# Practical Urban Localization for Mobile AR

Tiantu Xu*
Purdue ECE

Guohui Wang
ByteDance

Felix Xiaozhu Lin
Purdue ECE

## ABSTRACT

Emerging mobile apps render AR effects based on the places of interest (POI) that a user is currently in. To obtain the needed POI labels and a smartphone's camera position and orientation, such apps demand inexpensive localization. Yet, existing localization solutions either work poorly in urban areas or require expensive data collection. To this end, we advocate for an inexpensive, practical localization pipeline by integrating commodity vision operators. To instantiate the pipeline, we propose a system with three key designs: the cloud indexes image features as a forest rather than a monolithic tree; smartphones incrementally prefetch image features for on-device matching rather than uploading features to the cloud; smartphones tune the camera positioning algorithm dynamically based on its physical environment. Our preliminary results show that these designs can reduce the cost of image data collection by up to three orders of magnitude, reduce user-perceived delays, and scale to diverse AR resource demands and environments.

## CCS CONCEPTS

• **Human-centered computing** → **Ubiquitous and mobile computing**; • **Information systems** → **Image search**; **Personalization**.

## KEYWORDS

Mobile Augmented Reality; Urban Localization

**ACM Reference Format:**
Tiantu Xu, Guohui Wang, and Felix Xiaozhu Lin. 2020. Practical Urban Localization for Mobile AR. In *Proceedings of the 21st International Workshop on Mobile Computing Systems and Applications (HotMobile '20), March 3–4, 2020, Austin, TX, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3376897.3377855

## 1 INTRODUCTION

Places of interest (POI) apps render AR effects on-the-fly based on a smartphone's location and position. Integrated into video sharing apps such as TikTok, POI AR opens a new door to deliver relevant advertisements. For instance, while a user uses her smartphone to shoot a short video in front of a coffee shop, the smartphone renders coffee promotions as a virtual overlay on the screen. Of such POI AR apps: i) the typical scenarios are urban canyons where merchants proliferate; ii) the AR effects are specific to the exact camera location and position; iii) the effects are interactive and continuously rendered in realtime as the camera moves. Table 1 summarizes three classes of POI AR with a variety of visual effects and resource demands. These classes are *Marker-based*: AR effects are overlaid on known physical objects, i.e., markers; *Markerless*: AR effects are rendered in free space, e.g., an interactive 3D Medusa in front of a Starbucks shop; *Superimposition*: AR effects provide an alternate view of an object, e.g., a virtual overlay aligned on a wall.

| AR Type | Localization Error | Per-frame Delay |
|---|---|---|
| Marker-based | ~ a few meters | ~1$s$ |
| Markerless | < 1 meter | tens of ms (real-time) |
| Superimposition | < 10 centimeters | tens of ms (real-time) |

**Table 1: Tolerance of localization errors/delays of AR**

| | GPS | Wi-Fi | 3D Model | This work |
|---|---|---|---|---|
| Environment | Open Outdoor | Indoor | Indoor & Outdoor | Indoor & Outdoor |
| Error | High | Low | Low | Low |
| Position | 3DoF | 3DoF | 6DoF | 6DoF |
| Ahead-of-time cost | Low | Low | High | Low |
| Runtime cost | Low | Low | Low | Low |

**Table 2: A comparison of localization solutions**

This paper explores localization customized for POI AR in urban areas. To POI AR, the localization system supplies twofold information, as shown in Table 2: a POI label describing the user's current geolocation, e.g., "the Starbucks on Main Street"; the six degrees of freedom (6DoF) camera pose consisting of camera position (X, Y, Z) and orientation (yaw, pitch, roll). The localization system should meet the following objectives. *i) Low-cost data collection* in constructing geo-visual databases that cover urban areas; *ii) Soft real-time camera positioning* with tens of ms delay for smooth user experiences; *iii) Privacy*: the smartphone should avoid uploading visual data whenever possible; *iv) Proportional cost*: the system should incur low computational cost when the AR's error and latency requirements are relaxed, as listed in Table 1.

Most well-known localization techniques are inadequate for POI AR. First, their positioning errors exceed the tolerance of POI labels, especially in the urban outdoor environment. GPS sees errors as high as 30 meters in urban canyons due to obstruction [4, 15]; even calibrated with inertial sensors, the error is still around 10 meters [4, 5]. Although recent peer-assisted or multipath WiFi-based localization [16, 17] achieves centimeter-level errors, the techniques are restricted to indoor settings. More importantly, neither GPS nor WiFi can estimate 6DoF camera poses. Although fusing 3DoF WiFi localization and 3DoF orientation estimation, e.g., from gyroscopes, was reported effective, the technique is mostly limited to indoor environments [1].

*This work was performed during an internship at ByteDance.


*HotMobile '20, March 3–4, 2020, Austin, TX, USA*

ACM ISBN 978-1-4503-7116-2/20/03…$15.00
https://doi.org/10.1145/3376897.3377855

By contrast, vision-based approaches show a higher promise [36]. Among them, 3D model reconstruction is commonly used in commercial solutions [24, 25, 29, 32]: with reference to a colossal, pre-generated 3D point cloud, smartphones can estimate location and position with cm-level accuracy [32]. Yet, generating 3D models requires as many as a few thousand images collected for *each* building [27]. For POI AR, constructing 3D models for an entire urban area is unscalable.

To this end, we propose to construct an inexpensive localization pipeline with two vision algorithms in tandem: image feature matching [21, 26] and simultaneous localization and mapping (SLAM). As shown in Figure 1, the system vendor collects a moderate number of images covering the urban area, typically tens per POI, ahead of time; from the images, the vendor extracts and indexes visual features in K-D Trees called vocabulary trees [21, 26]. When a user starts a POI AR app, her smartphone identifies its current location by matching features extracted from the camera video feed with the features stored in the vocabulary tree, continuously estimates camera pose using SLAM, and renders AR effects accordingly.

To instantiate the pipeline spanning the smartphones and the cloud, we present three key system designs, as illustrated in Figure 1.

**Indexing image features as a vocabulary forest (§3.1)** While prior work typically constructs one vocabulary tree covering all POIs [26], doing so for the entire urban area results in a deep, monolithic tree that is expensive to search into. To this end, we build a *vocabulary forest* consisting of numerous shallow trees, each covering a region as large as GPS's typical error region, e.g., a circle with a ~30 m radius. Smaller coverage not only reduces the delay in feature search but also reduces false positives, as will be shown in Section 4.

**Incremental tree prefetch (§3.2)** While smartphones may upload image features to the cloud for matching against the features stored in the vocabulary trees, doing so raises privacy concerns over the uploaded visual data. As such, we propose for smartphones to fetch the vocabulary forest to execute matching solely on the devices. As the whole forest can be colossal (e.g., that for a small town like Palo Alto is more than 50 GB [26]), the smartphone fetches incrementally: it fetches the tress for the geo-regions that the user is about to enter, determined based on the current smartphone GPS location and the predicted trajectory. In prefetch, the smartphone prioritizes the trees that cover more popular POIs and have larger overlaps with the GPS's error region.

**Adapting camera positioning to diverse environments (§3.3)** For camera positioning, we exploit the rich tradeoffs between AR errors and delays. Accordingly, the system tunes SLAM parameters, e.g., video resolution, frame rate, and the number of per-frame SIFT features [19], on the fly. The system further adapts its positioning algorithm to diverse environments that challenge positioning accuracy. For instance, while SIFT features are often detected on object boundary edges, they become inadequate on smooth and distinctive building textures [33]. In this case, the system automatically boosts image resolution or frame rate to prevent potential accuracy loss.

Overall, this paper advocates for deep customization of localization for mobile AR applications, which serves as a stepping stone towards making POI AR deployable on billions of smartphones and affordable in thousands of urban areas.

# 2 A CASE FOR CUSTOMIZING LOCALIZATION FOR POI AR

## 2.1 Observations

We summarized a few observations that motivate our key design choices below.

**Observation 1. Constructing geo-visual databases is significantly cheaper than 3D models** Constructing a geo-visual database requires fewer images, and consumes fewer compute and storage resources of up to three orders of magnitude compared to reconstructing the entire 3D city model that has been carried out by industrial pioneers [24, 25, 29, 32]. Narrowing down the scenarios to POI AR applications, 3D model reconstruction is an overkill. We made a head-to-head cost comparison between the above two approaches on several stores on the University Avenue in Palo Alto, a small town in CA, as will be shown in Section 4.

**Observation 2. Smaller tree coverage speeds up image matching and reduces the false positive rate** Smaller region coverage corresponds to fewer images and fewer features, so that the vocabulary tree size could be significantly smaller, resulting in faster image feature searching and matching. Besides, smaller coverage also effectively reduces the number of POI label candidates, which often ends up with a higher top-1 accuracy during image matching. A measurement of the image matching speed and accuracy over a different range of coverage will be shown in Section 4.

**Observation 3. Uploading images to the cloud leaks privacy** Privacy is always a top concern, especially for location-based services involving sensitive private data. However, prior AR work [15, 31] that uploads the user's camera images to the cloud may incur the risk of data leakage threats. The attacks may come from malicious public WiFi [35], untrusted cloud [2], or side-channel attacks [37]. The above privacy issue motivates our design choice of blinding the cloud by fetching the vocabulary trees to the user's smartphone for on-device matching.

**Observation 4. SLAM parameters are tunable to adapt to different AR types and various environments** The accuracy of camera positioning varies with input video formats, e.g., resolution, frame rate, and SLAM parameters, such as the number of features extracted per frame. This tradeoff caters to the diverse accuracy requirements from different AR types shown in Table 1, which motivates our design of choosing a different set of parameter values corresponding to each AR type. Besides, under some special conditions, e.g., much fewer features could be extracted from smooth building textures, SLAM is elastic in preventing dramatic accuracy drops by consuming more expensive video formats. We emulate different conditions by benchmarking SLAM [20] on real-world datasets [30], and more details will be shown in Section 4.

## 2.2 System Overview

With the above observations, we advocate for a practical urban localization system for mobile POI AR applications, with the proposed pipeline shown in Figure 1. Figure 1(a) illustrates the ahead-of-time vocabulary forest construction, composed by data collection and geo-visual database construction. During the data collection stage, the system vendor collects image features from real street scenes.

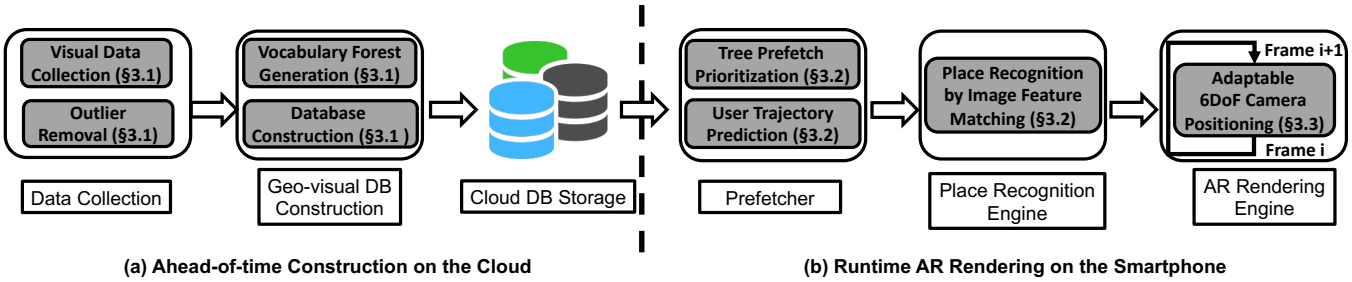



**Figure 1: Proposed System**

Irrelevant features from random dynamic entities, e.g., persons or automobiles that happen to pass by, are detected by general object detectors, e.g., YOLO [23], and are removed from the image ahead of time. With the image collected and outliers from irrelevant entities removed, the system vendor constructs numerous vocabulary trees from each group of images covering an independent region. Eventually, all those vocabulary trees are stored on the cloud server. Figure 1(b) illustrates a single life cycle of AR effect rendering on smartphones, which breaks down into vocabulary tree prefetch, on-device POI recognition, and adaptable camera positioning. To reduce the end-to-end latency, the smartphone will run a prefetcher that continuously pulls nearby the vocabulary trees prioritized by the user's current GPS location and predicted trajectory. Upon the app launch, the recognition engine on the smartphone will match the features extracted from the current camera view with pre-downloaded features from vocabulary trees. With the POI label returned by picking the top match, the user then picks up her favorite POI AR effect related to the current POI. The AR rendering engine then continuously updates the AR model pose by estimating the 6DoF camera pose on each frame through SLAM, which can adapt to various AR types and environments.

## 3 DESIGN

We next describe the three key aspects of our system design, as shown in Figure 1.

### 3.1 Indexing Features as Vocabulary Forest

Despite the fact that a single vocabulary tree can scale up to 1M images [21], for both latency- and privacy-sensitive applications, we seek to enforce the latency cap by shrinking the vocabulary size to match the GPS error bound (30m), motivated by observation 2 in Section 2.1. We propose to construct a vocabulary forest that covers the entire urban area with numerous shallow vocabulary trees, with each tree covering an independent region. Doing so has the following benefits compared to constructing a monolithic tree that covers the entire urban area. First, it significantly reduces the search latency, as the tree size goes down monotonically with the tree coverage size. Second, finer-grained segmentation in tree coverage effectively removes false positives in image matching, as will be shown in Section 4. Third, unifying the vocabulary tree coverage enables stable content transmission and computation across all regions, as compared to arbitrary segmentation, e.g., the shopping mall [15] or the entire street. Constructing a vocabulary forest ahead of time is the prerequisite of runtime on-device image matching and AR effects rendering, which consists of the following steps.

**Image collection** The image matching accuracy highly depends on the collection of visual data. For example, the user's camera could locate at either the left-hand side or right-hand side and could be close or far from the building wall. As a result, those differences in user's camera positions can incur distortions or occlusions on the building. In conclusion, the images collected should cover multiple angles and distances that the user's camera could position [26]. Empirically, to our best knowledge, collecting around ten images for one-side of the building is sufficient, and it can derive a higher than 90% top-1 accuracy out of nine POI candidates, as will be shown in our preliminary results in Section 4.

**Outlier removal** Typically, the street view images collected often contain irrelevant objects in dynamic environments, e.g., persons and automobiles passing by, and thus will usually incur a negative impact on image matching accuracy. To remove such impacts, the system could make full uses of existing state-of-the-art general object detectors, e.g., YOLO [23], to peel off the outlier features from irrelevant entities after the street images collection. Besides, the outlier removal also respects the privacy of persons or automobiles being captured during data collection [38], which may contain sensitive information such as human faces and car license plates.

**Vocabulary forest and database construction** On the cloud server, the system vendor constructs a geo-visual database that relates the geolocation with the image features. With the image collected and outliers removed, the system vendor groups the image collections based on geolocations, extracts per tree coverage image features, and constructs the vocabulary trees corresponding to each GPS range. During online image matching, the relevance of the camera image and the database image is determined by the similarity of the feature search paths down the vocabulary tree from two images [7, 21], and the database image with the highest relevance score is considered as the top match. Upon finding the best match with the smartphone's input image, the system will derive the POI label, e.g., the Starbucks on Main Street, as a reference for related POI AR effects, e.g., a virtual overlay aligned on a wall with current coffee promotions, for the user.

**Database maintenance** In the real world, outdoor environments change sporadically in urban areas. The system vendor needs to regularly update the POI information and corresponding image features in the database. Furthermore, computer vision algorithms are usually sensitive to the variation of seasons, weather, and lighting

conditions, calling for more diverse image feature collections under different conditions.

## 3.2 Incremental Tree Prefetch

Prefetching all nearby vocabulary trees may incur meaningless network resource waste, e.g., vocabulary trees downloaded are never queried later. To reduce unnecessary network traffics, we propose to prioritize the prefetch sequence and predict the user trajectory.

**Prioritizing prefetch** We propose to prioritize the prefetch sequence based-on the overlapping region size between the GPS error range and nearby tree coverage. Due to the uncertainty of GPS positioning in urban areas, the exact user location could be within a circle with a radius of nearly 30m, the GPS error range in urban areas reported in prior work [4, 15]. As a result, the error circle will overlap with the nearby tree coverage by different sizes. The system prioritizes the tree prefetch by downloading the trees that have more overlapped region with the GPS error circle, i.e., the user is more likely to appear in this region, and ignores the trees with overlapped regions smaller than a pre-defined threshold, e.g., 10%, indicating that the user is less likely to appear in that region without further movements.

**Predicting user trajectory** We propose to prefetch the vocabulary tree by predicting the short-term user trajectory based on the crowdsourced digital footprints. In densely populated urban areas, users enjoy sharing their digital footprints on social media while visiting different places of interest [34]. The POI AR effects are easy to go viral on short video sharing apps like TikTok and thus attract more and more users to visit. The system prioritizes the tree prefetch by downloading the trees covering more popular POIs.

**Adaptive tree replacement** Continuous vocabulary tree downloads stress the device storage space, and we propose to replace the downloaded trees based-on the Least Recently Used (LRU) policy. The system caches the trees with more frequent recent visits and removes the trees that have not been visited for a long time. It not only avoids repetitive downloads but also effectively removes the contents that are unlikely to be queried in the near future.

## 3.3 Adaptable Camera Positioning

We propose to utilize domain knowledge to provide smoother AR effects and user experiences towards various AR types under different real-world environments. As mentioned in prior sections, there are a set of parameters that could be tuned, e.g., the video resolution, the frame rate, and the number of features per frame, and different choices of parameter values will result in different positioning accuracy and latency. The above parameters have different impacts on the positioning accuracy. For example, for two consecutive frames, more extracted feature points rectify false correspondences, and a higher sampling rate smooths camera movements and avoids dramatic shifts of correspondent feature points, both contribute to higher positioning accuracy.

**Adaptation to different types of AR** We propose to explore the trade-off space by profiling the positioning accuracy and latency under different set of knob values to meet the requirement of each AR type, as shown in Table 1. Upon recognizing the POI, the system picks up a set of parameter values profiled ahead-of-time, catering to a specific AR type. For example, for markerless AR that renders an interactive 3D object in free space, the accuracy required is ∼1m, so that SLAM could relax the parameter values to support smoother 3D object rotations and movements; for superimposition AR that provides an alternate view of an entity, the system should choose higher parameter values to control the error within a few centimeters and naturally align the AR object with real-world entities.

| Methods | No. Images | Generation time | Storage cost |
|---|---|---|---|
| Geo-visual | ∼ 100K [26] | ∼ 12$s$/building | ∼ 0.1MB/building |
| 3D City | ∼ 10M [27] | ∼ 4hrs/building | ∼ 2MB/building |
| Ratio | ∼ 1:100 | ∼ 1:1200 | ∼ 1:20 |

**Table 3: A head-to-head comparison of constructing a geo-visual database vs a 3D city model. The former incurs significantly lower cost than the latter.**

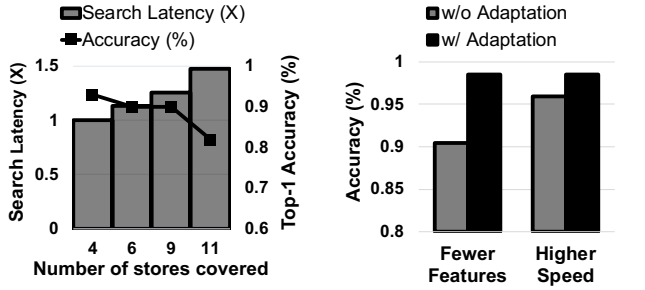
**Adaptation to various environments** Under real-world environments, the conditions sometimes could be unpredictable, e.g., the building texture or the camera moving speed. We propose to adapt the camera pose estimation to various environments automatically to meet the accuracy and latency goals. For example, upon detecting a dramatic drop in the number of feature points detected, the system automatically boosts the image resolution or the frame rate to offset the potential accuracy loss. Due to the elasticity of SLAM, as will be shown in Section 4, the positioning accuracy only drops slightly with higher image resolutions and frame rates, even if the number of feature points drops 40%.

# 4 PRELIMINARY RESULTS

We carried out several preliminary experiments that validate our key designs. The measurements are benchmarked by a few well-known computer vision libraries, including DBoW2 [7]: a vocabulary tree library; ORB_SLAM2 [20]: a real-time SLAM library; and colmap [27]: a general-purpose 3D reconstruction pipeline.

**Ahead-of-time construction cost comparison** Vocabulary forest construction is significantly cheaper than the 3D city model reconstruction, as shown in Table 3. For example, constructing the geo-visual database for a small town like Palo Alto in CA (≈ 26 miles$^2$) needs roughly 100K images [26]. On the contrary, constructing a 3D model for a single building requires ∼2K images taken from different relevant locations and angles [27]. To reconstruct 3D Palo Alto, the total number of images could easily go up to 10M [8], ∼100× more efforts in data acquisition. With sufficient images, the construction time for a single 3D building takes another four hours [27], nearly 1200× more expensive than constructing vocabulary trees. 3D reconstruction also incurs 20× more storage costs by storing RGB-D voxels [27] compared to feature vectors [19].

**Vocabulary forest vs monolithic tree** Smaller tree coverage speeds up image matching and reduces false positive rates. We collected our micro dataset on eleven stores (∼100m) on University Avenue, Palo Alto, CA, and each store has around ten photos taken




(a) Larger tree coverage incurs extra search overhead and accuracy drop.

(b) SLAM preserves high accuracy with the adaptation of parameters under different environments.

**Figure 2: Preliminary Results**

from different relative locations and angles. In the evaluation, every single image is excluded from the entire dataset *once* for cross-validation. We set the tree depth (L) as 5, and varies branching factor (K) between 9-11 to fit the image numbers in each coverage (~2000 features per image, $k^L$ total leaf nodes [26]). As shown in Figure 2(a), by enlarging the coverage from 4 stores (30m) to 11 stores (100m), the search latency goes up more than 40%, while the top-1 image matching accuracy drops from 0.93 to 0.82.

**Prefetch can save up to a few seconds** Transferring visual data on-the-fly incurs significant extra overhead. As a result, under the typical network bandwidths[1] provided by the measurement results from prior work [10, 13, 28], downloading a monolithic vocabulary tree with tens of GB will completely ruin user experiences. By confining the vocabulary tree coverage to the GPS error range (~30m), the tree size is nearly 1MB, and downloading these trees only incurs 1.01s and 1.2s overhead under the typical LTE and WiFi network bandwidths, respectively.

**Adaptable camera positioning** SLAM can adapt to diverse environments. We emulate different real-world environments on The KITTI Vision Benchmark Suite [30], whose video data is captured with resolution of 1241×376 at the frame rate of 10 by a monocular visual odometry. The positioning results measured on the highest parameter values provided by the dataset is considered as the ground truth. We picked the resolution of 930×282 (resized to 75%), the frame rate of 6, and the feature number of 1.5K per image as our test setup. For example, as shown in Figure 2(b), assume that the feature points extracted drops 40% on smooth building textures, the measured positioning accuracy drops 9.6%. In this case, raising the camera capturing resolution to 1241×376 will promote the positioning accuracy to 98.0%. Another example is a user moving in a 3× faster speed, i.e., missing 2 frames in every 3 frames compared to the default test setup. As a result, the positioning accuracy drops 4%. Accordingly, elevating the resolution to 1241×376 and extracting 500 more feature points on each frame will promote the positioning accuracy to 98.5%. Co-tuning above parameter values still performs real-time camera pose estimation.

[1]WiFi: downlink 7040kbps, RTT 50ms; LTE: downlink 9185kbps, RTT 70ms.

## 5 RELATED WORK

**Assisting GPS with various inertial sensors** Prior work has proposed several hybrid approaches to enhance outdoor GPS accuracy by utilizing low-power inertial sensors [4, 5]. However, they still incur significant errors and are unable to support POI AR. WiFi-based localization [16, 17] achieves centimeter-level errors, but this technique is unable to derive real-time 6DoF camera positions. Although fusing 3DoF WiFi localization and 3DoF orientation estimation from gyroscopes is practical [1], the technique is mostly limited to indoor environments.

**Image feature-based place recognition** City-scale place recognition [26] emerged more than a decade ago, with the advance of vocabulary tree [21] based feature [19] extraction, indexing, storage, and searching. Prior work [6, 15, 31] proposed various ways to reduce online query time by fully/partially offloading the recognition task to the cloud, which missed the opportunity of on-device computation that avoids potential severe private data leakage. To reduce memory usage, Hedau *et al.* [12] proposed to download pre-trained random forest classifiers on-the-fly based on the user's rough GPS location. This technique also avoids uploading the camera image to the cloud. However, to support the place recognition in the entire urban area, above work requires significant extra effort in classifier training. Besides, knowing only the POI label is not sufficient for AR applications relying on the real-time estimation of camera poses.

**SLAM-based AR solutions** SLAM algorithms are the core of ARKit [3], ARCore [9], and a wide range of real-time AR solutions [18, 22]. Prior work [14] augmented SLAM by inertial sensors to enable higher positioning accuracy. Hashemifar *et al.* [11] augmented SLAM with WiFi for higher accuracy and lower latency. All above approaches standalone cannot support POI AR apps whose POI information is the prerequisite of rendering relevant AR effects.

**Emergence of 5G** The emergence of 5G will benefit our system by enabling more downloads of vocabulary trees along the user trajectory with nearly no transmission delay. However, 5G still cannot root out potential privacy leaks while uploading the user images to the cloud server.

## 6 CONCLUSION & ONGOING WORK

The increasing prevalence of POI mobile AR applications presents a critical opportunity for a more practical large-scale urban localization system. We propose to construct an inexpensive localization pipeline with image feature matching and SLAM in tandem. With three key design points discussed in Section 3, we can significantly cut down the ahead-of-time construction cost, reduce online user wait time, and adapt the camera positioning to various AR types and real-world environments.

We are now prototyping our system according to the key designs detailed in Section 3 and will thoroughly evaluate our system with larger image collections.

## ACKNOWLEDGMENTS

The authors from Purdue were supported in part by NSF Award 1846102, 1718702, and a Google Faculty Award.

## REFERENCES


[1] Mary Alatise and Gerhard Hancke. 2017. Pose Estimation of a Mobile Robot Based on Fusion of IMU Data and Vision Data Using an Extended Kalman Filter. Sensors 17, 10 (Sep 2017), 2164. https://doi.org/10.3390/s17102164

[2] Cloud Security Alliance. 2010. Top Threats to Cloud Computing V1.0. https://cloudsecurityalliance.org/topthreats/csathreats.v1.0.pdf.

[3] Apple. 2019. ARKit. https://developer.apple.com/augmented-reality.

[4] Cheng Bo, Xiang-Yang Li, Taeho Jung, Xufei Mao, Yue Tao, and Lan Yao. 2013. SmartLoc: Push the Limit of the Inertial Sensor Based Metropolitan Localization Using Smartphone. In Proceedings of the 19th Annual International Conference on Mobile Computing and Networking (MobiCom '13). ACM, New York, NY, USA, 195–198. https://doi.org/10.1145/2500423.2504574

[5] I. Constandache, R. R. Choudhury, and I. Rhee. 2010. Towards Mobile Phone Localization without War-Driving. In 2010 Proceedings IEEE INFOCOM. 1–9. https://doi.org/10.1109/INFCOM.2010.5462058

[6] U. Drolia, K. Guo, J. Tan, R. Gandhi, and P. Narasimhan. 2017. Cachier: Edge-Caching for Recognition Applications. In 2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS). 276–286. https://doi.org/10.1109/ICDCS.2017.94

[7] Dorian Gálvez-López and J. D. Tardós. 2012. Bags of Binary Words for Fast Place Recognition in Image Sequences. IEEE Transactions on Robotics 28, 5 (October 2012), 1188–1197. https://doi.org/10.1109/TRO.2012.2197158

[8] Geographic.org/streetview. 2019. List of Street Names in Palo Alto, California, Maps and Street Views. https://geographic.org/streetview/usa/ca/palo$_a$lto.html.

[9] Google. 2019. ARCore. https://developers.google.com/ar.

[10] Yihua Ethan Guo, Ashkan Nikravesh, Z. Morley Mao, Feng Qian, and Subhabrata Sen. 2017. Accelerating Multipath Transport Through Balanced Subflow Completion. In Proceedings of the 23rd Annual International Conference on Mobile Computing and Networking (MobiCom '17). ACM, New York, NY, USA, 141–153. https://doi.org/10.1145/3117811.3117829

[11] Zakieh S. Hashemifar, Charuvahan Adhivarahan, Anand Balakrishnan, and Karthik Dantu. 2019. Augmenting Visual SLAM with Wi-Fi Sensing For Indoor Applications. CoRR abs/1903.06687 (2019). arXiv:1903.06687 http://arxiv.org/abs/1903.06687

[12] Varsha Hedau, Sudipta Sinha, C. Lawrence Zitnick, and Richard Szeliski. 2012. A Memory Efficient Discriminative Approach for Location Aided Recognition. In Proceedings of the 1st Workshop on Visual Analysis and Geo-Localization of Large-Scale Imagery (in conjunction with ECCV 2012) (proceedings of the 1st workshop on visual analysis and geo-localization of large-scale imagery (in conjunction with eccv 2012) ed.). Springer Verlag.

[13] Junxian Huang, Feng Qian, Yihua Guo, Yuanyuan Zhou, Qiang Xu, Z. Morley Mao, Subhabrata Sen, and Oliver Spatscheck. 2013. An In-depth Study of LTE: Effect of Network Protocol and Application Behavior on Performance. In Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM (SIGCOMM '13). ACM, New York, NY, USA, 363–374. https://doi.org/10.1145/2486001.2486006

[14] Shahram Izadi, David Kim, Otmar Hilliges, David Molyneaux, Richard Newcombe, Pushmeet Kohli, Jamie Shotton, Steve Hodges, Dustin Freeman, Andrew Davison, and Andrew Fitzgibbon. 2011. Kinectfusion: real-time 3D reconstruction and interaction using a moving depth camera. In In Proc. UIST. 559–568.

[15] Puneet Jain, Justin Manweiler, and Romit Roy Choudhury. 2015. OverLay: Practical Mobile Augmented Reality. In Proceedings of the 13th Annual International Conference on Mobile Systems, Applications, and Services (MobiSys '15). ACM, New York, NY, USA, 331–344. https://doi.org/10.1145/2742647.2742666

[16] Manikanta Kotaru, Kiran Joshi, Dinesh Bharadia, and Sachin Katti. 2015. SpotFi: Decimeter Level Localization Using WiFi. SIGCOMM Comput. Commun. Rev. 45, 4 (Aug. 2015), 269–282. https://doi.org/10.1145/2829988.2787487

[17] Hongbo Liu, Yu Gan, Jie Yang, Simon Sidhom, Yan Wang, Yingying Chen, and Fan Ye. 2012. Push the Limit of WiFi Based Localization for Smartphones. In Proceedings of the 18th Annual International Conference on Mobile Computing and Networking (Mobicom '12). ACM, New York, NY, USA, 305–316. https://doi.org/10.1145/2348543.2348581

[18] H. Liu, G. Zhang, and H. Bao. 2016. Robust Keyframe-based Monocular SLAM for Augmented Reality. In 2016 IEEE International Symposium on Mixed and Augmented Reality (ISMAR). 1–10. https://doi.org/10.1109/ISMAR.2016.24

[19] D. G. Lowe. 1999. Object recognition from local scale-invariant features. In Proceedings of the Seventh IEEE International Conference on Computer Vision, Vol. 2. 1150–1157 vol.2. https://doi.org/10.1109/ICCV.1999.790410

[20] Montiel J. M. M. Mur-Artal, Raúl and Juan D. Tardós. 2015. ORB-SLAM: a Versatile and Accurate Monocular SLAM System. IEEE Transactions on Robotics 31, 5 (2015), 1147–1163. https://doi.org/10.1109/TRO.2015.2463671

[21] David Nister and Henrik Stewenius. 2006. Scalable Recognition with a Vocabulary Tree. In Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Volume 2 (CVPR '06). IEEE Computer Society, Washington, DC, USA, 2161–2168. https://doi.org/10.1109/CVPR.2006.264

[22] Jarkko Polvi, Takafumi Taketomi, Goshiro Yamamoto, Arindam Dey, Christian Sandor, and Hirokazu Kato. 2016. SlidAR: A 3D positioning method for SLAM-based handheld augmented reality. Computers & Graphics 55 (2016), 33–43.

[23] Joseph Redmon and Ali Farhadi. 2016. YOLO9000: Better, Faster, Stronger. arXiv preprint arXiv:1612.08242 (2016).

[24] Tilman Reinhardt. 2019. Using Global Localization to Improve Navigation. https://ai.googleblog.com/2019/02/using-global-localization-to-improve.html

[25] Adi Robertson. 2019. Facebook says it will build AR glasses and map the world. https://www.theverge.com/2019/9/25/20883706/facebook-ar-glasses-prototypes-live-maps-announce-oc6.

[26] G. Schindler, M. Brown, and R. Szeliski. 2007. City-Scale Location Recognition. In 2007 IEEE Conference on Computer Vision and Pattern Recognition. 1–7. https://doi.org/10.1109/CVPR.2007.383150

[27] Johannes Lutz Schönberger and Jan-Michael Frahm. 2016. Structure-from-Motion Revisited. In Conference on Computer Vision and Pattern Recognition (CVPR).

[28] Joel Sommers and Paul Barford. 2012. Cell vs. WiFi: On the Performance of Metro Area Mobile Connections. In Proceedings of the 2012 Internet Measurement Conference (IMC '12). ACM, New York, NY, USA, 301–314. https://doi.org/10.1145/2398776.2398808

[29] Sturfee. 2019. Sturfee. https://sturfee.com/.

[30] The KITTI Vision Benchmark Suite. 2012. Visual Odometry / SLAM Evaluation 2012. http://www.cvlibs.net/datasets/kitti/eval$_o$dometry.php.

[31] Gabriel Takacs, Vijay Chandrasekhar, Natasha Gelfand, Yingen Xiong, Wei-Chao Chen, Thanos Bismpigiannis, Radek Grzeszczuk, Kari Pulli, and Bernd Girod. 2008. Outdoors Augmented Reality on Mobile Phone Using Loxel-based Visual Feature Organization. In Proceedings of the 1st ACM International Conference on Multimedia Information Retrieval (MIR '08). ACM, New York, NY, USA, 427–434. https://doi.org/10.1145/1460096.1460165

[32] Scape Technologies. 2019. Hyper-accurate location, powered by computer vision. https://scape.io/.

[33] A. Toshev, B. Taskar, and K. Daniilidis. 2010. Object detection via boundary structure segmentation. In 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. 950–957. https://doi.org/10.1109/CVPR.2010.5540114

[34] Z. Yu, H. Xu, Z. Yang, and B. Guo. 2016. Personalized Travel Package With Multi-Point-of-Interest Recommendation Based on Crowdsourced User Footprints. IEEE Transactions on Human-Machine Systems 46, 1 (Feb 2016), 151–158. https://doi.org/10.1109/THMS.2015.2446953

[35] A. Zafft and E. Agu. 2012. Malicious WiFi networks: A first look. In 37th Annual IEEE Conference on Local Computer Networks - Workshops. 1038–1043. https://doi.org/10.1109/LCNW.2012.6424041

[36] Amir R. Zamir, Asaad Hakeem, Luc Van Gool, Mubarak Shah, and Richard Szeliski. 2016. Introduction to Large-Scale Visual Geo-localization. Springer International Publishing, Cham, 1–18. https://doi.org/10.1007/978-3-319-25781-5$_1$

[37] Fan Zhang, Wenbo He, Xue Liu, and Patrick G. Bridges. 2011. Inferring Users' Online Activities Through Traffic Analysis. In Proceedings of the Fourth ACM Conference on Wireless Network Security (WiSec '11). ACM, New York, NY, USA, 59–70. https://doi.org/10.1145/1998412.1998425

[38] L. Zhang, X. Li, K. Liu, C. Liu, X. Ding, and Y. Liu. 2019. Cloak of Invisibility: Privacy-Friendly Photo Capturing and Sharing System. IEEE Transactions on Mobile Computing 18, 11 (Nov 2019), 2488–2501. https://doi.org/10.1109/TMC.2018.2878711