EPISODE 739

[INTRODUCTION]

[00:00:00] JM: Augmented reality glasses will one day let us walk through a world where the digital blends together with the physical. 3D objects will be rendered and superimposed on to our field of vision, creating an environment for people to build applications that we can hardly dream up today. These augmented reality glasses are probably 3 to 5 years away from being ready for consumer use, but developers are already building augmented reality applications for smartphones using Apple ARKit and Android ARCore. These augmented reality toolkits use powerful smartphone processors and computer vision to give developers simple primitives for placing and manipulating 3D objects.

Most of these AR applications are made for a single phone, and AR is useful for a single phone. For example, you could hold up your phone in front of an empty room and see on your phone how it would look if you had an IKEA couch sitting in the middle of that room, but shared augmented reality experiences can be much more exciting. Shared augmented reality can allow us to play a game of virtual basketball both controlling the game that is synchronized between us. Shared AR would let me go to a restaurant and create a virtual billboard in front of the restaurant that only you would see when you walked up to the restaurant and held your phone in front of you the next time you are at that restaurant.

Ubiquity6 is a company with the goal of enabling shared AR experiences. Ankit Kumar is the cofounder and CTO of ubiquity six. He joins the show to explain why building shared AR is a challenging technical problem. It requires building a digital model of the real world and mapping that model to coordinates in space so that users can reliably persist augmented reality objects that each other can see. In this episode with Ankit, we talk about computer vision, digital mapping, the increasing power of phone processors and the potential of shared AR.

We are conducting a listener survey, and if you are a listener to the show, we would love to get your feedback. Go to softwareengineeringdaily.com/survey and tell us what you like and what you don't like about the show. You can also sign up for our newsletter by going to softwareengineeringddaily.com/newsletter, and if you are looking to advertise on Software

Engineering Daily, you can go to Software Engineering Daily and click the advertise link at the bottom. You can reach out to me, jeff@softwareengineeringdaily.com. We are looking for Q1 sponsors. With that, let's get on with this episode.

[SPONSOR MESSAGE]

[00:02:52] JM: This podcast is brought to you by wix.com. Build your website quickly with Wix. Wix code unites design features with advanced code capabilities, so you can build data-driven websites and professional web apps very quickly. You can store and manage unlimited data, you can create hundreds of dynamic pages, you can add repeating layouts, make custom forms, call external APIs and take full control of your sites functionality using Wix Code APIs and your own JavaScript. You don't need HTML or CSS.

With Wix codes, built-in database and IDE, you've got one click deployment that instantly updates all the content on your site and everything is SEO friendly. What about security and hosting and maintenance? Wix has you covered, so you can spend more time focusing on yourself and your clients.

If you're not a developer, it's not a problem. There's plenty that you can do without writing a lot of code, although of course if you are a developer, then you can do much more. You can explore all the resources on the Wix Code's site to learn more about web development wherever you are in your developer career. You can discover video tutorials, articles, code snippets, API references and a lively forum where you can get advanced tips from Wix Code experts.

Check it out for yourself at wicks.com/sed. That's wix.com/sed. You can get 10% off your premium plan while developing a website quickly for the web. To get that 10% off the premium plan and support Software Engineering Daily, go to wix.com/sed and see what you can do with Wix Code today.

[INTERVIEW]

[00:04:51] JM: Ankit Kumar, you are the cofounder and CTO of Ubiquity6. Welcome to Software Engineering Daily .

[00:04:56] AK: Thanks for having me.

[00:04:57] JM: So augmented reality started working for smartphones in the last couple of years, and people have been thinking about augmented reality for a while, but it was not technically feasible. What are the technical changes that have recently made AR a possibility?

[00:05:14] AK: Yeah, that's a good question. So there are really two, and one I would say is more the primary driver. So the primary driver I would say is smartphone processing power, and so you see, for example, that ARKit doesn't work past the iPhone 6 or whatever it is, and that's primarily because there's not enough compute cycles on those earlier devices to support it. So smartphone processing power has been sort of drawing at a very fast rate. Especially nowadays I think we'll see even more sort of acceleration with respect to certain kinds of workloads.

For example, many – like device manufacturers are introducing these sort of specific neural engines that would be called, but ultimately they're really just matrix multiply chips, and we'll see that accelerating even more. So smartphone processing power is probably the critical one I think, and then of course there are also algorithmic improvements that have been coming from a number of disciplines to support something like AR actually kind of both computer vision and more classical, like robotics and what would maybe called SLAM or VIO.

[00:06:23] JM: Give me more detail on the smartphone technology that has improved with respect to AR.

[00:06:29] AK: Yeah. Ultimately it would come down to the ability of smartphones to implement a system like ARKit, which is ultimately would probably be called Visual Inertial Odometry, or maybe it would be called SLAM depending on who you talk to, and this is essentially a computer vision technique that is sort of constantly doing image processing every frame, so like 60 frames a second, to detect sort of key points or track certain features in the space and then integrating that very tightly with the gyroscope, the accelerometer, in some cases like the magnetometer as well to get a sense of where the phone is in 6° of freedom space frame-to-frame. So it's very high frequencies. It's running 60 frames a second, or I think on Android it's 30 frames a second, and it's very compute-intensive, which is why if you run an ARKit app, for example, your phone is going to heat up. Probably eventually it’s going to use a lot of battery, because really the phone is firing on all cylinders at the time. You sort of have the camera stream active. You’re doing a lot of compute. You’re probably rendering things. So the GPU is being invoked. You are also kind of probably listening to GPS and the sensors, like accelerometer and gyroscope. Really, the entire phone is being flexed at that time.

[00:07:52] JM: You mentioned ARKit, and Google also has ARCore. So ARKit is for Apple. As in augmented reality developer, what problems do these augmented reality frameworks solve for you?

[00:08:06] AK: They both solve kind of the critical core piece, which is there are a lot of sort of systems built around them, and especially on iOS, for example, ARKit is pretty deeply integrated with Simkit, which is the 3D authoring toolkit I supposed. But really those sort of integrations are just kind of sugar on top. What the AR SDK fundamentally is providing to you is a post-estimate of the camera of the phone at a very high frame rate, and then you use that post-estimate to sort of place a virtual camera in some scene that you might've built in some other authoring tool, maybe not the ones that have deep integration, like SimKit, and ultimately the kind of key critical piece that it's providing is 30 or 60 FPS estimates of the cameras pose in space. Pose means sort of both positionally, so translation as well as rotation, so orientation in space.

[00:09:04] JM: So the advancements that you're able to take advantage of at Ubiquity6 are not just around the hardware and the local software running on the smartphone. There's also advantages to deep learning algorithms that may be running in the cloud, or maybe they get trained in the cloud and they get deployed to your phone. Deep learning has advanced significantly recently, and this is improved computer vision. What are the computer vision algorithms that are involved in augmented reality?

[00:09:39] AK: Yeah. So just a note a bit about what we do. So we do very little on the device. We typically delegate to the device manufacturers for the on-device AR as you’re sort of describing, and most of our work is done on the cloud. So you're right about that. To answer your question of what is – Like in AR, what is computer vision doing?

So we just kind of need to decouple or like split up a couple things. So it kind of comes down to a question of like what are the features that AR provides, or what are the features involved in AR? So the core critical piece I talked about before, six degree of freedom, high-frequency tracking, VIO, SLAM. I'll talk about sort of where computer vision is used in kind of this technique, and it's actually relatively small, or it's relatively encapsulated I would say in one part of the pipeline. But then you can start thinking about things like forward-facing cameras in Snapchat or a lot of Apple kind of demos and stuff where you're sort of detecting faces and putting facemasks on faces, and that is also AR depending on how you think of it, as well as things like object detection and tracking, image recognition. These things are sort of I think all part of a sort of ecosystem of AR functionality where different apps will use different subsets of them.

I personally consider the core critical piece being the tracking aspect. I'll start with that. So in that tracking part, the high frequency sort of SLAM system where what algorithm is doing ultimately is building some representation of space and using that representation of space to very precisely tell where the camera is frame-to-frame.

So classically this could be called SFM or reconstruction in the computer vision community. In the robotics community it might be called Visual Inertial Odometry, or SLAM. These are actually very similar algorithms. The computer vision's side of it, the structural motion or reconstruction is typically thought of as images only and high compute availability. Whereas in the robotics it would be images, but also sensor data, like gyroscope, accelerometer, etc., and very high-frequency.

In really both of these systems – I mean, there are different approaches, but broadly speaking, the computer vision aspect is really just sort of detecting features and images and tracking them across frames, and everything else is sort of an optimization problem that's trying to do what amounts to sensor fusion and integrate all of the different sensors, like gyroscope, accelerometer, into some consistent formulation of like where is this? How has this phone or this camera, broadly speaking, moved over the last end frames?

So in that kind of regime, where deep learning has helped a lot and where it will help is just in that core piece of finding key kind of feature points in space, and then everything after that is sort of optimization and a lot of it is kind of geometric estimation and numerical methods that ultimately solve this. It's usually formulated as a big nonlinear optimization problem where computer vision was that kind of key core piece that formulated the problem at the beginning. But it's really only just detecting these kind of points in space and matching them. So that's the core part.

The other parts that I mentioned, so like detecting your face, extracting a mesh of your face, detecting objects, tracking those, image recognition, these are much more sort of core full end-to-end computer vision solutions.

[00:13:35] JM: Let's zoom out and talk about what you're building at a high-level, and then we'll come back to the engineering questions, because there's a lot of detail there in the computer vision discussion. So at Ubiquity6 you're building shared AR experiences. Give an example of an experience that you might want to build.

[00:13:54] AK: Yeah, sure. So I can talk about one that we've already done actually. So we did kind of a preview of the app at the SFMoMA a few months ago, and we had two shared experiences there. One was – They're kind of both in the public, in these two public spaces at the SFMoMA. One was ultimately something like an AR art exhibit, where we put a bunch of sort of floating – So let me take a step back. So the exhibit at the time at the MoMA was Magritte. This is Belgian surrealist painter. He has a famous painting of like a pipe and it says, "This is not a pipe in French," as an example.

So he's a virtuous painter. This was the exhibit at the time. So it kind of fit this AR narrative quite well. So one of the experiences was something like an AR art exhibit that was very Magritte-themed, very surreal and sort of you looked around and could kind of interact with other people a little bit, but ultimately you kind of seen the art.

The other was a much more interactive one, which is probably more similar to what we'll see in the app, and we basically put a big sort of sandbox in the MoMA where you could place – Every person could place little blocks and build on top of other people's blocks. You could basically pick a color of a block and place it.

So you're sort of collaboratively building a sculpture in the moment in a sense, and what you build would be persistent and you're kind of doing it with other people around you and it's sort of there in the MoMA. In a persistent manner, it resides there now. That would be an example of kind of a shared persistent AR experience.

[SPONSOR MESSAGE]

[00:15:41] JM: We are running an experiment to find out if Software Engineering Dailydaily listeners are above average engineers. At triplebyte.com/sedaily, you can take a quiz to help us gather data. I took the quiz and it covered a wide range of topics; general programming ability, a little security, a little system design. It was a nice short test to measure how my practical engineering skills have changed since I started this podcast. I will admit that though I've gotten better at talking about software engineering, I have definitely gotten worse at actually writing code and doing software engineering myself.

But if you want to check out that quiz yourself, you can help us gather data and take that quiz at triplebyte.com/sedaily. We have been running this experiment for a few weeks and I'm happy to report that Software Engineering Daily listeners are absolutely crushing it so far. Triplebyte has told me that everyone who has taken the test on average is three times more likely to be in their top bracket of quiz course.

If you're looking for a job, Triplebyte is a great place to start your search. It fast tracks you at hundreds of top tech companies. Triplebyte takes engineers seriously and does not waste their time, which is what I try to do with Software Engineering Daily myself, and I recommend checking out triplebyte.com/sedaily. That's T-R-I-P-L-E-B-Y-T-E.com/sedaily. Triplebyte, byte as in 8 bits.

Thanks to Triplebyte for being a sponsor of Software Engineering Daily. We appreciate it.

[INTERVIEW CONTINUED]

[00:17:39] JM: So this challenge of making a shared AR experience, this involves a lot of synchronization, because you have different people who are holding their smartphones and there's network partition issues, people’s smartphones have different performance, and you want the AR experience to be consistent among the different people who are using their phones in the same space. I know this is a difficult issue, because we did a show a while ago about the networking in like interactive mobile games, and it sounds pretty simple, similar. If you have a immobile fighting game and there's a network partition, it can be really problematic, because if your opponent just like hits you and you had a network partition and you dropped off network, then you have no chance to defend yourself. So tell me about the networking challenges of trying to deliver on this spatial sharing AR experience?

[00:18:42] AK: Yeah, absolutely. So that's a very key point is the networking or the multiplayer aspect of this. One of the reasons as we build our sort of content authoring tool and our content that we really try to build networking and multiplayer really deeply into the framework as a whole, is that it's very important especially in AR to have everything be, in a sense, natively networked to natively multiplayer. One of the reasons for that is ultimately AR – Actually many games are entertainment, but AR in particular is sort of trying to sell an allusion, and the real world is multiplayer, or like the real world is networked so to speak. So you really need that networking piece to be solid to sort of sell the illusion that this is actually some content in this physical space and like it just is in the world. So the networking challenge is important.

So firstly we decouple kind of the more traditional game networking stack, which is to say make sure the virtual content is synchronized across all peers. We decouple that question with the sort of AR networking maybe concept, which is everyone should be in a consistent space prior, or a consistent corner system.

So let's talk about the second one first, because it’s little simpler I think. In our backend, we have a sort of consistent shared coordinate system that refers to the space that some experience resides in. So in this case it was, let's say, the blocks building game in the MoMA. That lives in some corner space, and the first step is all peers need to get localized and tracked by our backend in that corner space so that they all kind of view space in the same way essentially. And so that is one kind of service that we have, which is really kind of the owner of that service is the same owner of the mapping service in the sort of 3D representation, computer vision set of infrastructure. So that is sort of a very kind of encapsulated service that says, “Here's a kind of an identifier for coordinate space. Tell me where I am in it and do that often and keep me synchronized with respect to that space.”

That's a little simpler in the sense that it doesn't really need to be such high-frequency, because really if we only update – So the first localization is very important, and then after that we’ll update with much lower frequency, because we rely on ARKit or the on-device ARCore on Android, we rely on the on-device tracking to kind of fill in the gaps between updates that we get from our persistent system, because ultimately you can sort of back out from – Typically speaking, these AR SDKs will report like absolute poses relative to some origin, but you can always back out how much the phone has moved from some previous update to our kind of consistent corner space and compose that against where we think you were in our shared space and now where you are at this time. So that's a low-frequency system really.

So now you have all peers knowing where they are in this one shared corner space, and now we move on to the problem of synchronizing the virtual content across all peers, and this is a much more traditional sort of game networking challenge where it’s got to be real-time. It’s got to low latency. You have to be able to sort of – You can't wait until the server kind of acknowledges every action before you show it to the client, because things will seem very, very slow and sort of high latency. So that's a much more kind of traditional game networking stack which we build very deeply into our sort of authoring tools in our content.

So I can talk a little bit about like at a high-level how those things work. I actually think it’s quite interesting. The way we do it – Well, there are a number of different approaches to game networking or to sort of this more fundamental problem, I suppose, of synchronizing data across peers. One approach that I think is it's not exactly what we do, but I think it's very – Well, it’s very similar and it’s also very easy to understand, and it's interesting as well, is to think of a game or to think of the kind of propagation of state every frame in a game or some experience, as if you think of it very functionally, where every peer is sort of submitting actions to some stateless pure function reducer that actually implements the propagation of space. It’s sort of reduces the problem of data synchronization to everyone having the same order list of these actions.

So first of all, that makes the problem much simpler now. So everyone starts with the same state, right? This is like a very common approach in something like StarCraft or League of Legends where like you find a match and then everyone starts a game, right? So everyone starts with the same space, and then as long as everyone has – All the peers agree on the ordered list of actions, then everyone will arrive at the exact same kind of current state as long as the reducers are actually peer and are deterministic, right? So that's kind of the high-level approach, and there are a bunch of additions that you need to make on top of that.

One thing is that, typically speaking, it's hard to get determinism in games, especially if you're doing something like physics, where just the floating-point operations itself will lose your determinism, and then that can actually grow without bound relatively quickly. So you need to have some kind of direct state synchronization for those pieces of data that are likely to be operated on in a nondeterministic way, if that makes sense. So every once in a while you send an update from like some concept of a mass or someone who has authority on that piece of data to all the peers to accept the story of the world that their nondeterministic compute has done on the state.

[00:24:28] JM: And that master in your setup is in the cloud or do you have some kind of like mesh networking or Bluetooth setup or is it just a centralized master in the cloud?

[00:24:42] AK: For us right now, it’s typically on the cloud. As you sort of pointed out, there are approaches where it could be kind of like a peer-to-peer connectivity where there’s sort of one master which is one of the peers, and it doesn't really realize that it’s the master. But there's some issues with respect to like cheating there, or like one of the other upsides of having like an authoritative server that's adjudicating these functions essentially is that you have control over like someone can maybe cheat locally so like their local client thinks that they’re like winning, but the source of truth is like rejecting it for whatever reason. So we have a master in the cloud.

[00:25:18] JM: The idea of persistence, you want people to be able to place objects in augmented reality and then have other people walk up to those objects at a later date, perhaps interact with them. Why is persistence a hard problem?

[00:25:33] AK: Persistence comes down – So the problem of persistence is part of the sort of mapping or computer vision side of the world. It's a hard problem for a number of reasons, first, because the environment changes often. For example, Android has the concept of cloud anchors. This is like an ARCore feature where you can sort of take a representation of some part of the space, then ARCore will post in the cloud and someone else can localize against it. So now you're on the same corner system, and those will only last for I think it's 24 hours. Part of the reason that that is a requirement is that the approach that they've taken there is very sort of local in scope. So you kind of make an anchor as it's called on some area of say a plane or a wall or whatever it is, and that might change. You come back 24 hours later, it might look totally different. Now the other peer isn't going to be able to find it really.

So environments change kind of not just in terms of actual items or whatever it is moving around, but also in terms of say like time of day will make the lighting change a lot or the seasons could change or or whatever it is, or it could be dark or it could be light. In order to really solve the problem of persistence, you need to have a representation of space that is, well, first of all, persistent.

So before in ARKit, for example, before AR world maps were a thing, which is a relatively recent addition to the SDK, there was no concept of persistence in the first place. So ARKit would run session-to-session, and once the session is over, then all of the sort of built up understanding of the spaces is essentially lost. Now you can save it in a thing called an AR world map, which is very useful, but it's kind of limited in scope and basically because it's what comes out of this session that is running on the device and there are limitations to what can be done on a device.

So that's kind of a long-winded way of how do we solve persistence, is that we build much kind of higher fidelity and more full representations of some space and we do that by reconciling session data or sensor data from a number of different devices into one consistent representation of the space that is more resilient to these changes and is constantly updated as people come back to the space and view the content, and that kind of lets us get around these issues of things changing or different times of day, or sort of limitations and scope, because fundamentally the system is built to reconcile across multiple different kind of streams of sensory data and that kind of captures a bunch of different conditions naturally.

[00:28:15] JM: So one idea here is that if you have people who are messing around with augmented reality applications, they're walking around, they're essentially waving their camera over the real world, you're picking up location aware visualizations of the world, location aware, positionally aware visions of the world, and you can stitch these together. We have done a few shows with Mapillary that is doing something kind of similar where they have people who are essentially crowdsourcing all of these smartphone images, and because they're positionally aware, Mapillary in the cloud can stitch all these together and they can get basically a vision of the physical world stitched together from these different phones. Then you can do all kinds of interesting things with that. You can do to object segmentation so that you can actually classify the objects that are in the real world.

But let's talk a little bit about the stitching together process. So people are taking these videos with their phone or they're just taking these images with their phone, and then these are getting passed to you in the cloud and then you're getting real-time or you're getting a positionally aware mapping of the physical world? Is that what you're doing?

[00:29:36] AK: Yeah, that's right. The mapping portion is not kind of designed to necessarily be real-time. The localization and tracking portion is. So one way to think of it – So consider the following case. So you have a space, say the MoMA mapped. So maybe we mapped it and maybe you mapped it when you set up the experience, whatever it is. Now you have another kind of consumer coming into the space and tracking themselves in the space or ultimately trying to view this AR experience, this AR content.

So what's happening there is that this user is sending sensor data to our backend, and in real-time our backend is reconciling that sensor data against the three representation of the space that we have at the time. Now, asynchronously, the stream of sensor data that is coming in from that person is being stored elsewhere and later will kind of go into a queue to update the map and keep it sort of up-to-date or more full or in many times it will increase the spatial extent or whatever it is.

So the mapping portion, the updating portion, we would call it – Internally we'd call it the merging portion. That's happening asynchronously in order to sort of – The goal of that is to essentially keep the map or the space up-to-date and big. The tracking portion, which is what you need to do when you actually want to view the content is having real-time, and the key kind of metric there will be latency.

[00:31:07] JM: So tell me about that problem of stitching together images or videos to make a map of the real-world. The last I remember about – Well, just tell me about what's the state-of-the-art in terms of the research or are there well-defined algorithms for doing that or you're kind of at the cutting-edge? How hard of a problem is that today?

[00:31:29] AK: There are well-defined, I would call them approaches or techniques – I mean, algorithms, fair enough. There are well-defined and mature algorithms for what would be called in the computer vision community structure for motion, or reconstruction, those would be used interchangeably. Those typically take as input an unordered set of images. That's like the classical problem statement. Given an unordered set of images, often times the techniques would be motivated by a desire to sort of take community photo collections from say Flickr or whatever and use those. So an unordered set of images build the representation of the 3D space. That's sort of the classical problem statement.

We have a slightly different problem statement, which is given a set of kind of sessions from devices build the 3D representation of space. So the difference there is, well, two things. One is we have video. So we have some temporal consistency that we can rely on, which typically speaking as if [inaudible 00:32:27]. The second is that we have much more sensor data than just images. We have gyroscope. We have accelerometer. We have GPS. We have magnetometer, and we also have ARKit estimates or ARCore estimates, which from the perspective of our backend, are just another probabilistic measurement coming from some sensor. So we have basically more data, and so our approach is something like a merging of SFM techniques and SLAM techniques which typically do have that kind of sensor data to kind of get the best of both worlds in a sense.

So how does it work? I mentioned earlier that there's really a very small part of it, which is computer vision, and really the only computer vision part is detecting and sort of representing feature points in images. So you take an image. You run a feature extractor, which nowadays would often be deep learning. This is like one of the places that deep learning can really help SFM or reconstruction, although the recent – I mean, interestingly enough, it's not a slamdunk.

It's not conclusive that deep learning is actually better for this right now. But I think it probably will be sooner or later.

So you take an image. You extract a bunch of features. You sort of match those features against a bunch of other images or features that you've extracted from other images and now you're kind of in a nonlinear optimization geometric estimation world. There’s really no more computer vision to happen. So now the process is an iterative one where you sort of initialize the system with some of the images. You triangulate landmarks. So a landmark would be considered kind of like a point in space. So the idea is that every feature that you extracted from one of these images, so what is that feature? Its light emanating from some surface in space and kind of hitting the camera at that pic's location.

So if you have two such detections of that surface and there's enough baseline between the images in which they were observed, you can triangulate where that point is in space. So that's kind of one of the key – The kind of like I would say three maybe core sort of building blocks of reconstruction. One is the feature extraction as I described. One as triangulation, and the triangulation is just typical triangulation where you have to kind of varying vectors corresponding to these feature points that you’ve detected and you find where they match in space. So now you have some idea of where that surface was that generated the light that was that feature extraction that you found. Then given that, you can estimate where another image that saw that point – You actually need four or five of them, but you can estimate where another image was in space, because it saw that point in a specific orientation on like the image. Then you kind of iterate this over and over and over kind of re-estimating more images, then re-triangulating more landmarks, then re-estimating more images until you’ve recovered as much as you can.

So it's a pretty amazing thing that these like very – Basically, I just mentioned three sort of building blocks. One is the feature extraction. One is triangulation, and one is this it would be called absolute post-estimation, where given the fact that you see certain like landmarks in space in a certain way on your image, this is where the image must be. That's kind of the problem that that’s trying to solve. You can just kind of iterate this over and over and over and recover very full 3D representation of space.

There is one thing that I didn't mention there, which is what would be called bundle adjustment, which is at various points along this process, you kind of formulate this very big nonlinear optimization problem and you solve it, and that would be called bundle adjustment. That problem is essentially if you hypothesize that this image saw this point at this location on the image – So you have a 3D point and you have a 2D pixel location on the image where your computer vision system essentially has told you that, "I observed that 3D point."

When you actually re-project that 3D point into the image's plane given your current estimate of where the image is, it won't match up perfectly. It will be a little bit off. So there's a cost there. That would be called like 3D projection error. What you're trying to do is you're trying to minimize the sum of all those costs.

So what I just described there would probably be called classical incremental reconstruction. Now, what we have is a bunch of more sensor data; gyroscope, accelerometer, GPS, as I described. So the question is in that story, how can you integrate this new sensor data? The key change there or the key thing there is to formulate – So that big nonlinear optimization that I just described is called bundle adjustment in the computer vision community. You can interpret all of those re-projection errors that I just mentioned as something like probabilistic measurement. So there are Gaussian measurement errors. Now that bundle adjustment problem is really a special case of a more general probabilistic graphical model called a factor graph, where you have a bunch of different measurements from which you derive a bunch of different errors that are basically the Gaussian like measurement error of like if the real thing is at this location and I measured it at this location and my measurement has this covariance, here's how much error that is.

So now you have a principled way to add new kinds of those sorts of measurement errors, and now you have a principal way to add some new sensor data like gyroscope, accelerometer, etc., into this big nonlinear opposition problem and solve it much better. So that's kind of what we do.

[SPONSOR MESSAGE]

[00:38:07] JM: OpenShift is a Kubernetes platform from Red Hat. OpenShift takes the Kubernetes container orchestration system and adds features that let you build software more quickly. OpenShift includes service discovery, CI/CD built-in monitoring and health management, and scalability. With OpenShift, you can avoid being locked into any of the particular large cloud providers. You can move your workloads easily between public and private cloud infrastructure as well as your own on-prim hardware.

OpenShift from Red Hat gives you Kubernetes without the complication. Security, log management, container networking, configuration management, you can focus on your application instead of complex Kubernetes issues.

OpenShift is open source technology built to enable everyone to launch their big ideas. Whether you're an engineer at a large enterprise, or a developer getting your startup off the ground, you can check out OpenShift from Red Hat by going to softwareengineeringdaily.com/redhat. That's softwareengineeringdaily.com/redhat.

I remember the earliest shows I did about Kubernetes and trying to understand its potential and what it was for, and I remember people saying that this is a platform for building platforms. So Kubernetes was not meant to be used from raw Kubernetes to have a platform as a service. It was meant as a lower level infrastructure piece to build platforms as a service on top of, which is why OpenShift came into manifestation.

So you could check it out by going to softwareengineeringdaily.com/redhat and find out about OpenShift.

[INTERVIEW CONTINUED]

[00:40:16] JM: It’s worth reiterating here why is it important to have a map of the physical world that all of your users can agree on. Why is that important? Why is mapping the real world so core to building an augmented reality world on top of that real world?

[00:40:35] AK: Yeah. So what you just mentioned at the end there is exactly what why. You want to align the virtual content on top of the real world, and if you want to support that kind of a functionality, you need to have some reference to which you can place virtual content and have that reference be fundamentally linked to the real world.

So if we don't agree on the coordinate system of the real world, or like where a certain say building is in space, then when I place content where I think that building is. Maybe for me it's at 10-10-10 or whatever. If in your corner space, your corner space is sort of translated and rotated, so you think it's at like 1-2-3, then even though I wanted to put it at the building, it's going to show up in some random place for you. The only way we can obviate this problem is if we all agree that the building is at 10-10-10. This other thing is that 5-5-5. So if I put some virtual content at some location, like that is not just some arbitrary coordinate system in some random 3D space. That's relative to where we agreed the origin is, which is some maybe street corner or whatever it is, or an officer, or a house, or whatever it is.

[00:41:47] JM: So given that algorithmic description that you gave, there is work to be done in implementing that, and actually putting this into practice, putting it in the cloud, building some data pipeline around doing all that, ingesting all of the image data and stitching it together using all of those different algorithms you described. Tell me about the data pipeline, from ingest to processing that mapping.

[00:42:20] AK: Yeah. So scaling this algorithm is no easy task, and there are a bunch of reasons for that. One is just quantity of the data that you're working with, which is kind of what you're describing here. There are other problems also. Like the way I described that problem is we'll have N-squared growth of complexity, because you have to match these features across different images. As that number of images grows a lot, like the number of matches you would have to compute in a naïve sense, grows very quickly. So you need to – That's just to say that there are sort of infrastructure scalability challenges that we have to solve, but there are also algorithmic scalability challenges that we have solve.

So you asked about the data pipeline. So we kind of ingest data from phones in a stream. They're stored as kind of frame-wise in S3, which is kind of like the source of truth of the data. Then essentially what goes through the sort of compute pipeline broadly speaking are pointers to those kinds of blobs, and then the important piece with respect to scalability of the algorithm is to kind of, at all times, break up the algorithm into meaningful but smaller chunks and then have some reconciliation stage at the end, or maybe not necessary at the end, but in some sort a treelike structure, which would be a more traditional approach.

So as the data goes through the pipeline, it's sort of clustered and split by way of our current estimates of the positions or like the relative kind of structure of the data. Then those estimates are kind of constantly being updated by the pipeline itself. So there's a sort of iterative approach there.

[00:44:06] JM: If I recall, as if these problems weren't enough. When I was talking a Mapillary, it sounded like if you don't pay attention to the costs or to like how your data growth is going, your costs can get completely out of control, because you just have so much processing you're doing. Do you have – Are there some bounds around like the context in which you're trying to solve this problem? Do you have like a testing ground, or how are you constraining the problem today before you're anywhere close to kind of launching and getting to profitability?

[00:44:46] AK: Yeah. So that's a very true point, is that these workloads are very compute-heavy and most things that run at the scale that we would hope to run at are much – Well, less compute-heavy. They're much kind of more bandwidth-heavy or whatever it is. But these are very, very compute-heavy, and that takes – It costs money to get CPUs.

So that's a very fundamental part of the system that we think about, and a lot of the kind of algorithmic challenges that I was mentioning around making sure, for example, that it doesn't grow as N-squared, which is obviously not going to work, are around sort of utilizing approximations where appropriate to bring down the amount of compute we have to do to arrive at kind of very, very close to the same solution, or in many cases sometimes it can be better depending on some properties of the data.

So we try to limit the scope in an algorithmic sense, which is to say like implement and design algorithms that fundamentally don't grow as kind of severely as the naïve way would do. Then the other thing is that as we're kind of in development nowadays and sort of as we develop these systems, we kind of focus on areas around the office, which is partly just for convenience, but also partly to kind of limit the scope and kind of provide ourselves with a bit of a sandbox to test this stuff out. I mentioned doing this preview at the MoMA. That was a great example, because one thing is that we believe that AR experiences should, generally speaking, serve to the bring out the venue in which they're in. So that's one of the real powers of AR, is that it's in the real world.

So the MoMA being kind of like a very interesting venue in the first place is in my opinion a great place for AR. But the other thing was the MoMA was just around the street from our office at the time. So it was very convenient for us, and we need to do some of those things while we develop to sort of provide ourselves that sandbox to develop these systems.

[00:46:52] JM: Many of the people that you've hired, I think, including yourself, it seems like you were impacted by science fiction about augmented reality. What is it about those science fiction stories that are so compelling?

[00:47:06] AK: Yeah, that's very true. So one of the big inspirations for the company actually when my cofounder and I were thinking about the idea is actually a book called *Rainbows End* by Vernor Vinge, which is a science fiction book, which is about sort of a future where AR is ubiquitous. This is kind of where the name comes from also [inaudible 00:47:26] ubiquitous AR. Everyone is sort of wearing or like has wearables where they kind of see this AR world overlaid on top of the real world. That was a big sort of inspiration.

So why is science fiction such a big thing? Well, I think that a lot of science fiction books, and Vernor Vinge is a great example, there are sort of a lot of the – A lot of it is about world building. So like kind of constructing this science fiction world in which a lot of things are different or there's new technology or whatever it is. I think oftentimes it's not necessarily about exactly how that world manifests itself in the story, but it's about the kind of promise of that world or like what that world could allow in general. I think people want to build that because they have their own ideas of what could we do with such technology.

[00:48:17] JM: As you're pushing the bounds of what is possible today in AR, are there any limitations that you're running up against where you're starting to think, "Man! Maybe we need – Maybe this area of technology still needs to get a little bit better before we can realize our dreams." Are there any bottlenecks you're sensing?

[00:48:38] AK: Well, so there are two answers to that. I think with respect to what we consider kind of the core technology that would support the kinds of content that we want to build now and that we think are engaging enough to sort of support the kinds of attraction that we want and the kinds of say vision that we want to deploy. I think in that regime, the technology is mature enough. I think we're pretty confident in the current approaches that we're taking, that academia is taking, and we feel pretty confident that those things will be delivered in a reliable way, in an inaccurate way, etc.

I think that if you think about the growth of this company as we sort of succeed and grow and try to add new and more and more novel features, what that looks is more and more understanding of space. So there's a bunch of features or content you can imagine that you could create if you had a much deeper understanding of space.

I think in that regime, there are some bottlenecks or let's say like invention that needs to get done to really like solidly deliver on some of those promises or some of those features. I don't think those are necessary by any means to sort of deliver the vision that we want to deliver on kind of time initially the core vision, let's say. But a lot of that is around the integration of – Well, in my opinion, it will be solved by more and more integration of deep learning into the 3D arena, which is not really happened too deeply.

I mean, recently there're some really interesting approaches coming out of places DeepMind for some of these ideas. But I think to really deliver on some of the much more forward-facing like long-term features or functionality that we want to provide, especially around like really deeply understanding spaces. I think there might be some kind of fundamental invention that needs to happen.

[00:50:32] JM: Where does deep learning fit into 3D? What? 3D modeling? 3D imagery? What are you talking about there?

[00:50:39] AK: I'm talking about understanding 3D space. So right now the common uses of deep learning where it really shines and has sort of conclusively come out is like the best approach would be kind of 2D images. So you pass a 2D image or a list of 2D images and video cases, and the algorithm is looking spatially in the image. So the 2D on the image plane and then in some of these video-based approach there would be temporally as well. So it's looking temporarily across images.

There're not that many approaches yet on kind of giving deep learning. So in those cases, the input to the algorithm is either an image or let's say a list of – An ordered list of images. There are not that many techniques out yet, or like approaches that have really been proven out yet where the input is essentially what would be an unordered set of images where you know their positions in space. So like you have six degree of freedom post-understanding of each image in space, and that would allow the algorithm to do things like triangulation or ultimately learn about the geometry, like geometric concepts, and then use those in its sort of understanding of – Or it's your operation to produce some prediction or whatever it is.

So I think that that's actually a very, very hard problem, because what it comes down to is how do you represent the geometric structure of an input to a network which, as I mentioned, there are some approaches that are kind of not very scalable, because they build like these big sort of plain sweet volumes and like the memory usage is just insane, or there're some approaches that are trying to do this in a more principled way that are kind of new and quite promising I would say, but it's not kind of proven yet, if that makes sense.

[00:52:24] JM: Yeah. In terms of hardware, how does what you're working on – Well, I guess in terms of hardware and software. In terms of what you're working on compared to what you believe to be in practice at places like Magic Leap, or the HoloLens. How did the approaches differ?

[00:52:44] AK: Well, I think one of the main ways they differ is in – Well, in the problem statement in many cases. So hardware companies or hardware platforms, generally speaking, are trying to solve on device challenges. So build a map, or build and measure, build a map, or whatever the problem may be on device in real time and use it as like constant feedback for the user as they're doing some experience. This is also what the AR SDKs do for what it's worth. We are more in the sort of asynchronous cloud setting where what we want to do is kind of reconcile a bunch of different device sessions into one consolidated representation of the space.

So it's a pretty fundamentally different problem statement I would say. These devices, the way the algorithms are set up and the way the, let’s say, paradigms are, is that they'll essentially allocate every session, coordinate system that is origin where they opened up the app with the device, and then everything is relative to that coordinate system. For us, it's different. For us, the coordinate system is the world is like GPS. There's some true coordinate system, right? The world is on a real coordinate system, and we're trying to sort of ultimately understand that shared coordinate system. Whereas the device SDKs, typically speaking, are trying to understand a local coordinate system. I mean, there are similarities, of course.

[00:54:12] JM: Right. That's a big difference. Final question; how long until the augmented reality glasses become a mainstream device or until they’re ready to use and what are the bottlenecks to that happening?

[00:54:27] AK: Well, first of all I would love for it to come sooner rather than later, because I’m very excited about that. I think that AR glasses are not quite there yet. I would predict something like 3 to 5 years is a bit of a shot in the dark for me.

[00:54:43] JM: That's what I've heard from other people too.

[00:54:45] AK: Yeah. But to be fair, that's just like a – It’s a pretty safe thing to say.

[00:54:50] JM: That’s true.

[00:54:52] AK: What are the bottlenecks? Well, compute is a big issue, because to support AR on the glasses – As I’ve been mentioning some relatively powerful compute to power these algorithms, and you don't want to put them on the glasses itself, because it’d be bulky and especially if it like heats up then it's on your face when is heating up. So I would say form factor, which is related to compute is a big issue. I also think that the displays are kind of a key technical piece that doesn't seem to be fully solved. I'm sure there are other bottlenecks as well, especially around say battery. I mean, I think the hardware challenge is very significant still. Then on the software side, well, you need an image stream that's coming from the glasses. You got to put a camera up there. I think it's primarily hardware.

[00:55:38] JM: Okay. Well, it's been great talking to you, Ankit, and I'm really excited about what you're building. You are clearly doing something you're passionate about and you have the domain expertise to tackle that problem. So best of luck.

[00:55:50] AK: Thank you. Thanks for having me.

[END OF INTERVIEW]

[00:55:55] JM: HPE OneView is a foundation for building a software-defined data center. HPE OneView integrates compute, storage and networking resources across your data center and leverages a unified API to enable IT to manage infrastructure as code. Deploy infrastructure faster. Simplify lifecycle maintenance for your servers. Give IT the ability to deliver infrastructure to developers as a service, like the public cloud.

Go to softwareengineeringdaily.com/HPE to learn about how HPE OneView can improve your infrastructure operations. HPE OneView has easy integrations with Terraform, Kubernetes, Docker and more than 30 other infrastructure management tools. HPE OneView was recently named as CRN's Enterprise Software Product of the Year. To learn more about how HPE OneView can help you simplify your hybrid operations, go to softwareengineering daily.com/HPE to learn more and support Software Engineering Daily.

Thanks to HPE for being a sponsor of Software Engineering Daily. We appreciate the support.

[END]