GLAMAR: Geo-Location Assisted Mobile Augmented Reality for Industrial Automation

Mostafa Uddin[§], Sarit Mukherjee, Murali Kodialam, TV Lakshman
Nokia Bell Labs
fastname.lastname@nokia-bell-labs.com

2020 IEEE/ACM Symposium on Edge Computing (SEC) | 978-1-7281-5943-0/20/$31.00 ©2020 IEEE | DOI: 10.1109/SEC50012.2020.00036

***Abstract*—Mobile Augmented Reality (MAR) is going to play an important role in industrial automation. In order to tag a physical object in the MAR world, a smart phone running MAR-based applications must know the precise location of an object in the real world. Tracking and localizing a large number of objects in an industrial environment can become a huge burden for the smart phone due to compute and battery requirements. In this paper we propose GLAMAR, a novel framework that leverages externally provided geo-location of the objects and IMU sensor information (both of which can be noisy) from the objects to locate them precisely in the MAR world. GLAMAR offloads heavy-duty computation to the edge and supports building MAR-based applications using commercial development packages. We develop a regenerative particle filter and a continuously improving transformation matrix computation methodology to dramatically improve the positional accuracy of objects in the real and the AR world. Our prototype implementation on Android platform using ARCore shows the practicality of GLAMAR in developing MAR-based applications with high precision, efficiency, and more realistic experience. GLAMAR is able to achieve less then 10cm error compared to the ground truth for both stationary and moving objects and reduces the CPU overhead by 83% and battery consumption by 80% for mobile devices.**

***Index Terms*—Mobile Augmented Reality, Localization, Tracking**

I. Introduction

The next wave of digitization promises to revolutionize the future industry, ushering in widespread automation and augmented intelligence in the manufacturing and service sectors [1, 2]. Augmented reality (AR) is going to play a significant role in this transformation [3, 4, 5]. An application built using AR enhances the perception of the real world view of a user by overlaying virtual information on real objects of interest that are within the field of view (FoV) of the user when seen through an AR-enabled device. AR-based applications require large amount of image and sensory data gathering and manipulation by the device in real-time. This severely limits their adoption in the industry floor as special purpose and expensive devices are needed. For widespread adoption of such applications, it is desirable to have them run on popular devices such as smart phones so that anyone carrying a phone can use such applications. However, smart phone based AR, popularly known as mobile augmented reality (MAR) [6, 7, 8], has some practical limitations due to the constrained capabilities of the devices. This paper proposes a framework for efficient support of MAR on smart phones by leveraging the enhanced facilities installed in future industry environment.

A. Motivating Use Cases

MAR-based applications for industrial automation will open the door for large scale application development and adoption. These applications will allow a user to observe the physical environment through the camera of the smart phone. Images of physical objects of interest (we refer to them as target objects in this paper) will be augmented on the screen to help the user identify them, inspect and control them in the virtual world and the effect of which will be carried out in the physical world in real-time. Below we provide a few simple, but illustrative, use cases that can be used in future industry:

In a **digital warehouse** shipping and receiving items in boxes are sorted using smart conveyor belts as shown in Figure 1. Automation in warehousing and logistics requires a mission-critical system that allows high-fidelity digital representations of the physical world and the ability to dynamically control physical assets for improving safety, productivity, and efficiency. MAR can help in this process by identifying a box (a target object) on the belt. When the object is viewed through the camera of the phone, the image of the box is augmented with labels and internal contents displayed on the screen in real-time (as shown in Figure 1).

Fig. 1: Digital warehouse with on premise edge computation and per box sensory tags. In addition, MAR view of the warehouse conveyor belt with boxes tagged/classified using color.

On an **indoor industrial floor**, an operator looking at a swarm of identical robots wants to identify a robot that is

[§]This manuscript was submitted while the author was with Nokia Bell Labs. Currently the author is affiliated with Facebook Inc.

978-1-7281-5943-0/20/$31.00 ©2020 IEEE
DOI 10.1109/SEC50012.2020.00036

performing a certain task of interest. The robot's image is marked up with identifiable information when it comes in the view finder of the camera of the operator's phone. It displays controlling information about the robot for the operator to control the same in real-time (e.g., move the robot to a particular place on the factory floor by dragging its image on the phone's screen).

In an **outdoor mining industry** there are a number of autonomous trucks that carry ore from the pit to the train track. If any truck in the convoy stops, the whole pipeline may come to a standstill, till the problem is addressed and the stalled truck starts moving. Therefore, real-time health checking of the trucks is needed for efficient operation. Using a MAR-based application, when a smart phone is held up, the trucks in the view are tagged with information about their health and various other vital readings.

B. Challenges in Supporting MAR for Industrial Automation

In order to support the above use cases, a MAR-based application must render virtual content attached with a target object when the object appears in the device's FoV. This requires the application to be aware of the precise position and stable position update of the target object in the real world so that the user gets realistic AR experience on interactions with the virtual content. AR applications typically use state-of-the-art vision based techniques to recognize and track target objects (both static and moving) in camera's FoV [9, 10, 8, 11, 12, 13]. Such techniques require recognition of objects appearing in camera frame and tracking procedures running at the user AR device. As these procedures are compute intensive and put huge demand on battery, they are not suitable to run on commercial smart phones for MAR applications. Therefore, they are not able to accurately position the target objects in real-world. A recent work [14] developed a scheme to infer 3D position of the target object from the recognized 2D view frame. However, such techniques are restricted to only planar objects (e.g., Jaguar [14]), which most of the real-world objects of interest are not. On the other-hand feature-based techniques (i.e., 2D-to-3D correspondences) [15, 16, 17] have been used for estimating the position of regular objects, but it requires actual 3D model of the target objects. Therefore, a repository of models have to be created and maintained in the device, which is not very practical for MAR.

Another approach is to leverage 3D point cloud, extracted using a depth sensing camera or a stereo camera, to recognize the target objects [18, 19]. Depth sensing cameras are rare in smart phones today, and when available in expensive models, become a drain on the battery due to heavy computation needs. Furthermore 3D points are limited by certain range as well [20, 21, 22].

Moreover, the above computer vision based algorithms are limited by the distance between the camera and the target object. Figure 2 shows the accuracy of detecting the position of a 2D marker as the camera moves away from it. Observe that for a fixed size marker, after certain distance the accuracy sharply drops to 0%.

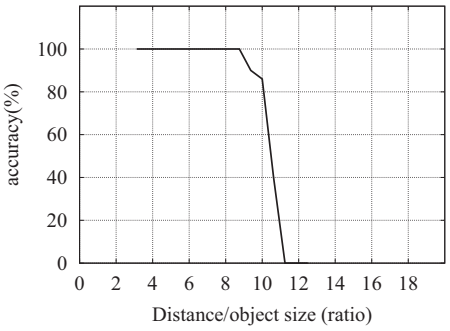

Fig. 2: Accuracy of pose estimation of a 0.15x0.15m^2 marker object at different perpendicular distance from the camera of Pixel 4 with an image resolution of 1920x1080.

In addition to the above limitations, vision-based techniques are prone to object occlusion, and are not able to distinguish similar looking objects [23]. For instance, for industrial automation, we need to uniquely identify a target object so that we can have a specific action (i.e., navigating a robot). Unfortunately, visual recognition techniques cannot provide this functionality. Furthermore, vision based object tracking also require camera calibration which can be erroneous.

C. Our Approach

Our goal is to appropriately leverage the future industry infrastructure to support MAR-based application more efficiently on smart phones. Future industry in the Industry 4.0 era will be equipped with next generation technology and services, such as positioning system (indoor or outdoor) to track the objects of interest, edge cloud to offload computation [24, 25, 26], low latency networking to edge cloud [27], to name a few. For example, 5G for enterprise [24], Bluetooth 5 [28], FiRa (fine-ranging) [29] are geared towards precise asset tracking and localization in enterprise. Real-time kinematic (RTK) with Global Navigation Satellite Systems (GNSS) for precise localization in outdoor mining environment is already in use [30]. Private 5G and its precursor private LTE networks [25] are poised to provide both edge computation and low latency networking.

For the MAR-based applications described above, we make use of the external geo-location, instead of on-device target object location tracking, to offload the burden on the smart phones. This makes MAR-based applications viable to running on smart phones without requiring high processing capacity or draining battery. Despite the requirement of infrastructure support, there are several other advantages of using an external geo-location. It can handle both stationary and mobile objects, and does not suffer from object occlusion. The separation between the user and the target object can be any arbitrary distance, and is not constrained by the limitation of the vision-based algorithms [31]. The location computation is further offloaded into the edge cloud, and streamed to the smart phone

in real-time using low latency networking, making it attractive for MAR-based applications.

While it is possible to get the location of a target object in real-time, the precision of the object's position can fluctuate over time and/or get affected by interference from nearby radio sources [32] at industrial premise. Figure 5 and 13(b) explain the scenario for stationary and moving objects, respectively. Figure 5 shows the variation of error for a stationary object, and Figure 13(b) shows that the moving object's trajectory (green) does not always follow the exact path taken by the object (red). The fluctuation makes the tagging of a target object in a MAR-based application inaccurate and its usability falters.

D. Our Contribution: GLAMAR

In this paper, we design and develop an edge computation based solution, called Geo-Location Assisted Mobile Augmented Reality (GLAMAR), that provides a runtime framework for supporting AR-based application on commercial smart phone platform in industrial settings. We use High Accuracy Indoor Positioning (HAIP) [33] system for locating and tracking an object. HAIP attaches a sensory tag to an object to locate it using Bluetooth Low Energy (BLE) mechanism. We further augment the tag with IMU sensors, and also stream the IMU sensor data to the edge cloud. The overall environment with respect to the digital warehouse use case is shown in Figure 1. We develop a novel event-triggered, regenerative particle filter-based algorithm with the sensory data that improves the accuracy of object's tracking in real-time. All these computations are carried out at the edge and final position information is streamed to the user's phone, making the overall framework MAR friendly. To the best of our knowledge, GLAMAR is the first of its kind to support AR-based application for tracking real-world object in MAR platform, without using any heavy-duty vision-based techniques.

GLAMAR supports MAR-applications developed using ARCore [34] or ARKit [35] (based on the phone platform) while not performing any extra vision-based computation for target object recognition and tracking at the phone. This makes GLAMAR light-weight and scalable for tracking large number of moving objects in real-time. The MAR application running on the phone uses its AR coordinate system, while target object's positions are based on a coordinate system overlaid on the industry premise. In order to tag target objects with virtual content in the phone's AR coordinate system a transformation matrix is needed. We develop a lightweight method to run locally in user's phone to compute such matrix, and continually improve it during the lifetime of the MAR session. This makes GLAMAR scalable to support large number of AR users, which is ideal for industrial settings. It also preserves the location privacy of the user within the phone.

Most of the recent vision-based edge-assisted MAR system [36] [37] [9] [8] uses camera pixel position of the target objects to render the virtual contents. Therefore, such system cannot understand the depth of the target objects, which makes rendering confusing for the target objects which are behind any physical occlusion. GLAMAR allows to render the virtual contents using MAR SDK (e.g., ARCore, ARKit), which uses depth map to make virtual objects accurately appear in front of or behind real world objects, enabling immersive and realistic user experiences.

For our evaluation, we build the infrastructure for the GLAMAR framework mimicking an industrial setting. Based on our evaluation, GLAMAR is able to achieve high accuracy object tracking (less then 10cm error compared to the ground truth) for both stationary and moving objects. The result shows that compared to feature-based vision techniques, the framework reduces the CPU overhead by 83% and battery consumption by 80% for mobile devices. The major contributions we made in this paper are:

(a) Show how MAR-based industrial applications can be made practical and easily adoptable. The application demands on smart phone resources are kept very low by leveraging the new industry 4.0 technologies like object localization, edge computation, low latency networking that are getting deployed in the industry premises.

(b) Develop an edge computation friendly framework, GLAMAR, that determines and distributes location of target objects in real-time to all MAR-based applications. Our edge-hosted regenerative particle filter-based location estimation, and on-device continuously improving coordinate transformation matrix computation (between AR and premise coordinates) ensure accurate image augmentation even in the presence of errors and fluctuations. While computation heavy operations are executed at the edge, the lightweight on-device coordinate transformation keeps user information private within the device itself.

(c) Implement MAR-based applications using commercial SDKs for demonstrating the significant advantages provided by GLAMAR compared to legacy vision-based techniques. For accurate comparison, we develop a novel ground truth measurement mechanism to track target objects in real world units (e.g., meter) instead of pixels, which most of the literature works in MAR use.

II. GLAMAR Framework

The framework consists of two main GLAMAR components, and an independent Location Service, as shown in Figure 3: (i) GLAMAR agent running in user mobile device, and (ii) GLAMAR service running at the industry premise edge server. There could be one or more user devices with each device running an agent. Furthermore, each device runs its own independent AR session (that creates its own AR coordinate system). The agent helps the AR session to overlay virtual contents onto the corresponding target objects that are in the current FoV. The target objects could be stationary or moving in the physical world. For instance, it could be a box on a conveyor belt or a robot moving at an industrial premise. We assume that the target objects are equipped with battery powered tags for localization and IMU sensors. The localization tag comes with the HAIP system; however,

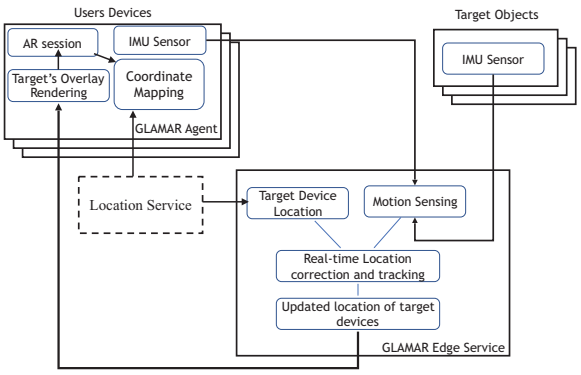

Fig. 3: GLAMAR System Overview.

integration of IMU sensor with the tag and its cost analysis are beyond the scope of this paper. These tags stream IMU sensor information to the GLAMAR service in real-time. GLAMAR service also receives location information, from the indoor/outdoor localization system, about the user devices and the target objects of the surroundings. Before we describe the functionalities of these components, we explain the coordinate system used by different entities in the GLAMAR framework, and their inter-relationship so that ultimately target objects and their corresponding virtual annotations can be displayed on the user device FoV.

A. Coordinate Systems

We deal with four independent coordinate systems as follows:

AR coordinate system: As the camera of the user device moves, it uses the visual features of the surrounding environment to build an AR coordinate system for the AR session [38, 39]. All objects, real or virtual, must be represented in this coordinate system for viewing on user device. The origin of this coordinate system remains at the starting point of the AR session. Note that, every user device has it's own independent AR session, which consists of an AR coordinate system that has `Y`-axis along the opposite direction of gravity, and `X`, `Z`-axes create the horizontal plane. As the user device moves around, the AR session sometimes may readjust it's coordinate system based on the updated visual features [40]. Also, long non-presence of visual features forces the user device to initiate a new AR session with completely new AR coordinate system.

Phone coordinate system: This is a three dimensional coordinate system with 6DoF (degree of freedom). All IMU sensor readings are expressed in this coordinate system [41]. This coordinate system is relative to the device's screen when the device is held in its default orientation (i.e., user holding the phone to see the screen). In its default orientation, the X axis is horizontal and points to the right, the Y axis is vertical and points up, and the Z axis points toward the outside of the screen surface.

Premise coordinate system: In GLAMAR, we get the location input of a target object based on the positioning system available at the industry premise. Such positioning system has its own point of reference or coordinate system, which we refer to as the premise coordinate system. This coordinate system helps in representing the location of an object in 2 or 3 dimension. In 2-dimensional case, both axes (X and Y) form a horizontal plane, parallel to the earth's surface, to represent the location of target objects. In 3-dimensional case, in addition to the horizontal plane, one axis (Z) is perpendicular to the horizontal plane (i.e., opposite of gravity). Any premise coordinate system defined as above can be considered as an input to the GLAMAR framework.

Reference (Earth) coordinate system: This is a 3 DoF world coordinate system on earth's surface plane. In this case, Y-axis is towards global north, X-axis towards east, and Z-axis perpendicular to the earth's surface plane, opposite of gravity. We use this coordinate system as reference to find the orientation of the target device in premise coordinate system.

B. GLAMAR Agent

GLAMAR agent continuously receives real-time location updates of the surrounding target objects from the GLAMAR service in premise coordinate system and corresponding meta data information about the target objects. The agent converts these coordinates to the device's AR coordinate system. The agent is responsible for computing the transformation matrix needed to translate a coordinate from the premise coordinate system to user device's AR coordinate system. Finally, based on the computed AR coordinates it renders the virtual contents using the meta-data for the corresponding target devices that are in the current FoV. Note that by not revealing the user location to the GLAMAR service, the agent preserves the privacy of the user.

Knowledge of both the origin and the orientation of the two coordinate systems are needed to compute the transformation matrix. Even though such information is easily available for the premise coordinate system, it is non-trivial to know the origin and orientation of the AR coordinate system. Furthermore, each agent in user device has different and independent AR coordinate systems. For computing the transformation matrix, GLAMAR agent needs to determine both the translation and rotation matrixes.

Independent of the translation between two coordinate systems, first, we estimate the rotation matrix by coinciding the origin of AR and premise coordinate systems as shown in Figure 4. Both these coordinate systems have one axis aligned with the gravity, which are Y and Z-axis for AR and premise coordinate systems, respectively. Rest of the two axes represent the same horizontal plane. Therefore, GLAMAR agent only needs to compute the rotation to align the X-Z plane of AR with the X-Y plane of premise coordinate system. We derive the rotation by measuring the position of the user device, both in AR and premise coordinate systems, at two distinct locations in the plane as illustrated in Figure 4.

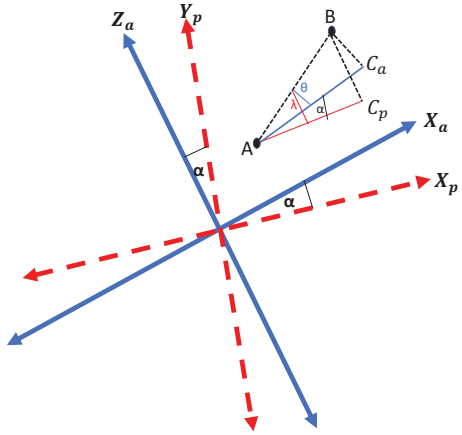

Fig. 4: Rotation estimation between AR and premise coordinate systems.

In Figure 4, X_p-Y_p and X_a-Z_a represent the two horizontal planes of the premise and AR coordinate systems. In the horizontal plane, A and B represent two measurement points of the user device. A and B have the coordinates (xa_1, za_1) and (xa_2, za_2), and (xp_1, yp_1) and (xp_2, yp_2), respectively, in AR and premise coordinate systems. As shown in the figure, $\overline{AC_a}$ and $\overline{BC_a}$ are parallel to X_a and Z_a axis, respectively. Similarly, $\overline{AC_p}$ and $\overline{BC_p}$ are parallel to X_p and Y_p axis, respectively. Thus, the angle of rotation between AR and premise coordinate systems can be derived as follows;

$$\alpha = \lambda - \theta = \arctan(\frac{yp_2 - yp_1}{xp_2 - xp_1}) - \arctan(\frac{za_2 - za_1}{xa_2 - xa_1}) \quad (1)$$

With the angle of rotation (α) determined, the quaternion vector, $q_1 = [0, \sin\alpha/2, 0, \cos\alpha/2]^T$ aligns the X_a-Z_a plane of AR to the X_p-Y_p plane of premise coordinate system. Finally, to align the Z_a axis of AR with the Y_p axis of premise coordinate, we apply following quaternion vector, $q_2 = [\sin 45, 0, 0, \cos 45]^T$. We convert the quaternion to the rotation matrix, and multiply them to derive the rotation matrix for mapping the AR coordinates to premise coordinates.

With the rotation matrix derived, it is easy to compute the translation matrix to shift the origin between the coordinate systems, keeping in consideration of the right-hand rotation rule, which makes the vertical axes align in the opposite direction in the two coordinate systems.

For instance, we apply the rotation on point A to convert its coordinate from AR to premise coordinate (Now assume 3d coordinate). For instance, we can apply following rotation on point A to align its AR location into premise coordinate system.

$$[xp'_1, yp'_1, zp'_1]^T = R_{q1} \times R_{q2} \times [xa_1, ya_1, za_1]^T \quad (2)$$

Then, we deduct the translation vector as follows:

$$[t_x, t_y, t_z] = [xp_1, yp_1, zp_1]^T - [xp'_1, -yp'_1, zp'_1]^T \quad (3)$$

Note that we used negative value for the Y axis, because, despite having the q_2 rotation on Z_a axis, it remains in the opposite direction of Y_p for right-hand rule rotation.

Ideally both coordinate systems should be fixed, therefore, one-time measurement should be enough. However, experiments show that as AR session continues, it sometimes re-adjusts it's AR coordinate based on the surrounding visual features. Further, location measurement (in premise coordinate) could be erroneous (there could be outlier measurement of location). In that case multiple estimations help in improving the accuracy of transformation from one system to another. In Section IV, we show the importance of continuous estimation of coordinate transformation matrix.

C. GLAMAR Service

GLAMAR service running at the edge of the premise is responsible for conducting the background operation of collecting location and IMU sensor information of the surrounding target objects. However, both the location and IMU sensor readings could be erroneous as shown in Figure 5 and 6. The following discusses how we accurately estimate the real-time position of the target objects in premise coordinate system. GLAMAR service shares the tracking information of the target objects with the GLAMAR agents.

1) Positional Events in GLAMAR: There are three types of measurement events that are generated in the framework.

Location Update (L)**:** Location updates give an estimate of the current location in premise coordinates. If Z is the current location then its estimate $Y \sim \mathcal{N}(Z, \sigma_L)$ where σ_L is the covariance matrix and can be estimated from measurements (Figure 5). (We use $\mathcal{N}(a, b)$ to denote a normal distribution with mean a and covariance b).

Acceleration Update (A)**:** The acceleration update gives the current acceleration of the target object. We use sensor fusion technique to measure the orientation of the target objects with respect to Earth coordinate system. Using the knowledge of the orientation of the premise coordinate system with respect to Earth coordinate, we convert the accelerometer reading of target objects to premise coordinate. As in the case of the location update, if A is the true acceleration, then the estimate of the acceleration is $\mathcal{N}(A, \sigma_A)$. As in the case of location update, the covariance matrix can be estimated from measurements (Figure 6). The acceleration measurement is noisy and will in general give non-zero values even if the target object is not moving.

Motion Update (M)**:** Combining the acceleration measurement with gyroscope readings using sensor fusion techniques, it is possible to derive with an extremely high level of confidence whether the target object is stationary or not. The target object sends out a motion update when it detects that it is not in motion. We assume that the motion updates are accurate.

These updates arrive at irregular intervals. Let the times at which the updates arrive at the edge server be denoted by $t_1 \leq t_2 \leq t_3 \ldots$. Each of these update arrival times t_j where $j = 1, 2, 3 \ldots$ is one of these three types of updates. We use $\Delta_j = t_{j+1} - t_j$ to denote the time between updates j and $j+1$. We use Z_{t_j} to denote the location of the target object at time t_j. In order to simplify notation, we use Z_j to denote Z_{t_j} and more generally we use the subscript j to denote t_j. We use V_j and A_j to denote the velocity and acceleration of the target object at time t_j. Assume that the initial location Z_0 is known and we assume that the target object is stationary, i.e, $V_0 = A_0 = 0$.

2) Regenerative Particle Filter: Particle filter [42, 43, 44] is a Bayesian update mechanism to track the location of moving objects. It is a robust alternative to Kalman Filter based tracking of moving objects. A particle filter represents the location of the target object using n virtual particles. Each particle is evolved in time using the update information received at the edge server. The actual location of the target object is estimated as the average of the location of the n particles.

A key component in the particle filter algorithm is the *resampling* step that is performed whenever there is a location update. Resampling the particles eliminates unlikely particles while reinforcing more likely particles. This controls the variance of the location estimator. A natural approach when we get a no motion update is to simply freeze the existing particles until we get an acceleration update. However, this leads to poor location estimation, (see Figure 13(b)&(c)). Therefore we introduce a *regeneration* step when we get a no motion update. In the regeneration step, we first determine the current expected location of the particle and then generate n particles around the current expected location. This regeneration step improves the performance of the particle filter algorithm significantly (see Figure 13(d)). We call this particle filter with the additional regeneration step, the *regenerative particle filter*. We now describe the complete algorithm.

The target object is tracked using a collection of n particles. Larger number of particles improves tracking accuracy at the cost of increased computation. We use X_j^m to denote the estimated location of particle m at time t_j. In addition, we use V_j^m and A_j^m to denote the velocity and acceleration of particle m at time t_j. At the initial time t_0, assume that we have an estimate X_0, V_0, A_0 of the initial target object location, initial velocity and acceleration respectively.

Acceleration Update: Assume that we get an acceleration update A_j at time t_j. In this case we first update the current location of the particles

$$\begin{bmatrix} X_j^m \\ V_j^m \end{bmatrix} = \begin{bmatrix} X_{j-1}^m \\ V_{j-1}^m \end{bmatrix} + \begin{bmatrix} \Delta_{j-1} & \frac{1}{2}\Delta_{j-1}^2 \\ 0 & \Delta_{j-1} \end{bmatrix} \begin{bmatrix} V_{j-1}^m \\ A_{j-1}^m \end{bmatrix}$$

We then generate $A_j^m \sim \mathcal{N}(A_j, \sigma_A)$.

Location Update: Assume that we get a location update Y_j at time t_j. We first update the particle locations using the following update equation.

$$\begin{bmatrix} X_j^m \\ V_j^m \end{bmatrix} = \begin{bmatrix} X_{j-1}^m \\ V_{j-1}^m \end{bmatrix} + \begin{bmatrix} \Delta_{j-1} & \frac{1}{2}\Delta_{j-1}^2 \\ 0 & \Delta_{j-1} \end{bmatrix} \begin{bmatrix} V_{j-1}^m \\ A_{j-1}^m \end{bmatrix}$$

Next we compute the conditional probability

$$w_j^m = \Pr\left[Y_j | X_j^m\right].$$

This probability can be computed since given X_j^m, the actual location of the target object is $\mathcal{N}\left(X_j^m, \sigma_L\right)$. Once we have generated the values of w_j^m for all particles m, we draw n samples with replacement to generate the new X_j^m. We set $A_j^m = A_{j-1}^m$

Motion Update: When we get a motion update at time j, we first compute the estimated location of the target object Z_j as the mean of the particle locations

$$Z_j = \frac{1}{n} \sum_m X_j^m.$$

We now generate a new set of m points $X_j^m \sim \mathcal{N}(Z_j, \sigma_L)$. We now set $V_j^m = 0$ and $A_j^m = 0$ for all particles m. Note that the particles are regenerated around the expected location at this step.

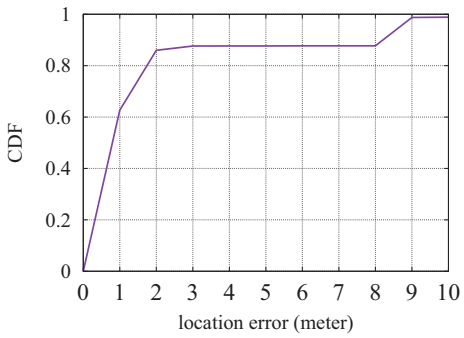

Fig. 5: Distribution of location error of the indoor localization system HAIP.

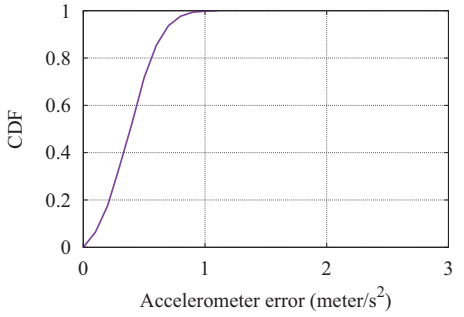

Fig. 6: Distribution of accelerometer sensor reading while target object is static.

III. Implementation

We develop a test-bed for experimenting with GLAMAR in our lab that emulates an industry premise. We equip the premise with indoor localization system called High Accuracy Indoor Positioning [33] (HAIP). Figure 5 shows the distribution of location error of HAIP. As shown in the figure mean error could be close to 1 meter. A number of HAIP locators are fixed on the ceiling of the premise so that one or more locators get line-of-sight measurement of the objects. HAIP is a localization system that provides real-time positioning data using Bluetooth Low Energy (BLE) technology leveraging unique Angle-of-Arrival (AoA) signal processing method. We calibrate HAIP deployment using 2 axes to define the horizontal plane and a vertical axis, and use this as our **premise coordinate system**. We know both the origin and the orientation of the HAIP, which has 215° azimuth with respect to global north (in earth coordinate system). HAIP localization system requires a tag attached to the target object for location tracking. In our implementation, we attach tags to all target objects and user devices. The location of all the tagged target objects in premise coordinate system are streamed to the GLAMAR edge server in real-time.

Target Object: We attach a HAIP tag at each target object. For flexibility and ease of experimentation, we also place an Android phone that plays the role of the IMU sensor with BLE and WiFi transceivers. In real-world deployment any battery powered off-the-shelf IoT device with IMU sensor and BLE/WiFi connectivity can be used. The scope of this paper is not to design such device, but how the IMU reading from such devices could be used for GLAMAR. Therefore, We develop an android application in the phone to emulate the tag for the target objects that applies sensor fusion technique using complementary filter [45] on IMU sensors to estimate the orientation of the target object with respect to reference coordinate system. With the knowledge of the orientation, the app converts its motion sensor data from phone's coordinate system to Earth coordinate system, and finally projects it to the premise coordinate system before streaming it to the GLAMAR edge server.

GLAMAR Agent: We implement the GLAMAR Agent in the user device as an Android App using the ARCore SDK. ARCore API helps in creating an AR session, and provides access to the pose of the user device in AR coordinate. In addition to running an AR session, the app runs a background thread to coordinate the calibration that determines the coordinate transformation matrix from premise to AR coordinate system. Note that, we ignore the vertical axis during the calculation as it is aligned in both coordinate systems.

For coordinate transformation matrix calculation, we need at least 2 recent position measurements of the user device at two distinct locations. In our implementation, we measure the position when the user device is stationary. We use threshold based technique on gyrosscope sensor data to determine if the user device is stationary. If so, the app starts collecting HAIP location data of the user device for 5 seconds. This eliminates any random fluctuation in HAIP location measurement. If the user device moves before 5 seconds, the measurement is discarded. In addition to HAIP data, the app collects the location of user device in AR coordinate. A coordinate calibration thread keeps updating the coordinate transformation matrix whenever it has a new measurement.

The app's activity class also runs a background thread that receives estimated real-time position of the target objects in premise coordinate system from the GLAMAR edge server. The app converts the location from premise to AR coordinate system, except for the vertical axis, using the currently computed coordinate transformation matrix. If the target object falls in the current FoV of the user device, the app renders the virtual content at the target object's AR coordinate location (keeping the absolute value of the vertical axis the same as the premise coordinate system).

GLAMAR Edge Server: We use a Dell server with Intel Core i-7-8700 CPU 3.2GHz, and 64GB memory as our GLAMAR edge server. We use `python` for implementing edge service, which consists of three independent processes. They use producer-consumer ring queue for each target object. We use a single variable lock per ring queue to synchronize the access among the three processes. One producer process is responsible for enqueueing HAIP location information of the target object in the ring queue. Another producer process uses gyroscope data from the target object to determine its state: stationary or in-motion. This process enqueues the state of the target object with the accelerometer sensor reading. In stationary case, it enters zero values for accelerometer reading. For in-motion, it enqueues motion state with the accelerometer data. The final consumer process runs a particle filter that dequeues all events sequentially from the ring queue. Based on the dequeued event, particle filter updates the particles (see Section II-C2). We use 4096 particles for the particle filter implementation. GLAMAR server runs separate thread for each particle filter corresponding to the target object.

IV. Evaluation

In order for the GLAMAR framework to be useful in practice, its accuracy must be comparable to the currently used vision-based techniques. Moreover, in order to be a more attractive framework of choice than the vision-based techniques, it must run more efficiently on a user device. Therefore, to establish the efficacy of the GLAMAR framework, in the following, we evaluate it for accuracy and efficiency. The results show that GLAMAR is as accurate as vision-based techniques with significantly reduced energy usage and computation overhead.

A. Accuracy:

The accuracy of the GLAMAR framework is measured based on the positional displacement of objects (physical or virtual) compared to the "ground truth" in 3D coordinate system. The following factors affect the accuracy in GLAMAR. Precise computation of coordinate transformation between premise and AR coordinate systems has direct impact

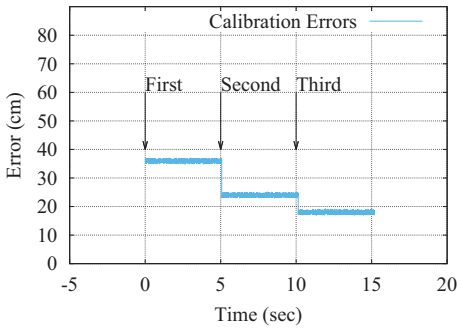

Fig. 7: Continuous coordinate calibration helps in reducing the error of tracking target objects in AR coordinate.

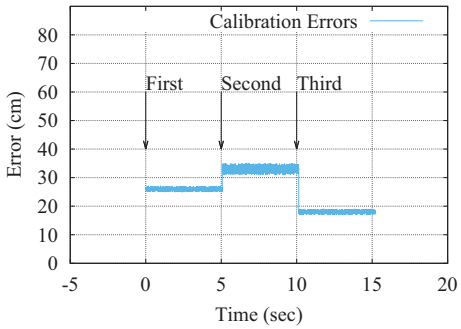

Fig. 8: Due to error in location readings, second calibration step increases the error in tracking target objects in AR coordinate.

on the accuracy of rendering the virtual contents. However, coordinate transformation depends heavily on the quality of the location data, which may fluctuate from time to time. Therefore, we evaluate the impact of having continuous coordinate conversion on the accuracy of tracking target objects. Unlike many other edge-assisted MAR systems, GLAMAR does not share large-size image data with the infrastructure. Instead, it relies on timely delivery of small amount of text data. This emphasizes the effect of network jitter on the accuracy. Finally, we discuss the improvement on accuracy by using various sensory inputs in our regenerative particle filter method.

1) Ground Truth Measurement:: For our evaluation, we need ground truth measurement of tracking target objects. Despite its popularity, vision-based technique is less reliable in tracking target objects in industrial premise [46, 47], specially due to occlusion and range limitation. Therefore, we develop a novel technique using multi-user AR to collect the ground truth measurement of tracking target objects. We leverage the open source code of "Just-a-line" [48], an Android app that allows two or more phones to sync up their AR coordinates through a real-time database in Google cloud, called Firebase [49]. Using this app, one phone can see the line that the other phone is drawing in the same physical space. We integrate the source code of "Just-a-line" with our GLAMAR agent in user device and place another phone running "Just-a-line" in the target object. The phone in the target object draws a line in it's own position, and uploads the points of the line to Firebase cloud. On the other hand, the GLAMAR agent running in user device uses the Google's cloud anchor APIs to convert the points of the target phone's position to its own AR coordinate. Thus the user device can track the position of the target objects. Note that, coordinate conversion using Google's cloud anchor APIs is closed source.

2) Effect of calibration on accuracy:: The calibration process estimates the coordinate transformation matrix between AR and premise coordinate systems. Any error in estimating this matrix can result in rendering the virtual contents in inaccurate positions. Therefore, it is important to have precise estimation of the coordinate transformation matrix. We find that repeated and frequent estimation helps in improving the accuracy in estimating the matrix. For instance, Figure 7 shows three consecutive estimations after every 5 seconds. Here we keep the target object at a fixed location, and the user device moves around to do the measurements for coordinate transformation. Figure 7 shows that multiple coordinate transformation measurements help improving the accuracy of rendering the virtual overlays on the target object. Moreover, we observe that as an AR session runs for longer duration, it may readjust the frame of reference of the AR coordinate system. However such changes do not happen often. But repeated estimation of coordinate transformation helps in readjusting for these changes.

3) Effect of calibration on accuracy:: The accuracy of coordinate transformation also depends on the accuracy of HAIP localization system. We observe that HAIP localization accuracy may very at different location at different time. Therefore, it is not always the case that more measurements for coordinate transformation improves the accuracy. Sometimes due to inaccurate HAIP location information, our estimated transformation could degrade. For instance, Figure 8 shows that the second measurement degrades the accuracy compared to the first one. However, continuous calibration helps in readjusting the error. As we see in Figure 8, third measurement helps in improving the accuracy. Therefore, the GLAMAR agent in user device runs continuous calibration to keep the coordinate transformation as accurate as possible.

4) Effect of wireless communication:: In GLAMAR, each target object sends IMU sensor reading to the edge server. Similarly, edge server sends HAIP location information to the user devices. Both communications have very little throughput requirement (1.6-2.6Kbs). However, delay and especially jitter affect the accuracy in GLAMAR. To evaluate this we use a shared WiFi network as our wireless communication medium. Figure 9 shows the very high jiiter experienced in the WiFi network (which is typical in industrial settings [50]). In addition, WiFi is power inefficient for the battery powered IMU sensor in target objects. Given low throughput requirement, therefore, as an alternative we choose Bluetooth Low Energy

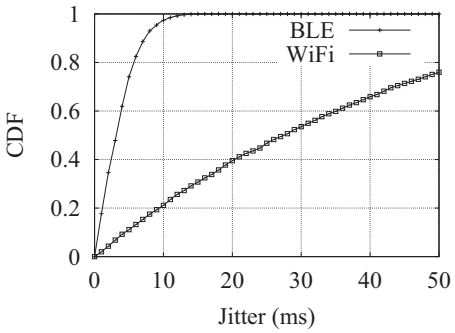

Fig. 9: Observed jitter in Wi-Fi and BLE in industrial setting.

(BLE) for the wireless communication in GLAMAR [51, 52]. In order to create a BLE network in industry settings, we deploy Raspberry Pi's that are both powered and connected to the backhaul Ethernet network. We implement a set of GATT services using the built-in Bluetooth of the Pi's to communicate with the user devices and target objects, and make the Pi's act as proxies for the edge server.

Figures 10(a) and 10(b), compare the effect of jitter on the accuracy of estimating the position of a target object. When there is any sudden movement of the target object (e.g., positions 1, 2, 3 in Figure 10(a)), GLAMAR falls behind in tracking the target object due to large jitter in WiFi network, resulting in rendered virtual content trailing the target object's position in the AR frame. However, for the same scenario with BLE as the network, we see (e.g., positions 1, 2, 3 in Figure 10(b)) more accurate and reliable tracking in GLAMAR. Furthermore, we observe smoother tracking of the target object in BLE compared to WiFi.

Relative to the ground truth based on "Just-a-line", Figure 11 compares the absolute positioning error between using WiFi and BLE for the communication. Furthermore, Figure 12 compares the power consumption between using BLE and WiFi in GLAMAR framework.

5) Accuracy of using regenerative particle-filter: Figures 13(a) through 13(d) show the gradual improvement of accuracy achieved by our particle filter process in tracking a moving target object (at $\tilde{2}$m/s) for rendering the virtual contents. In these figures, we use "Just-a-line" for tracking our ground truth (red dots). Note that, we only compare the X coordinates for easy visualization. Similar plot can be shown for other coordinates too.

Figure 13(a) shows only the *location update*, 'L' (green), which fails to track the real-path of the target object (i.e., lot of red dots are visible). Figure 13(b) depicts the case when we apply particle filter only on the *location update* ('L'). This case also has a lot of visible red dots.

Figure 13(c) shows how adding *acceleration update* ('A') in the particle filter process improves the tracking of the target object. It reduces the number of visible red dots compared to Figure 13(b). Finally, Figure 13(d) shows the case when the regenerative particle filter applied on *location update* ('L') from HAIP, *acceleration update* ('A') from IMU and *motion update* ('M'). This results in practically exact match with the true trajectory of the target object. Also note that the regenerative particle filter 13(d) (i.e., GLAMAR) provides more stable and smooth experience compared to the regular particle filter 13(c). Finally, Figure 14 shows the absolute error compared to the ground truth for the above four procedures.

The number of particles used in the particle filter process has an effect on the accuracy of tracking the target objects. Figure 15 shows the distribution of location error when different number of particles are used. Large number of particles shows more stable location measurement. However it increases overall computation cost and latency. We found 4096 particles are enough for our particle filter to provide stable location tracking for the target objects with low latency.

B. Efficiency

In order for GLAMAR framework to be useful in practice, besides accuracy, it must be more energy efficient and less computationally involved than legacy solution techniques. In this subsection, we compare GLAMAR framework with the feature-based visual tracking in terms of frame rate, energy consumption, and computational overhead. While comparison with previously proposed edge-based MAR systems would have been more appropriate, practical constraints like non-open-sourced systems, incompatible performance metrics, etc. prohibit us from doing so.

	Vision-based	GLAMAR
CPU utilization for stationary target object (%)	43.1±14	8±3
CPU utilization for in-motion target object (%)	62.7±7	9.3±4
Average frame rate (fps)	11±1.2	33±2.1

TABLE I: Compute and frame rate comparison between vision-based techniques and GLAMAR.

1) Effect on Computation: We implement an Android app using OpenCV [53] for object tracking. The app is a vision-based approach of mapping features (i.e., ORB) between 2D image and 3D model for object recognition and tracking. Here we compare the computation requirement in CPU utilization between the vision-based and GLAMAR framework. Note that, we use the CPU in Google Pixel 4, which is Octa-core (1x 2.84 GHz Kryo 485 Gold Prime 3x 2.42 GHz Kryo 485 Gold 4x 1.78 GHz Kryo 485 Silver) [54]. Table I shows the CPU utilization comparison when the target object is stationary and in-motion, respectively. Overall, GLAMAR reduces CPU requirement by 83.3% compared to vision-based techniques. For rendering, we see average frame rate of 33 and 11 fps in GLAMAR and vision-based technique, respectively, (Table I), an increase of about 200%.

2) Effect on Power Usage: Finally, we compare the battery discharge rate between the vision-based technique and GLAMAR. We use two Android apps (e.g., GSam Battery Monitor and AccuBattery) that monitor the power usage of different applications. Both apps show the estimated average

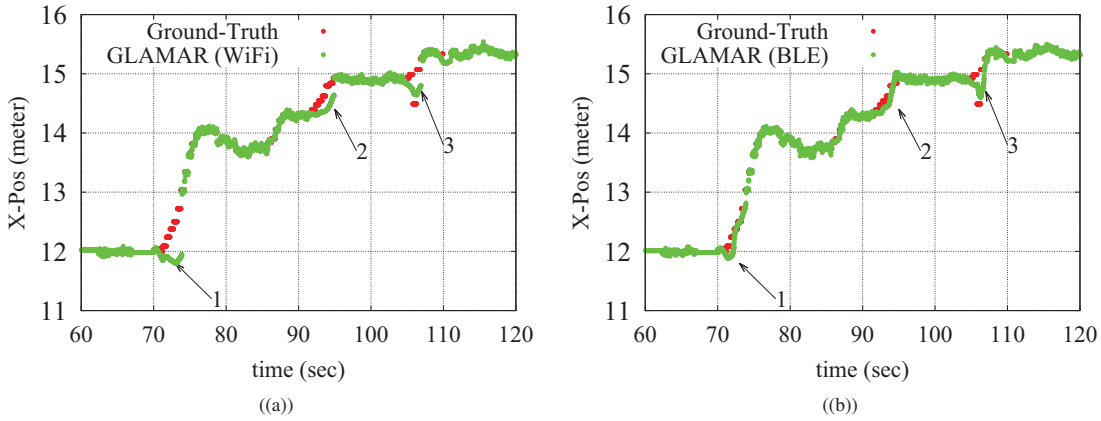

Fig. 10: Target object tracking in GLAMAR compared to the ground truth, while using (a) WiFi, and (b) BLE as communication medium.

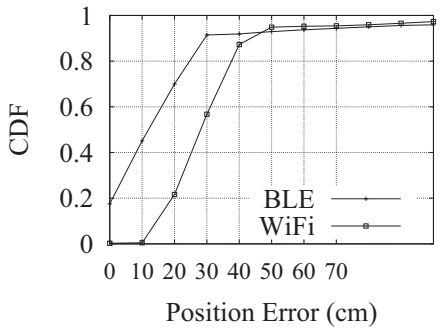

Fig. 11: Position error comparison between using BLE and Wi-Fi.

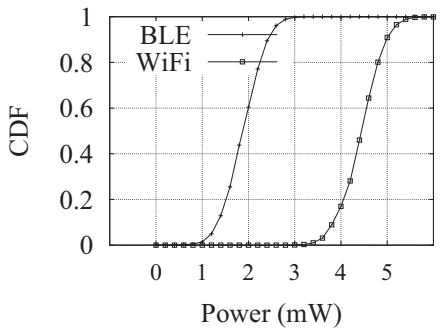

Fig. 12: Power consumption measurement using BLE and Wi-Fi.

battery discharge rate in mAh for different Android applications. However, the app requires us to run the application for longer duration for better estimation. Figure 16 shows the average battery discharge comparison between the vision-based technique and GLAMAR. GLAMAR reduces the battery discharge rate by 80% compared to the vision-based technique.

V. Prior Work

In AR application, besides real-time camera pose estimation [37, 55], it is important to accurately render the virtual content that is aligned with the physical scene for better visual acceptance. Therefore, real-time detection and tracking of the physical objects, compared to the pose of the camera, is a key part of AR applications [56]. Before comparing relevant work on object detection and tracking in the context of MAR, we briefly summarize the different techniques used for object detection and tracking in AR.

For practical deployment, most of the recent focus has been on developing marker-less object tracking methods [57, 58]. The general idea of marker-less method is to apply feature matching between the 2D image or 3D point cloud of the recognized object with the 3D model of the reference object to estimate the position of the object [59, 60]. But it requires heavy computation for object detection and tracking [61, 62]. Moreover, due to energy and computation constraints [63], most of the mobile platforms use monocular vision [64, 65] instead of stereo [66] vision for creating the virtual world of 3D point clouds, which is mostly sparse. Therefore, object detection degrades with the sparse point clouds extracted from the monocular 2D image [67]. In addition, as the object moves further away from the camera, the performance of the vision-based object tracking degrades drastically [68]. In contrast, GLAMAR shows that without vision, just by using location and IMU sensors with infrastructure support, it is practical to accurately track target objects for AR applications. Fur-

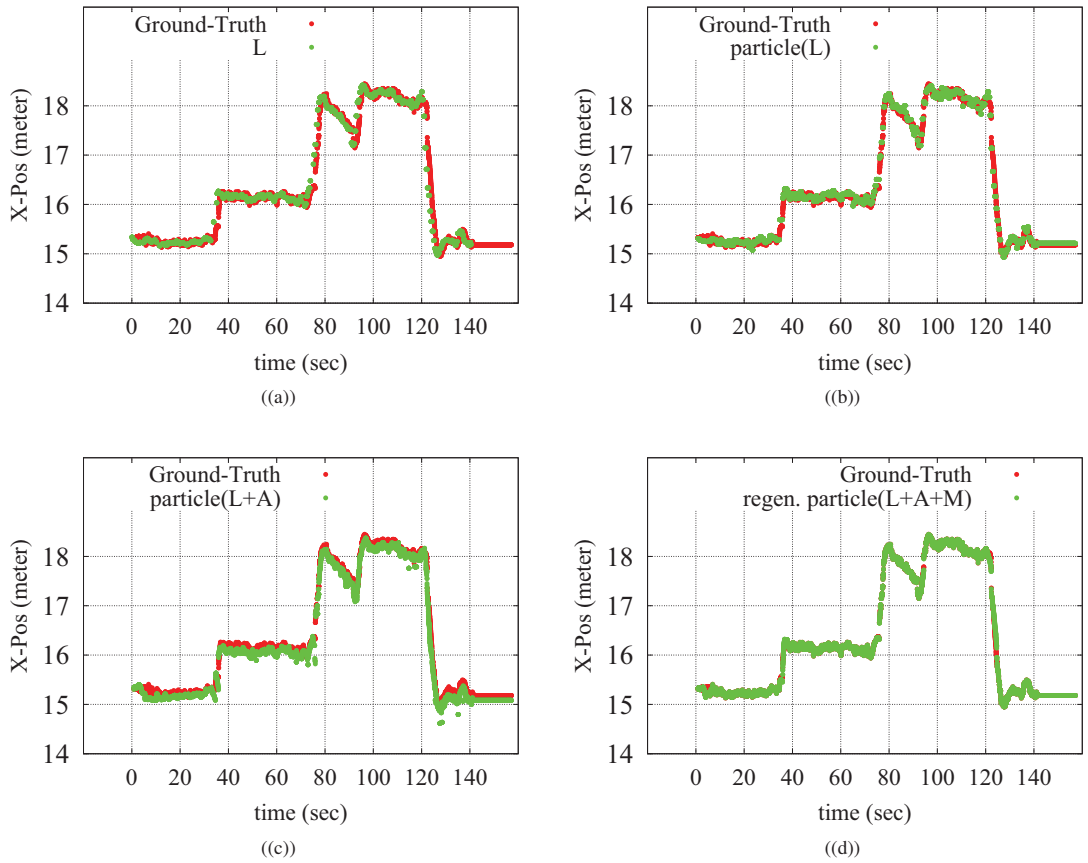

Fig. 13: In-motion target object tracking comparison (X-axis values only) over time between the ground truth and (a) *location update* (L) (b) the particle filter applied only on *location update* (L) (c) the particle filter applied on *location update* and *acceleration update* (L+A) (d) the regenerative particle filter applied on *location update*, *acceleration update*, and *motion update* (L+A+M) (i.e., GLAMAR).

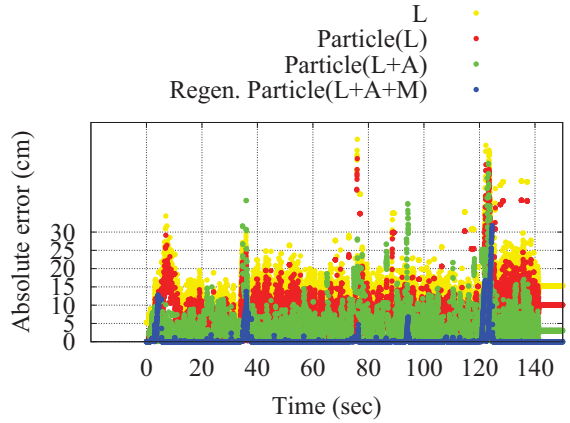

Fig. 14: Error (cm) compared to the ground truth for four procedures, L , particle filter(L), particle filter(L+A), and regenerative particle filter(L+A+M).

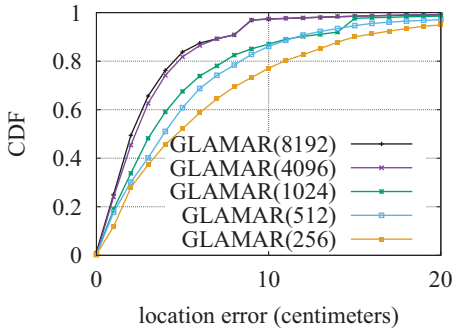

Fig. 15: Statistical distribution of absolute location error for different number of particles used in particle filter implementation.

Features \ MAR System	MARVEL [36]	Jaguar [14]	[10]	MARLIN [9]	GLAMAR
Object tracking	Stationary	Stationary	Stationary+Moving	Stationary+Moving	Stationary+Moving
Detected object type	N/A	Planar	Planar+Non-planar	Planar+Non-planar	Planar+Non-planar
Detected object position (coordinate)	3D Real-world	3D Real-world	2D Image frame	2D Image frame	3D Real-world
Solution Architecture	Edge-assisted	Edge-assisted	Edge-assisted	On-device	Edge-assisted
Integrable with external AR platforms	No	Yes	No	No	Yes
On-device computation	Moderate	Moderate	Moderate	High	Low
Privacy preservation	No	No	No	Yes	Yes

TABLE II: Comparison of GLAMAR with recently proposed MAR systems.

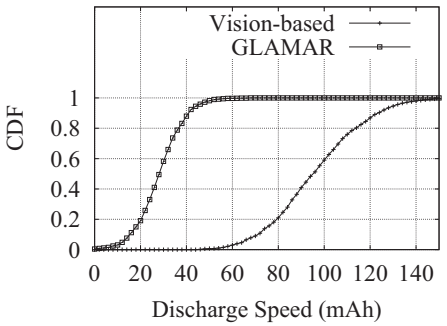

Fig. 16: Battery discharge rate comparison between GLAMAR framework and vision-based (using features) object tracking.

thermore, not using vision-based techniques makes GLAMAR more lightweight compared to other proposed solutions.

Mobile Augmented Reality (MAR): Despite the technological advancement of AR, due to resource constraint, MAR is still at its early stage [9]. Researchers have proposed different solutions to trade-off computation between on-device [9] and cloud/edge-server [14, 37, 69, 10, 36, 8, 70]. The basic idea of edge-assisted solutions is to reduce the on-device energy usage by offloading computation to the cloud/edge-server without sacrificing latency and accuracy significantly. Existing commercially available MAR platforms (i.e., ARCore [34], ARkit [35], Vuforia [71] etc.) are quite efficient in detecting planar objects (e.g., floor, ceiling, vertical wall, marker, etc.) at close proximity. However, they are limited in detecting and tracking the non-planar objects, which most of the real-world objects are. Furthermore, the accuracy evaluation of edge-assisted MAR solutions [36, 9, 70, 10, 8] in literature are based on pixel overlap between the target object and the detected object. Thefore we are unable to compare our accuracy detection model with the existing edge-based MAR solutions in literature. Moreover, most of these systems are closed source which make it challenging to use them as baseline for our quantitative comparison.

Table 2 summarizes the comparison among different MAR systems including GLAMAR. Among the few recently proposed MAR systems, MARVEL [36] and Jaguar [14] can only track stationary objects, and find their true position in the AR world. However they are incapable of tracking moving objects. On the other hand, MARLIN [9] and [10] can track moving objects. However, they are unable to track the true position in AR world, and instead they measure the 2D position of the target object in the camera frame for overlaying bounding-box. Note that, recognizing and tracking objects in camera frame (2D) may be enough to render overlaying information on the object, but they are incapable of rendering 3D virtual objects attached to the real-world position of the target objects. GLAMAR is the only framework that can track the real-world position of both stationary and moving target objects. In addition, GLAMAR is lighter-weight and more energy efficient compared to the proposed MAR solutions with vision-based object tracking. Unlike other edge-assisted MAR solutions, user agent in GLAMAR need not share any information with the edge server, which makes it protective of user privacy.

Particle-Filter for location tracking: Particle filter has been used in many object tracking applications to model the non-linearity and the non-Gaussianity of a physical model [72], especially in robotics [73]. There are several variants of the particle filter that have been introduced for real-time object tracking [74, 75, 72]. These particle filters consist of two fundamental steps; prediction and update (correct). For better visual experience in AR, it is essential to have more frequent and consistent update steps. However, due to low frequency and irregular update steps we see a lot more fluctuation in AR. In GLAMAR, we adapt the particle filter, where we introduce a new step of particle regeneration, besides estimating the state pdf for the prediction. Our method reduces error accumulation and fluctuation in the prediction in between the sparse location updates. This new action increases the accuracy of tracking target objects and improves overall user's experience.

VI. Discussion and Conclusion

In this paper we propose a practical but lightweight MAR-based application development framework called GLAMAR. It leverages the latest infrastructure deployed in an industrial premise, and uses that very effectively to reduce compute load and battery consumption on a smart phone without sacrificing the accuracy of MAR applications.

We conduct comprehensive evaluation of the system in an indoor industry environment and show the efficacy of the system. We have also experimented with the system in the outdoor environment where we used GPS instead of HAIP for location tracking. We observed that GPS is much more

erroneous and slower in location updates compared to HAIP. While this resulted in larger error (4-8m), we believe that incorporation of RTK in addition to GNSS will significantly improve the results. Moreover, due to larger size of the outdoor objects (e.g., vehicle, store front, etc.) compared to indoor objects, the relative error diminishes and has less impact on overlaying virtual content over real-world objects. We leave more detailed study in the outdoor environment as future work.

GLAMAR supports multiple AR users concurrently, where each user locally computes the coordinate transformation without revealing it's own location to the GLAMAR service. Thus GLAMAR preserves location privacy for each user. On the other hand, GLAMAR edge service tracks the location of all target objects, and delivers them to the users. This provides the opportunity to implement access control policies for each target object on a per user basis. We intend to explore the policy implementation challenges in GLAMAR in the future.

In GLAMAR, we attach location tracking tag and IMU sensor with the target objects. While we recognize this as an additional deployment cost, we argue that many target objects in industrial environment are already equipped with motion sensor and tracking devices. For example, industrial robots are equipped with IMU sensor and RF tracking [76]. Furthermore, for inventory management, objects are attached with RFID tags for tracking [77]. Therefore GLAMAR can leverage existing and forthcoming industrial infrastructure for deployment of MAR-based applications.

REFERENCES

[1] Yang Lu. 2017. Industry 4.0: a survey on technologies, applications and open research issues. *Journal of Industrial Information Integration*, 6, 1–10.

[2] Michael Rüßmann, Markus Lorenz, Philipp Gerbert, Manuela Waldner, Jan Justus, Pascal Engel, and Michael Harnisch. 2015. Industry 4.0: the future of productivity and growth in manufacturing industries. *Boston Consulting Group*, 9, 1, 54–89.

[3] Volker Paelke. 2014. Augmented reality in the smart factory: supporting workers in an industry 4.0. environment. In *Proceedings of the 2014 IEEE emerging technology and factory automation (ETFA)*. IEEE, 1–4.

[4] Paula Fraga-Lamas et al. 2018. A review on industrial augmented reality systems for the industry 4.0 shipyard. *Ieee Access*, 6, 13358–13375.

[5] Riccardo Masoni et al. 2017. Supporting remote maintenance in industry 4.0 through augmented reality. *Procedia manufacturing*, 11, 1296–1302.

[6] Tristan Braud et al. 2017. Future networking challenges: the case of mobile augmented reality. In *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 1796–1807.

[7] Wenxiao Zhang, Bo Han, and Pan Hui. 2017. On the networking challenges of mobile augmented reality. In *Proceedings of the Workshop on Virtual Reality and Augmented Reality Network*, 24–29.

[8] Qiang Liu, Siqi Huang, Johnson Opadere, and Tao Han. 2018. An edge network orchestrator for mobile augmented reality. In *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*. IEEE, 756–764.

[9] Kittipat Apicharttrisorn et al. 2019. Frugal following: power thrifty object detection and tracking for mobile augmented reality. In *Proceedings of the 17th Conference on Embedded Networked Sensor Systems*, 96–109.

[10] Luyang Liu et al. 2019. Edge assisted real-time object detection for mobile augmented reality. In *The 25th Annual International Conference on Mobile Computing and Networking*, 1–16.

[11] Xiuquan Qiao et al. 2019. Web AR: a promising future for mobile augmented realitystate of the art, challenges, and insights. *Proceedings of the IEEE*, 107, 4, 651–666.

[12] Jean-Pierre Lomaliza and Hanhoon Park. 2019. Learning-based estimation of 6-dof camera poses from partial observation of large objects for mobile ar. In *25th ACM Symposium on Virtual Reality Software and Technology*, 1–2.

[13] Chaoyun Zhang, Paul Patras, and Hamed Haddadi. 2019. Deep learning in mobile and wireless networking: a survey. *IEEE Communications Surveys & Tutorials*, 21, 3, 2224–2287.

[14] Wenxiao Zhang et al. 2018. Jaguar: low latency mobile augmented reality with flexible tracking. In *Proceedings of the 26th ACM international conference on Multimedia*, 355–363.

[15] Michael Lowney and Abhilash Sunder Raj. 2016. Model based tracking for augmented reality on mobile devices. (2016).

[16] Chi-Yi Tsai and Shu-Hsiang Tsai. 2018. Simultaneous 3D object recognition and pose estimation based on RGB-D images. *IEEE Access*, 6, 28859–28869.

[17] Eric Marchand et al. 2015. Pose estimation for augmented reality: a hands-on survey. *IEEE transactions on visualization and computer graphics*, 22, 12, 2633–2651.

[18] Roman Klokov and Victor Lempitsky. 2017. Escape from cells: deep kd-networks for the recognition of 3d point cloud models. In *Proceedings of the IEEE International Conference on Computer Vision*, 863–872.

[19] Charles R Qi, Or Litany, Kaiming He, and Leonidas J Guibas. 2019. Deep hough voting for 3d object detection in point clouds. In *Proceedings of the IEEE International Conference on Computer Vision*, 9277–9286.

[20] Yu Xiang and other. 2017. PoseCNN: a convolutional neural network for 6d object pose estimation in cluttered scenes. *arXiv preprint arXiv:1711.00199*.

[21] Alexander Grabner et al. 2018. 3D pose estimation and 3d model retrieval for objects in the wild. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 3022–3031.

[22] Mahdi Rad et al. 2018. Feature mapping for learning fast and accurate 3D pose inference from synthetic images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 4663–4672.

[23] Chung-Lin Huang and Wen-Chieh Liao. 2004. A vision-based vehicle identification system. In *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.* Volume 4. IEEE, 364–367.

[24] Tuyen X Tran et al. 2017. Collaborative mobile edge computing in 5g networks: new paradigms, scenarios, and challenges. *IEEE Communications Magazine*, 55, 4, 54–61.

[25] Sami Kekki et al. 2018. MEC in 5g networks. *ETSI white paper*, 28, 1–28.

[26] Yun Chao Hu et al. 2015. Mobile edge computinga key technology towards 5G. *ETSI white paper*, 11, 11, 1–16.

[27] Robert-Steve Schmoll et al. 2018. Demonstration of VR/AR offloading to mobile edge cloud for low latency 5G gaming application. In *2018 15th IEEE Annual Consumer Communications & Networking Conference (CCNC)*. IEEE, 1–3.

[28] Mario Collotta et al. 2018. Bluetooth 5: a concrete step forward toward the IoT. *IEEE Communications Magazine*, 56, 7, 125–131.

[29] FiRa Consortium. 2020. *fine ranging*. https://www.firaconsortium.org/about/consortium. FiRa Consortium.

[30] W Stempfhuber and M Buchholz. 2011. A precise, low-cost RTK GNSS system for UAV applications. *Proc. of Unmanned Aerial Vehicle in Geomatics, ISPRS*.

[31] DWF Van Krevelen and Ronald Poelman. 2010. A survey of augmented reality technologies, applications and limitations. *International journal of virtual reality*, 9, 2, 1–20.

[32] Abhishek P Patil et al. 2005. A study of frequency interference and indoor location sensing with 802.11 b and bluetooth technologies. In *Symposium, 2005 Wireless Telecommunications*. IEEE, 174–183.

[33] Jukka Rantala. 2012. High accuracy indoor positioning-technology solution and business implications. In *3rd Invitational Workshop on Opportunistic RF Localization for Next Generation Wireless Devices, NOKIA Research Center*.

[34] Google. 2020. *ARCore Google Developers*. https://developers.google.com/ar. Google.

[35] Apple. 2020. *Augmented Reality - Apple Developer*. https://developer.apple.com/augmented-reality/. Apple.

[36] Kaifei Chen et al. 2018. MARVEL: enabling mobile augmented reality with low energy and low latency. In *Proceedings of the 16th ACM Conference on Embedded Networked Sensor Systems*, 292–304.

[37] Puneet Jain, Justin Manweiler, and Romit Roy Choudhury. 2016. Low bandwidth offload for mobile ar. In *Proceedings of the 12th International on Conference on emerging Networking EXperiments and Technologies*, 237–251.

[38] Stefan Leutenegger, Paul Furgale, Vincent Rabaud, Margarita Chli, Kurt Konolige, and Roland Siegwart. 2013. Keyframe-based visual-inertial slam using nonlinear optimization. *Proceedings of Robotis Science and Systems (RSS) 2013*.

[39] Apple. 2020. *Understanding World Tracking*. https://developer.apple.com/documentation/arkit/understanding_world_tracking. Apple.

[40] Google. 2020. *Concept of AR*. https://developers.google.com/ar/reference/c/group/concepts. Google.

[41] Android. 2020. *Sensor Coordinate System*. http://josejuansanchez.org/android-sensors-overview/coordinate_system/README.html. Android.

[42] Seong-hoon Peter Won et al. 2009. A kalman/particle filter-based position and orientation estimation method using a position sensor/inertial measurement unit hybrid system. *IEEE Transactions on Industrial Electronics*, 57, 5, 1787–1798.

[43] Oliver Heirich. 2016. Bayesian train localization with particle filter, loosely coupled GNSS, IMU, and a track map. *Journal of Sensors*, 2016.

[44] Yafei Ren and Xizhen Ke. 2010. Particle filter data fusion enhancements for MEMS-IMU/GPS.

[45] Pengfei Gui et al. 2015. MEMS based IMU for tilting measurement: comparison of complementary and kalman filter based data fusion. In *2015 IEEE 10th conference on Industrial Electronics and Applications (ICIEA)*. IEEE, 2004–2009.

[46] Jochen Teizer. 2015. Status quo and open challenges in vision-based sensing and tracking of temporary resources on infrastructure construction sites. *Advanced Engineering Informatics*, 29, 2, 225–238.

[47] Alessandro Vandini et al. 2014. Vision-based motion control of a flexible robot for surgical applications. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, 6205–6211.

[48] Google. 2020. *Just A Line- Draw anywhere with AR*. https://justaline.withgoogle.com. Google.

[49] Google. 2020. *Firebase*. https://firebase.google.com. Google.

[50] Fabian Rincon et al. 2018. On the impact of wifi on 2.4 ghz industrial IoT networks. In *2018 IEEE International Conference on Industrial Internet (ICII)*. IEEE, 33–39.

[51] Guntur Dharma Putra et al. 2017. Comparison of energy consumption in Wi-Fi and bluetooth communication in a smart building. In *2017 IEEE 7th Annual Computing and Communication Workshop and Conference (CCWC)*. IEEE, 1–6.

[52] Ali Nikoukar et al. 2018. Low-power wireless for the internet of things: Standards and applications. *IEEE Access*, 6, 67893–67926.

[53] Gary Bradski and Adrian Kaehler. 2008. *Learning OpenCV: Computer vision with the OpenCV library*. " O'Reilly Media, Inc.".

[54] Google Pixel 4 Spec. [n. d.] *Google Pixel 4*. https://store.google.com/product/pixel_4_specs. Google Pixel 4 Spec.

[55] Kaifei Chen, Jonathan Fürst, John Kolb, Hyung-Sin Kim, Xin Jin, David E Culler, and Randy H Katz. 2018. Snaplink: fast and accurate vision-based appliance control in large commercial buildings. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 1, 4, 1–27.

[56] Xukan Ran et al. 2019. ShareAR: communication-efficient multi-user mobile augmented reality. In *Proceedings of the 18th ACM Workshop on Hot Topics in Networks*, 109–116.

[57] Ahyun Lee et al. 2011. Markerless augmented reality system based on planar object tracking. In *2011 17th Korea-Japan Joint Workshop on Frontiers of Computer Vision (FCV)*. IEEE, 1–4.

[58] Juan Lei et al. 2011. Real-time object tracking on mobile phones. In *The First Asian Conference on Pattern Recognition*. IEEE, 560–564.

[59] Changhyun Choi and Henrik I Christensen. 2010. Real-time 3D model-based tracking using edge and keypoint features for robotic manipulation. In *2010 IEEE International Conference on Robotics and Automation*. IEEE, 4048–4055.

[60] Michael Manz et al. 2011. Monocular model-based 3D vehicle tracking for autonomous vehicles in unstructured environment. In *2011 IEEE International Conference on Robotics and Automation*. IEEE, 2465–2471.

[61] Yin Zhou and Oncel Tuzel. 2018. Voxelnet: end-to-end learning for point cloud based 3d object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 4490–4499.

[62] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. 2019. Pointrcnn: 3d object proposal generation and detection from point cloud. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 770–779.

[63] Jungsik Park et al. 2017. Binocular mobile augmented reality based on stereo camera tracking. *Journal of Real-Time Image Processing*, 13, 3, 571–580.

[64] H Firouzi and H Najjaran. 2010. Real-time monocular vision-based object tracking with object distance and motion estimation. In *2010 IEEE/ASME International Conference on Advanced Intelligent Mechatronics*. IEEE, 987–992.

[65] Eric Royer et al. 2007. Monocular vision for mobile robot localization and autonomous navigation. *International Journal of Computer Vision*, 74, 3, 237–260.

[66] Maximilian Muffert et al. 2013. A stereo-vision based object tracking approach at roundabouts. *IEEE Intelligent Transportation Systems Magazine*, 5, 2, 22–32.

[67] Shaoshuai Shi et al. 2019. Part-a$\hat{2}$ net: 3D part-aware and aggregation neural network for object detection from point cloud. *arXiv preprint arXiv:1907.03670*.

[68] Bela Julesz and Dov Sagi. 1987. Short-range limitation on detection of feature differences. *Spatial Vision*, 2, 1, 39–49.

[69] Tiffany Yu-Han Chen et al. 2015. Glimpse: Continuous, real-time object recognition on mobile devices. In *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems*, 155–168.

[70] Wenxiao Zhang et al. 2018. CARS: Collaborative augmented reality for socialization. In *Proceedings of the 19th International Workshop on Mobile Computing Systems & Applications*, 25–30.

[71] Dennise Adrianto, Monica Hidajat, and Violitta Yesmaya. 2016. Augmented reality using Vuforia for marketing residence. In *2016 1st International Conference on Game, Game Art, and Gamification (ICGGAG)*. IEEE, 1–5.

[72] M Sanjeev Arulampalam, Simon Maskell, Neil Gordon, and Tim Clapp. 2002. A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking. *IEEE Transactions on signal processing*, 50, 2, 174–188.

[73] Sebastian Thrun. 2002. Particle filters in robotics. In *Proceedings of the Eighteenth conference on Uncertainty in artificial intelligence*. Morgan Kaufmann Publishers Inc., 511–518.

[74] António Almeida, Jorge Almeida, and Rui Araújo. 2005. Real-time tracking of moving objects using particle filters. In *Proceedings of the IEEE International Symposium on Industrial Electronics, 2005. ISIE 2005.* Volume 4. Citeseer, 1327–1332.

[75] Dieter Fox et al. 2001. Particle filters for mobile robot localization. In *Sequential Monte Carlo methods in practice*. Springer, 401–428.

[76] MiR. 2020. *Mobile Industrial Robots*. https://www.mobile-industrial-robots.com/en/. MiR.

[77] Xiaowei Zhu, Samar K Mukhopadhyay, and Hisashi Kurata. 2012. A review of RFID technology and its managerial applications in different industries. *Journal of Engineering and Technology Management*, 29, 1, 152–167.