SeeNav: Seamless and Energy-Efficient Indoor Navigation using Augmented Reality

Marius Noreikis
Aalto University
Espoo, Finland
marius.noreikis@aalto.fi

Yu Xiao
Aalto University
Espoo, Finland
yu.xiao@aalto.fi

Antti Ylä-Jääski
Aalto University
Espoo, Finland
antti.yla-jaaski@aalto.fi

ABSTRACT

Augmented Reality (AR) based navigation has emerged as an impressive, yet seamless way of guiding users in unknown environments. Its quality of experience depends on many factors, including the accuracy of camera pose estimation, response delay, and energy consumption. In this paper, we present SeeNav – a seamless and energy-efficient AR navigation system for indoor environments. SeeNav combines image-based localization and inertial tracking to provide an accurate and robust camera pose estimation. As vision processing is much more compute intensive than the processing of inertial sensor data, SeeNav offloads the former one from resource-constrained mobile devices to a cloud to improve tracking performance and reduce power consumption. More than that, SeeNav implements a context-aware task scheduling algorithm that further minimizes energy consumption while maintaining the accuracy of camera pose estimation. Our experimental results, including a user study, show that SeeNav provides seamless navigation experience and reduces the overall energy consumption by 21.56% with context-aware task scheduling.

KEYWORDS

augmented reality; indoor navigation; energy efficiency

1 INTRODUCTION

Since the first augmented reality (AR) system was prototyped in 1993, the AR devices and applications have developed a lot. Recent AR applications such as Pokémon GO have attracted tremendous attention. Their success shows that people are getting acquainted with AR applications. Meanwhile, the market of AR applications is expected to grow to $83 billion by 2021 [3]. However, there are still a lot of technical challenges to be solved in order to provide seamless user experience.

In this work we present SeeNav[1], a seamless and energy-efficient AR navigation system for indoor environments. SeeNav solves two key challenges to the user experience of an AR navigation: the **continuous and accurate** camera pose estimation, and the **energy efficiency** of the AR application.

A camera pose in general refers to a combination of a position and an orientation. The camera pose relative to a 3D space can be represented by at least six parameters, three representing the translation and the others representing the rotation. Currently, fiducial marker based tracking [7, 8] is widely used by popular AR platforms, such as Vuforia, ARToolkit and Metaio. However, predefined markers require installation and maintenance. To overcome this issue, researchers have proposed a marker-less tracking [13, 20] that works by extracting and matching natural visual features in the environment. Such technique is promising, yet faces unsolved issues when an environment contains insufficient visual features. Another approach is called inertial tracking, which utilizes inertial sensors such as a gyroscope and an accelerometer for estimating the heading and the orientation of the device. The inertial tracking suffers from severe drift problems [10, 11], thus cannot provide accurate tracking during the continuous operation. To solve the above-mentioned problems, we propose a novel tracking algorithm that complements a marker-less visual tracking with an inertial tracking to provide a continuous, accurate and robust tracking. Our approach fully utilizes the high accuracy of an image-based localization [6] and marker-less tracking in feature-full environments[2], and the robustness of the inertial sensing in feature-less environments.

Energy efficiency is a key criterion of designing applications for mobile devices such as smartphones and AR glasses [4]. Due to the heavy computation of vision processing and the continuous operation of cameras, the energy efficiency of AR applications remains a challenge. For example, after the launch of a widely played AR mobile game Pokémon GO, there were numerous complaints about the application draining phone's battery in a few hours [1, 21]. To shed more light towards this issue, we measured the power consumption of 5 freely available AR based applications. We compared the results to our solution and additionally included measurements of a non-AR navigation app – Google Maps, to emphasize the power consumption difference between AR based and conventional mobile applications. As illustrated in Figure 1, AR applications in general draw much more power. The most power hungry one, Campus Maps[3], would drain a typical smartphone battery of 2800mAh in less than two hours. To reduce the energy consumption on mobile devices, SeeNav offloads compute-intensive tasks (i.e. vision based camera pose estimation) to a cloud. In addition, SeeNav implements a context aware task scheduling algorithm which minimizes the frequency of image sampling while maintaining a high accuracy of the tracking, and utilizes a light weight rendering engine for

[1] Demo video available at https://tinyurl.com/n8tnsgg

ThematicWorkshops'17, October 23–27, 2017, Mountain View, CA, USA

DOI: 10.1145/3126686.3126733

[2] Our implementation of image-based localization provides 1 meter accuracy in our test environment.
[3] http://www.navvis.com/campus-maps

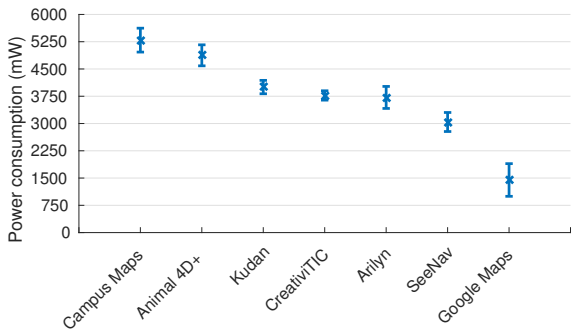


Figure 1: Comparison of power consumption between SeeNav, 5 other AR applications, and Google Maps - Navigation & Transit. Except for Google Maps, the other applications were run in AR mode (camera capture and the related vision processing is involved).

augmented objects to further reduce the power consumption. According to our experimental results, the energy consumption in general decreases by 21.56% with context-aware task scheduling.

The contributions of this work are summarized below.

(1) It combines a marker-less visual tracking with an inertial sensing to provide an accurate and robust camera pose estimation in indoor environments.
(2) It offloads heavy computations to a cloud to enable energy efficiency and scalability.
(3) It proposes an adaptive, context aware task scheduling algorithm that drastically reduces the power consumption imposed by the visual tracking.
(4) It provides an experimental evaluation of an AR navigation system from performance, energy-efficiency and usability perspectives.

Besides the scientific contributions, in this paper we also share our practical experience on developing and implementing an AR navigation system. The rest of this paper is structured as follows: Section 2 gives an overview of background and related work, Section 3 describes system design, Section 4 explains the tracking mechanisms, and Section 5 presents the experimental evaluation before we conclude the paper in Section 6.

2 BACKGROUND AND RELATED WORK

Before we summarize the previous works on the AR navigation, we will first introduce a 6 Degrees of Freedom (DoF) tracking, which is an essential building block of AR applications. Note that a 6 DoF movement of an object can be described as translation and rotation in a 3D space. For marker-less AR applications, it is vital to support the 6 DoF tracking in order to properly attach augmented objects to a scene.

2.1 6 DoF Tracking

Indoor positioning techniques that use Bluetooth low energy beacons [24] or Ultra-wideband (UWB) transmitters [22] only support 3 DoF tracking. The techniques can only provide a position of a smartphone but not its facing direction. Moreover, these techniques require installation and maintenance of an extra hardware infrastructure.

There are two typical ways of providing a 6 DoF tracking. The first is to couple beacons with inertial sensors [25]. Given an initial position obtained from beacons, inertial sensors on a mobile device can estimate an orientation of the device in respect to the world compass, and can report 3D rotations from a gyroscope and 3D movement from an accelerometer. Solely inertial sensor based tracking is resource efficient, however, it is known to be sensitive to noise and its tracking accuracy degrades over time [5].

The other approach is to calculate a 3D pose from a 2D camera image. For example, Huang et al. [12] utilized panorama images to estimate the position and the facing direction of a user in indoor environments. Jiang et al. [6] proposed *iMoon* that implements an image-based localization based on Structure from Motion (SfM) techniques. SfM supports large scale 3D reconstructions from unordered images [9]. Liu et al. [17] utilized deep learning for image matching and tracking along indoor paths. These approaches achieve high accuracy, compared with inertial tracking. However, it comes at a price of highly increased processing requirements and power consumption. Moreover, vision based approaches may fail in certain environments with insufficient visual features, e.g. plain white walls, windows, and glass panels. Therefore researchers proposed fusing visual tracking with other types of sensing [7, 16, 17].

Liu et al. [16] presented an AR application framework that incorporates both visual and inertial sensors to provide context aware tracking solution. They highly focused on mitigating inertial sensor errors on consumer devices and improving an overall scene tracking accuracy. Their tracking solution runs locally on a smartphone and requires acoustic beacons deployed in the area to obtain a coarse-grained location before starting the visual tracking. However, their solution may face serious challenges in large venues or places where acoustic beacons are not available.

In this work, we adopt ideas similar to *iMoon* for calculating position and orientation of a smartphone. Differently from *iMoon*, the focus of this paper is placed on the fusion of visual and inertial sensor data for a continuous real time tracking.

2.2 AR Navigation

Simultaneous Localization And Mapping (SLAM) has been employed by AR devices such as Google Tango[4] and Microsoft Hololens[5] for device tracking and localization. To implement SLAM, these devices are typically equipped with depth cameras.

Concerning the fact that depth cameras are rarely available on consumer devices, researchers have proposed to use visual SLAM for a smartphone-based indoor tracking, since the visual SLAM requires only a monocular camera. The visual SLAM can provide not as accurate, though reliable enough real time tracking and localization. Due to real time requirements, the visual SLAM works with an ordered sequence of images and is applied for tracking short trajectories only. For example, HyMoTrack [7] utilizes SLAM tracking, 2D to 3D feature matching to identify the initial position

[4]https://get.google.com/tango/
[5]https://www.microsoft.com/en-us/hololens

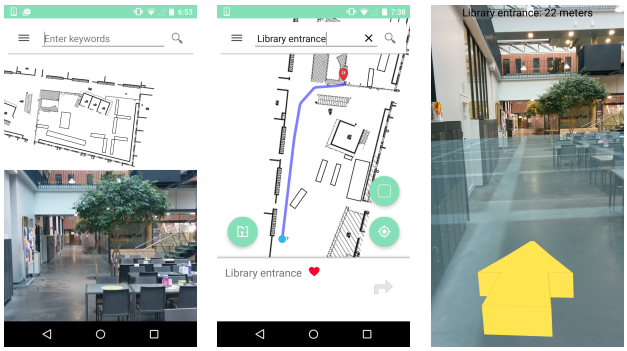


Figure 2: Screenshots of the SeeNav Android client. From left to right: a) identifying an initial location with an image; b) the main screen showing a map of the premises, the current position of a user (marked as a blue dot) and a planned route to a destination; c) AR navigation mode.

of a device, and falls back to an inertial tracking when the vision-based approach fails. Since all the processing is done on a device, they had to lower the frame rate to around 20 frames per second.

Mulloni et al. [19] introduced an AR navigation that utilizes step detection for tracking and fiducial markers as info points to calibrate a position of a user. However, the approach still requires effort to install and maintain additional markers.

Ventura et al. [23] proposed a client-server-based tracking solution for an AR navigation. Researchers combined a global localization on a server side with the visual SLAM tracking on a mobile device. The initial position of the client was obtained by matching an image sent by a client to an SfM model on the server side. In order to speed up the localization and narrow down the search space, the device also sends GPS coordinates, as well as a device compass rotation. Later, the position returned from the server is used in combination with position and orientation estimated by the local SLAM algorithm. The system achieved an accuracy of several meters outdoors and less than a meter accuracy in a small indoor area. However, the initialization of the tracking system took several seconds and the system lacked scalability for large areas and larger amounts of users.

In our work we utilize SfM based localization techniques, which enable fast localization in large scale environments and offer an instant system startup. Instead of combining SfM and SLAM, we fuse SfM-based pose estimation with inertial tracking to achieve high accuracy while enabling tracking in feature-less environments.

3 SYSTEM OVERVIEW

SeeNav implements the application scenario described in Section 3.1, and follows the cloud-based system architecture presented in Section 3.2.

3.1 Application Scenario

Alice arrives at a new conference center, which she visits for the first time in her life. She needs to find a conference room and would also like to take a cup of coffee on the way. The conference center is a complex building with multiple halls, rooms and floors. It is not trivial to find a way around, even by following signs inside the venue. Luckily, Alice has a smartphone with an AR based indoor navigation app. She opens the application and locates herself by simply pointing the phone towards the premises. She instantly gets a 2D map with her location and sees nearby Point-of-Interests (PoIs). She can already see that there is a café nearby. She then searches for her conference room and receives a navigation path to the place. However, instead of analyzing the route, she simply switches on the AR mode and an augmented arrow conveniently points towards a direction she needs to go to. In this way, Alice reaches her destination quickly and stress-free.

To implement the above-mentioned scenario, SeeNav provides the following features: (1) obtaining the initial position of a user based on an image captured inside premises, (2) offering PoI search, (3) supporting real-time and accurate camera pose estimation and tracking, and (4) containing a 3D rendering engine for showing augmented 3D objects inside the camera view.

3.2 Cloud-based System Architecture

Previous works [7, 23] showed lots of effort in running the vision based localization on mobile devices. However, due to resource demanding computer vision algorithms, the localization could only be done within a small area or highly influenced the device rendering frame rate. To solve this issue, SeeNav adopts a client-server architecture where compute-intensive and data-intensive tasks are offloaded to the backend server running on a cloud.

Backend Server The backend server is responsible for building and maintaining 3D point clouds, calculating navigation paths and instructions, and estimating real time camera poses from images. Regarding the implementation, SeeNav applies the same approach as iMoon [6] to create 3D models (in the form of point clouds) of indoor environments and implement an image-based localization using SfM. Differently from iMoon, SeeNav takes video as an input for 3D model generation, and is implemented using an open source library OpenMVG[6]. Since indoor environments may change over time, our service supports partial 3D model updates, when a video of the changed scene is uploaded to the system. We build an SfM based 3D model from the new video, match and append it to an already existing 3D point cloud and remove the points from the old point cloud that overlap with the new one. The image-based localization service accepts a smartphone-taken photo as an input, and calculates an accurate position and orientation (pose) of the device based on feature matching. The hybrid approach that combines image-based localization and inertial tracking will be detailed in Section 4. Since we execute localization services on a cloud which already holds 3D point clouds and venue maps, we also utilize the cloud environment for launching navigation service and PoI search. Our navigation service is based on an A* (A star) algorithm and can quickly find a waypoint path between a given source and a destination.

Android Client The client collects visual and inertial sensor data based on context (described in Section 4.2), conducts a light-weight sensor fusion (described in Section 4.1), and provides user interfaces. As shown in Figure 2, SeeNav Android client provides user interfaces for browsing a top-down floor plan, locating and displaying user's position on the floor plan, searching for PoIs, finding

[6] open Multiple View Geometry library https://github.com/openMVG/openMVG

paths to a chosen destination, and guiding the user to the destination using AR. To reduce power usage and provide impressive visuals, we designed and developed a light weight 3D rendering engine for AR. Our 3D rendering engine is based on Android framework Java bindings for OpenGL, thus provides rapid startup times and high frame rates. It contains all the essential parts of a rendering engine, including textured objects rendering, hierarchical scene graphs, external resource loading, caching and drawing. We analyzed the energy efficiency of the engine as shown in Section 5.2. We also evaluated the usability of our mobile client through a small-scale user study that is explained in detail in Section 5.3.

The above-described cloud-based architecture design of SeeNav allows us to balance the trade-off between accuracy, response delay, and energy efficiency. Firstly, offloading compute-intensive vision-based mapping and localization from resource-constrained mobile devices to the cloud reduces response delay and energy consumption. Secondly, downloading and storing a 3D point cloud – that is necessary for calculating camera poses – locally on a device would require a significant amount of the device networking and storage resources. For example, the size of a 3D point cloud can be more than 200 MB for a medium-size (around 4500 m^2) venue. Thirdly, the 3D point clouds need to be updated when the indoor scenes change. It is much easier to maintain and update the point clouds when they are stored in the cloud instead of the end devices. Finally, executing localization on a cloud enables easier deployment and maintenance of client applications on different consumer devices including smart phones, watches or glasses that may have different operating systems (i.e. Android, iOS, Windows), because client code is leaner and the only requirements for an end user device is to capture photos, collect data from inertial sensors and communicate with the server.

4 ROBUST AND ENERGY-EFFICIENT TRACKING

We propose to complement the marker-less visual tracking with an inertial tracking to improve the robustness and energy-efficiency. Regarding the robustness, whenever an image-based localization fails due to occlusion or lack of visual features, a subsequent camera pose can be estimated based on the inertial sensors alone. Meanwhile, we minimize the frequency of the image-based localization, in order to reduce the energy consumption while maintaining a sufficient level of tracking accuracy.

4.1 Sensor Fusion for Robust Tracking

Figure 3 illustrates the workflow of camera pose estimation when both visual and inertial sensors data is available. Here, the visual data refers to an image captured from a rear-facing camera of a hand-held device, while the inertial sensor data refers to gyroscope and accelerometer readings. We omit using an integrated digital compass, because magnetic fields are highly preturbed indoors [2] and compass readings may change significantly while passing by inductive or magnetic devices, computers, or metal structures. As we obtain an initial camera pose from the image-based localization service, we rely on the inertial sensors for estimating only delta rotations and therefore can completely avoid using a biased compass sensor.

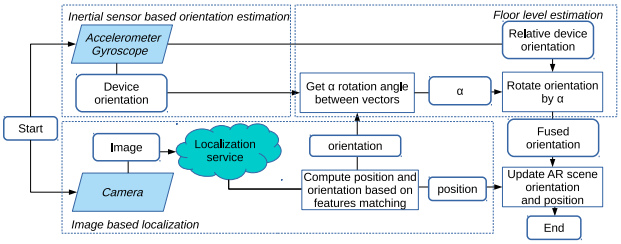


Figure 3: The workflow of a hybrid tracking. The input includes both image and inertial sensor data.

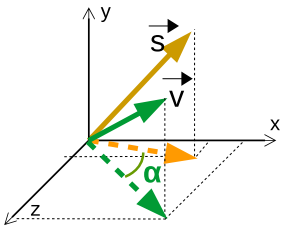


Figure 4: Estimation of the rotation angle α between orientations. $\vec{s}$ and $\vec{v}$ represent the estimates of orientations based on inertial sensor data and image, respectively.

The workflow starts by capturing an image from a camera preview session and obtaining the current device orientation from a gyroscope. We have to note, that in order to start the tracking, our algorithm has to obtain an initial position and orientation from the image based localization. The image is sent to the backend server for calculating a 6 DOF camera pose. Once the response arrives, the mobile client calculates the rotation angle α between projections of an inertial sensor based and an image based direction vectors (as illustrated in Figure 4). After that, the mobile client estimates the floor level based on a relative device orientation, and rotates the augmented floor level by the angle α to properly align real-world and augmented scenes together. At the end, the mobile client offsets the augmented scene according to the position returned from the image-based localization. The implementations of the inertial tracking and the floor level estimation are described below.

Rotation Estimation We utilize an Android framework rotation vector sensor[7] that combines readings of accelerometer and gyroscope to provide a quaternion rotation of a device. As the estimated rotation is noisy, we address this issue by averaging quaternion rotations. Concerning the high sensor sampling rate and the requirement for a low latency, we apply a fast averaging algorithm [18] to calculate the average over a small set of samples. In practice, we empirically set the number of samples to 20, which is enough to make the rotation tracking jitter-free and smooth.

Calculation of Position Offset Obtaining a relative position offset is not a trivial task. Previous works show that user's position can be tracked using dead-reckoning [11, 14] techniques. Dead-reckoning is a process of calculating a current position based on a previous position and the estimated speed over an elapsed time.

[7] https://source.android.com/devices/sensors/sensor-types.html#game_rotation_vector

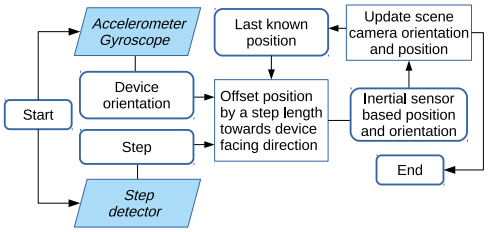
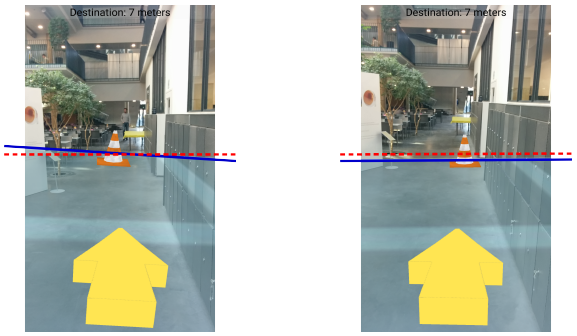


Figure 5: The workflow of inertial tracking.



(a) Without calibration **(b) With calibration**

Figure 6: Floor level estimation without/with using inertial sensors for calibration. Solid blue lines represent augmented world orientations, and dashed red lines represent real world floor orientations.

Step detection sensors became pervasive in recent years with an increasing number of phones that have a dedicated step detection hardware. Furthermore, the step detection proved to be more accurate than tracking the accelerometer readings [14]. While imaging data is not available or after the image-based localization fails, SeeNav utilizes the step detection to calculate subsequent positions of a user. As described in Figure 5, once a step is detected, the mobile client calculates a new position by offsetting the previous one by a step length towards the facing direction of the device. We assume that the user walks approximately towards the same way as the phone is facing. Concerning the drift problem of the dead-reckoning, the image-based localization is needed for a trajectory calibration. The scheduling between the image-based localization and the inertial tracking must take into account both accuracy and energy efficiency. The detailed tracking algorithm is described in Section 4.2.

Floor Level Estimation We render an artificial floor plane at a floor level to check how the augmented world will be placed on top of the camera view. If there is an error in the pose estimation, the floor plane looks tilted in respect to the real world floor (see Figure 6a). Even without rendering the augmented floor it is easy to spot that augmented objects that were supposed to be placed on the ground are either "flying" or "buried" below the ground. In order to eliminate the errors in the floor level estimation, we use the inertial sensors to estimate orientations of the device and the augmented floor. We obtain a quaternion rotation of a device in respect to an Earth's gravity vector, and exploiting the fact that a floor level indoors is mostly flat, we estimate augmented floor rotation, which then perfectly aligns with the ground level, visible in the camera view (see Figure 6b).

4.2 Context-aware Task Scheduling for Energy Savings

The aim of this work is not only to provide accurate tracking but to also minimize power consumption. As presented in the previous section, we track a user using inertial sensors between the two consecutive image based localization requests. As we show later in Section 5.2, the image based tracking requires much more power than the inertial tracking. In the AR view, we could issue a new localization request whenever the previous one is completed to have the best accuracy of the tracking. However, this would dramatically increase the number of requests made to the server, and would also increase the power consumption on a mobile device.

To minimize the energy consumption while maintaining the accuracy, SeeNav adopts the following four mechanisms. **1)** When a user is in a stationary position (only changing an orientation), we do not issue localization requests unless a user turns by at least δ degrees, since we already know the position of the user and have a fairly accurate orientation estimation. **2)** Once the user starts walking, we obtain an image based location every T steps to quickly account for any errors caused by the step detection. **3)** After obtaining each image based location, we calculate the distance error ϵ between last known sensor based position and the image based position. If $\epsilon > E + \varepsilon$, then we start increasing the frequency of the image based localization ($T = T - 1$). When the error is $\epsilon < E - \varepsilon$, we start decreasing the frequency ($T = T + 1$). Here, $E = 1$ meter, $\varepsilon = 0.15$ meter. We also enforce constraints on T: $T_{min} = 5$ and $T_{max} = 30$. **4)** The only time when two subsequent requests are sent to the server is when localization query fails and the client application needs to send another image for localization. Throughout extensive tests, explained in Section 5.1, we show that optimal values for our case are $T = 20$ steps and $\delta = 40$ degrees, thus we use those values to initialize our adaptive tracking algorithm.

5 EVALUATION

We have evaluated the performance and energy efficiency of SeeNav through experiments. More specifically, we analyzed the accuracy of the tracking algorithm described in Section 4.2 using different image sampling frequencies, and measured processing delay of localization requests. Regarding energy-efficiency, the mean power consumption of the SeeNav client in the AR mode is compared with other AR applications. In addition, we analyze the breakdown of the power consumption between software features. We have also conducted a small-scale user study to evaluate the usability and efficiency of the system in real environments. The experimental results are discussed in this section.

5.1 Performance

We evaluated the accuracy of our tracking algorithm through a field test in a university building that contains classrooms, long corridors, a restaurant area and a library. We firstly took videos of the premises and built a 3D point cloud using SfM. The point cloud contained more than 450.000 points and covered approximately

(a) 5 steps (b) 10 steps (c) 20 steps (d) 10 steps + angle (e) 20 steps + angle

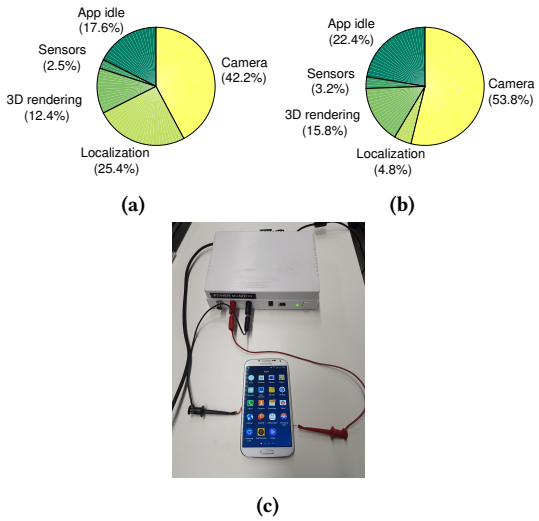
Figure 7: Comparison between planned navigation paths (in solid blue lines) and the estimated trajectories (dashed orange lines). For the estimated paths, small dots represent positions estimated using inertial sensors, while larger dots represent positions obtained from image-based localization. During the tests, users always walked along the planned paths. From left to right: a) Image based localization done every 5 steps, b) every 10 steps, c) every 20 steps, d) every 10 steps or when a turn was detected, e) every 20 steps or when a turn was detected.

$1500m^2$. It was used for image-based localization and path finding. In this venue we achieved position accuracy of less than 1 meter and facing direction accuracy of less than 5 degrees, 90% of the time. We then used the SeeNav Android client to conduct an AR navigation in the following four steps.

(1) Open the Android client and obtain the initial position based on a photo (see Figure 2)
(2) Choose a destination inside the premises
(3) Begin navigating and follow the planned navigation path (solid blue line in Figure 7)
(4) Finish the test once the application reports that the destination is reached

During the test we recorded every position update in the client application and drawn them as a path alongside a navigation route planned by the backend system. As presented in Figure 7, solid blue paths represent system generated navigation routes, which we followed, while dashed orange lines show the path positions estimated by the mobile client. Cyan and red markers represent position of a user and a destination, respectively, when the system notified that the user has reached the destination.

As described in Section 4.2, the image-based localization is scheduled according to the number of accumulated steps after getting the previous image-based location, and the relative rotation. Figures 7a - 7c show the results when the client issued localization requests every 5, 10 and 20 steps. While having localization every 5 steps showed a good accuracy, increasing the step interval resulted in notable navigation errors. It is clearly visible in the Figure 7c, where the image based localization was done only every 20 steps. This low frequency of the image-based localization allowed large drifts, especially after a user turned. According to Figures 7d - 7e, if the image-based localization was done also after every significant turn, the overall tracking accuracy dramatically improves, even at the interval of 20 steps. Our later tests concluded that the best threshold for detecting a turn was around 40 degrees. When the threshold



(a) (b)

(c)

Figure 8: (a) Power consumption breakdown for the different features and components (b) The breakdown after applying the adaptive tracking algorithm (c) Setup of the power measurement

was smaller, due to the movement of a user, the image based localization was issued even when the user was going along rather straight paths. When it was larger, some turns were undetected.

During the tests we also measured response delays of the image based localization. We recorded the duration from when a photo is sent from the client until a camera pose is returned. The mean response delay is $466.83ms$ ($\sigma = 99.8$). During the tests, the client phone was connected to a Wi-Fi network and was sending images of an average size of 222KB ($\sigma = 86.19$).

5.2 Energy efficiency

To evaluate the power consumption of the SeeNav Android client in the AR mode, we utilized Samsung Galaxy S4(GT-I9506) smartphone with Android 6.0.1 operating system. We connected the device to a Monsoon power monitor tool[8] to monitor the power consumption (see Figure 8c).

To get an initial understanding about the problem of power consumption, we evaluated 5 freely available AR applications: *creativiTIC* and *Animal 4D+* that are based on *Vuforia*[9] AR engine, *Kudan Simple Samples* which is based on *Kudan*[10] framework and *Arilyn*[11]. We also tested *Campus Maps* mobile application that has similar functionality to our AR navigation system. For a comparison, we included a widely used *Google Maps* mobile application in navigation mode. We ran each application for 60 seconds and measured the mean power consumption. We did not consider the power consumption during an application start up phase but rather during a regular use, e.g. real time tracking for the first four applications, image based localization mode for the *Campus Maps* and navigation modes for *Google Maps* and SeeNav.

Figure 1 shows the power consumption of the benchmarked applications. It is instantly clear that AR based applications consume way more energy than a contemporary navigation application, which proves that it is challenging to make an energy efficient AR based navigation application. *Kudan*, *Vuforia* and *Arilyn* based applications showed very similar results, while *Campus Maps* consumed the most power. SeeNav, on the other hand, showed to be more power efficient than the other AR based applications. However, it still consumes almost twice the power of Google Maps.

In order to understand which components of our client application contribute the most towards the power consumption, we executed the application with its different modules turned off. We started with a setup where the application is showing only a 2D map and has its basic user interface (UI) initialized. We then enabled camera view, inertial tracking, augmented 3D scene rendering, and image based localization services one by one. We ran the tests 10 times for 30 seconds and recorded the mean power consumption. For the localization component test, we sent localization requests periodically, every 1 second. After we obtained mean power consumption values from each aforementioned component, we subtracted the mean power consumption of an idle application to get raw power consumption for each component.

The comparison of power consumption for different mobile application components is presented in Figure 8a. It is noteworthy to mention, that the camera view consumes more than 42% of total power. Even after changing the camera resolution from the highest to the lowest, the power usage difference was negligible. During our experiments, we did not use camera flash light. Since the camera view is an essential component of an AR application, the power consumption of camera cannot be removed. The inertial tracking accounted only for 2.5% of power consumption and the augmented scene rendering for 12.4%. However, the image-based localization contributed significantly to the overall power consumption (25.4%). It is due to transmitting localization requests and receiving responses from the server. We calculated that each localization request on average consumes $485.62J$. Therefore, it is clear that decreasing the frequency of the server requests helps to notably reduce overall power consumption. In practice, due to a dynamic fashion of our context aware tracking algorithm, we send image-based localization requests much less often than once every second. Though it highly depends on premises, planned paths and users behavior, during our case study, explained in Section 5.3, we show that localization requests are typically issued every 7.2 seconds. Figure 8b shows a power consumption breakdown with our adaptive tracking enabled. Obviously, the power consumption of the localization component significantly reduces after applying the adaptive tracking algorithm.

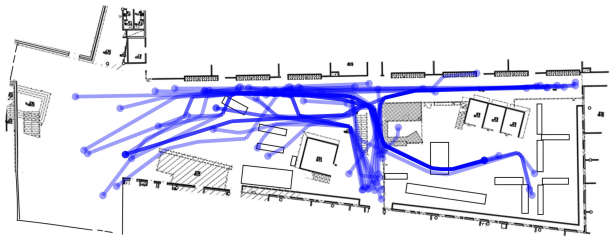


Figure 9: Recorded paths of study participants

5.3 User study

In order to prove usefulness and performance of our system, we conducted a small scale user study. We randomly asked 17 participants (8 female and 9 male) whom we met in our office building to try out SeeNav. All the participants were in age groups of 20-29 and 30-39 years. All the participants admitted being lost in indoors at least sometimes and 16 of them agreed that indoor navigation would be useful for them.

Each participant was asked to complete the following tasks: (1) Use localization function to locate themselves within premises. (2) Select a destination. (3) Follow the navigation arrow in the AR mode to reach the destination. During each study case, we observed the participants but did not interfere with them in any way. We also collected the following data on our mobile client: 1) traces where a participant walked, 2) time to reach the destination, 3) a number of image-based localization requests. Figure 9 shows all recorded paths that the study participants took. The participants travelled in total $1653m$. Regarding image-based localization, a request was issued on average once every $7.22s$ (σ = 2.21), or around 9 requests per minute. Therefore, comparing with power consumption of image-based localization when the localization is done every second (see Figure 8a), our adaptive tracking algorithm achieved more than 85% energy savings on localization and decreased the total amount of application power consumption by 21.56%.

Every participant was asked to fill in a survey about the user experience. We have designed our survey questions in a way that we could apply Likert scale [15] on the results. The questions were:

(1) Can you locate yourself with the application?
(2) Is the location accuracy good enough?
(3) Is the AR view useful in finding your destination?
(4) Does the navigation route show a clear and correct path to the destination?

[8] https://www.msoon.com/LabEquipment/PowerMonitor
[9] https://www.vuforia.com/
[10] https://www.kudan.eu/
[11] http://www.arilyn.fi

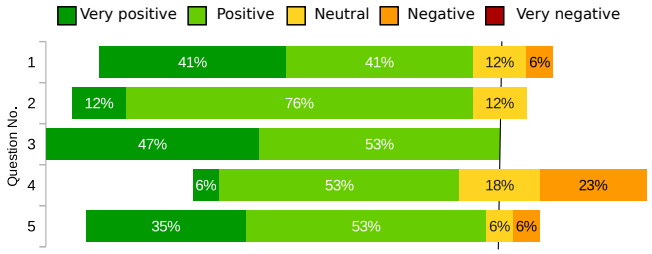


Figure 10: Answers to survey questions

(5) Is the floating arrow helpful in finding your way?

Additionally, each participant could write additional comments at the end of the survey. Figure 10 shows the results of the survey as a diverged stacked chart, where colored bars represent how well participants have felt about each feature that we asked. For example for question 1, very positive answer is "All the time", while the most negative is "Never".

The survey shows that 88% of the participants agreed that SeeNav provided good enough accuracy and 82% were able to easily locate themselves in the premises. Even though 6% of the respondents indicated that they could not easily locate themselves, all the respondents successfully arrived to their destinations. A few respondents experienced errors in finding their way (question 4). This was mainly due to failures of the image based localization, since those persons were mostly pointing the phone camera downwards. We are planning to address the issue in a future design of SeeNav, by introducing haptic feedback and more clear instructions for using camera based localization. From this user study we make two important conclusions: 1) Users are not keen to keep phone upwards even after instructed to do so, especially for prolonged periods of time, 2) an intuitive AR based navigation is easy to follow and is highly helpful for the users.

6 CONCLUSION

In this work we present SeeNav – a seamless and energy efficient AR navigation solution. We achieve high navigation accuracy and reliable pose estimation by fusing visual and inertial sensor data. We achieve low latency and minimize power consumption by adopting a cloud-based architecture and deploying a context-aware tracking algorithm. Finally, the user study proved our solution to be handy, effective and user friendly, with numerous positive responses from the respondents. As a future work, we plan on adding an augmented path and augmented information boards to the mobile client, to further enhance our AR navigation system.

ACKNOWLEDGMENTS

This work was supported by the Academy of Finland (grant number 268096 and 278207) and Tekes: Finnish Funding Agency for Innovation.

REFERENCES

[1] 2016. Pokemon GO Drains A LOT MORE Battery since the last update! https://pokemongohub.net/pokemon-go-drains-a-lot-more-battery-since-the-last-update/. (9 November 2016).

[2] Marzieh Jalal Abadi, Luca Luceri, Mahbub Hassan, Chun Tung Chou, and Monica Nicoli. 2014. A collaborative approach to heading estimation for smartphone-based PDR indoor localisation. In *Indoor Positioning and Indoor Navigation (IPIN), 2014 International Conference on*. IEEE, IEEE, 554–563.

[3] Michael Abrash. 2017. After mixed year, mobile AR to drive $108 billion VR/AR market by 2021. http://www.digi-capital.com/news/2017/01/after-mixed-year-mobile-ar-to-drive-108-billion-vrar-market-by-2021/. (January 2017).

[4] Clemens Arth and Dieter Schmalstieg. 2011. Challenges of large-scale augmented reality on smartphones. *Graz University of Technology, Graz* (2011), 1–4.

[5] Jeffrey R Blum, Daniel G Greencorn, and Jeremy R Cooperstock. 2012. Smartphone sensor reliability for augmented reality applications. In *International Conference on Mobile and Ubiquitous Systems: Computing, Networking, and Services*. Springer, 127–138.

[6] Jiang Dong, Yu Xiao, Marius Noreikis, Zhonghong Ou, and Antti Ylä-Jääski. 2015. imoon: Using smartphones for image-based indoor navigation. In *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems*. ACM, 85–97.

[7] Georg Gerstweiler, Emanuel Vonach, and Hannes Kaufmann. 2015. Hymotrack: A mobile AR navigation system for complex indoor environments. *Sensors* 16, 1 (2015), 17.

[8] Alexandru Gherghina, Alexandru-Corneliu Olteanu, and Nicolae Tapus. 2013. A marker-based augmented reality system for mobile devices. In *Roedunet International Conference (RoEduNet), 2013 11th*. IEEE, 1–6.

[9] Jared Heinly, Johannes L Schonberger, Enrique Dunn, and Jan-Michael Frahm. 2015. Reconstructing the world* in six days*(as captured by the yahoo 100 million image dataset). In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 3287–3295.

[10] Yingjuan Hu, Xuewen Liao, Qian Lu, Shulin Xu, and Wei Zhu. 2016. A segment-based fusion algorithm of WiFi fingerprinting and pedestrian dead reckoning. In *Communications in China (ICCC), 2016 IEEE/CIC International Conference on*. IEEE, 1–6.

[11] Baoqi Huang, Guodong Qi, Xiaokun Yang, Long Zhao, and Han Zou. 2016. Exploiting cyclic features of walking for pedestrian dead reckoning with unconstrained smartphones. In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing*. ACM, 374–385.

[12] Jiung-Yao Huang, Su-Hui Lee, and Chung-Hsien Tsai. 2016. A fast image matching technique for the panoramic-based localization. In *Computer and Information Science (ICIS), 2016 IEEE/ACIS 15th International Conference on*. IEEE, 1–6.

[13] Christian Koch, Matthias Neges, Markus König, and Michael Abramovici. 2014. Natural markers for augmented reality-based indoor navigation and facility maintenance. *Automation in Construction* 48 (2014), 18–30.

[14] Fan Li, Chunshui Zhao, Guanzhong Ding, Jian Gong, Chenxing Liu, and Feng Zhao. 2012. A reliable and accurate indoor localization method using phone inertial sensors. In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*. ACM, 421–430.

[15] Rensis Likert. 1932. A technique for the measurement of attitudes. *Archives of psychology* (1932).

[16] Kaikai Liu and Xiaolin Li. 2015. Enabling context-aware indoor augmented reality via smartphone sensing and vision tracking. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)* 12, 1s (2015), 15.

[17] Zhenguang Liu, Luming Zhang, Qi Liu, Yifang Yin, Li Cheng, and Roger Zimmermann. 2017. Fusion of Magnetic and Visual Sensors for Indoor Localization: Infrastructure-Free and More Effective. *IEEE Transactions on Multimedia* 19, 4 (2017), 874–888.

[18] F Landis Markley, Yang Cheng, John Lucas Crassidis, and Yaakov Oshman. 2007. Averaging quaternions. *Journal of Guidance, Control, and Dynamics* 30, 4 (2007), 1193–1197.

[19] Alessandro Mulloni, Hartmut Seichter, and Dieter Schmalstieg. 2011. Handheld augmented reality indoor navigation with activity-based instructions. In *Proceedings of the 13th international conference on human computer interaction with mobile devices and services*. ACM, 211–220.

[20] Karl Pauwels, Leonardo Rubio, and Eduardo Ros. 2016. Real-time pose detection and tracking of hundreds of objects. *IEEE Transactions on Circuits and Systems for Video Technology* 26, 12 (2016), 2200–2214.

[21] Hollister Sean and Fleenor Rebecca. 2016. How Pokemon Go affects your phone's battery life and data. https://www.cnet.com/how-to/pokemon-go-battery-test-data-usage/. (13 July 2016).

[22] Guowei Shi and Ying Ming. 2016. Survey of Indoor Positioning Systems Based on Ultra-wideband (UWB) Technology. In *Wireless Communications, Networking and Applications*. Springer, 1269–1278.

[23] Jonathan Ventura, Clemens Arth, Gerhard Reitmayr, and Dieter Schmalstieg. 2014. Global localization from monocular slam on a mobile phone. *IEEE transactions on visualization and computer graphics* 20, 4 (2014), 531–539.

[24] Lauri Wirola, Tommi A Laine, and Jari Syrjärinne. 2010. Mass-market requirements for indoor positioning and indoor navigation. In *Indoor Positioning and Indoor Navigation (IPIN), 2010 International Conference on*. IEEE, 1–7.

[25] R. G. Yudanto and F. Petr. 2015. Sensor fusion for indoor navigation and tracking of automated guided vehicles. In *2015 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*. 1–8.