EPISODE 649

[INTRODUCTION]

[0:00:00.3] JM: In the smartphone market, there are two dominant operating systems; one is closed source, that's the iPhone, and one is open source, that's Android. The market for self-driving cars could play out the same way, with a company like Tesla becoming the closed source iPhone of cars and a company like comma.ai developing the open-source Android of self-driving cars.

George Hotz is the CEO of comma.ai. Comma makes hardware devices that allow users with normal cars to be augmented with advanced cruise control and lane assist features. This means that you can take your own car, for example a Toyota Prius, and outfit that car to have something similar to the Tesla autopilot. Comma's hardware devices cost under $1,000 to order online.

George joins the show to explain how the Comma hardware and software stack works in detail, from the low-level interface with a cars can bus to the high-level machine learning infrastructure. Users who purchase the comma.ai hardware drive around with a camera facing the front of their windshield. This video is used to orient the state of the car in space. The video from that camera also gets saved and uploaded to Comma's servers.

Comma can use this video, together with labeled events from the user's driving experience to crowdsource their model for self-driving. For example, if a user is driving down a long stretch of highway and they turn on the comma.ai driving assistant, the car will start driving itself and the video capture will begin.

If the car begins to swerve into another lane, the user will take over for the car and the Comma system will disengage. This disengagement event gets labeled as such. Then when that data makes it back to Comma servers, Comma can use that data to update their models, so they can cross the disengagement event when the autopilot stopped working with the point in time in that video. It's a nice labeled data feedback loop, where you have video and the sections of video

where the car system didn't work properly labeled for you. From there, you can build quite a good supervised learning system.

That's really what this whole show is about is about an engineering problem that George sees as solvable, but very difficult. It's a supervised learning problem, there's not really anything that is impossible to solve within it, and George is very good at explaining these complex engineering topics end-to-end. He's also quite entertaining and he's open to discussing the technology, as well as other competitors in the autonomous car space.

This is great, because I have unfortunately not been able to do very many shows about autonomous cars. We've done one with Frank Chen from Andreessen Horowitz and one with Lex Freedman, who has worked as a self-driving car engineer and teaches classes about self-driving cars. Other than those two shows, which were both high-level, because I just didn't ask very low-level questions, we really haven't done a lot of material about this and I would love to do more. If you know of any self-driving car engineers, or you are one, then please send me an e-mail.

Before we get started, I want to mention that we are hiring a creative operations lead. If you are an excellent communicator, please check out our job posting for creative operations at softwareengineeringdaily.com/jobs. This is a great job for someone who just graduated a coding boot camp, or someone who has a background in the arts who's making their way into technology. If you want to be creative and you want to learn more about engineering, check it out at softwareengineeringdaily.com/jobs.

[SPONSOR MESSAGE]

[0:03:57.2] JM: Cloud computing can get expensive. If you're spending too much money on your cloud infrastructure, check out DoIT International. DoIT International helps startups optimize the cost of their workloads across Google Cloud and AWS, so that the startups can spend more time building their new software and less time reducing their cost.

DoIT International helps clients optimize their costs, and if your cloud bill is over $10,000 per month, you can get a free cost optimization assessment by going to doit-intl.com/sedaily. That's

D-O-I-T-I-N-T-L.com/sedaily. This assessment will show you how you can save money on your cloud, and DoIT International is offering it to our listeners for free. They normally charge $5,000 for this assessment, but DoIT International is offering it free to listeners of the show with more than $10,000 in monthly spend.

If you don't know whether or not you're spending $10,000 if your company is that big, there's a good chance you're spending $10,000, so maybe go ask somebody else in the finance department.

DoIT International is a company that's made up of experts in cloud engineering and optimization. They can help you run your infrastructure more efficiently by helping you use commitments, spot instances, right sizing and unique purchasing techniques. This to me sounds extremely domain-specific, so it makes sense to me from that perspective to hire a team of people who can help you figure out how to implement these techniques. DoIT International can help you write more efficient code, they can help you build more efficient infrastructure. They also have their own custom software that they've written, which is a complete cost optimization platform for Google Cloud, and that's available at reoptimize.io is a free service, if you want to check out what DoIT International is capable of building.

DoIT International are experts in cloud cost optimization. If you're spending more than $10,000, you can get a free assessment by going to doit-intl.com/sedaily and see how much money you can save on your cloud deployment.

[INTERVIEW]

[0:06:20.4] JM: George Hotz, you are the founder of comma.ai. Welcome to Software Engineering Daily.

[0:06:24.5] GH: Hi.

[0:06:24.9] JM: I have been watching some of your interviews for a couple years now. You're trying to build an open-source self-driving car, and I like how open you are in discussing self-driving technology, because it's actually been hard to get interviews with people who can talk

about self-driving, or who are willing to talk about self-driving. I think it fits with your thesis, because your thesis is that self-driving cars are going to play out like the smartphone market.

The argument is that you can be the Android version of a car, whereas some of these other companies or perhaps namely Tesla, will be the iPhone of the self-driving cars. There are some ways in which cars are different than phones. We may have fleets of cars, instead of people owning their own, but there are other ways in which cars are obviously very similar to phones and cars are becoming more and more like smartphones. What are the strongest arguments for and against the smart – the car market playing out like the smartphone market?

[0:07:21.9] GH: Do we not have fleets of smartphones? We have the AT&T fleet, we have the Verizon fleet, different things operate on them. It seems pretty much the same to me.

[0:07:29.2] JM: Indeed. Do you have any theses about why it might be different?

[0:07:33.5] GH: I don't really speculate much on the far-off future. I only say this because this is how the only industry that I can really make a comparison to has played out. I don't think much, like five years in advance. I think it's usually a waste of time.

[0:07:45.9] JM: Got it. What's the open source community like today for Comma? When you interact with the people, what are their beliefs about the open source nature of self-driving?

[0:07:56.0] GH: Well again, I mean, it's not about who cares really what happens in the long term. First off, coming back to why not many people will talk about this stuff, and the reason not many people will talk about this stuff is because they don't have anything, right? A lot of these companies are embarrassed by how little what they have actually works. It's extremely difficult to build these systems to make them actually work. As you've seen, even Tesla with high-class software talent, right? Tesla hires good software engineers, whose GM hiring. They're having trouble. They're having trouble replicating Mobileye.

[0:08:25.3] JM: How do you find out about that? Are there back alley conversations? Is there some secret forum where you gain intel about what's going on at GM, for example?

[0:08:34.4] GH: Well, you can watch the cruise cars drive. In general, I mean, I've seen this paradigm over and over and over again. With a few notable exceptions like Apple, the more secretive people are, the less they have. There is no secret stealth startup. I can't think of one in history. Can you think of a start-up in history which has been super-secret and stealth mode and then they launch something and it did well exceeded expectations?

[0:08:56.8] JM: I cannot.

[0:08:57.7] GH: Neither can I. Remember Apple is a company with a long, long history. Apple started out very similar with us.

[0:09:02.7] JM: What do you think about Apple's – their self-driving – I mean, do you have any intel on that, or could they potentially be in a good place?

[0:09:10.7] GH: I mean, the only intel I really have is you can read that, read the lawsuit, I guess that guy who leaked Apple's secret information. No. I mean, they have TPUs. It was interesting that you know that from the lawsuit. No. I mean, there's no reason Apple would be particularly good at self-driving cars.

[0:09:24.2] JM: TPUs. Apple has tensor processing? The ones from Google, or some different version?

[0:09:29.3] GH: Apple has a different version, right? Apple's probably shipping the only real machine learning ASIC in quantity today, right? The tensor core in iPhone.

[0:09:36.3] JM: Interesting. What is working and what is not working, like industry-wide? Because it sounds like there's a lot of consistencies across the industry, the way you're depicting it at least? What's working and what's not?

[0:09:47.9] GH: Well, so we have one example of level for self-driving cars, and that's humans. I almost don't even like the term self-driving car, because it implies that the car drives. I think what we're really trying to build is a computerized driver. Then you don't think of yourself as building a car. You think of yourself as building a human, right? I mean, I think everybody gets

this wrong. You're not trying to replace a car, you're trying to replace the human inside the car. I suspect that your thing would look a lot more like a human than a car.

You have a lot of people taking this DARPA urban challenge legacy approach. Everything in self-driving, at least Zoox, Cruise, Waymo, they all even come from the same codebase. It's open source. A lot of people don't know about this. You can actually read the Stanford DARPA urban challenge code, and all three of – maybe that self-driving car companies that can actually drive have based off that code.

Now does this approach work? Well, the approach is this, they use Lidars to very precisely map the world, re-localize themselves in the map. Then once they know exactly where they are, they know where to drive. You can draw the exact center line of the road and then they just follow that line. You do a few clever things, if there's a car in the way you might want to go around it, but for the most part, they're fancy line following robots.

Now this is very much not how humans drive cars. You have maybe us and Tesla in a camp trying to drive cars like humans drive cars. With cameras, with reactive, without precise maps, all that stuff.

[0:11:10.9] JM: What's not working?

[0:11:12.8] GH: Nothing. I mean, everything we're doing is working. It just takes forever to do anything. Things just take a long time. We're finally getting to the point now where a lot of it is infrastructure. A lot of making this stuff work is infrastructure, we're taking in and I think we took in 350,000 miles of data from our eons last month. Now okay, let's process that now back to our model.

[0:11:35.1] JM: I'm very familiar with the software development process of something like accounting software, right? You build your first version of the accounting software, and then you give it to users and the users tell you what works and what doesn't work and you iterate on features, and then you can look at your monitoring and be like, "Okay, this server went down. We need to fix this back-end issue."

I'm less familiar with the iterative process of a product that's mostly based on the quality of the machine learning models. Can you give me a description for the workflow and how you are benchmarking yourself as improving over that build measure learn cycle?

[0:12:12.0] GH: Sure. The main metric we track is called disengagements. A disengagement is when the human had to take control back of the car. Now you can divide this engagements roughly into two categories. There are planned disengagements, which is the human took control to say for example, get off an exit, or make a lane change. Then there are unplanned disengagements, where the car drove the wrong way and the human needs to correct the car, or the car wasn't breaking quickly enough and the human didn't feel safe, so they stepped on the brakes. Those are unplanned disengagements.

We want to drive unplanned disengagements to zero. Every month, we get all the data back from our cars about how they drove, everybody uploads them. We don't get all of it back. We have about 70% back. 30% gets lost in the ether. Of what we do get back, we can measure how many disengagements there were and we try to make that number go down every month.

[0:12:59.2] JM: What observations have you been able to make about what causes people to disengage?

[0:13:03.9] GH: Well, so one of the limitations right now and this is a limitation of our model, we only have about a 40 degree field of view. If a car cuts in very closely, if a car cuts in right in front of you, it doesn't see the car in time and the human has to step on the brake. This is one of these okay, we need –

[0:13:21.7] JM: I'm sorry. I said the person disengages. I should have said the car disengages.

[0:13:25.5] GH: Well, no. It's actually the person who disengages. Sometimes the car will disengage, but almost all the disengagements are user-triggered disengagements.

[0:13:32.9] JM: Oh, sorry. Okay, so disengagement means the human is interceding in the self-driving program.

[0:13:39.3] GH: Yes. It means, the self – I should have defined it. When the self-driving system is engaged, it means that the driving system is controlling the car right now. The human is paying attention at all times, but the software is turning the steering wheel, pressing the gas, pressing the brakes a disengagement is when the human overrides the system, when the human steps on either the gas pedal or the brake pedal, or turns the steering wheel.

Yeah. I mean, this is the main metric that we want to drive to zero, or as close as we can get to zero. It's never going to be perfect and it doesn't have to be. It just needs to be better than humans. Yeah, so we figure out what mistakes the cars are making. Some of them are happening, because of quality issues, like the machine learning. Okay, that was just a difficult scenario. Then we add that to a special test set that we have to make sure to try to get future models to work in that scenario, but there's all sorts of subtlety there. Then some of them are just caused, because okay, I mean, that car cutting really close and it wasn't available in the field of view of what we're currently looking at in the camera. That's a feature that we're going to have to add.

[0:14:35.9] JM: When you look at something like that case, I think there was a Tesla accident recently where the car just completely swerved into a divider and crash and killed the driver. That was something that was pretty concerning, because as far as I know, the car was just following the machine learning model, the auto pilot rules and it just slammed into a wall and killed the driver. How do you discover the disengagement scenarios that are more dangerous, or how do you benchmark the safety level when you have these kinds of, I assume that was some tale scenario that was really hard to identify during all the training. I mean, do you have any color on that experience?

[0:15:20.3] GH: It wasn't. If you're talking about the Sunnyvale crash where it hit the divider, or the compacted divider, so the main problem there was not that the car swerved into the divider. This is a misnomer about what happened. It's that the car slowly drifted out of the lane. The driver wasn't paying attention and then it crashed into the divider. If the driver was paying attention, they would have had plenty of time to correct the system, which is really important to communicate and make sure everybody knows that these are level two systems.

Everybody knows that, but even sometimes when humans are driving, they don't pay attention. It had a lot more to do with the lapse of the attention of the driver than anything about the system. You can't expect these systems to be perfect. Maybe someday you can, but none of the systems today are. Even Waymo is probably the farthest along; still has, except for a few PR stunts, attentive drivers in all their vehicles. The Uber crash was the same thing. The car made a mistake, but the person was watching a television show.

[0:16:12.7] JM: First of all, I agree with you that there will be some open source solution to self-driving. When there are car accidents, or fatal accidents with a open source self-driving system, how do you think the optics and the public response will differ from these closed source systems, where there's a little more opacity?

[0:16:32.7] GH: I don't really think much about stuff like that. I'm an engineer. As far as public relations go, I think my basic assumption is that people aren't stupid and just tell the truth and make it as open as possible. How it plays out, that's on other people. It's not on me. I don't spend a lot of time thinking about that.

[0:16:46.5] JM: Okay, fair enough. Let's get into the engineering then. You have a set of devices that are able to outfit a non-self-driving car, a traditional car with self-driving capabilities. Can you describe the hardware and software that is used to outfit one of these cars as a self-driving car?

[0:17:06.4] GH: Sure. I mean, the term self-driving car is a loaded term. Really, what we're adding is it's driver-assistance features. The two common ones are adaptive cruise control, which is what controls the gas and the brake, and then lane keeping assist, which is what controls the steering. As far as what we actually add to the car, we sell a device. No software on it. We just sell the device with the dashcam software. It's basically a smartphone in a case.

A smart phone has almost everything you need to drive a car, because smartphones look a lot like people. Smartphones have eyes and smart phones have ears and smart phones have different ways to communicate. Yeah, there's software which runs on that phone that uses the camera to watch the road and say, “Okay, this is the trajectory that I need to send the car along. This is the object that's in front of me that I need to make sure I maintain a safe distance from.”

Then we interface with the car using another device we make, called the panda. It's just basically a USB to CAN Bus bridge. Then we send signals over to CAN Bus to the different actuators in the car, so you can talk to the brake pedal and say, “Okay, push the brakes to 30%. Okay, put 2 Newton meters of torque on the steering wheel to the right.” Then our software computes, it runs an optimizer to say, “Okay, if I want the car to be here, this is the actuator commands that I should make in order for the car to end up here in two seconds.”

[0:18:20.3] JM: That CAN Bus that is as I understand the software brain, or I guess, the interface to software components that control the mechanical outcome of those software signals to the car? Can you talk more about the CAN Bus and how much accessibility that interface gives you to control over the car?

[0:18:44.4] GH: Sure. You can think about CAN like Ethernet. It's a little bit different, but the same way Ethernet can connect multiple computers together, CAN connects multiple car ECUs together. Now it's an actual bus. Ethernet’s on a bus. You need a hub, or switch, or something. Basically, the steering ECU is connected to the engine ECU, is connected to the brake ECU, is connected to the ECU that controls the power windows, they're all they're all connected on this bus and they can communicate with each other.

What our software does is basically, they're usually – the cars that we work on already have these features, but they have low quality versions of these features. What you do is you unplug the module in the car; the driver assistance module in the car. You plug our thing in instead. Then instead of it saying how to turn the steering wheel, we say, “Okay, turn the steering wheel this much.”

I mean, it varies immensely from car to car, but a lot of cars have these basically CAN commands that you can send that say things like, torque the steering wheel, or step on the brakes, or set my cruise speed to 55.

[0:19:40.9] JM: When you're interfacing with them, is it a frustrating interface? Do you have to build shim, a shim over the interface to make it more palatable to deal with, or is it – do you look at it and like, this is actually a reasonably well-defined piece of software?

[0:19:52.7] GH: Yeah. It is reasonably well-defined, because it's small. Whenever I found software geos are forced to go to bottleneck. The interesting thing about CAN is – so Ethernet has an MTU, has a packet size of 1,500 bytes, CAN has a packet size of 8 bytes. You really have to cram all the information in there. Now the protocol is different from car to car. We have this open source project called open DBC, which is basically a DBC file. It's a specification that documents what each signal on the canvas means, and we reverse engineered them for a lot of the cars, so you know how to look up the correct signal to send. We put a shim on top of it. We put a, maybe it's a hardware abstraction layer for cars. We call it interface, and then every car can be commanded at least through the same interface.

[0:20:37.5] JM: Is there a lot of variability in the underlying CAN Bus technology? Do you have to write a lot of connectors between your interface and the CAN Bus interfaces?

[0:20:47.9] GH: The CAN Bus itself is standard, in the same way Ethernet standard, at the hardware layer and at the framing layer. Once you get to the protocol layer, it's completely different from manufacturer to manufacturer. We wrote, maybe the shims are like 500 lines of Python for each car. People have now – there's been a few people in our community who've written ports. They've written ports for the Chevy Volt, to the pre-auto pilot Tesla, the Hyundai, 500 lines of Python.

[SPONSOR MESSAGE]

[0:21:21.9] JM: The Casper mattress was designed by an in-house team of engineers that spent thousands of hours developing the mattress. As a software engineer, you know what kind of development and dedication it takes to build a great product.

The result is an exceptional product and when you put in the amount of work and effort that went into the Casper mattress, you get something that you'd use and recommend to your friends. You deserve an exceptional night's rest yourself, so that you can continue building great software.

Casper combines supportive memory foams for a sleep surface that's got just the right sync and just the right bounce. Plus, its breathable design slips cool to help you regulate your temperature through the night. Stay cool people. Stay cool.

Buying Casper mattress is completely risk-free. Casper offers free delivery and free returns with a 100-night home trial. If you don't love it, they'll pick it up and give you a full refund. Like many of the software services that we have covered on Software Engineering Daily, they are great with refunds.

Casper understands the importance of truly sleeping on a mattress before you commit, especially considering that you're going to spend a third of your life on that mattress. Amazon and Google reviews consistently rank Casper as a favorite mattress. Try it out. Get a good night's rest and upvote it yourself today.

As a special offer to Software Engineering Daily listeners, get $50 towards select mattress purchases by visiting casper.com/sedaily and using the code SEDAILY at check out. Terms and conditions do apply. You'll get the select mattress purchases if you go to casper.com/sedaily and enter the code SEDAILy at check out.

Thank you, Casper.

[INTERVIEW CONTINUED]

[0:23:31.4] JM: Tell me more about the people in this community. What kind of people are these? Are they software engineers, or are they hackers that decide to learn how to become software engineers? What characterizes somebody who gets into the open-source car community?

[0:23:48.7] GH: You'd have to ask them. I don't really know. The way I think about community building is I don't care why you're here, as long as you're here, as long as you want to help, that's cool. I'm not going to look into your background or anything. I'm not going to ask you who you are, what your gender is, I don't care. Just show up and hack on cars. I don't know. You'd have to ask them. Slack.comma.ai.

[0:24:07.0] JM: Okay. Maybe I will. This CAN Bus interface, is that basically a solved problem and you've solved the interface between commands that you can issue to the car and that car using those commands and you're pretty much operating at a higher level of abstraction, where you're thinking more about the user interface and the machine learning models?

[0:24:30.1] GH: Yeah. Recently we've gotten, at least in some of the cars, we have a Honda Civic model is now very good. The Honda Civic goes where we tell the Honda Civic to go. Yeah. I mean, that's the controls part of the self-driving car. Then yeah, so our level of abstraction above the car specific actuators is how much gas do you want, how much brake do you want, how much steering do you want, what angle do you want the steering wheel to go to?

Then the level of abstraction on top of that is here's a trajectory in-car space that we want you to follow, right? Here's the line that we want the car to go and here's how fast you should be going at each point in that line. I mean, there's a level of abstraction above that, which is here's the path we want the car to follow. The distinction through a path and a trajectory is the trajectory must start with you. The path might not, right? The path might be 3 feet to your left, so then we use a optimizer to figure out how to bring the car onto the path. From a path, we generate a trajectory. Now the paths are what's outputted by our machine learning model.

[0:25:28.5] JM: Okay, so path is the ideal scenario. Trajectory is in reality we are at this point in space and we need to gravitate closer to the path.

[0:25:38.2] GH: Mm-hmm. You can think of path as like the GPS track. If you drove one GPS track and you want the car to go exactly on the same track again.

[0:25:45.6] JM: Then the trajectory gets translated into actual commands for gas and twitching the steering wheel and so on?

[0:25:54.4] GH: Yeah. After the trajectory, we have the lower-level interface which goes in and says, “Okay, well the trajectory says your steering angle should be 4 degrees right now. It's currently at 3, so apply some rightward torque to the steering wheel to get it to move to 4.”

[0:26:07.9] JM: Okay. Before we get into the path level and the machine learning stuff, can you tell me a bit about translating the path, versus the I guess trans – taking the path and saying we are at this point in space and we need to adjust our trajectory. Tell me about how that translation layer, that diffing algorithm works.

[0:26:26.9] GH: Sure. We use an optimizer. It's linear. It's model predictive control. Basically, you have costs and for example, one cost is jerking the steering wheel. You never want to jerk the steering wheel too much. This is a comfort thing. There's a safety thing way further down the line as well to make sure you never jerk the steering wheel. We say, okay, given that we don't want to move the steering wheel faster than this, or that we want to minimize jerk when we're using the brakes, output the idealized actuators, and then we even started to go a little bit further, which is forget the idealized actuator, what is the actual actuator model of the car, right?

You might tell the steering wheel to put 3 Newton meters of torque to the right, but it might not listen to you, or it might only put two, or it might put 3 Newton meters 100 milliseconds late and you have to model all of that. The more accurate your model is, the more you can drive the car without feedback. Now it's a closed loop controller, so it does eventually use feedback, which is, “Okay, I told you to go to 4, you're only at 3.8. Okay, the error is 0.2, put that into the optimizer next time and output that.”

[0:27:33.2] JM: The trajectory configurator or whatever you call this thing, whatever reforms the trajectory and aligns it with the path, do you have to make car specific things, because it's, if you – I guess, you have called different costs that relate to the Civic versus the Accord.

[0:27:50.6] GH: Yeah. We have a bunch of numbers that vary from car to car. I'm not really the expert on this stuff here, but we have things like the actuator delays, which vary from car to car. For example, the Civic takes about a 150 milliseconds to respond to a steering command, the Prius takes 350. You have to basically decide how early do I want to start steering based on my lag.

[0:28:13.2] JM: How did you discover those benchmarks? Did you just have to do a bunch of testing?

[0:28:17.9] GH: We're at the point now where we can just build models from the data. We have a lot of historical data about what it told the car to do, and then what the car actually did. We're getting all this data back from our users, and the ones you have that, you're like, “Okay, let's figure out what model we want to fit to that.” We are using all classical models for the vehicle dynamics. I mean, you could go full neural network. You could just try to have an RNN predict it. Yeah, we stick with the classical stuff there.

[0:28:43.9] JM: Yeah. I mean, if that's working, I mean, this is something you would probably find out pretty quickly if you tried to do that and like, “Oh, well the car is not responding as expected.”

[0:28:51.7] GH: Well, you can always do better. You can always – your model can always be more accurate, right? In theory if you had a perfect model, you wouldn't need a closed-loop controller at all. You would just be able to just tell the car where to go and they would go there. Now of course, drift and error actually accumulates in the real world, so you close the loop. Now how quickly does error accumulate? Well, if your model is really bad, it might accumulate in a 100 milliseconds. If your model is good, it might take 10 seconds.

[0:29:17.0] JM: Okay. I think we've covered how – Well actually, I want to go a little bit more into this. You've got the car at any given point in a drive has some I guess, miles per hour, it's going and other, like their state of the car, and then its position in space. Then you have the path, which is I guess the ideal – what are the variables in a path? Does a path include the ideal velocity that a car would be driving it once it was on a path in a given place? I guess, how rapid is the cycle of checking between the car’s point in space and tweaking the trajectories, so that it's closer to the path?

[0:29:56.6] GH: Sure. Our outer planning loop runs at 20 Hertz, so that's done 20 times per second. That goes all the way down from the vision, to the new trajectory. Then the low-level controls, which actually turn the trajectory into actuator commands run at 100 Hertz.

[0:30:10.7] JM: Can you tell me more about the variables of a state in a state transition and how a state transition proceeds?

[0:30:16.9] GH: This is all, it's in the open pilot. It's in serial LOGCAP PNP. These are all the messages that go back and forth between the different processes on the thing. The path is you can think about it just in XY coordinates with the car at the origin and facing forward, so this is called car frame. At this point, our car is reasoning in two dimensions, not three. Where the path actually is, you can imagine a big Cartesian grid and here's the path drawn out.

As far as trajectories go, so in the path we don't know how fast we should be going, but we do know where the other cars are. We know where there are obstacles and we know where your lead car is. The optimizer, the same optimizer that turns XY paths into trajectories also figures out, given that there's a car in front of me moving at this speed, how fast should I be going at each point. Then the low level controls turn that into okay, this is how much brake you need to apply to be going that fast at this point. Of course, you put costs on deceleration, acceleration and definitely cause some jerk.

[0:31:13.3] JM: Okay. Much like the, I guess the lower level CAN Bus interface layer, it sounds like this adjustment between the trajectory and the path is maybe not solved, could always get better, but you're in pretty good shape and you don't spend much time thinking about this?

[0:31:29.9] GH: Oh, yeah. Once you know the path and where the object is that you want to stop in time for, that's a complete state. I mean, it's not going to be a state problem really beyond that. Controls is largely yeah, a solved problem. What our controls engineers here are doing a lot of the time is adding new cars, and making the new car models as good as the old cars.

[0:31:47.8] JM: Okay. Is there a lot of demand for – I mean, do you have people that are like, “Oh, I'm ready to join as soon as you get my Ford Pinto interfaced.”

[0:31:55.4] GH: Oh, yeah. When we add cars, we see – we're getting into the lower hanging fruit now, but we added Toyota. Toyota is huge, just because they're so popular.

[0:32:03.5] JM: Fascinating. Let's go higher level. It sounds like, I guess your – what do you spend your time thinking about in the company? Should we just skip to the path area, where you're working on the machine learning models? Is that your area of focus, or are you –

because it sounds like there are some different teams and it is not just a singular focus. You've got some different areas of the project that are being worked on?

[0:32:24.2] GH: Yeah. I spend a lot of my time, I mean, we're a 15-person company now, so I do spend a lot of time managing. As far as the technical problems that I work on, maps have been consuming a lot of my time lately. We want to open source HD maps of all the highways in America by the end of the year. The thing is, the companies that have done this before us, because there are data sets you can buy now from Azure maps, or TomTom that already have this, but that was company spent millions of dollars to create those maps. Those companies literally outfitted fleets of cars with Lidars and super expensive GPS systems. We want to use our existing fleet that's already on the road and just make maps by processing that data in really good ways.

[0:33:02.8] JM: Do you have enough coverage of – Did you say the world, or United States?

[0:33:07.2] GH: We want to do the highways, the interstates in the United States. We have enough coverage of. There aren't that many interstates. There's only 40,000 miles of interstate.

[0:33:15.1] JM: Got it. When you say high definition maps, you mean if you've had – how many cars driving over an interstate do you need to generate a high definition map?

[0:33:23.2] GH: About five.

[0:33:24.1] JM: You see five iterations for each interstate?

[0:33:26.4] GH: Yeah. If we have five paths and they can't all be in the same way, and they have to be spread out of it. Yeah, then we can get everything we need pretty much.

[0:33:33.0] JM: Why did you set that goal?

[0:33:34.2] GH: Well, so part of the coolness of maps is we get to figure out where all our cars precisely are. We're improving our localizer, which is pretty much all like Waymo and stuff we need the Lidar for. They really only need the Lidar to figure out where they are. People think that

the Lidar is more about seeing objects in front of you. Yeah, you could do that really well now with computer vision. What you can't do that well yet with computer vision is localize yourself to 10 centimeters of accuracy, and that's what we're working on building.

The infrastructure that we're using to build maps has many other purposes in the company besides just making maps. The maps are a cool thing that we can give away for free, and hopefully not much of DC is invested in maps. I would love to see their investments go to zero.

[0:34:17.2] JM: Finding your point in space, is that like if I'm at some point in space driving on an interstate and I've got a camera facing forward on my car, you should be able to hash that image of the interstate, or the combination of that image in the interstate and maybe your GPS, or something to a known point in space in mapping space? Help me understand what you're talking about when you're trying to improve this accuracy.

[0:34:44.7] GH: Yeah. We use – it's a big Kalman filter. It's a big extended Kalman filter that fuses together everything we currently know about the car into a couple of state variables. Some of these state variables are like, the position, the velocity of the orientation, those kind of things.

[0:34:58.8] JM: What's a Kalman filter?

[0:35:00.4] GH: A Kalman filter is a way of combining multiple sensors together to get one estimate of a state. For example, we have a gyroscope inside of the phone and the gyroscope will measure your rotation in three dimensions; your roll, your pitch and your yaw. It's a degrees per second, right? Gyros have this interesting property, where they're called gyro bias. Occasionally, depending on the temperature of the gyro, depending on a few other things, the zero is not zero. When the gyro says zero, it's lying to you. It's actually rotating a little, or it says it's rotating a little when it's actually not rotating.

Now it's very easy to correct for gyro bias by using vision, because if you have a camera looking forward, when a camera's not rotating, it's really easy to tell cameras don't have the same bias. If you want to get a true estimate of the rotation of the car, maybe you combine the camera, which is very good at removing the bias with the gyro, which is very good at measuring changes

in that degrees per second and your gyro is a very high-frequency relatively low noise, you combine them and you get one estimate. That's what a Kalman filter does.

[0:35:58.9] JM: Cool. Okay, so sorry. Continue on how – what you're doing to improve the accuracy of known – where the car is in space.

[0:36:06.4] GH: We then put everything into the Kalman filter. You can take your camera, you can take your initial sensors, your gyro and your accelerometer. We have a raw GPS processor, so a lot of times people think that GPS is just something that outputs a latitude or longitude in the height. If you've already gotten to the latitude, longitude and height, the location is already way too garbage for us to use it. That's usually accurate to about 10 meters.

What we get from our GPS, and we've shipped for sale on shop.comma.ai a high precision GPS, called the gray panda, and it tells us not just where it is, but it tells us where each satellite is, and it tells us the measured distance to that satellite. Okay, it doesn't exactly tell you the measured distance. It tells you something called the pseudo-range, because you don't have a shared time clock between you and the satellite. The pseudo range is the distance to the satellite, plus some constant. That constant is the same for all 10 satellites you currently see.

You now have 10 equations, you have 10 linear equations to solve for your position based on the GPS. We put all of these equations into one big solver and get a really good estimate of where the car is.

[0:37:07.0] JM: Then that GPS estimate coupled with things like what's going on in the camera that's front-facing in front of the car and all that information that's pulled in there, between those different variables you can triangulate where the car is in mapping space.

[0:37:22.0] GH: Yeah. We can dead reckon extremely well, which is dead reckon is you know where you are, give or take some small offset. Then the reason we need five coverage is because we can combine the five together. We can take all those sensors from all those times, and it's really cool, you don't want the five to all be at the same time if you have different times; the satellites are in different locations. It just gives you even more equations to solve jointly for

the whole position and the whole map. This is the stuff that nobody else has built yet. No one even knows we have this stuff. This is the first time I'm talking about it.

[0:37:51.4] JM: How are you sure that other people don't know about that?

[0:37:54.2] GH: I'm not saying they don't know about it. I'm saying that, I don't think anybody else has really built it. Because I mean, it's not because nobody else can, it's just because most of the other people in self-driving don't see cost as an issue. They are like, "I don't care if the car costs half a million dollars." Each one of those Waymo cars costs half a million dollars. I'm like, "But that doesn't – I don't think that works." Uber drivers are cheaper than you think. I know you live in the Bay Area and I think everyone gets a 100k salaries, but they don't.

[0:38:18.1] JM: Okay, so you said that you have with this Kalman filter some stuff and you figure out where the car is in mapping space to some degree of accuracy, how much margin of error do you have on that accuracy?

[0:38:30.7] GH: When we release our maps, we want everything, like we're going to – so the HD maps contain every lane line basically. The location of every lane line and we're trying to get below 10 centimeters. Maybe something like the mean squared error is going to be 10.

[0:38:43.2] JM: Okay. Once you get – let's say you hit your goal of getting this mapping data by the end of the year. How can you leverage that work? What can you bootstrap that to? I guess, you get to bootstrap it to what you're talking about, the highly accurate state-space description.

[0:38:57.7] GH: Yeah. I mean, you can do really cool things once you have this map. The simplest thing we can do with it is every car right now has a model, and the model says, "I think this should be the path." If we can also localize the cars in a map, the map says, "I think this should be the path." Then we can do intelligent things, like fuse the two paths together. We can say, "Okay, well that's the uncertainty there. That's the uncertainty there, and we can make the car drive even more precisely."

We can also put things like red lights and stop signs in the map very easily. People think, "Oh, we're detecting red lights. That's a very easy problem. I can download a app on the App Store

that can do that." Yes, you can. You can even get that to be really, really accurate too. The problem though is figuring out where you should stop. Yes. I understand that there's a red light in the image. Okay, does it apply to you? Where do you stop? That's the stuff that you can very easily put in a map. How do you build that map? Well, where did people stop when they saw the red light? You're going to need to label it, right? Just where did people stop? I'd check the light, I see people stop there. I've seen dirty people stop at that light. They all stopped in pretty much the same place. Okay, got that in map.

[0:39:53.0] JM: Yeah, and what other – I mean, so red lights and stop signs, it's a very discrete example. I imagine there's other subtleties like, oh, there's a blind turn here for example, and we want to be able to train our models so that when you're approaching a blind turn, slow down, right? How generalized can you make that – because it sounds like – what you're talking about is okay, you can have in your mapping data discrete examples of a stop sign, but in different and maybe in certain places in Utah, you want to stop 10 feet behind the stop sign, instead of 5 feet and behind the stop sign, because in Utah, I don't know the sun is at a certain angle or something.

I'm just saying the more general case is you have circumstances in driving where it's hard to discreetly rule-based wise articulate how you should be driving, but this is enough of a blind turn that you want to be slowing down for it, and how do you model that?

[0:40:47.5] GH: I don't need to model that. I have big data, right? I've seen 20 people go around that turn before, here was the average speed. Okay, go that speed.

[0:40:53.4] JM: Average speed. I guess, average speed and direction and path. I guess, path just encapsulates how people should be behaving.

[0:41:01.4] GH: It's even more magical than this too. This is the real magic of coming, I probably shouldn't talk about. You realize that, let's even say we get that map and the map is only 95% accurate. Well, that's not going to really be accurate enough to drive a car, but it's accurate enough to train a machine learning model to drive a car. Your machine learning data sets don't need to be a 100% accurate.

[0:41:18.2] JM: Can you talk it more detail? What do you mean?

[0:41:19.9] GH: Let's say I added some random noise to M-nest, right? Let's say I messed with 5% of the labels in M-nest. You know M-nest?

[0:41:25.3] JM: Sure, yeah. The hand-drawn things.

[0:41:27.4] GH: Yeah. Hand-drawn to have that. I put in 5% of noisy data, right? When I train my model, it's going to do better than 95% accuracy on the test side.

[SPONSOR MESSAGE]

[0:41:42.3] JM: Citus Data can scale your PostgresSQL database horizontally. For many of you, your PostgresSQL database is the heart of your application. You chose PostgresSQL because you trust it. After all, PostgresSQL is battle-tested, trustworthy database software.

Are you spending more and more time dealing with scalability issues? Citus distributes your data and your queries across multiple nodes. Are your queries getting slow? Citus can parallelize your SQL queries across multiple nodes, dramatically speeding them up and giving you much lower latency. Are you worried about hitting the limits of single node PostgresSQL and not being able to grow your app, or having to spend your time on database infrastructure instead of creating new features for your application? Available as open source, as a database, as a service and as enterprise software, Citus makes it simple to shard PostgresSQL.

Go to citusdata.com/sedaily to learn more about how Citus transforms PostgresSQL into a distributed database. That's C-I-T-U-S-D-A-T-A.com/sedaily, citusdata.com/sedaily. Get back the time that you're spending on database operations. Companies like Algolia, Prosperworks and Cisco are all using Citus, so they no longer have to worry about scaling their database. Try it yourself at citusdata.com/sedaily. That's citusdata.com/sedaily.

Thank you to Citus Data for being a sponsor of Software Engineering Daily.

[INTERVIEW CONTINUED]

[0:43:27.8] JM: I get it. You get a really reasonable resolution set of data and you build paths based off of that, and then you have the trajectory of the car and you've got the trajectory to path state machine calibration. I think that outlines some of your trajectory as a company. I guess, we can talk a little bit more about the machine learning tactics, or the problems that you're encountering, or the problems that you're focused on at the machine learning level when you're trying to build these paths. Can you just talk a little bit about that?

[0:44:05.6] GH: The truth about machine learning is only people think oh, machine learning is playing with nets and tweaking your activation functions and changing the number of layers. That stuff is all what researchers do. We don't really do research, because we don't have to, right? The problem is not that the nets aren't good enough. The problem is what's the data set that you're training on?

In reality, a lot of machine learning problems in industry are infrastructural problems. Our Carpathia goes into this a bit, where it's like well look, here's the 5% of the system that's actually the net. Here's the 95% of the system that's all about cleaning up data and getting it labeled and putting in the right way and throwing out bad data and validating the net at the end, right? Then running it in production. Yeah. I mean, that's what a lot of the challenges are. I could talk about some fancy, "Oh, well our Gans are unstable," but we don't use Gans. It's all supervised learning.

[0:44:52.4] JM: That's such a relief to hear, because I have this – this deep learning textbook that's been sitting on my table for a year and a half now, and I'm stuck at the matrix algebra part. Luckily, I've been interviewing people about infrastructure for a while, so I can – I think I have a reasonable understanding of that stuff.

[0:45:07.8] GH: Oh, yeah. Deep learning, it's all an infrastructure problem.

[0:45:10.3] JM: Interesting. Tell me more about that. Where do you hold the data? In terms of the infrastructure problem, I guess I've heard these stories about how the self-driving cars, or the "self-driving cars" are driving around collecting data and then they take the disks out of the trunk of the car and load the disks into a data center, or something like that?

[0:45:30.9] GH: This is how Streetview did for the longest time. All Google Streetview was hard drives and FedEx. I mean, your latency might be very high, but your bandwidth is very good. Your bandwidth it's actually insanely good. FedEx's bandwidth. No, we don't do that. We use the internet. We compress.

[0:45:43.8] JM: Okay. The cars driving around; are you compressing it on the fly, or you're like every hour you're taking a snapshot and and/or sending it over when you have Wi-Fi, or something? Tell me more about the data collection and ingress process.

[0:45:58.1] GH: Sure. It's HVAC. The compression is just a normal video codec. A nice one. HVAC is pretty nice. Then yeah, it saves them all to the disk and then you can – it's a toggle. They have a cellular radio in them too. If you have unlimited data, you can upload over cellular. If you don't, you can turn that off and then it'll upload when you get home over Wi-Fi.

[0:46:15.8] JM: It's raw video. That's the data that you're capturing?

[0:46:18.7] GH: It's 5 megabit. It's less than HD video in Netflix.

[0:46:22.4] JM: Okay. The cars are driving around. I know you went over this hardware already. I think, what is, there are three different hardware pieces; you've got the camera at the front of the car, you've got the CAN Bus interface thing, and then you've got a user interface component, are those the three pieces?

[0:46:37.3] GH: The user interface and the camera are the same. The three pieces are the user interface – the phone, the fancy phone. It's got a nice cooling system and stuff. You can buy the phone and try to run it on it, but it won't work very well, because it won't be kept cool and your USB is going to be flaky. We have a little USB conditioner board. That's the main interface piece.

Then yeah, we have the CAN, that's the main camera user interface piece. We have the CAN interface. We have one more thing called the giraffe. The giraffe is basically just a wire, but it's a

nice wire and it lets you flip between the stock system and open pilot, so you can toggle which one you want.

[0:47:08.4] JM: Okay. Cool.

[0:47:09.3] GH: It's a wire. It's a wire. Think about, like you have a laptop, yeah, I plug my laptop into the internet to talk on the podcast, the Ethernet. I have my Macbook, I have my little USB Ethernet dongle and then I have the Ethernet wire, so same two parts.

[0:47:22.4] JM: By the way, altogether those three things are what? A thousand bucks, is that what the cost is?

[0:47:25.8] GH: A little under. Yeah.

[0:47:26.7] JM: Cool. Okay, so back to the infrastructure issues. Okay, so you can collect the videos pretty easily. You said some get lost in the ether, I don't know what causes that.

[0:47:37.1] GH: People delete them, people uploads fail, people reformat their ether, reformat theory on – up close don't fail. It retries nicely, but –

[0:47:46.8] JM: Sure, okay. Video hits your infrastructure. What do you do with it?

[0:47:50.2] GH: We have a big pipeline and some standard data processing pipeline. In the efforts of driving down disengagements, last month we launched this thing my.comma.ai. It lets you explore all of your drives. It lets you look through basically a timeline of when your car was driving and it lights up green when the system was engaged, blue when you were driving and then red or orange when you were taking control. Then what we're asking the users to do is label the reason that they took control at each place.

[0:48:15.5] JM: Do you send them e-mails and ask them to do it, or are people just – I mean, I guess, these are pretty engaged hacker people, so they're probably perfectly willing to do this stuff.

[0:48:24.4] GH: These are very enthusiastic people. Again, for now it's less of a we're using this machine learning and more of a, we're just, okay, you label your data, you show me where your car is disengaging. If you're one of the people who's enthusiastic enough to label, guess who I'm going to prioritize making the car better for?

[0:48:39.3] JM: Yeah. Okay, interesting. Again, disengagements are when the person intercedes. You're basically assuming the person that's driving the car is keeping track of what the "self-driving car", or I know you don't like that term, but whatever. The autonomous – you probably don't like that term either. The autopilot engaged, or what's the term?

[0:48:59.9] GH: What the computer is doing.

[0:49:01.0] JM: The lane assistant.

[0:49:02.6] GH: What the computer – computerized driver, driving agent.

[0:49:05.2] JM: Yes. Yes, the driving agent. The user is watching what the driving agent is doing and occasionally the user is saying this is not right, and grabs the steering wheel, or presses the gas. That is going to get labeled. That point in time where the user interceded is going to get labeled, and then the video hits your end and then you can flag those video frames and then present them to the user and say, "Hey, what made you do this?"

[0:49:27.9] GH: Yeah, that's pretty much what it is. I mean, most disengagements are not safety related. Most disengagements are just like, "Oh, I wanted to get off that exit, or I wanted to go faster."

[0:49:36.1] JM: Those ones you can just ignore?

[0:49:37.9] GH: Well, no. I mean, they're interesting data points. We didn't do what the user wanted us to do.

[0:49:42.4] JM: Fair enough. Yeah.

[0:49:43.2] GH: We have to make it better.

[0:49:45.5] JM: Yeah. Then so, do you have a schema for the labels that people can say? Can they say that I was going to hit a car here?

[0:49:52.6] GH: Yeah. We have a basic set of labels, and then we have another and you fill in a box. The labels are pretty good. This is something we're going to really push on starting next year. Right now, the car will just continue in a straight line. Maybe next year, we start thinking about, “Okay, let's actually start taking exits. Let's figure out what the user’s intent is. Where does the user want to go? What's the desire?” Yeah, that's something that becomes easier to think about when you have maps.

[0:50:15.1] JM: Right. Okay. Right now, the car just stays in a straight line. It only works in circumstances where you're going in a straight line, where you're not making turns?

[0:50:23.3] GH: No, no, no, no. That's not true. The car will work all the time, but if you have choices, right? Say you're in the right lane on a highway, you have a choice; you could continue straight on the highway, or you could get off the exit. If the system's left alone, it will always take whatever the more common choice is.

[0:50:39.0] JM: Oh, okay. I see. I should know this. When you get in the car, you enter some destination and the car –

[0:50:44.2] GH: No.

[0:50:44.7] JM: No, you don’t. Okay.

[0:50:45.7] GH: You don’t enter any destination. When you come to a fork in the road, it'll do whichever one looks like the obvious continuation of the road.

[0:50:52.6] JM: Okay.

[0:50:53.7] GH: If neither of them do, it'll just pick randomly.

[0:50:55.7] JM: Okay. Understand, how does the car know where I'm going if I don't put in a destination?

[0:50:59.8] GH: It doesn't. Think about you driving, right? If you drive maybe 40 minutes. Think about you as the passenger in a car and the driver has no idea where they're going. How often do you have to tell the driver anything?

[0:51:08.9] JM: Let's see. How often to have to tell the driver anything? I guess not very often.

[0:51:12.5] GH: Exactly. Think about it like that, right? Every time you would have to tell that driver something, you have to take control of the car, but most of the time the driver just keep on straight, just keep going straight, no, no, no, keep going.

[0:51:22.7] JM: Is that to say that I take my trip to the office every day and the car learns how to drive from my trips to the office?

[0:51:29.7] GH: I mean, it learns to drive from all the data. It took time. Not learns to drive. This is an interesting distinction. It learns the definition of driving and it learns the definition of driving from everybody. It doesn't learn to drive from your drive. This is a very layman's understanding of this stuff, right? To say that a imagenet classifier learns to recognize chairs, to recognize that's amorphized. Don't you actually mean learns to like –

[0:51:57.1] JM: Probabilistically estimate.

[0:51:59.1] GH: Yeah. Yeah, yeah, yeah. Or even just learns the definition of chair, right? That's more what it's doing.

[0:52:04.4] JM: Okay. Definition of chair according to some set of polygons, or colors from an image.

[0:52:10.3] GH: According to who knows what, according to here's a million examples of chairs and here's a million examples of not chairs. What do those ones have in common that those ones don't? Okay, that's a chair.

[0:52:18.1] JM: Right. Okay. Not to get lost in this semantic discussion, so when I get in the car on the first time I plug in my comma.ai Prius, it's just going to take me on the road traveled by the most cars that have recorded?

[0:52:34.3] GH: It's not going to take you anywhere. It's a lot more like a cruise control system, right? You get the car up to speed. You're like, "Okay, now I want to go. Okay engage." It will continue driving for you.

[0:52:45.8] JM: Okay.

[0:52:46.5] GH: Which is the truth is, that's most of the time what you want, right? How annoying is it if you had to put in your destination in the nav system every time you wanted to drive to work? The thing talk to you every time. I know, I know, I know I have to make that left. Yeah, I make it every day, right? You do the parts that are something and let it do all the unconscious parts.

[0:53:06.6] JM: Yeah, and I mean, at the same time, it's like you're on your 25-minute commute to work and some days you're tired and some days you might be more liable to slam into a car in front of you and I'm guessing that a lot of this the appeal here in this stage of things is the car may help you from slamming into the car in front of – No? Not so much today?

[0:53:29.5] GH: I mean, it might, but I don't sell safety, right? You're trying to sell safety. Don't sell safety. Build safe things, but don't sell safety. It's cruise control, right? This is nice. The worst part, driving in traffic. Driving in traffic is just terrible. It's just like, "Oh, I have to do this. Oh, this this hurts," right? Versus, you turn the system on and then you just watch. Think of the difference between being a passenger in traffic between and being a driver in traffic.

[0:53:50.2] JM: Well, a passenger in traffic I'm staring down at my phone.

[0:53:53.1] GH: Well, don't do that. You can't do that anymore. We have driver monitoring now. We're watching the drivers.

[0:53:58.2] JM: Okay. I mean, honestly speaking in that circumstance I'm not sure what I get, because I guess I could listen to music without moving my foot on the pedal, or I could focus on my podcast a little bit more.

[0:54:09.8] GH: Do you use cruise control?

[0:54:11.1] JM: There was a period of time where I was driving from Bellevue to Seattle to work at Amazon, and I did not use cruise control under that circumstance. The only circumstance I used cruise control in the past is when I was driving to San Antonio from Austin, which is like an hour-and-a-half drive.

[0:54:25.5] GH: Sure. Why don't you use cruise control?

[0:54:26.7] JM: I'm a little scared of it honestly. I'm afraid that if I put on cruise control, I'm going to yeah, I don't know. I'm incentivizing myself to give more control away, and I'm a little afraid of that. I mean, maybe I'm an anomaly there.

[0:54:41.9] GH: I mean, you can think about it like that. You can also think about it as you can now focus on other parts of the driving task, right? The driving task has a lot of parts. The driving task has, I don't know, I mean, I don't try to sell these things at all. Don't buy one. I'm not trying to make any – everything comma.ai sells is completely useless.

[0:55:02.5] JM: Well, it's a bootstrap.

[0:55:03.7] GH: No, no, no. It's not a bootstrap. It's just useless.

[0:55:05.6] JM: Useless, okay.

[0:55:06.8] GH: Yeah, useless. But it's so good. I mean, winning self-driving cars will be so much fun.

[0:55:11.8] JM: Yeah. I mean, so okay, I want to talk a little bit more about this data pipeline, because we've got 10 minutes left, and I want to talk a little more about that and then we can zoom out a bit. This data pipeline, okay what is interesting about it? You're processing video, you're doing machine learning on video, just I don't know, give me the abbreviated description of what's interesting and what's hard about it.

[0:55:31.4] GH: I think we're now at like – it's half a percent of a YouTube with the video we get. It's a huge amount of video being uploaded constantly. Just categorizing it, putting it all in its right place, so the data pipeline. You got to, okay what if you have out of memory? Well, you got to retry. What if you're getting an error? Well, and maybe you don't retry. Maybe you flag those as areas on the stage? It's a multi-stage pipeline. First, we extract a bunch of basic stuff from the data, put it into this new canonical format, and then we can do things like okay, add it to our internal map, pre-process it for the user's explorer thing. Maybe we're going to add a stage pretty soon that takes guesses for all the reasons for the disengagement, right?

The users shouldn't have to label everything from scratch. They can just refine the labels that our machine learning system uploads, right? You got to run that machine learning in production, then you got to deploy that and then you got to install that.

[0:56:17.0] JM: What framework are you using, and what cloud provider are you using?

[0:56:19.7] GH: We're using Azure. As far as a framework for the tasks, we've written it. There is really a good one, except maybe for Google Cloud dataflow, but that one's only on Google Cloud. Azure gave us a lot of free money, so we love that Azure.

[0:56:32.8] JM: Nice. Google Cloud dataflow. Oh, yeah. Google Cloud dataflow, and that's the spark interface, or something like that. What's appealing about that?

[0:56:41.4] GH: Yeah. It's based on Apache Beam, but they have the good Python client for it. Everything we do is in Python. I don't want to write any Java, right? I don't want to use Spark and this Java. Yeah, we've effectively written something. We're using I think some messes, like RabbitMQ, or one of those message queues. Actually much of Redis now. I think we're really

just using Redis and a bunch of long-running processes, and that thing is pushed onto the Redis queue. I've also moved off of – off of infrastructure and controls now. I work on mapping and machine learning.

Yeah, it's basically like when something comes in and it adds a event to a Redis queue, and then so now we – you start up a scheduler process, which tracks that file throughout its lifetime, and then can that process can launch more sub-processes and check for our failures and check for timeouts and check for all those things?

[0:57:28.2] JM: Cool. Yeah, maybe if any of your infrastructure engineers are listening, they can feel free to ping me. I'd love to do a show on that. I'm sure that would be an interesting show. Let's zoom out a little bit, because that's what you're doing in your job. I mean, so first of all, the process of zooming out from being the hacker who does the 10-hour or twitch sessions of simultaneous localization and mapping, to being a manager, this is more of a general software engineering question, but how have you found that transition? What's been difficult about it for you?

[0:57:57.3] GH: I don't know. I'd rather work with computers and people, but I also like high leverage. You can view people as just like – it still like an IDE. This is really just a reflective on how bad IDEs are, that people are better. I don't know. We have very good people here now. We have very good – our bar is extremely high. We fired half the people who've ever worked here. It's hard to get a job here. It's hard to keep a job here, but the advantage for this is everyone respects their co-workers, because they're all very good.

[0:58:25.1] JM: Definitely. Simulations, this is something that I think you do not do, but it's something that Waymo does a ton of. I read this, I think Atlantic article about Waymo's development and how much simulation they do. Tell me about the pros and cons of simulation.

[0:58:41.8] GH: Well, so you can think about simulation, it's a little bit better – building a simulator is a little bit better than trying to ride a car from the rules. Think about riding a car from the rules is like the computer vision problem, and the simulator problem is the computer graphics problem, right? It is computer graphics, it's how these simulators are actually written. Now part of the problem with using simulators as a scalable strategy to get to level 4 is at least if

you're hand coding your simulator, you're going to run into the same problem as hand coding the driving system. There's going to be all these edge cases that you're going to miss out. Simulators are good for – Chris Urmson has a quote that I like about simulator. Simulation is doomed to succeed. You can always build a system that succeeds in your simulator. Does that mean it works in the real world? Unclear. How good's your simulator?

[0:59:23.1] JM: Okay. You're just throwing it out entirely?

[0:59:26.3] GH: Well, no. We have a paper. We have one paper actually. It's on archive. It's self-published, but it's called learning a driving simulator, and that's when we'll start to use simulators. Once we can learn a simulator from data, because if you learned a simulator from data, you've now fixed this edge case problem, right? We don't have to worry about, "Yeah, but learning simulators from data is extremely hard." I'm waiting for I'm waiting for a few more papers to come out.

One of the cool things that I really like that came out machine learning last year was unsupervised learning of depth nets, which we've now started to do internally. You can learn just from data to predict the depth of every object, right? No labeling required. We have a lot of data.

[1:00:01.1] JM: How do you do that? How do you that with unsupervised learning?

[1:00:06.5] GH: It uses a technique called structure motion. I mean, you can effectively think that even though our car only has one camera, it actually has multiple cameras, right? Do you see the multiple cameras? Even though our car only has one camera inside of it, do you see that how it could have like two?

[1:00:16.5] JM: As you advance through time, your observe –

[1:00:18.2] GH: Yes, absolutely. Absolutely. Yeah, very good. Yeah. You have a picture taken now and a picture taken maybe 10 meters ahead on the road, so now you can use those things and you can triangulate points in the image.

[1:00:28.7] JM: You just pre-baked in physics?

[1:00:31.2] GH: Oh, yes. Yeah, so we pre-baked in the –

[1:00:33.9] JM: Physics is your labeled data set.

[1:00:36.6] GH: Sure. We pre-baked in the rules of a multi-view geometry.

[1:00:40.0] JM: That's pretty cool.

[1:00:40.7] GH: Yeah, this has been a cool new trend. Yeah, if you can – if there's very distinct rules that truly do summarize all of this, then yeah, of course bake them in. That is very short. It's part of your net architecture.

[1:00:51.3] JM: The Udacity team, aren't they doing an open-source self-driving car?

[1:00:55.4] GH: No, there is a – The other open source self-driving cars maybe you're thinking the Baidu team, Apollo?

[1:00:59.1] JM: Baidu, okay.

[1:01:00.3] GH: Yeah, Baidu is doing one. What I really want to see from Baidu is the problem is the only system their car runs on cost like a $100,000 and has to be bought from this specialty company, but I don't think that works. I think that if you want to make a open source self-driving car, you want to make it well, something people could buy and run. I imagine Linux only ran on a $1,000 Xeon processors. It'll never would've been adopted. It's got to run on the Hondas and Toyotas, right?

[1:01:23.0] JM: Okay, you've been really generous with your time. Just a couple more high-level questions. I heard this interview that I think Siraj Raval did with you on YouTube. That guy's pretty hilarious, by the way. I hadn't seen anything of his, but until I watched your interview with him and it was pretty funny. One thing he asked you is about the singularity and you had a pretty strong opinion on the singularity. Tell me what the singularity means to you and why you think it's inevitable.

[1:01:45.5] GH: When computers can do every task better than humans. I think it's inevitable, because computers are getting smarter and humans aren't.

[1:01:52.3] JM: Okay. Do you have a time horizon, or are there some triggering events in the future that you're looking for for indicators that this is going to be happening?

[1:02:02.6] GH: Oh, here's what I do to think about. I have about a 100 trillion weights, I have about a 100 trillion synaptic weights in my brain, I think we're about a million fold off from that. I think some of the biggest neural nets, well let's see, so the neural net that we use on our car, actually the tiny one only has 1 million weights, so that's actually off by a factor of a 100 million, but you can train big ones now that have about a 100 million weight, so you're off by about a factor of a million. A million, so that's 2 to the 20, I've probably got that wrong. Oh, yeah, it's 2 to 20, because 2 to 24 is 16 million. 20 Moore's laws, so 18 months and 30 years.

[1:02:31.8] JM: 30 years.

[1:02:33.1] GH: Contrary to popular, belief Moore's law is not over, Intel is over.

[1:02:36.7] JM: Okay.

[1:02:38.8] GH: Heavily, Intel, Intel 10 nanometer Fiasco, short Intel stock today.

[1:02:44.6] JM: That sounds like a good place to stop. George Hotz, it's been really fun talking to you. I'm a fan of your work, I'm a fan of your thesis and your attitude makes for some entertaining interviews.

[1:02:54.1] GH: Sounds good.

[END OF INTERVIEW]

[1:02:58.1] JM: GoCD is a continuous delivery tool created by ThoughtWorks. It’s open source and free to use and GoCD has all the features you need for continuous delivery. Model your

deployment pipelines without installing any plugins. Use the value stream map to visualize your end-to-end workflow. If you use Kubernetes, GoCD is a natural fit to add continuous delivery to your project.

With GoCD running on Kubernetes, you define your build workflow and let GoCD provision and scale your infrastructure on the fly. GoCD agents use Kubernetes to scale as needed. Check out gocd.org/sedaily and learn about how you can get started. GoCD was built with the learnings of the ThoughtWorks engineering team, who have talked about building the product in previous episodes of Software Engineering Daily, and it's great to see the continued progress on GoCD with the new Kubernetes integrations.

You can check it out for yourself at gocd.org/sedaily. Thank you so much to ThoughtWorks for being a long-time sponsor of Software Engineering Daily. We're proud to have ThoughtWorks and GoCD as sponsors of the show.

[END]