PRESENTER 1: The basic concepts of reachability-- I'm talking about how we can represent reach sets in continuous spaces. [INAUDIBLE] is going to take it over and talk about some applications in robust motion planning. And then Gerard is going to talk about how we compute reach sets and focus on these two, on flow tubes and funnels.

So reachability is a task of figuring out what stage a dynamical system can possibly reach or be at at a certain time. So for example, if we have these two systems, we have a finite state machine on this reach system and a continuous system. And these are the starting points. And these are our axes. We can move east or north here. And we can move east or north with these velocities. [INAUDIBLE]

So after one second or one time step, we can be in either of these two states. Or we can be anywhere in this area-- and after two seconds, after three seconds, and so on. OK? So this is what reachability does. And the motivation for this-- one of the applications for reachability is for verification. Verification is just the task of making sure that we are never going to reach any bad states.

So for instance, let's say we have a microwave. A bad state is the microwave being open and on. We want to make sure that these bad states are never reachable. So what we do is we compute the reachable sets, and we check for intersection with the bad states to see if the microwave is on and open. Or like over here, let's say we have these obstacles. Let's say we have a wall or somewhere else over here. We want to-- this is bad, right? Because we can reach this state, and it's the same as a wall.

Any questions so far? Feel free to interrupt me any time. Another motivation for reachability is for robust motion planning, which [INAUDIBLE] talk more about. And I have a video of this. So what we can do with reachability is stuff like this. So this is done using flow tubes, which is reach sets, reachability. Gerard is going to talk more this example.

So now before we talk more about these cool examples, let's just formalize concepts. Let's

define some things. So let's go back to the same system that we had earlier, this state machine, with a finite set of states, X; a finite set of inputs, U, which in this case are east and north; and a transition function. And let's say this is just [INAUDIBLE] to keep things simple. And we have a set of initial states, X0, which I didn't label as red.

So the reach set is defined as the set of states that we can possibly be in at time t. So more formally, there exists a sequence of t control inputs that will take us from this initial state to the corresponding reach set. So let's say, for example, you have bad state, it's in the reach set at time 3 because there exists a sequence of three inputs. In this case this is one such sequence. There is two more that will take us from the initial state to that. OK?

And then the reachable set, the reach set reachable set is defined as the union of all reach sets for time less than or equal to t. So the reachable set of time 0 is just the initial set of states. And time 1, we have the unit and so on.

So the reachable set is what we use to test for intersection with bad states because that's all the states that we can possibly reach from time 0 to time something. OK? Now as some of you may have figured out by now, there's a simple way for finding segmenting so we can compute reach sets. So for example, let's say we wanted the reach set of time 2. One way-- the easiest way we can do this is to break this up into time steps, into single time steps.

So instead of computing it straight for time 2, let's first compute it for time 1. And now we can use these two states as the set of initial states and compute the reach set for one more time step. And that gave us this reachable-- the reach set. OK? Any questions?

OK, and the reason why these works is because reach sets are semi-groups. And semi-groups satisfy this equality. So we can not just break it up into single time sets. We can break into any two times, s and t.

OK, so all of this was for finite state machines. But for continuous systems, things get a little more complicated. It's a lot of the same concepts, but we need to handle a very important issue, and that is how to represent reach sets, or just set, regions in space for continuous systems. Because in a finite state machine, we can just enumerate the list of states in the reach set or reachable set. In a continuous system, we need some other way. We can't just say it's every point for which there exists a sequence of control inputs, because there can be infinitely many points.

So what we do, we have two types of representations that we like. One is convex polytopes. And that's just a generalization of a convex polygon into n space, into n dimensions. And we have two basic ways that we can represent these complex polygons. One is with these hull vertices, like this, AB to F. [INAUDIBLE]

And then the convex polytope is defined as the convex hull. And in case someone doesn't remember what a convex hull is, let's say we have just a list of points. The convex hull, imagine have a rubber band and you are going to stretch it out, and you let it go. And what that's going to do is this. And the convex hull is just the whole area inside this-- the rubber band. OK?

And then another way we can represent convex polytope is with a set of inequalities, so for example, here we have four lines. Here's the equation for each line. So let's say this red line, we want anything below the red line. This blue line, we want anything to the right, so anything greater than, and so on. So this area is the intersection of all these inequalities. OK? And each of these representations has an advantage over the other. Vertices are good to test for emptiness because when our set of vertices is empty, then our convex polytope is going to be empty.

And this inequalities is very useful for testing membership, because for any points, we can just plug it in to every inequality. And if it's true for every inequality, then it is in the whole group.

And convexity, I'll just explain it briefly. Convexity means that any two points inside our region-- our region is not non-convex is for the line connecting these two points. All the points in that line are going to be in the region. So here you can see that this is convex. And here, this red area is outside the region, so this other region is non-convex. And convexity is going to be important for the algorithms later on. Gerard will talk more about them. OK?

Other than convex polytopes, we can also use ellipsoids to represent reach sets. So ellipsoids are just the extension of ellipses to n spaces, n dimensions. And a convex polytope is defined by this. x is just all the points inside the polytopes, inside the ellipse. v is the center. And A just tells us about this, how deform the ellipsis. Any questions? OK.

And one of the reasons why we like convex polytopes, why we like ellipsis, ellipsoids, is because of their enclosure under linear operators. So let's say we have P that is convex polytope or an ellipsoid. Then, if we have a matrix A, then A times P will be defined that way. It is still going to be a convex polytope or an ellipsoid. So if we have something like this, then this

can maybe getting formed, but it's still going to be convex, a convex polytope.

And the reason why this is nice is that, if we have a linear system defined this way or this way, and then if we start with a-- basically, if we started with a convex set of states, then our reach sets are always going to be convex because we're just multiplying by A. And this is very nice to represent-- this tells us that our representation is going to be good because we can just use convex polytopes, ellipsoids, and stick with those. And then just a technical--

OK, so even though the reach sets are going to be convex, the reachable set might not be convex. But it will always be a union of convex polytopes or ellipsoids. For example, let's say, in the previous example, we started here. Then we need, after one time set, we were here. So these are the two things. And then our reachable set is the union of both of them. This is clearly not convex. But it is a union of convex polytopes.

So we can represent everything within it. OK? Any questions before I move on to [INAUDIBLE]? OK, thank you.

PRESENTER 2: OK, so now that have understood what reachability is and how [INAUDIBLE] represents, and before we dwell into more theoretical aspects of [INAUDIBLE]. Here are some cool applications of reachability. So there are many applications of reachability to robots are like complex systems [INAUDIBLE] systems.

One of the applications that has been talked about before is the robust motion planning. The early idea is to more of a glider or an airplane through lots of obstacles, even in the presence of [INAUDIBLE] these and [INAUDIBLE]. They're already crashing into obstacles, even when they are [INAUDIBLE].

[INAUDIBLE] applications of reachability to control complex systems like by parallel working robots. Here you can see that we are controlling [INAUDIBLE] simple soccer ball with something. And more importantly, reachability is used for very fine safety properties, for safety reasons, like aircraft collision avoidance. So in those [INAUDIBLE] we're going [INAUDIBLE], robust motion planning. If you're interested in other things, there are papers and there are [INAUDIBLE].

So the goal of motion planning is to find a set of control inputs that take you from the start state to the goal state. So in this example-- without colliding with any of the obstacles. So here, this is one possible path from the start state to the goal state. So it just looks nice.

[INAUDIBLE] collective things of the obstacles. But if you get the controls to this cart and then give it a real robot, you might find it going-- doing something like this.

So this is because of the path that you-- algorithm that you use for planning this path is-- that's not taking into account any of the uncertain areas in the environment or [INAUDIBLE]. So there are different kinds of uncertainties. One of them is environmental disturbances, like there is wind here. So that can move you over off your planned path. And there can be modeling errors. The model that you use to represent your complex system might just be an approximate. So the real world is different.

And there are state uncertainty. So the things that you used for measuring your position or velocity are also approximate, so there are uncertainties. [INAUDIBLE] there can be some randomness in the initial conditions. You are thinking of starting the robot at 0,0, but in practice, you might actually be starting at [INAUDIBLE] 0, [INAUDIBLE]. So there is randomness in initial conditions.

The goal of robust motion planning is to have some guarantees with some confidence that the system will reach the goal set without getting on with the obstacles even in presence of these uncertainties. So this is the problem. So let's look at some of the solutions that come up for solving robust motion [INAUDIBLE].

So this is a timeline. So there are three different systems. And all of them have a goal concept where they use some representation to represent the reach sets, the set of possible state that the system be. And then instead of searching for a single trajectory to a state, they search for these representations so that each final self-serviced representations that can go from the start state to the goal state, you won't hit any of the obstacles.

So earlier Bradley and Zhao and Frazzoli came up with this concept called the flow tubes. It's [INAUDIBLE] presenting makes sense. And we will know all about the [INAUDIBLE] of the [INAUDIBLE]. So one approach that we use is that we have a set of initial states and we have a set of goal states. So flow tube is a combination of all the possible paths that will get you from any point in the initial state and point in goal state.

So this is kind of representing all possible [INAUDIBLE]. So and then you search for these flow tubes to be a path to get your [INAUDIBLE]. And [INAUDIBLE] constraints so that you can also reason about complex systems and revise spatial and temporal coordinations like the walking robot that you have seen before.

An [INAUDIBLE] about for using funnels for motion planning. So funnels is also a similar concept to produce. It's kind of like a [INAUDIBLE] region around your path. So that if you are inside a funnel, you're guaranteed to be inside the funnel. So in this picture, we're going to [INAUDIBLE] motion planning using funnels. Again, there are references for the other people's [INAUDIBLE].

So instead, you'll be using funnels [INAUDIBLE] for every path, you compute some region around the path that is a stable region. And that's called a funnel. [INAUDIBLE] your funnels [INAUDIBLE], but for this part of the lecture, we are assuming we don't have a hundred of funnels, and that you'll see how to use these funnels to do motion planning. So the previous example is here. The [INAUDIBLE] funnel [INAUDIBLE] flied by. You see that it collides with the trees and [INAUDIBLE].

Then [INAUDIBLE] like these funnels are [INAUDIBLE] uncertainty in the world that you have. If [INAUDIBLE] everything with uncertainties, [INAUDIBLE] and then you compare funnels for that. Any questions so far?

AUDIENCE: Can you please say again definition of funnel?

PRESENTER 2: So it's kind of something that [INAUDIBLE] around [INAUDIBLE] around-- like you have a [INAUDIBLE] around there. So I think Gerard with explain more about it. So the [INAUDIBLE] with funnels is that you are inside the funnel at the point. When you [INAUDIBLE]. It's like [INAUDIBLE].

AUDIENCE: Yeah, you have an [INAUDIBLE]. So a funnel is basically-- it's kind of like a reach set around a certain point too. So the way that you do it is you have some [INAUDIBLE] trajectory. And then at each node point in that trajectory, you compute some kind of reachable set at that node point. And so you blow it up, and it becomes some kind of ellipsoid.

And as long as you're within that ellipsoid, no matter what disturbance you get, you're going to go back towards the nominal path. And then so, when you join all of those ellipsoids over the trajectory, then you get a certain kind of funnel. And as long as you're inside that funnel, you're going to stay on that funnel. And you're going to be on the path you want to be at with some disturbance in this [INAUDIBLE]. So that the great thing that there would be an example of a funnel. As long as you're in the funnel, you're going to stay in the funnel, basically.

AUDIENCE: So is this considered the control, or is it only the dynamics?

PRESENTER 1: It's the dynamics with the control. So it's the flow tube system.

PRESENTER 2: Yeah, so, you'll learn more of those [INAUDIBLE].

AUDIENCE: Let me just quickly add. So a funnel is type of flow tube, right? I mean, just different people either use the same terminology, use different terminology. Each of the applications of flow tubes has some invariant, which is different. And the particular invariant on this one is that if you are you in the funnel and you apply the LQR component, [INAUDIBLE] then you're guaranteed to stay within that tube.

AUDIENCE: But I thought the flow tube is actually like getting smaller and smaller in the funnel [INAUDIBLE].

AUDIENCE: Not necessarily. In general, a flow tube is simply a bundle of trajectories, which capture some common features. And then maybe that they move to a limit of cycle, and maybe they stabilize to a point, and maybe that the move to a goal, maybe that they always maintain. They're stable within a tube. In a funnel they focus particularly on stability to disturbance and then staying within the tube.

PRESENTER 2: OK, so the [INAUDIBLE] example is kind of useful to combine the reachability motion planning. So here is another situation where getting to the goal from the start [INAUDIBLE] and there are a couple of sort of trees lined up. But there is a path that goes in between the [INAUDIBLE].

This path looks kind of risky because it's pretty close to the two trees. But there is another path we can take which travel around the trees. And so if you don't need the [INAUDIBLE] safe. But maybe that's [INAUDIBLE] too. So that's why we do the combined reachability with motion planning. So if you combine that, you might find that the first path, if you compute a flow tube for the first path, [INAUDIBLE] it does not hit any of the trees.

And it's indeed safe. But it might happen that the second path is more susceptible to the disturbances in [INAUDIBLE]. So that's why [INAUDIBLE] and also [INAUDIBLE] combined motion planning. Because what's intuitive is not actual.

So far it's all about [INAUDIBLE] planning. So the [INAUDIBLE] everything about the environment, like where the obstacles are. So that's not always true. [INAUDIBLE]. So you

don't always know all the obstacles beforehand. You can't [INAUDIBLE] as you [INAUDIBLE]. So one problem with doing online planning, there is that you can not do any expensive computations during runtime. And all of these funnels can be places that are very expensive. [INAUDIBLE] solving an optimization problem. It takes a couple of hours to finish them.

So you can not compute funnels on the fly. So the idea here is that you compute the funnels offline and do all possible funnels from the start state to the end state, and then use this library to [INAUDIBLE]. So you have the same [INAUDIBLE] going from the start state [INAUDIBLE] the goal state in the archives.

So these are the set of possible paths from start state to goal state. And then from each possible trajectory, you compute the funnel. And the idea of online planning is to now find the sequence of these funnels that you can compose that takes you from the start state to end state without hitting any of the obstacles.

So an important component here is finding a sequential composition of funnels [INAUDIBLE].

AUDIENCE: So why plan-- you're doing a forward search with funnels. Why use funnels instead of something like RIT?

PRESENTER 2: I actually don't know what RITs are.

AUDIENCE: Can I guess at the answer?

AUDIENCE: [INAUDIBLE] thing.

AUDIENCE: Why use funnels instead of RITs.

AUDIENCE: Well, or why use funnels as opposed to anything else.

AUDIENCE: Well, I guess RITs, you have like the classical example, where you want to go through a very-- well, if you want to go through a very narrow region. Well, first of all, I guess RITs are going to have a hard time finding the narrow region because you'd need like a point to go through that. And then, RITs by themselves don't really account for errors, maybe some extension. But funnels, you can be sure that you're always going to stay inside the funnel.

So this if for [INAUDIBLE]. This is to make sure that we don't crash into anything. Does that answer your question?

AUDIENCE: Do you want to answer it?

AUDIENCE: Yeah, I think what he just said, it's-- the point of the funnels is that they provide a control policy. The actual reference trajectory that you have on the left is the motion planning. But with the funnels, you get a control policy. Because when you start executing a motion plan there's always going to be a disturbance. So you want a control policy that keeps you on that trajectory.

And there's all sorts of control policies. You could use simple EID controllers. They work in some cases. But funnels provide better guarantees for the power plants.

PRESENTER 2: Yeah, so the idea is that you have trajectory [INAUDIBLE] funnels [INAUDIBLE]. You're using this funnels compliance. You compose them to find your path to those presets that you wanted to go in.

So you can not always compose funnels. Some [INAUDIBLE] positions that are legal, and some of those are not legal. So [INAUDIBLE] example, a time going from left to right. So you have a situation where there is a funnel. You're trying to compose a bigger funnel with a smaller funnel. And you have another situation where you're going to compose a smaller funnel with a bigger funnel.

So what if I would have said, [INAUDIBLE]. So which of these combinations do you think is a legal composition? [INAUDIBLE]. Legal or illegal, I mean that which of these [INAUDIBLE] do not receive the [INAUDIBLE]?

AUDIENCE: The bottom [INAUDIBLE].

PRESENTER 2: The bottom one is--

AUDIENCE: --is actually marked because-- with the top one, if you're at the very top [INAUDIBLE].

PRESENTER 2: Yeah, that's true. So the first one is like illegal. And the second one is a legal composition. So why is that? So if you are inside this funnel, what this guarantees that you will stay inside the funnel. But it could happen that you can go through the funnel and make your outside [INAUDIBLE] funnel. And after this step, you can go anywhere. So you can [INAUDIBLE].

But if you're inside the second case, after you're through the first part, you're still set at the second part then we can decide [INAUDIBLE]. So now let's actually look at those online

planning algorithms. So this the case of your planning [INAUDIBLE]. First you can [INAUDIBLE] with quantum's chronicles. It makes our analysis easier.

So this is the algorithm that's going through each of the steps one-by-one. So this is online planning. You will always one information beforehand. [INAUDIBLE]. And you get to know more about it as we go. But you still have the initial planned funnel sequence that takes you from the start state to goal state.

And then you check for any new obstacles information when you go out from the sensor. So this is the region that you can sense some obstacles. And you also get the current state of your robot. This is also something concern [INAUDIBLE]. And you check if your current funnel path collides with any of the obstacles you have seen so far. And in this case, the answer is no.

So then you apply the control corresponding to this funnel for this [INAUDIBLE] location and time. That take your one [INAUDIBLE] problem. Then you put Goto and then do this again and again. So [INAUDIBLE] update from certain information. And one obstacle is here. You can differentiate the other one. When we check at the Replan collides with any of the obstacles-- in this case, yes. So since it collides with the obstacles, you need to [INAUDIBLE] funnels.

So how do you delete funnels? You have all this library funnels. So you go to each of them, and then you find one set of funnels that has-- you find a set of funnels that starts with your current location and [INAUDIBLE] of the obstacles. So this was a point, first of all, of a new funnel sequence.

And then, again, you apply controls corresponding to the new sequence and doing it two dimensional [INAUDIBLE], In the position, if it collides, and again collides, so you redirect [INAUDIBLE] apply the control for that funnel. And get a full solution. And then it collides, so you have to replan that part. Then you're OK. So now you reached the goal.

So those are the planning plan using funnel libraries. And questions about it?

AUDIENCE: Can you say more about the connotation of the funnel libraries? So they generate a set of funnels, which covered the complete set space?

PRESENTER 2: Yeah, so that's like-- there's every possible [INAUDIBLE] end point. And so then you can make [INAUDIBLE] or least, have it for every initial [INAUDIBLE] you can compose all these [INAUDIBLE]. So if you're doing it for every initial [INAUDIBLE]. So you might have a funnel

that exactly starts at your initial position. If there is a funnel that already started somewhere, where it goes through the funnel-- goes through your initial funnel, you can use that. So it's a truncation of your funnel. [INAUDIBLE]. Any other questions?

AUDIENCE: Why can't you use-- if it really is an online problem that you're not going to run out of a funnel, that you won't reach a dead end, will it be adaptable enough to make a U-turn at all times, either that there's come latency that would perfectly time it. And the robustness, that you'll always be able to find a path, even if we have guarantees that you'll always stay within a funnel.

PRESENTER 2: So are you asking what the situation, like the funnel, if you're going [INAUDIBLE] backtrack?

AUDIENCE: Yeah, just because these obstacles come out of nowhere, seemingly, that you don't have a guaranteed that the funnel is safe through the entire duration of our execution.

PRESENTER 2: Yeah, so that's a valid point. [INAUDIBLE] you can use the adaptive [INAUDIBLE] backtrack [INAUDIBLE] combined with the [INAUDIBLE].

AUDIENCE: Even if the robots can't actually backtrack?

PRESENTER 2: If the robots can what? Come back, or--

AUDIENCE: I mean, I guess, it's just like a UAB, it couldn't do that. I'm just saying, what guarantees do you need to have about the obstacles , since it's an online setup, right?

PRESENTER 2: Maybe you'd have some initial information about some kind of obstacles. You can find a [INAUDIBLE] trajectory-- you can find a trajectory that goes from the start sequence. It's like [INAUDIBLE].

AUDIENCE: Well, I'm just going to say. If you're in some kind of glider that can't necessarily-- like a quart of stop, go backwards, stuff like that, depending on how funnels you can plan backwards. But if you're in some kind of glider going straight at a wall, there's no guarantee. You're just screwed up. I mean, even humans, they make errors.

There's situations that humans can't dynamically get out of. So if your system is dynamically not going to be able to make a sharp enough turn or go straight into something, no matter how smart your planning algorithm is, no matter if you're like, there's a wall right there. Stop. If you can't stop, and you just go straight at it. There's obviously situations where you can pretty

much [INAUDIBLE] algorithm [INAUDIBLE]. You're just going to crash.

Hopefully you're in a situation where you're not [INAUDIBLE]. It's also based on how far away your sets are received. Your sensor can only see 5 meters ahead, and you're going 5 meters per second, [INAUDIBLE]. You're going to have make perfect decisions. But if your sensor has perfect information about the environment, then what seems like the initial best path may not actually be once you know the full [INAUDIBLE]. So it depends on the [INAUDIBLE] sensors.

PRESENTER 2: Any other questions? OK, so finally I have some demos showing simulations of [INAUDIBLE]. Let's see how [INAUDIBLE]. So the [INAUDIBLE] applications. So we have seen how to use funnels to do robust motion planning. We also have the offline planning and come to do online planning using a sequence funnels and the funnel libraries.

I will talk more about computing these funnels and flow tubes and other kind of reach sets.

PRESENTER 3: Yeah, so I'm going to talk about how to actually compute flow tubes and funnels generally. I'm going to give an example of what each one is used for. So we had the question before, how are flow tubes and funnels different? Flow tubes are-- so easiest way is they're both some kind of way to guarantee that you're going to-- the robustness guarantees.

So say you are here. You want to be here. They're both going to-- in most planning algorithms, you've got some kind of path from here to here. But say there's some disturbance [INAUDIBLE] down here or something. You don't know if you're going to be able to reach there or not. What a flow tube is is it's some kind of set of trajectories that will basically go from some initial state, which are here, to some goal states over here.

And basically, [INAUDIBLE] if you're within this flow tube, and you get pushed out on a different trajectory, you know how to get from here to here anyway. So you don't really care if you're pushed, as long as you're still within here.

A funnel-- let's see. So a funnel is, say you compute some nominal trajectory around here. Using the system theory and stability theory, you know that, with your control inputs, you're going to stay at each of these points. And you're going to stay within here within these circles. So if you get pushed out here, you know you're going to be able to get back onto your path when you're going to try to converge back to your nominal path. So you have some sort of funnel. Yes?

AUDIENCE: I have a question. How complete is the funnel? It sort of related to what Brian was saying.

PRESENTER 3: Completeness as in like these edges are--

AUDIENCE: Well, if you have a high dimensional state space, does the funnel cover it completely?

AUDIENCE: There's a set of funnels.

PRESENTER 3: A set of funnels--

AUDIENCE: So the question wasn't about if the individual funnel, but in the particular [INAUDIBLE]. When he generates a set of funnels, does the set of funnels completely cover the space?

PRESENTER 3: It depends how much time you want to put into it, generally. If you only compute two funnels, one of them goes over here and one of them goes here, then you're not you're not going to be covered in these states. So it depends on how much computation you want to do offline previously.

So you get a set of funnels that's here, another one that's right here, another one that's here. And you have your big library of funnels that cover the whole space.

AUDIENCE: I think I skimmed that paper. And, yeah, I think it is probabilistically [INAUDIBLE].

PRESENTER 3: OK, yeah. All right, yeah, but I mean, as far as computing where you're going to go, if you compute that path of it, you should be good. But yes, you're right. The probabilistically--

So here is what I explained. Normally you have your optimal trajectory. But if you get pushed off of it, you may not get back to the goal state. But if you have a set of trajectories that go from some initial states to your goal state, and you stay within that, you know how to get to your goal state, even if you get pushed.

Again, so some ways to approximate flow tubes, you can use some kind of polytope, which is as Gabriel explained previously, as long as it's a convex polytope, you can take that cross-sectional area and you can propagate it down here for the tube. You can also use ellipsoids or rectangles as cross-sections.

And important point is to know that they're intercross-sectional approximations. So as you see here, you may-- this would be your actually flow tube. But your approximation has to be an inter one, because that guarantees that you're with a flow tube. But it means you may also

miss some of the outer trajectories , which also, if you want to compute a more thorough approximation area you get better flow tube representation.

OK, so robust planning, you plan a trajectory from here to here, and that should be good. But then you get disturbed. You're still within the trajectory, so you're still good. But some work done by Professor Williams and Hoffman shows that-- so what do you do when you're actually outside of the flow? Now what you can do is temporal planning. So we're using the algorithm that we previously learned in class.

You can take your goal state and move the timeline back to a different time. So here, this flow tube is, if you're here in a certain amount of time, can you get to here? And if you get outside of it, then you can't do that anymore. But you can just use our algorithm and move back the goal time.

And then when you move the goal time, the whole funnel shifts because now you're going to be over here. And you don't care about getting here at this time. You can here at this time now. So then you move over. And now where you, you get back into it because you're now inside the flow tube again.

OK, so one example is a humanoid footstep planning. So this take a complex systems. You have a bunch of different parts of the system. You have your center of mass dynamics. You have your leg dynamics, foot dynamics. So what you're trying to do is plan your footsteps forward and through time.

So you have your original plan of where you want your footsteps to be for different parts. So there's discrete things that happen in the system, as we've previously seen in a lecture. So one thing that can happen is your left foot hits the ground, and then your right foot lifts off. You're left foot lifts off-- or your right foot hits the ground, and then your left foot lifts off. So those are different things that can happen.

So you have certain temporal constraints on the system that you want to [INAUDIBLE] by. And so as you start planning, you have flow tubes to get you from one step to another step. And that guarantees that you're going to stay within those. Even if you get disturbed and get pushed, your feet can still get from one step to the other. But then also, with the thing that we saw in the previous slide, you can move the timeline back and forth, as long as you're withing your temporal constraints that you had.

So those are flow tubes for the feet. And that would guarantee that your feet go where they want to. But you can decompose the system to also have flow tubes for the center of mass. So what the center of mass does is to be broken down into separate flow tubes. So does that make sense?

So this is the humanoid footstep planning that we saw previously. And now you're going to understand how the feet are planned to go to each of the blocks. [INAUDIBLE] go video.

AUDIENCE: I have a quick question. You said that you have a separate flow tube for the center of mass. But aren't they really just the same flow tube, like a high dimensional--

PRESENTER 3: Yes, but that's-- when you decompose this and then you [INAUDIBLE] flow tube rather than a full system high dimensionality.

AUDIENCE: Is it hard to recouple them? How do you know that they're consistent [INAUDIBLE]?

PRESENTER 3: Well, it still adheres to the dynamics of the system. So you can couple how you want, how you want it to move. And you want to move any basic constraints. But eventually, as you said, when you move your feet, your center of mass is going to move. When you step forward, you're center of mass is going to move.

[INAUDIBLE] analysis of it.

AUDIENCE: Can I make a comment about that? Yeah, there's a technique called feedback linearization that can be used to decompose a complex non-linear dynamics assuming to the set of linear ones. And that allows you to treat the center of mass as a [INAUDIBLE] separately from a control in the flow tube computations and while guaranteeing that [INAUDIBLE].

PRESENTER 3: So this is work done by Professor Williams experiment, which is actually [INAUDIBLE]. This is the robot doing the same kind of thing on Mars. This is walking on stones and planning its footsteps through a pretty complicated environment. The stones are not exactly in the path where you would normally want to walk. But moving them around, as long as you're within the flow tube, you'll get to wherever [INAUDIBLE] Awesome.

So that was how to compute flow tubes and-- oh, and another way you can also compute flow tubes is by learning. So you can have some kind of example, move through trajectories from one state to the other. And you can estimate a flow tube based on where the trajectories are, as long as the flow tube encompasses all the possible trajectories.

AUDIENCE: Is the dynamics always linear? Because I though if your polytope, and it would be like kind of, is it a complex proposal when you would use a non-linear dynamic, then you move a little bit. Then you would move a little bit, not being the convex shape then.

PRESENTER 3: I mean, you can plan non-linear systems too. But as far as I'm concerned, [INAUDIBLE].

AUDIENCE: So, yeah, the polytopes generally require linearization with the system. And there's a limit to the validity of the-- the range of the linearization. And that's concluded in-- that's one of the things that limits the size of the polytopes.

AUDIENCE: But it is the case-- I mean, you've just got to understand. The thing that you do for non-linear systems, which is a piece-- if you're making a piece-wise linear, but you are-- you are linearizing them at the w step. And there's other techniques that people have developed with flow tubes where you do require the input system to be linear. Right here it can be non-linear. But... it's an approximation.

PRESENTER 3: OK, so funnels, the goal is basically to find a region that guarantees you safety on a bounded uncertainty. So there are regions of finite time variance throughout all time during that trajectory that you've defined. And so in practice, there's a trade-off between computation time and guarantees.

So if your trajectory, if you compute more nodes, then you're going to be able to fill in these little gaps between each of the ellipsoid. Whereas, if you lessen them, it's going to take less time to compute each ellipsoid. But it's also going to not be as guaranteed. And especially important when planning is that, if you're planning in some kind of point cloud, then some of the points may be right here, where one ellipsoid doesn't intersect the other. And you might think that's a safe path, when actually it's not at all.

Yeah, so an example is this little glider, which is given in the paper by Anirudha and Russ. And so what you start doing is you create your system, your function. And this is dependent on your state, time, and w, which is some kind of bounded uncertainty. And you choose the bounded uncertainty, which means that you choose some kind of reasonable estimate of how much can happen to your system that you didn't account for.

So you start it. Your velocity can be different from what you want. Say your nominal velocity is 10 meters per second. You can have some kind of drag, or you can have some inconsistency in your motors. And so what you expect is that there's a plus or minus 0.5 meters per second

to your nominal.

So when you start running your equations, this is what you expect your robot to do. And then that can vary forward and backwards depending on what your actual was. But then you add some kind of wind on your system, which-- some kind of cross-sectional wind, [INAUDIBLE] direction, which we see there. And so what that ends up doing is pushing you system away from where you want it to actually be.

So now you have some kind of velocity uncertainty. And you also have wind that can push you one way or the other. So how do you deal with that? Well, you start with your nominal trajectory that has a plane flying perfectly straight. There's no uncertainty. Everything is deterministic. And it acts like you want it to.

And then, so right there the wind hits it. So what do you do? You have to have some kind of control, some kind of controller to bring you back to your nominal trajectory and try to control you around the system. So a standard way to do that is to have a cause function and create your optimal trajectory. And this cause function basically, it kind of mirrors a LQR controller.

And by solving a lot of these equations, which is a pretty standard process, and to find out how to do that, you get a controller that tried to bring you back to your nominal trajectory. And so you get back. You're happy.

But you don't have any guarantees just with the controller. You don't have any guarantees on how far you can get pushed. You don't have any guarantees on where you're going to go and where your system really can be pushed.

So when you compute the funnel, what you do is you choose a Lyapunov function. And I'll explain what that is in the next slide. And you try to minimize the space around that function that tried to bring it back. So what this is is-- so you have some kind of your initial state. This would be the initial state. And the whole region, what you're trying to do is minimize it around it. So does that makes sense why you do that?

So your trajectory is basically going to go anywhere around here. And what you're trying to do is find the region of space that minimizes-- that encompasses all of the places that your system is going to get to. So that's easy to do when you it out and you know exactly where your system's going to go. It's easy to just shrink your ball around it. But you actually compute that.

No, forget it. OK. So when you have that for each point in the trajectory, you know that you're going to stay within here, because you're going to have some kinds of-- from your nominal trajectory, you're going to have different places where your system can go under the given conditions of velocity and wind uncertainty. You're going to go anywhere withing that.

But as long as you know you're still in that ellipsoid, then you're going to stay within those ellipsoids as you go down the path. Does it make sense? Good. So the reason they use Lyapunov functions is there's different ways to measure stability. And so I'll move forward to this. So there's different ways to measure stability in a non-linear system. So in a linear system, you're either unstable or you're globally exponentially stable. So you're going to converge back to your-- to whatever stability equilibrium, what you have at an exponential rate.

But there is also, in non-linear systems, there's global exponential stability. There asymptotic stability. And there's stability in the sense of Lyapunov, which is what we care about. So the difference between that is-- so say this is your nominal. And globally exponentially stable means that no matter where you start anywhere on the board, you're going to be converging back to this equilibrium going at an exponential rate. So that's the best case scenario. That's what you want.

Asymptotic stability means that you're going to be going from somewhere on the board, within a certain region, back to this nominal trajectory. But it doesn't have to be an exponential rate. So you can do something like that, as long as eventually you sort of converging back to it. And then there's stability in the sense of Lyapunov, which is-- so, say you're here. Your system can be doing anything, anything crazy, as long as it stays within some certain thing, which in the 1D case we saw was the little ball going around it. So imagine there's like a ball for each of these. and this is where your trajectory could possibly go. As long as it stays within that, you can say your system is stable in the sense Lyapunov. Does that make sense? Cool.

So know this, does that--

AUDIENCE: I just have a basic question. So the Lyapunov function is around a stable set point. But the trajectory has non-zero velocity. So what is the point that it's stabilizing around?

PRESENTER 3: You can stabilize around a trajectory, which also what the funnels do. So we have your little point here. And this is your nominal trick, some kind of state. And you're wind can knock you

this way. Your velocity difference can knock you this way or this way. But as long as you're still within that, You've got to converge back to it.

And this is the value for your Lyapunov function. And these are the different states you can have. An important thing to notice is the derivative is a negative definite-- or negative semi-definite in the case. And that means that basically all your trajectories, no matter where you go, they're going to be going down this cone. And eventually they're going to to down to some value. So does that make sense? That's like pretty amazing stuff and your system stuff. Cool.

So once you know that, it makes sense to use Lyapunov functions as candidates for computing funnels.

So an ellipsoid is actually defined by a quadratic Lyapunov function, which is down here. [INAUDIBLE] shown the equation for an ellipsoid is some V times S. So that was the equation for an ellipsoid that Gabriel showed previously. The quadratic Lyapunov function has the same structure, where you have this being x to [INAUDIBLE], which is basically the error of you state. So it's some kind of exact mean of the state that you're actually in minus x0, which is the nominal point that you want to reach.

So this defines the region around your-- the nominal point, which would be somewhere in there. And the ellipsoid is computed by doing that minimization that I showed you on the previous slides. And then you can figure out the region inside of it by solving that equation. And if it's less than 1, I believe you inside. And those are all the red dots that show that you're inside of it.

So that's used when computing trajectories. You solve for-- say that you're inside a point cloud, you know which points you're going to get inside of this ellipsoid. So when you're planning, you solve that equation for each of the ellipsoids. And you know if you're going to hit something or not.

All right, so this is a video that [? Ani ?] did. And this is using the funnel libraries for collision avoidance. This is showing for just one funnel. The glider dynamics were shown in one of the previous slides. They were pre-set by the glider dynamics. But [INAUDIBLE].

So you see right there that there's objects, and the glider plans a path around them. So the funnels-- this is one possible one. Right here, I'm pretty sure they tell it where objects are, even though they're not actually there. And it finds that funnel being the one the guy wants.

And you see that throughout the entire path, the glider will stay within the funnel.

And then when you're searching with the online planning algorithm that was shown previously, it goes through all of these funnels and eventually finds the best safe one. You won't always find a safe one, as discussed. In this case you did. But it finds the best one around the path.

So one way that it finds the best one is by finding all of-- so if you're planning a point cloud, you find all the points that lie with the ellipsoids. And the one with the least amount of points is probably going to be the safest one. So here's a bunch of different obstacles. And it safely maneuvers all. And there's some pretty dynamic movements, which is cool too.

So within that there is-- let's see. I'm going to back to-- is there a way to get back to the full screen [INAUDIBLE]?

AUDIENCE: [INAUDIBLE]

PRESENTER 3: [INAUDIBLE]

So with the online planning algorithm, here's one of the possible paths. You have your point cloud of trees. And those were shown in the previous, the little round thing with the green foxy tree-looking things. Using a sensor model, you get a point of that. And as you're going, you plan your path. And the way that this works is every five meters, you plan a path. And then as you're flying through it, you get new sensor information. And you can replan your path using the algorithm.

And that will guarantee you that you're safe. And when you're flying, obviously you don't see the funnels. But right here you can see there's probably one funnel here and then another funnel that plans this way, another funnel that goes to here, and then another one that goes there. And that's a full path. And there's also a cool one , because it shows that you go through the tree. And that's where one of the funnels is rather than going around. It is possible that you start computing a new funnel-- or you start searching a new library here. And you may not be able to make it up the way in time, given what speed you're going at. Does anyone have any questions?

And this is the online planning algorithm that we saw before written out. And some of the references, the paper from the video that we just saw, that's given here. And that also go to the example [INAUDIBLE]. Frazzoli was one of the original people that computed a flow tube. Hoffman and Williams did the temporal planning for footsteps and decoupled humanoid

system. And I did most of the stuff on [INAUDIBLE] for that years ago.

Another important thing to note that was a big thing in my personal research was that even though you may note be able to compute-- or that you may not find a funnel that necessarily goes-- finds a safe path, you can shift that funnel around. You could even move it slightly up and down, as long as the first ellipsoid is still within-- as long as your initial state is still within better ellipsoid. And that's easy to do because you can just give it a simple transformation between one state and the other. And if you transform this S matrix, then that will transform your entire funnel and it will shift it around. So you can find your best funnel, even though it may hit something. If it was slightly shifted to the left, you can just shift it using that transformation. So that's one of the [INAUDIBLE] funnels too.

Does everyone understand funnels and flow tubes now?

AUDIENCE: So I have two questions. So the side of that funnel is computed by the transform script. Is that right?

PRESENTER 3: Yeah, if you want to get it some ridiculous wind, like 1,000 miles per hour, then you're going to get huge funnels. But that's under some kind of reasonable-- you know in this region of the forest or something there's only three mile per hour winds or something like that. Yeah, that's based a lot on what your counts are.

AUDIENCE: So do you have any algorithm that in your mind that is not using offline [INAUDIBLE]? So this is like mostly you do the offline and the you do the online search. But if you were in a wild environment, you do nothing. Then do you have any [INAUDIBLE] that isn't--

PRESENTER 3: I mean, there are algorithms that will-- well, just the basic controller will try to get you back to nominal trajectory. But there's nothing that I know of at least that will guarantee you safety and will give you an all-encompassing region like flow tube or a funnel. So flow tube you find there's an infinite number of paths that will get you from here to here with your system, basically every little minute change that you have. And that obviously takes awhile to compute and approximate.

And the funnels, it takes awhile to compute the regions where it can go. So I don't know anything that will online compute this region. But online you will be able to compute some trajectory. And then let your controller will take your state error and control it. But it won't guarantee [INAUDIBLE].

AUDIENCE: With the funnels, how do you compute your initial set of states, or the allowed initial set of states? You have these uncertainties, like for the wind and velocity, but what about the initial state before those disturbances happen? You'd probably want more than one initial state.

AUDIENCE: Here's and eraser.

PRESENTER 3: Yeah, that'd be good. Cool. Thanks. So with the funnels, so you have your nominal trajectory again, and then your ellipsoids around it. So when your system starts, you have your initial state. And your initial state is just-- it's going to be here at time t0. But over the time computed, you can go over here, or anywhere around here really, depending on what happens.

AUDIENCE: But I just me that you'd want to apply the funnel multiple times. And you're initial state may not be exactly the same. So it should be possible to use the same funnel even though the initial state is off by a little bit.

PRESENTER 3: Yeah, OK. So say that you chose this funnel first. And eventually what actually happens is you do this and you get pushed around. And then you end up here instead of this funnel. As long as your next funnel that you plan-- so you can start a new funnel at any point in this trajectory, as long as the next funnel is also-- as long as you have V intersection between the point and the next funnel. And if you're within this region, and you plan the next funnel, what do you do is you plan the next funnel with this being the original x0. So as long as that's you next x0, and then this will take on a new nominal trajectory.

AUDIENCE: Yeah, I'm just saying, asking how you, like for the one all the way on the left, that one, yeah, how do you define how big that initial region is?

PRESENTER 3: Oh, OK, so that one, say, if you start, say, not moving or something and you got pushed? Is that what you're asking for--

AUDIENCE: Well, I mean, I understand that you're using the V for uncertainty, and V and W. So that essentially defines how big those funnels are. Do the also define how big the first one is?

PRESENTER 3: Yes. Yeah, the do.

AUDIENCE: OK.

PRESENTER 3: So what you're doing, because what you want to do is use the funnels and have some funnel library and be able to use those throughout all time. So you don't want-- so I guess what

you're asking, the first funnel, you're going to start at some initial state 0 with no velocity, stuff like that. And so you're not going to have any uncertainty on the first thing because you know where you start, the initial state.

AUDIENCE: Yeah, I'm just asking independent of the wind and stuff like that, you might just be starting at a point other than what that nominal point is. So you may want to define that initial funnel size by some other criteria besides wind. Is there--

AUDIENCE: So [INAUDIBLE] sensor how sure you are about where initial location is.

AUDIENCE: Yeah, and then-- I mean, there's sensor uncertainty. But you just may happen to be in the-- if you're doing the glider experiment 10 times, you may not start it from exactly the same position every time.

PRESENTER 3: Yeah, you should-- your funnel, every time you plan a new funnel, you start at the nominal trajectory above that new funnel. So you're not going to have this funnel. And you're never going to start over here on the first one. You're always going to start at this point.

[INTERPOSING VOICES]

AUDIENCE: I though the point of the funnel was so that you can reuse it without having to replan.

PRESENTER 3: Yeah, you can move the funnels around.

AUDIENCE: Oh, you can move? OK.

PRESENTER 3: Yeah, yeah, that was-- so every time you start a new funnel, you start at this nominal trajectory. But the next time you use the same funnel, you could be over here and just shift the funnel down to exactly the point. So the initial state is never going to be anywhere starting at t0, it's not going to be somewhere other than the nominal trajectory. But at t1 it could be over here. And so you just shift the whole funnel around. And as long as your t0 starts at the new funnel, and as long as t0 is within the funnel above t-- last funnel, then you just shift it around. You can also turn it, as long as you can multiply this and this.

AUDIENCE: It seems like the shifting and turning is a pretty complex search problem in itself. Is that a hard problem?

PRESENTER 3: No, that was actually the-- [INAUDIBLE] I got rid of the slide. I might have gotten rid of that slide. I shouldn't because that was the work that I did for the [INAUDIBLE]. Let's see

[INAUDIBLE]. But, yeah, there's a bunch of different ways to do it. I did a pretty not very smart search, where I just looked for the funnels, the safest funnel being the one with the least number of points in the point cloud within that entire funnel. And then-- oh, man, I did get rid of this.

But basically what I did was find the safest funnel. And if there still was a collision within the safest funnel, the I admit I had a minimization problem as you shifted back and forth. And there was a cost function that-- see if I can bring it up.

AUDIENCE: So you're doing an optimization of the online running of the--

PRESENTER 3: Yeah. That was an input that was implemented in realtime simulation. But that wasn't implemented on the actual glider as of [INAUDIBLE]. So let's see, all right [INAUDIBLE]. And you also want-- I also made is so you minimize the amount that you shift it to. So you don't want to shift it completely somewhere else because it won't be in your path. And you'll end up doing some dynamic [INAUDIBLE]. Like, you'll be flying, and then if you compute a funnel that's here that's kind of safe, but you shift it all the way down to here. Then you're gliders going to be doing something crazy. And that may-- I guess it would be [INAUDIBLE]. But I minimized that for at least [INAUDIBLE] shift it. Does anybody have other questions?

Yeah, so does anybody have any questions on--

AUDIENCE: Can I ask one? So you've given us models of bounded uncertainty in your dynamics. So if you use probabilistic models, then your funnels start looking like risk allocations. Is there an interpretation of one in terms of the [INAUDIBLE]?

PRESENTER 3: I don't know. No, but I'm sure if you look it up, yeah. I'm sure somebody's been working on that. I'm not really sure that fits-- I know I haven't done that.

AUDIENCE: Talk a little bit about the interaction between the flow tube selection or the funnel selections and the underlying [INAUDIBLE]. So let's say you have something that's planning the reference trajectory that's separate from your flow tube selection or your funnel selection. And then you have to select the funnels and flow tubes that allow you to follow that trajectory the best you can. How do the interact if you have a flow tube-- or don't have a safe flow tube or funnel [INAUDIBLE] reference trajectory?

PRESENTER 3: So you're asking basically the difference between planning with a funnel laddering and

trajectory laddering.

AUDIENCE: Yeah, kind of sort of-- well, because I know the checkoff planner that uses an incremental pathfinder, so the planned path or something like that. And then you fit-- are you fitting flow tuber to the reference trajectory?

PRESENTER 3: Oh, you do that off and on. So you compute-- they're called funnel libraries. Or usually they're trajectory libraries. So ever computing a trajectory in realtime for a fast-moving dynamic system is going to be expensive. And then computing a funnel on top of that is going to be even more expensive. So what a lot of people do is they compute the library offline for a trajectory and plan using that. But on this one, you compute your trajectory, compute the funnel around that trajectory offline. And then you search through the funnel library. You don't search through trajectories and then try to fit a funnel around that one. You just search through the funnel storage.

PRESENTER 2: If you don't have a very good library, then you will not get a very good trajectory.

PRESENTER 3: Also, a cool thing about funnels versus trajectories is a lot of trajectories will be discretized for speed, so you have your nodes. And so, say this is what your trajectory does, and you have node points here, and you're trying to fly past a wall or something. If your wall goes here, you're algorithm is going to search these points, and it's going to say, OK, well, this is a safe path because none of the node point intersect with this wall. So you fly [INAUDIBLE] a little [INAUDIBLE].

But a funnel, since you have this region around it, you-- no, I did a bad job of drawing that, but OK. When we search the algorithm and see if it's less than or greater than 1, you're going to find all these points being in that region also. I guess that depends on how well you want to discretize it. If your trajectory is [INAUDIBLE] of really, really small points, then you're not going to run into that issue.

AUDIENCE: Seems like you could-- like when you're building a normal RIT without funnels, from concave points, if you say, yeah, they're illegal, it seem that if you're automatically adding on to that funnel instead of generating an entire trajectory and then making a funnel, you could save computational time for [INAUDIBLE] things like that. Do you know if that's done?

PRESENTER 3: No, I don't. I do know that, I mean, computing, you obviously want to minimize computation time. But it's not a critical thing when computing, I guess, funnels and flow tubes, because you

do that all offline. And you want to-- I don't think it's even close to the point where you go to online to compute a funnel. It takes like six hours to compute it, so-- depending on how many dimensions your state has. So, yeah, it's not like you're flying and you can compute one of these things. You'll crash like [INAUDIBLE]. OK.

[APPLAUSE]