# Trainable Robot Dog

Rob Berger and Paul Heltemes

December 15, 2019

## 1 Abstract

A robot dog is a robot with a doglike appearance or behavior. To build a robot dog for teaching children how to train a dog, the robot needs to be autonomous, yet influenced through a model of conditioning. This conditioning was based around the biological studies on a dog's cognition and conditioning literature. The development of the Robot Dog was programmed onto a LEGO EV3 robot with commands given through an Android app. The tests of the robot dog found that it could learn through the conditioning system and felt trainable. However, in order for the robot to feel truly doglike, the appearance and behaviors of the robot would have to be adjusted. Furthermore, the conditioning system should rely more on gestures, as the current training is mostly just giving vocal commands and telling the robot if it did the right action. The project brings to light how difficult it is to build a robot that is viewed as organic.

## 2 Introduction

This project is dedicated to the development of a robot dog in order to provide a way for children to practice training dogs before taking care of a living one. Robot dogs imitate dogs in either appearance, behavior or in both regards. For this robot dog to be trainable, a system of conditioning needs to be built, one with basis in biological dogs. The basic understanding of conditioning is that when a repeated unconditioned neutral stimulus, such as whistle, spoken line, or clap, is followed by a desired unconditioned response, the dog will start giving the same now-conditioned response every time the now-conditioned stimulus occurs. This is the fundamental way a dog is trained to perform tricks. Combining this conditioning system with a range of actions, voice commands through an Android app, and a reward-and-punishment system, a robot dog software that learns new commands with the affiliated action is created. Conditioning models from previously done research gave insight into how to develop the rate of growth and deterioration of conditioned responses. The commands and their respective responses were built as a list of probabilities as percentages for each of the possible tricks. These percent values would increase or decrease based on whether the dog received positive or negative feedback from the trainer. All of this code was run on a LEGO EV3 brick connected to various motors and sensors. In all, the robot dog consisted of a LEGO EV3 Robot, the code of the robot functions and the Android App for voice commands.

## 3 Related Works

One of the earliest examples of a robot dog was a guide dog robot for blind developed in the late 1970s and early 1980s ("Guide Dog Robot"). Although lacking distinctly dog features, this robot's ability to avoid obstacles leading the blind is a dog behavior (Tachi & Komoriya, 1984). In most modern instances, the fundamental components of a dog robot included interaction with the user and movement. One such modern example is CHiP the Robot Dog, a toy for children (WowWee, 2016). But, this toy has the ability to message phones and command via remote rather than training it like a normal dog. According to "Children's Cognitive and Behavioral Reactions to an Autonomous Versus Controlled Social Robot Dog," allowing the dogs to be fully autonomous grants the feeling of sentience and therefore helps promote the social-emotional development in children. Since the purpose of the robot of this project is to help kids learn how to train and interact with a dog, it is important for the robot dog to be autonomous but also be influenced through command. Therefore, a system of conditioning based off a biological dog's development through training is the centerpiece of the robot's decision making.

To understand conditioning, the book "The Wiley Blackwell Handbook of Operant and Classical Conditioning" laid out some of the basics of conditioning. In order for a dog to be conditioned, it must receive a neutral unconditioned stimulus, a sound or visual that triggers no response. Several factors determine the rate at which a dog becomes conditioned. In general, performing extra trials in a training session gives marginal returns (McSweeney, 2014). Furthermore, the time between training sessions also plays a role in the conditioning effectiveness. In respect to the project, this means the conditioning model should include a growth term that is affected by both time since the last training session and number of tricks done consecutively.

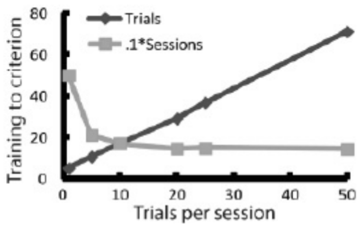

Figure 1

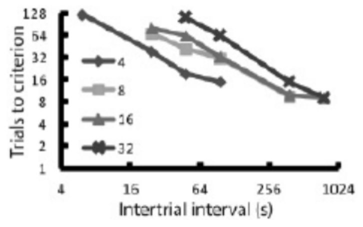

Figure 2

To implement a conditioning system in code, inspiration was necessary to see how it is translated to a mechanical process. The research paper on "Programming of a Pavlovian-like conditioning circuit" showed an extremely complex way to do it by using logic gates to create associations. It showcased that this truly biologically accurate conditioning system would be impractical for the purpose of the project (Zhang, Lin, Shi, . . . , & Ouyang, 2014). What matters is not that the robot's brain functions follows the biological process but that to the user it is perceived to be the biological process.

How dogs respond to visual and auditory cues was important to the construction of sight and listening of the robot dog. The article “Successful Application of Video-Projected Human Images for Signalling to Dogs” gave evidence that dogs learn tricks easier when the auditory command is paired with hand gestures, especially for the tricks “Fetch” and “Lay down” (Pongrácz, Miklósi, Dóka, & Csányi, 2003). Although dogs do not understand the meaning behind human speech, “Neural mechanisms for lexical processing in dogs” argues that they can interpret the tone (Andics et al., 2016). Therefore, they will inherently know if they are being praised or scolded by the intonation.

## 4 Experiments

### 4.1 Main Structure Implementation

The first step in this project was to decide on the very basic behavior for the robot dog. A general structure of wandering until the dog received a command was developed and prototyped, with the only tricks it knew being “stop” and “play dead.” It received a command through another thread waiting for command line input, while the main thread would simply randomly jolt in one direction for a few seconds, check in with the other thread, and repeat this simple process if no command was given. Though this implementation worked, it left much to be desired. First, it took too long for a trick to be registered and executed after a command was given. Though it was only a couple seconds at most, it would feel much better if it registered near-instantaneously, or at least took the same amount of time in every instance. Therefore, the structure was changed so that the secondary thread listening for commands gave an interrupt to the main program. The main program would be a loop of checking for and dealing with a potential command and then wandering indefinitely, and an interrupt would be handled at any time. When the interrupt was handled, it forced the thread to repeat the loop, at which point it would immediately see that it was given a command. This implementation also allowed the user to give another command while one was already being executed, as the command execution was within the block of code that could handle an interrupt.

## 4.2 Wandering with Potential Fields

The second problem with the initial prototype of the main structure was that the wandering behavior was too violent and jerky to be imitating a dog peacefully wandering around. A smoother solution was to use the concept of Potential Fields, which takes many objects the robot may want to go to or avoid and turns them into individual vectors. Then, all of these vectors are added up, resulting in one (hopefully ideal) vector in which the robot should move. In the case of the robot dog, four potential field vectors are combined. Three of them measure the distance in centimeters at certain angles using an ultrasonic sensor and return a vector at that angle, with a magnitude of $-\frac{100}{\text{distance}} + 1.75$. This way, closer objects return very strong forces in the opposite direction, while distances over about 57 centimeters have a very slight pull to them. The last potential field vector is simply a pull straight forward with a slight left or right angle, making the dog tend to move forward unless there is an obstacle right in front of it. Using this method, the robot dog successfully manages to avoid most obstacles, but still occasionally gets caught on skinny obstacles that were not seen at any of the three angles and low obstacles that are below the ultrasonic sensor. This could be solved using touch sensors, but the LEGO EV3 used for the robot only has so many input ports.

## 4.3 Speech Command App

While our initial prototype of the program featured command line input, this would not suffice for a final product intended for children. Therefore, the thread that once waited for input through the command line now sets up a socket server, which can be connected to through an Android app. All the user has to do is open the app, type in the IP address of the robot, and hit the “Say Command” button to give it voice commands. Assuming the phone and robot are on the same network, the app should use a socket to connect with the robot. Both the “Connect” and “Say Command” buttons change colors to intuitively let the user know the current status of their respective functions. It is intended to be as simple of a process as possible for the user.

## 4.4 Trick Implementation

The main tricks themselves are implemented as five functions: “wander,” “stay,” “rollover,” “come,” and “fetch.” “Wander” simply ignores the user and continues wandering, as a disobedient dog might do. “Stay” has the dog stop wandering for five seconds before continuing. “Rollover” uses the robot’s gyroscope to turn about 360 degrees. Finally, “come” and “fetch” are very similar commands, both relying on the PixyCam2 attached to the robot dog. This camera has the ability to find certain color signatures and return rectangles of all the signatures it finds. “Fetch” looks for a green ping pong ball, while “come” looks for an orange shoe. Both of these functions rotate the robot until a color signature is seen, at which point they move forward and slightly left or right, depending on where the PixyCam2 saw the object. “Fetch” stops running when the green ball is caught in the robot dog’s “mouth,” essentially a cage with a one-way door for the ball to enter through. “Come” stops when the aforementioned ultrasonic sensor reads in a close distance, presumably the user. Finally, there is an additional command that does not fit with the rest of them: “play dead”. This simply stops all motors and safely exits the program.

## 4.5 Conditioning System

The implementation of conditioning starts with probability lists of tricks. For example, each time the user says a command the robot has not previously heard, it has a 50% chance of performing the “wander” trick, a 20% chance of performing the “stay” trick, 15% chance of running “rollover,” 10% chance of “come here,” and a 5% chance of “fetch.” The robot randomly performs one of those tricks, and the user can provide feedback: “good boy” if it was the intended trick for that command, or “bad dog” if not. If it receives positive feedback, it increases the likelihood for performing that same trick the next time the user says that command, while the likelihood of all the other tricks goes down, making sure to keep the sum of probabilities for each trick at 100%. If it receives negative feedback, it does the opposite. The robot maintains a dictionary of these probability lists, with the keys being their respective commands. This process is shown below in pseudocode.

**Algorithm 1** Conditioning System

```
procedure takeCommand(Str $commandString$)
    if $commandString = "PlayDead"$ then return $"PlayDead"$
    if $commandString$ in $Commands$ then
        $lastAction \leftarrow$ runCommand($Commands[commandString]$)
    else
        $Commands[commandString] \leftarrow defaultProbabilities$
        $lastAction \leftarrow Commands[commandString]$
    $lastCommand \leftarrow commandString$
    UpdateGrowthValue
    return $tricksNames[lastAction]$
procedure runCommand(List $trickProbs$)
    $num \leftarrow random.random()$
    for $i$ in $size(trickProbs)$ do
        if $num < trickProbs[i]$ then return $i$
        else
            $num$ -= $trickProbs[i]$
procedure giveFeedback(bool $isGoodFeedback$)
    if $isGoodFeedback$ then
        $lastCommand[lastAction] += growth * (1 - lastCommand[lastAction])$
        for every other $trickProb$ in $lastCommand$ do
            $trickProb$ -= $growth * trickProb$
    else
        $lastCommand[lastAction]$ -= $growth * (lastCommand[lastAction])$
        for every other $trickProb$ in $lastCommand$ do
            $trickProb += growth * trickProb/(1 - lastCommand[lastAction])$
    $lastCommand \leftarrow None$
```

Additionally, the prior research indicated that having a nonverbal stimuli makes dogs more likely to do certain tricks (Pongrácz et al., 2003). Therefore, when a green object is seen by the PixyCam2, the program will roll the dice with its probability list three times, and if it outputs "fetch" one or more times, it executes fetch. This way, the presence of a green ball essentially makes the robot dog three times more likely to perform "fetch."

## 4.6 Growth Speed

The next step after the tricks was to enhance the conditioning system. Since the purpose of the robot is to imitate the training of a dog, it needed to at least somewhat imitate the specific models in the conditioning process. From the research in "The Wiley Blackwell Handbook of Operant and Classical Conditioning," the growth of conditioning could not be made completely linear. Specifically, as time increased between trials, it was more effective. Therefore, multiplying the growth constant by $e^{-\frac{1}{t}}$, where $t$ is the time in seconds since the last trick was done, would give the effect of decreased effectiveness of conditioning if there is no wait period between tricks being done. The graph of the growth value in terms of time in seconds is demonstrated in figure 3. Furthermore, the number of tricks done within a short time would also have lower the effectiveness of the training. Thus, the growth constant was also multiplied by $\frac{1}{\frac{x}{2}+1}$ where $x$ is the number of tricks done in last $t_s$ amount of time as shown in figure 4. In this case, $t_s$, the session time, was set to 5 minutes. The value of $x$ was the queue size of recent tricks. Each time a trick is performed, the time is added to this queue. Then, if those times were greater than $t_s$ away from the current time, they were removed.

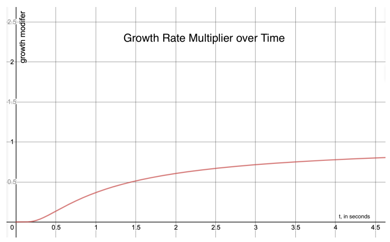

Figure 3

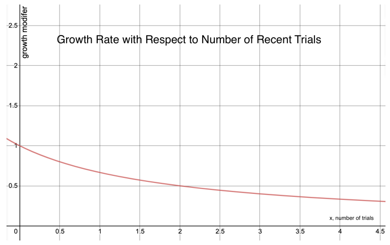

Figure 4

**Algorithm 2** Growth Formula

```
1: procedure UPDATE GROWTH VALUE
2:     $currentTime \leftarrow time.time()$.
3:     if $lastTrickTime \neq$ None then
4:         $t \leftarrow currentTime - lastTrickTime$
5:         $x \leftarrow getNumRecentTricks(trickTimesQueue, currentTime)$
6:         $growth \leftarrow defaultGrowth * \frac{1}{\frac{x}{2}+1} * e^{\frac{-1}{t}}$
7:     $lastTrickTime \leftarrow currentTime$
8:     $trickTimesQueue.append(currentTime)$
```

## 4.7 Extinction Implementation

Since dogs can forget progress made towards a trick (extinction in classical conditioning terms), a trick forgetting method was implemented to reduce conditioned probabilities if the robot dog is left alone for too long. Generally, it takes quite a long time for a dog to fully become untrained due to their good associate memory (Fugazza & Miklósi, 2014). But, for the purpose of making it feel somewhat relevant to the trainer in the short term, the extinction algorithm was designed so if the robot dog was ignored for 100 hours straight, it would lose all progress towards conditioning. However, for tricks that become fully trained, which is specified by the trick occurring at least 90 percent of the time, no deterioration would occur. Each time the program shuts down, it saves the current trick probabilities for every command in a text file, as well as the current time. Each time the program is started up, it reads in the values from the text file, and deteriorates them based on the current time minus whatever time was in the text file. Thus, the robot dog program is able to pick up where it left off, except some tricks will have been slightly forgotten.

**Algorithm 3** Extinction Formula

```
1: procedure Lower Conditioned Percents
2:     $hoursPassed \leftarrow currentTime - lastSessionTime/3600$
3:     for $Each\ Command$ do
4:         for $Each\ Trick\ in\ Command$ do
5:             if $defaultPercent < Trick.Percent < .9$ then
6:                 $Trick.Percent* = (1 - \frac{h}{100})$
7:                 if $Trick.Percent < defaultPercent$ then
8:                     $Trick.Percent \leftarrow defaultPercent$
9:         $Command.Wander \leftarrow 1 - sum(Command.Tricks)$
```

## 5 Results

The conditioning system worked as intended, with values being saved and updated as expected. However, sometimes it would feel either too slow or too fast to train the robot dog. Tweaking the parameters for growth rate, initial percents and decay rate would likely give better results, but it still feels a bit too robotic in responses. Although a real dog will respond fairly predictability to stimulus once fully trained, it still feels like its own choice, which the robot dog cannot achieve as an inorganic being. Therefore, it serves its purpose as a learning device, but falls short of behaving like a real dog. This is partially due to the appearance of the robot not really looking like a dog. Even though the robot played barking noises to seem more doglike, without having a much more complex LEGO design to resemble a dog or its mechanical movements, such as having actual legs rather than wheels, it is hard to think of the robot as a dog.

The regular movement of the dog could have been modeled more effectively. Dogs do not wander aimlessly and have some meaning in their paths. Furthermore, due to the removal of the touch sensors to fit the ball capturing cage, the robot dog had to rely only on the ultrasonic sensors to avoid collisions. So, the robot would often run into thin or low objects that went undetected by the ultrasonic sensor. Fortunately, these issues weren't very harmful to the overall effectiveness of the robot.

The voice commands worked and would transfer to robot in little computation and transmission time. However, one of the faults is that the system mishears a word it treats it as an entirely new command. But, dogs generally can tell the command due to its intonation. However, having to repeat oneself when training a dog is not unusual. Also, the misheard commands will also build a new list of trick probabilities, which over time will shift to the same trick as the original intended command as the robot receives feedback on whether it performed the desired trick for the intended command.

Perhaps the biggest problem with the robot dog training method is the lack of hints to give the robot dog. For example, when teaching a dog "rollover", the trainer could spin a treat around it so does the motion while telling a dog to "shake", the owner would grab the hand and shake it. Although hints were given through the ball color for fetch or the orange paper on the shoe for "come", it does not truly emulate how hints on how to perform a trick are given to a dog. Using the camera and other senses, a more complex way to show the robot dog what to do could be created with more time.

In all, the robot dog was successfully functioned properly but did not live up to truly real feeling dog. The robot dog still could be used a training device for children. The precise numbers of the model would like have to be tweaked over testing with children to see what works best. In addition, the inclusion of some other features of conditioning and more advance trick hints through complex gestures would make the training feel much more complete.

## 6 Conclusion

The final product of all this work is a robotic dog that wanders around until it receives a speech command from an Android app. The robot then randomly selects a trick to perform based on a probability list. This list starts out the same for each unheard command, and changes per command based on the feedback of the robot's performance. This feedback is much less effective if it's immediately following another command. When the user exits the program, all probability lists are saved to a text file. Finally, when the program opens back up, the probability lists are read back in, except regressed back to the default if it's been a while since the program was last opened

For a more in-depth conditioning system, could've implemented relapse for tricks. This means that if a trick was extinguished over time, then the dog would be able to re-learn it much faster because it had already learned it once. Additionally, it would've been useful to keep track of the last time each command was given, to track extinction separately from trick to trick. This would be much more realistic for a dog learning many tricks, rather than the current implementation, in which the robot dog recalls the probability list for every single command each time a training session is started at once, then saves them all, even if some commands weren't used during the session at all.

The main takeaway from this project is that it is very hard to code a robot that feels organic. The appearance of the robot is extremely important for the user to simulate a connection with the robot. This project mainly just focused on getting a classical conditioning system somewhat operational, and put minimal effort into making the robot feel anything like interacting with a real dog. Nonetheless, this project should suffice for giving children a starting point for how to train a dog before trying it with a real dog.

## References

Andics, A., Gábor, A., Gácsi, M., Faragó, T., Szabó, D., & Miklósi, Á. (2016). Neural mechanisms for lexical processing in dogs.

Fugazza, C., & Miklósi, Á. (2014, Mar 01). Deferred imitation and declarative memory in domestic dogs. *Animal Cognition*, *17*(2), 237–247.

McSweeney, F. K. (2014). *The wiley blackwell handbook of operant and classical conditioning.* Oxford: Blackwell Publ.

Pongrácz, P., Miklósi, Á., Dóka, A., & Csányi, V. (2003). Successful application of video-projected human images for signalling to dogs. *Ethology*, *109*(10), 809-821.

Tachi, S., & Komoriya, K. (1984). Guide dog robot. *Autonomous Mobile Robots: Control, Planning, and Architecture*, 360–367.

WowWee. (2016). *Meet chip: The world's first lovable robot dog.* Retrieved from https://www.youtube.com/watch?v=fSByydx1JdU.

Zhang, H., Lin, M., Shi, H., . . . , & Ouyang, Q. (2014). Programming a pavlovian-like conditioning circuit in escherichia coli.