OverLay: Practical Mobile Augmented Reality

Puneet Jain
Duke University
Durham, NC, USA
puneet.jain@duke.edu

Justin Manweiler
IBM Research
Yorktown Heights, NY, USA
jmanweiler@us.ibm.com

Romit Roy Choudhury
University of Illinois
Urbana-Champaign, IL, USA
croy@illinois.edu

ABSTRACT

The idea of augmented reality – the ability to look at a physical object through a camera and view annotations about the object – is certainly not new. Yet, this apparently feasible vision has not yet materialized into a precise, fast, and comprehensively usable system. This paper asks: *What does it take to enable augmented reality (AR) on smartphones today?* To build a ready-to-use mobile AR system, we adopt a top-down approach cutting across smartphone sensing, computer vision, cloud offloading, and linear optimization. Our core contribution is in a novel *location-free* geometric representation of the environment – from smartphone sensors – and using this geometry to prune down the visual search space. Metrics of success include both accuracy and latency of object identification, coupled with the ease of use and scalability in uncontrolled environments. Our converged system, *OverLay*, is currently deployed in the engineering building and open for use to regular public; ongoing work is focussed on campus-wide deployment to serve as a "historical tour guide" of UIUC. Performance results and user responses thus far have been promising, to say the least.

Categories and Subject Descriptors

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

General Terms

Algorithms, Design, Experimentation, Performance

Keywords

Augmented Reality, Gyroscope, Localization, Mobile, Tagging

1. INTRODUCTION

The concept of mobile augmented reality is tantalizing. Researchers, designers, and the authors of science fiction have converged on the vision of using a hand-held camera as a sort of "magnifying glass" to browse the physical world. Digital annotations may seamlessly appear when the camera is pointed at an object. Looking at spaghetti in a grocery store may trigger recipes and product reviews; pointing the camera at a sculpture in a museum may pop up the artist's bio data; viewing a corridor in an airport can display restaurants located downstream in that path. Despite such exciting potential applications, a generic and usable solution remains elusive. The core challenge lies in carefully mitigating the tradeoff between accuracy and latency, i.e., the ability to precisely recognize an object and pop-up its annotation without perceivable time-lag. Given that hundreds of objects may be annotated in the same vicinity, slight inaccuracies or delays can degrade the quality of human experience. Moreover, authoring and retrieving annotations should be simple, and on the spot. If Alice is the first to annotate a painting in a museum, Bob should be able to view her annotation in the very next moment.

It is natural to wonder *why this is not a solved problem despite substantial research in augmented reality.* We began this project with the same question and realized that while several prototypes have been built in isolation – each making technical contributions in vision, sensing, and even information fusion – to the best of our knowledge, no system has made a holistic effort end to end [11, 13, 20, 35, 41].

Pure sensor based approaches such as Wikitude [9] rely on the smartphone GPS to position the camera, the compass to infer direction, and the accelerometer to identify the tilt. By computing the camera's 3D line of sight (LoS) from these data, Wikitude estimates the object that should be within the camera's viewfinder, and pops up the corresponding annotation. Unfortunately, sensory data has proven largely insufficient. Compass error measurable in tens of degrees severely derails the LoS accuracy; GPS errors of $5m$ or more exacerbates the condition. The application works when large objects are located near the camera, such as a user looking at the White House while standing at its fenced perimeter. For less favorable cases, especially when the annotation density is high, results are far from acceptable. Indoor environments lacking precise localization are of course out of scope.

In contrast to sensory approaches, computer vision enables a tradeoff of latency for better accuracy. Latency arises from matching the image in the camera's view against various images in the annotated database. When the match occurs, the results are precise – the annotation perfectly pops up on the screen atop the object. However, in uncontrolled environments, users view an object from different angles, under different lighting conditions, and from different distances – all adding to the complexity of the problem. If the database contains many images of the same object (e.g., Alice and many others have annotated the same painting), image matching exploits this diversity to gain robustness. However, it takes longer. With only one or few images in the database per object, the operation is faster, but at the cost of accuracy.

Recent years have witnessed hybrid approaches that fuse sensor data and computer vision [24, 28, 33]. By utilizing the location of the user and orientation of the phone, authors in [38] prune the search space for image matching. This certainly offers improvement, however, the lack of precise location and large compass fluctuations make them unreliable indoors. The latency is also in the order of several seconds, requiring users to take a picture and wait for the annotation. Real-time object browsing – like a magnifying lens – really warrants sub-second latency.

Based on extensive literature search, we conclude that despite meaningful technical advances, and many released apps and media articles, no single effort has fully delivered on the complete vision. There are holes in the end to end processing pipeline – achieving the strict latency and accuracy targets would warrant a holistic look at the system. The system also needs to relax any assumptions on location to be deployable universally in the near future. Finally, sheer engineering is necessary to cope with corner cases in uncontrolled environments, including hand vibrations in some users, some objects being identical, weak network connections, etc.

We aim to complete and test the vision – to deliver, from a top-down design, a viable mobile AR framework that enables a useable and rewarding experience. While many of the challenges that must be addressed are not fundamental, they are "hard" and previously undemonstrated. It needed many design iterations to get the pipeline sufficiently optimized so that a viable solution could even be within grasp. Initially, we expected our efforts to be primarily engineering. *However, we also found ripe opportunity for novel heuristics to improve accuracy, responsiveness, and ultimately, the human user experience.* Especially, we developed an optimization framework that underpins computer vision with a geometric representation of objects in the surroundings. We are able to learn that geometry from the natural human movements across space and time, bringing the system to a desirable level of robustness.

Finally, we evaluate our approach not only through micro benchmarks, but also live, through the natural usage of independent and unbiased volunteers. For the first time, we were able to observe real human behaviors when interacting with a viable mobile AR solution.

In developing our solution, we learned the key contribution of this paper: *Mobile AR is best addressed not through sensing, computer vision, or any modality in isolation. Instead, each may be most advantageously combined within a stateful model, underpinned by the geometry and time of motion. With this spatiotemporal awareness, its possible to deliver content to the human user, at consistent precision and sub-second latency.*

For a sense of the accuracy and realtime responsiveness of the prototype, we invite the reader to watch the following video demonstration of our live system. **http://synrg.csl.illinois.edu/projects/MobileAR**

2. MEASUREMENTS AND GUIDELINES

OverLay aims to enable real time augmented reality on today's smartphone platforms. This section develops basic design guidelines through measurements and observations.

Location and Orientation Sensors Inadequate

We believe that sensor-only approaches are inadequate to realize OverLay in any generalized settings. Even in outdoor settings, where GPS locations are precise to around $5m$, sensor based approaches fall short. Figure 1 shows measurements performed with Wikitude [9], a popular app on the app store that uses the phone's location, compass, and accelerometer to display annotations. The graph plots Wikitude's error in line of sights (LoS) measured by computing the perpendicular distance from the true object to the LoS. The median separation is around $12m$, implying that an object has to be at least $12m$ wide for the annotation to still be correct. Clearly, this offers limited applicability.

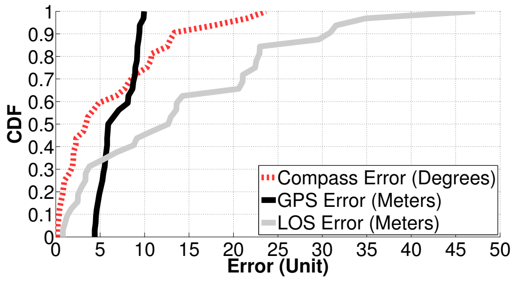


Figure 1: Difference in estimated LoS vs. ground truth when measured through the Wikitude app.

Indoor environments are far worse. Figure 2 measures compass errors due to ferromagnetic material in the ambience – as a user walks through a straight corridor, compass angle deviations are measured against ground truth. The median deviation is more than 15° with a heavy tail of up to 180°. Finally, indoor location is still not universally available, and some that are starting to roll out in a few places are limited to $5m$ accuracies, at best. Such accuracies easily derail a augmented reality approach. Thus, to make OverLay robust and immediately deployable in all environments, we must desensitize the solution to localization and orientation estimates.

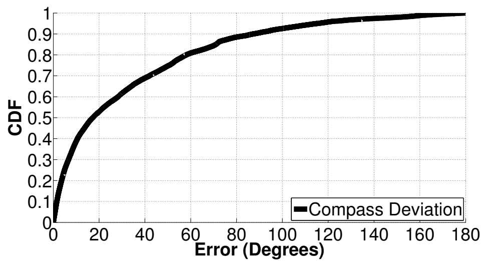


Figure 2: Distribution of compass deviation in various indoor environments.

Computer Vision and Cloud Offload Essential

The reality we intend to (digitally) augment is visual. In light of this, it seems most intuitive to take advantage of the camera, a sensor that "sees" the world similar to humans[1]. Of

[1]We assume current generation cameras in smartphones. While 3D cameras would open further opportunities, we wish to ensure the widest generalizability across devices today.

course, computer vision algorithms (needed to "perceive" objects and display annotations) warrant non-trivial computational support. The way in which this computation is to be delivered presents a fundamental design decision.

Google's Project Tango [6] convincingly presents the premise that tomorrow's hardware might have computational and (therefore) visual sensory powers far beyond anything on the marketplace today: a laptop-grade GPU able to perform heavyweight computer vision on-device, such as simultaneous localization and modeling (SLAM) through bundle adjustment. While the concept is fascinating, today's reality is quite different, and we target a nearer-term solution. It is also worth noting upfront that as we are without access to this hardware, we will not be able to compare our techniques against it.

Excluding such forward-looking hardware prototypes, we establish our second design guideline. Mobile devices do not have sufficient computational capability, even with embedded GPUs, to perform suitable computer vision for immersive (and thus compelling) Mobile AR. Contrastingly, today's cloud environments afford on-demand provisioning of vast computational resources, to include dedicated GPUs (especially, Amazon EC2 and IBM SoftLayer). As evidence of the performance contrast, Figure 3 shows CDFs of computational latency in extracting local image features from a 1080p video frame (using the state-of-the-art SURF heuristic) on a desktop CPU, GPU, and a current-generation mobile CPU. Mobile CPU performance is a factor of 1000 – *three orders of magnitude* – slower than desktop GPU. Any hope to attain real time AR today will probably be infeasible without cloud offloading.

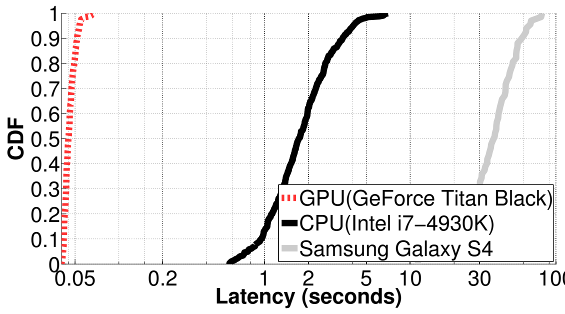


Figure 3: Compute latency: extracting SURF local features on GPU vs CPU (single thread) vs mobile CPU (single thread) for 1 HD-quality video frame.

Network Latency Dictates Lower Bound

The latency gain from cloud offloading is obviously offset by the network latency in moving images to the cloud. Figure 4 shows a CDF of latency for moving a single HD video frame from a mobile device, over Wi-Fi, to a local server for processing. Even with a low-latency connection, limited throughput makes realtime operation untenable. While reduced image resolution will reduce the data burden (and hasten transmission times), a commensurate reduction in computer vision efficacy is an undesirable penalty. In response, OverLay must be highly selective in *which* imagery it uploads. Selected portions of selected frames will need to be transmitted to reduce data transfer by orders of magnitude.

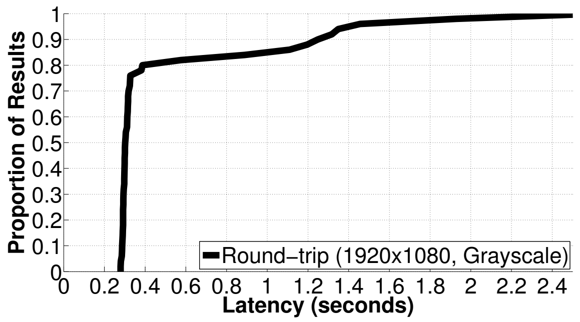


Figure 4: Network latency: upload time for a single HD-quality video frame (1920 x 1080).

Computer Vision: Still Too Slow

Cloud offloading only alleviates the resource crunch. Even when running on the cloud, the accurate computer vision techniques are still heavyweight (hence slow and perhaps unscalable to many users), while the lighter and quicker ones are less accurate. The choice of the suitable technique is a matter of engineering. Figures 5(a) and (b) plot image matching accuracy and computational latency of a range of well-established feature extraction and matching techniques. Unsurprisingly, there is a tradeoff. Our intuition is that accuracy must not be overly sacrificed – false positive and negative pop-ups will simply "break" the user's sense of immersion. The latency penalty has to be mitigated some other way, suggesting that computer vision is necessary but insufficient for delivering the end to end experience.

As a side note, its possible to bring down the latencies by parallelizing the image matching computation on many CPU/GPUs in the cloud. However, that would not scale to real-world many-user deployments. With an eye towards scalability, we perform our experiments on a single consumer-grade desktop GPU (resembling a cloudlet [36]), and support tens of users. The same techniques should hold with real-world user densities on a real-world cloud.

Opportunities for Geometric Optimization

The computational latency in Figure 5(b) is dominated by image matching, which is in turn a function of the number of candidates in the image database. Pruning the candidate set can aid in bringing down the latency to sub-second. To this end, it may be useful to develop a spatial understanding of the objects in the physical surrounding. If objects A, B, C, and D are known to be in spatial proximity, it may be possible to "prefetch" objects B, C, and D when the user is currently viewing object A. If among these four, A and B are known to be in angular proximity, only B can be prefetched. In the absence of location information, spatial proximity may be statistically inferred from the temporal separation observed between various pairs of objects. Angular proximity can be deduced from gyroscope rotation as users scan across objects.

By synthesizing sensor data through a geometric optimization framework, it may be possible to infer a spatio-angular representation of objects in a non-absolute coordinate system. In other words, anchoring any given object at the origin of such a coordinate system, it may be feasible to understand how other objects are relatively located. This allows for prediction and prefetching, offering opportunities to attain our real-time goals. Humans use such multi-sensor anchoring to

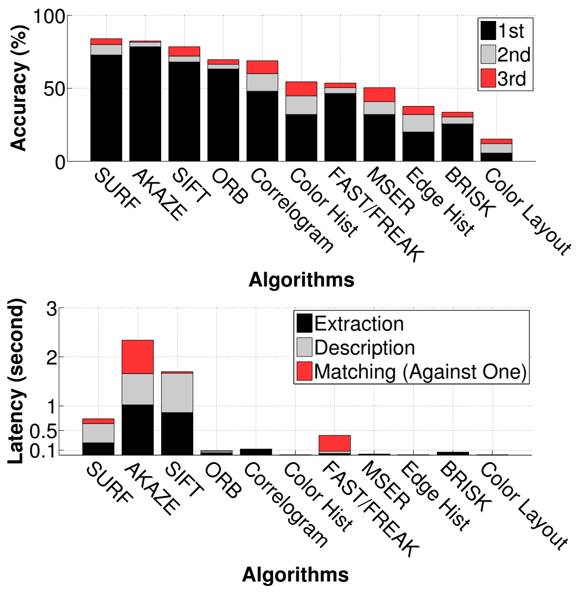


Figure 5: (a) Accuracy/latency of image matching based on local or global features. Accuracy for 1st, 2nd, or 3rd-best match plotted from an 100-image database. (b) Latency bars reflect stages of the matching process (all numbers for server CPU)

.

reason about their movement through environments; OverLay makes an attempt to mimic some of these abilities.

3. SYSTEM OVERVIEW

Desired User Experience

Our ideal end goal is as follows. As a user Alice points her mobile camera at an object in the physical world, an appropriate annotation pops up immediately atop the corresponding object. Alice moves her hand to browse other objects, and tags keep popping up with non-perceivable lag. If Alice wishes to annotate a particular object, she simply brings the object in the center of her viewfinder and enters the annotation (which could be text, multimedia, or even software code). Bob passing by that object soon after can look at the same object and see Alice's annotation pop-up on his screen. When multiple annotated objects are in the viewfinder, all the annotations are overlaid atop the respective object.

We have not been able to achieve this "seamless" goal – the current system requires a median of $480ms$ (includes $302ms$ network latency) to pop-up annotations, along with more than 95% image matching accuracy. Thus, when using OverLay, Alice must pause her camera at an object of interest, and the annotation pops up within a second (except in rare occasions). Figure 6 presents a screenshot of our prototype running on an Android smartphone. Even this relaxed goal was non-trivial.

Main Technical Modules

OverLay's end to end design firmed up after multiple iterations – in retrospect, we see three logical phases in the development process. (1) As an initial target, we focused on laying the groundwork with computer vision techniques, characterizing the best possible tradeoff between accuracy and latency. This phase included building the complete processing

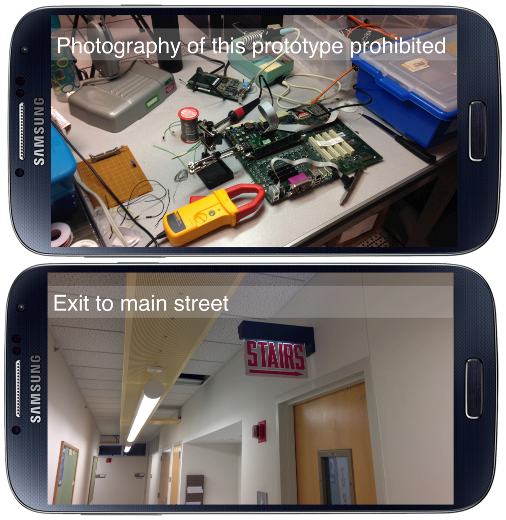


Figure 6: Live object retrieval using OverLay.

pipeline, evaluating a wide range of vision techniques, understanding their assumptions and possible modifications for our application, and finally reducing latency to the extent possible, through GPU parallelization, code optimizations, and sheer engineering. We do not make novel contributions here, nonetheless, the exercise was "hard" and time consuming to attain a desired degree of robustness and predictability. (2) With this framework in place, we heavily leveraged sensing (especially gyroscope) to reduce the amount of imagery that would be uploaded from device to server. This alleviated load on the GPU cutting the response time in half. (3) Finally, using sensor data from users, we developed an optimization framework to create the geometric representation of annotated objects. This pruned down the search space, translating to $4x$ load reduction in image matching and ultimately translating to sub-second response latency.

Terminology

Tag or Annotation: Used interchangeably, they refer to content associated to physical objects. The user may tag or annotate an object, and these tags/annotations pop up when a OverLay–enabled camera is pointed at it. Tags and annotations can be text, multimedia, or a trigger for more actions (e.g., purchase of an item).

Constant Scan: The ability to continuously browse the physical environment with the camera, even when the user is moving. Tags expected to pop up when camera pauses on an object.

Search Space: Set of candidate objects against which a given object is being matched. Our search space is a spatial/angular region surrounding the most recently matched objects (better explained later).

Micro/Macro Trajectory: Observed or predicted human path through physical space, exploited to refine the search space. A macro trajectory is substantial movement with translational and rotational components, such as walking and taking turns. A micro trajectory is minor hand movement, likely while standing in a fixed position – common when entering a room and rotating the camera to scan objects on the wall.

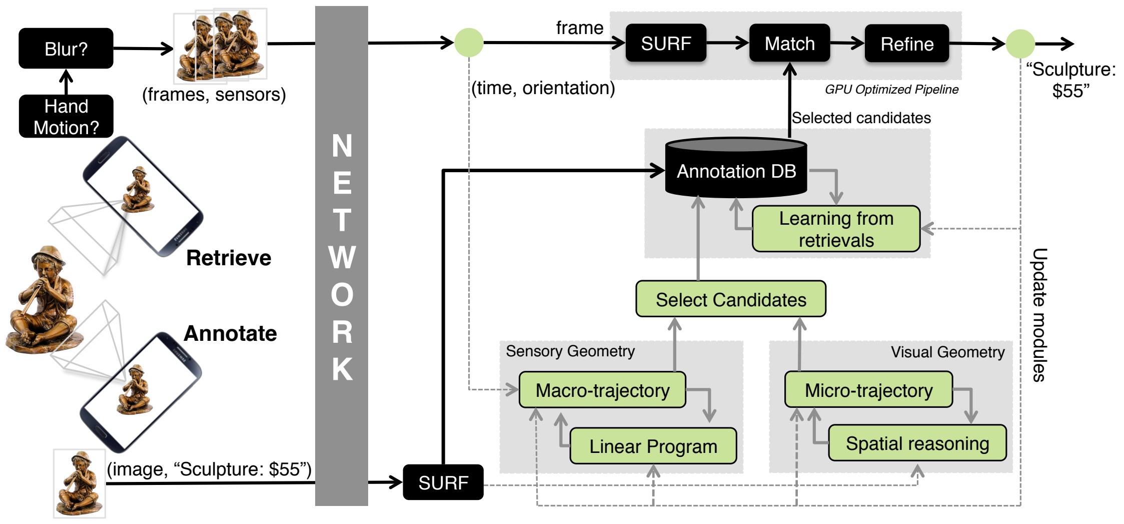


Figure 7: System Overview. Left: client-side. Right: offloaded compute, database on server (cloud).

4. SYSTEM DESIGN

Figure 7 illustrates our simplified system architecture. In the basic execution flow, an annotated object (bottom left) is uploaded to the OverLay server, where SURF features are extracted and systematically indexed in an annotated (image) database. When objects are retrieved later (top left), the phone uses vision and sensing data to filter out unusable frames caused by hand vibration and blur. The uploaded frames are processed through a GPU optimized pipeline – the operation includes SURF feature extraction, matching with selected candidate sets, and refining the match. If a confident match is found, the server returns the corresponding "annotation" to the smartphone for on-screen display.

OverLay underpins this basic execution pipeline with optimizations that prune the search space during retrieval. To this end, 3 modules are invoked at different places in the overall system. (1) Sensor data received during the retrieval process (i.e., time and gyroscope orientation) is fed into a "Sensory Geometry" module responsible for inferring the the spatiotemporal relationships between objects. Two objects retrieved in a corridor can be "connected" in terms of their relative time and angular separation. As new retrievals arrive, and as images get correctly matched (top right), all these information are fed back into a *linear program* to ultimately converge on a geometric representation of annotated objects in the environment. (2) Such inter-object relationships are also extracted in the visual domain – if object A and B are viewed in the same image, OverLay infers relationships such as "A is on the left of B". The "Visual Geometry" module uses data from annotated images to develop this understanding. (3) Finally, correctly matched images are fed back into the annotated database to improve/complete the visual models of an object – this allows for accurate recognition even when users are viewing objects from different locations, angles, lighting conditions. The details of these modules are described next.

4.1 Object Geometry (Sensory and Visual)

When Alice's camera points at an object, the video frames are uploaded to the OverLay server, which matches the frame against annotated images in its database. Matching against all database objects will be prohibitively slow. For maximized performance, both in terms of accuracy and speed, it is important to prune the matching search space to only the likely candidate objects. GPS location, erroneous to $30m$ or more in indoor environments, prunes the candidate set to the order of a building or a wing in a shopping mall. A real-world deployment could easily present more than 100 annotations in such areas – far more pruning is necessary to attain our accuracy and latency targets.

Towards this goal, we prune across *spatial* and *temporal* dimensions by learning a spatiotemporal relationship among annotated objects. At a high level, this results in a graph of objects where the shape of the graph essentially reflects how humans observe the (angular and temporal) separation between objects, as they walk through them. As a simplistic example, imagine we have three tagged objects A, B, and C in sequence on along a hallway (Figure 8). Unsurprisingly, if we observe A then B, it is quite likely we will soon observe C. Further imagine a fourth object, D, located on the left in a perpendicular hallway as shown in Figure 8. To make a left turn, we would expect a rotation $\approx 90°$ counterclockwise. If instead, a clockwise rotation is observed, the user has likely turned away, and the possibility of observing D is dramatically reduced. As a result, once OverLay recognizes an object X, it is able to infer the user's location in the object graph. Only objects near X are now candidates for the next recognition task, resulting in substantial pruning. Once the next object Y is recognized, OverLay knows that the user is now close to Y and selects objects near Y as the new candidate set. This operation carries on and the user's motion path is tracked through this spatiotemporal object graph. Of course, the first object recognition must rely on computer vision alone and crude GPS location.

We limit our understanding of this natural human trajectory to time and rotation only (note that no form of localization is necessary). The gyroscope is accurate enough to capture a high-fidelity awareness of user rotation. Further, we learn and leverage these trajectories at both *macro* (long-lived with

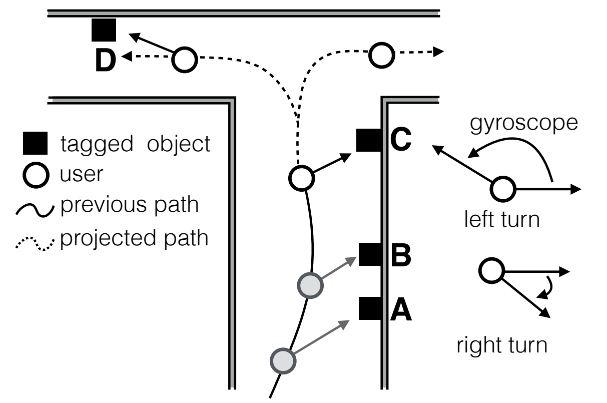


Figure 8: Example macro trajectory. Counterclockwise rotation observed after C is predictive of D.

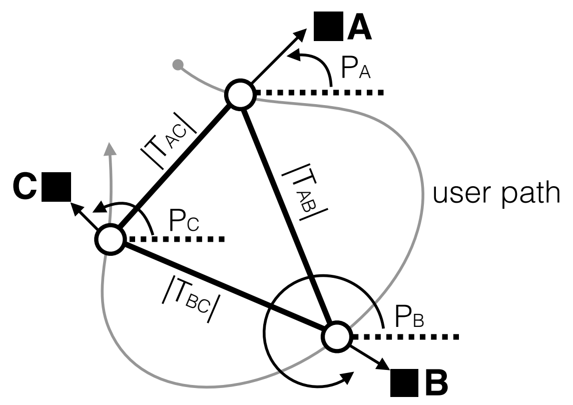


Figure 9: User macro trajectory as it relates to rotational and temporal optimizations. P values reflect rotation of the user as she views multiple tagged objects. T values reflect time elapsed walking.

substantial motion) and *micro* scales (small movements of the device in hand). The details follow.

Learning Macro Trajectories

As the server matches images to its object database, it tracks which objects the user has observed in the recent past (across minutes of walking motion). When observing object i followed by object j, the server records an event k represented by a net rotation R_k (normalized between $0 \leq R < 2\pi$) and by measured time M_k (in arbitrary units). Of course, not all users will view i and j from the same location, and hence R_k will vary. Let E_k^R denote some unknown "error" by which R_k will deviate from the mean rotation from i to j (computed across all users). Similarly, let E_k^M denote some unknown "error" by which M_k will deviate from the mean time taken between the observations of i and j.

With K observation pairs, we may construct a pair of optimizations for N tags, independently solving for rotation and time. In addition to the known values R_k, M_k and unknown values E_k^R, E_k^M, we further introduce unknowns for each annotation i, P_i and $\forall j > i | T_{ij}$. Values of P_i may be understood as the "rotation" of annotation i relative to an arbitrary frame of reference (consistent across all annotations). Thus, $|P_i - P_j|$ may be understood as the the mean observed rotation between annotations i and j. T_{ij} may be understood as the mean time to observe annotation j after observing i.

Values of P_i and T_{ij} do not represent absolute properties of the annotated object. Instead, they reflect *where the object is typically observed*, relative to others – how much rotation is typical from annotation A to B to C, and how much time typically elapses in between – illustrated in Figure 9.
Tables 1 and 2 provide optimizations for rotation and time. To ensure computational tractability in dense areas with many annotations, it was important to formulate each as a linear program (LP). An earlier attempt using a mixed integer linear program (MILP) was unsolvable in days of compute time – these solve in seconds using CPLEX. Given low latency, the server re-solves both LPs for each new annotation, immediately as it is added to the database.

These optimizations explicitly track error attributed to each observation recorded in the annotation database. The solutions to these error terms yield "confidence" estimates for the solved rotation and time estimates, per annotation. For annotation i, median error per observation is computed as:

$$\begin{aligned} E_*^R(i) &:= \forall j,k : \text{median}\{E_k^R | \exists R_k^{i \to j} \vee \exists R_k^{j \to i}\} \\ E_*^M(i) &:= \forall j,k : \text{median}\{E_k^M | \exists M_k^{i \to j} \vee \exists M_k^{j \to i}\} \end{aligned}$$

OverLay combines rotational (P) and temporal (T) values with these error terms (E_*^R, E_*^M) to score which annotations should be prioritized as the best *candidate set* for a user.

Learning Micro Trajectories

Macro trajectories reflect the general, typical behavior of users as they move from annotation to annotation. This is quite useful for users who also move in this typical fashion. For others, the value degrades. For example, if most users walk down the center of a hallway, the angles at which various annotations are observed will reflect observation from this typical trajectory. If a user walks atypically against the wall, the angles she observes will be shifted, and the spatial optimization will become less correct *for her*. Equivalently, a user walking exceptionally quickly will incur faster timings than the typical walking pace. To accommodate these atypical motion patterns, OverLay builds a simple model of *invariant spatial relationships* that hold true even for the atypical user. Simply, these relationships characterize if an annotation A is observed, some annotation B may be known to appear to the *left*, *right*, *above*, or *below* A. Especially as the user makes *micro* (rotational) motions (e.g., scanning around a room looking for annotations), these pairwise spatial relationships enable OverLay to shortlist those annotations the user is most likely to encounter next.

Smartphone cameras have a substantial field of view at their widest setting (zoom is not used in our prototype). In environments with dense annotations, often multiple annotations will be in view simultaneously – detected when a camera frame matches to two or more annotations concurrently. Figure 11 provides pseudocode for inferring *micro trajectories* when two annotations appear simultaneously in view. When a visual match is made we can compute the 2D centroid of the match in the image. Centroids are compared to infer the general direction from one to the other (e.g., right). Figure 10(a) shows the centroid for B appearing right of A (the circle in the figure denoting the camera).

$$\begin{array}{rcl}
\textbf{Minimize} & & \sum_{\forall k \in 1\ldots K} E_k^R \\
\textbf{Subject to} & & \\
\forall R_k^{i\to j} | i < j & : & P_i - P_j \le R_k^{i\to j} + E_k^R \\
 & & P_i - P_j > R_k^{i\to j} - E_k^R \\
\forall R_k^{i\to j} | j < i & : & P_i - P_j \le -R_k^{j\to i} + E_k^R \\
 & & P_i - P_j > -R_k^{j\to i} - E_k^R \\
\forall i \in 1\ldots N & : & 0 \le P_i < 2\pi(N-1) \\
\forall k \in 1\ldots K & : & 0 \le E_k^R < 2\pi \\
\textbf{Solving for} & & \\
\forall i \in 1\ldots N & : & P_i \\
\forall k \in 1\ldots K & : & E_k^R \\
\textbf{With parameters} & & \\
\forall k \in 1\ldots K | \exists i,j & : & 0 \le R_k^{i\to j} < 2\pi
\end{array}$$

Name	Parameter / Variable Interpretation
$R_k^{i\to j}$	Observation k; rotation from anno. i to j
P_i	Rotational position of a anno. i
E_k^R	Rotational error for observation k

Table 1: LP, relative angular (rotational) position.

$$\begin{array}{rcl}
\textbf{Minimize} & & \sum_{\forall k \in 1\ldots K} E_k^M \\
\textbf{Subject to} & & \\
\forall M_k^{i\to j} | i \le j & : & T_{ij} \le M_k^{i\to j} + E_k^M \\
 & & T_{ij} > M_k^{i\to j} - E_k^M \\
\forall M_k^{i\to j} | i > j & : & T_{ji} \le -M_k^{j\to i} + E_k^M \\
 & & T_{ji} > -M_k^{j\to i} - E_k^M \\
\forall i < j \in 1\ldots N & : & 0 \le T_{ij} \\
\forall k \in 1\ldots K & : & 0 \le E_k < \max\{\forall k | M_k\} \\
\textbf{Solving for} & & \\
\forall i < j \in 1\ldots N & : & T_{ij} \\
\forall k \in 1\ldots K & : & E_k^M \\
\textbf{With parameters} & & \\
\forall k \in 1\ldots K | \exists i,j & : & 0 \le M_k^{i\to j}
\end{array}$$

Name	Parameter / Variable Interpretation
$M_k^{i\to j}$	Observation k; time from anno. i to j
T_{ij}	Time separation between anno. i and j
E_k^M	Temporal error for observation k

Table 2: LP, relative temporal spacing.

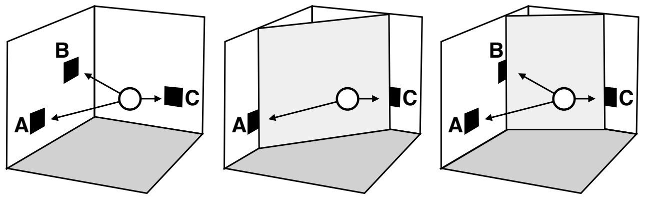


Figure 10: Micro trajectories for annotations A, B, C on walls of a room. (a) $A \leftrightarrow B$ invariant. (b) $A \leftrightarrow C$ *conditionally invariant* – effectively invariant, so long as the observer remains outside the shaded boundary. (c) $B \leftrightarrow C$ also conditionally invariant.

Under certain conditions, *these micro trajectory relationships will hold invariant, regardless of where in the room A and B are observed.* $A \leftrightarrow B$ is *spatially invariant* when A and B are coplanar with a room's wall (e.g., mounted). OverLay assumes $A \leftrightarrow B$ *could* be spatially invariant, if they have been found simultaneously in view by one or more observers. Assume w.l.o.g. A is positioned left of B. At the time of retrieval, once A is observed, we may immediately shortlist B as a potential candidate. If the user rotates right (tracked by gyroscope), we increase our confidence: B should shortly appear in view.

Shortlisting is most productive when A, B are truly spatially invariant, the condition does not have to hold universally to be useful. Often, a *conditional invariance* occurs. That is, a region in which the observer may view A, B such that the condition holds. In Figure 10, $A \leftrightarrow B$ reflects true spatial invariance. $A \leftrightarrow C$ are invariant under the condition that the observer remains outside the illustrated shaded "barrier." Similarly, $B \leftrightarrow C$ are also conditionally invariant.

Prioritizing Candidate Tags

At the *macro* and *micro* scales, OverLay develops an understanding of which annotation a user is most likely to encounter next, given a current observation. OverLay prioritizes candidate annotations according to this understanding before matching against the user's live camera imagery. The prioritization is applied as follows.

Consider Figure 12. OverLay considers time then rotation. First, we construct a spatiotemporal radius M_U around the user, reflecting the set of annotations it would have been *possible* for the user to visit since the last match (illustrated as A) in M_U time. For each annotation i, we expand the radius to $M_U + E_*^M(i)$, to account for the computed temporal uncertainty. Annotation i is accepted to step two only if $T_{Ai} < M_U + E_*^M(i)$, otherwise it is immediately rejected.
Next, we are able to impose an ordering based on rotation. Given a user's rotation R_U since tag A, we find orientation $P_U := (P_A + R_U) \bmod 2\pi$, the user's current orientation. We consider the rotational distance from any tag i as $|P_i - P_U| \bmod 2\pi + E_*^R(i)/2$. We include only half of the error term to reflect the 50-50 probability that the error should work in favor or against accepting tag i.

Since micro trajectory is rotation, we may factor it into the (macro) rotational distance scores. If a micro trajectory relationship {Above, Below, Left, Right} is inferred with annotation i and the user's gyroscope reflects motion in the corresponding direction (i.e., clockwise for right, counterclockwise for left, up for above, and down for below) from i, we subtract a value $w\pi$ from the rotational score. w is a weight which lets us (de)emphasize micro trajectory context.

Regionalized Analysis

When the OverLay application is first launched, the user has no history of recent matches – OverLay has no notion yet of the user's macro/micro trajectory. To limit the search space, OverLay leverages the user's rough GPS location. The user's

```
Input : $t_i$ = Descriptors of image $i$
        $t_j$ = Descriptors of image $j$
Output: Spatial relationships $\in \{A, B, L, R, \text{NONE}\}$
$match_{ij} = descriptorMatch(t_i, t_j)$
if $|match_{ij}| \geq threshold$ then
  $centroid_i = computeCentroid(t_i \in match_{ij})$
  $centroid_j = computeCentroid(t_j \in match_{ij})$
  $W_i = Width_i; H_i = Height_i;$
   /** inferRelation example **/
  if $centroid_i \in [(W_i/3, 2W_i/3), (0, H_i/3)]$ and
  $centroid_j \in [(W_j/3, 2W_j/3), (2H_j/3, H_j)]$ then
     relationship is above-below
  end if
   /** inferRelation example end **/
  return $inferRelation(centroid_i, centroid_j)$
end if
return $none$
```

Figure 11: Pseudocode: micro trajectory inference.

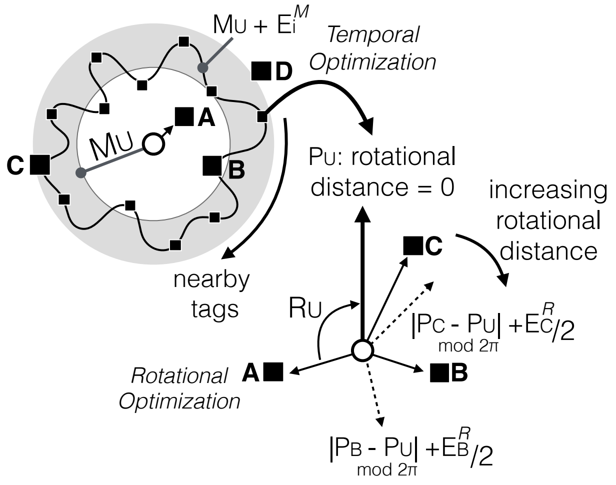
location and the location of every annotation in the database is discretized using the "Geohash" algorithm. OverLay attempts matches only against annotations in the user's square Geohash "cell" or in any of the eight neighboring cells.

4.2 Learning from Retrieval

Initially, an annotated object is only represented by a single image in the database. Intuitively, the crowdsourced effect of many users viewing the same annotated object (over time) might be useful: more views → more visual context → a refined database → higher precision for later users. OverLay exploits this potential for *learning from retrieval*.

Learning from retrieval is a periodic four step process, run as a background server daemon. (1) All past retrievals and the original annotation images are used to reconstruct a 3D model of the object and surroundings (a process known as *structure from motion*, or specifically, *bundle adjustment*). The 3D model provides a geometric representation of both the object and each *camera pose*, the 3D position and orientation of the user during retrieval. (2) Due to false positive matches ($\approx$ 5% in our tests), 3D reconstruction [10] can yield erroneous camera poses. Fortunately, these retrievals are outliers in the resultant 3D *point cloud*. We remove these false camera poses by performing outlier removal based on the point cloud centroid and variance of the model points. (3) All remaining retrievals contain the same object, but often contain redundant visual information. Including all retrieval imagery in the annotation database would create bloat and increase matching latency. Instead, OverLay identifies the most diverse retrievals (i.e., angles from which the object appears sufficiently different). For this, we use KMEANS clustering on 3D camera pose angles to cluster similar retrieval poses together. (4) We select one representative view from each of the retrieval clusters to construct a secondary matching database for each annotation, to be invoked only when primary matching with the original fails.

Since learning from retrieval creates a secondary database, it imposes zero latency for image frames which match the primary database. When the primary database is insufficient, the secondary database provides valuable matches. The overall experience is thus improved as users who would otherwise



MU + E$_i^M$
Temporal Optimization
D
A
MU
C
B
PU: rotational distance = 0
increasing rotational distance
nearby tags
RU
C
|PC - PU| mod 2π +E$_C^R$/2
Rotational Optimization
A
B
|PB - PU| mod 2π +E$_B^R$/2

Figure 12: Prioritizing the annotation candidate set. We compare the user's last matched tag A with all other tags. Elapsed time T_U defines a radius (containing B), expanded by error term $E_*^M(i)$ for each annotation (containing C, D excluded.) B, C, and others in this expanded radius proceed to step 2 (rest rejected). They are prioritized by rotational deviance from the user, including error terms $E_*^R(i)$.

see no annotation content are presented with results from the secondary database, albeit at a slight delay.

4.3 Real-time Cloud Computer Vision

A primary requirement for Mobile AR is that annotations must quickly appear as the user scans around a room. This section describes necessary engineering refinements to ensure that OverLay can surpass a tight usability bound.

Sub-selecting Frame Regions from Gyroscope

Often the user will hold the device reasonably steady while reading on-screen annotations. Frames generated during this time are near duplicate of the previous frames and need not be uploaded. We leverage accelerometer and gyroscope to cheaply identify such frames. No further processing steps are invoked; simply the previously-identified annotation remains on screen.

Even when the user makes nontrivial movements, there is still often partial overlap between the prior and current frames. This overlap can be inferred by (1) monitoring accelerometer and gyroscope to deduce that a user's movements are primarily rotational (and not from walking); and (2) inferring the angular difference from gyroscope to see which portions of the view are new. The gyroscope measures the relative three-axis angular movement of device from a previous position (a point in time), and infers the percentage of new content that should have appeared in the field of view. Specifically, for a lens projecting a rectilinear image, angle of view may be computed in radians as $\alpha = 2\arctan(d/2f)$ where d is the sensor size and f is the focal length. Assume w.l.o.g. that the gyroscope records a clockwise rotation $0 < \beta < \alpha$ radians. Only $100\beta/\alpha$% of the screen contains new content, and only that portion should be shipped to the server for analysis.

Managing Frame Motion Blur

User hand movements from "scanning" the physical environment result in blurred camera frames (motion blur and active adjustment from the autofocus system), causing computer vision underperformance. By applying a Canny edge detector for blur detection, we score frames to select only those most likely to contain usable imagery. As Canny is relatively robust to image resolution, we can apply it at low computational latency to a down-sampled preview image – the first step of the algorithm is to apply Gaussian blur to remove image speckle, roughly equivalent to resolution downsampling. By throwing away useless frames without further processing, we achieve a higher effective frame rate of useful image data, both improving accuracy and latency.

Extracting, Matching, and Refining Image Features

Once a crisp frame (or frame region) has been uploaded to the server and a prioritized candidate set of annotated images has been identified, we must then match the frame imagery to these annotations. Each annotation is represented in the database as a collection of local image features, computed using the SURF [14] feature extractor and descriptor. Each feature is represented as a keypoint, an x, y position in the image at which the feature has been detected and a descriptor vector of 64 floating point values. Two features are considered similar as an inverse function of the Euclidean distance (l^2-norm) between their respective descriptor vectors.

We find M_{AB}, the minimum distance bipartite matching between feature vectors for a pair of images I_A and I_B. Each descriptor value d_i^A of I_A may be compared with each descriptor value d_j^B of I_B. Let m_i^{AB} be the feature descriptor of I_B which has the lowest Euclidean distance from feature i in I_A. Formally, $m_i^{AB} := \forall d_j^B$, argmin $l^2(d_i^A, d_j^B)$. Let $M_{AB} := \{\forall i, m_i^{AB}\}$, the set of all such best matches from image I_A into image I_B.

Although optimal, M_{AB} likely has many false positive, poor image descriptor matches. We may now refine to $M_{AB}^* \subseteq M_{AB}$, the subset of high quality matches from I_A into image I_B. From common practice, we apply the following tests to construct M_{AB}^* from M_{AB}:

Distance Ratio Ratio of the Euclidean distance of the best match value (m_i^{AB}) to the distance of the second best match value. Typically, threshold $\geq$ 0.8.

Cross Check Ensures the inverse match $I_B \rightarrow I_A$ would result in the same pairs; $m_i^{AB} = j \Leftrightarrow m_j^{BA} = i$. Thus, $M_{AB}^* = M_{BA}^*$.

Homography Runs RANSAC (random sample consensus) to find an approximate projective transformation from I_A to I_B. A match m_i^{AB} is rejected if it is an outlier to this transformation.

Early testing revealed an excess number of false positive or negative matches, depending on the thresholds given to homography. So, we add a final binary test for the entire image.

Slope Variance Imagine I_A and I_B composited into a single image, I_A on left and I_B on right. For each match $m_i^{AB} = j$, we draw a line from I_A to I_B between the corresponding feature keypoints. We compute the slope for each line. If the variance of slope is low, all remaining values m_i^{AB} are accepted. Otherwise we reject all matches and set $M_{AB}^* := \emptyset$.

OverLay performs all computer vision (SURF feature extraction, feature matching, and feature filtering) on GPU.

Asynchronous GPU Pipeline

To exact the value of mobile-to-cloud computer vision computational offloading, we must leverage the extreme parallelism available on a modern GPU. The cost of that GPU is only acceptable if we can amortize across multiple concurrent users. However, we found that available implementations of feature extraction and matching assume a single thread of execution (i.e., a single CPU core controlling the entire GPU for each synchronous operation).

When multiple CPU threads attempt concurrent access, conflicts arise. While it is possible to use a CPU semaphore to isolate access to the GPU, this approach leads to intolerably poor GPU utilization. Instead, we heavily modify the CUDA (NVIDIA architecture GPU programming language) implementations of SURF, feature matching, and feature filtering to utilize CUDA Streams. CUDA Streams provides an asynchronous GPU processing pipeline based on a series of micro operations (e.g., copying data into or out of the GPU memory, memory allocation, and kernel execution).

5. EVALUATION

From the onset, we were keen on a real-time evaluation of OverLay with a live fully-functional prototype in the hands of unbiased volunteers. However, we did not want to burden these volunteers with tasks of identifying correct versus incorrect annotations, or other feedback on ground-truth. This could slow them down or distract them from the seamless (environment scanning) experience that OverLay should offer. Thus, for purposes of evaluation, we added instrumentation to the server side of our system. All uploaded imagery and sensory data were recorded to disk. Later we were able to replay the image and sensing data, and exactly observe the ground truth – what the users saw in each video frame, their camera pose, their camera motion, the time between viewing, etc. We labeled $\approx 10K$ video frames based on whether the annotations displayed on screen were correct or not. Note that in some frames, one or many annotations should appear (if multiple objects are present). In others, "NO ANNOTATION" is expected. Importantly, this form of "offline" control allowed us to improve the system even after the live user studies, and test the results through accurate data playback and emulation.

5.1 Experiment Methodology

We annotated ≈ 100 objects on the 2^{nd} floor of the coordinated science lab (CSL) – the floor is around $50m$ long and $10m$ wide, as shown in Figure 13. Objects are annotated just once, and selected arbitrarily – they include printers, exit signs, posters, research prototypes, water fountains, objects on display, etc. Now, with a completely functional prototype, we invited volunteers[2] to use it live. With limited guidance, volunteers were free to explore the building, scan whatever interested them, and they would see the matching annotations live on their phone screen – all without our interference. In this way, their walking and scanning with the device reflects their natural behavior. If they found the experience unenjoyable, they were free to give up at any point. Thankfully, our volunteers were universally curious and excited by

[2]In compliance with our institutions' policies for IRB.

the experience, each spending far longer exploring the space than we asked or anticipated (table below)

Number of tagged objects	100
Volunteers, Men / Woman / Total	6 / 3 / 9
Images captured, Min / Mean / Max	700 / 1188 / 1800
Volunteer time, Min / Mean / Max	12 / 18 / 27 min

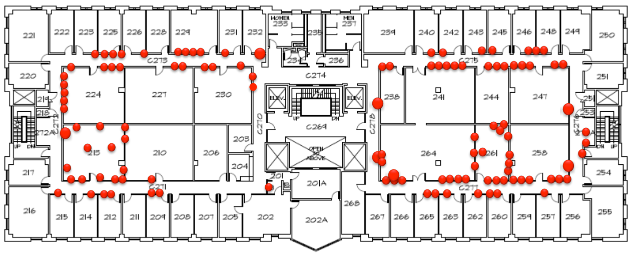


Figure 13: 2^{nd} floor CSL: dots denote annotations.

Our volunteers were only asked to use a single version of our system, CONSERVATIVE. The CONSERVATIVE version includes our complete solution except for optimizations to prune the search space of candidate annotations. From the CONSERVATIVE test data, we emulate (offline) results for our ROTATION and TIME macro trajectory optimizations as well as with micro Trajectory identification. In all subsequent versions, the processed client imagery and sensory data are identical (e.g., gyroscope-based frame sub-selection is used in all). As CONSERVATIVE performs worse in terms of latency, the end user experience can be assumed to be the lower bound of what OverLay can achieve.

5.2 Metrics

Across all volunteers, we evaluate accuracy and latency *per frame* and *per annotation*. Each frame uploaded from the device app to the server is considered in isolation. It may be blurred or crisp; it may capture an annotated object or not; it may be captured intentionally (while the user is actively looking at an object) or incidentally (as the user moves about the space). Similarly, each annotation has unique performance characteristics – it may be easy or hard to identify; the search space optimizations may characterize its location well or poorly. Therefore, each graph presents a CDF (empirical distribution) of accuracy or latency with 100 points, one for each annotated object, generated from $\approx 10K$ samples, one for each frame.

Let Y be the set of all processed frames. Let $V \subset Y$ be the set of frames containing annotated object k. Let $P \subset Y$ be the set of frames identified by our system as having k. Then, $V \setminus P$ denotes our system's false negative predictions, $P \setminus V$ denotes false positive predictions, and $Y \setminus V$ denotes the set of frames which do not contain object k. *We evaluate OverLay's prediction efficacy by the following standard metrics of information retrieval:*

Precision$_k$ $= |V \cap P|/|P|$
: i.e., among all frames that OverLay believes has object k, what fraction truly has k.

Recall$_k$ $|V \cap P|/|V|$
: i.e., among all frames that truly have object k, what fraction was identified by OverLay.

Fallout$_k$ $|(Y \setminus V) \cap P|/|Y \setminus V|$
: i.e., among all frames that truly do not have object k, what fraction was believed to have k.

5.3 Comparison with Approximate Matching

Our image-to-database matching process employs brute force (on GPU) to find the optimal match. As an alternative, a number of heuristics exist for approximate matching [30]. By design, these approximate schemes are computationally inexpensive (i.e., runs faster), at the cost of greater error. To understand this tradeoff, we compare all proposed schemes against APPROXIMATE. Briefly, APPROXIMATE matching computes a set of FLANN indices and loads them into CPU memory – a one-time operation. Later, each incoming image descriptors are matched in real time against these indices (using schemes like KDTREE, KMEANS). The best matched descriptor is up-voted, ultimately resulting in a best match image. To be favorable to APPROXIMATE, we pick the $top-5$ images, and perform a brute-force search on it – as long as the correct candidate is in this set of 5, brute force should output the correct match. This reflects an optimistic view of APPROXIMATE, and we will plot its results alongside ours.

5.4 Overall Results: Accuracy and Latency

Figure 14 presents CDFs of overall accuracy as (a) precision, (b) recall, and (c) fallout. Graphs compare the accuracy of our CONSERVATIVE scheme against optimizations ROTATION, TIME, and FULL, as well as APPROXIMATE. Figure 15 presents a CDF of end-to-end latency (system responsiveness). Each curve reflects 100% of volunteer data, every frame captured by the system. None of the volunteers had any prior exposure to the system nor were given any special knowledge of its technical approach.

We would expect, and confirm here, that the strongest accuracy performance is achieved by the CONSERVATIVE version of our system. Precision is consistently high indicating that when an annotation is shown, it is almost always accurate. ROTATION, TIME, and FULL reflect optimizations aimed to improve system responsiveness (not accuracy). ROTATION is found to be comparatively unreliable in isolation. TIME is highly accurate, and to our surprise, often outperforms even the FULL version of the system (which jointly optimizes across time and rotation, along with micro trajectory context). Upon deeper inspection of the data (the $\approx 10K$ camera frames recording by our volunteers), we find our volunteers were more consistent in time taken to move between objects than they were for angular separation. All these schemes consistently outperform APPROXIMATE, implying how latency reduction indeed incurs a strong penalty in vision-only schemes.

Of course, recall is not impressive, implying that some annotations are not captured/displayed in time. We partly expected this since – if a user's hand shakes, if lighting is poor, if the person views the object from a corner, they will all affect recall. This clearly motivates the need to learn from new retrievals made by users over time – Figure 17 will later demonstrate the efficacy of retrieval.

Latency results also align well with intuition (Figure 15). The sensor optimizations (ROTATION, TIME, and FULL) heavily outperform CONSERVATIVE. APPROXIMATE performs slightly worse than sensor optimizations. Since GPU implementation of APPROXIMATE is unavailable, we estimate latency numbers using 3-dimension implementation of KDTREE and scale

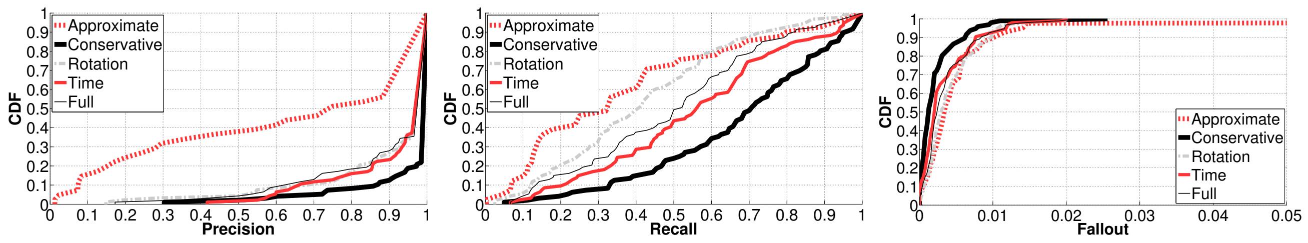


Figure 14: Main accuracy results: (a) Precision; (b) Recall; and (c) Fallout. Graphs compare accuracy of our CONSERVATIVE system against enhancements through optimizations, ROTATION, TIME, and FULL.

accordingly for 64-dimension SURF descriptors. For each of the optimization-based approaches (ROTATION, TIME, and FULL), our budgeted candidate set size is 10 annotations, one tenth of the total 100 annotated objects. As shown in Figure 16, matching is the primary latency component in the CONSERVATIVE system version, reduced here by 90%. Overall speedup from the added optimizations is around $4x$.

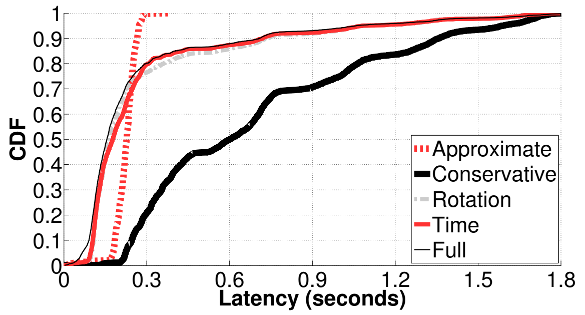


Figure 15: Latency: Optimizations reduce candidate set from 100 to 10, decreasing median end-to-end latency by more than a factor of 4, to 180ms.

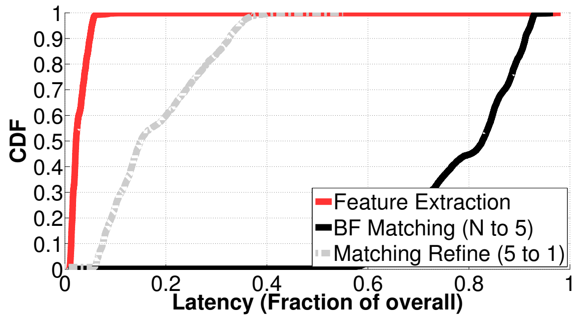


Figure 16: Computer vision latency by component, CONSERVATIVE system version. Optimized versions eliminate 90% of the primary component, matching.

5.5 Learning from Retrievals

OverLay only requires users to annotate objects once, with a single image. While this places minimal burden on an end user, diverse perspectives of the object cannot be captured. Thus, when another user retrieves from a sufficiently different location/angle/lighting condition, the object is sometimes not recognized, diminishing recall (as noted earlier). The issues can get pronounced over time, with minor changes in object's appearance and background. Robust multi-view feature matching is still an unsolved problem in computer vision [37], so its mandatory that features of a stored object are updated periodically. Figure 17 shows the efficacy of such learning from retrievals. Consistent improvements are evident in both precision and recall. Some objects were still unrecognized – delving into the data, we recognized cases of poor network connection (even transient disconnection), heavily delaying the frames from reaching the server. Such cases are entirely out of our control for OverLay.

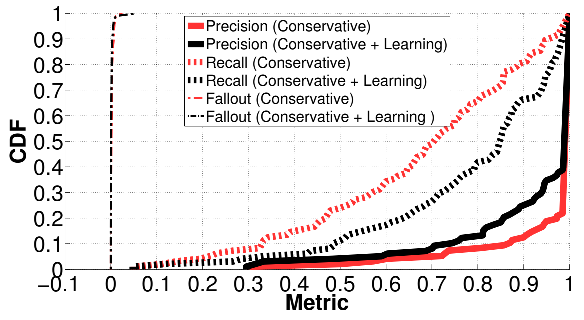


Figure 17: Enhanced accuracy results based on learning from past three retrievals chosen based on diversity in the camera pose in the 3D reconstruction when compared against CONSERVATIVE system.

5.6 Micro Benchmarks

We evaluate some details of OverLay's performance through micro-benchmarks. Figure 18 shows decrease in responsiveness with multiple simultaneous clients on the same server, under (a) the CONSERVATIVE (most GPU intensive) version of the system and (b) the optimized FULL version. The CONSERVATIVE version cannot support more than three concurrent clients; FULL supports six at equal user latency.

Figure 19 shows energy consumed (Samsung Galaxy S4 Android). System power measured using Monsoon Solutions hardware monitor through the battery contacts. Figure 20 shows app UI performance in display frames per second (FPS). Running the complete system, the app maintains 8 FPS (median) on a Samsung Galaxy S4.
Figure 21 shows the percentage of blurred frames rejected without upload to the server, per volunteer. Overall, 46% of frames are rejected without upload, saving substantial upload bandwidth and server processing.

6. LIMITATIONS AND DISCUSSION

We consider several limitations of the OverLay prototype as implemented today, and avenues for future enhancement.

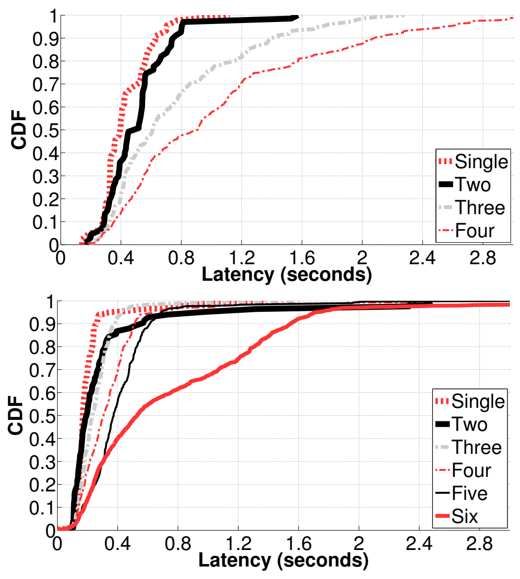


Figure 18: Responsiveness, simultaneous clients, (a) CONSERVATIVE and (b) FULL optimized version.

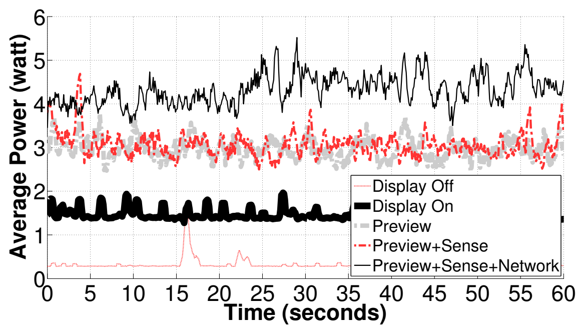


Figure 19: Energy consumption. DISPLAY + PREVIEW (camera enabled) + SENSE (GPS, gyroscope, accelerometer, and compass) + NETWORK TRANSFER reflects the complete system (average ≈ 4.5 watts).

Exact placement of the annotations on screen

OverLay displays object annotations at a fixed screen location. An enhanced user interface might display the annotation directly atop or adjacent to the corresponding object. Our rationale for this this simplification is twofold: (1) when the annotation is authored, the user does not explicitly mark which part of the image corresponds to the object of interest – *multiple objects* might be in view, and (2) during retrieval, the annotation would need to remain aligned to the object, even as the user makes fine hand movements.

The "multiple objects" issue may be addressed in varied ways: (A) requiring the user to draw a rectangle or denote the intended object; (B) marking *candidate objects* on screen and allowing the user to make a simple multiple-choice selection; or (C) assuming the center of the user's screen most likely corresponds to the intended object, and tolerating errors when this assumption does not hold. The "annotation alignment" issue can be roughly accommodated by tracking fine hand movements and shifting the annotations on screen in the opposite direction. Optical flow and gyroscope based techniques are both possible for such motion compensation; recent work has even combined the two for enhanced precision and speed [22].

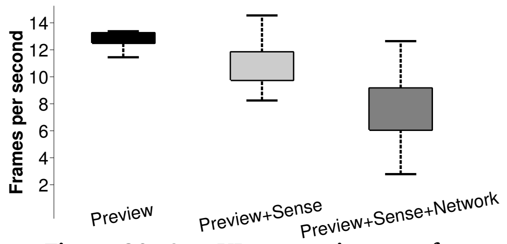


Figure 20: App UI responsiveness, frames/second.

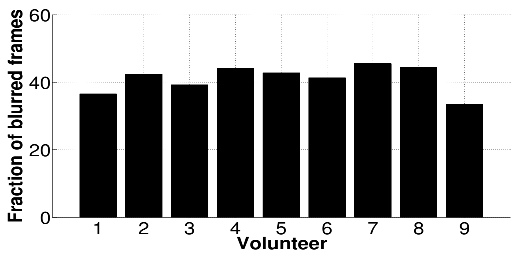


Figure 21: Percentage of frames rejected due to blur.

Searching for annotations

OverLay is most compelling when annotations are dense – when many objects around are annotated. Value decreases with sparsity: the user must "hunt" more between annotated objects. One possibility is to use rough location estimates passively to alert the user when annotations are nearby, for example, through a signature vibration. Once the user opens the app, on-screen arrows might indicate which direction the user should rotate until a nearby annotation appears in view. OverLay is being readied for the *Illinois Distributed Museum* project on the UIUC campus [3], and such "on-screen arrows" will be added in the next release.

Handling appearance changes

OverLay records image data to identify the physical location of an annotation. However, OverLay must cope with environmental dynamism, including changes to the object's appearance. One of our users annotated his office desk, and its appearance changed every day. While we cannot expect to address extreme changes, micro-movement of objects or minor appearance changes could be accommodated through continuous database enhancement, discarding older visual data to be replaced with that from later retrievals. OverLay does not apply to objects such as digital displays — whose appearance changes continuously. Similarly, OverLay does not immediately apply if a retrieval is made from a significantly different angle to that of annotation. However, some of these limitations are addressable using techniques discussed in 5.5.

Distinguishing objects in close proximity

OverLay cannot distinguish visually-similar objects in close proximity. This might be problematic in environments with repetitive design features, such as different name tags on adjacent doors in a corridor. False positives will undoubtably result, although our geometric linear optimizations will partly help (Section 4.1).

7. RELATED WORK

Mobile augmented reality has been studied for more than a decade. [21, 31] outline a broad vision and discuss various possible applications. [42] considers the challenges, strategies, and limitations one needs to overcome in building mobile AR systems. [27] and [16] address two important aspects of Mobile AR – low-power continuous vision and code offloading to cloud. Several works establish sensing-based primitives for Mobile AR: GPS-compass triangulation [17, 19, 32]; camera-pose estimation using Kinect depth sensors [23]; and gyroscope based camera-pose tracking [44]. Some have leveraged physical markers deployed in the environment such as: QR codes [34], color markers [29], and RFID tags [39]. Others have applied 3D camera pose estimation, finding a correspondence between 2D image features and a 3D world coordinate system [25, 40, 45]. [38] is the closest research prototype to our work, using SURF feature extraction and location based pruning to enable mobile AR in outdoor environments. Our primary advantage beyond this effort is in applicability of our search space optimizations to indoor environments, *with zero reliance on location information*.

While prior art has often considered computer vision and mobile sensing in isolation, there has been some recent success in hybridization. Smartphone inertial sensors can be used to imitate or enhance the computation of various vision algorithms (e.g., bundle adjustment, optical flow) [24, 28]. [4, 26] fuse vision and sensing to create hyperlapse video summaries (a form of offline augmented reality). [26] takes a vision-oriented approach, strongly relaxing any latency requirement. [4] priorities lightweight computation, emphasizing gyroscope. [43] proposes energy efficient design of a distributed image search engine. Most of these research are bottom-up and do not deliver an end-to-end application – the challenges of very little training, physical indoor space, and human authoring of content, combined with opportunities of geometric optimizations, makes this paper's design constraints unique.

Object recognition (OR) for mobile devices is a overlapping research effort to Mobile AR, such as Amazon's Fire phone or Google Goggles [1, 2, 15]. Visual MIMO [12], communication between LED screens and a camera, can be used for Mobile AR but the approach is not generalizable. In contrast to mobile AR, OR systems: (1) typically operate on a trained model of multiple images of the same object [1, 2, 18, 20]; (2) are typically invariant to the user's context; and (3) typically do not allow dynamic insertions to the content database as retraining costs are often high. Qualcomm Vuforia [7] is a commercial Mobile AR SDK for object recognition and 3D object tracking. Videoguide [8] is a Vuforia app used to animate architecture work in Barcelona museum. Contrary to Vuforia which requires deployment in advance, OverLay is an anywhere, anytime system for everyone.

8. CONCLUSION

Mobile AR has been an exciting, yet unrealized, vision. This paper attempts to complete the vision within the constraints of today's smartphones. We combine cloud-offloaded computer vision with an optimization framework on space and time – capturing typical human behavior and pruning the computer vision search space. In conjunction with a suite of engineered systems optimizations, these techniques enable a practical system for mobile AR. We demonstrate a response time well within tolerable bounds yet while preserving strong accuracy. Our approach provides a ready-to-use platform for enabling a broad spectrum of compelling mobile AR applications, and is currently deployed in our building and being readied for a campus wide roll-out.

9. ACKNOWLEDGMENTS

We sincerely thank our many volunteers, Professor Peter Druschel our shepherd, as well the anonymous reviewers for their invaluable feedback. We are also grateful to Intel, Google, and NSF for partially funding this research through the grant NSF 1430064.

10. REFERENCES

[1] Amazon fire phone. https://developer.amazon.com/public/solutions/devices/fire-phone.

[2] Google goggles. https://play.google.com/store/apps/details?id=com.google.android.apps.unveil&hl=en.

[3] Illinois distributed museum. http://distributedmuseum.blogspot.com/.

[4] Instagram hyperlapse. http://hyperlapse.instagram.com/.

[5] Mobile augmented reality, video demonstration. http://synrg.csl.illinois.edu/projects/MobileAR/.

[6] Project tango. https://www.google.com/atap/projecttango/.

[7] Qualcomm vuforia. https://www.qualcomm.com/products/vuforia.

[8] Videoguide, antoni gaudi modernist museum in barcelona. http://www.casabatllo.es/en/visit/videoguide.

[9] Wikitude. http://www.wikitude.com/.

[10] S. Agarwal, Y. Furukawa, N. Snavely, I. Simon, B. Curless, S. M. Seitz, and R. Szeliski. Building rome in a day. *Communications of the ACM*, 54(10):105–112, 2011.

[11] C. Arth and D. Schmalstieg. Challenges of large-scale augmented reality on smartphones. *Graz University of Technology, Graz*, pages 1–4, 2011.

[12] A. Ashok, M. Gruteser, N. Mandayam, J. Silva, M. Varga, and K. Dana. Challenge: mobile optical networks through visual mimo. In *MobiCom*, pages 105–112. ACM, 2010.

[13] R. Azuma, Y. Baillot, R. Behringer, S. Feiner, S. Julier, and B. MacIntyre. Recent advances in augmented reality. *Computer Graphics and Applications, IEEE*, 21(6):34–47, 2001.

[14] H. Bay, T. Tuytelaars, and L. Van Gool. Surf: Speeded up robust features. In *ECCV*, pages 404–417. Springer, 2006.

[15] D. M. Chen, S. S. Tsai, R. Vedantham, R. Grzeszczuk, and B. Girod. Streaming mobile augmented reality on mobile phones. In *ISMAR*, pages 181–182. IEEE, 2009.

[16] E. Cuervo, A. Balasubramanian, D.-k. Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl. Maui: making smartphones last longer with code offload. In *MobiSys*, pages 49–62. ACM, 2010.

[17] S. Feiner, B. MacIntyre, T. Höllerer, and A. Webster. A touring machine: Prototyping 3d mobile augmented

reality systems for exploring the urban environment. *Personal Technologies*, 1(4):208–217, 1997.

[18] P. Föckler, T. Zeidler, B. Brombach, E. Bruns, and O. Bimber. Phoneguide: museum guidance supported by on-device object recognition on mobile phones. In *Proceedings of the 4th international conference on Mobile and ubiquitous multimedia*, pages 3–10. ACM, 2005.

[19] S. Gammeter, A. Gassmann, L. Bossard, T. Quack, and L. Van Gool. Server-side object recognition and client-side object tracking for mobile augmented reality. In *CVPR Workshops*, pages 1–8. IEEE, 2010.

[20] B. Girod and C. et al. Mobile visual search. *Signal Processing Magazine, IEEE*, 28(4):61–76, 2011.

[21] A. Henrysson and M. Ollila. Umar: Ubiquitous mobile augmented reality. In *Proceedings of the 3rd international conference on Mobile and ubiquitous multimedia*, pages 41–45. ACM, 2004.

[22] M. Hwangbo, J.-S. Kim, and T. Kanade. Inertial-aided klt feature tracking for a moving camera. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pages 1909–1916. IEEE, 2009.

[23] S. Izadi et al. Kinectfusion: real-time 3d reconstruction and interaction using a moving depth camera. In *UIST*. ACM, 2011.

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

[25] G. Klein and D. Murray. Parallel tracking and mapping on a camera phone. In *ISMAR*, pages 83–86. IEEE, 2009.

[26] J. Kopf, M. F. Cohen, and R. Szeliski. First-person hyper-lapse videos. *TOG*, 33(4):78, 2014.

[27] R. LiKamWa, B. Priyantha, M. Philipose, L. Zhong, and P. Bahl. Energy characterization and optimization of image sensing toward continuous mobile vision. In *MobiSys*, pages 69–82. ACM, 2013.

[28] 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.

[29] P. Mistry and P. Maes. Sixthsense: a wearable gestural interface. In *ACM SIGGRAPH ASIA 2009 Sketches*, page 11. ACM, 2009.

[30] M. Muja and D. G. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. In *VISAPP (1)*, pages 331–340, 2009.

[31] H. E. Pence. Smartphones, smart objects, and augmented reality. *The Reference Librarian*, 52(1-2):136–145, 2010.

[32] W. Piekarski and B. Thomas. Arquake: the outdoor augmented reality gaming system. *Communications of the ACM*, 45(1):36–38, 2002.

[33] C. Qin, X. Bao, R. Roy Choudhury, and S. Nelakuditi. Tagsense: a smartphone-based approach to automatic image tagging. In *MobiSys*, pages 1–14. ACM, 2011.

[34] J. Rekimoto and Y. Ayatsuka. Cybercode: designing augmented reality environments with visual tags. In *Proceedings of DARE 2000 on Designing augmented reality environments*, pages 1–10. ACM, 2000.

[35] J. P. Rolland, L. Davis, and Y. Baillot. A survey of tracking technology for virtual environments. *Fundamentals of wearable computers and augmented reality*, 1:67–112, 2001.

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

[37] F. Schaffalitzky and A. Zisserman. Multi-view matching for unordered image sets, or âĂIJhow do i organize my holiday snaps?âĂİ. In *Computer VisionâĂŤECCV 2002*, pages 414–431. Springer, 2002.

[38] G. Takacs and C. et al. Outdoors augmented reality on mobile phone using loxel-based visual feature organization. In *Proceedings of the 1st ACM international conference on Multimedia information retrieval*, pages 427–434. ACM, 2008.

[39] R. Tenmoku, M. Kanbara, and N. Yokoya. A wearable augmented reality system using positioning infrastructures and a pedometer. In *ISWC*, pages 110–110. IEEE Computer Society, 2003.

[40] D. Wagner, G. Reitmayr, A. Mulloni, T. Drummond, and D. Schmalstieg. Pose tracking from natural features on mobile phones. In *ISMAR*, pages 125–134. IEEE Computer Society, 2008.

[41] D. Wagner and D. Schmalstieg. First steps towards handheld augmented reality. In *ISWC*, pages 127–127. IEEE Computer Society, 2003.

[42] D. Wagner and D. Schmalstieg. Making augmented reality practical on mobile phones, part 1. *Computer Graphics and Applications, IEEE*, 29(3):12–15, 2009.

[43] T. Yan, D. Ganesan, and R. Manmatha. Distributed image search in camera sensor networks. In *SenSys*, pages 155–168. ACM, 2008.

[44] S. You, U. Neumann, and R. Azuma. Hybrid inertial and vision tracking for augmented reality registration. In *Virtual Reality, 1999. Proceedings., IEEE*, pages 260–267. IEEE, 1999.

[45] F. Zhou, H. B.-L. Duh, and M. Billinghurst. Trends in augmented reality tracking, interaction and display: A review of ten years of ismar. In *ISMAR*, pages 193–202. IEEE Computer Society, 2008.