Catalyzing Computing Podcast Episode 32 – Autonomous Flight and Landing on Mars with Behçet Açikmeşe (Part 2)

The transcript below is lightly edited for readability. Listen to “Autonomous Flight and Landing on Mars with Behçet Açikmeşe (Part 2)” here.

[Intro - 00:10]	1
[Control Systems and Spaceflight - 01:31]	2
[Convex vs. Non-convex Optimization - 10:25]	6
[Markov Decision Process - 14:02]	8
[Model Predictive Control - 19:43]	10
[Lossless Convexification - 21:04]	11
[University of Washington Autonomous Controls Lab - 25:00]	13
[Assured Autonomy - 31:15]	16
[Outro - 36:50]	19

[Intro - 00:10]

Khari: Hello, I'm your host, Khari Douglas, and welcome to Catalyzing Computing, the official podcast of the Computing Community Consortium. The Computing Community Consortium, or CCC for short, is a programmatic committee of the Computing Research Association. The mission of the CCC is to catalyze the computing research community and enable the pursuit of innovative, high-impact research.

In this episode of the podcast, I interview Dr. Behçet Açikmeşe. Behçet was a technologist and a senior member of the Guidance and Control Analysis Group at the NASA Jet Propulsion Laboratory, from 2003 to 2012, where he developed guidance, control, and estimation algorithms for formation flying spacecraft and distributed network systems, proximity operations around asteroids and comets,

and planetary landing. He is currently a professor in Aeronautics and Astronautics, as well as Electrical and Computer Engineering at the University of Washington. He is also a member of their Autonomous Controls Lab. In this episode, Dr. Açikmeşe discusses control theory and the University of Washington Autonomous Controls Lab.

This is part two of my interview with Dr. Açikmeşe. If you haven't heard part one and would like to, go catch that and come right back. Enjoy.

[Control Systems and Spaceflight - 01:31]

Khari: So we've kind of alluded to it, but I guess we'll dive into it now. What is a control system and what is control theory?

Behçet: This is what I tell my undergraduate students — I teach the undergraduate controls courses here at the University of Washington — the best example is us, human beings, and our navigation if you like. If I tell you to close your eyes and go to an object which is like ten yards away from you, you may do a decent job with your eyes closed. But if I turn you around once and I ask you to do the same thing, then you will probably do a very bad job. If I turn you two or three times you will do worse probably; if I put the target far away from you, not ten yards but 100 yards, even if I tell you, “Go straight in that direction,” you will miss it badly I'm sure.

Not you but all of us.

[Laughter]

The reason is you don't have sensing. Ultimately, you don't have active sensing. What we do is, when we walk towards the target you know the general direction but we look at it quite often. Why? Because we need that information, that feedback, to correct for any errors that accumulated during our walk. If I also tie your feet down, you can't move at all. Your actuators are dead. What's happening is your eyes are detecting and

estimating the location of the target relative to your current position. Your legs are executing comments from your brain or wherever, which is doing the processing. Your brain is doing the image processing. It's generating the control commands that you have to take and sending them to your legs and your legs are doing it.

This is what we call a closed-loop system: sensing the object, processing the data, deciding on what to do, and actuating our actuators, which are our legs in this case, and walking towards the target. This is what we call a closed-loop feedback system. Actually closed-loop feedback control system to be more precise, because it's ultimately doing the controls of the system. That's what we do with any vehicle; for example, a drone, like a quadrotor people fly these days. In my lab, we develop these autonomous quadrotors and what do they do? When they fly outdoors their eyes are the GPS sensor. GPS tells where they are. It's not eyes, but it's a form of information that they need.

Then that information together with other onboard sensors like accelerometers and such that generate acceleration information...I know the state of the vehicle and I know where I want to go, let’s say I want to go to some target, some coordinates. I know where I am relative to that target and then I turn that information into some control actions. This is done by the onboard control algorithms that tell each actuator, which are propellers, [what to do]. It sends some comments to them, some numbers. Those numbers basically determine propeller speed, and that determines the force that it exerts on the vehicle.

I have four propellers let’s say, or six, however many you have. They generate an effective force and an effective torque. Torque causes rotation, force causes translation. By controlling them that way I can go to the target I want. So this is another feedback control system. To drive these feedback control systems you need algorithms to process the data and to decide on the actions, the job of a control engineer is to design and implement those algorithms.

Khari: Ok, that makes sense. It's obvious why that would be important for space flight or landing; but, more specifically, how do you do the real-time optimization of spacecraft flight?

Behçet: Oh, that's a good question. That's actually at the heart of my research. Maybe I'm going to get a bit more technical now, into the gritty, nitty things. Basically, if you think of any control system it's also a hierarchy of decision-making. One thing about that...again, a drone example, a quadrotor example. If I'm just hovering somewhere — I'm not going anywhere I'm just hovering — I still have to control the vehicle.

First of all, I have to control my altitude and my horizontal location to make sure that even if there are some winds coming, some aircurrents, maybe somebody is pushing me, some students coming and touching me, I have to come back. I'm still doing a lot of work to keep myself there. This is the lowest form of control, reacting to uncertainties just to keep your location. That's always active, but there is also a planning stage.

Now, let's say I'm not just going to hover but I want to go to a point maybe 100 yards away from me, but there are obstacles in the middle. Maybe there is a person in the middle, or ten people, and some boxes along the way that I shouldn't hit and so on. So now I have to generate a path that takes me from point A to B while avoiding all these obstacles. I have to generate a trajectory and track that trajectory, meaning that I have to plan a trajectory like we do, right? I plan a path going from A to B then I start executing.

What the quadrotor does in this case is...this generation of the path is a high-level of decision function. Then we execute this path, meaning that we try to keep up this path and get to the point B we want. If we keep up with this path that we plan, that means we are making our mission objectives. Designing that path, which is sometimes called trajectory planning or motion planning is a high-level function. And what we do with optimization is that we convert this trajectory planning or motion problem into an optimization problem.

An optimization problem is a generic decision making problem. It’s a math problem, which says, “Optimize the reward or a cost matrix subject a bunch of mission constraints.” Like I want to optimize my mission rewards while minimizing, maybe, my mission penalties like fuel. I don’t want to spend too much fuel or something like that. You may have different mission constraints. So, any decision making problem can be set that way, but these motion planning problems also lend themselves to this formalism.

Now, just because you formulated it doesn't mean it can be solved. Actually, most optimization problems are not real-time tractable. You may not be able to solve them with confidence. My approach, which has been quite...it was actually quite a leap for rocket landing and aerospace applications. [In my approach] I do another conversion, one more conversion where I take these optimization problems and turn them into what's known as convex optimization problems.

Convex optimization problems are a subclass of optimization problems for which we have really good algorithms, which ensure, within some numerical guarantees, that we will get to a solution. Not only that, we will get to the best solution in many cases. So those are good optimization problems, the ones that you want to have. What I did is convert the optimization problem, the original one, which is typically non-convex (which means it's really hard) to something that's tractable in real-time.

Once I achieved that mathematically, then what that means is I can now write numerical algorithms and code which can solve these problems in real time. That allowed us to fully exploit vehicles' flight capabilities. That allows me to compute any feasible trajectory. It allows me to use the vehicle at its full capacity. If it can escape a threat — suddenly something shows up, a bird let’s say, and I can do a very aggressive manner to escape it (let’s say this is physically possible) this algorithm will find it.

But if my algorithm is not that smart of an algorithm, then, even though the vehicle may be able to do it, the algorithm may not be able to compute that motion. You see what I mean? So having smart algorithms, which can solve these problems actually is an

enabling technology, because it uses the vehicle's full capabilities so that the vehicle can do its job the best. In some cases it can actually [only] do the job that way, otherwise it can't. And there are many examples for this actually.

Am I making sense?

[Convex vs. Non-convex Optimization - 10:25]

Khari: Yeah, that makes a lot of sense. Maybe this is too technical, but what is the difference between convex and non-convex optimization? I understand that convex is easy to solve, but why is it easier to solve?

Behçet: I see. That's a great question too. I have a mental picture, I wish I had an actual picture too but the mental picture is the following. Any optimization problem can be seen as follows: let's say I have a set, a box if you like for non-technical people, and I have a bunch of choices in this box. First of all, for any decision-making — think about if you are making a decision — you have a bunch of choices in front of you, some of these choices are simply infeasible, you can't execute them. It's just infeasible, let's say.

This is a silly example, but let me give one. Let's say I want to go across a canyon and there are three bridges. Which bridge should I take? One is longer, one is shorter, one has these properties...each has some of their own properties. Of course, one other approach is to try to fly. But it's infeasible for me, I'm not a bird. I don't have the actuation, I can't fly across. That's an infeasible choice for me. Maybe it's a choice for a bird.

[Laughter]

Now, among the feasible solutions that I can pick some are better than others. Setting up an optimization problem is defining these choices mathematically. And typically in our mathematical problems we have infinitely many solutions. Not only a finite number, but infinitely many. We optimize our solutions in these feasible sets and that's really

what makes the problems convex or non-convex. If these feasible sets have nice shapes, like convex shapes, then the problems become much more tractable.

Let me give you a sense of a convex shape. The interior of a circle is a convex shape. Why? If I take any two points in that circle and connect them with a line, the line is entirely in the circle. You can think of this? An ellipse has the same property or a rectangular shape has the same property. But the shape of a kidney, the classical example — you know, like a kidney bean?

Khari: Uh-huh.

Behçet: There are points you can find and connect them with a line and the line will go out of the shape. Those are non-convex shapes and you can make very, very complicated non-convex shapes. Those don't lend themselves well to numerical algorithms. Actually, in some cases, you may not be able to even find a feasible solution. You may not be able to even get into the set. It becomes really hard mathematically to do; but for convex shapes it's much easier. So that's the main geometric difference. I think that's easy for people to see.

We are optimizing over a set of choices, and if one set is convex it has this property that any two choices can be connected by a line and the line is entirely within the set. We always talk about choices that are infinitely many — not like human choices, there are those types of problems too — but they are inherently difficult.

[Laughter]

That’s why decision-making is difficult. And of course, you may have three choices then it's easy, but if the combination of them gets really complicated then it’s not easy. So that's the main difference, the shape of the feasible sets over which you are optimizing your problem.

Am I making sense?

[Markov Decision Process - 14:02]

Khari: Yeah, I think the shape example is pretty clear. That makes a lot of sense. It's a good way to think about it. So, now I'll just ask you about some of the different terms I saw reading up about this, and maybe you could kind of explain them and talk about how they apply to the work you've been doing.

So what is a Markov decision process?

Behçet: Oh, I hope I will do a decent job on this one. My colleagues, hopefully, will not shoot me down. "Oh, he explained it really badly." I feel a bit self-conscious.

[Laughter]

Think about the decision-making process, as I make decisions I change my state. For example, let's say I want to go from point A to B. I'm at a different point, I'm not at A but I'm at C, somewhere in the middle. Now, my decision is to go from C to B. B is my final target. Then I go to D, now my decision is to go from D to B. As I progress what I have done before coming to D doesn't matter. The point that matters is that I'm at D. How I got there doesn't matter. These kinds of decision-making problems where your future decision, your next decision if you like, only relies on your current state, not how you got there is typically named as "Markov decision process."

Again, this may be my limited view of that, but that's how I see it. And let me add more to this about how Markov decision processes work. Let's say, just to give a relevant example, maybe that's the best thing to do?

Khari: Mhmm.

Behçet: Let's say I can be happy or unhappy, and there are five levels of happiness. I'm grossly exaggerating this but, you know, when you go to the doctor they ask you, "How is the pain from one to ten." This kind of thing.

Let’s say I went to my psychologist and he asked me, “Are you depressed today?”

[I can respond,] “No, I'm happy or I'm not happy,” or whatnot.

Or they are running an experiment and I'm a test subject, and the experiment is on how much coffee I drink. I'm just making this up. I have five levels of happiness, and if they run a bunch of tests among people, they say, “Oh, if a person is on happiness level two and there are, let's say, four different levels of coffee drinking: dark roast, light roast, and...four levels, let’s say. Somehow, magically, they have an impact on our emotional state. If a person is at happiness level tw, if he drinks coffee at level three, what's the probability that you will stay at that happiness level or go to other happiness levels? Worse or better?

Khari: Ok.

Behçet: Think about this. If I'm at level two of happiness and I drink coffee at level three, I have a 10 percent probability I'll go down, 20 percent probability I’ll stay where I am, 20 percent I'll go to the next level, and 50 percent I'll go to the highest level of happiness. These kinds of models are Markov decision models. Basically, depending on your current state of happiness and the input, which is the coffee in this case, you're trying to estimate your next outcome.

Now, I can give these decisions over and over during the day, and, let’s say, I want to maximize my happiness during the day. I want to sum my happiness up along the way. I want to look at the overall happiness index during the day. In this case, a Markov decision problem is maximizing the sum of happiness that I obtained during the day, for example, by giving these decisions. And if I had a model that told me with what probability I would move to the better happiness [level] or worse [level], depending on how much coffee I get at a certain state, I can use that model and I can estimate, more or less, what I should be doing. If I had a good mathematical model. That's what we try to do.

Now, of course, this is a silly example that I give. Maybe I wouldn't do that in this case, but...actually, I don't know, maybe my happiness is somehow impacted by coffee. If somebody figured it out and they gave me the algorithm, based on the algorithm I can improve my happiness potentially…

[Laughter]

...which I highly suspect in this example, but there are engineering examples where this type of modeling becomes an effective tool to model real-world decision-making problems. And we call this kind of decision-making model Markov decision models.

Khari: Ok, is this a model that's used often in space flight or spacecraft building?

Behçet: Not in the spacecraft business. This is more for advanced research. We are trying to use them in different autonomy models, like, I know that people use these kinds of models for airplane collision avoidance, for air traffic. If you detect a collision what do you do? Do you fly up or down or left or right? People try to use those models because they want to predict how the other guy will act. Based upon that prediction, and given our current relative state, what should I do? What's the best action so that I avoid the probability of collision?

There are people who use these kinds of models. I don't know how much into real-world practice these methods get into, but since I do a lot of advanced research sometimes it's not my immediate concern to produce results that will fly tomorrow. I have that kind of research too but this is not that kind of research, I would say.

[Model Predictive Control - 19:43]

Khari: Ok, interesting. Another thing that I saw cropping up a lot was "model predictive control." What is that? Maybe we've kind of already touched on that a little bit.

Behçet: Oh yeah. Actually, this has a lot of strong connections with Markov decision processes too, but, basically, model predictive control is a relatively new control methodology. Again, it's something that I do very actively, and I also implement these things on real vehicles, including a real test rocket. I'm one of the first to do this, so I'm very passionate about this type of methodology.

What it does is it formulates planning and control problems for vehicles as optimization problems, and, like I mentioned, it's based on solving these optimization problems constantly as we execute the mission. So we both optimize the mission outcomes as well as we keep the vehicle in check. We say we keep the vehicle stable.

It's a new technique that combines control theory with optimization I would say. The benefit is it uses optimization to gain controls domain. That's also something that I do in my research. I believe it's a quite powerful framework for control theory.

[Lossless Convexification - 21:04]

Khari: OK. Final term I'll throw at you: what is "lossless convexification?" Which is a word I've never heard before.

[Laughter]

Behçet: People use these terms, I use it, quite a bit. Basically, what it is...I'll define this in terms of control theory. In some problems, as I mentioned, we take the control problem, set up an optimization problem for it, and it's typically non-convex. If we convert it into a convex problem, typically we have to give up something. Think about the kidney example. Do you remember the kidney example?

Khari: Yeah.

Behçet: Now, instead of optimizing our kidney example, you may try to put a circle inside the kidney. And I may say, “I'll only focus on this part of the kidney and I'll optimize over this subset of the kidney, not the whole kidney.” Maybe there's a better solution beyond this circle in the kidney, but I'm not going to find it. I'll just focus on this circle. That's how I constrain myself just to make the problem numerically tractable. This is what we call a lossy convexification — I convexified the problem, but I lost [solutions] maybe. I really don't know.

Lossless convexification is a way to relax the problem to a convex problem in a bigger problem. It's like putting a circle around the kidney, in this case, so that the kidney is encircled by the circle completely but tightly. Because now I’m optimizing over a big circle, if you imagine that, but this big circle has a bunch of solutions which are infeasible for me because they are not in the kidney. But in some cases, you can set up this big circle in such a way that, when you optimize over this big circle, the optimal solutions that you will find in the big circle are also a part of the kidney. So I'm optimizing over a big set, but I can guarantee in advance that all the optimal solutions are also a member of the smaller shape that is inside, which is non-convex. In those cases I still convexify the problem by searching for a solution over this big set, but I ensure that the optimal solutions obtained fall into the little set, which is non-convex. So in this case, I don't have a loss.

Khari: Ah...

Behçet: I am mathematically clever. I fool the problem in a way of speaking. In a good way, not in a bad way.

[Laughter]

That way I make sure the problem is numerically tractable while ensuring the solution of the original problem. Again, I'm oversimplifying this, but in that case what I accomplish is lossless convexification. I didn't give up anything. Because in this case particularly, one possibility was the optimal solution for the bigger set fell out of the kidney and it

became infeasible, [if that happens] then I lost completely and I cannot afford those numbers. But I can guarantee in advance with the way that I set it up that will never happen. And that's what he did for the rocket example and we actually flew that example onboard the test rocket. We flew this lossless convexification solution on a test rocket example in 2012 and 13. We showed that these things actually work and they are relevant to real-world examples.

Khari: The thing you flew, was that the Xombie rocket? I was reading about that.

Behçet: Yeah. It's a smaller rocket than SpaceX rockets, but it was quite a major leap at the time. you know. Actually the SpaceX engineers who designed their algorithms worked with me in the past, they were very familiar with these techniques and they also use convex optimization for the landing of their rockets. So, in that way these methods have influenced people's thinking, if nothing else.

[University of Washington Autonomous Controls Lab - 25:00]

Khari: Wow. cool. So you're a leader of the Autonomous Controls Lab at the University of Washington. Do you have any specific research you're doing there that you want to talk about?

Behçet: Sure, of course. That's my research lab. I'm a professor there and I established that lab. I wouldn't call myself….I mean, I'm the leader I guess because there's no other leader.

[Laughter]

Everybody transitionary, you know, they are students, they come and go. They are typically doing graduate level degrees. Some of the work I do: I still work on rocket control problems. I work on drones, controlling drones, and irplanes. These are all interesting problems that I work on. Spacecraft that's going in deep space or Earth orbit. Those are my application examples, but in terms of theories that I work on: optimal

control theory and robust control theory, those are my core mathematical tools. Numerical optimization, convex optimization, those are my skill sets that I try to utilize to solve these real-world control problems.

Swarms of vehicles, that's something that I work on actively. They can be spacecraft or ground vehicles. We also have a good testbed. I must say, I don't really have much to do with the hardware development and the development of the tests, but, in general, I have very good students who actually developed this lab. We have a bunch of drones, quadrotors, and we exercise our new algorithms on these drones.

Also we have ground vehicles too, omnidirectional ground rovers. We can exercise our algorithms in flight or on the ground, indoors and outdoors. We can do both types of experiments. Indoors we use motion capture cameras instead of GPS, because we don't have GPS reception — or reliable GPS reception. Outdoors, we use GPS to determine our state. So I think it's pretty exciting work that's going on. That's why I do it. I'm a biased observer.

[Laughter]

We have a website if students or people who are interested in this kind of work want to visit. We have a website, as well as a YouTube channel where you can see some movies of our drones. We are not like the YouTube channels of popular artists with several hundreds of followers. My youngest daughter says that I have a "lame" YouTube channel, but I don't think that's true.

[Laughter]

If you are technically minded it's quite interesting. Quite often we put out movies of demonstrations, of flight, and experiments on the YouTube channel. Also I have a website where you can see our papers, our work, and so on.

Khari: Yeah, and I'll include the link to those on the podcast website.

So, what percentage of your work would you say is theoretical in terms of math or more abstract algorithm kind of work versus practical, like building something and seeing if the algorithm then does what it's supposed to?

Behçet: Most is still on math and theory, especially in academia. Typically, I have 10 to 12 or 13 Ph.D. students at any given time...or 8 to 12 or 13. And most of them work on theory and math. A few of them are more focused on implementation and experimentation, but even those students spend quite a bit of their time in maths and developing algorithms.

So, the majority of the time we are working on theory and mathematics and the abstract parts of algorithm development. Especially these days because, due to COVID, we can't go to the lab that often. So, we are stuck, pretty much, working on the theory these days; but when we have the chance we experiment quite often in the lab, indoors. And once in a while we also go outdoors. We have an. outdoor flight arena that we have access to about a 40 minute drive from the campus. We go there once a while to test out our algorithms in flight.

Khari: Hmm, and I saw you got an NSF grant for autonomy research. I saw that on your website.

Behçet: Yeah.

Khari: Are there any details about that project that you'd want to share?

Behçet: Sure, this is with two other professors, one is from Stanford [Marco Parvone], the other is from Johns Hopkins University [Marin Kobilarov]. What we proposed is that we have a lot of experience from space autonomy. Very autonomous missions succeeded spectacularly, and Mars Science Lab is one of the examples where autonomy was essential — it's not something nice to have, without that you couldn't execute the mission.

So we want to use these experiences — there were a lot of lessons learned, especially in terms of control — and develop autonomous control algorithms based on optimization. We also thought that an optimization-based framework for control is the right framework to handle many of these problems. We want to merge the lessons learned from space applications with the optimization-based theory to generalize these ideas to more general autonomous vehicles and vehicle applications.

So that's the idea: take what we learned in the space business, generalize these ideas and concepts by using optimization-based control theory. Like merging these two for the next-generation autonomous missions. It doesn't have to be spacecraft, it can be self-driving cars, it can be drones, it can be underwater vehicles. Actually, one of our example applications is underwater vehicles. One of my collaborators is at Johns Hopkins and he's focused on underwater vehicles.

[Assured Autonomy - 31:15]

Khari: That's exciting. I'll ask you one final question and then we can wrap up. So we met at the CCC's workshop on assured autonomy. What would you say are generally the biggest challenges to assuring autonomous systems currently?

Behçet: Assurance typically implies verification and validation type of activity. Like you develop a lot of good ideas, how do you independently verify that they will work? Not necessarily in the way you think about it. "Oh, because I thought about it. It should work," is not enough. There must be independent ways of ensuring that these algorithms will work because they will encounter numerous scenarios, some of which you can anticipate, some of which you can't. Once you develop an autonomous system how do you say with confidence that it will work in these types of scenarios to this confidence level? In other cases that I did not anticipate, as long as some of these assumptions hold, it will survive. These are pretty strong statements.

Ultimately, if an asteroid comes and hits you when you're on the ground, you can't protect against that. This is a one in a trillion event maybe but, you know, there are certain things you cannot protect against. But quantifying what you protect against and what you can ensure that you will be able to do is, I think, a very difficult problem in autonomy.

What you are doing is ultimately...think about this, when you interact with software — I mean, most of us interact with software these days when we take an iPad or some sort of tablet and we start going around the Web and things like that. That's where we really interact with software quite a bit. There are other places we interact with software, but we don't even recognize. Now, in this case, like when you drive your car, a lot of things are done with software that you don't have any clue about. Neither do I by the way.

[Laughter]

Now this drive for more autonomy is bringing software algorithms together with hardware. People in the past wrote software that humans would interact with, and if I write the wrong direction or command in my laptop it won't harm anybody. But if I design an autonomous system like a vehicle, the things you will do can hurt people. If drones are flying around delivering goods and these guys don't have the right instructions there are a variety of anticipated and unanticipated scenarios where it can actually cause harm.

So this is really the difficulty: can you ensure guarantees such that people can use these devices with confidence. If something goes wrong how can you also trace back where things went wrong? And of course these are legal issues also — like who is to blame and so on? How do you form the infrastructure around it, both technically and legally? Those are secondary, but they are also important problems in terms of adopting these technologies into our lives.

I think there are different layers: first the technical layer and then the societal layer. Once we are comfortable with the technical level, then now we can go to the next level.

Those problems are technical, but also, in some cases, legal, and maybe there are even other problems that I don't know.

Khari: Yeah, I guess that's the future. Will we have self-driving cars if we can figure these things out?

Behçet: Yeah, how do you insure a self-driving car is probably a difficult problem.

[Laughter]

What will the premium be? Will they evaluate the algorithms on each car, for example. Like if I have a bad driving record, you know, it increases my insurance premiums...when I don't have a clean record. Now, how do you do that with self-driving cars is an interesting question.

[Laughter]

You know one company has a better record than the others, suddenly if you buy that car you will pay more. I mean, there are a lot of interesting things that are going on right now; so again, before all of this, there are technical problems that we have to address. And one of the biggest ones is verification.

Khari: Yeah, well, thanks for taking the time to speak to me today. I hope everything goes well with your research. Any final thoughts?

Behçet: First of all, thank you for this opportunity to participate in your podcast. I really appreciate that. My hope is that the public shows more interest in these technologies. I know that the people are scared of some of these technologies. They are anxious about them at least, because they are destructive and they will change the way that they function as a society. I think they are not things to be afraid of. They are things that we have to understand because, I believe, that's the next step in our technical progress. The question is how to integrate this properly into the society. I think a lot of minds have

to think about it, and I hope more young people get into this area. I sincerely hope that it will happen so that we can progress faster.

[Outro - 36:50]

Khari: That's it for this episode of the podcast. We'll be back soon with new episodes. Until then remember to like, subscribe, and rate us five stars wherever you get your podcast. Until next time, peace.