value of h denotes that more queries are requested from that area. The query clustering is designed to emulate areas of interest in the real world. The query size M_q is chosen as a small fraction of the simulation space. The simulation parameters are summarized in Table 3.

Captured videos are either (1) immediately and completely uploaded to the server (Immediate) at a transmission rate that is determined by the mobile node's proximity to an access point, or (2) alternatively, video upload is delayed and only videos where a query request overlaps with the region that was captured in the video will be uploaded in an on-demand manner (OnDemand). Importantly, only the video segments, not an entire video clip, that overlap with the query are transmitted. The query response latency for $L_{Immediate}$ is assumed to be zero (or close to zero), since the data is readily available on the server and can be immediately returned to the observer. With the OnDemand policy, the relevant video segments must be requested and uploaded

Module	Parameter	Description	Values
Network Topology Generator	w	WiFi type used	**802.11b**
	N_{AP}	Number of access points	400, 800, **1,000**, 1,200, 1,600, 2,000
		Simulation space	14.295 km × 13.623 km
Node Trajectory Generator	N_{node}	Number of mobile nodes	**2,000**
	T	Simulation time in 1,000s	**150**
	ts	Timestamp in seconds	**5**
Query Generator	λ_q	Query event arrival rate (per timestamp ts)	0.1 – 0.9, **0.5**
	M_q	Mean query rectangle size as a percentage of the map area	0.01 – 0.05, **0.03**
	h	Query rectangle clustering parameter, see Figure 6	0 – 1, **0.5**
FOV Generator	λ_c	Capture event arrival rate (per timestamp ts)	0.01 – 0.04, **0.02**
	D_c	$(e^{D_c} \times ts)$ is the mean duration of a capture event	0.5 – 2.5, **1.5**
Power Model		Mobile device power parameters	See Table 1

Table 3: Simulation parameters (values in bold are the default settings).

to the server before the query request is considered satisfied. The worst case response latency of OnDemand, $L_{OnDemand}$, is hence determined by the worst case upload time (we assume that all uploads start concurrently and in parallel from the nodes involved).

Using our simulation testbed we evaluate the Immediate and OnDemand strategies based on following three main metrics: the energy consumption (and hence the lifetime) of mobile nodes, the query response latency, and the total amount of data transmitted to satisfy all queries.

4.2 Simulator Architecture and Modules

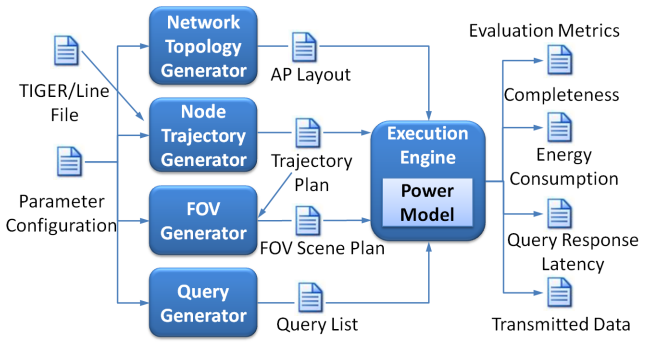


Figure 5: The block diagram of the simulator architecture.

Figure 5 provides a detailed architectural view of the simulator with the following components: generators for (1) the network topology, (2) the node trajectories, (3) FOV viewable scenes, and (4) queries. Additionally, the (5) power model is a part of the (6) execution engine.

Network Topology Generator: A number of WiFi access points (N_{AP}) are uniformly distributed in the simulation area and this module emulates the mobile access range with the realistic Auto-Rate Fallback (ARF) mechanism of WiFi, which provides a number of declining transmission rate levels with an increase in the distance between the access point and a mobile client. We implemented and tested both the 802.11b and 802.11g standards for our simulation. However, we found that the results with 802.11g follow the same trend as with the slower standard (i.e., the transmission times are proportionally reduced). Hence we present only the results for 802.11b.

Node Trajectory Generator: We use the Brinkhoff generator to produce movements of mobile objects along a road network [3]. The input to the generator consists of a TIGER/Line road network file of the city of San Francisco from the U.S. Census Bureau. The output is a set of objects that move on the road network of the city.

FOV Generator: This module synthesizes the mobile nodes' recording behavior and generates the representations of nodes' viewable scenes. The recording start times follow an exponential distribution based on the capture event arrival rate λ_c (expressed in events per timestamp ts, see Table 3). The duration of recordings is log-normally distributed. The FOV generator obtains the camera location information from the trajectory generator and, for the camera direction, a random orientation is generated for each node's viewable scenes. The maximum visible distance R is set to 200 m.

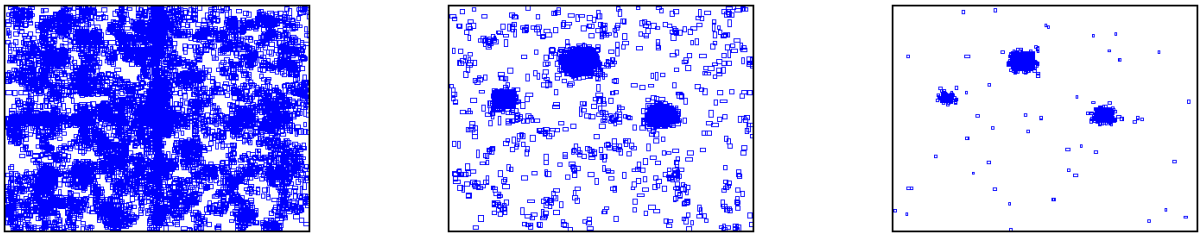
Query Generator: The query workload consists of a list of query rectangles that are mapped to specific locations in the simulation space. The query arrival interval is exponentially distributed with λ_q (measured per timestamp ts, see Table 3). The rectangle size is determined by a normally distributed random variable with the mean value M_q. The parameter h is used to generate different distributions of queries in the given space to evaluate the performance of our proposed system framework and test its robustness with different clusterings, both spatially and temporally. Figure 6 shows three spatial query distributions with different values of h. As can be seen, the larger the value of h, the more clustered the queries are.

Power Model: Power consumption is modeled for each node based on the specifications presented in Section 2. The power model is embedded in the execution engine so that mobile nodes' battery life can be updated during each time step. The power level for the mobile nodes in different states is summarized in Table 1.

Execution Engine: The simulation is executed after reading in the access point (AP) layout, the trajectory plan, the FOV scene plan, and the query list. The engine then simulates the movement of the mobile nodes within the simulation space, keeps track of their video recordings, executes the queries, and manages the simulation status. At every timestamp, the engine computes all the evaluation metrics.

4.3 Experiments and Results

The utility of a mobile device depends on the duration of operational hours before its battery needs re-charging. Thus, one of the key metrics we use to evaluate the en-



(a) $h = 0$. (b) $h = 0.5$. (c) $h = 1$.

Figure 6: Spatial query distribution with three different clustering parameter values h.

ergy efficiency of our approaches is the expected reduction in power consumption, which directly translates into an extended battery lifetime. We further evaluate the query response latency that a user experiences when searching for real-time mobile video. Finally, mobile bandwidth is still a relatively scarce resource, especially in scenarios where the infrastructure may be limited (e.g., after a disaster). Hence we also calculated the overall size of the video data that must be transmitted to satisfy all queries.

Through simulations our OnDemand approach is compared to Immediate in terms of power-efficiency, latency to obtain the results, overall bandwidth use and result completeness. Recall that using the Immediate approach nodes upload videos without delay, therefore, we assume that the query response latency is zero for this method.

4.3.1 Performance: Without Battery Recharging

In our first experiment we assume a closed system where batteries cannot be recharged. Hence, all the nodes will eventually run out of energy and cease operation. We are interested in the system lifetime and the query completeness under these conditions. The simulation area is populated with $N_{node} = 2,000$ mobile nodes. We track all the nodes' energy consumption and battery levels. The default simulation parameters of Table 3 are used.

Figure 7(a) illustrates that the OnDemand method consumes considerably less energy than the Immediate method, and the lifetime of the last alive node is prolonged from about 84,000 s ($\approx$ 23.3 h) to 120,000 s ($\approx$ 33.3 h). The query workload imposed by the observing users leads to an uneven utilization of the nodes and some deplete their batteries earlier than others.

Once some nodes begin to cease operation, the results of queries may increasingly become *incomplete* because the requested data become unavailable. Figure 7(b) compares the completeness of the query results returned by the two strategies. We compute the completeness of results from both methods every one thousand timesteps using the following fraction: (video segments actually returned)/(video segments that should be returned). To compute the video segments that should be returned, we assume an ideal baseline case in which there are no battery constraints. Hence, all requested video segments are never missed. Figure 7(b) shows that as time advances, the completeness of results of both methods decreases because the number of alive nodes decreases and the number of videos uploaded to the server also declines. However, the downward trend of OnDemand begins later in time because the nodes with OnDemand last longer. This is directly attributable to the energy saved from fewer unnecessary video transmissions, so mobile nodes retain more battery energy to capture additional videos.

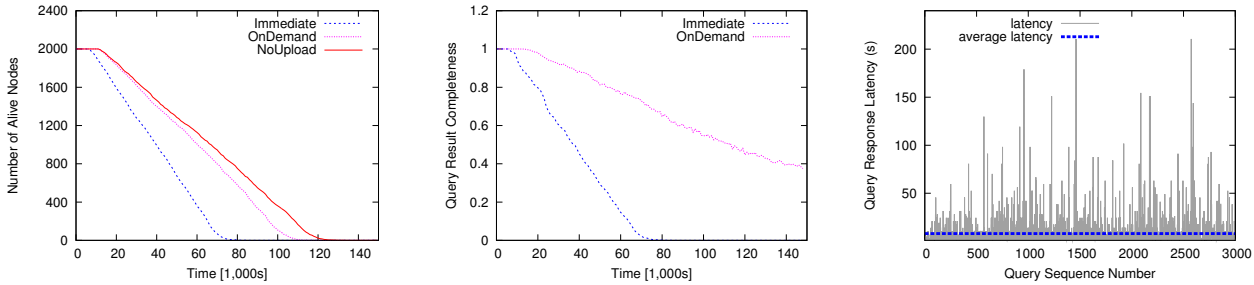
Figure 7(c) shows the worst case query response latency with OnDemand. The average query response latency is 8.08 s while there are some exceptionally long latency. It should be noted that the latency represented here straightforwardly refers to the duration from the time when a query is initiated by a user to the time when the last frame of the latest arriving result video segment has been received. With smarter streaming techniques some of the video segments can be browsed much earlier, which may significantly reduce the effective response latency observed by users. The figure shows some comparatively large values which are due to some "lazy" mobile nodes (a property of the node mobility model). These nodes almost do not move during the simulation, but they keep capturing video and uploading meta-data. For a long time initially, the videos captured by these nodes may not be queried. However, once a query arrives, many videos are to be uploaded consecutively, which causes a considerable delay.

4.3.2 Performance: With Battery Recharging

In some large-scale application scenarios batteries can be recharged or replaced so that the mobile node density will eventually reach a dynamic equilibrium. In the steady state, nodes continuously join and leave (i.e., their batteries run out). We evaluate the Immediate and OnDemand approaches in this scenario using the proposed metrics.

First, we would like to determine the appropriate rate for meta-data uploading. Figure 8 shows the tradeoff between energy consumption over an entire simulation and access latency with varying meta-data rate, λ_s. The access latency represents the average duration from the time meta-data are produced until the time a user is able to search the data (i.e., the meta-data become available on the server). Since the size of the meta-data file is very small, we can effectively ignore its transmission time. When λ_s grows large, the meta-data upload period ($1/\lambda_s$) approaches near zero and the node sends the meta-data whenever it is produced. In this case, the mobile device continuously uploads the meta-data and it will not turn off the WiFi interface. Therefore, the mobile device will continuously consume a certain amount of power (i.e., 658.93 mW) for the meta-data transmission.

In general, the node sends meta-data every $1/\lambda_s$ seconds. To save energy when using WiFi, the mobile device can turn off the WiFi interface while it is not transmitting data. During this transition from the off to the on state, a startup energy of 6.47 J will be consumed. Thus a higher number of meta-data transitions means more startup energy over-



(a) Number of nodes alive. (b) Query result completeness (PDF). (c) Query response latency.

Figure 7: Node lifetimes (i.e., energy efficiency), result completeness, and query response latency with $N = 2,000$ nodes.

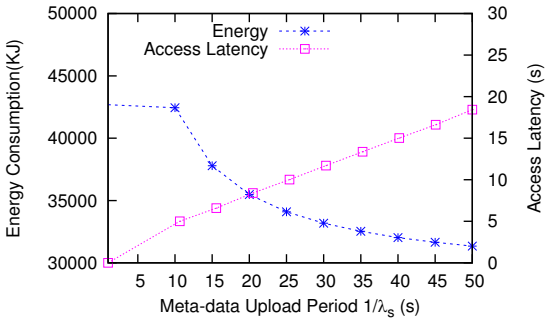


Figure 8: Energy consumption and access latency with varying meta-data upload period ($1/\lambda_s$).

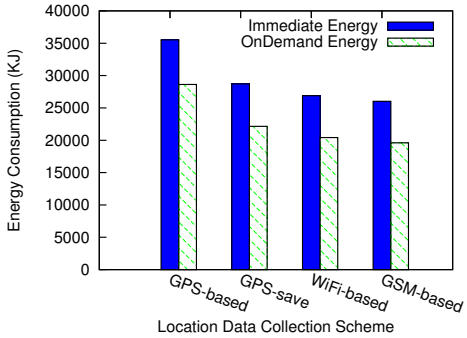


Figure 9: Energy consumption with varying location data collection scheme.

head. The figure shows a tradeoff between access latency and energy-efficiency. As the meta-data upload period increases, the energy consumption decreases while the access latency grows.

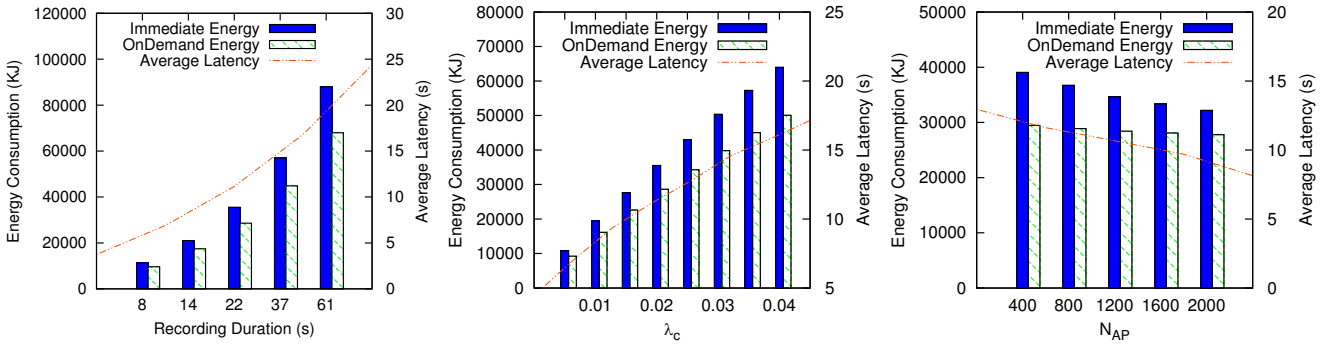
Collecting location data itself costs a significant amount of energy so we consider and compare the four different location data collection schemes as mentioned in Section 3.1. With the GPS-based scheme the GPS receiver is always on during recording. The GPS-save scheme means that when the device is not moving or changing direction, GPS sampling is not executed. The WiFi-based and GSM-based schemes are described in [4]. Figure 9 shows a comparison of the energy consumption using these four approaches. The GPS-based scheme consumes the most energy, while the GPS-save scheme indeed saves a significant amount of energy. Also, the energy consumption of the WiFi-based and GSM-based schemes is much less than that of the GPS-based scheme.

Next we evaluate the impact of the capture and network topology parameters on the performance. First, Figure 10(a) shows the trend for an increasing video recording duration, which is log-normally distributed based on parameter D_c. We calculate the average duration with $e^{D_c} \times ts$ (see Table 3). As expected for both Immediate and OnDemand, a longer average recording duration results in a higher energy consumption and a longer query response latency. However, OnDemand consumes less energy, up to 30% less compared to Immediate. Predictably, the query response latency for OnDemand increases as the recording duration increases. This is because more FOVs are captured in the simulation area. When the same query is executed within a region, mobile nodes will have more video frames to upload, which results in a longer latency. A similar trend can be observed when nodes capture videos more frequently (Figure 10(b)). When the capture event arrival rate λ_c increases, the energy consumption and the latency will increase.

Next we investigate the impact of the number of access points (N_{AP}). APs are uniformly distributed in the simulation area. When more access points are deployed the average available data rate for video transmissions increases, and conversely the average transmission duration decreases. Our simulation results shows that energy consumption is reduced for the Immediate strategy as the number of APs grows (Figure 10(c)). However, with OnDemand the energy usage remains steady while the latency decreases. This indicates that OnDemand is less affected by the number of APs than Immediate, implying that OnDemand utilizes the limited bandwidth more effectively.

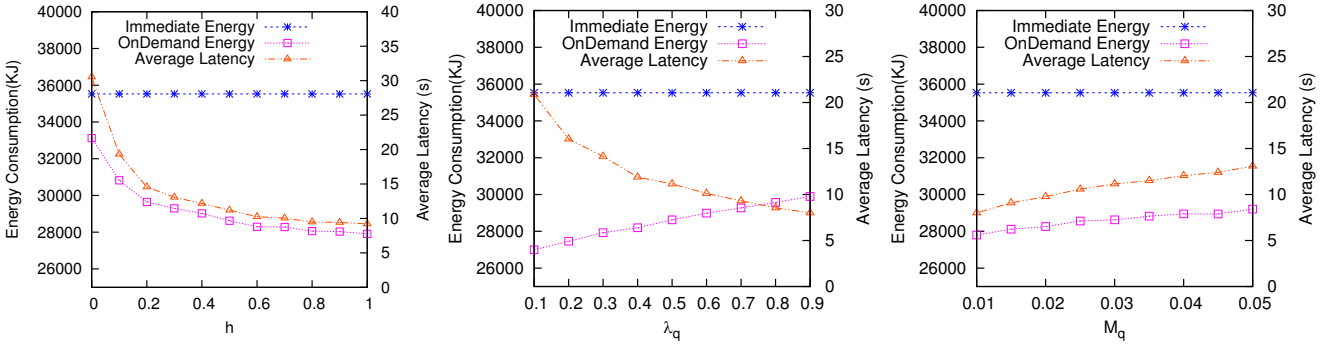
We next turn our attention to the impact of the characteristics of queries on the performance metrics. For different query model parameters (i.e., h, λ_q, and M_q), Figure 11 shows how the energy consumption and latency are affected while Figure 12 plots the total amount of transmitted data. Note that the performance of Immediate does not change with the query model parameters because Immediate's behavior determines the data collection and transmission independently from any query models.

Figure 11(a) illustrates the effects of increased query clustering on the energy consumption and average query re-



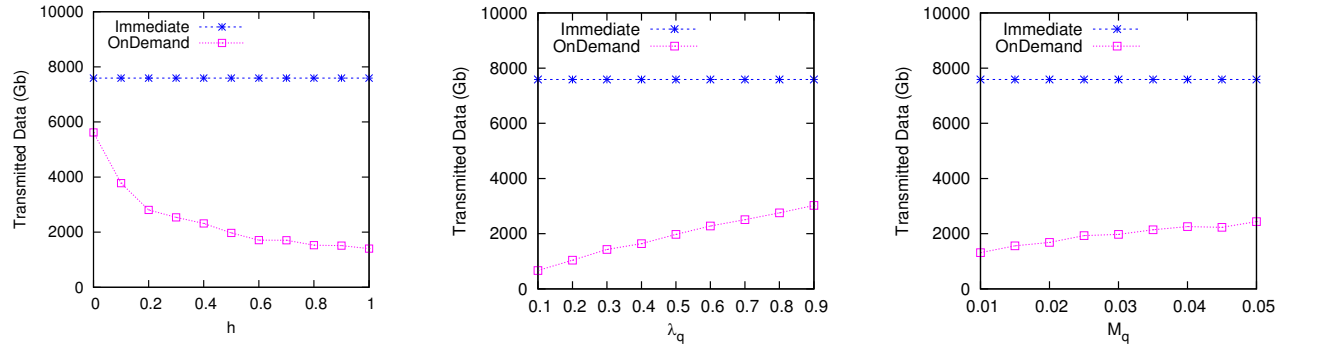
(a) Average recording duration. (b) Capture event arrival rate λ_c. (c) Number of access points N_{AP}.

Figure 10: Energy consumption and average query response latency with varying FOV and Network topology generator parameters.



(a) Query clustering h. (b) Query arrival rate λ_q ($h = 0.5$). (c) Query size M_q ($h = 0.5$).

Figure 11: Energy consumption and average query response latency with varying query model parameters.



(a) Query clustering h. (b) Query arrival rate λ_q ($h = 0.5$). (c) Query size M_q ($h = 0.5$).

Figure 12: Total transmitted data size as a function of various query model parameters.

sponse latency. OnDemand's performance significantly improves as the query distribution changes from uniformly random (i.e., $h = 0$) to most clustered ($h = 1$). With $h = 1$ some regions within the simulation space will never be queried, thus the nodes in those areas do not need to transmit collected videos to the server. Consequently, their energy consumption becomes minimal. For popular areas, when a query arrives most of the videos have already been uploaded to the server. Thus, the query response latency can be reduced. Figure 12(a) clearly indicates that the total amount of video data transmitted is much less with OnDemand than with Immediate. This is especially true when queries are concentrated on some popular hotspots. Our OnDemand strategy clearly demonstrates its strength over Immediate in a highly clustered query distribution which better reflects a realistic situation when user attention focuses on some popular areas.

Figures 11(b) and 12(b) show the trends for an increasing query arrival rate λ_q. As the query frequency increases, the energy consumption and the size of the transmitted video data grow up with OnDemand. However, both metrics still stay well below their corresponding values of Immediate, demonstrating clear benefits. Furthermore, the query response latency also decreases. Intuitively, if a node is in a

frequently queried area, the videos captured by the node will be probably uploaded sooner rather than later. Therefore the latency for each query can be reduced substantially. We also evaluate the impact of a varying mean query size M_q on the performance. As expected, Figures 11(c) and 12(c) illustrate that for OnDemand the energy consumption, the query response latency and the total transmitted data all rise gradually with increased M_q.

4.3.3 *Hybrid Strategy*

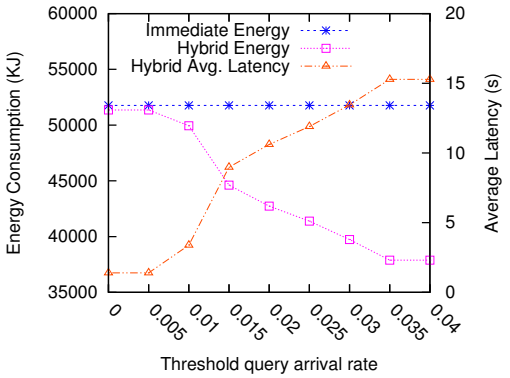


Figure 13: The overall energy consumption and query response latency when using a hybrid strategy with both Immediate and OnDemand as a function of the switching threshold ($h = 0.5$).

As seen in the previous section, the Immediate strategy has an advantage when it comes to query response latency, since videos are always pro-actively uploaded to the server. However, OnDemand has an advantage in power consumption. To get the best combination of response time and battery life one may devise a hybrid strategy that essentially select a method based on the popularity of a region. To achieve this we can divide the map into a grid of small partitions and maintain a popularity threshold for each grid cell. The server continuously computes the query arrival rate λ_q^{est} for each partition per time interval t as $\lambda_q^{est} = \frac{N_q}{t}$. N_q represents the number of queries that are executed during time t. If λ_q^{est} is above the threshold for a partition, the server will mark it as popular and ask the mobile nodes moving in that area to switch to Immediate mode. Figure 13 shows the energy consumption and average query response latency for the hybrid strategy under different threshold values. In the illustrated case, a threshold value in the range of 0.01 to 0.02 may achieve a good compromise in terms of both energy use and query latency.

5. PROTOTYPE

We are currently implementing the proposed system as part of our ongoing project work (hence the reason for presenting simulation results in Section 4). We implemented a prototype geo-referenced video acquisition module on an Android G1 handset, which provides the necessary built-in GPS receiver and compass functionality. Below we describe our current application implementation. Please note that we are just starting to collect real-world data with this platform for further studies.

Parameter		Description
Format		MPEG-4
Format profile		3GPP Media
Overall bit rate		349 Kbps
	Video	Audio
Format	H.263	AMR
Format profile	Baseline@4.0	Narrow band
Bit rate mode	Variable	Constant
Bit rate	334 Kbps	12.8 Kbps
Resolution (pixels)	320 × 240	
Aspect ratio	4:3	
Frame rate	15 fps	
Colorimetry	4:2:0	
Channel(s)		1 channels
Sampling rate		8.0 KHz

Table 4: Android audio/video capture parameters.

5.1 Android Geo-Video Application

Our Geo-Video App was developed with the Google Android SDK v1.5 for Android OS 1.0 or later. The program was written in Java. The Geo-Video App is composed of the following six functional modules: (1) video stream recorder, (2) location receiver, (3) orientation receiver, (4) data storage and synchronization control, (5) data uploader and (6) battery status monitor. Below we will describe each module in more detail.

Video Stream Recorder. This module employs the Android MediaRecorder to invoke the built-in camera. On G1, H.263 is the only supported video encoder, together with an AMR_NB encoder for audio. Table 4 summarizes the audio and video acquisition parameters.

Location and Orientation Receiver. Android provides some system services for getting data from the sensors. Available sensors are an accelerometer, magnetic field sensor, and a built-in orientation sensor. To get the camera orientation, one can use either the orientation sensor or compute it using the accelerometer and magnetic field sensor. But the latter is more precise than the former, at the cost of more computation. We choose the latter in our application. The GPS data is straightforwardly provided by location service.

An interesting aspect in sensor data acquisition is the sampling frequency. In our application we set a fixed sampling rate for the location and orientation information. The sampling rate is set to 5 samples per second. Experimentally, with these settings we can discover the changes in the viewable scenes well while saving battery energy as much as possible.

Data Storage and Synchronization Control. This module manages the storage of the sensor data on the device's flash disk. The goal is to utilize a flexible data format that can be easily ingested at a server. In this situation we choose JSON (JavaScript Object Notation) as the data interchange and storage, since it has the equal descriptive power comparable to XML and an order of magnitude less complexity than XML. The data format consists of four mandatory key attributes:
format_version: Version number of the data format.
video_id: Relevant video ID associated with the sensor data.
owner_properties: User account associated with sensor data.
device_properties: Device dependent information.
sensor_data: Raw sensor data collected from a mobile device.

Here is a sample specification of the data format that stores sensor data.

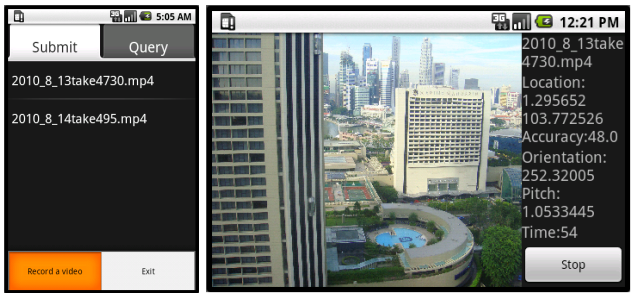
```
{
    "format_version":"0.1",
    "video_id":"a uniquely identifiable video id",
    "owner_properties":{
        "id_type":"google account",
        "id":someone@google.com
    },
    "device_properties":{
        "SIM_id":"an id taken from SIM card",
        "OS":"Android",
        "OS_version":"1.0",
        "firmware_version":"1.0"
    },
    "sensor_data":[
        {
            "location_array_timestamp_lat_long":[
                ["2010-03-18T07:58:41Z",1.29356,103.77],
                ["2010-03-18T07:58:46Z",1.29356,103.78]
            ]
        },
        {
            "sensor_array_timestamp_x_y_z":[
                ["2010-03-18T07:58:41Z",180.00,1.00,1.00],
                ["2010-03-18T07:58:46Z",181.00,1.00,1.00]
            ]
        }
    ]
}
```

To provide synchronization between meta-data and video streams, we extract the duration, encoded date and time from the video. We then add timestamp information to every sensor data record to establish the relationship between a video clip and its corresponding geo-sensor information. Time is represented in Greenwich Mean Time (GMT), to avoid time zone issues. Files include the timestamp as part of their filename to avoid ambiguity..

DATA UPLOADER. This module makes use of open source class *ClientHTTPRequest* written by Vlad Patryshev. This class helps to send POST HTTP requests with various form data to the server, which is not natively supported by Android environment. This third-party class makes some of the more tedious aspects of communicating with web servers easier. The Data Uploader transparently utilizes WiFi, 3G or 2G cellular networks to transmit data files. Importantly, this module implements our two different upload strategies: (1) both video and sensor files are uploaded concurrently and (2) only the sensor files are uploaded first, while the video files may be transmitted later. Video files on the flash disk are tagged whether they still need to be uploaded.

5.2 User Interface

Figure 14 shows two screenshots of our Geo-Video App. When the user launches the software, he or she will first see the main menu (Figure 14(a)). A list of the captured videos is displayed on the screen. The user can choose to continue to upload video clips, whose sensor data was previously uploaded, by choosing name of the needed video file. The main menu consists two tabs: a submit tab for uploading videos, and a query tab for future extension of video query function on mobile device. When the user press **MENU** button, it will show "Record a video" and "Exit". If the user touches the "Record a video" button, a camera viewfinder will be displayed (Figure 14(b)) and the user can then record, stop, cancel or edit a video clip via this interface just like they



(a) Main menu (b) Geo-Recorder

Figure 14: Geo-Video Android application prototype.

usually do in the G1's default camera view. However, our system additionally starts to record geo-referenced information from the GPS and the digital compass and these information is also shown on the phone screen. The sensor data is stored to the device at the time when the video is saved to the camera roll and flash disk. Next, an uploading screen guides the user through the next step. A destination URL is displayed (which can be changed) and either the sensor information only or both the sensor and video files can be uploaded. As mentioned earlier, saved videos can be uploaded at a later point in time directly from the main menu screen.

6. RELATED WORK

There exist only a few systems that associate a large set of sensor values with mobile video. Most of the existing work is limited to images and location coordinates, without considering compass direction. There is no specific work investigating energy issues for mobile video transmissions. Below we provide an overview of some of the existing work.

6.1 Digital Media with Geo-Locations

Associating GPS coordinates with digital photographs has become an active area of research [17]. There has been research on organizing and browsing personal photos according to location and time. Toyama *et al.* [21] introduced a meta-data powered image search and built a database, also known as World Wide Media eXchange (WWMX), which indexes photographs using location coordinates and time. A number of additional techniques in this direction have been proposed [13, 15]. There are also several commercial web sites (e.g., Flickr, Woophy) that allow the upload and navigation of geo-referenced photos. All these techniques use only the camera geo-coordinates as the reference location in describing images. We instead propose a much broader, sensor-based description of video scenes. More related to our work, Ephstein *et al.* [6] proposed to relate images with their view frustum (viewable scene) and used a scene-centric ranking to generate a hierarchical organization of images. Some approaches [20, 11] use location and other meta-data, as well as text tags associated with images, and the images' visual features to generate representative candidates within image clusters. Geo-location is often used as a filtering step. Our work considers a much more comprehensive scenario that is concerned with continuous sensor-streams of mobile videos, which are dynamically changing over time.

6.2 Energy Management on Mobile Devices

Limited battery power has been a fundamental problem in the field of mobile computing. A great deal of work has focused on energy management on mobile devices. Viredaz *et al.* [22] surveyed many energy-saving techniques for handheld devices in terms of improving the design and cooperation of system hardware, software as well as multiple sensing sources. Wang *et al.* [23] proposed a hierarchical approach for managing sensors in order to achieve human state recognition in an energy efficient manner. The SeeMon system [10] is a scalable and energy-efficient context monitoring framework for sensor-rich and resource limited mobile environments. Authors in [16] have drawn attention to the tradeoff between energy and location accuracy. Our work on mobile geo-referenced video management achieves energy efficiency by separating the descriptive sensor information from the bulky video data and by delaying the transmission of the actual video.

6.3 Video Sensor Networks

A few video-based sensor networks have been developed for monitoring and surveillance. Panoptes [7] used a camera device based on a Intel StrongARM PDA platform with a Logitech Webcam as the vision sensor and 802.11b for wireless communication. SensEye [12] is a multi-tier network of heterogeneous wireless nodes and cameras. Low-power cameras, which are capable of taking low-resolution images, form the bottom level. When an object of interest is identified, these sensors trigger cameras at a higher tier on demand to take better images. In contrast, our proposed system uses off-the-shelf mobile devices. This provides mobility and can also simplify the deployment burden.

7. CONCLUSIONS

Capturing video in conjunction with descriptive sensor meta-data allows a new strategy of uploading the sensor information in real-time while transmitting the bulky video data on demand later. This key idea can reduce the transmission of uninteresting videos and hence significantly lower the energy consumption in battery-powered mobile camera nodes. In our study we presented the design and prototype implementation of a mobile video management system that uses smartphones as mobile video sensors. We demonstrated the energy efficiency of our system with simulations and the experimental results showed that our technique can substantially prolong the device usage time, while ensuring a low search latency. We expect this method to be useful for a wide range of novel applications.

Acknowledgments

This research has been funded in part by A*Star grant 082 101 0028. We also acknowledge the support of the NUS Interactive and Digital Media Institute (IDMI).

8. REFERENCES

[1] I. Akyildiz, T. Melodia, and K. Chowdhury. A Survey on Wireless Multimedia Sensor Networks. *Computer Networks*, 51, 2007.

[2] S. Arslan Ay, R. Zimmermann, and S. H. Kim. Viewable Scene Modeling for Geospatial Video Search. In 16^{th} *ACM Intl. Conference on Multimedia*, pages 309–318, 2008.

[3] T. Brinkhoff. A Framework for Generating Network-Based Moving Objects. *GeoInformatica*, 6(2):153–180, 2002.

[4] Y. Cheng, Y. Chawathe, A. LaMarca, and J. Krumm. Accuracy Characterization for Metropolitan-scale Wi-Fi Localization. In 3^{rd} *Intl. Conference on Mobile Systems, Applications, and Services*, page 245, 2005.

[5] T. Cheung, K. Okamoto, F. Maker III, X. Liu, and V. Akella. Markov Decision Process (MDP) Framework for Optimizing Software on Mobile Phones. In 7^{rd} *ACM Intl. Conference on Embedded Software*, pages 11–20, 2009.

[6] B. Epshtein, E. Ofek, Y. Wexler, and P. Zhang. Hierarchical Photo Organization Using Geo-Relevance. In 15^{th} *ACM Intl. Symposium on Advances in Geographic Information Systems (GIS)*, pages 1–7, 2007.

[7] W. Feng, E. Kaiser, W. Feng, and M. Baillif. Panoptes: Scalable Low-power Video Sensor Networking Technologies. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMCCAP)*, 1(2):151–167, 2005.

[8] I. Google. Android – An Open Handset Alliance Project. http://developer.android.com.

[9] C. H. Graham, N. R. Bartlett, J. L. Brown, Y. Hsia, C. C. Mueller, and L. A. Riggs. *Vision and Visual Perception*. John Wiley & Sons, Inc., 1965.

[10] S. Kang, J. Lee, H. Jang, H. Lee, Y. Lee, S. Park, T. Park, and J. Song. SeeMon: Scalable and Energy-Efficient Context Monitoring Framework for Sensor-Rich Mobile Environments. In 6^{th} *Intl. Conference on Mobile Systems, Applications, and Services*, pages 267–280, 2008.

[11] L. S. Kennedy and M. Naaman. Generating Diverse and Representative Image Search Results for Landmarks. In 17^{th} *Intl. Conference on the World Wide Web (WWW)*, pages 297–306, 2008.

[12] P. Kulkarni, D. Ganesan, P. Shenoy, and Q. Lu. SensEye: a Multi-tier Camera Sensor Network. In 13^{th} *ACM Intl. Conference on Multimedia*, page 238, 2005.

[13] M. Naaman, Y. J. Song, A. Paepcke, and H. Garcia-Molina. Automatic Organization for Digital Photographs with Geographic Coordinates. In 4^{th} *ACM/IEEE-CS Joint Conference on Digital Libraries*, pages 53–62, 2004.

[14] T. Pavlidis. Why Meaningful Automatic Tagging of Images is Very Hard. In *IEEE ICME 2009*, pages 1432–1435, 2009.

[15] A. Pigeau and M. Gelgon. Building and Tracking Hierarchical Geographical & Temporal Partitions for Image Collection Management on Mobile Devices. In 13^{th} *ACM Intl. Conference on Multimedia*, 2005.

[16] M. Ra, J. Paek, A. Sharma, R. Govindan, M. Krieger, and M. Neely. Energy-delay Tradeoffs in Smartphone Applications. In 8^{th} *Intl. Conference on Mobile Systems, Applications, and Services*, pages 255–270, 2010.

[17] K. Rodden and K. R. Wood. How do People Manage their Digital Photographs? In *Conference on Human Factors in Computing Systems (SIGCHI)*, pages 409–416, 2003.

[18] A. Shye, B. Sholbrock, and G. Memik. Into The Wild: Studying Real User Activity Patterns to Guide Power Optimization for Mobile Architectures. In *Micro*, 2009.

[19] B. Tiwana and L. Zhang. PowerTutor. http://powertutor.org, 2009.

[20] C. Torniai, S. Battle, and S. Cayzer. Sharing, Discovering and Browsing Geotagged Pictures on the Web. In A. Scharl and P. K. Tochtermann, editors, *The Geospatial Web: How Geo-Browsers, Social Software and the Web 2.0 are Shaping the Network Society*. Springer, 2006.

[21] K. Toyama, R. Logan, and A. Roseway. Geographic Location Tags on Digital Images. In 11^{th} *ACM Intl. Conference on Multimedia*, pages 156–166, 2003.

[22] M. Viredaz, L. Brakmo, and W. Hamburgen. Energy Management on Handheld Devices. *Queue*, 1(7):52, 2003.

[23] Y. Wang, J. Lin, M. Annavaram, Q. Jacobson, J. Hong, B. Krishnamachari, and N. Sadeh. A Framework of Energy Efficient Mobile Sensing for Automatic User State Recognition. In 7^{th} *Intl. Conference on Mobile Systems, Applications, and Services*, pages 179–192, 2009.