MediaQ: Mobile Multimedia Management System

Seon Ho Kim† Ying Lu† Giorgos Constantinou† Cyrus Shahabi‡ Guanfeng Wang‡ Roger Zimmermann‡
†Integrated Media Systems Center, University of Southern California, Los Angeles, CA 90089
‡School of Computing, National University of Singapore, Singapore 117417
†{seonkim, ylu720, gconstan, shahabi}@usc.edu
‡{wanggf, rogerz}@comp.nus.edu.sg

ABSTRACT

MediaQ is a novel online media management system to collect, organize, share, and search mobile multimedia contents using automatically tagged geospatial metadata. User-generated videos can be uploaded to the MediaQ from users’ smartphones, iPhone and Android, and displayed accurately on a map interface according to their automatically sensed geospatial and other metadata. The MediaQ system provides the following distinct features. First, individual frames of videos (or any meaningful video segments) are automatically annotated by objective metadata which capture four dimensions in the real world: the capture time (when), the camera location and viewing direction (where), several keywords (what) and people (who). We term this data W4-metadata and they are obtained by utilizing camera sensors, geospatial and computer vision techniques. Second, a new approach of collecting multimedia data from the public has been implemented using spatial crowdsourcing, which allows media content to be collected in a coordinated manner for a specific purpose. Lastly, flexible video search features are implemented using W4 metadata, such as directional queries for selecting multimedia with a specific viewing direction.

This paper is to present the design of a comprehensive mobile multimedia management system, MediaQ, and to share our experience in its implementation. Our extensive real world experimental case studies demonstrate that MediaQ can be an effective and comprehensive solution for various mobile multimedia applications.

Categories and Subject Descriptors
I.4.8 [Image Processing and Computer Vision]: Scene Analysis – Sensor Fusion

Keywords
Geo-tagged videos, crowdsourcing, keyframe selection, geospatial metadata, mobile multimedia

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from Permissions@acm.org.

MMSys ’14, March 19 – 21 2014, Singapore, Singapore
ACM 978-1-4503-2705-3/14/03...$15.00.
http://dx.doi.org/10.1145/2557642.2578223.

1. INTRODUCTION

Due to technological advances, an increasing number of video clips are being collected with various devices and stored for a variety of purposes such as surveillance, monitoring, reporting, or entertainment. These acquired video clips contain a tremendous amount of visual and contextual information that makes them unlike any other media type. However, even today, it is very challenging to index and search video data at the high semantic level preferred by humans. Text annotations of videos can be utilized for search, but high-level concepts must often be added by hand and such manual tasks are laborious and cumbersome for large video collections. Content-based video retrieval – while slowly improving in its capabilities – is challenging, computationally complex and unfortunately still often not satisfactory.

Some types of video data are naturally tied to geographical locations. For example, video data from traffic monitoring may not have any meaning without its associated position information. Thus, in such applications, one needs a specific location to retrieve the traffic video at that point or in that region. Hence, combining video data with its location coordinates can provide an effective way to index and search videos, especially when a repository handles an extensive amount of video data. Since most videos are not panoramic the viewing direction also becomes very important.

In this study, we are specifically focusing on mobile videos generated by the public. By 2018, more than 60% of the worldwide Internet traffic is expected to result from video data transmissions from and to mobile devices [12]. Mobile devices such as smartphones and tablets can capture high-resolution videos and pictures. However, they can only store a limited amount of data on the device. Furthermore, the device storage may not be reliable (e.g., a phone is lost or broken). Hence, a reliable backend storage is desirable (e.g., Dropbox, Google Drive, iCloud). Unfortunately, it is very difficult to later search these large storage systems to find required videos and pictures as they are usually file-based and without a facility to systematically organize media content with appropriate indices. This becomes especially troublesome when a huge amount of media data and a large number of users are considered. Moreover, current online mobile video applications mainly focus on simple services, such as storage or sharing of media, rather than integrated services towards more value-added applications.

We address these issues with the proposed MediaQ system by attaching geospatial metadata to recorded mobile videos so that they can be organized and searched effectively. We
believe that geo-tagged video search will play a prominent role in many future applications. However, there still exist many open, fundamental research questions in this field. Most captured videos are not panoramic and as a result the viewing direction is of great importance. Global positioning system (GPS) data only identify object locations and therefore it is imperative to investigate the natural concepts of viewing direction and viewpoint. For example, we may be interested to view a building only from a specific angle. The question arises whether a video repository search can accommodate such human friendly queries. The collection and fusion of multiple sensor streams such as the camera location, field-of-view, direction, etc., can provide a comprehensive model of the viewable scene. The objective then is to index the video data based on the human viewable space and therefore to enable the retrieval of more meaningful and recognizable scene results for user queries. Cameras may also be mobile and thus the concept of a camera location is extended to a trajectory. Consequently, finding relevant video segments becomes very challenging.

One example query that a user may pose to an existing video hosting site could be as follows. Let us consider YouTube as an example to answer the following question. “Find images (or video frames) of myself captured in front of the implemented framework. Specifically, the contribution in my day.” A search like this will retrieve a top video called Tommy Trojan (a statue of the University of Southern California video hosting site could be as follows. Let us consider video segments becomes very challenging.

Most up to date data management technologies can handle text data very efficiently (as exemplified in Google search) but provide limited support for videos and images (as can be seen from the YouTube search facilities). Unlike text documents, understanding visual content correctly has turned out to be a very challenging task. In the past, two main approaches have been utilized to annotate videos and images for indexing and searching. First, manual text annotations by users have been the most practical and preferred way to identify textual keywords to represent visual content. However, this approach suffers from the following drawbacks: 1) the human perception of visual content is subjective, 2) manual annotations are both error-prone and time consuming. Second, content-based retrieval techniques have been applied to automate the annotation process. However, such methods also suffer from their own limitations such as: 1) inaccurate recognition of visual content, 2) high computational complexity that makes them unsuitable for very large video applications, and 3) domain specificity such that they cannot handle open-domain user videos.

In an effort towards addressing the above challenges we introduce the MediaQ as a novel mobile multimedia management system. Figure 1 illustrates the overall structure of the implemented framework. Specifically, the contributions of the presented system are as follows.

- MediaQ is the first system to use an underlying model of sensor metadata fused with mobile video content. Individual frames of videos (or any partial video segment) are automatically, without manual intervention, annotated by objective metadata that capture time (when), location (where), and keywords (what).

We also extensively review the related work on spatial crowdsourcing.

2.1 Sensor Rich Video Systems

There exist a few systems that associate videos with their corresponding geo-locations. Hwang et al. [15] and Kim et al. [19] proposed a mapping between the 3D world and videos by linking objects to the video frames in which they appear. However, their work neglected to provide any details on how to use the camera location and direction to build links between video frames and world objects. Liu et al. [22] presented a sensor enhanced video annotation system (referred to as SEVA) which enables to search videos for the appearance of particular objects. SEVA serves as a good example to show how a sensor rich, controlled environment can support interesting applications. However, it did not propose a broadly applicable approach to geo-spatially annotate videos for effective video search.
In our prior and ongoing work [9, 8], we have extensively investigated these issues and proposed the use of videos’ geographical properties (such as camera location and direction) to enable an effective search of specific videos in large video collections. This has resulted in the development of the GeoVid framework based on the concept of georeferenced video. GeoVid introduced a viewable scene model that is utilized to describe video content. With mobile apps and a web portal (located at http://geovid.org), we demonstrated how this model enhances video management, search and browsing performance.

However, the systems mentioned above were limited in that they presented ideas on how to search and manage the video collections based on the where information (i.e., geographical, e.g., locations, directions) of videos. In addition to the where information, MediaQ also considers the when, what and who information of video contents. Furthermore, the existing systems do not provide crowdsourcing services while MediaQ exploits the idea of spatial crowdsourcing termed GeoCrowd [18] to collect on-demand media content on behalf of users. Moreover, MediaQ can provide social network services for sharing and following media content in which users are interested.

2.2 Spatial Crowdsourcing (GeoCrowd)

While crowdsourcing has recently attracted interest from both research communities (e.g., database [14], image processing [11], NLP [24]) and industry (e.g., Amazon’s Mechanical Turk [1] and CrowdFlower [2]), only a few earlier approaches [7, 10, 17, 18] have studied spatial crowdsourcing which closely ties locations to crowdsourcing.

A well developed concept of spatial crowdsourcing was first introduced by [18], in which workers send their locations to a centralized server and thereafter the server assigns nearby tasks to every worker with the objective of maximizing the overall number of assigned tasks. In [10], the problem of location-based crowdsourcing queries over Twitter was studied. This method employs a location-based service (e.g., Foursquare) to find appropriate people to answer a given query. This work does not require that users should go to the specific locations and perform the corresponding tasks. Instead, it selects users based on their historical Foursquare check-ins. Participatory sensing is related to spatial crowdsourcing, in which workers form a campaign to perform sensing tasks. Examples of participatory sensing campaigns include [6, 17], which used GPS-enabled mobile phones to collect traffic information.

Volunteered geographic information (or VGI) is also related to spatial crowdsourcing. VGI (e.g., WikiMapia [5], Open-StreetMap [4], and Google Map Maker [3]) aims to create geographic information voluntarily provided by individuals. However, the major difference between VGI and spatial crowdsourcing is that in VGI, users unsolicitedly participate by randomly contributing data, whereas in spatial crowdsourcing, a set of spatial tasks are explicitly requested by the requesters, and workers are required to perform those tasks.

3. MEDIAQ FRAMEWORK

3.1 Overview

The schematic design of the MediaQ system is summarized in Figure 1. Client-side components are for user interaction, i.e., the Mobile App and the Web App. The Mobile App is mainly for video capturing with sensed metadata and their uploading. The Web App allows searching the videos and issuing spatial crowdsourcing task requests to collect specific videos. Server-side components consist of Web Services, Video Processing, GeoCrowd Engine, Query Processing, Account Management, and Data Store. The Web Service is the interface between client-side and server-side components. The Video Processing component performs transcoding of uploaded videos so that they can be served in various players. At the same time, uploaded videos are analyzed by the visual analytics module to extract extra information about their content such as the number of people in a scene. We can plug in open source visual analytics algorithms here to achieve more advanced analyses such as face recognition among a small group of people such as a user’s family or friends. Automatic keyword tagging is also performed at this stage in parallel to reduce the latency delay at the server. Metadata (captured sensor data, extracted keywords, and results from visual analytics) are stored separately from uploaded media content within the Data Store. Query Processing supports effective searching for video content using the metadata in the database. Finally, task management for spatial crowdsourcing can be performed via the GeoCrowd engine.

3.2 Media Collection with Metadata

3.2.1 Field Of View Modeling

In our approach, we represent the media content (i.e., images and videos) based on the geospatial properties of the region it covers, so that large video collections can be indexed and searched effectively using spatial database technologies. We refer to this area as the Field Of View (or FOV) of the video scene [9].

Figure 2: 2D Field-of-View (FOV) model.

As shown in Figure 2, the scene of video frame \( f \) is represented in a 2D FOV model with four parameters, \( f \equiv (p, \theta, R, \alpha) \), where \( p \) is the camera position consisting of the latitude and longitude coordinates (an accuracy level can be also added) read from the GPS sensor in a mobile device, \( \theta \) represents the viewing direction \( \vec{d} \), the angle with respect to the North obtained from the digital compass sensor, \( R \) is the maximum visible distance at which an object can be recognized, and \( \alpha \) denotes the visible angle obtained from the camera lens property at the current zoom level. For simplicity, this study assumes that the camera is always level so the vector \( \vec{d} \) points towards the camera heading on the horizontal plane only. Note that extending a 2D FOV to a 3D FOV is straightforward. Let \( \mathcal{F} \) be the video frame set \( \{ f \forall f \in v, \forall v \in V \} \). All the video frames of all the videos in \( V \) are treated as a large video frame set \( \mathcal{F} \).
Within our mobile app (detailed in Section 3.7), we implement a custom geospatial video module to acquire, process, and record the location and direction metadata along with captured video streams. The app can record H.264 encoded videos at DVD-quality resolution. To obtain the camera orientation, the app employs the digital compass and accelerometer sensors in the mobile device. Camera location coordinates are acquired from the embedded GPS receiver sensor. The collected metadata is formatted with the JSON data-storage and -interchange format. Each metadata item in the JSON data corresponds to the viewable scene information of a particular video frame \( f_i \). For the synchronization of the metadata with video content, each metadata item is assigned an accurate timestamp and video time-code offset referring to a particular frame in the video. The frame rate of the collected videos is 24 frames per second. Note that referring to a particular frame in the video, the frame rate assigned an accurate timestamp and video time-code offset of the metadata with video content, each metadata item is captured video streams. The app can record H.264 encoded and record the location and direction metadata along with every frame. Figure 3 shows the screenshots of the acquisition app. The recorded geo-tagged videos are uploaded to the server, where postprocessing and indexing is performed afterwards.

![Screenshots of the acquisition app](image)

**Figure 3:** Screenshots of the media collection with metadata module in our mobile app for Android-based (top) and iOS-based (bottom) smartphones.

### 3.2.2 Positioning Data Accuracy Enhancement

As described previously, \( p \) is the latitude/longitude coordinate that indicates the camera location which is obtained from an embedded GPS receiver. The accuracy of the location data is critical in our approach. However, in reality, the captured locations may not be highly exact due to two reasons: 1) the varying surrounding environmental conditions (e.g., reflections of signals between tall buildings) during data acquisition, and 2) inherent sensor errors (e.g., the use of low-cost sensors in mobile devices). In our system, we enhance the accuracy of the positioning data with a post-processing step immediately after the server receives metadata. We have devised a data correction algorithm based on Kalman filtering and weighted linear least square regression [25] as follows.

An original GPS reading \( p_k \) is always accompanied with an accuracy measurement value \( a_k \). The accuracy measure indicates the degree of closeness between a GPS measurement \( p_k \) and its true, but unknown position, say \( g_k \). If \( a_k \) is high then that means that the actual position \( g_k \) is far away from \( p_k \). We utilize a model of location measurement noise with \( p_k \) and \( a_k \) [21], where the probability of the real position data is assumed to be normal distributed with a mean of \( p_k \) and its standard deviation \( \sigma_k \). We then set \( \sigma_k^2 = g(a_k) \), where the function \( g \) is monotonically increasing.

**Kalman Filtering-based Correction.**

We model the correction process in accordance with the framework of Kalman filters. Two streams of noisy data are recursively operated on to produce an optimal estimate of the underlying positions. We describe the position and velocity of the GPS receiver by the linear state space:

\[
\pi_k = [x_k \ y_k \ v_{kx} \ v_{ky}]^T,
\]

where \( v_{kx} \) and \( v_{ky} \) are the longitude and latitude component of velocity \( v_k \). In practice, \( v_k \) can be estimated by some less uncertain coordinates and their timestamp information. We define the state transition model \( F_k \) as

\[
F_k = \begin{bmatrix}
1 & 0 & \Delta t_k & 0 \\
0 & 1 & 0 & \Delta t_k \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}.
\]

where \( \Delta t_k \) is the time duration between \( t_k \) and \( t_{k-1} \). We also express the observation model \( H_k \) as

\[
H_k = \begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0
\end{bmatrix}.
\]

\( H_k \) maps the true state space into the measured space. For the measurement noise model, we use \( a_k \) to present the covariance matrix \( R_k \) of observation noise as follows:

\[
R_k = \begin{bmatrix}
g(a_k) & 0 \\
0 & g(a_k)
\end{bmatrix}.
\]

Similarly, \( Q_k \) can also be determined by a diagonal matrix but using the average of \( g(a_k) \), whose corresponding position coordinates \( p_k \) and timestamp \( t_k \) were used to estimate \( v_k \) in this segment. We apply this process model to the recursive estimator in two alternating phases. The first phase is the prediction, which advances the state until the next scheduled measurement is coming. Second, we incorporate the measurement value to update the state.

**Weighted Linear Least Squares Regression-based Correction.**

The second correction model is based on a piecewise linear regression analysis. Since we post-process the GPS sequence data, we can fully utilize both previous and future GPS readings, from \( p_i \) to \( p_j \), to estimate and update the current position \( p_k \), where \( i < k < j \). With the assumption that the error of different GPS readings are uncorrelated with each other and with the independent variable \( p_k \), we
utilize the weighted least squares method to generate estimators $\hat{\beta}$ for each GPS trajectory segment. We denote the longitude and latitude of one GPS value as $x_k$ and $y_k$, respectively. With the assumption that $x_k$ and $y_k$ are two independent variables, we estimate model function parameters $\beta_k$ for longitude and latitude values with respect to time separately. The goal of our method is to find $\hat{\beta}$ for the model which “best” fits the weighted data. By using the weighted least squares method, we need to minimize $R_k$, where

$$R = \sum_{k=1}^{j} W_{kk} r_k^2, \quad r_k = x_k - f(t_k, \hat{\beta})$$ (1)

Here $r_k$ is the residual defined as the difference between the original measured longitude value and the value predicted by the model. The weight $W_{kk}$ is defined as:

$$W_{kk} = \frac{1}{\sigma_k^2}$$ (2)

Here $\sigma_k$ is the deviation of the measurement noise. It is proven that $\hat{\beta}$ is a best linear unbiased estimator if each weight is equal to the reciprocal of the variance of the measurement. As described before, we model the measurement noise as a normal distribution with mean $x_k$ and standard deviation $\sigma_k = g(a_k)$ in the longitude dimension. Base on this model, measurements $x_k$ with a high $a_k$ value, which indicates high uncertainty, will not have much impact on the regression estimation. Usually, these uncertain measurements reflect many scattered GPS locations, which are far away from where the real positions should be. Considering that the regression line is estimated mostly by the confidence data and these data are mostly temporally sequential, we are able to correct those spotty GPS locations to positions that are much closer to the real coordinates.

To quantitatively evaluate the two proposed algorithms, we compute the average distance between every processed sample and its corresponding ground truth for each GPS sequence data file, and compare these values to the average distance between every measurement sample and the ground truth position. In our experiments, we evaluated 10,069 GPS samples from 63 randomly selected videos. On average, the Kalman filtering based algorithm and the weighted linear least squares regression based algorithm improve the GPS data correctness by 16.3% and 21.76%, respectively. Figure 4 illustrates a Cumulative Distribution Function (CDF) for both algorithms. The results show an increased proportion of GPS data with low average error distance and a shortening of the largest sequence average error distance by around 30 meters (the line of processed data meets $y = 1$ at $x = 50$ m, while the line of the original measurements achieves a value of one at $x = 80$ m).

3.2.3 Automatic Keyword Tagging

Geocoordinates and directional angles from sensors provide the essential metadata to organize, index, and search FOVs by computer. However, humans are not familiar with such numeric data in browsing videos even when a good map-based user interface is provided. Still, the most efficient and user friendly way for video search is based on textual keywords such as the name of a landmark or a street name, rather than by latitude and longitude. Thus, in MediaQ, every video is automatically tagged with a variety of keywords during the postprocessing phase when arriving at the server.

Automatic video tagging is based on captured sensor metadata (e.g., FOVs) of videos introduced in the work of Shen et al. [23]. Figure 5 illustrates the process flow in the tagging module. The tagging has two major processing stages. First, the object information for the covered geographical region is retrieved from various geo-information services (we use OpenStreetMap¹ and GeoDec²) and visible objects are identified according to 2D visibility computations. Occlusion detection is performed to remove hidden objects. Afterwards, the system generates descriptive textual tags based on the object information retrieved from the geo-information services, such as name, type, location, address, etc. [8]. In MediaQ, we currently use the names of visible objects to serve as tags. We generate tags obtained from a limited number of sources, but one of the benefits of our approach is that tag generation can be extended in many ways, for example

¹http://www.openstreetmap.org
²http://infolab.usc.edu/projects/geodec/
by employing geo-ontologies, event databases and Wikipedia parsing techniques. In the second stage, the following relevance criteria are introduced to score the relevance of each tag to the scene (e.g., relevance ranking):

- **Closeness to the FOV Scene Center**: Research indicates that people tend to focus on the center of an image [16]. Based on this observation, we favor objects whose horizontal visible angle range is closer to the camera direction, which is the center of the scene.

- **Distance**: Intuitively, a closer object is likely to be more prominent in the video. Thus, we score objects with a higher value if they have a shorter distance to the camera.

- **Horizontally and Vertically Visible Angle Ranges**: An object that occupies a wider range of the scene (either along the width or height) is more prominent from our experiences.

- **Horizontally and Vertically Visible Percentages**: These two criteria focus on the completeness of the object’s appearance in the video. The video scenes that show a larger percentage of an object are preferable over scenes that show only a small fraction of it.

After obtaining the scores for each criterion, we linearly combine them to compute the overall score of an object in an individual FOV scene. Additionally, we promote the scores of well-known objects (or landmarks), which are more likely to be searched, for the object information retrieved from the geo-information services that include several clues to identify important landmarks. For example, in OpenStreetMap data, some landmarks (e.g., the Singapore Flyer) are given an “attraction” label. Others are augmented with links to Wikipedia pages, which might be an indirect hint about an object’s importance, since something described in Wikipedia is believed to be significant. In the future, we also plan to further adjust the scores according to visual information.

After scoring tag relevance, the video segments for which each tag is relevant are determined. Unlike many other video tagging techniques, MediaQ’s module associates tags precisely with the video segments in which they appear, rather than the whole video clip. Therefore, when a user searches videos for a certain tag, only those relevant video segments are returned. The ranked tags are stored and indexed to allow further search through textual keywords.

### 3.3 GeoCrowd Spatial Crowdsourcing

The advanced connectivity of smartphone devices allows users to have ubiquitous access to networks (Wi-Fi and broadband cellular data connections). As a result, spatial crowdsourcing using smartphones is emerging as a new paradigm for data collection in an on-demand manner. Spatial crowdsourcing can be used in MediaQ to collect data efficiently and at scale in the cases where media contents are not available to users, either due to users’ lack of interests in specific videos or due to other spatial and temporal limitations.

Our implementation of spatial crowdsourcing, GeoCrowd, is built on top of MediaQ and provides the mechanisms to support spatial tasks that are assigned and executed by human workers. The current version is an implementation of the method described by Kazemi and Shahabi [18] where the server is responsible for assigning workers to tasks, the algorithm of which is introduced below.

![Figure 5: The process flow of automatic keyword tagging for sensor-rich videos.](image)

#### 3.3.1 GeoCrowd Algorithm

Requesters (users who are in need of labor to collect media content) can create spatial tasks and send them to the server. Each spatial task is defined by a requester id $i$, its geo-location $l$, the start time $s$, the end time $e$, the number of videos to be crowdsourced $k$ and the query $q$. A task is represented by the tuple $(i, l, s, e, k, q)$. Workers (users who are willing to collect media content for requesters) can send task inquiries (i.e., spatial regions of workers’ interests) to the server. Each task inquiry is defined by a worker id $i$ and two constraints: the spatial region where the worker can perform tasks $R$ (rectangular region defined by $SW$, $NE$ coordinates) and the maximum number of tasks $maxT$ that the worker can execute. Task inquiries are represented by the tuple $(i, R(SW, NE), maxT)$. The goal of the GeoCrowd algorithm is to assign as many tasks as possible to workers while respecting their constraints. For an example, an instance problem of the Maximum Task Assignment (MTA) is depicted in Figure 6.

![Figure 6: Instance problem of maximum task assignment (MTA).](image)
Figure 7: Reduction of MTA to the maximum flow problem.

Figure 6 shows three workers \( (w_1 \text{ to } w_3) \) along with their constraints, \((\text{max}T_1-\text{max}T_3)\) and \((R_i-R_3)\) and the tasks \((t_1-\text{t}_{10})\). In this scenario, it is clear that the tasks \(t_1\) and \(t_3\) are not possible to be assigned to any of the workers since they are outside of every spatial region \(R\). In addition, worker \(w_1\) can only accept tasks \(t_2, t_5\) and \(t_7\) but can perform only two of them because of the \(\text{max}T_1\) constraint.

In [18], it was proved that the MTA problem can be efficiently solved in polynomial time by reducing it to the maximum flow problem. Figure 7 shows how the above mentioned instance problem can be reduced to the maximum flow problem. Each worker and task are represented as vertices in a graph \((v_1-v_3 \text{ for } w_1-w_3 \text{ and } v_4-v_{13} \text{ for } t_1-t_{10})\). There is an edge between a worker and a task iff the task location is within the spatial region \(R\) of the worker. The edge capacity between workers and tasks is limited to 1, since we desire that each worker can perform a specific task once. Two new vertices are added, i.e., source (src) and destination (dest). There is an edge between the source node and each worker node with a weight equal to the \(\text{max}T\) of the worker’s constraint, thus restricting the flow and extend the number of assignments. Similarly, there is an edge between each task node to the destination node with a weight equal to \(K\) (the number of times that the task is going to be crowdsourced). In Figure 7 all weights are equal to 1 assuming that each task will be performed once. In the current algorithm implementation, the system is not restricted to \(K\) being equal to 1. After the graph construction any algorithm that solves the maximum flow problem can be used. In MediaQ, the well-established Ford-Fulkerson algorithm is implemented.

### 3.3.2 GeoCrowd Architecture

Using the above algorithm, we have integrated GeoCrowd into MediaQ. An overview of the GeoCrowd module architecture is illustrated in Figure 8. In order to support all necessary operations (e.g., task publishing, assignment processing, etc.), the GeoCrowd system consists of two main components: 1) a smartphone app based on a mobile architecture, and 2) a server that manages all back- and front-end functionalities (i.e., web services, user management, task assignment, interfaces). The two main components are detailed below.

### 3.3.3 GeoCrowd Web and DB Servers

GeoCrowd’s back-end is deployed on the same server and shares the PHP CI framework with MediaQ. The server side mainly consists of:

- **User Interfaces**: Provides all interfaces to capture the user inputs that are needed to perform the task assignment process. Moreover, using the latest version of Google Maps JavaScript API V3 that provides a multi-functional map-based interface, MediaQ allows requesters to publish tasks. Specifically, requesters can setup the task details, which include Title, Description, Location, Expiry Date, Max \(K\) and media type to be captured. In addition, MediaQ supports interfaces to monitor tasks, view their status per requester and accept or decline a worker’s response.

- **Web Services**: Provides the connection between web and mobile interfaces to the database. The web services are built on top of a PHP CI framework which follows the Model-View-Controller (MVC) development pattern. GeoCrowd data (tasks, task inquiries, etc.) are posted to the appropriate controller and then it is decided which query to perform from the appropriate model. Data are safely stored in a MySQL database with spatial extensions for further processing. Spatial indices are created to support spatial queries performed by the Query Interface and to speed up the retrieval time.

- **Task Assignment Engine**: In the current implementation, a controller is used as a crontab (a time-based job scheduler) to solve the MTA problem periodically. UNIX / Linux system crontab entries are used to schedule the execution of the MTA solver.

- **Notification API**: The notification API uses Google Cloud Messaging (GCM) for Android to notify newly task-assigned workers in real-time.

### 3.3.4 GeoCrowd Mobile Application

A MediaQ mobile application was implemented to support GeoCrowd operations. The GeoCrowd app runs on Android OS as a first step with a future view towards cross-platform compatibility. The app includes a map-based interface to enter and post workers’ task inquiries and interfaces to check assigned tasks. The app capabilities are exploited to capture videos and metadata are extended to include worker ids, task ids and other related GeoCrowd information. Moreover, a push notification service (GCM mobile service) is running in the background to notify a worker in real-time when tasks are assigned to him/her by the GeoCrowd server.

### 3.4 Query Processing

MediaQ can support region, range, directional, keyword queries and temporal queries. All the following queries are based on the metadata described in Section 3.2.
3.4.1 Region Queries

The query region in our implementation implicitly uses the entire visible area on a map interface (i.e., Google Maps) as the rectangular region. The search engine retrieves all FOVs that overlap with the given visible rectangular region. Our implementation of this kind of query aims to quickly show all the videos on the map without constraints.

3.4.2 Range Queries

Range queries are defined by a given circle, within which all the FOVs are found that overlap with the area of the circle. The resulting FOV \( f(p, \theta, R, \alpha) \) of the range circle query \( q, r \) with the center point \( q \) and radius \( r \) fall into the following two cases:

- Case 1: As shown in Figure 9(a), the camera location is within the query circle, i.e., the distance between the camera location \( p \) and the query location \( q \) is less than the query radius \( r \) of the query circle.
- Case 2: As shown in Figure 9(b), although the camera location is outside of the query circle, the area of the FOV partially overlaps with the query circle. Specifically, line segment \( pp' \) intersects with arc \( \hat{ab} \), which is formulated in Eqn. 3, where \( \beta \) represents the angle between vector \( \vec{pq} \) and \( \vec{pp'} \), and \( p' \) denotes any point on the arc \( \hat{ab} \) of the FOV.

\[
R \geq \text{Dist}(p, q) \times \cos \beta - \sqrt{r^2 - (\text{Dist}(p, q) \times \sin \beta)^2} \quad (3)
\]

![Figure 9: Two cases of FOVs results for range queries.](image)

3.4.3 Directional Queries

A directional query searches all video segments whose FOV direction angles are equal to or less than the range of an allowable error margin to a user-specified input direction angle. The videos to be searched are also restricted to their FOVs residing in the given range on the map interface. A user can initiate a directional query request through MediaQ GUI by defining the input direction angle which is an offset from the North. Then the directional query is automatically submitted to the server and the final query results, similar to those of other spatio-temporal queries, are rendered accordingly.

3.4.4 Keyword Queries

Section 3.2.3 describes that textual keywords can be automatically be attached to incoming video frames in MediaQ. The tagged keywords (i.e., “what” metadata) is related to the content of the videos. The textual keyword search provides an alternative and user-friendly way to search videos. In the MediaQ system, given a set of query keywords \( S \), keyword queries are defined as finding all the video frames such that the associated keywords of each video frame contain all of the keywords in the query keyword set \( S \). Keyword queries can be combined with region queries, range queries, and directional queries to provide richer query functions.

3.4.5 Temporal Queries

Temporal queries are defined as “given a time interval, find all the video frames within the duration.” Note that the region queries, range queries, directional queries, and keyword queries described above can be combined with temporal queries, and they have been implemented in MediaQ.

3.4.6 Presenting Query Results

The queries discussed so far can return resulting FOVs, i.e., discrete video frames, which is sufficient when searching images, but not for videos. Videos should be smoothly displayed for human perception. Hence, MediaQ presents the results of a video query as a continuous video segment (or segments) by grouping consecutive FOVs in the same video into a video segment. However, since we are targeting mobile videos, there exist some cases where the result consists of several segments within the same video. When the time gap between two adjacent segments of the same video is large, individual segment will be displayed independently. However, when the time gap is small it would be desirable to display the two adjacent segments as a single segment including the set of FOVs during the time gap (even though these FOVs are not really part of the result of the given query) for a better end-user viewing experience.

To achieve this, we group all the identified FOVs by their corresponding videos and rank them based on their timestamp values within each group. If two consecutively retrieved FOVs within the same group (e.g., in the same video) differ by more than a given time threshold (say, 5 seconds), we divide the group into two separate video segments. For example, in Figure 10, if for the range query \( q_1 \) all the frames \( f_1, \ldots, f_{10} \) are part of the query result, then the entire video \( V \) is returned and displayed as a single video. However, for query \( q_2 \), two groups of video frames \( \{f_1, f_2, f_3\} \) and \( \{f_9, f_{10}\} \) represent the exact results. Then, there are two different ways to present the results: 1) when the time gap between \( t_3 \) and \( t_9 \) is more than the threshold time, 5 seconds, since \( f_1, f_2, f_3 \) are continuous and part of the same video \( V \), we combine them together to generate a video segment result from \( t_1 \) to \( t_3 \). In the same way, query FOV results \( f_9 \) and \( f_{10} \) are continuous so another video segment is generated from \( t_9 \) to \( t_{10} \). Hence for query \( q_2 \), we return two video segment results, and 2) when the time gap between \( t_3 \) and \( t_9 \) is less than the threshold time, we combine all frames to connect the two groups and present the result as one video, i.e., \( V \).

3.5 Panoramic Image Generation

Since MediaQ can provide the continuous fusion of geospatial metadata and video frames, such correlated information can be used for the generation of new visual information, not only for plain display of video results. This section describes an example of such an application, the automatic generation of panoramic images, to demonstrate the poten-
produced algorithms based on the geo-information are follows: from videos to achieve a high efficiency without sacrificing to effectively and automatically generate panoramic images selection methods have been proposed in our prior work \[20\] locations and directions). Several novel key video frame se-
culation methods have been proposed in our prior work \[20\]

Figure 10: Query result representation through video segments. Circles \(q_1\) and \(q_2\) are two range query circles. Ten FOVs \(f_1, \ldots, f_{10}\) are part of the same video data, named \(V\), with \(t_1, \ldots, d_{10}\) being their corresponding timestamps.

Figure 10: Query result representation through video segments. Circles \(q_1\) and \(q_2\) are two range query circles. Ten FOVs \(f_1, \ldots, f_{10}\) are part of the same video data, named \(V\), with \(t_1, \ldots, d_{10}\) being their corresponding timestamps.

Potential use of MediaQ for diverse video applications.

By providing an omnidirectional scene through one image, panoramic images have great potential to produce an immersive sensation and a new way of visual presentation. Panoramas are useful for a large number of applications such as in monitoring systems, virtual reality and image-based rendering. Thus, we consider panoramic image generation from large-scale user-generated mobile videos for an arbitrary given location.

To generate good panoramas from a large set of videos efficiently, we are motivated by the following two objectives:

- **Acceleration of panorama stitching.** Panorama stitching is time consuming because it involves a pipeline of complex algorithms for feature extraction, feature matching, image adjustment, image blending, etc.

- **Improving the quality of the generated panoramic images.** Consecutive frames in a video typically have large visual overlap. Too much overlap between two adjacent video frames not only increases the unnecessary computational cost with redundant information \[13\], but also impacts blending effectiveness and thus reduces the panorama quality\(^3\).

MediaQ can select the minimal number of key video frames from the videos based on their geographic metadata (\(e.g.,\) locations and directions). Several novel key video frame selection methods have been proposed in our prior work \[20\] to effectively and automatically generate panoramic images from videos to achieve a high efficiency without sacrificing quality. The key video frame selection criteria of the introduced algorithms based on the geo-information are follows:

- To select the video frames whose camera locations are as close as possible to the query location;

- To select video frames such that every two spatially adjacent FOVs should have appropriate overlap since too much image overlap results in distortions and excessive processing for stitching while too little image overlap may result in stitching failure.

\[http://www.cs.ou.edu/class/digitalmedia/Photomerge_Help/Using_Photomerge.htm\]

3.6 Social Networking

In addition to the basic functions of media content collection and management, MediaQ also provides the following social features: group sharing and region following of media contents.

**Group Sharing.**

In MediaQ, users can join in multiple community groups (\(e.g.,\) University of Southern California Chinese Students & Scholars Association (USC CSSA), USC CS). In a community group, users can share their media contents. In our system, before uploading the recorded videos/images, we allow users to select with which group they want to share the videos/images. There are three sharing options: public, private, and group.

**Region Following.**

Different from the person following and topic following in existing social network services such as Twitter, MediaQ proposes a new concept of “Region Following”, \(i.e.,\) MediaQ users follow spatial regions. For example, a Chinese student studying in the U.S. may follow his/her hometown of Beijing as the following region. Then, any public media content covering the hometown will automatically be brought to the attention of the student immediately after it is uploaded.

3.7 Mobile App

The MediaQ mobile app is a complementary component of MediaQ web system. The primary goal of the mobile app is the collection of media contents accompanied with their metadata by exploiting all related mobile sensors, especially representing the spatial properties of videos. Figure 11 depicts the design of the mobile app. It is comprised of four main components, \(i.e.,\) the media collection component, the user verification component, the GeoCrowd component, and the storage component. The media collection component is responsible for capturing video data and their metadata. Thus, while the user is recording a video, various sensors are enabled to collect data such as location data (from GPS) and FOV data (from digital compass). A timer keeps track of the recorded sensor data by relating each sensor record to a timestamp. The correlation of each record with a timestamp is extremely important because video frames must be synchronized with the sensed data. In addition, user data are added to the metadata and a JSON-formatted file is created.

The mobile app provides the interface to register and login to the MediaQ system. After login, users can use their device to record videos and upload them to the MediaQ server. However, at times users may not have Internet access for login due to unavailable wireless coverage. In such cases users can still record a video and store it locally without logging into the system. Afterwards, when Internet access becomes available they can upload it to the server. The reason behind this is that every video belongs to a user and the server needs to know who the owner is. We can only achieve that when the users are logged in to the system. After capturing a video, the mobile user is able to select which videos to upload, while others can remain in the device. Before uploading, the user can preview the recorded videos and their captured trajectories to ensure that each video’s metadata
are correct and the quality of the video is acceptable. As discussed in Section 3.3.4, GeoCrowd is integrated into the MediaQ mobile app to support on-demand media collection.

4. EXPERIMENTS

We have tested the implementation of MediaQ system in several real world cases.

4.1 Nato Summit 2012 Coverage

We used MediaQ as media platform for covering the NATO Summit event that was held in Chicago in May 2012. This was a joint collaboration between the USC Integrated Media Systems Center, the Columbia College at Chicago, and the National University of Singapore. More than twenty journalism students from Columbia College at Chicago covered the streets in Chicago during the event using iPhones, iPads, and Android phones as the video collecting devices. The focus of this experiment was mainly on real-time video data collection and searching of videos.

In the experiments, we used a Linux server machine with an 8-core processor, 8GB of main memory, and 10 TB of disk space. A gigabit fiber network was connected to the server. We supported two types of videos: 480p (720 × 480 or 640 × 480) and 360p (540 × 360 or 480 × 360). 480p was the original video quality recorded with mobile devices, whose bandwidth requirement was around 2 to 3 Mbps. 360p was the transcoded video quality of the original 480p video and its bandwidth requirement was 760 kbps. By default, 360p video was served during video streaming.

During a three day period, more than 250 videos were recorded and uploaded. Videos were collected in two different ways. One group of recorded videos was stored in smartphones and uploaded later when enough Wi-Fi bandwidth was available. The other group of recorded videos was uploaded in a streaming manner for real-time viewing. The challenge for real-time streaming was the lack of enough cellular network bandwidth since thousands of people in the streets were sharing the same wireless network. Thus, we installed two directional antennae at the corner roofs of the Columbia College campus to cover two target areas. A wired fiber network was connected from the antennae to the MediaQ server. Several students were carrying backpacks to carry directional cantenna for wireless Wi-Fi connections to the antennae installed on the roofs. These worked as wireless access points (hotspots) for nearby students. Thus, private wireless communication was available when a line of sight between antennae and cantenna was maintained (see Figure 12).

Overall, video collection and uploading from mobile devices worked satisfactorily even though some transmissions were retried several times to achieve a successful upload. We observed that the performance of the server significantly depended on the transcoding time of uploaded videos (which also depends on the capacity of a server hardware). The transcoding time per 5-second-long video segment varied from 6 to 20 seconds (on average 10 seconds). Usually, uploading happened in a bursty manner, i.e., all incoming video segments (unit segment) were scheduled for transcoding immediately. At the server, we observed that the peak number of video segment transcoding jobs in a queue was around 500 at the peak time. During the busiest day, the queue length varied from 100 to 400. Since a newly uploaded video is not retrievable until all previous transcoding tasks are completed, there existed some delays between the uploading time and the viewing time.
4.2 PBS Inaugblog 2013

We used MediaQ as the media coverage platform for the Presidential Inauguration in Washington DC in January 2013. It was a joint collaboration with the PBS Newshour College Reporting Team. More than 15 journalism students selected from all across the United States covered the streets in Washington DC during the event using the MediaQ Android app as the video collecting device and MediaQ as the server. This experiment mainly focused on video data manipulation and presentation, especially in the generation of panoramic images from the collected videos from smartphones.

Figure 15: Example images from the PBS experiments.

Figure 16 provides an example for visual verification on the panoramic image generation algorithm discussed in Section 3.5. This panorama was generated by selecting the 17 “best” video frames (the 17 frames were from 3 different videos) among 69,238 candidate frames. The total processing time (the selection time and the stitching time) was 11.8 seconds. This example illustrates that video geo-metadata can facilitate the generation of panoramic images efficiently.

4.3 GeoCrowd

Although the GeoCrowd component, which was the most recent implementation, was not fully utilized in the previous experiments, we have tested the feature within the USC campus with real scenarios. Specifically, geospatial tasks were created through the MediaQ web application, by selecting different locations around the USC campus and varying task constraints (i.e., expiration date, number of videos to be crowdsourced, and task descriptions). An example of the task posting is shown in Figure 17. To imitate multiple workers, we used the MediaQ mobile app with different user accounts to send multiple task inquiries. The rectangular regions of the task inquiries were intentionally chosen as multiple cases such as none, one or multiple tasks, while varying the maxT parameter (see Section 3.3). Figure 18 demonstrates a case when a spatial region is selected and a new task inquiry is sent to the MediaQ server. The GeoCrowd algorithm was scheduled to solve an MTA instance every one minute, and new workers’ assignments were inserted into the database. Thus, assigned workers were automatically notified of their task assignments within one minute. The results were correct at this campus-scale experiment with ten workers. However, this experiment was for proof of concept. We plan to run more tests on a larger scale using both real data (i.e., tasks and task inquiries) from public users and also synthetic data.

5. CONCLUSIONS AND FUTURE DIRECTIONS

MediaQ is an online mobile media management framework that lets public users store their recorded mobile multimedia content such as images and videos. It provides unprecedented capabilities in organizing and searching the media contents using W4 metadata from advanced geospatial analysis and video analytics technologies. Through various experiments, we demonstrated that MediaQ can enable media collection with geospatial metadata, keywords extraction, innovative content browsing and search capabilities, and media content sharing.

So far, we focused on outdoor videos where GPS signals are available. To manage indoor videos without GPS signals, we are working on including indoor geo-tagging with indoor localization techniques. We are also adding more visual analytics, such as facial recognition, to annotate people. Once basic annotation about people is added, we will extend our system towards social network feature for videos.

Acknowledgments

This research has been funded in part by Award No. 2011-LJX-K054 from the National Institute of Justice, Office of Justice Programs, U.S. Department of Justice, as well as NSF grant IIS-1320149, the USC Integrated Media Systems Center (IMSC) and unrestricted cash gifts from Google.
and Northrop Grumman. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of any of the sponsors such as the National Science Foundation or the Department of Justice. This research has also been supported in part by the Singapore National Research Foundation under its International Research Centre @ Singapore Funding Initiative and administered by the IDM Programme Office.

6. REFERENCES