RUSS TEDRAKE: OK, so your project presentations we drew randomly last time. Phillip, unfortunately, got the short straw. He's presenting today. Don't feel too bad for him. It's because he's going to [INAUDIBLE] next week, to ICRA, right? So the plan is eight minute talks plus two minutes of questions and transitions between speakers.

You're free to use your own laptop. Hopefully it'll just plug in and work. You're also free to send me slides so that I can-- I'll put on my laptop, whichever is better for you. If you have movies or something, send them to me a few minutes before so I can make sure they play, but whatever is better for you.

I'd like to ask you to try to sort of-- in eight minutes-- I mean, it's actually really hard to give an eight minute talk, right? It's much easier to give an hour and a half lecture. You can just ramble on. But you should try to sort of efficiently cover the things that I-- you know, the most important things to say are roughly, of course, describing the problem you chose, and it's always good to say for a minute why it's important or it's interesting.

I'd like you to describe the technical approach you took-- you chose, and why you chose it. And then a quick results, I mean, as much as you can get in, and if you have any sort of interesting implementation details that came up, that reared their heads, that's always fun to say.

You're not going to be able to fit much into eight minutes. But tell us what you've done so far. And it's typically pretty fun. You guys are not going to believe how cool the projects you've all picked are. It's going to be a lot of fun. We did-- so we're in this room Tuesday and Thursday. I've asked them in the-- because there's a real chance that we might run a little bit over class, I asked them if we could try to keep the room for the next hour. They actually told me that I could yet. But hopefully we'll be able to stay here till 4:15 or something if we need to to finish.

And the report-- I said it before, but the report is-- is it working?

AUDIENCE: No, the same thing has--

RUSS TEDRAKE: Why don't you power it down. That's when it worked last time. Yeah. It's due May 21st, I said, right? That's a Thursday. So if you choose the double column-- so I like to think of them as ICRA format, for instance, would be perfect. That would be the robotics conference. But I'm not asking for a novel, just a quick presentation of what you did. And I think you should have the same points as in the presentation, but you'll have a little bit more room to tell about the implementation.

AUDIENCE: Is there a monetary penalty for extra pages?

RUSS TEDRAKE: No, not at all. That's right. No publishing costs, and you can use color figures as much as you'd like. Any other questions about the project?

AUDIENCE: [INAUDIBLE]

RUSS TEDRAKE: I'm just giving you a ballpark six to eight pages. If you come in at five, I'm not going to give you a huge deduction if you say everything you needed to say. And you're free to-- it at least means there's no publication cost for extra pages, I guess. And six to eight pages double column in ICRA format is more than six to eight pages the other way. So it's a rough guideline. I'm just telling you, you know, roughly--

AUDIENCE: So we can just reformat it [INAUDIBLE] six to eight pages?

RUSS TEDRAKE: That's what I'm trying to avoid. [LAUGHTER] Thank you. I'm just trying to give you a sense of the amount of content that should be in there. And you can format it as you would like.

OK, excellent, so last official lecture-- so I wanted to, as promised, end my part of the class, at least, on some success stories of how all this stuff works and has worked in the research world over the last 15 years. And so the ones I picked out to talk about were Emilio Frazzoli's helicopter motion planning, which-- so Emilio is upstairs in LIDS. Some of the really primal-- some of the first good work on learning in robots was by Chris Atkeson. And Stefan Schaal was his student at the time, who is now quite established, obviously, at USC. And then I'll tell you a little bit about our perching project depending on how much time we have left, OK?

So let me start off maybe with the lights on telling you some of the big ideas from Emilio's planning for the helicopters. This was actually Emilio's thesis work. And they've continued to use-- nobody here's from Emilio's group, are they? No. Anybody know Emilio? Yeah? He's on your committee, OK perfect. That's good. So this work already. OK, correct me if I say anything wrong.

OK, so the big challenge that Emilio wanted to address in his thesis was-- well, he does everything very, very rigorously and formally. So he's got many theoretical contributions in his thesis. But the experiment that he wanted to get working was we've got a helicopter, let's say an RC class helicopter. And I want to drive it from point A to point B through potentially a very cluttered environment where the obstacles are even moving in real time.

So he really wanted a real time lightweight planning algorithm that was as close to optimal as possible in a minimum time optimality sense but that could run on a-- you know, with 1 megabyte of memory or something on a computer that's on board the helicopter. And what he ended up doing was a beautiful combination of feedback control and randomized motion planning. And I want to tell you some of those details.

OK, so the first big idea in the-- if you talk to Emilio about doing motion planning and you say I've got some nonlinear system f of x, u, and I want to find a path from start to the finish, and I've got obstacles, whatever, the standard motion planning problem-- but these obstacles could be moving-- the first thing he would immediately say and he has said to me is never ever search over u's. That's just seems like a bad idea in general. The [INAUDIBLE] that we talked about, the naive implementation is pick a point at random and then try to find the closest point, and then find the u which takes you as close as possible to that. He says, if you're doing that, you're not thinking about feedback stabilization, for instance, and you're working potentially on a harder dynamical system than you need to be working with.

So he advocates first designing a class of feedback policies. Let's say it's-- they could be time varying or time invariant, right? And they have maybe some-- they're parameterized, for instance, by some desired point. And these feedback policies if-- ideally you generate them carefully and have some provable stability guarantees about them. And instead, then, of picking in some u, which could potentially be working on a system that's very unstable, open loop unstable, it might require tight integration, all these things. Instead, you should sample over the parameters of your feedback policy.

So, for instance-- I'll make it more specific in a second, but for instance, if I take a point here, I find my closest point on the map, and I want to grow towards it, instead of trying a bunch of u's, I say, what if I ran pi 1 with x desired of this? How close-- x desired equals, let's say, this point, how close would it get me? Or let's say I run-- or maybe pi 10 would have been the closest one. I'll pick the closest feedback controller that's parameterized that's going to try to get me close to that point.

OK, there's a lot of benefits to doing that. In a very real way, it's talking-- we talked about feedback motion planning as having advantages. You know, if I-- I could come up with a trajectory here in the open loop sense that if I went to stabilize, it was not stabilizable. This avoids that shortcoming by actually searching in the space of feedback laws. And it also has advantages, for instance, that the dynamical system you're simulating is probably stable. So you could take larger steps, integration steps. You don't have to integrate very carefully if you've got a stable system.

So Emilio has advocated this sort of trim and maneuver view of the world in motion planning, especially on the helicopters. So each of these feedback laws that he considers at the motion planning time are some carefully designed controller. They're of two types. One of them is a trim controller. A trim controller for a helicopter, for instance, would be the hover, or something that flies forward with a steady speed, something that banks with a steady speed. All of these things are trims where, in the relative frame of the helicopter, it's a fixed point.

All right, so-- and Emilio was very careful in his work to point out that you can make heavy use of symmetries and invariants in those dynamics. So, for instance, the horizontal position of the helicopter doesn't really matter. If I know how to go forward from x equals 0 in a trim condition, then I also know how to go forward from x equals 1 in a position, right? If I know how to bank at a certain speed, I can do that from any x location. The location of the helicopter doesn't affect that feedback policy, OK?

All right, so if you have a few-- what that allows you to do, essentially, is come up with a handful of trim controllers which do a lot of things. In fact, what I think he does in the end is he comes up with at around 10-- I think 25 was the number. I remember-- I brought his paper just in case.

Yeah, so he comes up with 25 trim trajectories, which in his case was level flight, no sideslip, with forward velocities tiled over, just to give you a sense here, 0, 1.25, 2.5, 5, and 10 meters per second and then angular heading rates that are sort of similar from negative 1, negative 0.5, 0, 0.5, and 1 radians per second. OK, so if he knows how to do those 25 things, every combination of those, then he's got a pretty darn good library of trim behaviors that he can use as possible controllers that he could evaluate on his system.

Now those are the steady state behaviors. And in order to transition between trims he does these maneuvers, which are finite time stabilized transitions between trim conditions. And those maneuvers are actually pretty cool. I'll show you. So some of them can be-- so, for instance, he learned a maneuver that was the transition from going forward this way to being in the opposite orientation going forward this way. And that maneuver-- I shouldn't say learned-- he solved using a model of the helicopter, right, with dercall or something like that, a trajectory that was actually a snap roll, I think. It kind of went up, and did this thing, and then came back down and went the other way. And I'll show you a video that. I had lunch with Emilio and get his videos.

OK, so how does he design the feedback controllers for the trims? He uses value iteration, right? So we're current and relevant, see? So because he's got actually a relatively low dimensional space by the time he exploits these feedback invariants and just worries about the coordinates that matter in these things, he's actually able to tile the space and run value iteration. And the advantage of that, of getting these trims, is that he not only gets the nominal controller, but he gets the cost to go from value iteration.

For the maneuvers, he uses some sort of back stepping optimization, and he feedback stabilizes. And he uses as an estimate of the cost to go on the maneuvers just the duration of the time. Everything's a minimum time problem. So he uses the duration of the trajectory as the cost to go of these maneuvers.

And now when you go back into the planning problem, the search goes like this. I pick a random point. I start now asking every-- each of the existing points in my graph-- let me give it an initial graph here, just to make it a little bit more interesting. I ask all of the points on my graph not if they're close in the Euclidean sense, but what's the cost to go, what's the value function of trying to go from here to here. Each of them will vote on the cost to go. And he takes-- he starts-- he considers-- he tries to go in order of the minimum cost to the maximum cost.

Once he decides to try-- this one has the lowest cost to go. He actually tries to expand that controller in the direction of here and checks whether it's actually feasible to running this pi 1 and with this x desired, whether it can feasibly get here, for instance, without-- if there was an obstacle right here, let's say, then it might be that the motion plan is not feasible to get from there to there. So he throws it out and takes the next lowest cost. And every time he expands a node, he'll pick the next lowest. He'll search in these, try all the policies until he finds one that's feasible and tries to grow to there.

The result, actually, when he puts it all together-- you should read the papers, actually. This is a-- it's hard to do justice in such a complicated piece of work. He's really-- he thinks of everything, it feels like, in his thesis. The five minute presentation here doesn't do it justice.

But he's got an algorithm that works fast enough that it works in real time even if the obstacles are moving. Because he's computed-- the big idea there is that you used value iteration to compute the cost to go for the non-obstacle case, right? You know how to control the helicopter when there's no obstacles. And you use that as your heuristic, your distance metric, for the case where you're planning with obstacles. And it works fast enough that he can be flying his helicopter around-- he's got simulations of like windows closing, and it goes a different way. Or the window's open, and he goes through this minimum time way. And a lot of times they still accomplish finding a minimum time trajectory from start to the goal, even with dynamic obstacles. It's pretty good, pretty good stuff. Yeah?

AUDIENCE: So when you decide to expand [INAUDIBLE] tree based on the value iteration result, and then you simulate it in [INAUDIBLE] came across an obstacle. So the values there, is that actually the q function, or is it the v function?

RUSS TEDRAKE: It's a value function, v function-- j.

AUDIENCE: OK, so if that actually happens to cross an obstacle, there might be another action getting [INAUDIBLE] from that space. And it wouldn't still have the lowest cost. But because the value iteration, you'd just say the main-- [INAUDIBLE] another state which has the lowest value.

RUSS TEDRAKE: It's true. So he's careful to say that when you go to this trims and maneuver-- whenever you're using feedback controllers in here, one of the things, you have to sacrifice, actually, not only optimality, but completeness, actually, is that you can only now be complete if the solution lies in this class of feedback policies. So you're sort of-- he calls it pi complete or something like this.

If there's a path from the start to the goal that is executed by these predesigned feedback-- parameterized feedback policies, then you'll find it. But you give up completeness. And potentially give up optimality.

Now, I would guess that if it was-- there are cases where he can try multiple things and actually get around the immediate problem you said. But I'm sure-- I would guess also that there's cases where he does say I've just missed that solution. So I'm sure there's some things where you could try-- for instance, maybe pi 1 would have said it's feasible, but there's an obstacle. But then maybe if I ran pi 10 it would actually get me there.

Let me see what he gave me. He gave-- so this is views from his small helicopter.

AUDIENCE: Does-- this state space has a velocity inside, or [INAUDIBLE]?

RUSS TEDRAKE: The state space absolutely has velocities. It's very much about learning a dynamic controller. Some of these are invariants. The obvious invariants are in position and translation. So the velocities maybe are not one of the invariant-- maybe they're supposedly reasoned about. But yes, it's very much a [INAUDIBLE] dynamic planner.

So here's the on board. This is sort of the success story. He's-- an on board video, and you'll see a few of the-- I think this is the longer version that we already had, because you'll see some flips and stuff in the middle. So this is, I think, the-- no he called it a hammerhead. I don't know all my helicopter stunts. This is the maneuver which changed directions pretty quickly, going back and around. So they're doing pretty aggressive things here with, given the model, very strong convergence guarantees.

OK, so especially in these vehicles-- they're doing it now with the forklift to do inserting into a pallet, and lift up pallets, and sort of moving around boxes and everything. These real-time motion planning strategies based on clever implementations of the RRTs, they're real. They work. And value iteration is sort of real, and it works.

I asked Emilio what he thought about the fact that value iteration doesn't solve the double integrator rate. And he was actually very interested. But it's sort of-- it's good enough. For the purposes of this, it wasn't-- it needn't be a perfectly optimal controller, just a coarse discretization, solve it pretty fast. That was good enough to give you a working controller with some stability guarantees to use in the rest of the planning algorithms.

AUDIENCE: [INAUDIBLE]

RUSS TEDRAKE: It's just dynamically replanning all the time.

AUDIENCE: Did he have like all the information [INAUDIBLE]?

RUSS TEDRAKE: So I think that the strongest results in the moving, I think, were in a simulation environment where it was fully known. I don't know if-- I don't honestly know if they had moving obstacles in their physical demos. That'd be cool.

OK, good, so evidence one that these things really work, yeah? I can't sort of talk about-- I can't do a whole class talking about learning control without mentioning Chris and Stefan who are really the guys that got a lot of people excited about these things. So let me show you some of their work.

Chris apparently also had some helicopter stunts that he did while he was at MIT. And I haven't seen those videos. But I hear they were pretty good too. So helicopters seem to be a popular thing.

So this is a case of learning sort of a cart pull problem. This is a pole balancing task with a big arm, a Sarcos arm. The arm has actually been sort of mapped out with inverse kinematics and inverse dynamics control. So really, he's moving the sled. You can think of it almost equivalently as a 2D cart-- and a pole that's not attached. It's not a pin joint.

There's vision coming off the side with-- that's why there's a bright pink ball and a bright yellow ball. So there's a vision system watching, doing the state estimation. And they did a lot of work. This was, really, some of the first people that were making real robots use these tools. They were using cue learning, value learning, and they call it model-based reinforcement learning. Because they were trying to distinguish between two versions of the algorithm.

One was do system identification on every trial. Between trials, solve your offline-- on that model, solve your reinforcement learning problem. And then put the best controller back on the robot versus doing the trials exactly on the robot. So this was their model-based case. And they argued heavily that in these robots it made a lot more sense to use a model, because even if the model was designed from a very few trajectories, they could learn a good enough model of the robot that the task would be completed in just a few trials.

OK, so eight real trials-- and that's the only data they had from the real robot. They had some initial guess at the parameters. But then the system identification really happened each time. They took the tennis ball off.

AUDIENCE: Is the range of motion of the arm limited, or is it just losing control when the pole falls?

RUSS TEDRAKE: I think it's losing control. I think they're trying to be safely in the middle of the workspace. OK, now the modeling, the system identification was only on the pole. The arm was well known, well characterized. So they just learned a handful of parameters for the pole. But they did it in a few trials. And then they showed that the policy could adjust that quickly. And they became strong advocates for model-based, doing it offline.

Now, I would say that in some of the problems we've talked about in class, for instance, the fluid stuff that Jon told you about, that's a clear case-- I think, we think-- that learning the model is not the shortest path to getting success. But on a robot arm, maybe it is.

They did that in a lot of different cool tasks. So they did, for instance, double sticking. They had a bunch of different tasks that worked with sort of similar--

AUDIENCE: Is it using vision?

RUSS TEDRAKE: The other ones were. I would guess, since there's no bright orange balls and there is a stick, that there's probably some sensor on the other side. But I don't know. I don't remember, actually.

AUDIENCE: It's got strings [INAUDIBLE].

RUSS TEDRAKE: Yeah, I don't know how they're doing the sensor.

AUDIENCE: I thought that was just to keep it from [INAUDIBLE].

RUSS TEDRAKE: From wiping out, yeah, from hitting the experimenter. OK, so this was really a cool time in robotics, right? So the people, they're coming in, and they're doing all-- making these robots do things they hadn't done. I'm sure cracking a whip or something was on the near horizon.

But interestingly-- so I have to tell you that after a bunch of experiments here and pushing it and pushing it, I think it's fair to say that Chris and Stefan decided that reinforcement learning by itself was not the route to super advanced robots. And they switched to working more heavily on imitation learning, which was still done sort of in a reinforcement learning context. But they tried to speed up learning by getting trials from humans.

So there's a couple of ways they did that. First of all, they'd try to learn the reward function from the human. Sometimes they'd try to learn the dynamic model from the human. Sometimes they would try to prime a value function from a human, from trials from a human. They'd play all these sort of-- at any place in the reinforcement learning framework where you could expect to try to use information from humans, these are the kind of games that people were playing and continue to play.

The new helicopter results by Pieter Abbeel and Andrew Ng are really a success story for imitation learning, because they used human pilots to drive those initial trajectories. So this is the pole balancing with a human.

AUDIENCE: [INAUDIBLE]

RUSS TEDRAKE: They're just watching the dynamics of the pole, but with a controller that works. And they can use that to sort of-- for instance, you could just find the policy that--

[LAUGHTER]

That's Chris.

This is Auk, who was just working with Stefan. They were having DB play tennis, and do drums, all these things.

[LAUGHTER]

Now, this particular example-- the other one obviously was not-- but this particular example was more about trajectory learning. And they could execute the trajectory on a fully actuated robot. But these are definitely the success stories in imitation learning.

AUDIENCE: He's no terminator, for instance.

[LAUGHTER]

RUSS TEDRAKE: Oh, you should see the new one.

[LAUGHTER]

Yeah, it doesn't have teeth. But it could crush through walls. You know, it's this huge hydraulic-- you've seen it, Rick. It's scary looking, right? And the videos they have, they're like--

AUDIENCE: They have one where it's learning to kick box.

[LAUGHTER]

The researcher was a kick boxer [INAUDIBLE].

RUSS TEDRAKE: So that's the one you've got to watch out for. They're made by Sarcos, all these big hydraulic-- I mean they're powerful tethered-- serious power output robots.

Yeah, so the same-- back to the non-imitation, but the learning. Jun Morimoto working with Kenji Doya and with Chris Atkeson did things like reinforcement learning to make this little robot stand up. It's kind of cute. I have to show it.

[LAUGHTER]

And then after a few trials it would successfully stand up, right? And these are really-- these are the cue learning kind of algorithms I told you about. So there's a lot of good work out there in that line of thinking.

Interestingly, I wanted to show, to find the videos. I couldn't find Jan's videos. But I told you Stefan and Chris had started going towards imitation learning and away from pure RL in some ways. But Stefan, I think, is back. I don't know about Chris. Chris is still on the fence maybe. But Stefan and Jan Peters are the ones that did this natural actor critic algorithm which I just wrote the reference on the board on Tuesday.

And they-- now there's a new wave of videos of a batting-- from a few years ago, you know it's playing tee ball or something. It's hitting a bat to try to hit a ball off. And they've got a-- Jan's got a wham arm doing a ball and cup task like this, and maybe paddle ball or these kind of tasks. And it's the next generation of these. But it's without the imitation learning. It's the natural actor critic. And they believe that these algorithms have gotten better enough that it's interesting again.

OK, so that-- I mean, that's actually-- I told you there's [INAUDIBLE] that run back and forth between pegs to optimize their gaits. There's a handful of stories from robotics of these learning algorithms working well. There's a lot of success stories of the motion planning algorithms working well. But they're sort of still not quite mainstream robotics, I'd say, the learning stuff. If you really wanted to design a controller, we'd still-- there's a handful of people out there that are still working on it. But I obviously believe that's one of the key ingredients going forward.

So let me take a minute and show you some of the stuff that we've been doing in a little bit more detail. What you guys saw yesterday still has my yesterday data, yeah?

OK, so in our lab, we're-- I told you this in snippets throughout. But just to show you really how we're going at this today in the research, I really believe that fluid dynamics is a great domain where there's tons of good control problems to be solved. And most of the problems that are unsolved are unsolved because we don't have good models that are useful for designing controllers.

So this is, I think, a particular domain where I think the machine learning algorithms and the model-free section of the course is really going to help out. You can do system identification, but you can also do reinforcement learning to take your best model-based controller and improve it in a fluid, for instance, setting where-- with just approximate models and model-free control synthesis.

There's actually other places. So Elena in our group is now-- she says she wants to do plasmas or something. So there are plenty of other domains where the dynamics are still just ridiculously poorly understood. And you can imagine if you wanted to do control of some high energy something or something where the models aren't good, there's really not a lot of good control work in those domains. You could do some of the first stuff.

So we've got two major fluid projects in the lab right now. One is the perching. I showed you the video before. I'll tell you the story now. And I'll tell you a little bit more about our robotic birds.

So perching is a hard fluids problem. Because if you go to a high angle of attack with your airfoil then the flow becomes very complicated. It becomes nonlinear and unsteady. And linear control and classical control works well and in these sort of low angle of attack regimes on the left. And that's what fighter jets use today. And when you go to the high angle of attack, the fighter jets aren't doing bad.

We actually-- Rick was just showing me today he found the instruction manual for doing a Pugachev's Cobra, which is this ridiculous task they do in airshows. And step one is like turn off your flight control systems, turn off all the warnings or something like that, right? [LAUGHTER] It's like, wow, OK. But then it's just a pretty rote maneuver. You go through a handful of trajectory-- controls. And it does this sort of nose up, nose back down. So it's hard to say that that's a high performance control system just yet. It's just sort of an loop thing the pilots do in air shows.

Birds all the time are doing beautifully complex dynamic maneuvers. My favorite now is when they land on a perch, because they go up to incredibly high angle of attack always in order to actually generate that complicated flow behind the wing to generate more drag. So when they get a-- when you get separation on the back of your wing, you get a pocket of low pressure. And you get a pressure drag, that they sort of go up, hit their wings out like this. It's like hitting the air brakes, and they stop dramatically faster.

So there's a handful of projects out there doing this. This is the Cornell morphing plane project. It's-- the idea in there is that you actually morph your body up to try to get a lot of drag. But you keep your wings morphed back down so that they have attached flow, and you can do your linear control on that. And then they move the tail out of the way so that the airflow on the tail is again attached. And they can do standard linear control.

And John Howe does these prop hang perching trajectories, where he uses a lot of thrust to stabilize his plane to go over and sort of do a helicopter kind of landing on the perch. But these are sort of not the approaches that we would advocate in the class. This is-- my take on that is that a lot of people are trying to perch by making planes that work with old school control.

And I'd like to advocate using our newer tools to do nonlinear control on more exciting vehicles, let's say. There's a couple of military vehicles that do these kind of things too.

So Woody has been-- and the group have been thinking about how to make a fair comparison between the performance of a bird and a plane. And they point out that the important variables here are the mass, the wing area, the density of the fluid. If you're willing to sort of characterize all these terms, then you can come up with a dimensionless quantity which scores how effectively a plane would land on a-- how quickly a plane, for instance, would decelerate or a bird would decelerate relative to what you'd expect if you were a flat plate and peak drag. That's a fair way to sort of back out the stopping abilities of a small bird versus a big plane or something like this. So it gives you a dimensionless number. Woody's not quite happy with us calling it the perching number. But I still call it the perching number.

It means you're impressive if-- if you stop well, it's impressive to be heavy, operate in a low density fluid, have a small wing area, or stop in a short distance. And now these numbers, you have to take with a grain of salt, because it's hard to get these numbers out of the literature. But our first crack at trying to take these numbers from papers and things like that-- a Boeing 747 not trying to stop quickly, this is trying to get you safely to your destination, is-- if you look at as it's decelerating, just before it touches down, it gets a number of 0.08. So it's roughly using-- it's a very small fraction of its available drag. Because it's taking a conservative approach.

Now the X31 was trying to stop fast. It's getting a perching number, again, very roughly estimated, of about 0.15 with sort of a 24 degree angle of attack landing. The Cornell perching plane's somewhere around 0.125. That number we sort of believe is accurate, because we know the guys. But they pointed out that they were more worried about, in their perching trajectory, not hitting the ground on the way to the building. So they optimized for something other than perching number, of course, too.

AUDIENCE: [INAUDIBLE] from biology?

RUSS TEDRAKE: What's that? Good. So the birds are more like this, yeah? Now, that's using a lot of thrust, which we can penalize if we know how to use metabolic-- measure their metabolic cost. And we're trying to get numbers of it without thrust. It's actually hard to get a bird to land without flapping its wings, right?

[LAUGHTER]

But we're actually working with biologists at the field station hopefully to get some real numbers like that. So what we'd love to have is sort of a plot of a length scale or something over perching number and show that it's invariant. You know, the small, small things that fly like planes get a bad perching number. And big things that fly like birds get a small-- or a high perching number and things like that. And we're working on that.

AUDIENCE: [INAUDIBLE] birds that naturally land without flapping.

RUSS TEDRAKE: Are there any birds? So the case, I think, where they land without flapping is when they wind hover. So it's not so much about the bird as about if there's enough wind over the perch, then they'll sometimes hover. And then we could try to find that kind of data.

And there are people that do-- so Rick had a picture of a parrot with-- a parakeet with an oxygen mask on trying to do energy metabolics on a bird. And, you know, everything's possible, but--

AUDIENCE: There is an s in the denominator. That basically means like that bigger planes [INAUDIBLE]?

RUSS TEDRAKE: Yes, so it's impressive if you stop quickly with a small wing. Right?

AUDIENCE: But there is a b zero and b f as well.

RUSS TEDRAKE: It's all factored out in order to be sort of-- so the initial case, actually, was actually m over rho s x, but that doesn't handle the case where you don't stop, go to zero speed. So you actually need this extra factor, which is also dimensionless in itself to handle the case where you go between some relative speeds. I think it's-- yep?

AUDIENCE: Distance, to me, [INAUDIBLE] usually things that we make are really bigger than birds [INAUDIBLE].

RUSS TEDRAKE: That's why we need to have a plot. And we found that we have a-- in our very sparse data collection, we have some inclination that there are some small planes that have a worse perching number than some big birds and that it's actually-- we have successfully taken out the dimensionality. But this is work that we're still trying to finish.

But just for fun, if you wanted your Boeing 747 to get a perching number of 11.7, it means you'd have to go from 450 mile an hour cruise speed to zero in 20 meters.

[LAUGHTER]

Right, so that's pretty good, right? OK, so why is it a good control problem? So this is all the reasons that we've used in class. But in these domains, the fluid dynamics I think are just prohibitively complicated. Their time varying nonlinear, CFD simulations are often very slow. We're in a very-- we're around 10 to the 5 Reynolds number where CFD craps out. And there's just not a lot of good design accessible models. Design accessible means something I could run any of my control synthesis tools on.

AUDIENCE: Can the wings of most planes handle the kind of drag that would be--

RUSS TEDRAKE: So UAVs, I think, can. But I think, yeah, I mean, no question that if we took-- even a fighter jet, probably, would rip its wings off. But a 747 could not physically stop in 20 meters without ripping its wings off, right? So I'm not advocating that we ride in planes with perching numbers of this. But UAVs, I think, could do this-- small UAVs.

OK, so even if we did have a perfect model, it's still a hard problem for all the reasons we've talked about in class. We have limited control authority. We have partial observability. We didn't spend a long time on that. But if the dynamics of-- the state space, the state of the fluid matters when the things are unsteady and nonlinear. And we don't have sensors for that. The control forces take time to develop. So it's under-actuated in that sense. There's intermittent control authority from the stalling on the wings. The control surfaces saturate. There's a lot of good reasons why it's still a hard control problem. You have to use the kind of tools we work with.

So we did this first by doing all these model-based approaches. We actually-- when we started the project, I didn't expect to have a good model of the glider. We thought this was immediately a case where model-free was going to be the solution. But Rick's initial data collection started coming out beautifully clean data. And we tried to learn an aerodynamic model and started doing model-based control. So in this case, it's actually maybe what Chris and Stefan would have called as model-based reinforcement learning, where you learn a model and then run the best policy.

It's in a motion capture environment, these planes. We-- Rick's built a lot of foam planes. Woody now has two. So we've got a small foam plane. It's got markers on here so that the cameras offboard can sense the position, orientation of the plane relative to the perch at about 119 Hertz. The only thing onboard is that servo motor you can just barely see in the middle, which is actuating the elevator in the tail and a battery and a receiver. That's it. Everything else is offboard control.

You see that the wings are bent up with some dihedral. That gives it passive role stability, just the center of pressure is above the center of mass. And it's got a big tail. So it's got sort of yaw stability. It's basically a dart. It would never do anything interesting out of the plane, which is the way they designed it. But it's got perfectly interesting and rich longitudinal dynamics.

And that allows us to simplify the model to be sort of a planar model, rigid body model like we've been using a class. In fact, in my notes when I get back to posting the most recent chapters, you'll see our simple rigid body model of the plane.

So Rick shot the plane like 240 times into the motion capture environment with-- we basically flew into the middle of the room, then kicked its elevator up, and then fell down, or kicked its elevator to a random place, fell down, and collected a lot of data. We built a careful rigid body vehicle model, a careful model the equator. We're making it better and better. Woody's got the next wave of system identification now.

And you plot that data over angle of attack, the lift and drag coefficients, which are the standard ways to do the system identification in aircraft. And the first thing you'll notice here is that it's data over very large range of angles of attack, which is kind of cool, because you don't get that in real flight data from real planes. Because pilots don't do that. It's the luxury of working with foam planes and motion capture.

And the other cool thing is, at least on our initial plane designs, we've actually broken it-- we built a plane we thought was going to be more ideal, and we now have worse models. But at least in these initial planes, we seem to have gotten very, very good match to some of the textbook flat plate models of quasi-steady flow.

Now, the drag coefficient looks a little bit messier. There's a-- it looks like noise there. But actually, we think that's not noise. We think if you watch a time trajectory of this, it's actually a periodic oscillation, which looks like the vortex shedding of the-- it has the same characteristic frequency as the vortices we see in the pictures I'll show you in a minute.

OK, so we have a model where the aerodynamics are fit from data. The state space is 7, roughly, sometimes 8, depending on how we model the actuator. But that's just sort of-- I told you when we talked about value iteration, I told you, ah, you could do it in 6 dimensions. If you're crazy, you could do it in 7.

So Rick's crazy, right? And he did value iteration on it first, which I think is the right thing to start with. To find a cost function, we said get to the perch as close as you possibly can and designed a nonlinear feedback policy over the state space that required discretizing the entire big state space very coarsely and then doing some optimized dynamic programming on the cluster and on all these things. And he came back with a policy which, when we play it out could nicely do these maneuvers, the post stall, and we say one out of five times it would catch the perch.

Now those specifications at the top there, it's entering motion capture at 6 meters per second. The perch is 3 1/2 meters away. Those are designed to try to be a trajectory that you could only acquire the perch if you're willing to go post stall. You could only slow down fast enough if you're post stall.

So these are the pictures. We actually now have a wind tunnel downstairs. Rick, can I show everybody your wind tunnel, Rick?

This is our wind tunnel downstairs.

AUDIENCE: [INAUDIBLE]

RUSS TEDRAKE: Box fans-- so you have to-- so my life changed when I met Jun Zhang who works at NYU. He's one of the best fluid dynamicists you'll ever meet, experimental fluid dynamicist. And you go to his lab. And he has box fans, and like he takes pictures with his digital camera. And he's got, like-- it's the most humble lab I've ever seen. And he ends up with fluid pictures that are on the cover of Science and you name it. There's a real art form to doing things, I don't know if minimally, on the cheap, something like that. But we were very much trying to do these sort of-- we're trying to channel Jun a little bit with box fans and you name it.

But we put the-- in that sort of chamber, we emit smoke. This is with still air, before we put the box fans on. You emit smoke from the leading edge of the plane. And you can watch the vortices pull off during a perching maneuver. And we count those vortices. And they're pretty close alignment to what we saw in the [INAUDIBLE] data.

This is what-- this is a simpler picture with a better fluid. This is actually moving the plane by hand. But it had the vortices with a lot less smoke. And we were trying to-- tried to capture-- we're trying to get quality like this in pictures like this now.

AUDIENCE: Could you just like burn something at the front of the wing?

RUSS TEDRAKE: It's titanium tetrachloride. It reacts with the vapor in the air and just-- yeah, and you don't want to breathe it too much, right? Rick says-- and you don't want to leave it open next to your wrench, right? It corroded the wrench like in an hour or something. It was pretty scary.

[LAUGHTER]

OK, so how are we doing so far in this dimensionless analysis? We've got these planes doing this much. And our glider is getting about a 0.55 in our current best estimates. Again, the pigeon's still whomping us. So we're working on it.

So nowadays we're actually trying to do LQR trees on it. So the model-based approach we're using is this, is exactly what I presented on the LQR trees. In fact, we started thinking about LQR trees because we were trying to figure out how to do a better-- the value iteration felt like it was wasting all of its resolution in parts of the state space that were completely irrelevant. And the LQR trees were a way to design trajectories exactly in the relevant parts of state space and nowhere else. so that's where that idea came from.

It's not working yet. We've got a-- our model-based LQR stabilization, Woody's best thing is hitting the perch about half the time now. But still, there's sort of this-- the model-based-- when the model's a little bit off the real system, we're still trying to figure out how to handle the differences between the model and the real system.

AUDIENCE: [INAUDIBLE] perching a real perch [INAUDIBLE]?

RUSS TEDRAKE: In the simulator, we can hit it from some small initial conditions. It depends on the model. In Ricks' flat plate model, it hits it from a beautiful number of initial conditions. In sort of the more careful system identification, the problem is harder, because the second derivatives are larger, it seems. So we're hitting it-- I mean, it still hits it from the nominal trajectory and hits it from a smaller range of initial conditions. But what we can't do, for instance, is take a simulated trajectory and put it into our model, which is not quite a feasible trajectory of the model. It's close, but we can't stabilize that.

We've got a big dynamical system with lots of interesting dynamics. We've got one actuator. And it's just-- it's hard to make it do things. So we're still trying to figure out if LQR trees are the best way for it. We'll see. And Phillip's trying to push LQR trees on the compass [INAUDIBLE]. And Michael's trying to do it on the dog.

OK, so now we're trying to do it with flapping wings too. So Rick's thesis is going to be perching with flapping wings. This is his simulation, which I think looks like a Dragon.

[LAUGHTER]

So were those actually the initial parameters of our bird, or did you pick them arbitrarily?

AUDIENCE: They're actually closer to what the glider would do if it was flapping.

RUSS TEDRAKE: Really? Yeah. So--

AUDIENCE: [INAUDIBLE]

RUSS TEDRAKE: The time-- oh, because it's a whole second.

AUDIENCE: A 1-second trajectory played over [INAUDIBLE].

AUDIENCE: Oh, so it's flapping [INAUDIBLE].

RUSS TEDRAKE: Yeah.

AUDIENCE: It has the same flapping frequency as the bird.

RUSS TEDRAKE: As the ornithopter, yeah. So this previous plot that I went past quickly is-- basically says that if you look at the-- if you do an LQR stabilization of a perching trajectory, given our original model, and you look at the cost to go matrix, and you look at the single-- I'm going to represent that cost to go matrix is the single costliest direction of the s matrix going backwards, yeah? That's a quantifier of how much error you should expect to get if you were tc get a gust of wind, let's say, in the worst possible direction during your perch. And it's a function of time, because the s evolves backwards.

So if you get a gust in the worst possible direction at your initial condition in the model stabilized, it can reject that nicely. But in the last tenths of a second, a gust of wind is going to be very hard to stabilize, because you have limited control authority, particularly because your airspeed is very small. And therefore your aerodynamic control authority is very small. And even if you put in thrust or thrust vectoring, it's still a big problem. If your airspeed goes to 0, your control authority goes to 0. And that's one of the reasons we think the flapping is such a good idea. That's one of the reasons we think that the birds always flap when they're landing, because they're trying to maintain air speed on the control surfaces. It's actually also a very good way to decelerate.

So this is the big bird upstairs in the lab. You're welcome to come see it any time, Zach, who you remember coming in with some robots at the beginning, he's the one who built it. He's an amazing designer. He's also a welder. This is a titanium welded gearbox in the front. It's a 300 watt outrunner motor. He's designed in all the failure points of the bird. So when it inevitably crashes, it's a breakaway beak. The wing spars have breakaway parts. So you bring bucket of parts out.

And this thing is-- it flew for the first time last summer. That's your TA throwing it for you. And it worked. And it worked. And then, oh my God, it's going to hit the building! So we turned it off, right? But that was our first successful flight with an autonomous ornithopter that was stabilized just in pitch.

All right, so I'll tell you quickly just why we care about birds here. And that'll be a fun note to end on. So basically, we care-- I care about birds not so much about flapping, but just about doing all the great things that birds can do because I think birds do fantastic things. But you have to be a little careful saying that. Because if you look at speed or efficiency sort of in still air steady level flight, then a propeller is very, very efficient. An airplane wing is very highly optimized. So you have to be very careful trying to say that birds are going to be more efficient or something than a plane unless you go to the more interesting flight regimes.

So birds are actually incredibly efficient-- this is in the notes. An albatross can fly for hours or days without flapping its wings just by-- even upwind-- just by exploiting gradients in the sheer layer, right? So when the wind is blowing, then there's a lot more beautiful things you can do than a plane isn't doing. So the cost of transport of this guy, the dimensionless quantity, it's 0.1, which is about the same as a 747 flying across the Atlantic, which is cool. But what's cooler is the fact that the cost of transport for this thing seems to be the same when it's sitting on the beach versus flying across the ocean. So it's basically flying across the ocean with almost no control energy. It just locks its wings in, uses the wind power, and off it goes.

Butterflies go thousands of kilometers carried by the wind. Falcons can dive really fast. The speed isn't the impressive thing, but the fact that they can do super agile maneuvers during these incredible dives, like catching a sparrow out of the air while it's diving is really impressive. Bats can catch prey on their wings. They've been studying down at Brown, Kenny Breuer's lab, they talk about how these bats can fly through thick rainforest with 1,000 other bats on their back, through caves with stalactites and stalagmites. And they're doing all these crazy things. They've got a video of this bat that can go-- he's going full speed this way. And in five laps, or about-- actually, he said that he does most of the turn in two flaps. And they say in five it's completely done. In basically two flaps, he's able to go full speed again the other way. And the whole maneuver takes just under half a wingspan-- full speed this way, full speed the other way in half a wingspan, wow.

And the story goes, and our mission here in my robot locomotion group upstairs is to try to make machines that can do this kind of stuff, birds that are far surpassing the performance of our best engineered systems in a lot of the performance and efficiency, acceleration, maneuverability, but not in steady level flight in still air, but in the interesting fluid cases. And the trick is that they're exploiting unsteady aerodynamics. They're doing control in places where we're not doing good control yet.

And it's a lot like, for the robotics community, I like to try to tell them it's a manipulation problem. If you're manipulating a piece of chalk or something, that's relatively easy. You can see the chalk. You know what your fingers are supposed to do. This is manipulating vortices that you can't see. You don't know what's coming. It's a much nicer manipulation, maybe the ultimate manipulation problem. And I like to say that once you start thinking of it as manipulating vortices and doing these sort of clever things, then suddenly you can see why fixed wings aren't as exciting anymore. It's almost like trying to pick up a coffee cup with flippers on or something like this, right?

So I don't actually care about flapping per se. And sometimes it matters. But I care about having a really delicate interaction with the fluid. And we're trying to figure out how to do that.

So here's the-- did I show this in the beginning? Yeah, good, so you've seen the dead fish. So that's another example of the efficient swimming. And it's the story of our pursuit of these controllers.

Awesome, OK, so I'm done. Now it's your turn. And we'll be-- let me know if you have any more questions about the projects. We're going to pass out evaluations next week too through-- we have an online system in EECS for evaluations. I hope you give us all the feedback. And I hope you enjoyed everything. And I'll see you next week.