Poster: User Location Fingerprinting at Scale

Puneet Jain
Duke University
puneet.jain@duke.edu

Justin Manweiler
IBM Research
jmanweiler@us.ibm.com

Romit Roy Choudhury
UIUC
croy@illinois.edu

ABSTRACT

The goal of this project is to uniquely fingerprint an environment in the visual domain. The fingerprinting should be such that if a user takes a video swipe of the environment with the phone camera, her precise location can be immediately inferred. The problem is challenging because the user is continuously scanning the environment and the opportunity of capturing unique parts is fleeting. A typical environment may consist of repeating visual patterns such as ceiling, floor, wall texture, etc. and a painting hanging on the wall may be the only unique identifier of the location. Our approach to solve this problem is motivated by the ability of human brain in differentiating two similar looking environments by observing subtle differences in them. Our contribution in this work is two fold: (1) summarize global uniqueness of a location in few bits of information to quickly determine whether the current phone camera view contains a unique part, (2) scale this notion of uniqueness to buildings of arbitrary sizes yet delivering near real-time performance. Once fully developed, we believe our approach can apply to accelerate mobile augmented reality and correct indoor localization errors.

Categories and Subject Descriptors

H.3.4 [**Information Storage and Retrieval**]: Systems and Software

General Terms

Algorithms, Design, Experimentation, Performance

Keywords

Continuous Vision, Cloud Offloading, Localization

1. INTRODUCTION

With advances in the mobile computing, camera-based applications are slowly moving from offline processing on a stationary image to real-time processing on continuous video feed. Unfortunately, phone's processing power is limited and network latencies are too high for real-time computation offloading. Therefore, hybrid architectures are gaining popularity: where phone does light-weight processing to avoid resource wastage and the cloud periodically performs heavyweight computation to deliver high accuracy [3, 4]. The current generation phone cameras are capable of capturing up to 30 frames in a second; therefore offloading decision needs to be made within a few milliseconds. In this project, we are trying to design a system, which can quickly identify user location from a video swipe. Our intuition is that an environment possesses enough visual diversity that most locations can be uniquely identified (without moving) if scanned for a sufficient interval. The intuition is synonymous to functioning of our brain, which helps us in locating ourselves in an already visited environment by identifying unique visual anchors. Harnessing this intuition in a system is hard because the user is continuously moving the camera and a video frame containing unique part may be transient. If the computation on the phone was not a bottleneck, matching every video frame with pre-captured visual data of the environment could have solved the problem. This brings a need of uniqueness summarization scheme, especially in a way that it can be queried efficiently on current generation phones. Note that the summarization doesn't have to be accurate, since periodically, the computation can be offloaded to the cloud to achieve higher confidence.

Motivated by these intuitions, we propose *InstantLoc*: a system capable of identifying user location from a video swipe in the environments of arbitrary sizes. InstantLoc consists of three major components : (1) a Google Project Tango [1] based wardriving app to quickly construct a database of visual features of various parts of the building. A user is required to walk across the building with the InstantLoc tango wardriving app. The app captures and stores colored photographs, depth-map, and the location of the user relative to the start position. (2) cloud cluster to perform the following tasks: (a) process the captured data and construct feature database (b) efficiently answer location based queries emerging from user phones. For database construction, visual features such as keypoints are extracted from the colored photographs and 3D point cloud from the depth-map. Since color and depth cameras are co-located on the device, each keypoint can be mapped to a 3D point in the point cloud. By merging these two information, a mapping between keypoints and corresponding 3D points is constructed. We replicated these mappings on a distributed key-value store for efficient query and retrieval. Note that given plethora of key-value stores available today, selection of the one fitting this problem is important. A keypoint descriptor is a multi-dimensional vector and efficiently answering queries based on them requires complex partitioning of the database. Motivation behind the platform selection and database summarization is discussed in detail in the next section. (3) A phone camera app, which captures and extracts visual features from continuous video frames on the phone. The extracted features are first tested against bloom-filters for rough estimate of uniqueness in the global key-value store. If sufficient evidence is found, selected set of most unique features are uploaded to the cloud and a reverse look-up (from features to 3D point cloud) is performed.

MobiCom'15, September 7–11, 2015, Paris, France.

http://dx.doi.org/10.1145/2789168.2795175.

The result of lookup operation is a subset of 3D point cloud from where indoor location of the user can be estimated using advance mobile sensing based techniques.

2. SYSTEM DESIGN

In this section, we first introduce existing technologies used in InstantLoc and then expand on their usability in the following components: (1) Wardriving App (2) Client App (3) Server.

HyperDex Primer

Distribute key-value stores are now widely used across various cloud computing applications. These key-value stores often achieve benefits over traditional SQL databases by allowing restrictive set of search primitives e.g., object can be retrieved only based on the key it was inserted. On contrary to this, HyperDex [2] is a distributed key-value store, which provides an efficient querying on secondary attributes or range queries. HyperDex achieves this using the concept of hyperspace hashing where multiple attributes are mapped into a multidimensional hyperspace. For example a traditional SQL table with three columns is mapped into a 3D dimensional space. A region in this hyperspace is stored on a particular server. An object is queried by tessellating the hyperspace into a grid of non-overlapping regions.

We apply this concept of hyperspacing hashing to our multidimensional feature database. In computer vision, a keypoint descriptor is represented by set of fixed length real numbers. These numbers can be considered as attributes of a hypercube. Therefore, storing keypoint descriptors in a HyperDex cluster can enable their efficient lookups.

Project Tango Primer

Project tango is a prototype tablet-like hardware built by Google. Project tango combines 3D motion tracking with depth sensing to enable precise indoor localization and 6-DOF pose estimation. While not ready for widespread use due to limitations in energy, portability, and stability; the following properties of project tango still makes it good fit for the research and wardriving purpose:

Area Learning

Project tango tries to learn visual features (such as the edges, corners) of the area it visits. The concept of area learning is very similar to the concept of visual-SLAM in robotics. It uses learnt features in the past to: (1) correct drift by resetting deadreckoning errors at places seen before (2) localize by orienting and positioning itself with respect to the same place.

Depth Perception

Project tango is among first hardware prototype to use IR depth sensor on a portable hardware. It uses projected IR light to estimate depth based on how the light is shaped by objects in the environment. The depth information is made available to the application layer in the form of XYZ_{ij} point cloud, where (X, Y, Z) represents 3D coordinates of visible points and (i, j) represents its pixel coordinates in raster format.

2.1 Wardriving App

The wardriving app is a project tango app used to collect a series of snapshots consisting of 6-DOF pose of the device (w.r.t. to the start location), RGB image in the view, and a lower resolution depth map of the corresponding view. To wardrive a venue, an user needs to walk across the building – periodically capturing snapshots and uploading them to the cloud server. These snapshots are pre-processed to create mapping between image keypoints and corresponding 3D locations. The number of such mappings inside a building can be exceedingly large, on the order of millions to billions. This process is currently manual but it can be automated using area-covering robots like Roomba in the future.

2.2 Client App

The client app is a camera app, which runs on user's phone and is used to query indoor location. When the client app starts for the first time, the server sends precomputed bloom filters to the phone (explained in the next section). These counting bloom filters encode frequency of database features in a certain floating point hyper-ranges. To query a location, user makes a swipe in the environment. During the swipe, features from the non-blurred frames are extracted and their frequencies on the server database are extracted from the bloom filter. A feature of high frequency implies it is less unique, hence cannot help in disambiguating two locations. Therefore, all extracted frequencies are sorted in the increasing order and a small fraction of less frequent features is uploaded to the server for the location identification.

2.3 Server Implementation

Our server implementation consists of four parts:

Input pre-processing

In the pre-processing step, our goal is to construct image keypoint to 3D point mapping. These mappings need to be extracted from the snapshots uploaded from the wardriving app. We achieve this in the following steps: (1) Each snapshot contains device's rotation and translation with respect to the start of the app. These values are converted to a global coordinate system by starting tango app from a known GPS location and applying respective transformations. (2) From the RGB image in the snapshot, keypoints and descriptors are extracted. A keypoints is typically represented by a pixel value (i, j) (2D image coordinates) and corresponding N length descriptor vector. (3) Using depth map of the snapshot, a pixel (i, j) is also mapped to a 3D point (X, Y, Z) in the world. Note that (X, Y, Z) are initially captured in the device frame of reference, they need to be converted to the world coordinates. (4) Since the project tango depth map resolution is typically lower compared to the RGB image – it is interpolated by applying linear filters. Once done, every keypoint may be mapped to the corresponding 3D point. (5) Since a 3D point may be visible in multiple snapshots, snapshots are merged using iterative point cloud algorithms (ICP) to avoid duplications. (6) All keypoints to unique 3D point mappings are created and stored in the HyperDex cluster.

Bloom Filters

The keypoint to 3D point mappings are combined at a global cloud service with mappings collected from many other buildings. At this point, the database size could range in the size of hundreds of millions to billions. Periodically, an O(n.log(n)) scan on the entire data store is conducted. For each keypoint, we make constant number of range queries (hence O(log(n))) in keypoint descriptor space. The output of each query is a count of the number of keypoints "similar" to the queried keypoint. For our purposes, more similarity is bad – implies a

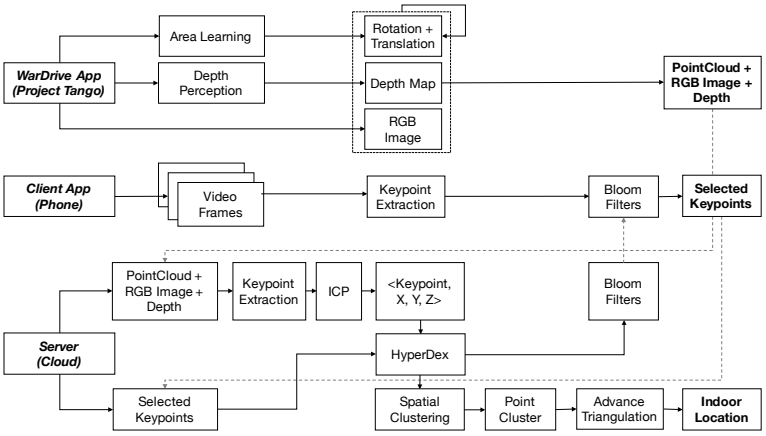
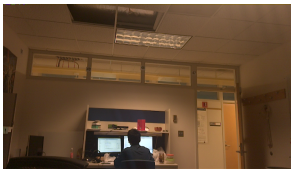
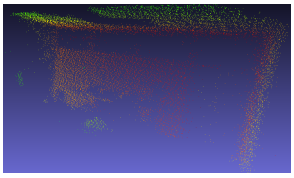


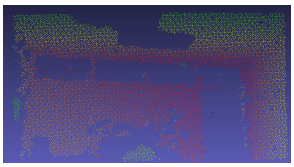
Figure 1: System Overview. Top: wardriving-app. Middle: phone-app. Bottom: Server consisting of HyperDex and vision/pointcloud libraries.



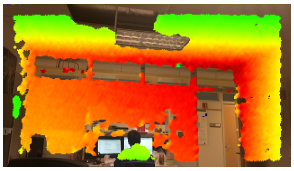
RGB image



Depth map 3D



Depth map 2D projection



Processed depth map on the RGB image

Figure 2: Pre-processing life-cycle of a snapshot on the server

keypoint is less unique, less discriminating. Consider the distribution of such counts and order them in decreasing order. These counts can be divided into sets of quantiles – likely candidates, the 0^{th}, 50^{th}, 75^{th}, 90^{th}, 95^{th}, 99^{th} percentiles. For each quantile, we create a set of bloom filters. Each bloom filter here encodes keypoints that meet a "uniqueness criterion" corresponding to the respective quantile. For example, a keypoint that only appears once would be encoded in every bloom filter. A common keypoint (e.g., the corner of a tile in a repeating pattern) might only appear in the 0th or 50th. All of the bloom filters are pushed to the client app when it starts for the first time and refreshed periodically. Though the global data store is expected to be huge (many TB), the bloom filters can be several orders of magnitude smaller (MB).

Feature Matching

When the user opens client app for the first time, bloom filters are downloaded on the device. An user making a swipe continuously captures image and inertial sensor data such as accelerometer and gyroscope. From the image, keypoints are extracted – typically ranging between 1000-20K. Each keypoint is tested against the bloom filters. This identifies how "rare" each keypoint is, relative to the global data store. Each keypoint is ranked based on it's rarity – selecting the top K most rare points as candidate for uploading on the server (K typically range between 5-50). On the server, We query the global data store with these K selected keypoints, returning N_k 3D points corresponding to each of them (a keypoint may belong to multiple 3D points). This gives us $\sum_k N_k$ 3D points. On these 3D points, spatial clustering such as k-means is applied – leaving us with the largest cluster, discarding others.

Location Estimation using Triangulation

For each of the point in the largest spatial cluster, it's slope from a random origin is computed. If two points were visible in the same frame, the exact angular disparity is known. Otherwise it can be estimated with gyroscope/accelerometer or optical flow. Finally, we use "enhanced" triangulation technique discussed in [5] to triangulate to a single location from arbitrary number of such points. The solution to this optimization provides us exact location of the user.

3. CONCLUSION

Many emerging mobile computing applications are continuous vision based. The primary challenge these applications face is computation partitioning between the phone and cloud. The indoor location information is one metadata that can help these applications in making this decision. In this extended-abstract, we propose a vision based scheme to uniquely fingerprint an environment which can in turn be used to identify user's location from the uploaded visual features. Our approach takes into account that the opportunity to identify location is fleeting and the phones are resource constrained – therefore minimal yet sufficient computation needs to be performed to make the offloading decision. Our work aims to achieve near real-time performance while scaling to buildings of arbitrary sizes. The current work is in preliminary stages but holds promise for the future – may apply to many applications in this area.

4. REFERENCES

[1] Project tango. https://www.google.com/atap/projecttango/, 2014.

[2] R. Escriva, B. Wong, and E. G. Sirer. Hyperdex: A distributed, searchable key-value store. *ACM SIGCOMM Computer Communication Review*, 42(4):25–36, 2012.

[3] P. Jain, J. Manweiler, A. Acharya, and K. Beaty. Focus: clustering crowdsourced videos by line-of-sight. In *SenSys*, page 8. ACM, 2013.

[4] P. Jain, J. Manweiler, and R. Roy Choudhury. Overlay: Practical mobile augmented reality. In *MobiSys*, pages 331–344. ACM, 2015.

[5] J. G. Manweiler, P. Jain, and R. Roy Choudhury. Satellites in our pockets: an object positioning system using smartphones. In *MobiSys*, pages 211–224. ACM, 2012.