MediaScope: Selective On-Demand Retrieval of Media from Mobile Devices

Ramesh Govindan

February 26, 2013

1 Introduction

Cameras on mobile devices have given rise to significant *sharing* of media sensor data (people share photos and videos).

However, capabilities of mobile device cameras have outstripped the cellular network capability; significant cellular data usage is required to upload media, and this costs money.

This has resulted in an *availability gap*: media is shared/uploaded often much later than the data is generated.

In this paper, we take the first step toward bridging this availability gap. We are motivated by the following scenario: users at a mall or some other location take pictures and video of some event (e.g., an accident or altercation). An investigative team wants visual evidence of the event.

We develop a system called MediaScope that permits concurrent timely geometric queries in feature space on media data that may be distributed across several mobile devices. MediaScope queries permit *searches* for images and videos closest to a given image in future space, *spanners* that the retrieve samples of images that span the feature space, or *cluster representatives* that are samples of natural clusters in the feature space.

...

2 Motivation and Challenges

2.1 Motivation

Cellphones with build-in high resolution camera becomes more and more popular. People take photos and videos with their phones all the time.

However, bandwidth limitations mean that image and video sharing has an availability gap. Explain how graph was taken.

Describe the implications of availability gap: lots of useful sensing information lacks *freshness.* This problem is not going away, because mobile device memories are fairly large. A 64 GB ipad can store over 100,000 photos; with a 2 GB monthly plan, these photos would take over a year to upload using cellular.

Many apps could be enabled by instant access to *fresh* visual information:

- Surveillance example (mall/etc.)
- Sportswriter looking for perfect picture of a play
- others...

Main focus of paper: looking for ways to bridge the availability gap.

2.2 Approach

We propose to explore ways to retrieve stored images from mobile devices on-demand. Our approach is inspired by *image search* techniques, which support similarity searches on image feature space.

Briefly describe how image searches work. Start by describing how features are extracted (give concrete examples of features). Then described actual similarity metrics etc. (This might need a separate subsection).

On top of this image similarity search primitive, we build a query interface that supports the following queries:

- top K matching images
- spanners or contours in feature space
- clusters in feature space

(in describing each query, discuss carefully practical examples of why the query may be useful).

Queries may:

- be constrained by other attributes (e.g., location, time, freshness etc.)
- may have timeliness associated with them: i.e., constraints on returning maximal information within a specified time limit.

This approach works well because:

- image features are relatively compact
- only matching images are retrieved, thereby minimizing bandwidth usage

Downside of our approach:

- current image search-based approaches incur a *semantic gap*; they don't understand the semantic information in images and can be imprecise; in our system, we assume that the retrieved images are further filtered by humans in order to make semantic sense

2.3 Challenges of approach

- Query timelines
- Variable and limited bandwidth availability on the mobile devices
- Designing geometric algorithms for the queries
- Efficiently supporting multiple queries, and permitting resource sharing between queries while still allowing maximal information upload

Other challenges: our approach is essentially a form of crowdsourcing. In this paper we have not addressed privacy and incentive issues; other work has done that [Medusa].

We have design a system called MediaScope which addresses all these challenges, in a manner described below.

3 MediaScope Design and Implementation

3.1 System Architecture

The system consists of two partitioned components: MSCloud is the Mediascope component that runs on a cloud-based service, and MSMobile is the component that runs on the mobile device.

MSMobile uploads feature vectors for each image to MSCloud. Explain why this is beneficial.

Users pose search queries to MSCloud, which has two components:

- MSCloudDB: image features are stored here
- MSCloudQ: the query processing engine on the cloud

Finally, MSMobile processes metadata and uploads search results.

(Add a figure which shows the query result workflow).

3.2 Design: Supporting multiple concurrent queries

The central, and most challenging component of Mediascope, is support for concurrent queries.

In the design of MediaScope, we have to design for a situation in which there may be multiple concurrent queries, such that the answers to these queries cannot all be satisfied within their timeliness constraints because of bandwidth limitations.

In this case, our goal is to attempt to maximize the amount of information delivered across all queries. In particular, this means that it would be preferable to obtain as much of the "high order bits" of the query result for each query, than to obtain all of the results for one query.

Our approach uses two mechanisms to address this challenge:

- a system defined assignment of "credits" to individual queries: this assignment roughly determines the relative proportion of resources that this query can use
- for each query, we also define a method by which credits can be allocated to query results and design a scheduling algorithm that attempts to maximize the total number of credits retrieved

Thus, query semantics are "approximate" necessarily because of timeliness constraints one may not get complete queries; level of completeness can be varied by adjusting the credits assigned to each query.

3.2.1 Queries and Credit Assignment

Algorithm for each different query not only generates a list of media objects to upload, but each media object is associated with a credit, indicating the "importance" of such object.

Discuss design of different queries:

- top K: basically a nearest neighbor search in feature space; discuss how credits are assigned to each object in the result
- spanner: an MST in feature space; discuss how credits are assigned.
- clusters: discuss, for each of these, credit assignment
  - clusters with most users (in the largest cluster, pick one representative photo)
  - clusters by topic: pick one representative photo in each cluster

Extensibility:

- discuss what someone might have to do to extend this framework and add new queries (e.g., a convex polytope)
  - define a fast implementation of the query
  - define credit assignments for query results

3.2.2 Credit-based Scheduling

Server evaluates a query with the metadata and assigns credit values Then, tasks phones to upload media objects with assigned credit values

Scheduling on the phone is credit-based:

- describe the credit-based algorithm
  - show how the algorithm works with an example
  - include pseudocode for the algorithm (pseudo should consider query arrival etc.)
- discuss the theoretical properties of this algorithm

3.2.3 Feature extraction on the phone

Describe what feature extraction algorithms we have tried

Discuss challenges of feature extraction on the phone

- video frame extraction
- image feature extraction

For video frame extraction, show the result of the evaluation of the time taken to extract at different rates; then say we picked 1Hz for this paper.

Image feature extraction:

- feature extraction from large photos exceeds memory
- show the resizing/accuracy trade-off

3.2.4 MSCloud to MSMobile communication

Use Medusa, a crowd-sensing platform

- (since this is a double-blind submission, do not refer to this as our work)
- in one paragraph, try to describe how Medusa works

Say what we had to add to Medusa to support MediaScope

- polling-based notifications to reduce latency
- support for task deletion
- push notification using C2DM
- support for machine generated Medusa tasks
- others?

Then, describe how MediaScope uses Medusa:

- generating metadata is a task;
- upload is a task
- credit based upload is a new transfer manager component
- Medusa has a callback to MediaScope to notify object/metadata upload

4 Evaluation

4.1 Methodology

Briefly discuss our implementation, saying how many lines of code (use SLOCCount if you can).

Say that all our results are from implementation. Are we including results from a simulator?

4.2 Metrics

Total credit retrieved, for different query mixes

- compare against a variety of alternative approaches:
  - max credit, earliest deadline, etc.

System overhead:

- time to upload a small file end-to-end: use Wifi and cellular network
- breakdown of latency:
  - where is the latency going? (Medusa server, notification, medusa phone etc.)

4.3 Results: Credit retrieval comparison

4.4 Results: Overhead

5 Related Work

- Image Search
- UIUC related work, other mobile papers

6 Conclusions