GIORGIO METTA: So I'll be talking about my work for the past 11 years. So this has been, certainly, exciting, but also was long in duration, so we had to sort of stick to the goal. And I'll show you also a couple of things-- I mean, most of this work has been possible because we have a team of people that contributed to both the design of the robot and the research we're doing on the robot, so I'll be freely drawing from the work of these other people. I just cited them as the iCub team, because I couldn't list everybody there, but you'll see a picture later that shows how many people were actually involved in developing this robot.

So our, let's say, goal, although we didn't start it like this, is to build robots that can interact with people, and maybe one day be commercially-available to be deployed in the household. Everything we've done is-- on the design of the robot has to do with a platform capable of interacting with people in a natural way. And this is reflected in the shape of the robot, that it's humanoid. It's reflected in the type of skills we tried to implement in the robot. And overall on the design, the platform excels in terms of strength, in terms of sensors, and so forth.

There was an, let's say, hidden reason. We wanted to design a platform for research, also, so when we started, we didn't think of a specific application. Our idea was to have a robot as complicated as possible to give researchers the possibilities of doing whatever they liked.

So the robot can walk, has cameras, tactile sensors. It can manipulate objects. We put a lot of effort into the design of the hands. And it's complicated.

And it breaks often, so it's not necessarily the best platform, but it's the-- I believe, the only platform that can provide you with mobile manipulation, and at the same time with a sophisticated oculo motor system in the eyes and cameras. And maybe it doesn't give you lasers, so you have to do with their vision.

The result is this platform that's shown here. This started as a European project, so there was an initial funding that allowed for basically hiring people to design the mechanics and electronics of the robot. And unfortunately, the robot is not very cheap. I mean, the overall--

we tried to put the best components everywhere. And this is reflected in the cost, which doesn't help diffusion, to a certain extent.

In spite of this, we managed to, let's say, "sell," between quotes, because we don't make any profit out of it, 30 copies of the robot. There are still two of them to be delivered this year, so there are, at the moment, 28 around there. And four of them are in our lab, and are used daily by our researchers.

And given the complexity of the platform, we managed, at best, to build four robots per year. And at best means that we're always late in constructions. We're always late in fixing the robots. And that's because, I mean, we have a research lab trying also to do-- to have this, let's say, more commercial side or support side to the community of users, which, in fact, doesn't work. I mean, you cannot ask your PhD students to go and fix a robot somewhere in the world.

It was striking a bit that we managed to actually sell the robot in Japan. And that's because, you know, you see Japan as the place of humanoid robots. And having somebody asking a copy of our robot there was a bit strange. But nonetheless, the project is completely open-source. If you go to our website, you can download all the CAD files for the mechanics, for the electronics, all the schematics, and the entire software, from the lowest possible level up to whatever latest research has been developed by our students.

Why we think the robot is special? As I said, we wanted to have hands. And we put considerable effort into the design of the hands. There are nine motors driving each hand. And-- although, there are five fingers and 19 joints, which means some of the joints are coupled, so the actual dexterity of the hand is all to be demonstrated, but it works to a certain extent.

There are some sensors. It's entirely human-like. We don't have, for instance, let's say, we don't have lasers. We don't have ultrasound or other fancy sensors that, from engineering standpoint, could also be integrated. But we decided to stick to certain subset of possible sensors.

There's one thing that I think is quite unique. We managed along the way to run a project to design tactile sensors. And so I think it's one of the few robots that has almost complete body coverage with tactile sensors. There are about 4,000 sensing points in the latest version. And we hope to be able to use them.

I mean, you'll see certain things that we started developing. But for instance, we-- there was discussion about manipulation and the availability of tactile sensors. We just scratched the surface in that direction. We haven't been able to do much more than that.

As I said, we designed, also, the electronics. And the reason for doing this was that wanted to be able to program the very low-level of the controllers of the robot. This didn't pay off for many years, but at a certain point, we started doing torque control. And we started hacking also the low-level controllers of the brushless motors.

And so it paid off eventually, because that wouldn't have been possible without the ability to write low-level software. Not that many people are modifying that part of the software. It's open-source, also, that part, but it's very easy to burn your amplifiers if you don't do the right thing at that level.

And the other thing is that, as I said, the platform is reproducible. And at the moment there is GitHub repository-- well, a number of GitHub repositories which contain, whatever, it's some, a few millions of lines of code, whatever it means. It just means probably that a lot of students are just committed to the repositories, not necessarily that the software is super high-quality at this point.

There are a few modules that are well-maintained. And that's the low-level interfaces, which is something we do. Everything else can be in different ranges of readiness to be used and things.

Well, why humanoids? There were, at least at the beginning, scientific reasons. One, paraphrasing Rod Brook's paper, Elephant's Don't Play Chess, the reason of developing intelligence in a robot that has a human shape may give an intelligence that is also comparable to humans, but also provides for natural human-robot interaction. The fact the robot can move the eyes is very important, for instance, has a very simple face, but it's effective in communicating something to the people the robot is interacting with.

And also, building a humanoid of a small size-- the robot is only a meter tall-- was very challenging from the mechatronics point of view. So for us, engineers, was a lot of fun too-- the initial few years when we were designing, every day was very, very funny, our-- a lot of satisfaction seeing that the robot was growing and being built, eventually.

The fact that the platform is open-source I think is also important, allows for repeating experiments across different-- in different locations. So we can develop a piece of software and run exactly the same module somewhere else across the world. And this may, again, give advantages in-- first of all, debugging was a lot easier, so many people complaining when we do-- when we did something wrong, and allowed for also, let's say, shared development, so building partnerships with many people, mostly across Europe, because there was funding available, so for people to work together. And this may eventually enable better benchmarking and better quality of what we do.

As part of the project, we also develop middleware. So maybe you may think that we have been a bit crazy. We went from the mechanical design to the research on the robot, and passing through the software development, but actually, this was a middleware that was started before ROS even existed. And in fact, it was a piece of my work at MIT with a couple of the students there in 2001, 2002.

So the first version actually ran on COG and run on QNX, a real-time operating system. Later we did a major porting to Linux, and Windows, and MacOS, which-- so we never committed to a single version. And that because we had this community of developers from the very beginning, and there was no agreement on what development tool to use, and so we say, why don't we cover almost everything.

And this part of the software is actually very solid at the moment. This has been, you know, growing, not in size, but in quality, in this case, so the interfaces remain practically the same. And I think the low-level byte coding of the messages passing across the network didn't change since the COG time.

Everything else changed. It is completely new implementation now. But it has portability, so as I say, this was a sort of requirement from the researchers not to commit to anything, and so we have developers using Visual Studio on Windows or maybe using GCC on Windows, and other developers running whatever IDE available on Linux or MacOS. And this worked pretty well.

And there's also language portability. We can link-- so all this middleware is just a set of libraries, so we can link the libraries against any language. And so we have bindings for whatever, Java, Perl, MATLAB, and a bunch of other languages. And this helped researchers also to do some rapid prototyping maybe using Python and so forth.

As I said, the project is open-source, so you will find, if you go to the website, there's a manual, not particularly well taken care of. It works. At least, it works with our students, so it should work for everybody. But it also, the drawings-- so you can go with drawing like those to mechanical workshop. And you get the parts in return. And then from those, you can also figure out how to assemble the components.

Although it's not super-easy. It's not something you do, just because you have the drawings, you do in your basement. I mean, one of the groups in one of our projects tried doing that. And I think they stopped after building part of a arm and maybe part of a leg. I mean, it was very challenging for them. And you need a very, let's say, a proper workshop for building the components, so it takes time, anyway.

Continuing on the sensors, I mentioned that we have skin. And I'll show you a bit more about that in a moment. But we also have force-torque sensors, and gyroscopes, and accelerometers. So if you take all these pieces and you put them together, you can actually sense interaction forces with the environment.

And if you can sense interaction forces, you can make the robot compliant. And this has been an important development across the past few years that allowed the robot to move from position control to torque control. And this has been needed, again, to go in the direction of human-robot interaction.

And so these are standard force-torque sensors, although we designed, as usual. We spent some time and designed the sensors. And this was a reason of cost. The equivalent six-axial force-torque sensor, commercially, cost, I don't know, $5,000. And we managed to build it for $1,000. So it maybe is not as super rock-solid as the commercial component, but it works well.

And about the skin, this was a sensing modality that wasn't available. And again, we managed to get funding for actually running a project for three years to design the skin for the robot. And we thought it was a trivial problem, because at the beginning of the project, we already had the idea of using capacity sensing. And we actually had a prototype. And we say, oh, it's trivial. Then we spent three years to actually engineer it to make it work properly on the robot.

So the idea is trivial, so since capacity sensing is available for cellphones, we thought of moving that into a version that would work for the robot. There were two issues. First of all, the robot is not flat, so we can't just stick cell phones on the robot body to obtain tactile sensing.

So we had to make everything flexible so they can be conformed to the surface of the robot.

The other thing is that the cell phones only sense objects that are electrically-conductive. That's because the way the sensor is designed, so we had to change that, because the robot might be hitting objects that are not-- that are plastic, for instance.

So what we've done was to actually build the capacitors over two layers. There's an outer layer and a set of sensors that are etched on a flexible PCB that is shown there. And what the sensor measures is actually the deflection of the outer layer, which is conductive, towards the sensors. And in between, we have another flexible material.

And that's another part of the reason why it took so long. We started with materials like silicone that were very nice, but unfortunately, they degrade very quickly, so we ended up running sensors for a couple of months. And then all of sudden they started failing or changing their measurement properties. We didn't know why. We started investigating all possible materials until we found one that was actually working well.

The other solution we had to, basically, design was the shape of the flexible PCB. So we had the challenge of taking 4,000 sensors and bringing all the signals somewhere to the main CPU inside the robot. And, of course, you cannot just connect 4,000 wires.

So what we've done on the back side of the PCB there's actually a routing for all the sensors from one triangle to the next until you get to a digitizing unit. And-- sorry, each triangle digitize its own signals. And they travel in digital form from one triangle to the next until they reach a micro-controller that takes all these numbers and sends them to the main CPU. And this saves on the connection side, and so it actually enables the installation of the skin on the robot.

So this is a, let's say, industrialized version of the skin. And that's the customization we've done for a variant arm. And those are parts of the skin for the iCub. So the components that we just screw onto the outer body and to make the iCub sensitive.

This is another solution, which is, again, capacitive for the fingertips, simply because the triangle was too big, too large for the size of the iCub fingertips, but the principle is exactly the same. It was just more difficult to design these flexible materials, because they are just more complicated to fabricate on those small sizes. And the result, when you combine the force-torque sensors and the tactile sensors is something like this, which is a compliant controller on the iCub, where you can just push the robot around.

This is in zero-gravity modality. So you can just push the robot around and move it freely. And this has to be compared to the complete stiffness in case you do position control.

And another thing that is enabled by force control is teaching and demonstration. This is a trivial experiment. We just recorded trajectory and repeated exactly the same trajectory, so it's not-- I mean, you can do learning on top of that, but we haven't done it. It's just to show that the fact that you can control-- the robot in torque mode enables these type of tasks, so teaching a new trajectory that was never seen by the robot.

There's another less trivial thing you can do. Since we can sense external forces, you can do something like this, which is, we can build a controller where you keep the robot compliant. You impose certain constraints on the center of mass and the angular momentum, and keep the robot, basically, stable in that configuration like this one, in spite of external forces being, in this case, generated by a person. This is part of a project that is basically trying to make the iCub walk, more or less, efficiently.

And as part of the project, we actually also redesigned the ankles of the robot, because, initially, we didn't think of bipedal walking, and so they weren't strong enough to support the weight of the robot. And this is basically the same stuff that was shown on the previous videos, just the same combination of tactile and force-torque sensing used to estimate counter forces. We actually added two more force-torque sensors in the ankles, so we have six overall here in this version of the robot.

Now, as part of this, we also played a bit with machine learning. For mapping the tactile formation and force-torque sensor information to the joints, since they are not localized on the joints of the robot, we have-- and also for separating what we measure with the sensors from the forces generated by the movement of the robot by its internal dynamics, we have to have information about the robot dynamics. And this is something we can do, or we can build a model for using machine learning, since we have measurements from the joint position velocities and accelerations, and the torques measured from the force-torque sensors, we can compute the robot dynamics. And this can be done either using a let's say, computer model from the CAD, or from learning the model via machine learning.

And so we collect the data set from the iCub. In this case, it was a data set for the arm, for the first four joints. We didn't do anything for the rest. And in this case, we used-- we sort of customized a specific method, which has custom processes, to be incremental, and also to be computationally-bounded in time, so we wanted to avoid the explosion of the computational time due to the increase in the number of samples.

And this was-- well, it was basically an interesting piece of work because everything we do on the robot, if it's inserted in a control loop, has to have a predictable computation time, and possibly limited enough so that we can run the control loop at reasonable rates. And this is some of the results. And actually, we also compare with other existing methods. This is just to show that the method we developed, which uses an approximate kernel, works pretty much as well as a standard Gaussian process regression in this case, and works much better than other methods from the literature. This was just to have a rough idea that this was entirely doable.

Also, by shaping the kernel, it's possible to compensate for temperature drifts. Unfortunately, the force-torque sensors tend to change response due to temperature, not that the lab is changing temperature, but often, the electronics itself is heating up around the robot, so it's making the sensor read something different, and but it's possible to show that, again, through learning, you can build a compensation also for the temperature variations just by shaping the kernel to include a term that depends on time. This is one example of how we've done machine learning on the robot, although the problem is fairly simple.

A problem that is more complicated is learning about objects. So in this scenario, we-- targeting is shown here, where we have, basically, a person that can speak to the robot, tell the robot that it's a new object. And the robot's acquiring images. And we hope to be able to learn about objects from-- just from these type of images. This is maybe the most difficult situation. We can also lie objects on the table and just tell the robot to look at a specific object, and so forth. Again, the speech interface is nice, because you can basically also attach labels to the objects that are what it's seeing.

The methods we tried, in the recent past, we've done-- we basically applied sparse coding and then regularized least squares for classification. This was basically how we started a couple of years ago. And more recently, we used an off-the-shelf convolutional neural network. And again, the classifiers are linear classifiers. And this, I mean, has proved to work particularly well, but also, since we aren't the robot, we can, let's say, play tricks.

One trick that is easy to apply, and it's very effective, is actually, you're seeing an object, but you don't have a single frame. You can actually take subsequent frames, because the robot may be observing the objects for a few moments, for seconds, whatever. And in fact, there's an improvement that is shown in this plot there, the one to the right. If you increase the number of seconds you're allowed to observe the object, you improve, also, performance. And the plot is over the number of classes, because we also like to improve on the number of classes that a robot can actually recognize, and which was limited until, let's say, a couple of years ago, but now, with all these new deep-learning stuff, it seems to be improving quite a lot, and our experiments in that direction.

There's another thing that can be done, which is try to see what happens if we have-- since we have, again, the robot interacting with people for entire days, if we collect images on different days, and then we can play with different conditions on the testing case. So for instance, the different plots here show what happens if you train and test on the current day, so you train cumulatively on up to four days and you test on the last day only. And you see, of course, performance improve as you increase the train set.

Conditions may be slightly different from one day to the next. Light may have changed, just because it was more a sunny day or a cloudy day. And the other conditions are to test also on past days or to test on future days, so where conditions may have changed a lot. And in fact, performance is slightly worse in that situation.

OK and this is a video that shows, basically, the robot training and some of the experiment on testing how the robot perceives a number of objects. And unfortunately, there's no speech here, but this basically a person talking to the robot and telling the robot what is the name for this specific object, then putting another object there, drawing the robot's attention to the object, and then, again, telling the name. This is the Lego. It becomes faster in a moment.

OK, and then you can continue training basically like that. And the video shows also testing-- was showing a bunch of objects simultaneously to the robot. And here, we simply click on one of the objects to draw the robot's attention. And on the plot there, you see the probability that a given object is being recognized as the correct one.

OK, I think I have to cut this short, because I'm running out of time. Another thing I wanted to show you is, basically, now we have this ability to control the robot. We have the ability to recognize objects. We also have the ability to grasp objects.

And this is something that uses stereo vision. And in this case, what we wanted to do is to present an object to the robot, no prior knowledge about the shape of the object. We take a snapshot. We reconstruct a stereo pair. We have to construct the object in 3D.

And then we apply optimization, constrained optimization, to figure out a plausible location for the palm of the hand. And then that will maximize the ability to grasp the object by closing the finger around that particular position. This is our, let's say, definition of power grasp. So put the palm of the hand of the robot in a region of the object that has a surface, which has a similar shape or a similar size of the palm itself, and where the orientation is compatible with the local orientation of the surface.

And this works with mixed results. So it works with certain objects. It doesn't work always. There are objects that are intrinsically more difficult for this procedure, so some of them will only be grasped with 65% probability, which is not super-satisfactory.

If you run long experiments, you want to grasp three, four objects, you start seeing failures. It becomes boring to actually do the experiments. So it works well for soft objects, for instance, as expected.

We moved a bit into the direction of using the tactile sensors, and-- but at this point, we've only been able to try to characterize forces out of the force of the tactile sensor measurement. So we-- basically, taking a fingertip, we have 12 sensors, and we're trying to-- and this is another case where we apply machine learning trying to reconstruct the force direction and intensity from the tactile sensor measurements.

And this is basically the procedure, is we take the sensor. We move our six-axial force-torque sensor. We take the data. And we approximate this, again, with a Gaussian process.

Just one last video, if I can. OK, so basically, we've put together all these skills. We may be able to do something useful with the robot. In this case, the video shows a task where the robot is cleaning a table. And it's actually using the grasp component, and the ability to move the object, to see the object, recognize them, and grasp them, and put them at a given location, which was pre-specified, in this case, so it's not recognized that this a container. It's just putting things there.

And there's one last skill that I didn't have time to talk about, which is recognizing certain objects as tools, and one specific object, like this one. An object like the tool here can actually be used for pulling another object closer. And this is, again, something that can be done through learning. So we learn the size of the sticks or set the sticks, and we also learn how good they are for pulling something closer through experience, by, basically, trial and error over many trials.

And the result is that you can actually generate a movement that pulls the object closer so they can later be grasped. And that's basically a couple of ideas on how to exploit the object affordances, not just recognizing them, but also knowing that certain objects have certain extra functions which may end up being useful.

OK, I just wanted to acknowledge the people that are actually working on all this. I promised that I will do that. And this is actually a photo around Genoa showing the group that has been mainly working on the iCub project over-- let's say, this is the group last year, so there may be more people that just left, or some of them moved to MIT. OK, thank you.