Object Recognition Models for Indoor Users' Location

Franco M. Borrelli✉[1,2] [0000-0002-1185-4461] and Cecilia Challiol [1,3] [0000-0001-5140-0264]

[1] LIFIA, Facultad de Informática, UNLP, La Plata, Buenos Aires, Argentina
[2] UNLP Master's Scholarship, Argentina
[3] CONICET, Argentina
{fborrelli,ceciliac}@lifia.info.unlp.edu.ar

Abstract. Despite technological advances, precise positioning within buildings remains a considerable challenge. In this context, the present paper explores the research of user location in indoor spaces, embracing object recognition models executed directly on mobile devices. Our proposal is based on designing a generic solution architecture adaptable to any physical environment, enabling the definition and usage of relevant generic objects within the environment to determine the users' current location. This proposal uses Computer Vision, employing object recognition models for positioning. This kind of indoor positioning benefits from the growth of smartphones' functionalities and capabilities, thus avoiding the need to install additional infrastructures in physical spaces. A specific implementation of this architecture for *React Native* is presented, using the *TensorFlow* platform to support object recognition. This implementation allows demonstrating how this positioning works through concrete use cases. In addition, some lessons learned are discussed, which we hope will contribute to this topic.

Keywords: Object Recognition Models, Indoor Location, User Location, Lightweight Networks.

1 Introduction

The general adoption of mobile devices, coupled with technological advancements in recent years, has facilitated the growth of a diverse range of mobile applications, particularly those context-aware [1, 2]. These applications use, for instance, environmental data surrounding the user to provide different kinds of services or information. Location is a relevant context in these applications, which can be obtained using different technologies such as GPS, Wi-Fi, or Bluetooth [3]. While this context has been extensively utilised in outdoor spaces thanks to the advantages of GPS, resolving the users' location in indoor spaces still poses a challenge [3].

Although several solutions have been proposed for indoor locations, a scalable and effective solution that works universally has not yet been achieved. In [4] and [5], the use of beacons as a location-sensing mechanism is explored, while in [6] and [7], Wi-Fi network fingerprinting is employed for user location. Besides, these technologies require installing additional infrastructure to work in the environment. For instance, beacons require the installation of numerous battery-operated devices that transmit

Bluetooth signals, demanding regular maintenance. Something similar occurs with Wi-Fi fingerprinting, which implies the installation of at least three Wi-Fi antenna networks. Therefore, deploying and maintaining such infrastructures can be expensive, which could be a limitation.

On the other hand, the approach known as *vision-based indoor positioning* [3] has emerged in recent years. This approach proposes using object recognition models to locate users in indoor spaces. Although promising ad-hoc solutions have been developed using this approach [3], generalised implementations are still lacking. This is part of the motivation of this paper.

Object recognition (and detection) aims to enable computer systems to identify or classify objects in images and videos in an automated and precise manner [8]. In the traditional vision, the execution of recognition models is linked to data centers and large clusters of machines with powerful GPUs. However, this alternative can be costly and requires transferring all mobile device data and sending them through a network connection, which can be time-consuming.

Technological advancements allow nowadays to perform object recognition locally on mobile devices [9] using *lightweight networks* [8, 10]. This is a new area of research, and its viability for particular uses still needs to be evaluated [8]. Furthermore, as far as we know, *lightweight networks* are little explored for positioning in indoor spaces. This paper contributes in this direction.

Another aspect to mention is that currently, the implementation of object recognition and detection models has been significantly simplified with the development of platforms such as *TensorFlow*[1], which offers a wide range of pre-trained models that are ready to use. Additionally, these platforms often provide facilities for training new models using techniques such as transfer learning [11].

Considering the above mentioned, this paper presents a generic solution architecture for positioning users in indoor spaces using object recognition models that run locally on mobile devices (via *lightweight networks*). This proposal allows for the definition and utilisation of any relevant generic object within the environment to position users, and this positioning can be embedded in different kinds of applications. It is essential to highlight that this solution aims to leverage the smartphones' capabilities and avoid the need to install additional infrastructure.

Furthermore, this paper presents a specific implementation of the proposed architecture to demonstrate, through use cases, how this positioning works. In particular, a library was implemented in *React Native*[2] combined with the *TensorFlow* library for *Javascript* to support object recognition. This paper also discusses some lessons learned, which we hope will contribute to this field.

This paper is structured as follows. Section 2 describes the state of the art regarding object recognition and its applicability for positioning, as well as some details of *lightweight networks*. Section 3 presents a generalised solution architecture for indoor positioning using object recognition. A specific instantiation of this architecture is described

[1] Tensorflow. https://tensorflow.org/, last accessed 2024/03/15
[2] React Native. https://reactnative.dev/, last accessed 2024/03/15

in Section 4. Section 5 illustrates through use cases how this instantiation works. Finally, Section 6 presents some conclusions and future work.

2 Review of literature

Object recognition and detection have experienced remarkable evolution in the last two decades [8, 10]. This section explicitly addresses two concepts: *vision-based indoor positioning* and *lightweight networks*, which form the foundation of the research presented in this paper.

2.1 Vision-based indoor positioning

An interesting use of object recognition explored in recent years is *vision-based indoor positioning* [3], which seeks to employ object detection to guide and locate the user within indoor spaces. The concept of *vision-based indoor positioning* was first coined by Microsoft [12] in 1998. At that time, Microsoft's Vision Technology group analysed how technology could facilitate life in the future, and one of the proposed fundamental technologies was locating people in the home using various video surveillance techniques. Since then, this concept has evolved to three possible implementation options [3], which vary in how information is stored and processed. These are detailed below:

- The first one is known as *'object-based reference'*, which focuses on detecting static objects in images and then searching for correspondences of these objects with a database of buildings (often represented as 3D models) containing information about the objects' location within them. This approach can be executed through various methods; for example, in [3], a *'control point'* method is used where each object of interest in space is composed of a series of these points, representing physical characteristics with associated precise coordinates. Subsequently, an algorithm compares the distances between the coordinates detected by the camera with those recorded in a database, allowing the calculation of the user's distance from these objects. Although the results obtained in [3] are promising, it is essential to note a significant limitation: the used model requires training with specific environment data to identify control points correctly. This requirement limits the use of this solution in different environments. This motivates our proposed generic architecture in this paper, which is adaptable to various physical spaces where databases may contain generic objects that could be present in different environments.
- The second alternative is known as *'reference from images'*, which proposes comparing previously taken images of specific routes within the building with the current view of the device's camera. This approach requires an initial data collection stage where images covering relevant locations of the environment should be taken. Then, these images must be labelled by establishing connections between visual elements and specific locations within the building. Subsequently, on real-time execution, the device's camera's current view is compared with previously captured image se-

quences. The system uses algorithms to identify correspondences between the current scene and the stored images. The primary obstacle of this approach resides in achieving real-time positioning capability because image correspondence involves an exceptionally high computational load. Additionally, the *'reference from images'* option is highly coupled to specific environments, making achieving a generalisable solution to any physical space more complex.

- Finally, the use of *'reference from deployed coded targets*' proposes using objects such as barcodes, QR codes, and dot patterns. For this, it is necessary to generate the markers, link them with relevant information about the place, and physically place them in their corresponding location. This option presents the inherent challenges in using QR codes, such as their deterioration. Additionally, these codes must be generated and placed in relevant locations for each physical space, making this kind of approach ad-hoc for each environment.

As mentioned before, the *'reference from images'* option is highly joined to specific environments because it requires the development of models capable of recognising specific buildings' features, making it less versatile and too complex to propose a scalable solution. The case of the *'reference from deployed coded targets'* presents previously mentioned challenges, such as being ad-hoc for each environment. For these reasons, this paper focuses on exploring the first alternative, *'object-based reference'*, proposing a flexible and adaptable architecture for various physical spaces. In this architecture, the objects in the databases are generic and could, therefore, be present in different environments.

2.2 Lightweight networks

In recent years (from 2017 onwards), a new branch of research has emerged within the Computer Vision and Deep Learning areas, known as *lightweight networks* [8, 10]. These investigations aim to design small and efficient networks for environments with limited resources, such as mobile and IoT devices. Generally, these networks use techniques such as pruning, quantisation, and hashing to improve model efficiency. A common characteristic of all these networks is that they sacrifice accuracy for increased speed in inference. As mentioned in [8], several architectures make use of *lightweight networks*, such as *MobileNet*, *ShuffleNet*, and *Once-For-All (OFA),* all of them present two interesting features:

- They can adequately recognise objects in real-time on medium to high-end devices, ensuring their utility in a wide variety of mobile devices.
- Additionally, they run locally on the device, eliminating the need for an internet connection and optimising both response speed and user experience.

The *lightweight networks* are promising but still face some outstanding challenges that should be resolved [8-10, 13]. For example, one of the main issues is that a solution is still needed to address detection on video streams for these devices [10]. Furthermore,

the speed gap between a machine and human eyes remains large, especially for detecting small objects [10]. Moreover, *lightweight networks* are still few explored for user positioning in indoor spaces; thus, this paper aims to contribute to this area.

3 Proposed architecture for Object-based Positioning

This section proposes a generic architecture for any physical space, allowing the creation and use of relevant generic objects within the environment to users' location. According to that, two key aspects were considered when this architecture was designed, such as:

- A mechanism is required to recognise objects present in the physical space. This task will be addressed through object recognition and detection models capable of recognising generic objects in the environment.
- It is essential to have additional information so that, from the detected objects, the user can be contextualised and located in a specific physical space.

To ensure that both aspects mentioned above work to locate users effectively, two stages are needed: one focused on defining and creating these relevant positions, and the other centred on using this information to infer the user's current location. Below, each of these stages is detailed:

- *Creation*: The architecture needs to provide a mechanism for users to register in-situ data in the contextual object database. This database which will store detailed information about each detected (and relevant) object, linking each one with its respective spatial location and other relevant data. This location could be taken with any positioning mechanism. The solution must be flexible and extensible to enrich the data that could be loaded in the future.
- *Usage*: Once the contextual object database is built, the available information can be used to locate a user. When the user points the device's camera at an object, it is determined whether the object is registered in the contextual database and whether the user is *'near'* the registered location for that object. Thus, in case of success, an application (using this kind of positioning) can, for example, provide relevant information or specific services associated with that location and recognised object.

Figure 1 depicts the proposed architecture. Different applications can embed the *Object-based Positioning* module to load the contextual database (*creation*) or access previously loaded data (*usage*) and thus use this positioning mechanism to locate the user. Additionally, applications could use both functions (*creation* and *usage*). It is important to mention that the contextual database could be local or not to the mobile device, depending on the requirements and complexity of the implementation.

Furthermore, Figure 1 shows that the *Object-based Positioning* knows and observes two modules: one for object recognition and another for positioning. *Object-based Positioning* observes the modules because it constantly *'listens'* for any changes (whether a new detected object or a new location). The *Object-based Positioning* can also com-

municate directly with these modules (through the *'knows'* relationship) to request information, which usually occurs in response to the user's action. Below are more details about these two modules.

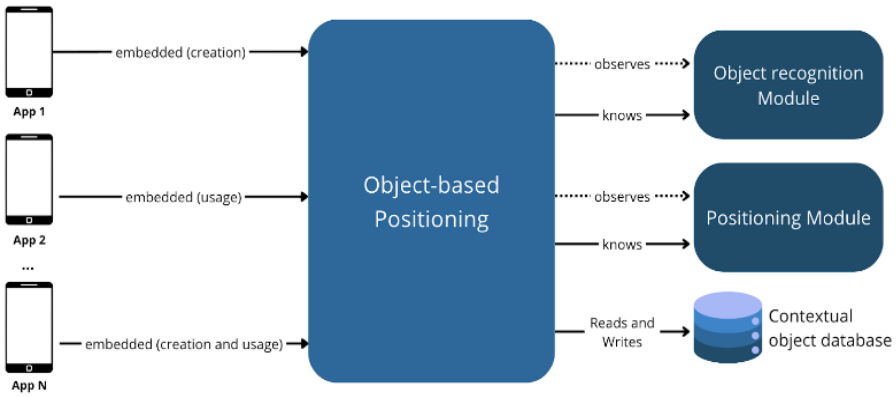

Fig. 1. Proposed Architecture for *Object-based Positioning*.

The *Object Recognition* module recognises objects in the environment through the device's camera. It can know several models, allowing it to be extensible, but the module only uses one to perform the recognition at a given time. Figure 2 depicts the flow involved in this module, which implies three specific steps.

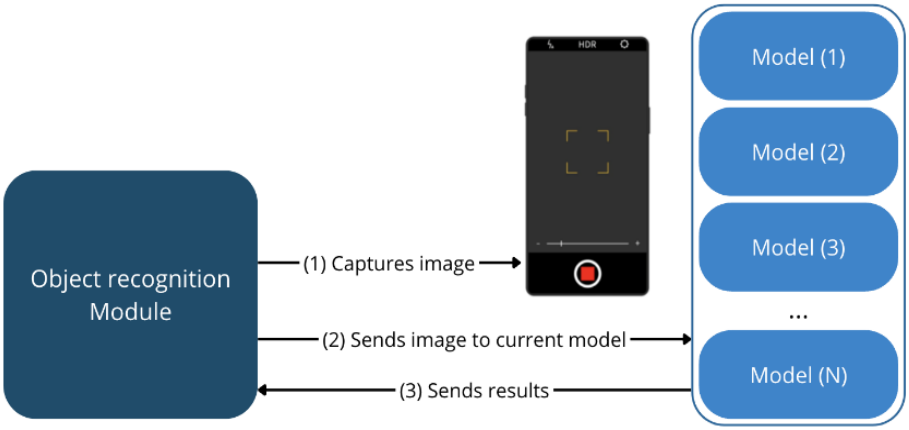

Fig. 2. Flow involved in the *Object Recognition* module.

To provide a clear understanding of the three steps of the *Object Recognition* module depicted in Figure 2, they are detailed below, using the same enumeration as in the figure:

1. The current image is obtained, which involves establishing communication with the device's camera to capture the view at that moment.

2. The captured image is sent to the current object recognition or detection model. Depending on the chosen model, some pre-processing may be necessary to adapt the image to the format expected by the model. Additionally, the *Object Recognition* module allows the configuration of the model to be used and the selection of the objects of interest during the inference process, which is especially relevant when employing *Object-based Positioning*.
3. Finally, the corresponding result is received once the model inference is performed. It is important to highlight that the format and structure of the response obtained in the inference may vary from one model to another. Therefore, before returning a final response, the *Object Recognition* module standardises the response to a standard interface. This facilitates communication with different object recognition and detection models.

On the other hand, the *Positioning module* aims to obtain information about the user's location using various positioning sensing mechanisms, as shown in Figure 3. This module is designed to constantly monitor changes in the selected sensing mechanism, allowing it to detect events such as user movement and request updated location information as needed. The module provides a standardisation layer regarding the response provided by each sensing mechanism to ensure coherence and uniformity of location data. The chosen sensing mechanism may depend on the required accuracy, infrastructure availability, and environmental limitations.

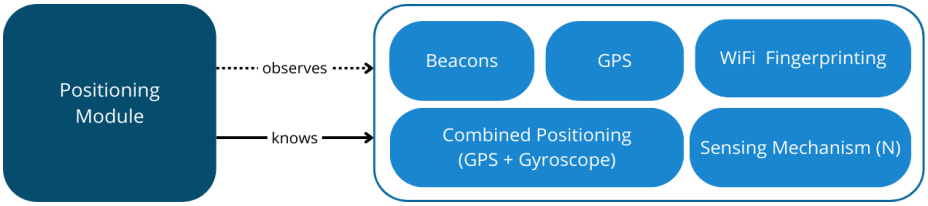

Fig. 3. Representation of the *Positioning* module.

4 An implementation of the Object-based Positioning

This section presents a concrete implementation of the *Object-based Positioning* module from the architecture proposed in Section 3. This module was implemented as a library[3] for the *React Native* framework (which can create cross-platform applications for Android and iOS with the same source code). Note that the *Object-based Positioning* library can be embedded by any application, as presented in Figure 1.

React Native offers various modules and libraries that facilitate integration with native phone functionalities. These include access to the device's camera through the *Re-*

[3] The project's source code is available at the following GitHub repository: https://github.com/francoborrelli/object-based-positioning-library. This repository contains all the documentation and files needed to understand and utilize the *Object-based Positioning* library.

act Native Camera library and the execution of object recognition models with the specific *Tensorflow* library for *React Native*. In particular, the library has been implemented to allow the use of two models based on *MobileNet*. The first one is a model for recognising doors (trained by one of the authors of this paper), and the other is a pre-trained *TensorFlow* model that can recognise various general objects.

Notably, the *Positioning module* was chosen to explore the combination of GPS and Gyroscope. This does not require the installation or presence of additional infrastructure in the physical environment; that is, everything is solved from the mobile device, which simplifies the use of this architecture in any physical location. This facilitates obtaining a generic solution that could be used in any environment. *React Native* has the *React Native Geolocation* and *React Native Sensors* libraries, which allow access to the device's GPS and sensors (such as the Gyroscope), respectively.

Firestore was chosen for the contextual database. It is a NoSQL cloud database from *Google Firebase*[4] that offers efficiency, scalability, and an SDK for multiple platforms. Thus, a *Firestore* database is used to store locations that involve data collected from the object recognition and positioning modules. Figure 4 shows the technologies used for the implemented *Object-based Positioning*.

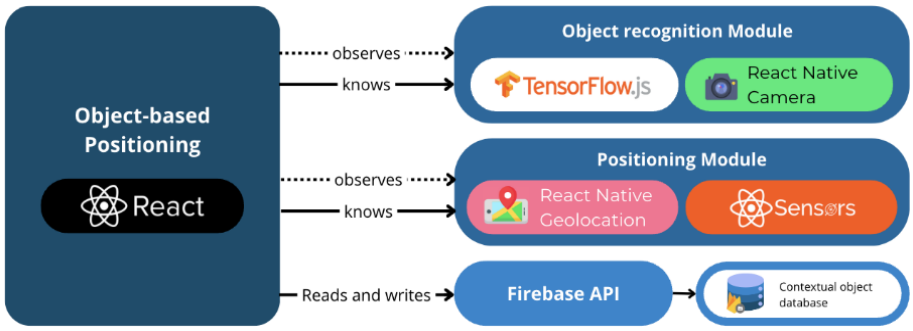

Fig. 4. Technologies used in the *Object-based Positioning* library.

5 Practical application of the Object-based Positioning library

The topic of positioning has been previously explored by the authors of this paper. In particular, in earlier works, positioning was investigated using Wi-Fi network fingerprinting to locate the user in in-situ co-design authoring tools [6, 7]. Based on this, it was decided to continue with this line of research, but in this paper, the implemented *Object-based Positioning* library is used as the positioning mechanism. Below, we detail how the *Object-based Positioning* library was embedded in an in-situ co-design authoring tool, and then we exemplify its usage, emphasising the lessons learned.

[4] Firebase. https://firebase.google.com, last accessed 2024/03/15

5.1 Object-based Positioning library embedded in an in-situ co-design tool

The *Object-based Positioning* library was embedded into an in-situ co-design tool, which used the two functionalities provided by this kind of positioning, such as *'creation'* and *'usage'* (described in Section 3). This tool focuses, in particular, on co-designing (creating) *Positioned Information* using object-based positioning to subsequently provide (use) this information when a user is in that location. For more details on the functioning of in-situ co-design tools, [6] and [7] can be consulted.

Some configurable parameters related to *Object-based Positioning* were added to the tool to facilitate an easy way to change these values through an interface. This allows the creation of different scenarios to explore how the library behaves with different values. These configurable parameters are:

- *Recognition or detection model to use at one point in time*. It can choose from the two models based on *MobileNet* (one focuses only on doors, and the other is oriented to general objects), as mentioned in Section 4.
- *Classes of interest*. Depending on the selected model, it can choose among the available objects, which the object-based positioning will consider.
- *The minimum precision adjustment in inferences*. This parameter allows setting the minimum level of precision necessary for the model outputs to be considered valid inferences. The default value is 20%.
- *The number of frames.* This parameter controls how recognition models operate, specifying inference on 1 out of every N frames from the phone camera. Lowering it will increase the frequency of inferences, though resource consumption will also rise. The default value is 100 frames.
- *The maximum distance in meters to consider an object is 'near' to the user.* An object detection is only considered valid if the user is at a distance less than or equal to the indicated amount of meters of this parameter. GPS information is used to obtain the user's current location. The default value is 1 meter.
- *The degrees to determine that an object is 'in the viewing angle' of the user*. It is evaluated if, with the phone's current orientation plus this degree parameter, it is possible to *'see'* a detected object according to the viewing angle registered for this object. Gyroscope information is used to obtain the user's current orientation. The default value is 70 degrees.

5.2 Usage of the Object-based Positioning library - Single Object Class

This prototypical experience was designed to assess the functionality of *Object-based Positioning* when registering *Positioned Information* using multiple objects of the same type. To achieve this, the parameter *'Recognition or detection model to use at one point in time'* was configured with the door recognition model presented in Section 4. For this experience, *Positioned Information* was created using the domain of a hypothetical conference that could take place between classrooms 1-2 to 1-4 on the first floor of the Faculty of Informatics at UNLP, assigning a specific track (topic) to each classroom.

Then, it could use this *Positioned Information* (for each of these tracks) to provide them to the users when they were in those locations.

Specific aspects were established to observe how the library behaved in response to nearby doors (classrooms 1-2 and 1-3 of the faculty), such as variation in viewing angles when registering information, changes in user orientation, and walking speed. It should be noted that some of these aspects to observe emerged from our previous knowledge using other location mechanisms.

During the *'creation'* phase, two participants were asked to load *Positioned Information* for the different classrooms into two distinct Workspaces[5]. In one, the information was loaded facing the doors, while in the second, different angles were used. Upon observing the recorded information for both Workspaces, a positioning discrepancy between the devices used (Xiaomi 12 and Samsung S22) was noticed. To investigate this issue further, a direct comparison was made between the latitude and longitude readings of both devices, as shown in Figure 5. This test revealed a 6-meter difference between the locations recorded by the two devices, even though they were physically next to each other.

Several factors were identified that could have contributed to the discrepancies in GPS values (shown in Figure 5), such as antenna positioning, the type of GPS technology used, GPS calibration, the devices' power-saving mode, among others. Solving this is beyond the scope of this paper.

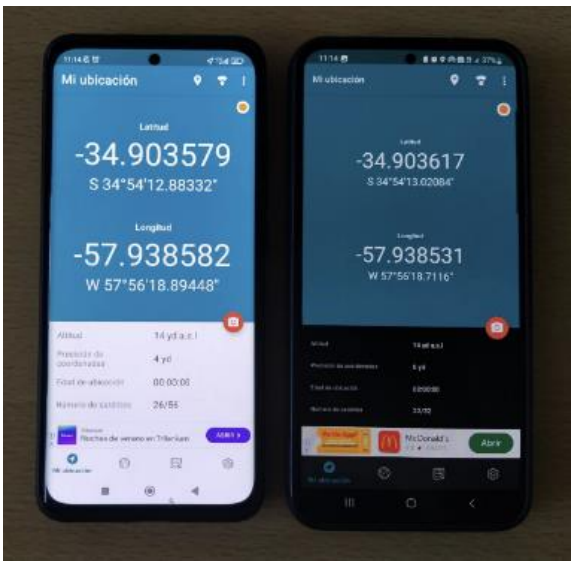

Fig. 5. Comparison of GPS performance between Xiaomi 12 (left) and Samsung S22 (right) devices.

[5] Each Workspace contains all the information co-designed within an application. Due to the nature of this tool (for in-situ co-design), each Workspace is linked to a building or physical location.

Considering these disparities with the GPS, it was decided that a single participant (the one who had the minor discrepancy in positioning) would be responsible for loading all the *Positioned Information* since the focus was on testing the library rather than *'improving'* GPS performance.

When the selected participant finished loading all the *Positioned Information*, both participants took part in a *'usage'* phase. During this last stage, some interesting points were detected that are worth mentioning:

- With the default settings, which include a *'The maximum distance in meters to consider an object is near to the user'* of 1 meter and a viewing angle aperture of 70 degrees, participants were asked to slowly approach the classroom doors to see if they could access the information. Only the participant who had loaded the information could access it by doing this. The other participant experienced difficulties, which were partly expected given the discrepancies detected with the GPS. This was done in both Workspaces, where the same outcome was experienced.
- Based on the previous findings, the parameter '*The maximum distance in meters to consider an object is near to the user*' was changed from 1 to 3 meters. With this adjustment, both participants could access the information from all classrooms correctly in both Workspaces.
- The other test involved participants being asked to approach the classroom doors at a much faster pace. The result was the same in both Workspaces. It was identified that objects cannot be recognised when users move at very high speeds, as it requires at least a few seconds for the recognition model to process and infer the object.
- As in all previous tests, participants could access the correct information for each classroom. It was desired to determine at what distance the library began to mix information, for example, between the content of classrooms 1-2 and 1-3, which are physically very close to each other. Only when reaching a vast *'The maximum distance in meters to consider an object is near to the user'* setting (10 meters) did a problem of information confusion start to appear.
- An analysis was conducted on how the viewing angle aperture affected the detection of *Positioned Information*. Two parameters were adjusted for this purpose: the '*The maximum distance in meters to consider an object is near to the user'* was set to 3 meters in both Workspaces and the *'The degrees to determine that an object is in the viewing angle of the user'* was reduced to 30 degrees; when the tests carry on revealed that with a 30-degree aperture, it was no longer possible to detect the information.
- Based on the previous findings, the *'The degrees to determine that an object is in the viewing angle of the user'* was increased to 45 degrees. In this case, participants could detect the information if they positioned themselves precisely in the same place with the same angle used to register the information.

Images and screenshots were taken during these tests. Figure 6.a reflects the '*creation'* stage, where it can be observed how the tool can recognise different elements of the environment (in this case, doors). The user can choose one of the objects identified by the model and load *Positional Information* related to it. In Figure 6.a, the loading form can also be seen along with some positioning data (such as the object detected and

latitude and longitude information provided by the GPS; the Gyroscope angle is not seen because it is below the screen but is considered too). Figure 6.b presents an example of '*usage'*, where the user approaches the registered door and accesses its content.

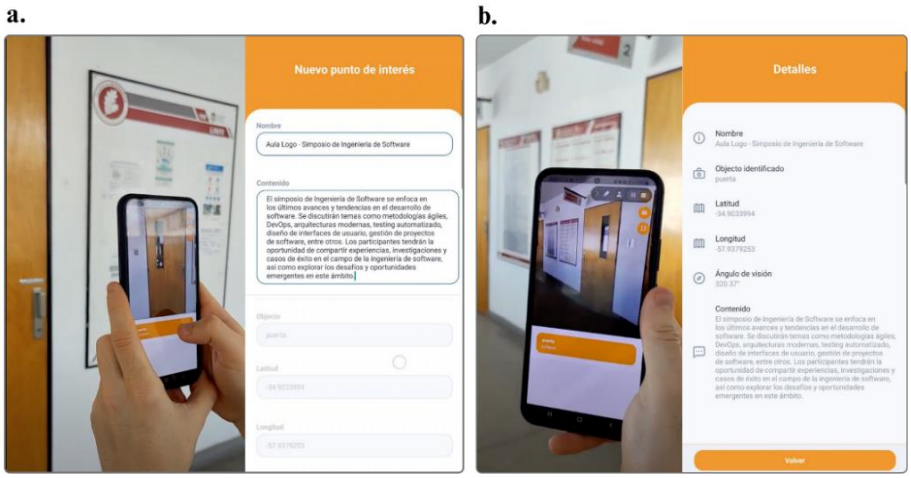

Fig. 6. Images and screenshots taken during the *Object-Based Positioning* library testing.

5.3 Usage of the Object-based Positioning library - Multiple Object Classes

This experience aimed to develop a hypothetical application for children visiting an informatics exposition. To simulate this, a booth was set up with various technological objects to offer interesting facts about them (see Figure 7.a). The objective was to analyse the behaviour of the *Object-based Positioning* library when different objects are used in a reduced space. For this experience, two participants were also involved: the first with a high-end phone (Samsung S22) and the second with a mid-range phone (Xiaomi Mi A3).

Before starting, a Workspace was defined, and the parameter *'Recognition or detection model to use at one point in time'* was set with the pre-trained *MobileNet* model capable of recognising various general objects. This model can recognise laptops, mice, cellphones, hard drives, etc. Only the objects in Figure 7.a were selected for the parameter *'Classes of interest'*. This decision was made to prevent unwanted objects from being detected with the tool.

During the *'creation'* phase, one participant loaded the *Positioned Information* with the tool, considering the issue detected in Section 5.2 with the GPS accuracy. Although the GPS accuracy was not precise even with the same device, it remained within acceptable limits; this margin of error is shown in Figures 7.a and 7.b. Figure 7.a represents how the objects were actually distributed, while Figure 7.b shows how the *Positioned Information* was recorded in the tool. Additionally, in Figures 7.c and 7.d, two examples of detections with the tool are presented, showing the identified object's name and the model's corresponding precision percentage.

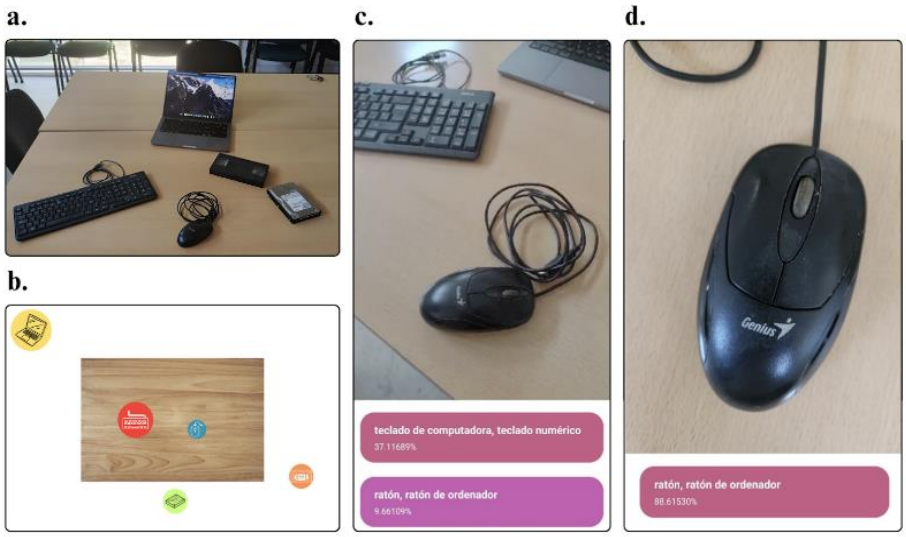

Fig. 7. Testing the library with multiple objects.

During the *'usage'* phase, both participants were asked to try accessing the *Positioned Information* while facing the stand. Several tests were conducted, and some interesting points were detected that are worth mentioning:

- Both participants encountered difficulties with the default settings (defined in Section 5.1). This was due to the previously identified GPS issues, which prevented them from consistently accessing the *Positioned Information*. Specifically, the participant who defined the information could only recognise some objects, while the others failed to recognise any of the objects.
- Based on the previous results, the parameter '*The maximum distance in meters to consider an object is near to the user*' was changed from 1 to 3 meters. Both participants were able to access all the information with this setting.

These results, combined with the experience of Section 5.2, revealed that '*The maximum distance in meters to consider an object is near to the user'* of 3 meters is the most optimal. Furthermore, it was interesting to test how the recognition model performed with different configurations of *'The minimum precision adjustment in inferences'* and *'The number of frames'*. Below is a description of what was discovered:

- When using the '*The minimum precision adjustment in inferences'* configuration with its default value (20%), recognition of all objects was achieved, albeit with a delay of approximately 5 seconds.
- However, by reducing '*The minimum precision adjustment in inferences'* to 10%, recognition occurred almost instantly, with precision levels close to 17%.
- Upon further analysis of the *'The minimum precision adjustment in inferences'*, it was observed that the accuracy of the model gradually increased over time, always

starting with a very low value (see Figure 7.c) and gradually increasing within seconds to much higher values (between 40% and 80% in most cases), as shown in Figure 7.d.

- In addition, it was found through another test that the time window during which the model achieves high precision percentages can be significantly reduced by modifying *'The number of frames'* configuration. This configuration was changed from its default value (100 frames) to a much lower one (10 frames) in the conducted test. This caused the tool to instantly generate results with higher precision when using the Samsung S22 device. However, this modification caused the application to experience inevitable hitches with the Xiaomi Mi A3, as this phone is a mid-range device with 4GB of memory.

6 Conclusions and Future works

The implemented tool, which embeds our *Object-based Positioning* library, has demonstrated the practical viability of the proposed general architecture for indoor positioning using object recognition models. Although preliminary in nature, the conducted tests have provided valuable insights into the potential adoption of this positioning approach. During these tests, it was identified that the implemented *Object-based Positioning* solution works for recognising different objects and detecting objects of the same type in limited spaces. Furthermore, the tests confirmed that the solution can operate effectively on high-end and mid-range devices.

It should be noted that depending on the space's features, the objects within it, and the devices being used, the settings of the implemented library may need to be adjusted to achieve optimal performance. For instance, it was found that adjusting with low values of the *'The number of frames'* configuration on a high-end device resulted in significantly better performance but led to operational issues on mid-range devices.

The Gyroscope's functionality was tested, and it was found that the phone's Gyroscope was effective in disambiguating nearby information. In particular, a *'The degrees to determine that an object is in the viewing angle of the user'* configuration of 70 degrees yielded satisfactory results. Note that, with these few tests, we cannot reach conclusive results, so more exploration is needed to confirm this preliminary discovery.

Concerning GPS functionality, it was noted that the performance can exhibit variations across different devices even when they are physically situated in the same exact location. Although GPS issues were detected, the proposed solution is a viable alternative. Related to this, the parameter *'The maximum distance in meters to consider an object is near to the user'* should be configured with 3 meters to get optimal results.

As a future direction, it would be valuable to investigate the implementation of the *Object-based Positioning* solution with other technologies such as Wi-Fi positioning, Bluetooth, or proximity sensors, considering the limitations identified with GPS. This could hold promise for enhancing the accuracy and reliability of indoor positioning, but other issues could emerge. For example, the feasibility of this future implementation will necessitate an assessment of additional infrastructure requirements, including their cost-effectiveness and scalability across various physical spaces.

References

1. Alegre, U., Augusto, J.C., Clark, T.: Engineering context-aware systems and applications: A survey. Journal of Systems and Software, 117, 55-83 (2016). doi: 10.1016/j.jss.2016.02.010
2. Alegre-Ibarra, U., Augusto, J. C., Evans, C.: Perspectives on engineering more usable context-aware systems. Journal of Ambient Intelligence and Humanized Computing, 9(5), 1593-1609 (2018). doi: 10.1007/s12652-018-0863-7
3. Xiao, A., Chen, R., Li D., Chen, Y., Wu, D.: An Indoor Positioning System Based on Static Objects in Large Indoor Scenes by Using Smartphone Cameras. Sensors (Basel, Switzerland), 18(7), 2229 (2018). doi: 10.3390/s18072229
4. Kaulich, T., Heine, T., Kirsch, A.: Indoor Localisation with Beacons for a User-Friendly Mobile Tour Guide. KI-Künstliche Intelligenz, 31(3), 239-248 (2017). doi: 10.1007/s13218-017-0496-6
5. Borrelli, F.M., Brost, P., Challiol, C., Orellano, D.H., Mendibru, F.I., Santoleri, M.A., Alconada Verzini, F.M.: Desarrollo multiplataforma de Aplicaciones Móviles combinadas con el uso de Beacons. In: XXIV Congreso Argentino de Ciencias de la Computación (CACIC 2018), pp. 847-856. RedUNCI, Tandil (2018).
6. Challiol, C., Borrelli, F.M., Mendiburu, F.I., Rouaux Servat, C.M., Goin Plexevi, F., Orellano, D.H., Gomez-Torres, E., Gordillo, S.E.: Design Thinking's resources for in-situ co-design of mobile games. In: 2019 International Conference on Information Systems and Computer Science (INCISCO 2019), pp. 339-345. IEEE, Quito (2019). doi: 10.1109/INCISCOS49368.2019.00060
7. Challiol, C., Borrelli, F.M., Mendiburu, F.I., Goin Plexevi, F., Rouaux Servat, C.M., Orellano, D.H., Gomez-Torres, E., Gordillo, S.E.: Co-diseño in-situ de Juegos Móviles usando un abordaje con recursos de Design Thinking. Enfoque UTE 11(1), 1-14 (2019). doi: 10.29019/enfoque.v11n1.586
8. Zaidi, S., Ansari, M. S., Aslam, A., Kanwal, N., Asghar, M., & Lee, B.: A survey of modern deep learning based object detection models. Digital Signal Processing, 126, 103514 (2022). doi: 10.48550/arXiv.2104.11892
9. Xu, R., Zhang, C. L., Wang, P., Lee, J., Mitra, S., Chaterji, S., Bagchi, S.: ApproxDet: content and contention-aware approximate object detection for mobiles. In: Proceedings of the 18th Conference on Embedded Networked Sensor Systems, pp. 449-462. ACM, Japan (2020). doi: 10.1145/3384419.3431159
10. Zou, Z., Chen, K., Shi, Z., Guo, Y., & Ye, J.: Object detection in 20 years: A survey. Proceedings of the IEEE, 111(3), 257-276 (2023). doi: 10.1109/JPROC.2023.3238524
11. Estrada, J., Paheding, S., Yang, X., & Niyaz, Q.: Deep-learning-incorporated augmented reality application for engineering lab training. Applied Sciences, 12(10), 5159 (2022). doi: 10.3390/app12105159
12. Shafer, S.A., Krumm, J., Brumitt, B., Meyers, B., Czerwinski, M., & Robbins, D.C.: The New EasyLiving Project. Microsoft Research (1999).
13. Bagchi S., Aggarwal V., Chaterji S., Douglis F., El Gamal A., Han J., Henz B., Hoffmann H., Jana S., Kulkarni M.: Vision Paper: Grand Challenges in Resilience: Autonomous System Resilience through Design and Runtime Measures. IEEE Open Journal of the Coputer Society 1, 155-172 (2020). doi: 10.48550/arXiv.1912.11598