Satellites in Our Pockets: An Object Positioning System using Smartphones

Justin Manweiler
IBM T. J. Watson Research*
Hawthorne, NY, USA
jmanweiler@us.ibm.com

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

Romit Roy Choudhury
Duke University
Durham, NC, USA
romit.rc@duke.edu

ABSTRACT

This paper attempts to solve the following problem: *can a distant object be localized by looking at it through a smartphone*. As an example use-case, while driving on a highway entering New York, we want to look at one of the skyscrapers through the smartphone camera, and compute its GPS location. While the problem would have been far more difficult five years back, the growing number of sensors on smartphones, combined with advances in computer vision, have opened up important opportunities. We harness these opportunities through a system called *Object Positioning System (OPS)* that achieves reasonable localization accuracy. Our core technique uses computer vision to create an approximate 3D structure of the object and camera, and applies mobile phone sensors to scale and rotate the structure to its absolute configuration. Then, by solving (nonlinear) optimizations on the residual (scaling and rotation) error, we ultimately estimate the object's GPS position.

We have developed *OPS* on Android NexusS phones and experimented with localizing 50 objects in the Duke University campus. We believe that OPS shows promising results, enabling a variety of applications. Our ongoing work is focused on coping with large GPS errors, which proves to be the prime limitation of the current prototype.

Categories and Subject Descriptors

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

General Terms

Algorithms, Design, Experimentation, Performance

Keywords

Augmented Reality, Localization, Structure from Motion

*This work was conducted at Duke University.

MobiSys'12, June 25–29, 2012, Low Wood Bay, Lake District, UK.

1. INTRODUCTION

Imagine the following scenario in the future. While leaving for office, Alice needs to ensure that the repairman comes to her home later in the day and fixes the leakage on the roof. Of course, the leak is small and Alice must point out the location of the leak. To this end, she walks across the road in front of her house, points her camera towards the leak, takes a few photos, and types in "`leaking from here`". Later, when the repairman comes to Alice's house, he points his camera towards the roof and scans – when the leak is inside the camera's view-finder, Alice's message pops-up. The repairman repairs the leak and leaves. Alice comes back home in the evening, points her camera towards the leak, and sees the repairman's tag: "`repaired, received payment, thanks!`". Before returning into her house, she cursorily scans the neighborhood with her phone to see if there was anything new. She finds a "`pool party Saturday evening`" tag at the community swimming pool, and another on a tall crane at a nearby construction site, that read "`too noisy: 13 votes`". Alice remembers how she has been frustrated as well, so points her camera at the crane and votes. She looks at the tag again to confirm, which now reads "`too noisy: 14 votes`".

While this may be an intriguing vision of the future, the core idea of tagging objects in the environment, and viewing them through a smartphone's viewfinder, is old. A variety of augmented reality applications have already built such frameworks – Wikitude and Enkin even offer them on the app store [16]. However, these applications implicitly assume that objects in the environment have been annotated out-of-band – that someone visited Google Earth, and entered a tag for the swimming pool. Later, when an Enkin user looks at the same pool through her camera viewfinder, tags of all the objects in her viewfinder pops up. We believe that out-of-band tagging is one of the impediments to augmented reality (AR) becoming mainstream. The ability to tag the environment spontaneously will be vital if users must embrace AR applications in their daily lives.

This project – *Object Positioning Systems (OPS)* – is tasked to address this "missing piece" in today's AR applications. Our ultimate goal is to offer a service that allows a lay user to point her smartphone to any object in the environment and annotate it with comments. While this is the front-end functionality of our system, the key challenge in the back-end pertains to object localization. Our system essentially needs to compute the GPS location of the desired object, and then

trivially associate the user-generated tag to that location. Another user standing at a different location should be able to look at the same object, run our system to compute its location, and retrieve all tags associated to it. Ideally, the system should operate in real time, so the user can immediately view the tag she has created.

While translating this vision to reality warrants a long-term research effort, as a first step, we narrow down its scope as follows. We sidestep indoor environments due to their stringent requirements on object positioning accuracy – a tag for a chair cannot get attached to the table. Therefore, we focus on outdoor objects and assume desktop-type CPU capability (which if unavailable on today's phone, may be available through the cloud). Even under this narrowed scope, the challenges are multiple: (1) State-of-the-art in computer vision is capable of localizing objects from hundreds of pictures of the same object [19, 21]. In the case where a *few* pictures are available – such as those taken by Alice of her rooftop – computer vision becomes inapplicable. Our intuition suggests that sensor information from mobile devices should offer opportunities to compensate for the deficiencies in vision, but the techniques for such information fusion are non-trivial. (2) The smartphone sensors, such as GPS, accelerometer, compass, and gyroscope, are themselves noisy, precluding the ability to pivot the system on some ground truth. Hence, aligning sensor information with vision will become even more difficult, requiring us to formulate and solve a "mismatch minimization" problem. (3) OPS needs to identify the user's intention – different objects within the viewfinder may be at different depths/locations, and only the intended object's location is of interest. (4) Finally, the system needs to be reasonably lightweight in view of the eventual goal of on-phone, real-time operation.

The design of OPS has converged after many rounds of testing and modification. Our current prototype on Android NexusS phones has been used to localize 50 objects within the Duke University campus (e.g., buildings, towers, parking lot, cranes, trees). Performance evaluation shows that the system exhibits promising behavior. In some cases, however, our errors can be large, mainly stemming from excessively-high GPS errors. Nonetheless, OPS is able to identify and communicate such cases to the user – like a confidence metric – allowing them to re-attempt the operation. While not ready for real-world deployment, we believe OPS demonstrates an important first step towards a difficult problem with wide-ranging applications.

The key contributions in OPS are summarized as follows.

1. **Localization for distant objects within view:** We show opportunities in *multimodal sensing* to localize visible objects in outdoor environments, with core techniques rooted in mismatch optimization.
2. **System design and implementation on the Android NexusS platform:** Reasonably lightweight algorithms achieve promising location accuracy, with marked improvements over an optimized triangulation-based approach using GPS and compass.

The rest of the paper expands on these contributions, beginning with motivation and overview in Section 2 and primitives of OPS localization in Section 3. Next, in Section 4, we present the design of OPS. In Section 5, we address additional practical challenges for translating the core design into a complete system. We provide results from our testing experiences in Section 6 and our ongoing work to improve OPS in Section 7. We compare OPS with the state of the art in Section 8. Section 9 concludes with a brief summary.

2. MOTIVATION AND OVERVIEW

This section visits the motivation of the paper, with a generalization of OPS to other applications, and then presenting a functional overview of the system. The subsequent sections elaborate on the core technical challenges and solutions.

2.1 Applications beyond Tagging

An Object Positioning System (OPS) has natural applications in tagging the environment. While this was our initial motivation, we observed that the core capability to localize a distant object is probably a more general primitive. In contemplating on the possibilities, we envisioned a number of other applications that can overlay on OPS:

(1) Location-based queries have been generally interpreted as queries on the user's current location (e.g., "restaurants around me," "driving directions from here to the airport"). However, queries based on a distant object can be entirely natural, such as "how expensive are rooms in that nice hotel far away," or "is that cell tower I can see from my house too close for radiation effects?" While walking or driving up to the object location is one way to resolve the query, the ability to immediately look up the hotel price based on the hotel's location, is naturally easier. OPS could enable such "object-oriented queries."

(2) OPS could potentially be used to improve GPS, particularly where the GPS errors are large or erratic. This is true even though OPS actually depends on GPS. The intuition is that combination of multi-modal information – vision and GPS in this case – can together improve each of the individual dimensions. Thus, knowing the location of the object can help improve the location of the camera.

(3) High-end cars entering the market are embedded with a variety of safety features [13], such as adaptive cruise control, lane change detection, blind spot alerts, etc. Existing cars remain deprived of the capabilities since upgrades may be expensive, even if feasible. High accuracy OPS technologies on mobile smartphones may enable services that approximate these capabilities. Smartphones mounted near the car's windshield could estimate location of other objects in the surroundings, and trigger appropriate reactions.

To summarize, one may view OPS as somewhat analogous to GPS – GPS satellites help receivers estimate self-location, while OPS phones estimate other's-locations. It is this analog that motivates our metaphor – *satellites in our pockets*.

2.2 System Overview

We present an overview of OPS with the goal of introducing the functional components in the system, and their interactions. We expect it to help the transition to technical details.

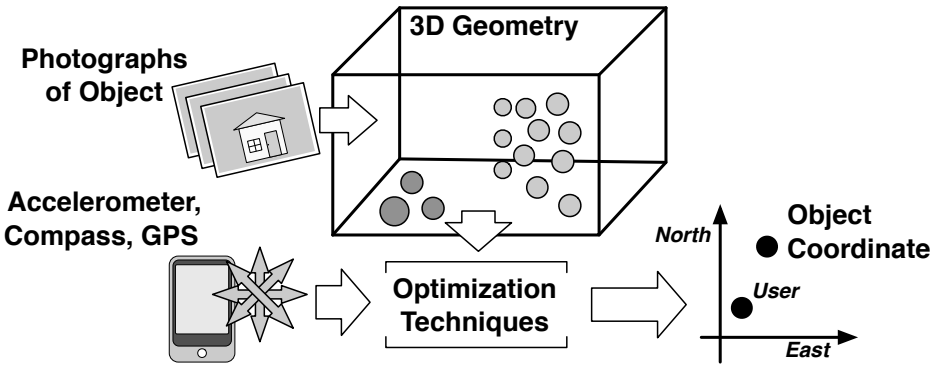

Figure 1: An architectural overview of the OPS system – inputs from computer vision combined with multi-modal sensor readings from the smartphone yield the object location.

When a user activates OPS on her smartphone, the camera is automatically turned on, along with the GPS, accelerometer, compass, and gyroscope. The user is expected to bring the object of interest near the center of her viewfinder, and take a few pictures from different positions. These positions can be separated by a few steps from each other in any direction – the goal is to get multiple views/angles of the same object. As few as 4 photos are adequate, however, more the better. Once completed, OPS displays the object's GPS coordinate.

While this is a simple front-end, Figure 1 shows the flow of operations at the back-end. The pictures taken by the user are accepted as inputs to the computer vision module, which implements a technique called *structure from motion* (SfM) [9]. Briefly, SfM is the process of extracting a 3D structure of an object from diverse views of a moving camera. As a part of this process, SfM first identifies *keypoints* in each picture – keypoints may be viewed as a set of points that together capture the defining aspects of the picture. The keypoints are matched across all the other pictures, and those that match offer insights into how the camera moved (or its angle changed) while the user clicked the different pictures. The final output of this process is a 3D structure, composed of the object and the camera locations.

Importantly, the 3D structure – also called the *point cloud* – is not in absolute scale. Rather, the point cloud offers information about the *relative* camera positions, as well as the *relative* distances between the cameras and the object. To be able to obtain the GPS location of the object, the point cloud needs to be "grounded" on the physical coordinate system. In an ideal scenario, where the point cloud and the GPS locations are both precise, it would be easy to scale the relative camera locations to match the GPS points. This will scale the object-distance as well, eventually yielding the absolute object location. Unfortunately, errors in the point cloud, and particularly in GPS readings, introduce a mismatch. Therefore, OPS uses the configuration of the camera-locations in the point cloud to first adjust the GPS positions. To this end, OPS formulates and solves an optimization problem to minimize the total adjustments.

The next goal is to use the compass readings from these (corrected) locations to triangulate the object of interest. Again, if all compass readings were accurate, any pairwise triangulation from the GPS points should yield the same object location. Unsurprisingly, compasses are noisy as well – therefore OPS executes another optimization that minimizes the total adjustments on all compasses, under the constraint that all triangulations result in the same object location. This corrects the compass readings, and also offers a rough estimate of the object's distance from the GPS locations. By applying the compass readings back on the 3D point cloud, and again solving an optimization problem (detailed later), OPS finally converges on the object location.

OPS also extracts the height of the object, by incorporating the *angular pitch* of the phone while taking the picture. Thus, the final output is a location in 3D space, represented as a GPS coordinate and a height above the ground. The following sections zoom into the details of each of these components, beginning with the primitives of object localization.

3. PRIMITIVES FOR OBJECT LOCALIZATION

Inferences of a distant location from a known point-of-origin is an old problem. Historically, the principles of *triangulation* date to Greek philosophers of the 6th Century BC. Land surveying applies the same basic techniques today at great precision. The related technique of *trilateration* (location determination through known distances, rather than angles) is the technical basis of GPS. As a starting point, we investigate the applicability of these techniques to object localization.

Why not use GPS/compass to triangulate?
Smartphones have embedded GPS and compass (magnetometer) sensors. The precise location and compass bearings from any two points determines a pair of lines[1]. The object-of-interest should fall at their unique intersection. We illustrate compass-based triangulation in Figure 2.

In principle, if a user points her phone at an object-of-interest from two distinct locations, we should be able to easily infer the object's location. Of course, to obtain these distinct locations, we cannot ask the user to walk too far, or using the system would be impractical. Instead, we can imagine the user walking just a few steps to infer the location of the object, say 40 meters away. When the scenario is

[1]The two points must not be collinear to the remote location.

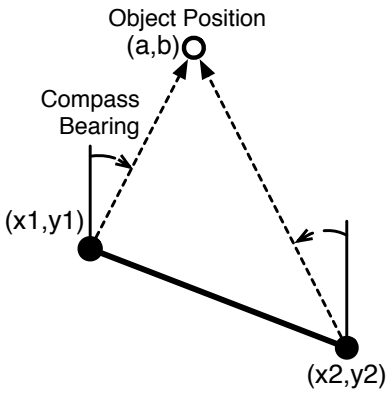

Figure 2: Compass-based triangulation from GPS locations (x_1, y_1), (x_2, y_2) to object position (a, b).

such (i.e., distance between camera views is much smaller than the distance from the camera to the object), compass precision becomes crucial. A few degrees of compass error can dramatically reduce the accuracy of triangulation. Similarly, if the distance between the camera views are slightly erroneous, the result can also be error-prone. Smartphone sensors are not nearly designed to support such a level of precision. GPS can be impacted by weather (due to atmospheric delay), clock errors, errors in estimated satellite ephemeris, multipath, and internal noise sources from receiver hardware. Compass magnetometer readings are imprecise and subject to bias, due to variation in the Earth's magnetic field and nearby ferromagnetic material. Triangulation, at least under these extreme conditions, does not apply immediately.

Can smartphones apply trilateration?
Trilateration requires estimating the distance to the object-of-interest (range), from two vantage points. GPS is a popular implementation of trilateration – the distances from multiple satellites are computed from the propagation delay of the corresponding signals. Unfortunately, a GPS-like scheme is inapplicable for object positioning, since the objects are not collocated with a wireless radio. The phone camera, however, may partially emulate this functionality without any requirement of infrastructure at the object. This can naturally be an appealing alternative.

So long as the object-of-interest remains clearly in the camera view, the size of an object in the picture is a function of the camera's distance to that picture. The size can be estimated by the *visual angle* needed for that object (Figure 3), which can be computed as $v = 2\arctan(s/d)$, where v is the visual angle, s is the size (or height) of the object, and d is the distance to the object. Since we do not know object size s, we cannot compute d. However, knowing two different visual angles from two distinct locations, it is possible to eliminate s and obtain a ratio of the distances to the object from these locations. Let σ denote this ratio; then σ can be computed as

$$\sigma := \frac{d'}{d} = \frac{\tan(v/2)}{\tan(v'/2)}$$

Thus, although visual trilateration cannot precisely localize the object, the value of σ can certainly offer hints about the

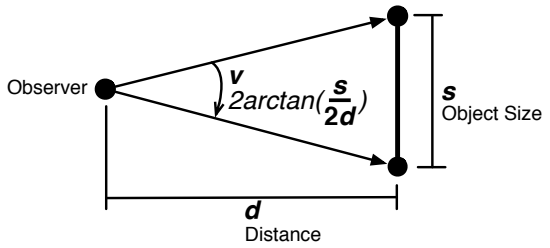

Figure 3: The visual angle v relates the apparent size s of an object to distance d from the observer.

object's position. If one plots all points in space that are away from two camera locations in the ratio of σ, one gets a curve as shown in Figure 4. The object will naturally lie at some location on this curve.

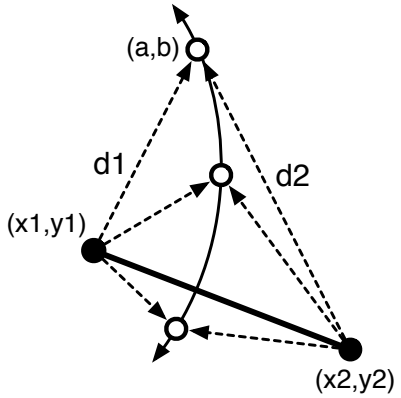

Figure 4: Visual Trilateration: unknown distances from GPS locations (x_1, y_1) and (x_2, y_2) to object position (a, b) are in a fixed ratio d_2/d_1.

Can phone cameras also triangulate?
Land surveying systems typically use optical sensing for precise triangulation. Possibly, the camera could be exploited to improve the accuracy of compass-based triangulation as well. Multiple views of an object from different angles, even if only slightly different, produce visual distortions, due to the phenomenon of *parallax*. Points in the foreground appear to change in relative position to points in the background. The properties of parallax, and visual perception in general, are well-understood. For example, stereo vision leverages parallax effects to invoke a three-dimensional perception from two-dimensional images. Thus, with a careful analysis of images taken from multiple nearby locations, it should be possible to invert these effects. In particular, it would be possible to infer the interior angle between a pair of GPS locations and the object position. However, knowing the interior angle is again not adequate to pinpoint the object location – instead it offers a curve and the object can be at any location on this cure. Figure 5 captures this efficacy of visual triangulation.

Combining Triangulation and Trilateration
While neither triangulation nor trilateration can pinpoint object location, observe that computing the intersection of the two curves (in Figure 4 and Figure 5) yields a small number of intersection points. Moreover, if compass triangulation is added, there is more than adequate information to

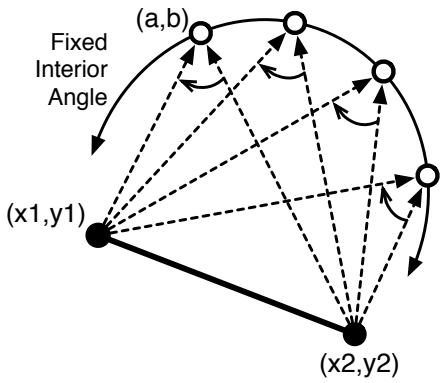

Figure 5: Visual Triangulation: fixed interior angle from known GPS location (x_1, y_1) to unknown object position (a, b) to known GPS position (x_2, y_2) .

uniquely identify the object position (a, b). Figure 6 shows the superimposition of all four curves – observe that this is an over-constrained system, meaning that there is more than sufficient information to compute an unique solution.

This excess of information will later form the basis for noise correction on imperfect sensors. This is necessary because, with errors from GPS, compass, and inaccurate parameter estimation from the visual dimensions, we do not obtain a single point of intersection across all curves. While increasing the number of camera views will help, it will also increase the number of curves (each with some error). Thus, ultimately, we are left with many points of intersection, many of which can be far away from the true object position. To find a single point of convergence, we will rely on optimization techniques, finding the most-likely true object point by minimizing estimates of sensor error.

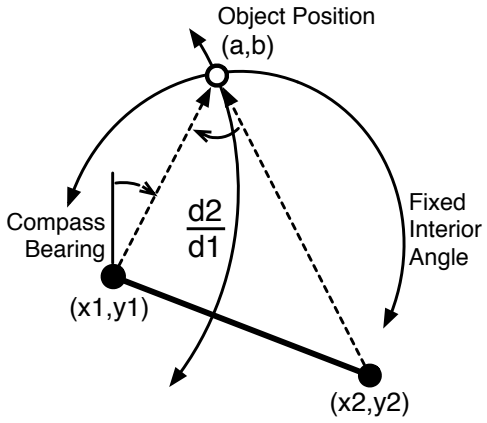

Figure 6: Intersection of the four triangulation curves for known points $(0, 0)$ and $(10, -4)$, localized point $(4, 8)$, distance ratio $\sigma = 6\sqrt{(5)}/4\sqrt{(5)} = 1.5$, and internal angle $\gamma = 2 \cdot \arctan(1/2) \approx 53°$.

Next, we describe the OPS system design, focusing mainly on how advanced computer vision techniques can be applied to implement visual trilateration and triangulation. In particular, vision will quantify relative distance and invert the effects of parallax to find the interior angle between a pair of photographs, which will guide the other sensors to ultimately yield object location.

4. OPS: SYSTEM DESIGN

In an ideal world, visual information should not be necessary – noise-free sensors should be able to triangulate the object position. Since real-world sensors are noisy, OPS uses visual information to combat the impact. However, visual information relies partly on sensors, and thus, the overall system needs to be optimized jointly, to marginalize the noise. For ease of explanation, we first describe the individual techniques in isolation (i.e., without considering the effect of noise). Then, we explain how noise forced many of our designs to fail, motivating our ultimate methods of "mismatch optimization." Figure 7 captures this flow of operations.

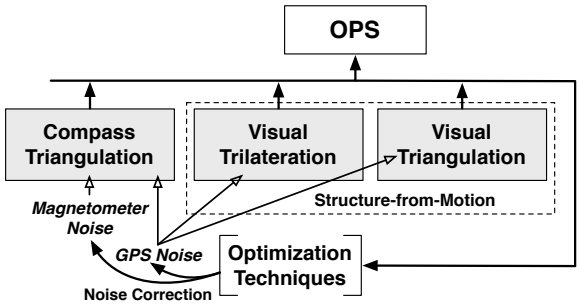

Figure 7: OPS builds on triangulation and trilateration, each underpinned by computer vision techniques, and multi-modal sensor information. Sensor noise affects the different techniques, and makes merging difficult.

4.1 Extracting a Visual Model

We begin with a discussion of the kind of information deducible from multiple photographs of the same object. Figure 8 shows how two observers each experience a different *perspective transformation* of the same object – a building. Note that the apparent *size* of the same building is different in each, as is the *shape*. The differences in scale[2] can be used to determine relative distances to the building. The shape of the transformation can reveal something about the difference in angle from each view to the building. As evident from our discussion of visual trilateration and triangulation, relative distances and angles from the cameras to the object can be valuable for estimating its location. OPS relies on a state-of-the-art computer vision technique, called *Structure from Motion* (SfM), to extract these distances and angles [9, 17]. As we will see next, SfM derives a 3D model of objects in the visible environment, including the object-of-interest and the camera, and computes these relations from them.

Structure from Motion

The SfM module functions as a multi-step process as follows. First, SfM accepts multiple photos from the user, and on each photo, runs an algorithm known as a *feature detector* [1, 11, 15]. The algorithm identifies various "interesting" points of the photo, called *keypoints*. Ideally, the keypoints

[2]After accounting for differences in camera focal length and lens distortions.

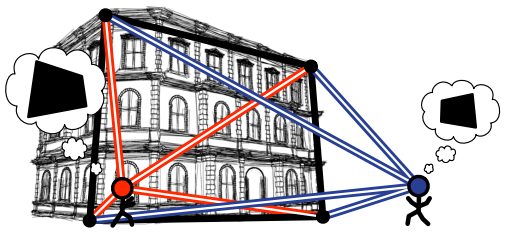

Figure 8: Two vantage points of the same object of interest. The "thought-bubbles" show the two different perspective transformations, each observing the same four feature corner points.

should be those that are likely to be robust (stable) across different perspectives of the same object. Put differently, a robust keypoint is one that consistently reflects the same physical point on the object. For example, the peak of a pitched roof may be taken as a keypoint in each image of the same house. Of course, the keypoint detection algorithm is a heuristic, and is prone to inconsistencies across multiple photos. However, with a large number of keypoints per image, there is likely to be a substantial number of keypoints that derive from the same physical point in all photos.

The keypoints from each photo are updated to a server, which then executes a keypoint matching algorithm. The matching process entails comparison of *feature descriptors* associated with each keypoint. A feature descriptor can be thought of as a unique "fingerprint" of a photograph, taken from the pixels around the keypoint. If a pair of feature descriptors are a strong match (numerically), the corresponding pair of keypoints can be assumed to likely capture the same physical point in the real world. Once keypoints are linked across multiple photos, SfM now prepares to analyze the spatial relationship between the locations at which the photographs were taken.

For spatial reasoning, SfM applies algorithms that bear similarity to stereo vision. Perspective differences from multiple views of the same object (arising from parallax) can be used to reconstruct depth information. However, unlike stereo vision, SfM does not require a (known) fixed distance and relative orientation between a pair of views. Instead, SfM takes multiple sets of matched keypoints and attempts to reconstruct (1) a sparse 3D *point cloud* of the geometry captured by those keypoints, and (2) the relative positions and orientation of the camera when the original photographs were taken, known as *pose*. Figure 9 shows an example point-cloud for a building – observe that the points in the cloud are located on the surface of the building and other visible objects in the environment, as well as at the location of the camera.

SfM relies on *Bundle Adjustment* to perform a simultaneous refinement on the estimated 3D point cloud and parameters for each camera view (including, camera pose and lens distortions). Popular implementations of Bundle Adjustment use the Levenberg-Marquardt algorithm to perform a numerical nonlinear optimization on *reprojection error* between what is seen in each image (as described by the keypoints) and what is predicted by different parameterizations of camera pose

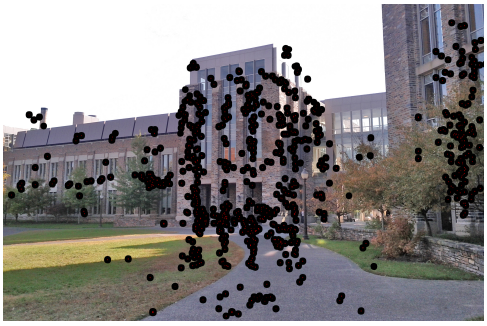

Figure 9: Example of a 3D point cloud overlaid on one of the images from which it was created.

and 3D geometry. Thus, in summary, the final output from SfM is a considerably-accurate 3D point cloud.

From 3D Point-Cloud to Physical Location

For OPS, we utilize SfM as a "black box" utility. As input, SfM takes the matched keypoints of the user's images. As output OPS receives a 3D *point cloud* of estimated <X, Y, Z> coordinates for each keypoint that was successfully matched across a sufficient number of images. We also have estimated <X, Y, Z> camera pose coordinates from where each photo was originally taken. This coordinate system, however, even if completely precise, exists at an unknown relative scaling, translation, roll, pitch, and tilt from the corresponding locations in the real-world. To compute the physical location of the object, the camera locations and orientations in the 3D model needs to be "aligned" with the GPS and compass readings from the smartphone. However, since the GPS/compass readings themselves will be noisy, this alignment will be non-trivial – the GPS/compass values will need to be adjusted to minimize the mismatch. Moving forward, OPS will focus on addressing these challenges.

4.2 Questions

Before we continue further into the challenges of mismatch minimization, we briefly discuss a few natural issues related to the system build-up.

(1) Capturing User Intent

The use of computer vision entails a second practical challenge – many objects may appear in the camera view. OPS must be able to infer, automatically, which object in view the user is most-likely interested in localizing. For example, a building may be partially occluded by trees. Thus, the point cloud may contain many keypoints that are not reflective of the specific object-of-interest. In general, we assume that the user positions the object-of-interest roughly at the center of the camera's viewfinder. Across multiple photographs, the intended object will become a "visual pivot." Near-foreground and distant-background points appear to shift away from this central point, due to parallax. More sophisticated techniques based on the computer vision techniques of *segmentation* are also relevant here. For example, in Section 5, we will consider an alternative approach for cases where we can assume the user is focused on a building. In our evaluation, however, we will avoid such assumptions.

(2) Privacy

We note that while OPS may offload computational tasks to a central server/cloud, users need not ever upload actual photographs of objects-of-interest (in fact, they can be discarded from the smartphone as well). Instead, they can only upload the keypoints and feature descriptors, that contain all the information needed by SfM. This serves to address any privacy concerns that a user may have with OPS.

4.3 Point Cloud to Location: Failed Attempts

In our original design, we expected that once a Structure-from-Motion point cloud is extracted, estimation of the real-world coordinate of the object would be reasonably straightforward. This would only require mapping vision <X, Y, Z> coordinates to real-world <latitude, longitude, altitude>. In practice, substantial sensor noise makes this mapping more difficult and error-prone than we had anticipated.

The point cloud reflects the structure of the object location relative to the user's locations, when the original photographs were taken, but at an unknown relative scaling, translation, roll, pitch, and tilt from the real-world. Importantly, the locations at which the photographs have been taken are known in both real-world coordinates (through GPS) as well as in the SfM-derived coordinate system. In principle, some *affine transformation* should exist to convert from one coordinate system to the other. We sought to apply state-of-the-art computer vision optimization techniques to estimate this affine transformation, and we describe three of our failed attempts, followed by the actual proposal.

Attempts using Point Cloud Registration

We applied the computer vision technique of Iterative Closest Point (ICP) to find the mapping. ICP is commonly used in the *registration* of one point cloud to another. Before applying ICP, we first eliminated what is typically a key challenge; we pre-defined the association of points from one point cloud to the other. We also eliminated as many degrees-of-freedom in the transformation as possible: we normalized the <X, Y, Z> coordinates to eliminate translation from the search space and constrained scaling to be uniform in all dimensions. We attempted three mechanisms for estimation of the affine transformation: first, an approach based on the Singular Value Decomposition (SVD); next, a nonlinear minimization of transformation error based on the Levenberg-Marquardt algorithm; third, we exploited knowledge of surface normals of camera pose (the 3D direction at which the camera points) in the SfM point cloud and attempted to match with a 3D phone rotation matrix from compass and accelerometer (to find orientation relative to the vector of gravity). In all cases, sensor noise (in GPS, compass, and accelerometer) resulted in a nonsensical transformation. With only a few camera locations to connect one coordinate system to the other, and a still-large number of remaining degrees-of-freedom in the transformation, there is simply insufficient data to overcome sensor noise.

Attempts Intersecting Triangulation/Trilateration

After many unsuccessful attempts at applying computer vision techniques to estimate the affine transformation between coordinate systems, we attempted to simplify the localization problem. We tried to directly apply our intuitions for (1) compass triangulation; (2) visual trilateration; and (3) visual triangulation. The parameters of relative distance and interior angles can be trivially estimated from a SfM point cloud. If all sensors and vision techniques were fully-precise, the true (a, b) object location should fall at the intersection of those equations. In practice, after numerically solving the roots of all equation pairs, sensor noise and bias create many intersection points. We applied a 2D hierarchical clustering on these intersection points, hoping that one of the clusters would be distinctly dense, and the centroid of that cluster would be the estimated object location. In many cases, this proved correct. However, in more cases, sensor error (especially GPS) was large. Intersection points became diffused, and no longer indicative of the true location. To be practical, OPS would need to apply a more robust approach.

Attempts Optimizing Across Error Sources

We were encouraged by the partial-success of our second approach, directly applying our equations of triangulation and trilateration. Next, we attempted to push this technique further, this time integrating an optimization approach. We formulated a minimization problem on the error terms for each sensor value and vision-derived parameter, and with the optimization constrained to find a single object location (a, b). This led to a complex nonlinear optimization with many error terms and constraints. While this would occasionally converge to the correct object location, more often it found a trivial and nonsensical solution. For example, GPS error terms would "correct" all GPS locations to the same point. Further, the complexity of the optimization led to an impractically-long running time.

4.4 The Converged Design of OPS

From our early attempts to build a real-world object location model, we learned two important lessons that influenced the final design of OPS. First, any optimization would need to be constrained to be limited in the number of degrees-of-freedom and avoid degenerate cases. Second, we would need a mechanism to reduce the impact of GPS error. The final design of OPS consists of two optimization steps, each designed to limit the potential for degenerate solutions.

Before continuing, it is helpful the simplify our notion of location. We do not consider latitude and longitude directly, as angular values are inconvenient for measuring distances. Instead, we apply a precise Mercator projection to the square UTM coordinate system. Therefore, we can now refer to a latitude/longitude position as a simple 2D (x, y) coordinate. Recovery of the final <latitude, longitude> coordinate of the object is a straightforward inversion of this projection.

"Triangulation" via Minimization on Compass Error

Before explaining the optimizations underlying OPS, it is instructive to consider a reasonable baseline comparison. Since we assume that the user will take more than two photographs when using OPS, it would be unfair to compare OPS to a triangulation with only two GPS readings $(G_1^x, G_1^y), (G_2^x, G_2^y)$ and two compass bearings θ_1, θ_2. Instead, we must generalize the notion of triangulation to support as many measurements as will be available for OPS.

In Table 1, we present a nonlinear optimization that represents a triangulation-like approach to object localization.

Unlike standard triangulation, this scales to support an arbitrary number of GPS (G_i^x, G_i^y) and compass heading θ_i pairs. In noise-free conditions, all lines of compass bearing originating at the corresponding GPS point would converge to a single point (a, b). Of course, due to sensor error, we can expect that all pairs $\binom{n}{2}$ of compass lines will result in $\binom{n}{2}$ different intersection points. The optimization that follows seeks to find the most-likely single point of intersection by rotating each compass bearing as little as possible until all converge at the same fixed point (a, b). We experimented with a number of other approaches for "generalized triangulation." For one, we considered joint optimizations on GPS and compass. For another, we considered the 2D median of all $\binom{n}{2}$ intersections points. After experimentation, we believe this is the most-effective technique, and thus the fairest baseline comparison method to OPS.

$$\begin{aligned}
&\textbf{Minimize} && \sum_{\forall i} |E_i^\theta| \\
&\textbf{Subject to} && \\
&\forall i && : \; b - G_i^y = (a - G_i^x) \cdot \cot(\theta_i + E_i^\theta) \\
&\textbf{Solving for} && a, b \\
&\forall i && : \; E_i^\theta \\
&\textbf{With parameters} && \\
&\forall i && : \; G_i^x, G_i^y, \theta_i
\end{aligned}$$

Name	Parameter Sources
G_i^x, G_i^y	GPS position (of user at each photograph)
θ_i	Compass bearing (at each photograph)
Name	**Solved Variable Interpretation**
a, b	Estimated object location at (a, b)
E_i^θ	Estimated error for compass bearing θ_i

Table 1: Optimization for Triangulation

Now, we turn our attention to the two-step process employed in OPS. First, we apply the output of computer vision to correct for noise in GPS measurements. Second, we extend this baseline sensor-only optimization for triangulation to (1) use our corrected GPS points; and (2) exploit our intuitions for visual trilateration and triangulation.

Minimization of GPS Noise, Relative to Vision

From our earlier attempts, we realized that a direct application of our intuitions for visual trilateration and triangulation would be insufficient. Instead, we need a mechanism to reduce sensor noise before a object-positioning step can be effectively applied. Here, we rely on the output of structure from motion to correct for random GPS noise. Bias across all GPS measurements will remain. However, a consistent bias will be less damaging to the final localization result than imprecision in the relative GPS positions across multiple photographs. Structure from motion can help eliminate this noise between relative positions, as it tends to capture this relative structure with far greater precision than GPS. We design a nonlinear programing optimization that seeks to move the GPS points as little as possible, such that they match the corresponding relative structure known from vision.

We design an optimization that maps the original GPS points where photographs were taken $\{\forall i : (G_i^x, G_i^y)\}$ to a set of fixed GPS points $\{\forall i : (F_i^x, F_i^y) = (G_i^x + E_i^x, G_i^y + E_i^y)\}$. The optimization will also solve a scaling factor λ that proportionally shrinks or expands the point-to-point distances in the structure-from-motion point cloud to match the equivalent real-world distances measured in meters. The constraints simply enforce that the distance between any pair of GPS points, $\sqrt{(G_i^x - G_j^x)^2 + (G_i^y - G_j^y)^2}$, is equal to the distance between those same points in vision coordinates, $\sqrt{(V_i^h - V_j^h)^2 + (V_i^d - V_j^d)^2}$, after multiplying the vision distance by a constant factor λ.[3] Since we expect the GPS points to have some noise relative to the same points in vision, we introduce error terms for the GPS distance, $\sqrt{(E_i^x - E_j^x)^2 + (E_i^y - E_j^y)^2}$. With these error terms, the optimization is simply a minimization on the sum of squared error. Table 2 presents the complete optimization.

$$\begin{aligned}
&\textbf{Minimize} && \sum_{\forall i} \left({E_i^x}^2 + {E_i^y}^2 \right) \\
&\textbf{Subject to} && \\
&\forall i, j && : \; \left[(G_i^x + E_i^x) - (G_j^x + E_j^x) \right]^2 + \\
& && \left[(G_i^y + E_i^y) - (G_j^y + E_j^y) \right]^2 = \\
& && \lambda^2 \cdot \left[(V_i^h - V_j^h)^2 + (V_i^d - V_j^d)^2 \right] \\
&\textbf{Solving for} && \sigma \\
&\forall i && : \; E_i^x, E_i^y \\
&\textbf{With parameters} && \\
&\forall i && : \; G_i^x, G_i^y, V_i^h, V_i^d
\end{aligned}$$

Name	Parameter Sources
G_i^x, G_i^y	GPS position (of user at each photograph)
V_i^h, V_i^d	Vision position (arbitrary units)
Name	**Solved Variable Interpretation**
λ	Scaling factor, vision coordinates to GPS
E_i^x, E_i^y	Estimated camera GPS error

Table 2: OPS Optimization on GPS Error

OPS Optimization on Object Location

From the GPS-correction optimization (Table 2), we are left with a set of fixed GPS points, $\{\forall i : (F_i^x, F_i^y)\}$, and a scaling factor λ from vision to GPS coordinates. Now, we take these parameters to extend the baseline sensor-only optimization for triangulation (Table 1), along with additional context from visual trilateration and triangulation. We present this final optimization as Table 3.

The additional context for visual trilateration and triangulation is encoded as parameters $\{\forall i, j : \gamma_{ij}\}$, (C_x, C_y), and D. Each value γ_{ij} represents the angle from (F_i^x, F_i^y) to

[3]To avoid confusion with GPS x and y dimensions, we use h and d to represent the relevant two dimensions of the vision coordinate system. The h, or horizontal, dimension runs left/right of the object from the perspective of the user. The d, or depth, dimension runs towards/away from the object.

(a, b) to (F_j^x, F_j^y), estimated from vision. As represented in the notation, this is the positive acute angle between vectors $\vec{V}_i$ and $\vec{V}_j$. Thus, $\{\forall i, j : \gamma_{ij}\}$ directly encodes our original interpretation of visual triangulation. To avoid redundancies in the constraints (since triangulation and trilateration parameters are directly measured from the same vision point cloud), we only need to partially encode visual trilateration. Instead of encoding each of the relative distance from each camera point, we can simply enforce that the distance from the user's position to the object as a known, fixed value. We compute (C_x, C_y) as the 2D median across $\{\forall i : (F_i^x, F_i^y)\}$, by applying convex hull peeling. Next we enforce that the distance from (C_x, C_y) to the object at (a, b), $\sqrt{(a - C_x)^2 + (b - C_y)^2}$, is equal to the distance D. We can compute D from the vision point cloud along with the vision-to-GPS scaling factor λ.

The minimization function must change to accommodate context from triangulation (visual trilateration is fully incorporated as hard constraints). The addition of γ_{ij} error terms $\{\forall i, j : E_{ij}^{\gamma}\}$ allow angular error to be traded between magnetometer-derived compass headings and vision-derived angles. The compass error scaling factor, $(n-1)/2$, balances for the lessor quantity of compass error terms relative to pairwise vision angles.

$$
\begin{aligned}
\textbf{Minimize} \quad & \frac{n-1}{2} \cdot \sum_{\forall i} |E_i^{\theta}| + \sum_{\forall i,j} |E_{ij}^{\gamma}| \\
\textbf{Subject to} \quad & (a - C_x)^2 + (b - C_y)^2 = D^2 \\
\forall i \quad : \quad & b - F_i^y = (a - F_i^x) \cdot \cot(\theta_i + E_i^{\theta}) \\
\forall i \quad : \quad & \vec{V}_i = (a - F_i^x)\hat{i} + (b - F_i^y)\hat{j} \\
\forall i, j \quad : \quad & \gamma_{ij} + E_{ij}^{\gamma} = \\
& \arccos\left(\vec{V}_i \cdot \vec{V}_j / |\vec{V}_i||\vec{V}_j|\right) \\
\textbf{Solving for} \quad & a, b \\
\forall i \quad : \quad & E_i^{\theta} \\
\forall i, j \quad : \quad & E_{ij}^{\gamma} \\
\textbf{With parameters} \quad & C_x, C_y, D \\
\forall i \quad : \quad & F_i^x, F_i^y, \theta_i \\
\forall i, j \quad : \quad & \gamma_{ij}
\end{aligned}
$$

Name	Parameter Sources
F_i^x, F_i^y	Fixed GPS position (at each photograph)
θ_i	Compass bearing (at each photograph)
γ_{ij}	Vision estimate for vector angle $\angle \vec{V}_i \vec{V}_j$
C_x, C_y	2D Median of $\{\forall i : (F_i^x, F_i^y)\}$
D	Estimated distance from (C_x, C_y) to (a, b)
Name	**Intermediate Results**
$\vec{V}_i$	Vector from (F_i^x, F_i^y) to (a, b)
Name	**Solved Variable Interpretation**
a, b	Estimated object location at (a, b)
E_i^{θ}	Estimated error for compass bearing θ_i
E_{ij}^{γ}	Estimated error in vector angle $\angle \vec{V}_i \vec{V}_j$

Table 3: OPS Final Object Localization

5. DISCUSSION

Extending the Location Model to 3D

In Section 4.4, we described how OPS estimates the object-of-interest location in two dimensions, namely as the point (a, b). In some contexts, the 3D location of the object can also be useful. For example, we might want to localize a particular window of a multistory building. Ultimately, OPS should provide a $<$ latitude, longitude, altitude $>$ location tuple. However, the height dimension adds additional challenges not faced on the ground, following a plane tangential to the Earth's surface. First, GPS-estimated altitude is prone to greater inaccuracy than latitude and longitude. Second, while it is natural for a user to take multiple photos by walking a few steps in-between, a requirement to take photographs at multiple heights would become awkward. Thus, while vision provides three-dimensional geometry, GPS locations for photographs are roughly planar. Further, since the object localization task is already challenging in two dimensions, it is desirable to avoid integrating 3D into our location optimizations.

Instead, OPS finds the two-dimensional object location first. Next, it uses the now-known distance to the object, along with accelerometer and vision inputs, to estimate height. For each photograph, OPS records the raw three-axis accelerometer output. Since we can expect the phone to be roughly still while the photograph is taken, the accelerometer is anticipated to measure only gravitational force. This gravitational vector defines a unique orientation in terms of phone roll (rotational movement on the plane of the phone screen, relative to the ground) and pitch (rotational movement orthogonal to the plane of the phone screen, relative to the horizon). Pitch provides a rough estimate of how much higher (or lower) the user is focusing, relative to a plane parallel to the ground and intersecting the user at eye-level. To improve the accuracy of this measurement, we can "average" pitch measurements from every photograph. Importantly, the user might not align the object in every photo at exactly the same pitch. For example, the window might appear higher or lower on the screen. We can correct for this by leveraging vision once again. From our 3D point cloud, there is a unique mapping of every 3D point back to each original 2D image. We can now compute an adjustment value, measured in pixels, from the horizontal center line of the screen. We can convert this pixel value to an angle, given the known camera field-of-view. Next, the angular sum of pitch and adjustment, averaged across all photographs, can be used to estimate height when combined with the known two-dimensional distance.

$$h_{\text{object}} = h_{\text{observer}} + \frac{\text{2D distance}}{2 \cdot \tan\left(\frac{\text{pitch}+\text{adjustment}}{2}\right)}$$

Alternatives for Capturing User Intent

If we can make assumptions regarding the structure of the object-of-interest, computer vision techniques of segmentation can assist in isolation of the object from the structure-from-motion point cloud. For example, consider a typical multistory building with large flat sides. Images of an exterior wall will tend to yield many keypoints along a flat plane, roughly perpendicular to the ground plane. These points on the wall plane are often clearly distinct from points on the ground, smaller clusters of points in the nearest-foreground from occlusions, or sparse points in the distant background.

To determine where the object-of-interest lies within a point cloud, we attempt to segment the point cloud and find such a predominate plane. We apply Random Sample Consensus (RANSAC) [5], an iterative method to estimate a model for a plane, under an angular constraint that it must be roughly-perpendicular to the ground and parallel to the field-of-view. All points in the point cloud are then classified as either inliers or outliers to the plane. Next, we find the spatial centroid among inliers to the plane. This point is considered to be the object-of-interest.

6. EVALUATION

We take a systems-oriented approach in evaluating OPS, so as to capture real-world performance. Phone sensors are subject to noise and biases. GPS can be impacted by weather (due to atmospheric delay), clock errors, errors in estimated satellite ephemeris, multipath, and internal noise sources from receiver hardware. Compass magnetometers are affected by variations in the Earth's magnetic field and nearby ferromagnetic material. Computer vision techniques, such as structure from motion, can break down in a variety of scenarios. For example, keypoint extraction may fail if photographs have insufficient overlap, are blurred, are under or over-exposed, or are taken with too dark or bright conditions (such as when the sun is in the user's eyes). The primary goal of our evaluation is to consider how well OPS overcomes this naturally-challenging operational context.

6.1 Implementation

OPS is implemented in two parts, an OPS smartphone client and a back-end server application. We built and tested the OPS client on the Google NexusS phone, as a Java extension to the standard Android 2.4 camera program. Photographs may be pre-processed locally on the phone to extract keypoints and feature descriptors (reducing the required data transfer), or simply uploaded to our server for processing (faster with a high-performance WiFi connection). Along with photographs (or keypoints and descriptors), the phone uploads all available sensor data from when each photograph was taken, to include GPS, compass, and accelerometer. Our server is a Lenovo desktop running Ubuntu Linux 11.04.

Computer Vision

Both the client and server applications support the basic computer vision tasks of keypoint detection and extraction of feature descriptors. We use the SURF (Speeded Up Robust Feature) algorithm [1]. We choose SURF over the related SIFT (Scale Invariant Feature Transform) [11] as it is known to be considerably faster to compute and provide a greater robustness against image transformations. SURF detection and extraction are performed using OpenCV. For both the client and server-side applications, JavaCV provides java wrappers of native C++ calls into OpenCV.

For server-side structure from motion, we use Bundler, an open-source project written by Noah Snavely and the basis for the Microsoft Photosynth project [17]. Bundler operates on an unordered set of images to incrementally-build a 3D reconstruction of camera pose and scene geometry. As input, Bundler expects keypoints matched across multiple photos (on the bases of the corresponding feature descriptors). Bundler can operate on any keypoint type, expecting SIFT by default. We adapt the output of the OpenCV SURF detector to match the SIFT-based input expected by Bundler, substantially decreasing processing time per photo on the phone. As output, Bundler provides a sparse point cloud representation of the scene in view. OPS builds its real-world localization model on top of this point cloud, which exists at an unknown relative scaling, translation, roll, pitch, and tilt from the corresponding locations in the real-world.

Server-side Nonlinear Optimization

The OPS server is implemented primarily in Java. Mathematica is used through a Java API for solving nonlinear optimizations. We use differential evolution as the optimization metaheuristic for its strong robustness to local minima (simulated annealing proved to be similarly effective, with both outperforming the default Nelder-Mead) [18].

6.2 Accuracy of Object Localization

We tested OPS at more than 50 locations on or near the Duke University campus. We attempted to use OPS in the most natural way possible, focusing on localization tests that mirror how we would expect a real user would want to use the system. Primarily, we considered objects at distances between 30m and 150m away, for two reasons. First, the object should be far enough away that it makes sense to use the system, despite the hassle of taking photographs. Though it only takes about a minute to take the required photographs, a user should not be more willing to simply walk over to the object to get a GPS lock. Second, distances are limited by the user's ability to clearly see the object and focus a photograph. Building densities, building heights, and presence of significant occlusions (such as trees), constrain the distances at which photographs can be easily taken.

We compare OPS accuracy to "Optimization (Opt.) Triangulation." To provide the fairest comparison, Triangulation reflects the triangulation-like optimization described in Section 4.4, designed to scale to an arbitrary number of GPS, compass-heading pairs. For some graphs, we also show the distance from the position at which photographs were taken (centroid across all photographs) to the true object location. This is an important consideration; as sensor values are projected into the distance, noise and bias can be magnified.

Testing Parameters

OPS performance is dependent on the way OPS is used. For our evaluation, we took four photographs for each localization (the minimum required for structure from motion). Each successful photograph was taken between 0.75m and 1.5m from the location of the preceding photograph, in no specified direction. This flexibility is important, as it allows the user the compose the shot as is natural, given obstacles in the vicinity and visual occlusions. The distance (about one or two paces) is sufficiently far that GPS is able to estimate distance with sufficient (albeit still poor) accuracy. The distance is sufficiently small as to ensure a relatively small angular separation to the object of interest, enabling structure from motion to work well. Further, the distances are not too burdensome for the user; the required photographs can be taken in less than one minute. Once the user takes the required photos, processing requires approximately an additional 30-60 seconds, primarily attributable to structure from motion computation and depending on the

Figure 10: Sampled tests; circle denotes object-of-interest (top), Google Earth view (bottom): (a) smokestack of a coal power plant; (b) distant building with vehicles in foreground; (c) stadium seats near goal post.

number of keypoints detected in each photo. With additional photographs, both accuracy and processing time increase.

Structure from motion is sensitive to the quality of photographs. In poor lighting, keypoint detection becomes less robust. Thus, OPS is sensitive to the quality of lighting conditions. At night, there is unlikely to be sufficient light for high-quality photographs. Further, a camera flash is useless for the distances under consideration. At dawn and dusk, direct sun into the camera lens can also ruin photographs. We test OPS in daylight conditions, and avoid early morning and late evening. We are considering extensions to OPS to enhance performance in poor lighting, likely avoiding structure from motion in these cases.

Example Usage Scenarios

In Figure 10, we show three example photos, taken while using OPS. Below each photo, we show a screenshot taken from Google Earth with four annotations, (1) the location at which the user photographed the object-of-interest; (2) the true location of the intended object (positioned at the center of the screen in each photo; (3) the object position inferred by Opt. Triangulation; and (4) the object position inferred by OPS. We show these particular examples to highlight that OPS is a general approach, and can therefore localize for a wide variety of distant objects. Further, while the final result may still be a substantial distance from the precise true location, OPS is typically able to find a position that is far more representative of the object-in-view than triangulation.

Overall Performance

Figures 11 and 12 present overall performance of OPS across all locations. Figure 11 shows three CDF curves, (1) error in meters from the OPS-localized position to the true object position; (2) error in meters from the Opt. Triangulation position to the true object position; and (3) distance in meters from the position at which photographs were taken (centroid across all photographs) to the true object location. Figure 12 shows individual localization results, sorted by user-to-object distance. Overall, OPS provides a substantial performance improvement over triangulation.

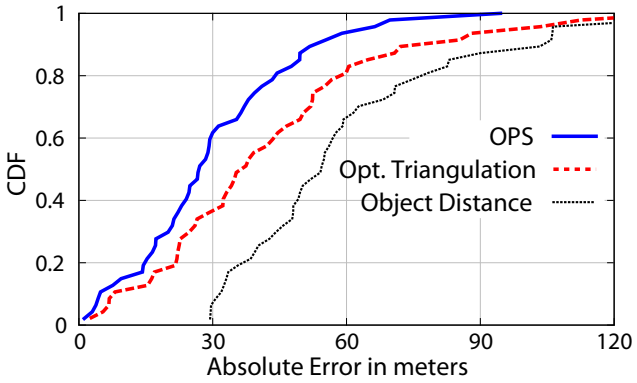

Figure 11: CDF of error across all locations. Graph reflects four photos taken per location. 50 locations.

Sensitivity to GPS and Compass Error

To better understand OPS's robustness to sensor noise, we wanted to carefully evaluate the impact of GPS and compass noise, in isolation. We took a set of four photographs of a Duke University Science Center building, from a mean distance of 87m. We ensured that this particular set of photographs provided a robust model from vision, through a manual inspection of the point cloud. For each photograph, we used Google Earth to mark the exact location at which

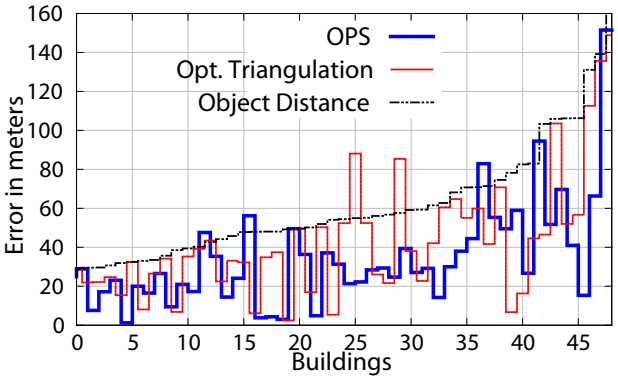

Figure 12: OPS and triangulation error at 50 locations. Graph reflects four photos taken per location.

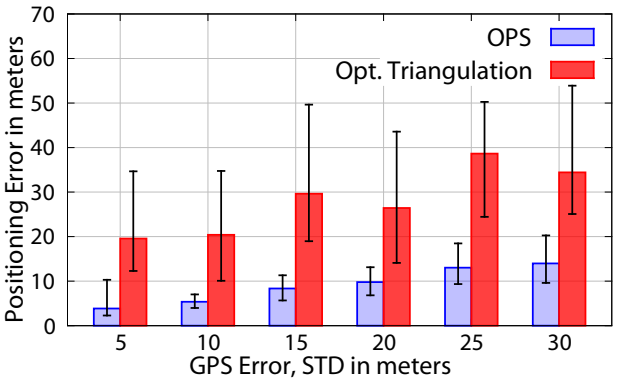

Figure 13: Error from ground-truth GPS camera locations. X-axis shows the standard deviation of introduced Gaussian GPS errors. Bars show median error; whiskers show first and third quartiles.

it was taken, with less than one meter error. From these carefully-marked points, we mathematically computed the compass bearing to the object, assuring less than 5° error (attributable to error in marking the points). Next, we randomly perturbed each GPS point or compass angle according to random deviates of a Gaussian (Normal) distribution with mean 0 and a varied standard deviation. Figure 13 shows performance as a function of the standard deviation of injected GPS error (in meters). OPS is able to leverage vision to remove much of this GPS noise, providing strong robustness. Figure 14 shows performance as a function of the standard deviation of injected compass error (in degrees). Superior performance of OPS is especially reflective of how visual triangulation enforces the true relative angle between a pair camera locations and the object position. Bars reflect median of 50 trials with first and third quartiles on whiskers.

Sensitivity to Photograph Detail

Figure 15 shows OPS performance with varied levels of photographic detail. For this experiment, four photographs were used, along with ground-truth GPS and compass data. Full-resolution photos were downsampled, reducing the efficacy of keypoint detectors. With fewer keypoints, especially at resolutions below 1024x768 pixels, structure from motion suffers and can impact the object position.

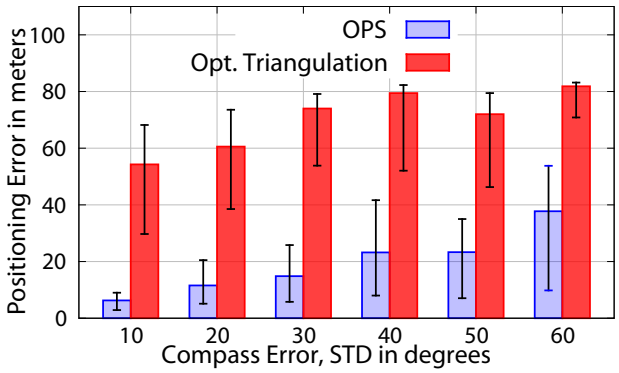

Figure 14: Error from ground-truth GPS camera locations. X-axis shows the standard deviation of introduced Gaussian compass errors. Bars show median error; whiskers show first and third quartiles.

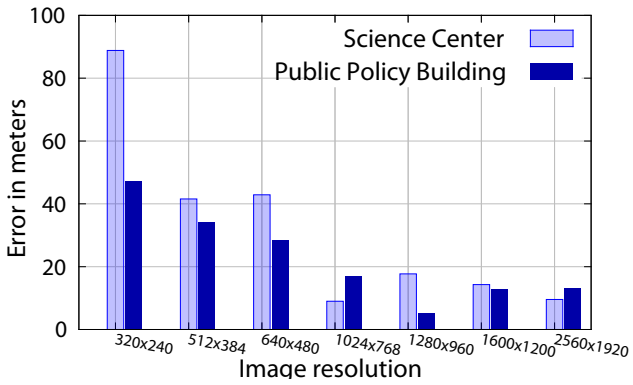

Figure 15: OPS error by photo resolution. Keypoint detection is less reliable below 1024x768 pixels.

7. ROOM FOR IMPROVEMENT

OPS remains an ongoing research project. We are exploring a variety of mechanisms to improve accuracy.

Live Feedback to Improve Photograph Quality

OPS is impacted by the quality of user photographs. Poor angular separation between two photographs (too small or large) can reduce the efficacy of structure from motion. We imagine a system of continuous feedback to the user, aiding in shot selection and framing. By combining an especially-lightweight keypoint detection heuristic, such as FAST [15], on the camera video with information from the phone gyroscope, it would be feasible to suggest when a "good" photograph can be taken. Otherwise, we would inform the user to take a corrective step (to the left or right).

Improving GPS Precision with Dead Reckoning

From Figure 13 and 14, it is clear that OPS is already relatively insensitive to GPS and magnetometer noise. Although OPS explicitly uses the output of structure from motion to cancel GPS noise, large GPS errors can still become disruptive – this is quite often the cause of poor performance. One possibility is to apply additional sensors to improve GPS precision. In particular, a constant bias is less damaging

than imprecise relative positions across camera locations. We are considering mechanisms to leverage gyroscope and accelerometer for dead reckoning between camera locations.

Continual Estimation of Relative Positions with Video
Continuous video could potentially be used to substantially augment or even replace the structure-from-motion point cloud. A frame-by-frame comparison of these keypoints, in fine-grained comparison with accelerometer, gyroscope, compass, and GPS, could provide a highly-detailed trace of how the smartphone (1) moved in 3D space and (2) viewed the object-of-interest in terms of relative angles/distances.

8. RELATED WORK

To the best of our knowledge, OPS is a first-of-kind system to find the precise position for objects of significant distance away, despite noisy sensors and without requiring a database of pre-existing photographs or maps. Nonetheless, there is a substantial body of related work, especially leveraging computer vision for recognition of well-known landmarks.

Localization through Large-Scale Visual Clustering
OPS is related to the problem of worldwide localization on the basis of visual classification [4, 6, 10, 19, 21]. Most closely related to OPS, [21] is the back-end system underlying Google Goggles: (1) online travel guides are mined for possible tourist landmarks; (2) photo-sharing databases of millions of GPS-tagged photos are searched using keywords from these travel guides; (3) unsupervised visual clustering on these search results provide a visual model of the landmark. From 20 million GPS-tagged photos, 5312 landmarks can be recognized by visual comparison of a photograph with these landmark models. OPS is designed to be more generic, able to localize objects which cannot be considered landmarks (without a pre-exiting photo library of these objects). We believe that a hybrid approach can be valuable, which leverages a photo database to enhance accuracy (where useful photos are available), but can also provide OPS functionality in the general case.

Aligning Structure from Motion to the Real World
Substantial computer vision literature has considered improvements and applications of structure from motion (SfM) [17]. However, our notion of "object positioning" should not be confused with the computer vision concept of object localization, which seeks to find the relative location of objects within an image or point cloud. For mobile systems, closely related to OPS, [7] uses SfM for a "landmark-based" directional navigation system. SfM, along with a preexisting database containing many photographs of the area in view, enable a augmented reality view with highlighted annotations for known landmarks. More-directly related to the goals of OPS, [8] seeks to align the output of structure from motion to real-world coordinates, using GPS coordinates to improve the scalability of structure from motion when using hundreds of photos. However, the techniques require the availability of overhead maps and are not suitable to the extremely small number of photos (typically four) expected for OPS.

Building Object Inventories
Our use of multimodal sensing inputs for estimating object position is related to the techniques in [3] for building inventories of books in a library. The authors project a rough indoor location for a book through WiFi and compass, then apply vision techniques to visually detect book spines.

Applying Computer Vision to Infer Context
CrowdSearch [20] combines human validation with location-specific image search (for objects such as buildings) through computer vision. By leveraging Amazon Mechanical Turk for human-in-the-loop operation, CrowdSearch enables precision in cases of poor image quality. TagSense [14] infers "tags" of human context for photographs, by combining computer vision with multimodal sensing. OPS is complementary to CrowdSearch and TagSense, by enabling awareness of precise location for objects in photographs.

Object Recognition and Location Inference
In [12], the authors try to estimate the 3D-orientation and location of an object in the real world scene using a polygonal 3D-model of a depicted object. The accuracy of OPS could be enhanced opportunistically by utilizing the size of known, recognizable objects when they happen to appear in the view.

Robotics Navigation and Obstacle Avoidance
Mobile robot navigation requires awareness of object position (localization) for navigation and obstacle avoidance. [2] surveys various techniques such as odometry, inertial navigation, active beacons and model matching for relative and absolute position measurement. These techniques typically require specialized hardware, OPS attempts to solve object localization using off-the-shelf smartphones.

9. CONCLUSION

Today's augmented reality applications are less useful than their potential. We believe this is due, at least in part, to an unfortunate usage asymmetry. Users can retrieve available localized content in a natural way, viewing pop-up annotations for the world through their smartphone, but there is no means to equivalently introduce new annotations of objects in the vicinity. By providing localization for objects in a user's view, this paper seeks to enable convenient content creation for augmented reality. Beyond augmented reality, we believe that a precise *Object Positioning System* can be widely enabling for a variety of applications, and is worthy of a significant research endeavor. In this light, we believe that our approach, OPS, takes a substantial first step.

10. REFERENCES

[1] H. Bay, T. Tuytelaars, and L. Van Gool. Surf: Speeded up robust features. In *ECCV*, 2006.

[2] J. Borenstein, H. Everett, L. Feng, and D. Wehe. Mobile robot positioning-sensors and techniques. Technical report, DTIC Document, 1997.

[3] D. M. Chen, S. S. Tsai, B. Girod, C.-H. Hsu, K.-H. Kim, and J. P. Singh. Building book inventories using smartphones. In *Proceedings of the international conference on Multimedia*, MM '10, pages 651–654, New York, NY, USA, 2010. ACM.

[4] G. Cuellar, D. Eckles, and M. Spasojevic. Photos for information: a field study of cameraphone computer vision interactions in tourism. In *CHI '08 extended abstracts on Human factors in computing systems*, CHI

EA '08, pages 3243–3248, New York, NY, USA, 2008. ACM.

[5] M. Fischler and R. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[6] J. Hays and A. Efros. Im2gps: estimating geographic information from a single image. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8, June 2008.

[7] H. Hile, A. Liu, G. Borriello, R. Grzeszczuk, R. Vedantham, and J. Kosecka. Visual navigation for mobile devices. *IEEE MultiMedia*, 17:16–25, April 2010.

[8] R. Kaminsky, N. Snavely, S. Seitz, and R. Szeliski. Alignment of 3d point clouds to overhead images. In *Computer Vision and Pattern Recognition Workshops, 2009. CVPR Workshops 2009. IEEE Computer Society Conference on*, pages 63–70, June 2009.

[9] J. Koenderink, A. Van Doorn, et al. Affine structure from motion. *JOSA A*, 8(2):377–385, 1991.

[10] Y. Li, D. Crandall, and D. Huttenlocher. Landmark classification in large-scale image collections. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 1957 –1964, 29 2009-oct. 2 2009.

[11] D. G. Lowe. Object recognition from local scale-invariant features. In *Proceedings of the International Conference on Computer Vision-Volume 2 - Volume 2*, ICCV '99, pages 1150–, Washington, DC, USA, 1999. IEEE Computer Society.

[12] M. Magnor. Geometry-based automatic object localization and 3-d pose detection. In *Image Analysis and Interpretation, 2002. Proceedings. Fifth IEEE Southwest Symposium on*, pages 144–147. IEEE, 2002.

[13] S. Moon, I. Moon, and K. Yi. Design, tuning, and evaluation of a full-range adaptive cruise control system with collision avoidance. *Control Engineering Practice*, 17(4):442–455, 2009.

[14] C. Qin, X. Bao, R. Roy Choudhury, and S. Nelakuditi. Tagsense: a smartphone-based approach to automatic image tagging. In *Proceedings of the 9th international conference on Mobile systems, applications, and services*, MobiSys '11, pages 1–14, New York, NY, USA, 2011. ACM.

[15] E. Rosten and T. Drummond. Machine learning for high-speed corner detection. In *European Conference on Computer Vision (ECCV)*, pages 430–443, 2006.

[16] G. Schall, J. Schöning, V. Paelke, and G. Gartner. *A survey on augmented maps and environments: Approaches, interactions and applications*. Taylor & Francis Group, 2011.

[17] N. Snavely, S. M. Seitz, and R. Szeliski. Photo tourism: exploring photo collections in 3d. In *ACM SIGGRAPH 2006 Papers*, SIGGRAPH '06, pages 835–846, New York, NY, USA, 2006. ACM.

[18] R. Storn and K. Price. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization*, 11(4):341–359, 1997.

[19] G. Takacs, V. Chandrasekhar, N. Gelfand, Y. Xiong, W. Chen, T. Bismpigiannis, R. Grzeszczuk, K. Pulli, and B. Girod. Outdoors augmented reality on mobile phone using loxel-based visual feature organization. In *Proceeding of the 1st ACM international conference on Multimedia information retrieval*, pages 427–434. ACM, 2008.

[20] T. Yan, V. Kumar, and D. Ganesan. Crowdsearch: exploiting crowds for accurate real-time image search on mobile phones. In *Proceedings of the 8th international conference on Mobile systems, applications, and services*, MobiSys '10, pages 77–90, New York, NY, USA, 2010. ACM.

[21] Y.-T. Zheng, M. Zhao, Y. Song, H. Adam, U. Buddemeier, A. Bissacco, F. Brucher, T.-S. Chua, and H. Neven. Tour the world: Building a web-scale landmark recognition engine. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 1085–1092, June 2009.

Acknowledgments

We sincerely thank our shepherd, Gaetano Borriello, as well as the anonymous reviewers, for their invaluable feedback. We are grateful to NSF for partially funding this research through the following grants: CNS-0916995 and IIS-910846.

APPENDIX

We provide equations for curves of visual trilateration and triangulation. We assume at least two GPS points are known $(x_1, y_1), (x_2, y_2)$. The object position is inferred at (a, b).

(1) Equation of Visual Trilateration

Let $\sigma = d_2/d_1$ be the ratio of distance from (x_2, y_2) to (a, b) divided by the distance (x_1, y_1) to (a, b).

$$(a - x_2)^2 + (b - y_2)^2 = \sigma^2 \left[(a - x_1)^2 + (b - y_1)^2\right]$$

To derive, construct two right triangles with hypotenuses (x_1, y_1) to (a, b) and (x_2, y_2) to (a, b). Apply the Pythagorean theorem on each, substituting the first into the second.

(2) Equation of Visual Triangulation

Let γ be the interior angle from (x_1, y_1) to (a, b) to (x_2, y_2).

$$\left[(x_2 - x_1)^2 + (y_2 - y_1)^2\right] = \left[(a - x_1)^2 + (b - y_1)^2\right] + \left[(a - x_2)^2 + (b - y_2)^2\right] - 2\sqrt{(a - x_1)^2 + (b - y_1)^2}\sqrt{(a - x_2)^2 + (b - y_2)^2}\cos\gamma$$

To derive, apply the law of cosines ($C^2 = A^2 + B^2 - 2AB\cos\gamma$) with C taken as the side $\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$.