



Robots: Crash Course Computer Science #37

Crash Course: Computer Science

<https://youtube.com/watch?v=3XkL0qQ21Oo>

<https://nerdfighteria.info/v/3XkL0qQ21Oo>

Hi, I'm Carrie Anne, and welcome to CrashCourse Computer Science!

Today we're going to talk about robots. The first image that jumps to your mind is probably a humanoid robot, like we usually see in shows and movies. Sometimes they're our friends and colleagues, but more often, they're sinister, apathetic, and battle hardened.

We also tend to think of robots as a technology of the future. But the reality is they're already here by the millions, and they are our workmates, helping us to do things harder, better, faster, and stronger.

===== Intro (0:30-0:37) =====

There are many definitions for robots, but in general, these are machines capable of carrying out a series of actions automatically, guided by computer control. How they look isn't part of the equation --- robots can be industrial arms that spray paint cars, drones that fly, snake-like medical robots that assist surgeons, as well as humanoid robotic assistants.

Although the term **robot** is sometimes applied to interactive virtual characters, it's more appropriate to call these **bots**, or even better, **agents**.

That's because the term "robot" carries a physical connotation: a machine that lives in and acts on the real world. The word "robot" was first used in a 1920 Czech play to denote artificial, humanoid characters. The word was derived from "robota", the Slavic-language word for a forced laborer, indicating peasants in compulsory service in feudal, nineteenth century Europe.

The play didn't go too much into technological details, but even a century later it's still a common portrayal: mass-produced, efficient, tireless creatures that look human-esque, but are emotionless, indifferent to self-preservation, and lack creativity.

The more general idea of self-operating machines goes back even further than the 1920s. Many ancient inventors created mechanical devices that performed functions automatically, like keeping the time and striking bells on the hour. There are plenty of examples of automated animal and humanoid figures that would perform dances, sing songs, and do other physical actions.

These non-electrical and certainly non-electric machines were called **automatons**. For instance, an early automaton created in 1739 by the Frenchman Jacques de Vaucanson was the *Canard Digerateur*, or Digesting Duck, a machine in the shape of a duck that appeared to eat grain and then defecate.

In 1739 Voltaire wrote, "Without the voice of Le Maure and Vaucanson's duck, you would have nothing to remind you of the glory of France." One of the most infamous examples was the Mechanical Turk, a chess-playing humanoid automaton.

After construction in 1770, it toured all over Europe, wowing audiences with its surprisingly chess playing. It appeared to be a mechanical, artificial intelligence. Unfortunately, it was a hoax - there was a dainty human stuffed inside the machine.

The first machines controlled by computers emerged in the late 1940s. These **Computer Numerical Control**, or CNC, machines could run programs that instructed a machine to perform a series of operations.

This level of control also enabled the creation of new manufactured

good, like milling a complex propeller design out of a block of aluminum - something that was difficult to do using standard machine tools and with tolerances too small to be done by hand.

CNC machines were a huge boon to industry, not just due to increased capability and precision, but also in terms of reducing labor costs by automating human jobs, a topic we'll revisit in a later episode.

The first commercial deployment was a programmable industrial robot called the Unimate, sold to General Motors in 1960 to lift hot pieces of metal from a die casting machine and stack them. This was the start of the robotics industry. Soon, robots were stacking pallets, welding parts, painting cars, and much more.

For simple motions, like a robotic gripper that moves back and forth on a track, a robot can be instructed to move to a particular position, and it'll keep moving in that direction until the desired position is reached, at which point it'll stop.

This behavior can be achieved through a simple control loop. First, sense the robot's position. Are we there yet? Nope. So keep moving. Now, sense position again. Are we there yet? No, so keep moving. Are we there yet? Yes! So we can stop moving, and also please be quiet.

Because we're trying to minimize the distance between the sensed position and the desired position, this control loop is, more specifically, a **negative feedback loop**.

A negative feedback control loop has three key pieces. There's a sensor that measures things in the real world, like water pressure, motor position, air temperature, or whatever you're trying to control. From this measurement, we calculate how far we are from where we want to be: the **error**.

The error is then interpreted by a controller, which decides how to instruct the system to minimize that error. Then, the system acts on the world through pumps, motors, heating elements, and other physical actuators. In tightly controlled environments, simple control loops like this work okay.

But in many real world applications, things are a tad more complicated. Imagine that our gripper is really heavy, and even when the control loop says to stop, momentum causes the gripper to overshoot the desired position. That would cause the control loop to take over again, this time backing the gripper up.

A badly tuned control loop might overshoot and overshoot and overshoot, and maybe even wobble forever. To make matters worse, in real world settings there are typically external and variable forces acting on a robot, like friction, wind, and items of different weight. To handle this gracefully, more sophisticated control logic is needed.

A widely used control loop feedback mechanism is a **proportional-integral-derivative controller**. That's a bit of a mouthful, so people call them PID controllers. These used to be mechanical devices, but now it's all done in software.

Let's imagine a robot that delivers coffee. Its goal is to travel between customers at two meters per second, which has been determined to be the ideal speed that's both safe and expedient.

Of course, the environment doesn't always cooperate. Sometimes there's wind and sometimes there's uphill and downhill and all sorts of things that affect the speed of the robot, so it's going to have to increase and decrease power to its motors to maintain the desired speed.



Robots: Crash Course Computer Science #37

Crash Course: Computer Science

<https://youtube.com/watch?v=3XkL0qQ21Oo>

<https://nerdfighteria.info/v/3XkL0qQ21Oo>

Using the robot's speed sensor, we can keep track of its actual speed and plot that alongside its desired speed. PID controllers calculate three values from this data.

First is the **proportional value**, which is the difference between the desired value and the actual value at the most recent instant in time or the present. This is what our simpler control loop used before. The bigger the gap between actual and desired, the harder you'll push towards your target. In other words, it's proportional control.

Next, the **integral value** is computed, which is the sum of the error over a window of time, like the last few seconds. This look back helps compensate for steady state errors, resulting from things like motoring up a long hill. If this value is large, it means proportional control is not enough, and we have to push harder still.

Finally, there's the **derivative value**, which is the rate of change between the desired and actual values. This helps to account for possible future error, and is sometimes called **anticipatory control**. For example, if you're screaming in towards your goal too fast, you'll need to ease up a little to prevent overshoot.

These three values are summed together, with different relative weights, to produce a controller output that's passed to the system.

PID controllers are everywhere, from the cruise control in your car, to the drones that automatically adjust their rotor speed to maintain level flight, as well as more exotic robots like this one that balances on a ball to move around. Advanced robots often require many control loops running in parallel, working together, managing everything from robot balance to limb position.

As we've discussed, control loops are responsible for getting robot attributes like location to desired values. So you may be wondering where these values come from. This is the responsibility of higher-level robot software, which plans and executes robot actions, like plotting a path around sensed obstacles, or breaking down physical tasks, like picking up a ball, into simple, sequential motions.

Using these techniques, robots have racked up some impressive achievements: they've been to the deepest depths of Earth's oceans and roved around on Mars for over a decade.

But interestingly, lots of problems that are trivial for many humans have turned out to be devilishly difficult for robots, like walking on two legs, opening a door, picking up objects without crushing them, putting on a t-shirt, or petting a dog. These are tasks you may be able to do without thinking, but a supercomputer-powered robot fails at spectacularly.

These sorts of tasks are all active areas of robotics research. Artificial intelligent techniques which we talked about a few episodes ago are perhaps the most promising avenue to overcome these challenges.

For example, Google has been running an experiment with a series of robotic arms that spend their days moving miscellaneous objects from one box to another, learning from trial and error. After thousands of hours of practice, the robots have cut their error rate in half.

Of course, unlike humans, robots can run 24 hours a day and practice with many arms at the same time. So it may just be a matter of time until they become adept at grasping things. But for the time being, toddlers can out-grasp them.

One of the biggest and most visible robots to break through in recent years has been self-driving autonomous cars. If you think about it, cars don't have too many system inputs. You can speed up

or slow down and you can steer left or right.

The tough part is sensing lanes, reading signs, and anticipating and navigating traffic, pedestrians, bicyclists, and a whole host of obstacles. In addition to being started with proximity sensors, these robotic vehicles heavily rely on computer vision algorithms, which we discussed in episode 35.

We're also seeing the emergence of very primitive androids, robots that look and act like humans. Arguably, we're not close on either of those goals as they tend to look pretty weird and act even weirder. At least we'll always have West World.

But anyway, these remain a tantalizing goal for roboticists that combine many computer science topics we've touched on over the last few episodes. Like artificial intelligence, computer vision, and natural language processing.

As for why humans are so fascinated by creating artificial embodiments of ourselves, you'll have to go to Crash Course Philosophy for that. And for the foreseeable future, realistic androids will continue to be the stuff of science fiction.

Militaries also have a great interest in robots. They're not only replaceable, but can surpass humans in attributes like strength, endurance, attention, and accuracy.

Bomb disposal robots and reconnaissance drones are fairly common today, but fully autonomous armed to the teeth robots are slowly appearing, like the Samsung SGR-A1 Sentry Gun, deployed by South Korea.

Robots with the intelligence and capabilities to take human lives are called lethal autonomous weapons. And they're widely considered a complex and thorny issue.

Without a doubt, these systems could save soldiers' lives by taking them off the battlefield and out of harms way. It might even discourage war altogether, though it's worth noting that people said the same thing about dynamite and nuclear weapons.

On the flip-side, we might be creating ruthlessly efficient killing machines that don't apply human judgment or compassion to complex situations. And the fog of war is about as complex and murky as they come.

These robots would be taking orders and executing them as efficiently as they can and sometimes human orders turn out to be really bad. This debate is going to continue for a long time and pundits on both sides will grow louder as robotic technology improves.

It's also an old debate. The danger is obvious to science fiction writer Isaac Asimov, who introduced a fictional Three Laws of Robotics in his 1942 short story, Runaround. And then, he later added a 0th rule.

In short it's a code of conduct on moral compass for robots, guiding them to do no harm, especially to humans. It's pretty inadequate for practical application and it leaves plenty of room for equivocation.

But still, Asimov's laws inspired a ton of science fiction and academic discussion. And today, there are whole conferences on robot ethics.

Importantly, Asimov constructed his fictional rules as a way to push back on robot as a menace means, common in fiction from his childhood. These were stories where robots went off the rails, harming or even destroying their creators in the process.



Robots: Crash Course Computer Science #37

Crash Course: Computer Science

<https://youtube.com/watch?v=3XkL0qQ21Oo>

<https://nerdfighteria.info/v/3XkL0qQ21Oo>

Asimov, on the other hand, envisioned robots as useful, reliable, and even lovable machines.

And it's this duality I want to leave you thinking about today. Like many of the technologies we've discussed throughout this series, there are benevolent and malicious uses.

Our job is to carefully reflect on computing's potential peril and wield our inventive talents to improve the state of the world. And robots are on of the most potent reminders of this potent responsibility.

I'll. See. You. Next. Week.

===== Credits (12:00) =====

Crash Course Computer Science is produced in association with PBS Digital Studios. At their channel, you can check out a playlist of shows like PhysicsGirl, Deep Look, and PBS SpaceTime.

This episode was filmed at the Chad and Stacey Emigholtz Studio in Indianapolis. And it was made with the help of all these nice people and our wonderful graphics team, Thought Cafe.

Thanks for watching, and try turning it off and then back on again!