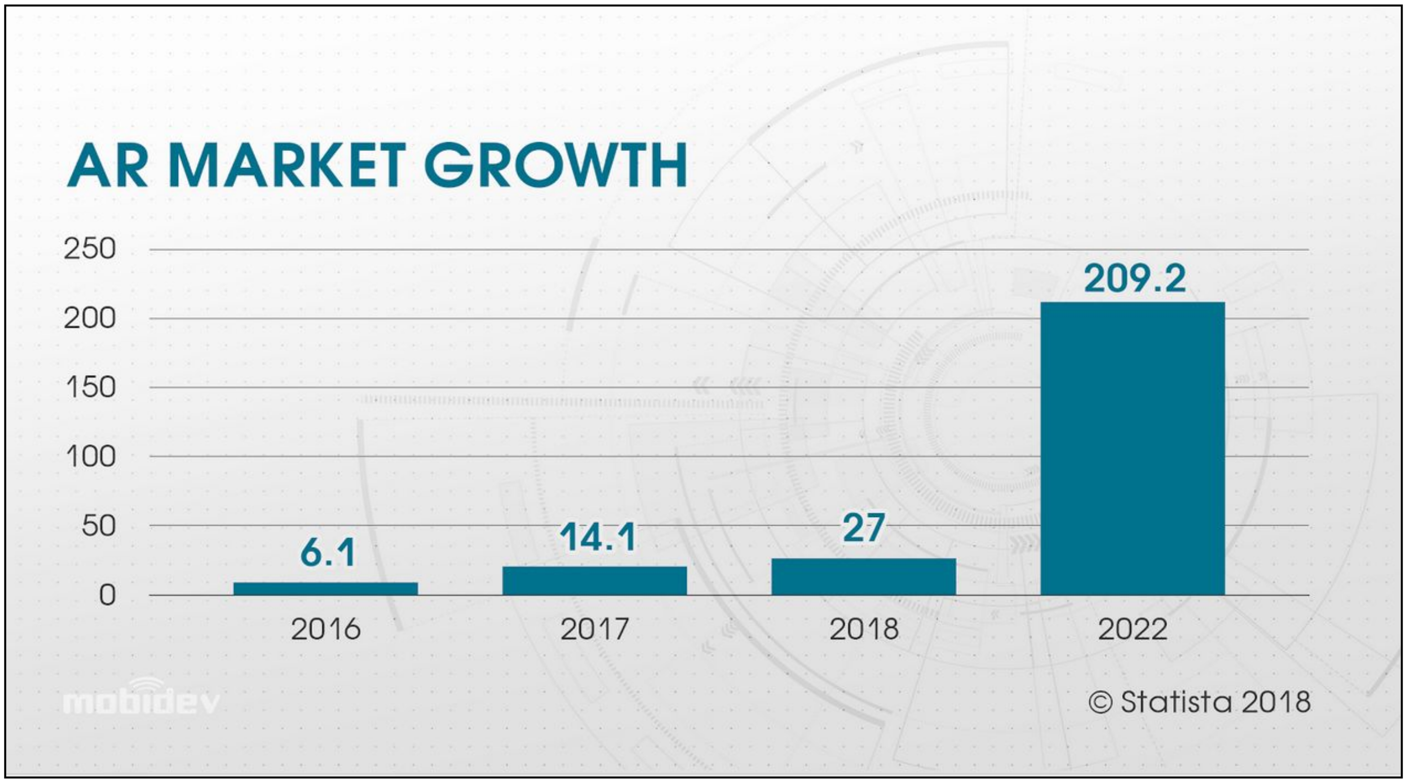
How to Overcome Challenges of ARKit Indoor Navigation App Development

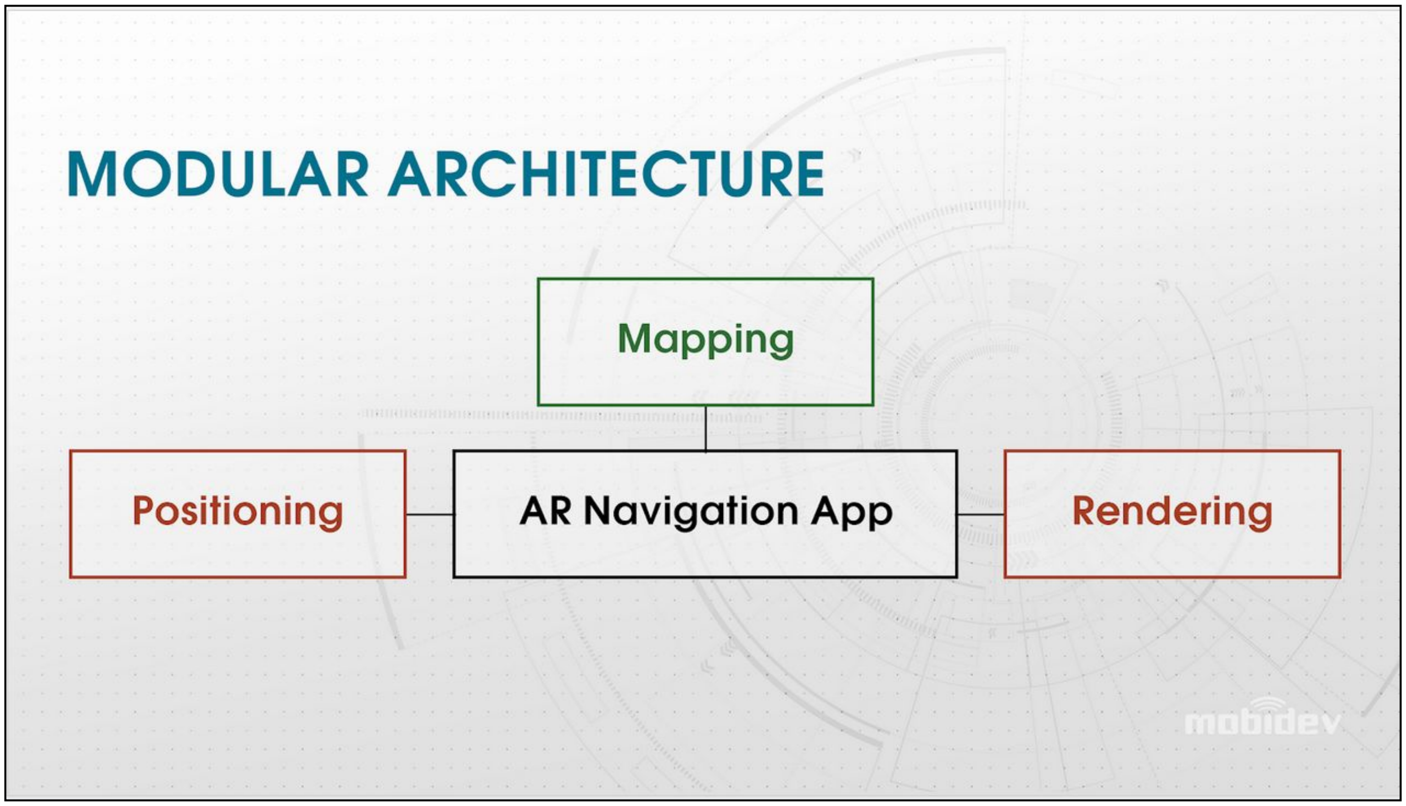
Andrew Makarov

Head of Mobile Development at MobiDev

One of the notable recent forays into the field of mobile augmented reality was the release of Apple's ARKit in 2017. Since then, a number of related projects in the field of AR have been developed, furthering advancements and growth of AR solutions spanning across multiple industries and markets. AR-based indoor navigation is one of the most popular but still a relatively unexplored niche, as the technology is far from being perfect at present, and it's necessary to develop greater flexibility and scalability to enable it to flourish.



In this case we use a modular solution that allows us to upgrade or update any of these components independently of the others. The solution includes three modules: Mapping, Positioning, and Rendering.



Mapping is the one which has been studied most thoroughly. And the concept of mapping is simple – If we possess both the map and a set of coordinates, we can create a route.

Positioning is slightly trickier. Currently, there's no standardized and easily implementable way to find the exact coordinates of a user who is indoors.

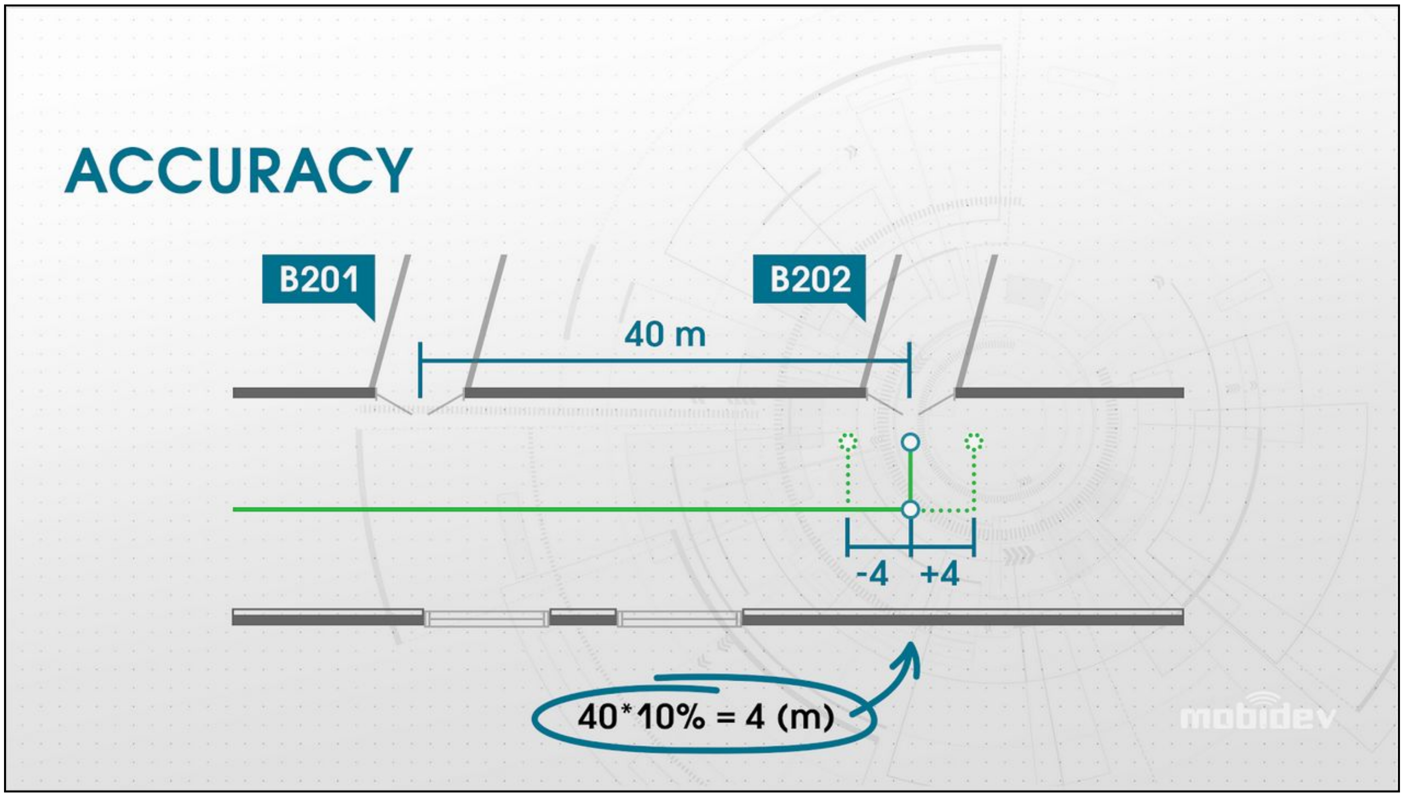
Rendering, in turn, is dependent upon the accuracy from which Positioning can be determined. It's easy to render a route in 3D, in the abstract. But it's more challenging to make sure that the virtual is matching up with the real world, accounting for obstacles and objects like doors, walls and any furniture or other large items.

Rendering will continue to improve as ARKit continues to evolve. And we're also able to alter the design of AR content to our specifications.

At present, AR Indoor Navigation faces the following challenges:

First, we must grapple with the challenge of finding the user's precise location within a building, including their floor within a multi-story building.

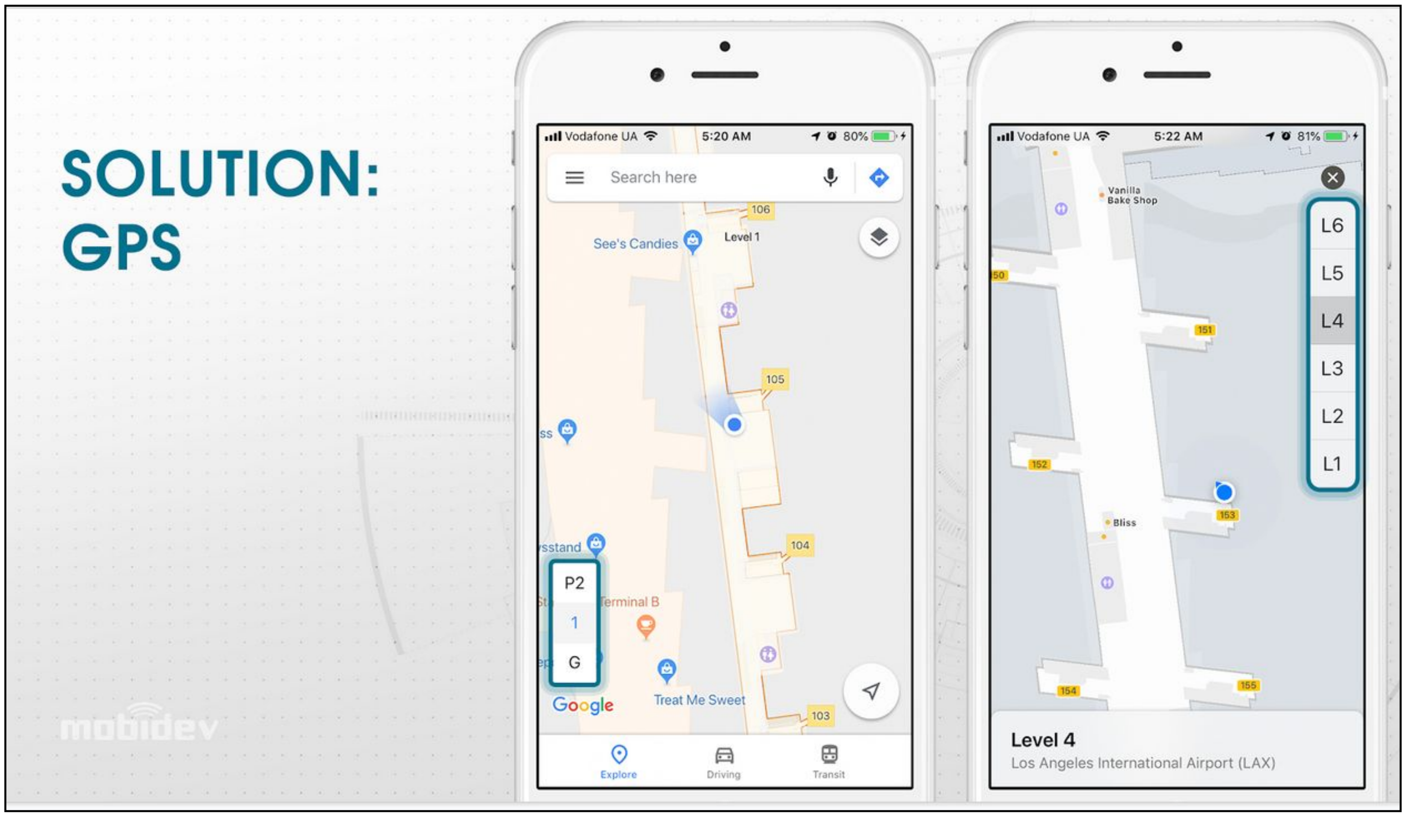
It turns out that we require a minimum accuracy in AR function that equals around 10% of the distance between various points of the destination. To give one example, let's say there's a distance of 40 meters between halls within a convention center. We would require AR navigation accuracy of 40*0.1 = 4 meters. In a campus or an office building where doors in a hallway are 5 meters from one another, we'd require 4*0.1 = 0.4 meters, or half a meter.



The next question is how can we get this level of accuracy?

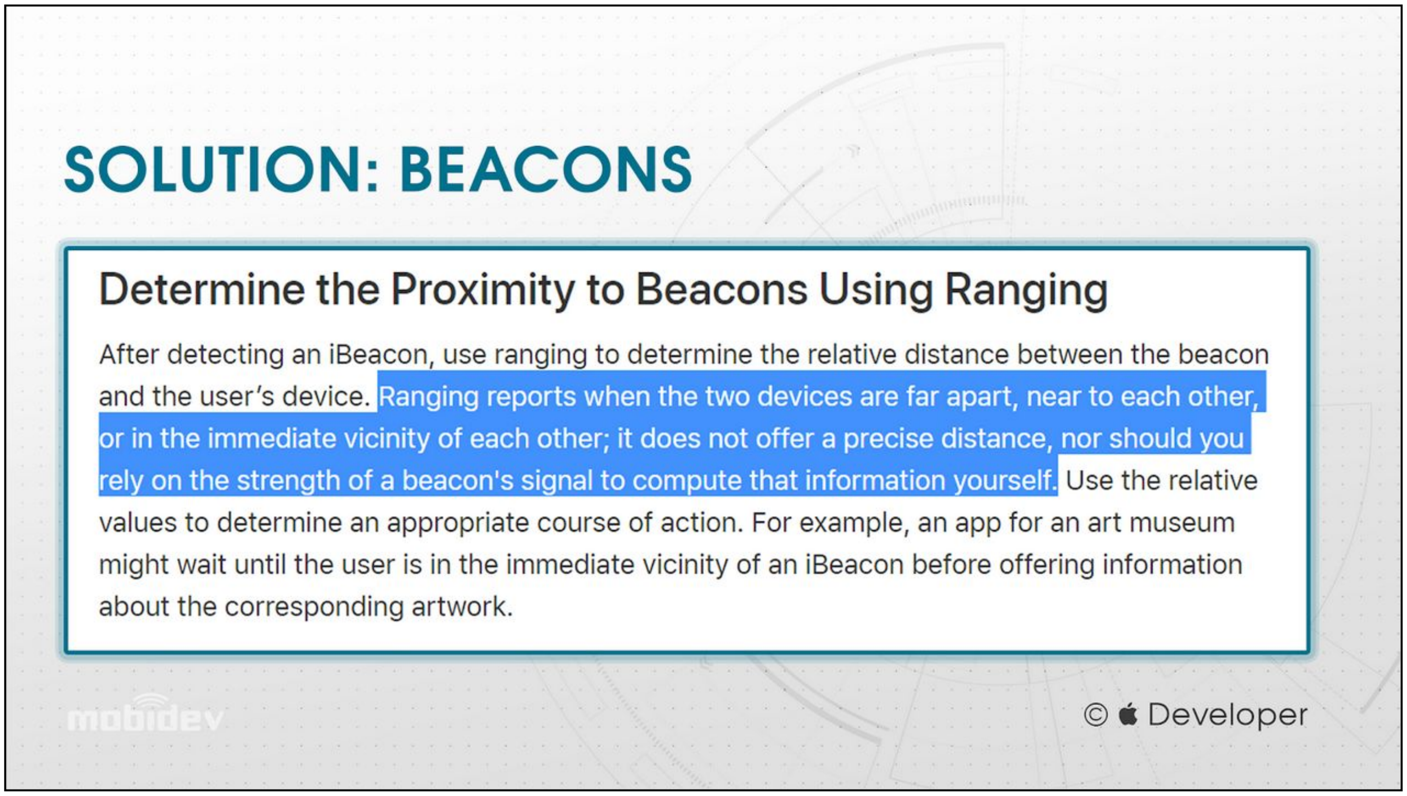
GPS

Indoor GPS isn't a suitable technology for this type of implementation. While it functions relatively well in low-rise buildings and larger buildings like airports, it faces a problem of determining a user's floor level, and it requires manual selection to work properly. We can't find the precision we need with GPS.



Beacons

Beacons are often a technology people bring up when discussing indoor navigation. However, iBeacon documentation from Apple states that beacons are only able to provide an approximate distance, and that it's dangerous to rely on beacon signal strength in order to calculate positioning data.


SOLUTION: BEACONS

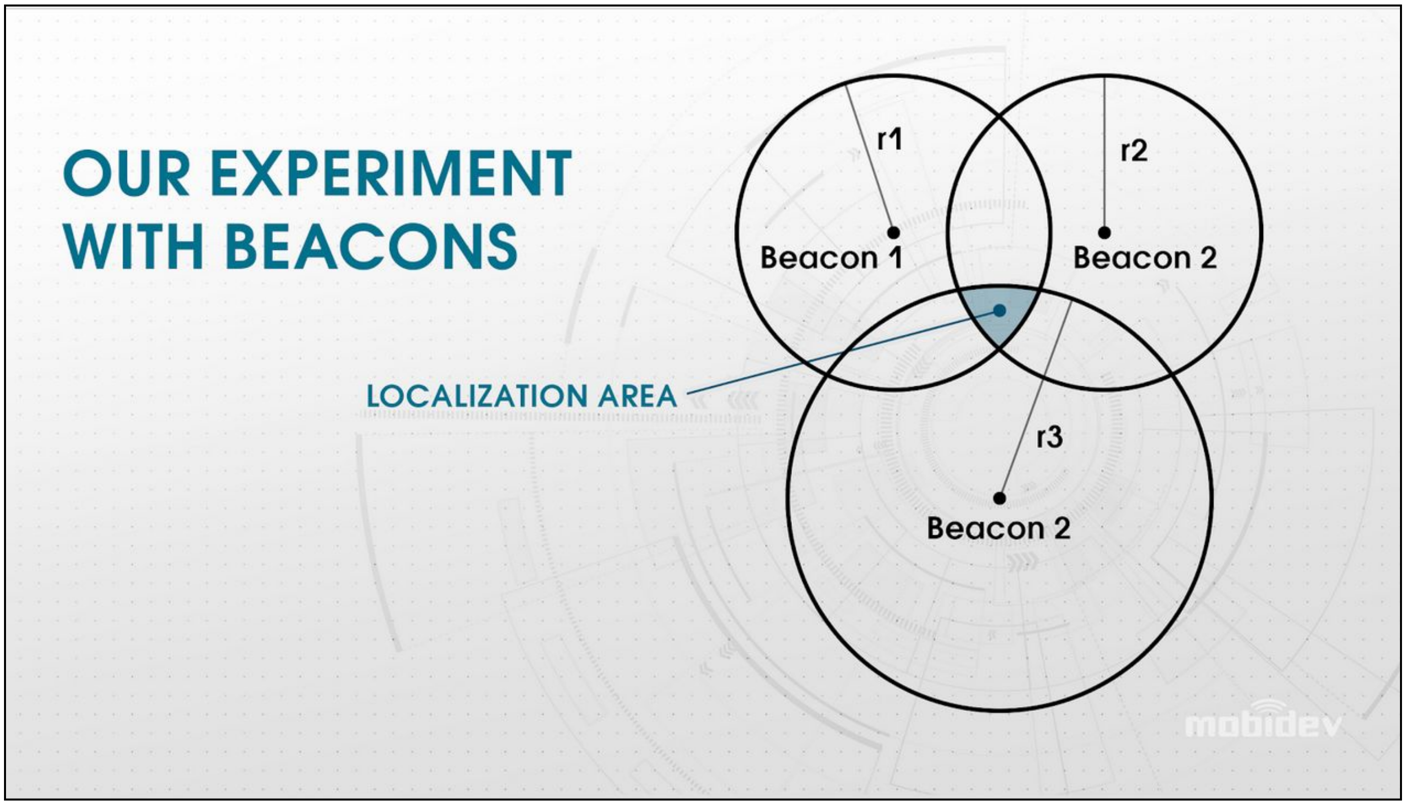
Determine the Proximity to Beacons Using Ranging

After detecting an iBeacon, use ranging to determine the relative distance between the beacon and the user's device. Ranging reports when the two devices are far apart, near to each other, or in the immediate vicinity of each other; it does not offer a precise distance, nor should you rely on the strength of a beacon's signal to compute that information yourself. Use the relative values to determine an appropriate course of action. For example, an app for an art museum might wait until the user is in the immediate vicinity of an iBeacon before offering information about the corresponding artwork.

mobidev

© Developer

We decided to run an experiment with a few beacons: created a prototype, researched the algorithms of true range multilateration and smoothing, applied some work with the accelerometer and gyroscope. It appears that the beacons can reach around 7 to 8 meters in accuracy. With some more complicated algorithms we reached an accuracy of 5-6 meters. However, it doesn't appear possible to get the accuracy down any finer with the current implementation of beacon technology.



As a result, beacons don't appear to be a viable option for AR Indoor Navigation when high accuracy of positioning information is required. Some other solution will be considered.

Visual Positioning System

VPS (Visual Positioning System) is a positioning technology which shows some promise for deployment within an AR solution. Google uses Street View data for this technology, locating buildings in the area as their reference points. The ARKit 2.0 offers a class called ARWorldMap for this type of purpose. The ARWorldMap class offers a set of feature points surrounding the users that can be identified and used to determine positioning.

SOLUTION: VISUAL POSITIONING SYSTEM

© ai.googleblog.com

In practice, ARWorldMap is a cloud of feature points created by a scan of the surrounding area. This data is then written onto the map, and then any future launches will reference the ARWorldMap instances which had previously been saved. Once we recognize the area, we can receive accurate coordinates.

However, there are some issues with using ARWorldMap for determining the precise location of a user. First, many buildings and indoors spaces look extremely similar from a visual standpoint. It’s simply not possible to use visual cues to determine one from the other in some cases. Further, interior settings can change visually as time goes on, which ‘tricks’ the ARWorldMap. And finally, these types of calculations can be processor-intensive, making the process too slow to be relied upon.

VPS may be a solution in the future, as the technology matures. But right now, we must look elsewhere.

Visual Markers

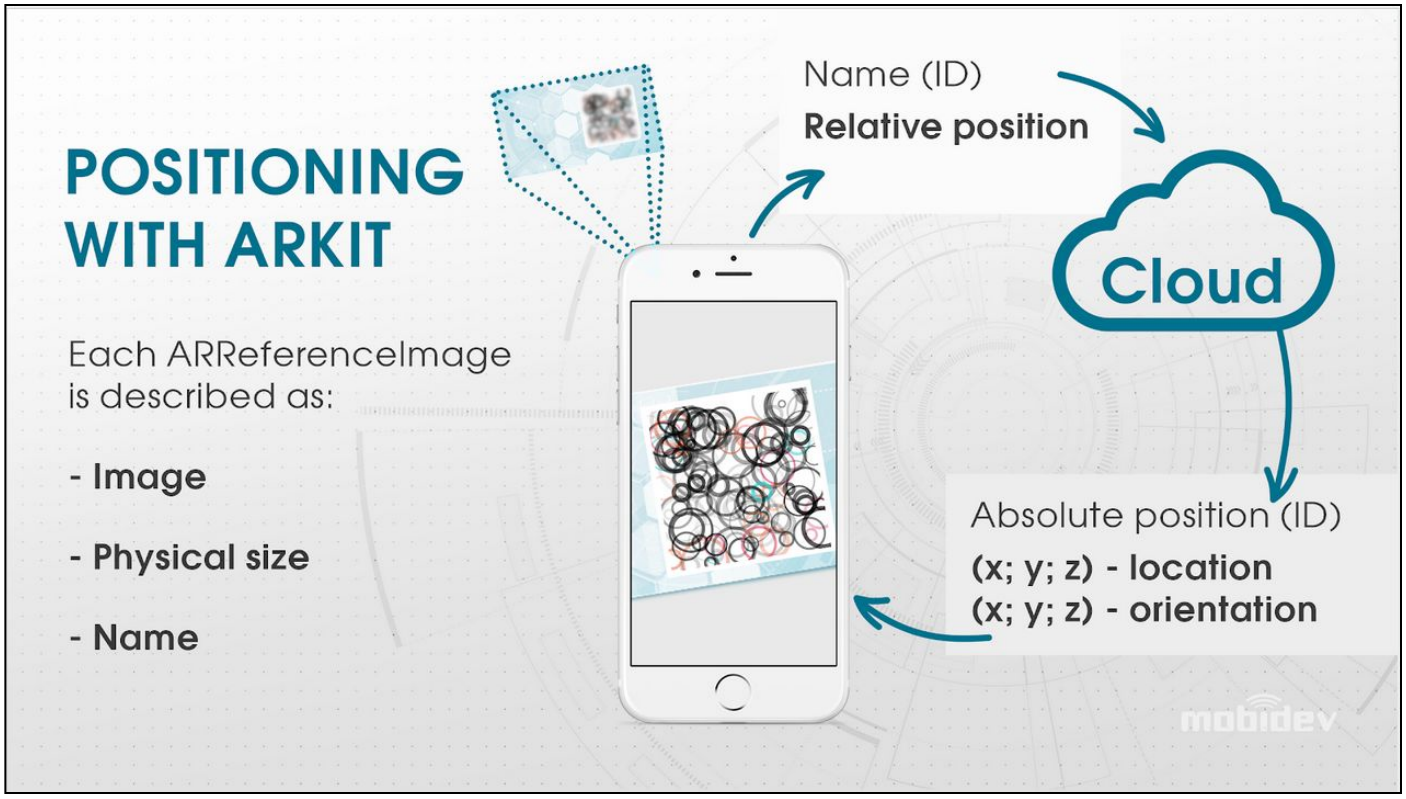
A visual marker, aka an AR Marker or a Reference Image, is an image which ARKit 2.0 recognizes. These visual markers can inform ARKit 2.0 where it should be placing any AR content. In effect, they tell ARKit 2.0 where an AR element should be rendered within the overall environment.

If a visual marker is fixed within a space, for example on a wall or on a floor surface, this precise location can be stored in the cloud. Then, when the time comes to scan, we have the real world coordinates of that particular location.

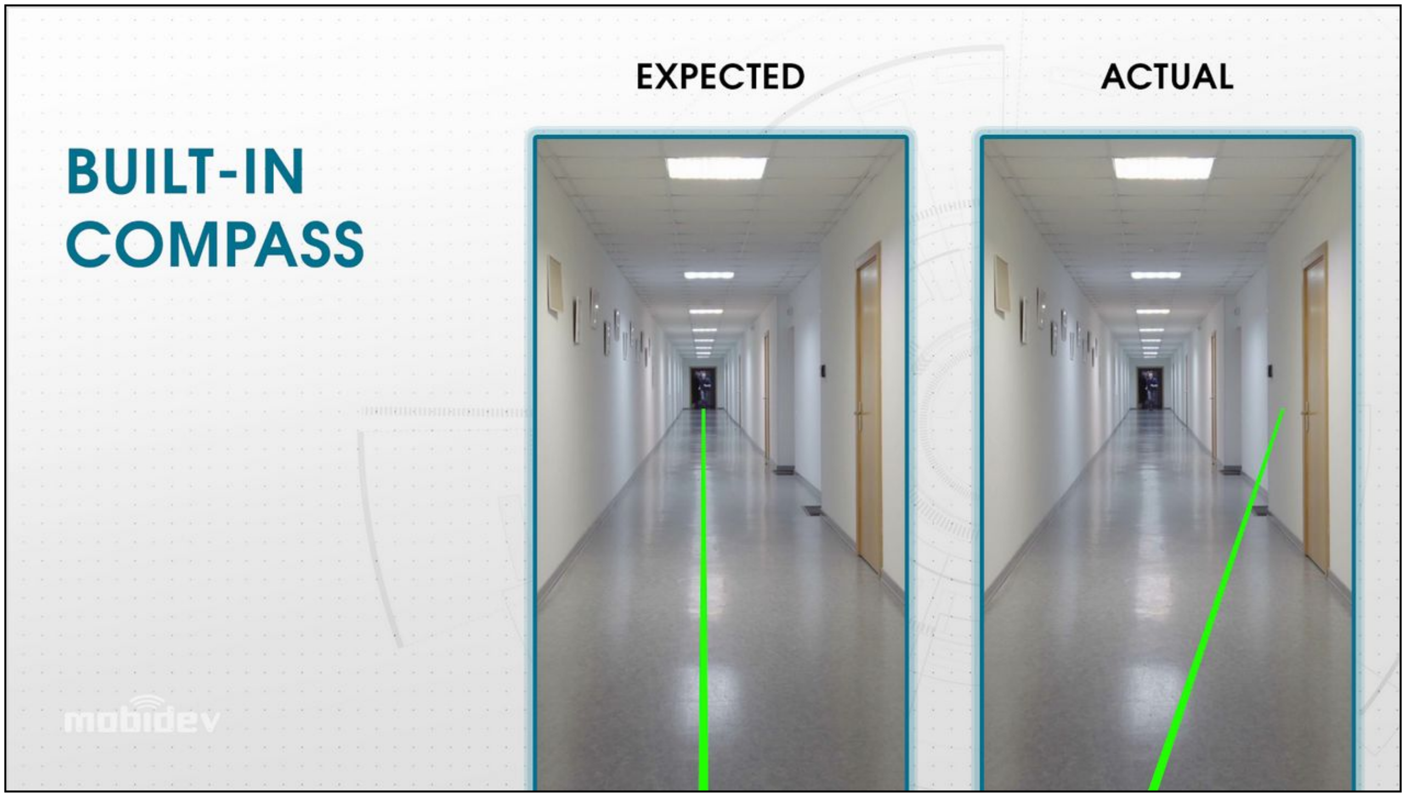
This type of visual marker-based positioning can deliver extremely accurate ranges. In some cases, as precise as within a few millimeters. This process is akin to VPS, so I would name it SVPS, or Simplified Visual Positioning System. The concept is similar, but we require far less data in order to process locations.

The process of finding a user's positioning via ARKit 2.0 is as follows:

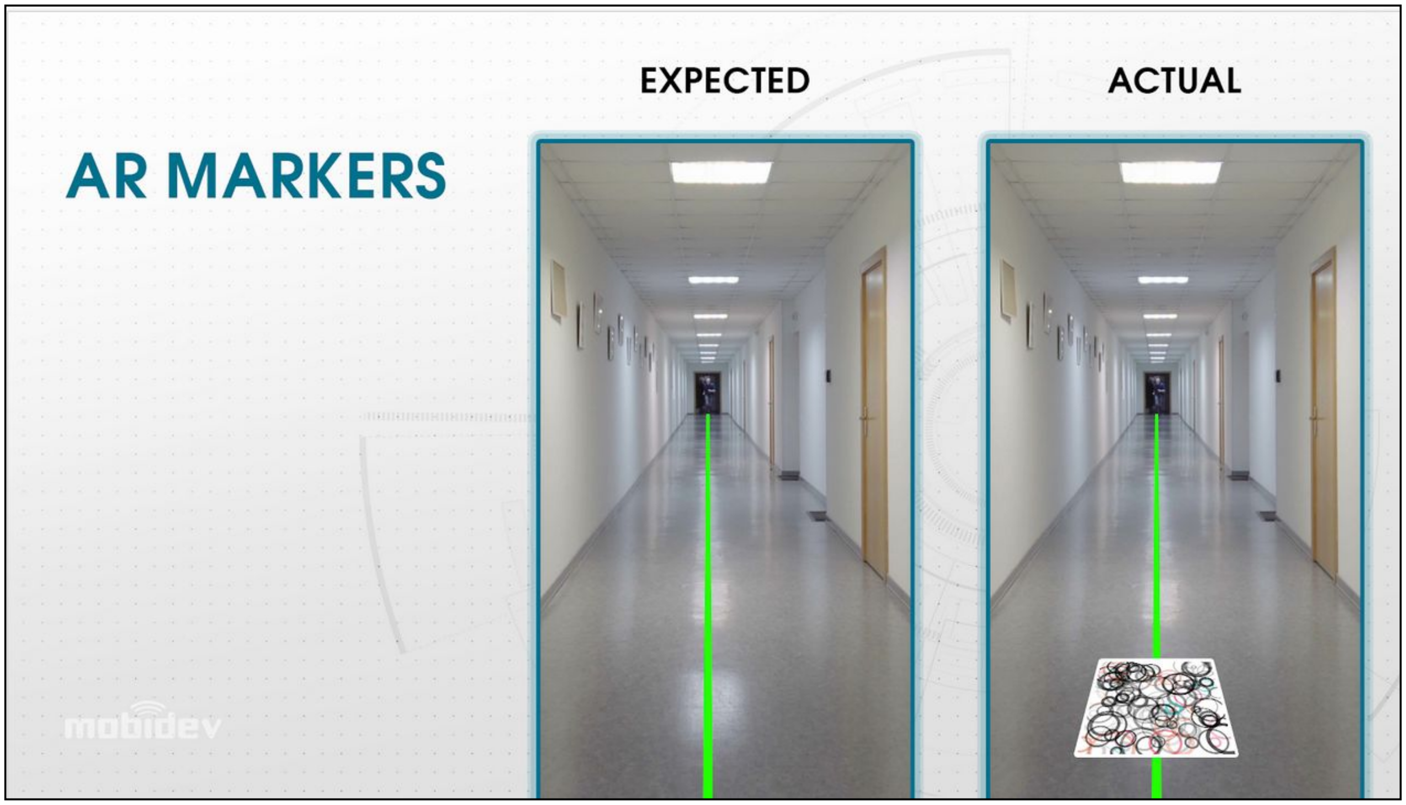
An ARReferenceImage consists of a name, the physical size and an image. As you'll note, an ARReferenceImage contains no data on positioning. However, if each object's name field is populated with a unique identifier, we can link that ID with the positioning of this image from the cloud.



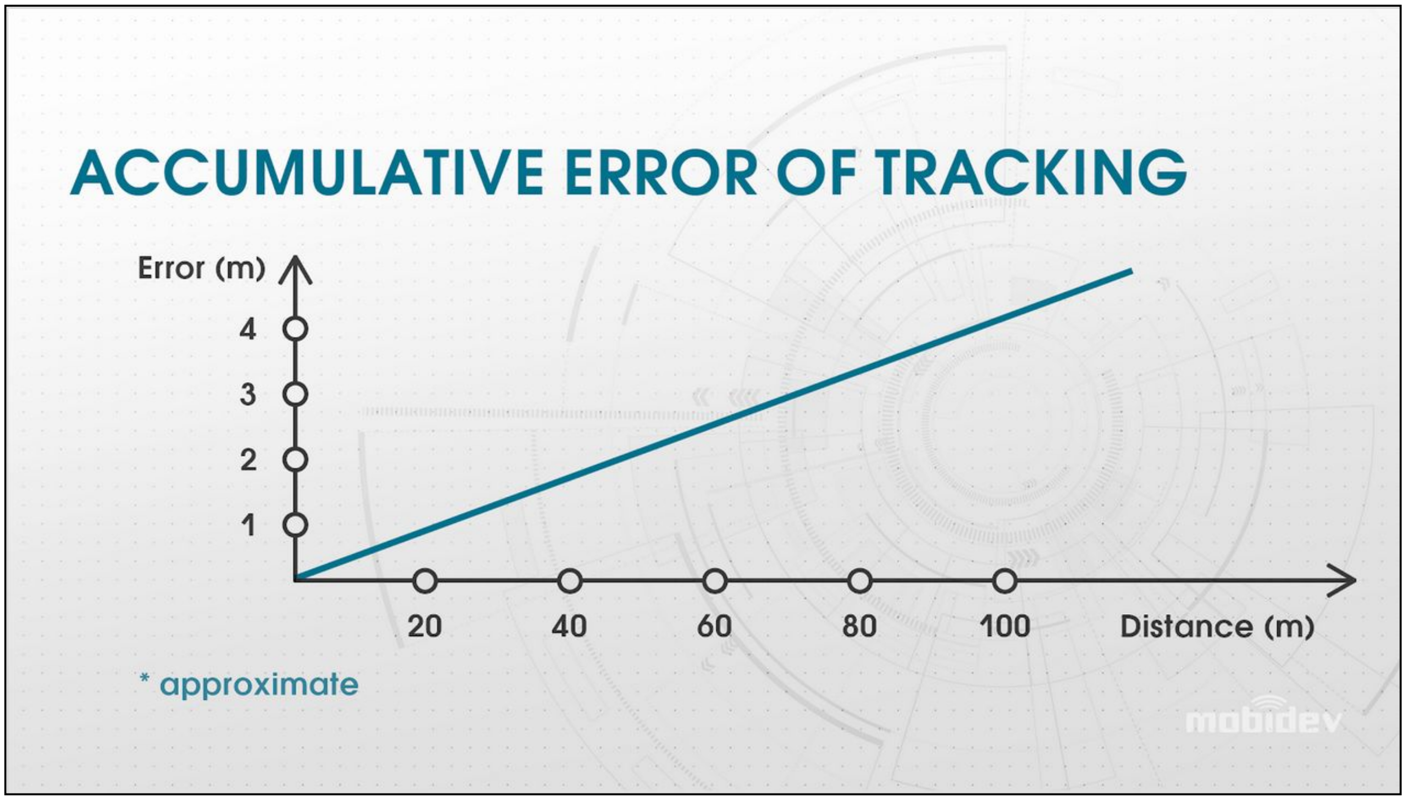
Compass is also relevant here. However, the built-in compass has a bias of around 20 degrees



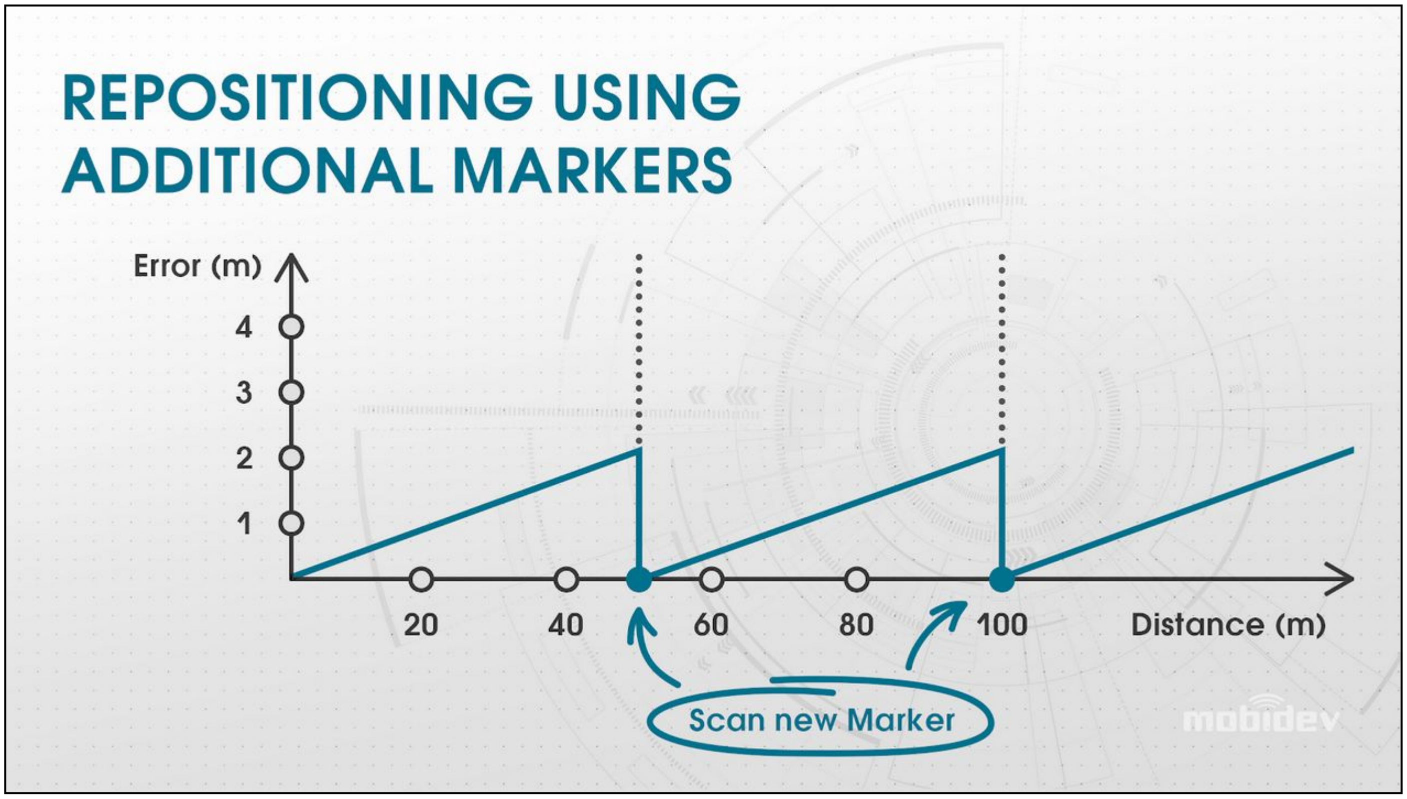
Visual markers were able to help immensely. We know the orientation and location of each of the markers within a building or other indoor location, so we are able to adjust the compass without needing any further actions or inputs from our users.



A drawback to using visual markers is the ARKit 2.0’s accumulative bias factor. The further we find ourselves from a given starting point, the less accurate our results will be.

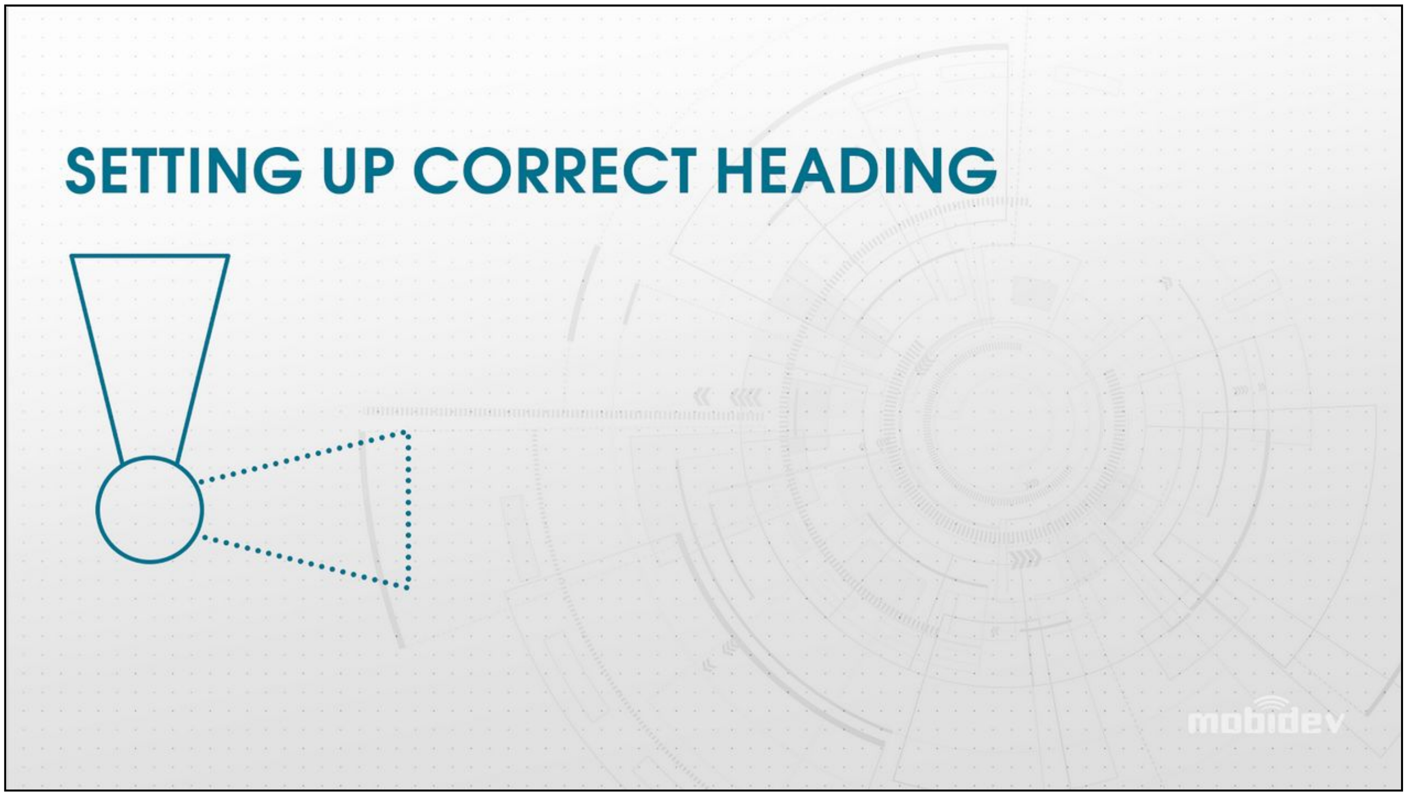


In order to solve this issue, it’s necessary to deploy additional visual markers at intervals of approximately 50 meters or so. New versions of ARKit are leading to improvements in tracking quality with each iteration as well.

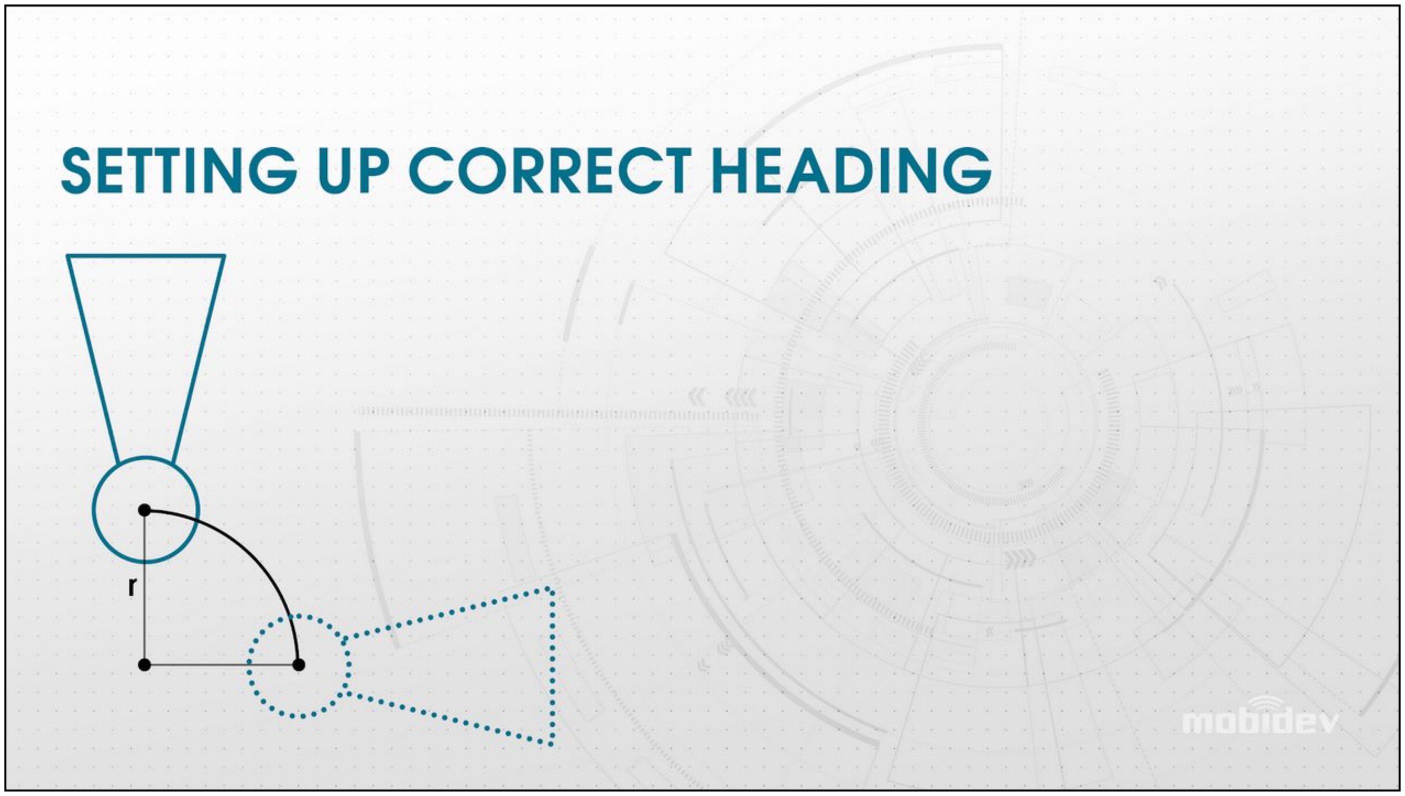


Interestingly, the initial accuracy right after we scanned the first visual marker was so high that it presented a problem.

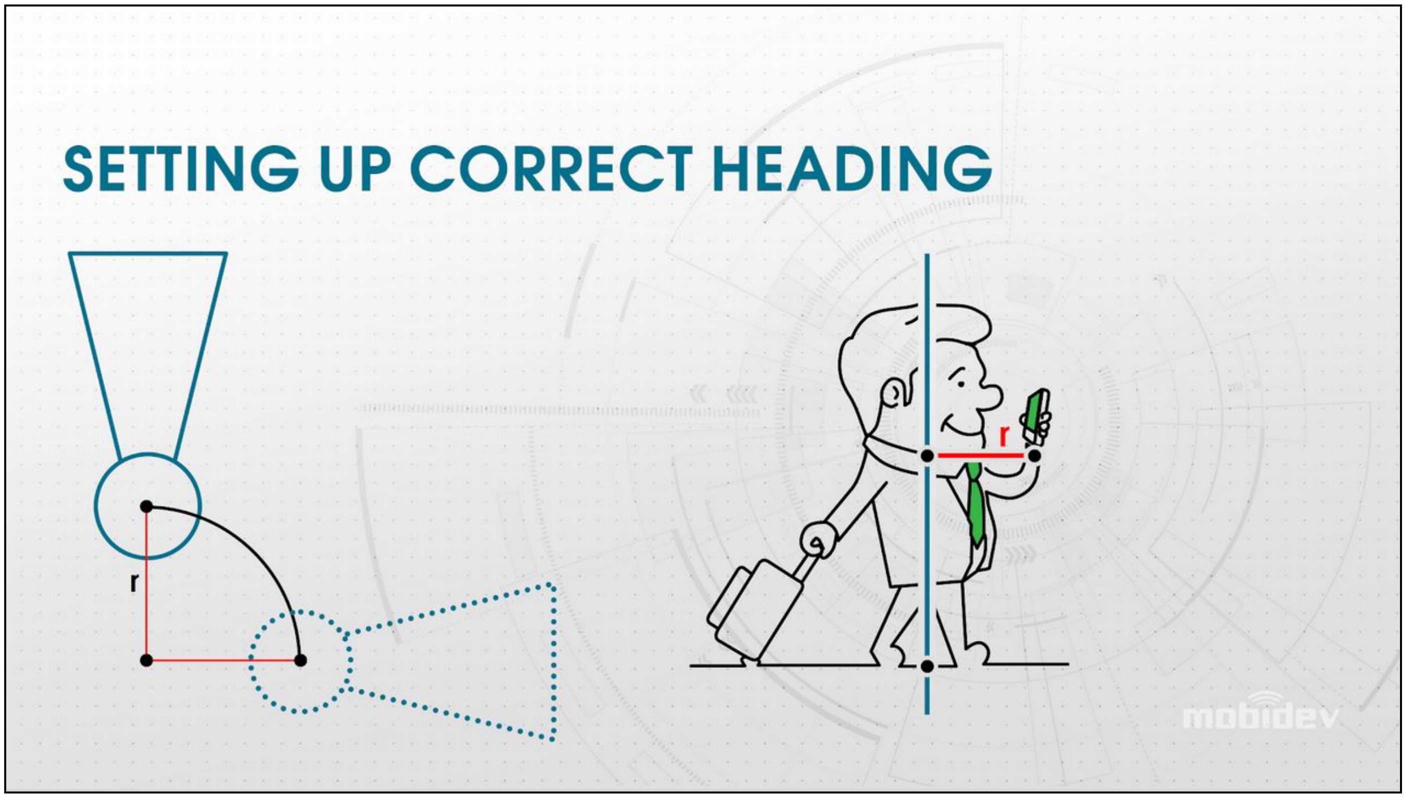
When a person rotates, everything we calculate and render uses the smartphone as the ultimate basis.


SETTING UP CORRECT HEADING

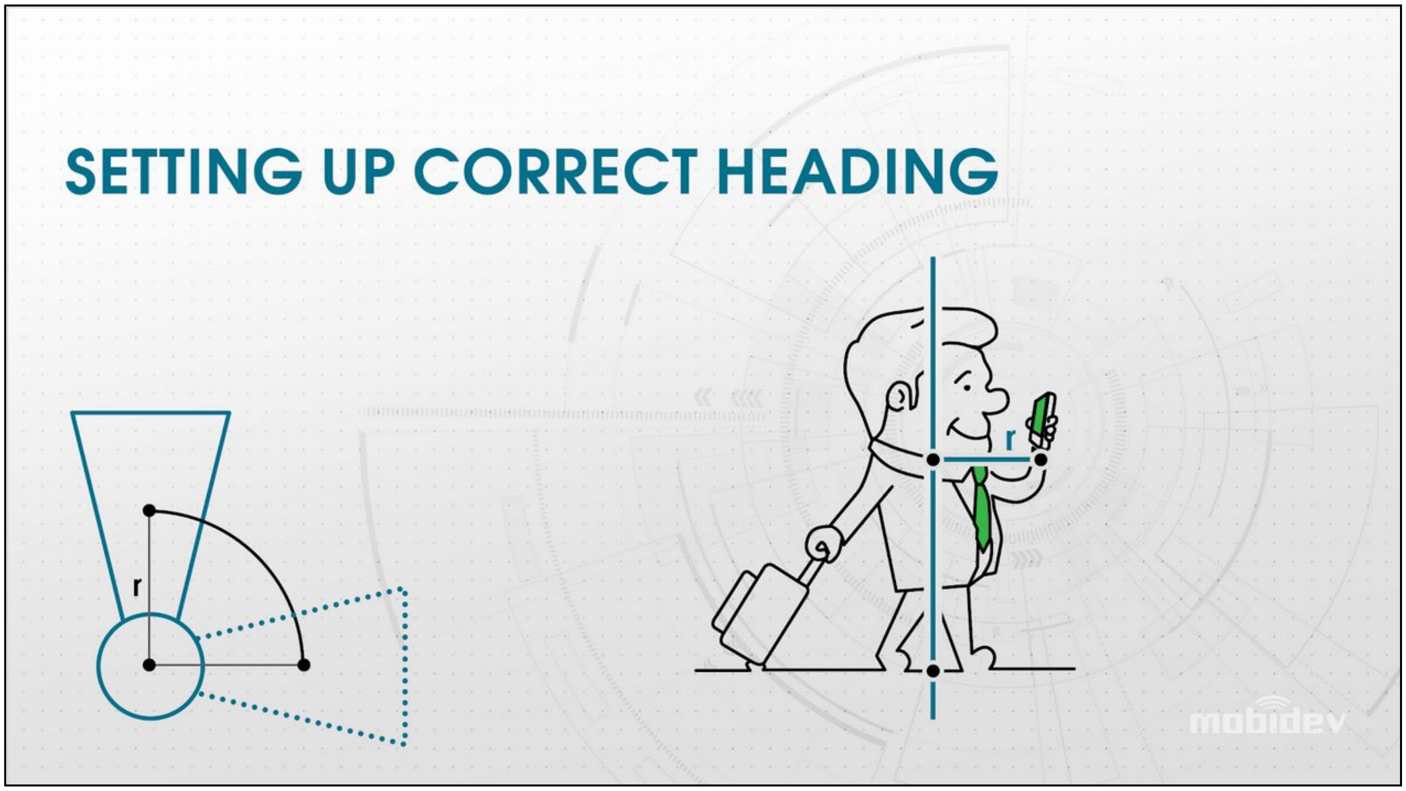
If we do this, a person rotating 90 degrees will find that the 'location' has shifted forward in doing so, despite them not having stepped forward.


SETTING UP CORRECT HEADING

The reason this is the case is that the smartphone is extended from the center of the user's body. So, even though the user hasn't moved forward, the smartphone has. The rendering in this case is 100% mathematically accurate. However, it presents an odd effect for the user.



We're able to smooth this out by shifting the coordinates a little back to the body's axis.

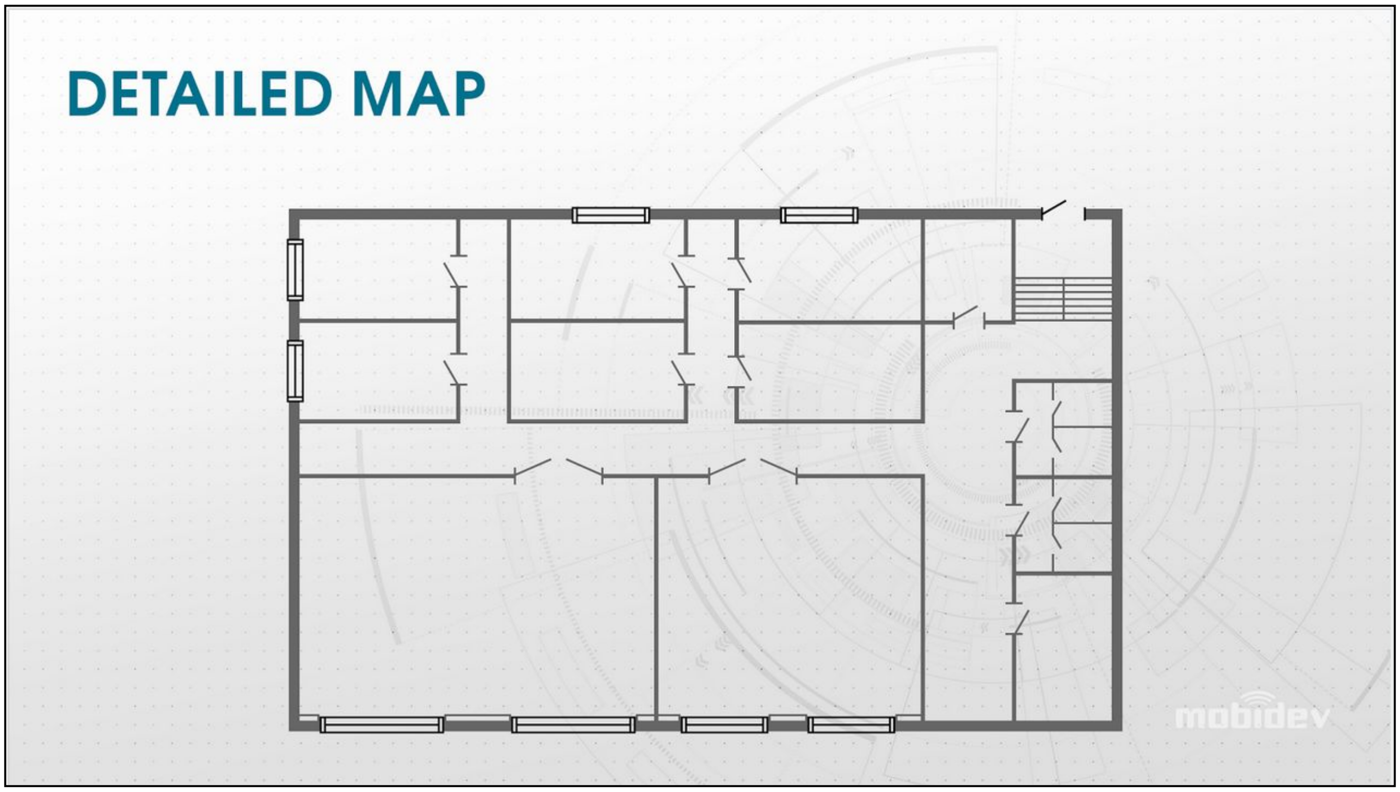


Mapping

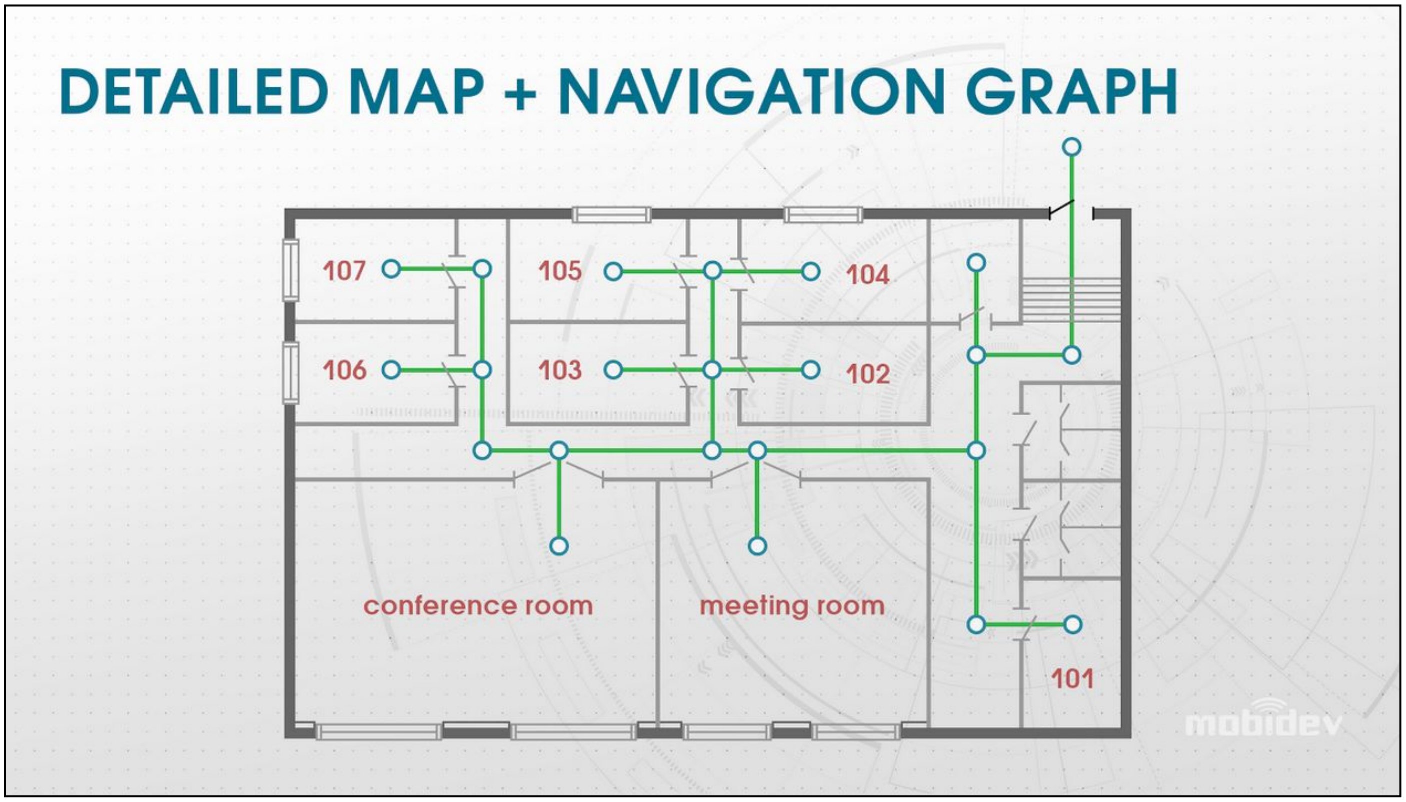
We can't navigate without the use of a map. In some cases, we're able to use a third-party service like Google Maps or similar services. In those cases, we can quickly and conveniently get the map we need, but the maps aren't always existent or available for each building we might require. It's also sometimes not possible to get the flexibility we require. Custom maps are a choice that ameliorates some of these issues.

We can create a map using Cartesian coordinates, and then this map can be aligned with the azimuth and geographic coordinates. The advantages of being able to use a system of Cartesian coordinates are significant. Now, we can work with existing building maps or create new ones with accuracy measuring to the millimeter. Simply using geographic coordinates doesn't give us that level of precision.

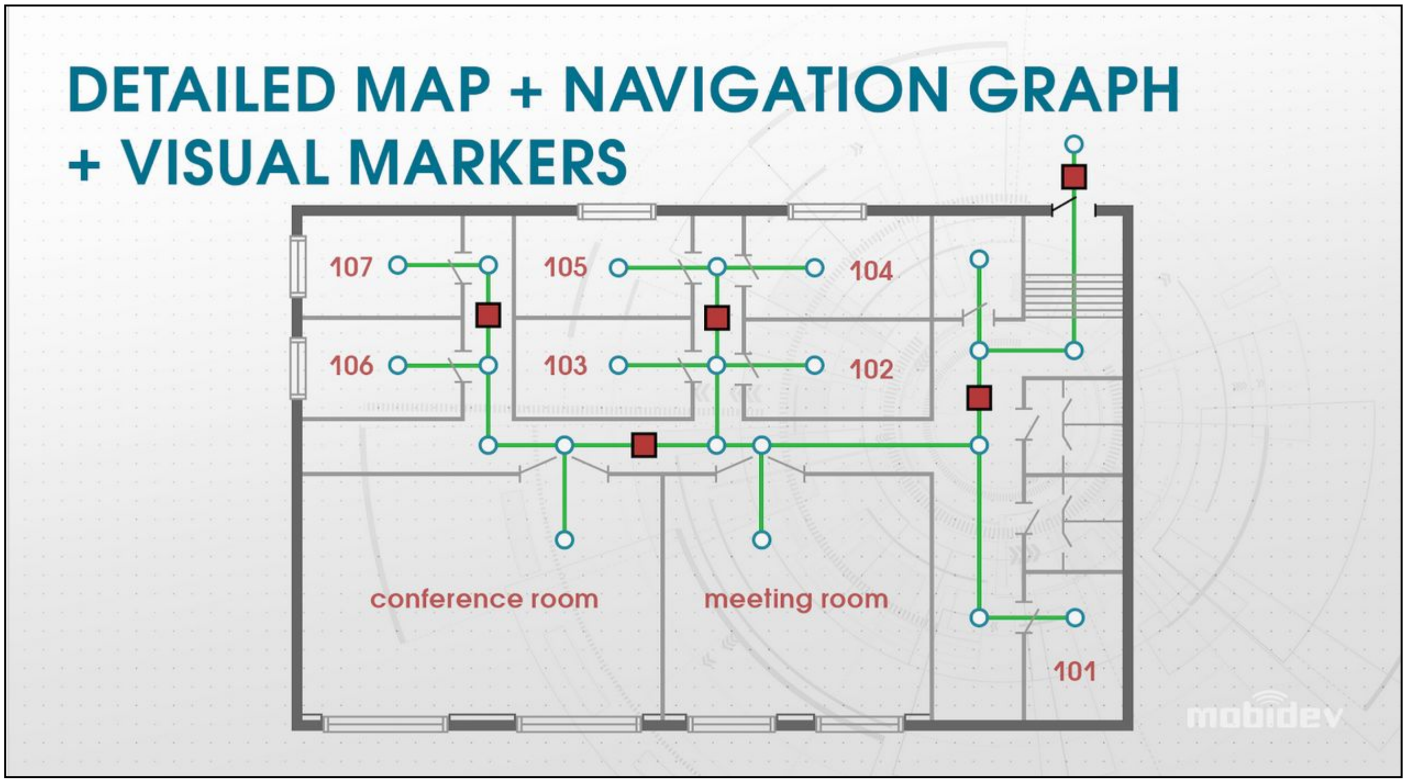
This Cartesian system is also easy for users to perceive given the 1:1 scale it maps back to.
And it’s critical to link Cartesian coordinates to geographic ones. Geographic coordinates can be obtained and aligned using satellite imaging from Google Maps. Here, it’s not necessary to determine coordinates with special accuracy, meaning we don’t need to rely on specialty equipment. This process allows for greater scalability and lets us adjust our compass based on the plan’s coordinates within space.
We avoid using bitmap images, instead storing each map as a vector image. This facilitates high performance, with the ability to zoom in while requiring a minimum of transmitted data. When designing our maps, we avoid the AR Ruler function, as its bias renders it less accurate. Instead, we use more traditional measurement tools to maximize the accuracy of all our maps.



Now, we’re left with a map containing geographical coordinates with north facing up and precise physical dimensions. The next step after this is to begin to implement metadata which allows us to navigate. These are the hallways, corridors and rooms of our interior space. This can be thought of as a graph.



We’re now able to implement visual markers in their appropriate places, whether it be in a hallway, at the point where an elevator or escalator is located, at an entrance, or anywhere else.



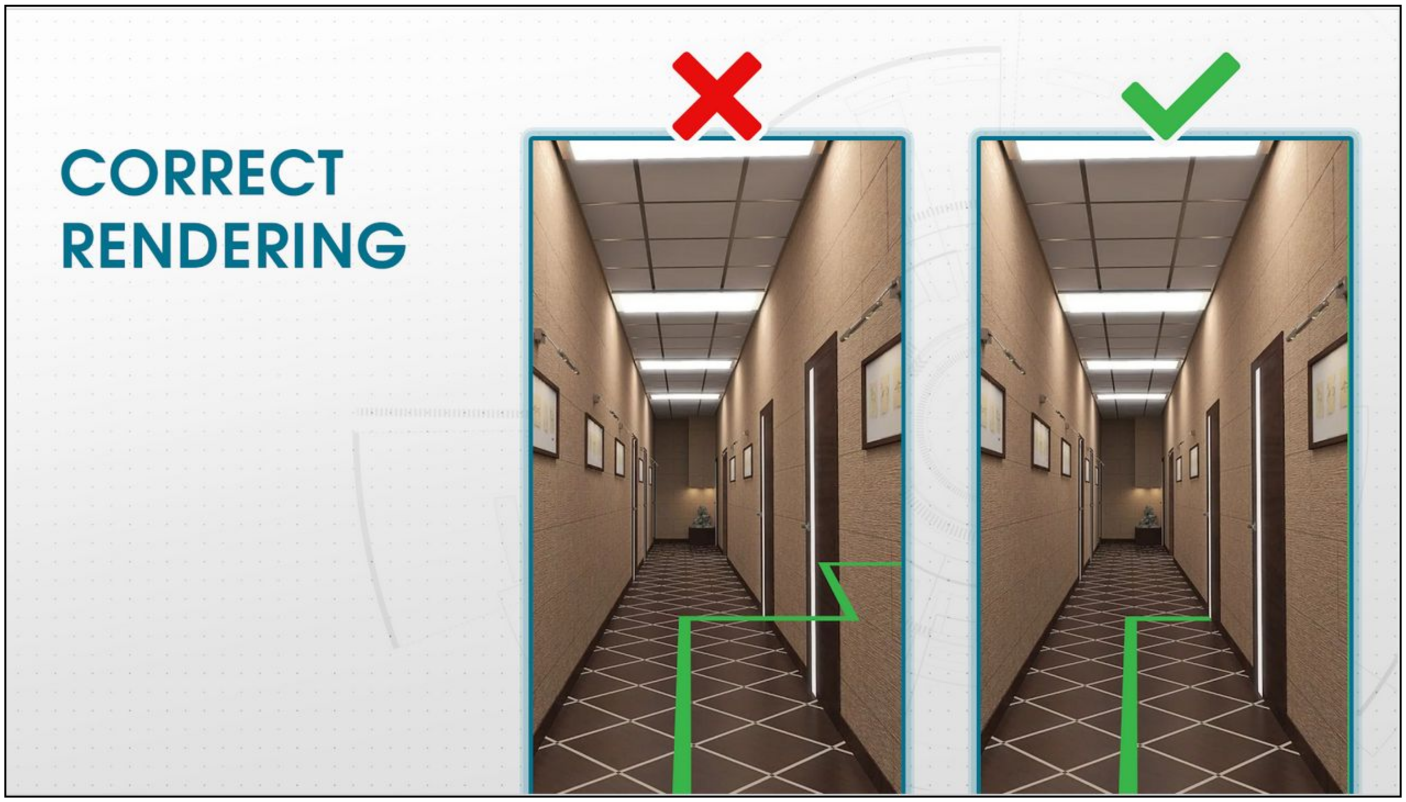
At this point, our map has almost all the data we require in order to navigate in an AR environment. However, we're missing one big piece. Our rendered map needs to match up with the real world out there. In order to accomplish this, it's necessary that our map contains visual markers that perfectly match the locations in the real world as specified by the plan.

An implementation of this process can be referred to as Fine Tuning and includes the following steps:

1. Placing a marker onto the plan.
2. Placing the marker within the real world, with as much accuracy as we're able to get.
3. Scanning the marker and running the Fine Tuning process: turning and shifting the map until it reaches precise agreement with the real world.
4. Saving the marker's updated position into the cloud.

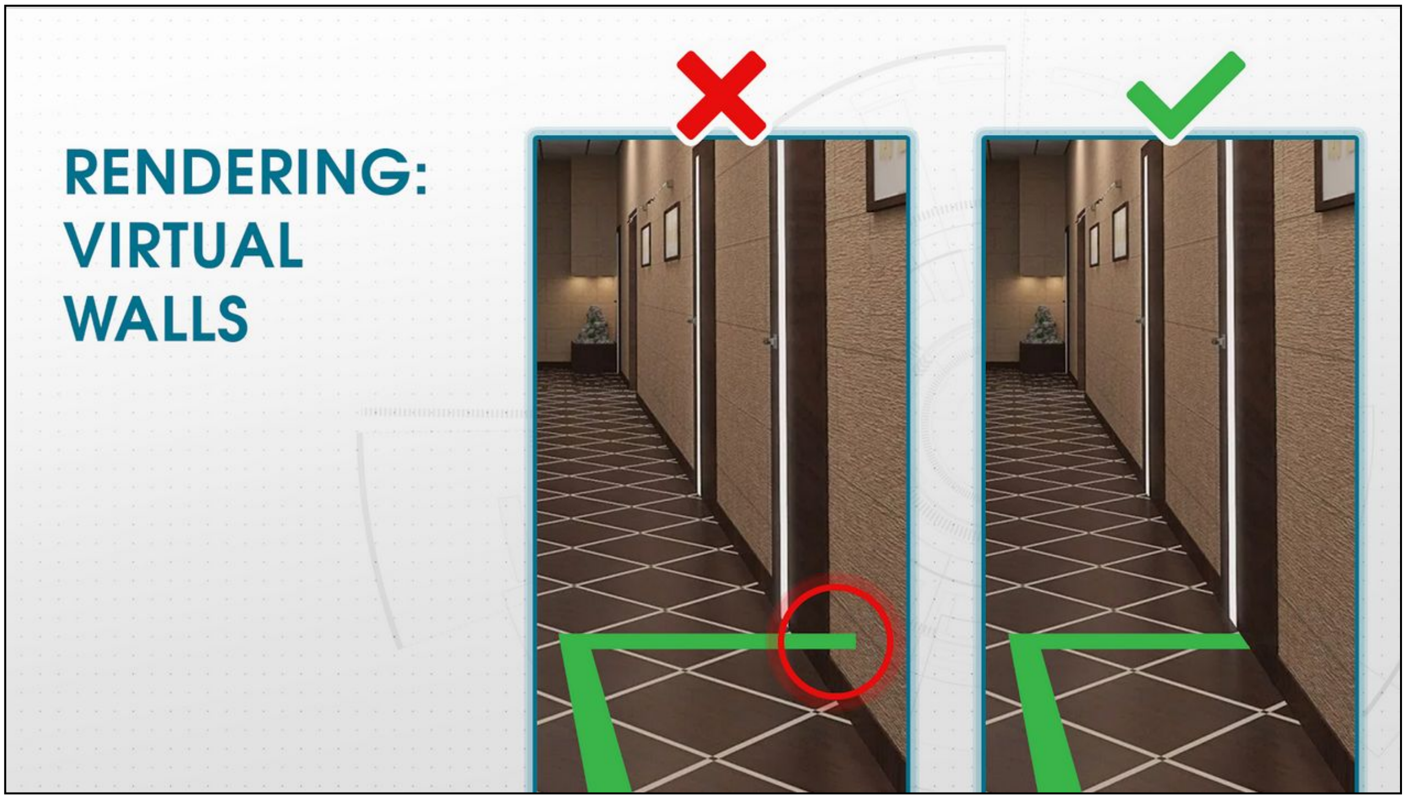
Rendering: Virtual Walls

Now, we're able to build our route through the use of graph theory. In doing so, we'll create arrows or polyline based on our coordinates. However, there's an issue we'll often times come up against in doing so. AR images and content are always rendered on the image we get from the camera, which allows us to create the illusion of digital content existing in the real world. But if we render the full route, it will appear to be drawn onto the walls, which creates an odd visual effect.

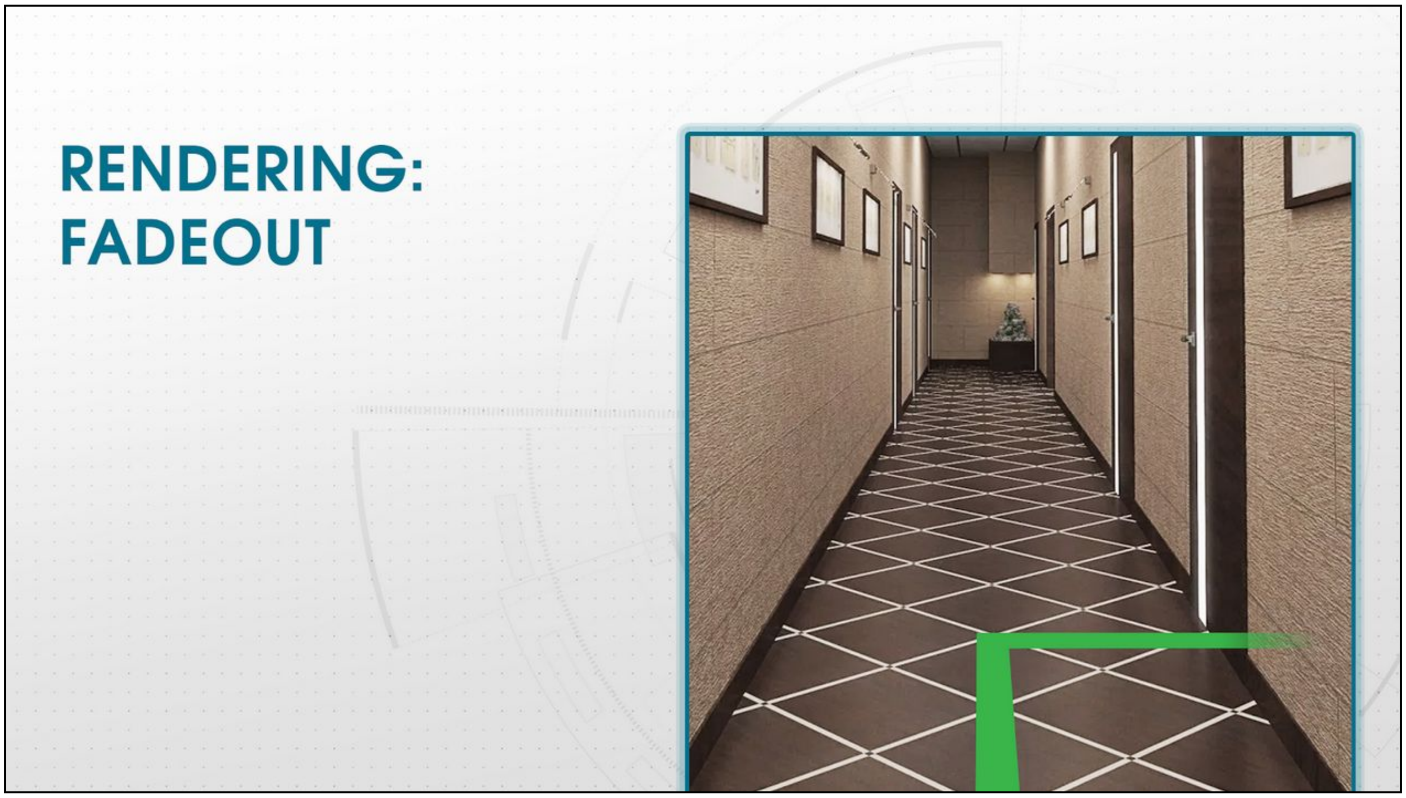


In order to make a given route look less artificial and strange, it's necessary to apply occlusion to the visible part of the route. There are a handful of ways by which to accomplish this.

One solution is to use virtual walls. The problem with virtual walls is that inaccuracy will accumulate, causing the resulting visual output to look a bit worse than we'd optimally like it to look.

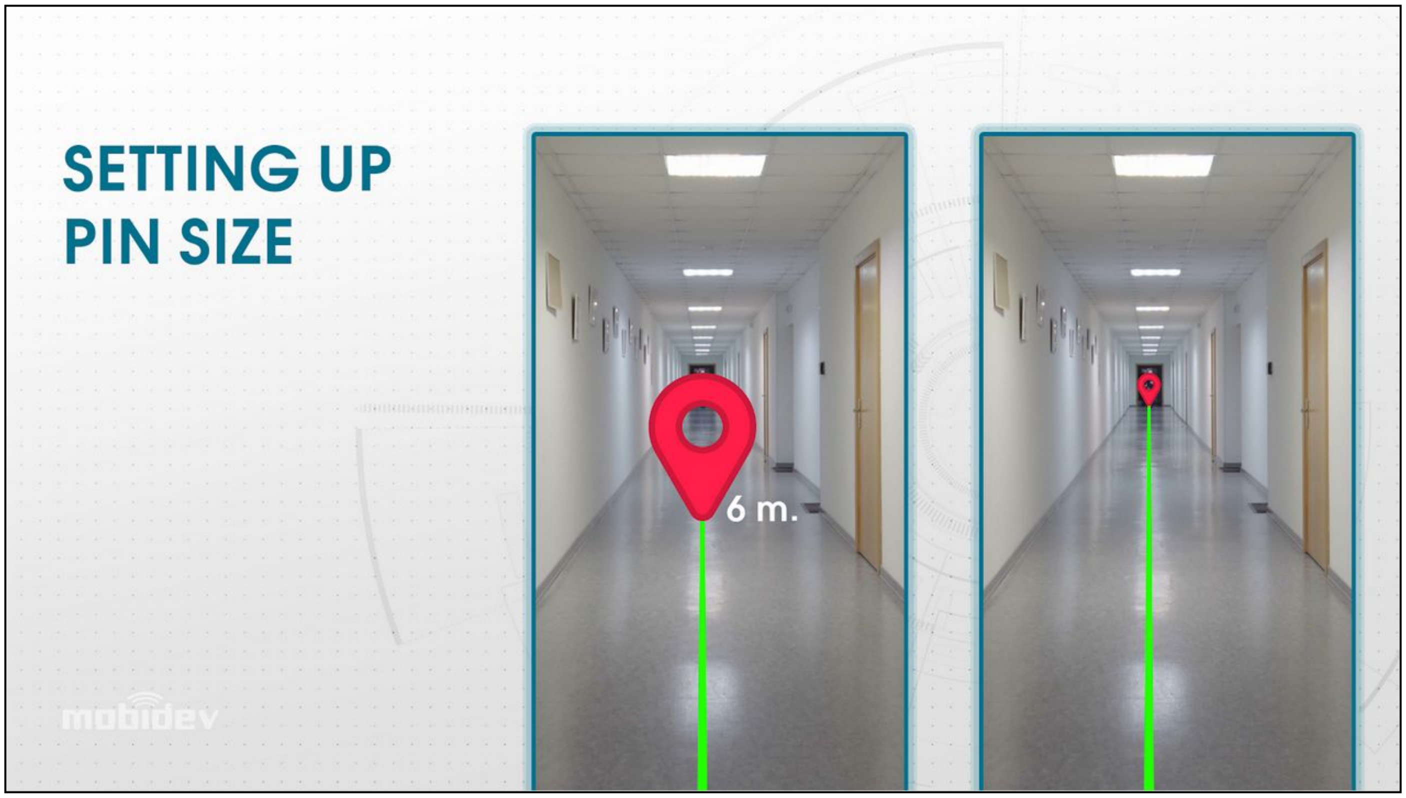


A better option is to create a route with a fadeout effect along the radius. This solution is easier to render, meaning it's less resource intensive, and it produces a similar if not superior results to the other options.

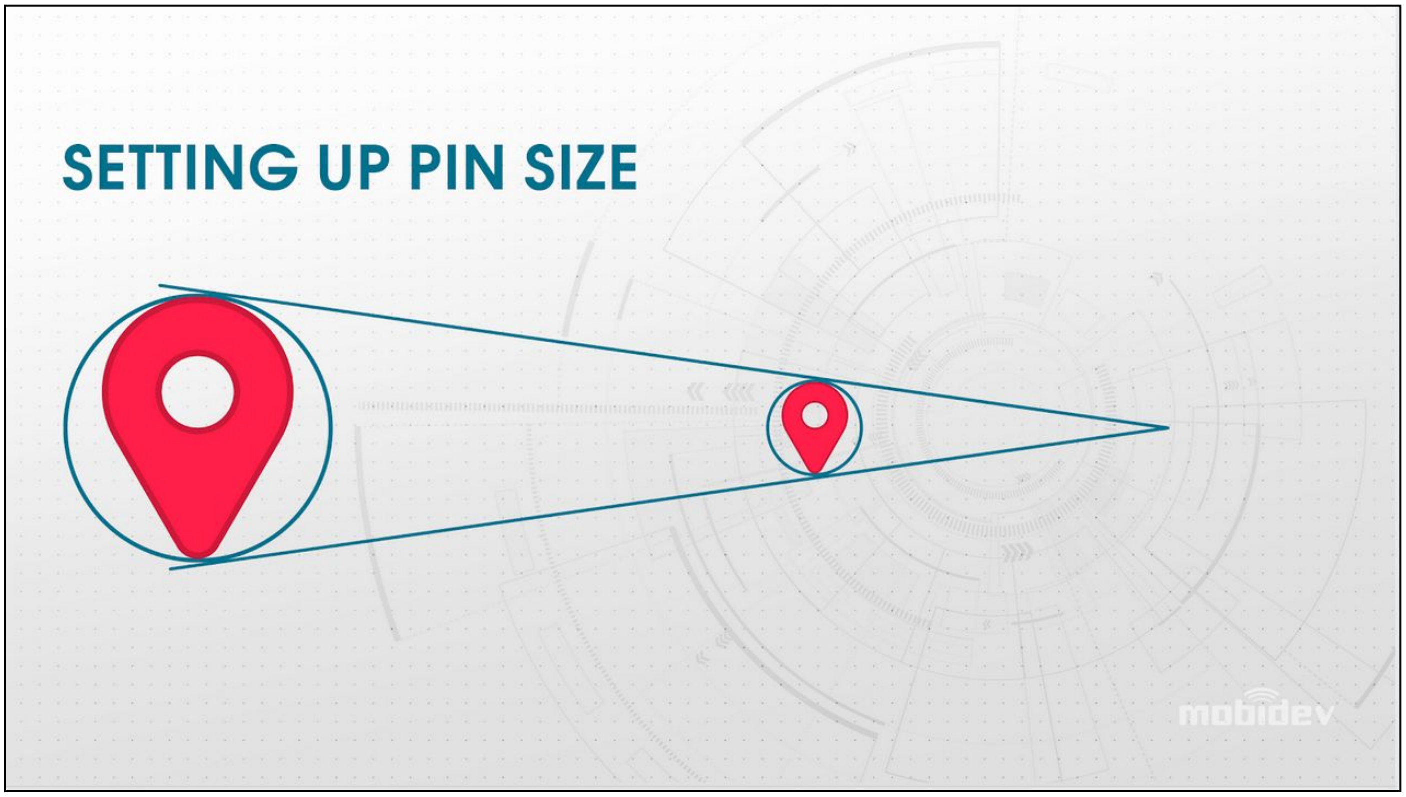


Draw Route: Problem with Pin Size

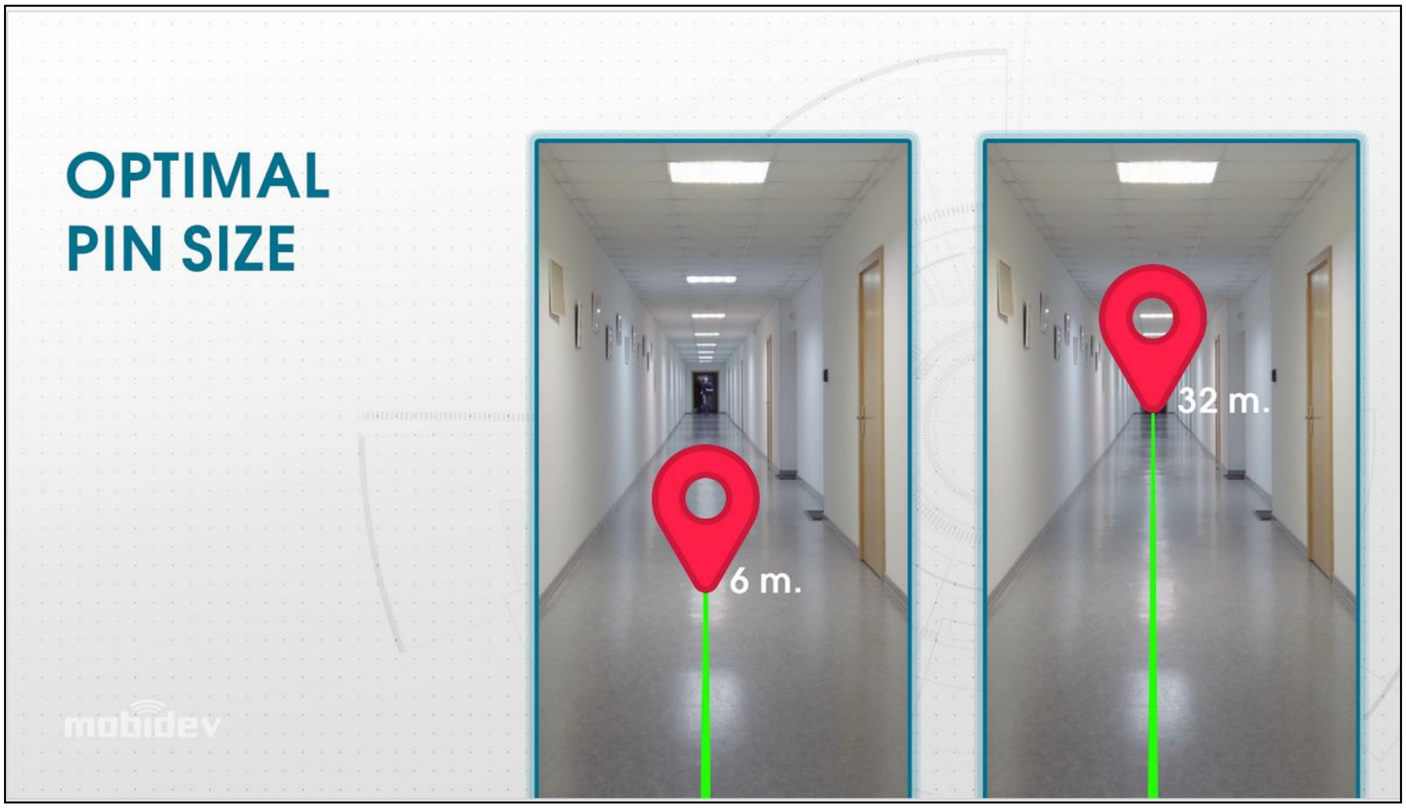
Yet another challenge we face when designing an AR solution is rendering the final pin destination so that it remains constant in size regardless of distance. It’s necessary that it be the same size so that users can easily read labels and handle touches. Normally, perspective means that objects further from the camera appear to be smaller as the distance increases.



But we’re able to overcome the effect of perspective by the angular size.



In other words, depending on the distance in question, we'll render the destination pin as larger or smaller so that it will appear as the constant size to the user.



Visual Marker Requirements

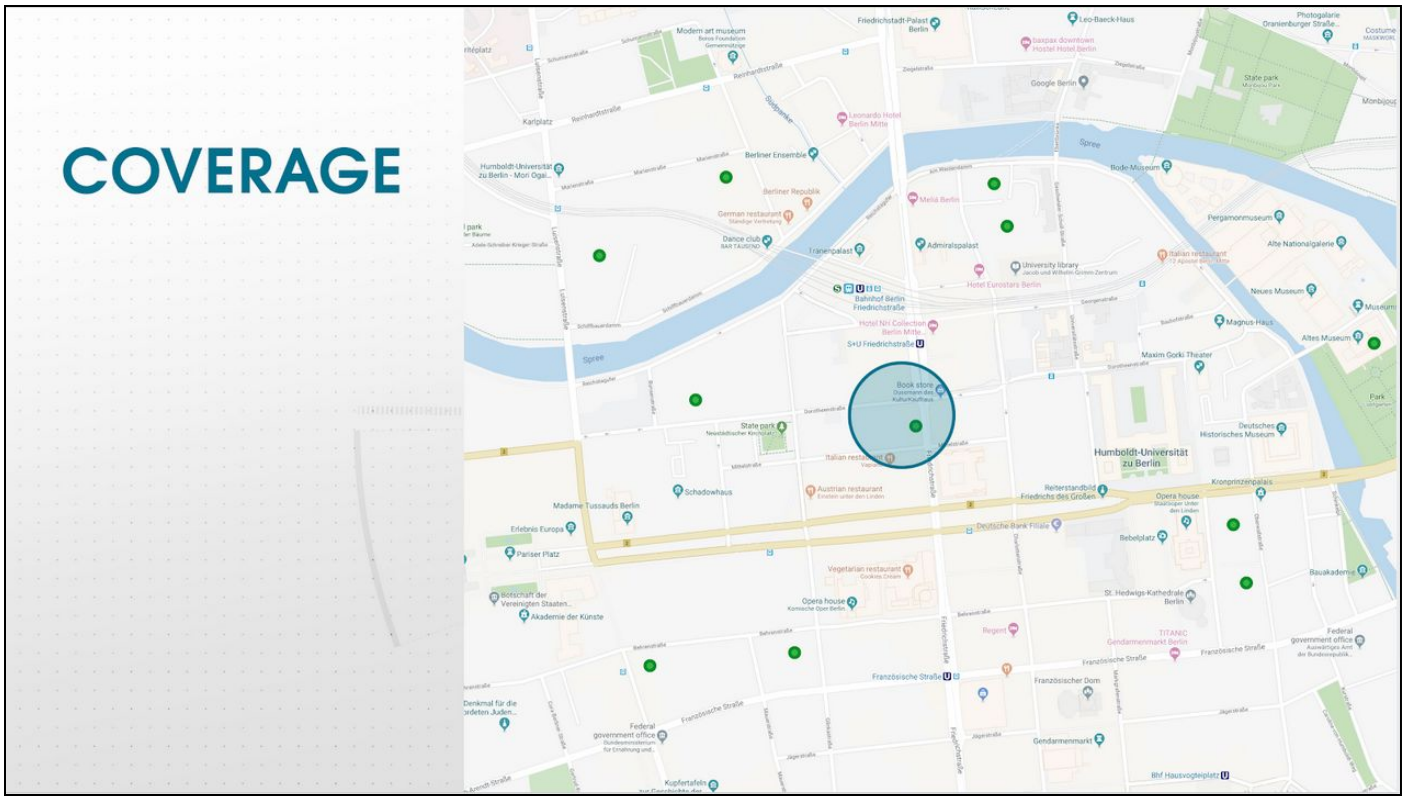
When it comes to visual markers, there are a few standard requirements.

- We want all markers to have a sufficient level of visual complexity.
- We want markers to have distinct patterns from one another, and for them to be asymmetric.
- And we need ARKit 2.0 to be aware of all markers in advance.

With a smaller number of markers, there are no performance issues. But once the number of markers grows into the triple digits or higher, we can start to see major issues with processing power.

This is because it's necessary to parse through all markers to locate our matches. For simple environments, there's no problem, but more complex real-world environments would quickly become unworkable.

Fortunately, there's a fairly easy workaround, as we can use GPS to find our local area and only concern ourselves with the markers in the immediate vicinity at any given time. This means we can create as many maps as needed, storing them in the cloud. But we only need to access a limited number of markers depending on our location.



Future of AR Indoor Navigation

What does the future hold for Augmented Reality development? We can certainly expect that ARKit will continue to grow and evolve. Recently, Apple has revealed iOS 13 and ARKit 3.0.

A new iPhone with a triple camera and depth sensor is awaiting by the community this September. These types of advances will dramatically improve tracking quality.

Another area for improvement will come from increases in mobile device performance. More powerful devices will enable more complicated algorithms, and new positioning techniques like Wi-Fi Round-Trip-Time will provide greater accuracy.

We'll also be able to implement new algorithms and solutions that blend existing and developing technologies to produce less resource-intensive, more accurate results in handling AR location and routing.

The demo video below describes how AR-based indoor navigation works.

mobidev

YOUR SOFTWARE DEVELOPMENT PARTNER

CONTACT US

info@mobidev.biz

MOBIDEV.BIZ