2022

Twitch Plays Simulated Soft Robotics

Sidhanth Kafley
University of Vermont

Twitch Plays Simulated Soft Robotics

Sidhanth Kafley

skafley@uvm.edu

Supervisor: Josh Bongard

Thesis Committee members: Josh Bongard & Nick Cheney

Morphology, Evolution and Cognition Laboratory

Department of Computer Science

College of Engineering and Mathematical Science

University of Vermont

Burlington, VT 05405

UVM Honors College Senior Thesis

04/30/2022

Abstract

Scalability has been a long-standing open problem in the field of robotics and especially in the field of soft robotics where there has never been a technology that allows for crowdsourcing the creation and training of soft robots. The crowdsourced approach is very advantageous because it strives to maximize the use of available resources, cut expenses, diversify contributions, and promote transparency (Uhlmann et al., 2019). I posit that by using Voxcraft[1], Google Colab[2] and Twitch[3] we can not only create a way to crowdsource the creation of simulations of soft robots but also do it in an accessible, replicable, and scalable manner. I not only demonstrate that this can be done for the first time ever but also demonstrate that soft robots can be collectively created using this approach. The crowdsourced, low barrier to entry, replicability and scalability of this approach suggests that Twitch Plays Simulated Soft Robotics (TPSS) can be replicated and built upon in the future to not only get other user inputs such as fitness function or mutation rate to train robots but also scaled up to possibly crowdsource the crowdsourcing of soft robotics.

1. Introduction

1.1 The Scalability Problem

The scalability problem is a long-standing open problem in the field of robotics. It takes a while for one person to set up a simulator, a robot, artificial neural networks, fitness functions and mutation rates to start training and evolving robots. If we want our simulated robots to be more complex, sophisticated, and capable, we're going to need more complex simulation environments, more and more simulations and more and more people building simulations or collectively guiding the evolutionary trajectory of robots. Evidence from previous research done by Joey Anetsberger and Josh Bongard (2016) suggests that crowdsourcing the training of robots is possible using a platform like Twitch, which is one of the leading live-streaming platforms. In their research, they try to teach robots to ground symbols in their felt, sensorimotor experiences. They do this by crowd deploying Twitch Plays Robotics. David Matthews and Josh Bongard (2020) later build upon this project to bypass the loss function by allowing users to supply their own commands and provide rewards to obeying robots respectively. Inspired by their approach I also used Twitch to create Twitch Plays Simulated Soft Robotics to crowdsource the creation of simulated soft robots to non-experts via Twitch.

Soft robots are robots made from highly compliant materials similar to those found in living organisms. And the main reason for using Twitch for this research is because Twitch's chat functionality makes it easier to get user inputs to create simulations of soft robots. Twitch also makes it easy to stream this simulation created from user input and display it to any user on Twitch in real-time.

---

[1] https://voxcraft.github.io/
[2] https://colab.research.google.com/
[3] https://www.twitch.tv/

1.2 Citizen Science: Crowdsourcing Soft Robotics

Crowdsourcing is a useful research technique that allows members of the public to contribute ideas and time to projects even if they lack official expertise in that field (Lichten et al., 2018). Many research projects have successfully employed this method of research. Picbreeder is one of such collaborative art projects where using the methods of evolutionary arts, users are able to evolve images. The user-selected image is evolved using Compositional Pattern Producing Networks (CPPNs), which are neural networks that contain activation functions that allow them to create and evolve images (Secretan et al., 2011). Another such project is Folding@home which, using principles of crowdsourcing, enables any interested user to become a citizen scientist by donating their unused computational power to help fight global health threats (Beberg et al., 2009). One more popular example of a crowdsourcing project is the Red Balloon Challenge by Darpa. They employed crowdsourcing and citizen science to investigate the role of the Internet and social networking in the timely communication, wide-area team formation, and rapid mobilization required to solve large-scale, time-critical challenges. For the task, they deployed ten moored red weather balloons at ten different fixed and visible locations in the United States and asked competitors to be the first to submit the locations of all ten different balloons. The Red Balloon Challenge Team of the Massachusetts Institute of Technology (MIT) found all ten balloons in less than nine hours (Tang et al., 2011). In fact, the crowdsourced approach has been such a great asset to research that a popular platform called Zooniverse[4] was created to enable millions of volunteers to become citizen scientists and assist professional researchers.

Twitch Plays Simulated Soft Robotics (TPSS) follows a similar principle of crowdsourcing science that these projects follow. By crowdsourcing the creation of simulated soft robots to non-experts, TPSS allows the public to contribute their ideas, resulting in diverse, unique, and creative robot morphologies. Similar to the red balloon challenge, TPSS tasks the users with creating a simulated soft robot that moves across the screen as fast as possible. However, TPSS does not rely on evolutionary algorithms like Picbreeder and rather solely relies on the users and their creativity to create simulated soft robots. Additionally, TPSS does not require users to donate unused computational resources like Folding@home and instead uses Google Colab to create simulations of soft robots. Also, unlike any of the above-mentioned projects, TPSS uses a live streaming platform, Twitch, to stream the simulations of soft robots and get user input in real-time.

1.3 The Goals of Twitch Plays Simulated Soft Robotics

Some of the main goals of Twitch Plays Simulated Soft Robotics was to attempt to lower the barrier of entry and allow more and more people to collectively create, train and evolve more robots. Additionally, we were also trying to make simulating soft robots cheaper by using Google Colab, which provides access to computing resources like GPUs and TPUs free of charge directly through the browser (Bisong, 2019). We were also trying to make simulating soft robots more accessible by creating tutorials to scale up Twitch Plays Simulated Soft Robotics to potentially crowdsource the crowdsourcing of soft robotics. As a result, we hoped that more

[4] ttps://www.zooniverse.org/

people will be able to contribute to the field and realize the potential of soft robots. Another goal was also to explore experimental design, how to deploy systems and how to improve the interaction between humans and computers and finally make Twitch Plays Simulated Soft Robotics scalable and replicable in other settings.

By creating Twitch Plays Simulated Soft Robotics, the main question that we wanted to answer was, is it possible to create a software program that allows the crowd to collectively create soft robots? If so, will showing the crowd the soft robots previously created by other users result in the crowd innovating on the robots created by previous users to better fulfil the specified task? Or will this design choice, act as a confounding factor and result in imitation rather than innovation? Additionally, we were also trying to answer the questions of whether non-experts can collectively build soft robots using Twitch Plays Simulated Soft Robotics? What kind of technology and knowledge do they need to help build soft robots? How intuitive is our program for non-experts to use and build soft robots? What sort of morphology can the users come up with to achieve a specific task? We were planning to answer all these questions by crowd deploying Twitch Plays Simulated Soft Robotics and potentially continue to investigate these questions in the future by performing A/B testing with user interface design and scaling up Twitch Plays Simulated Soft Robotics.

2. Methodologies

I used Voxcraft to simulate and visualize soft robots. Voxcraft is a GPU software consisting of two parts namely, Voxcraft-sim and Voxcraft-viz. Voxcraft-sim is a GPU accelerated simulator that I imported into Google Colab to create simulations based on inputs from the crowd, while Voxcraft-viz is a visualization software that visualizes voxelbots in action. Voxels are hollow silicone blocks that can expand and contract in volume (actuation). When multiple voxels are attached together, they become a robot called a Voxelbot (Sida Liu et al., 2020).

2.1 The Task

The crowd was called to make a simulated soft robot that moves across the screen as fast as possible. I allowed users to create simulated soft robots using three different types of materials: red, green, and blue voxels. The blue voxel represents a passive material whereas the red and green voxels represent actuating materials that actuate at different phases based on the temperature of the environment. I used only three materials for simplicity and to minimize confusion for the crowd. Users can create a voxel at any x, y, and z coordinates between 0 and 9 for each x, y and z coordinate. To place a voxel at a certain x, y, z coordinate, the user must enter an exclamation mark “!” followed by the initials of the color of the voxel that they want to create, followed by the x, y and z coordinate. An example command: !r,0,0,0 would place a red voxel at (0,0,0) x, y and z coordinate. The users can remove voxels from certain coordinates by entering in “!” followed by the x, y, and z coordinates. For example, entering !0,0,0 in chat, would remove any voxel located at (0,0,0) x, y, and z coordinate. Users can make the robot move by strategically placing the three different materials.

2.2 Phase I: Crowd Deployment

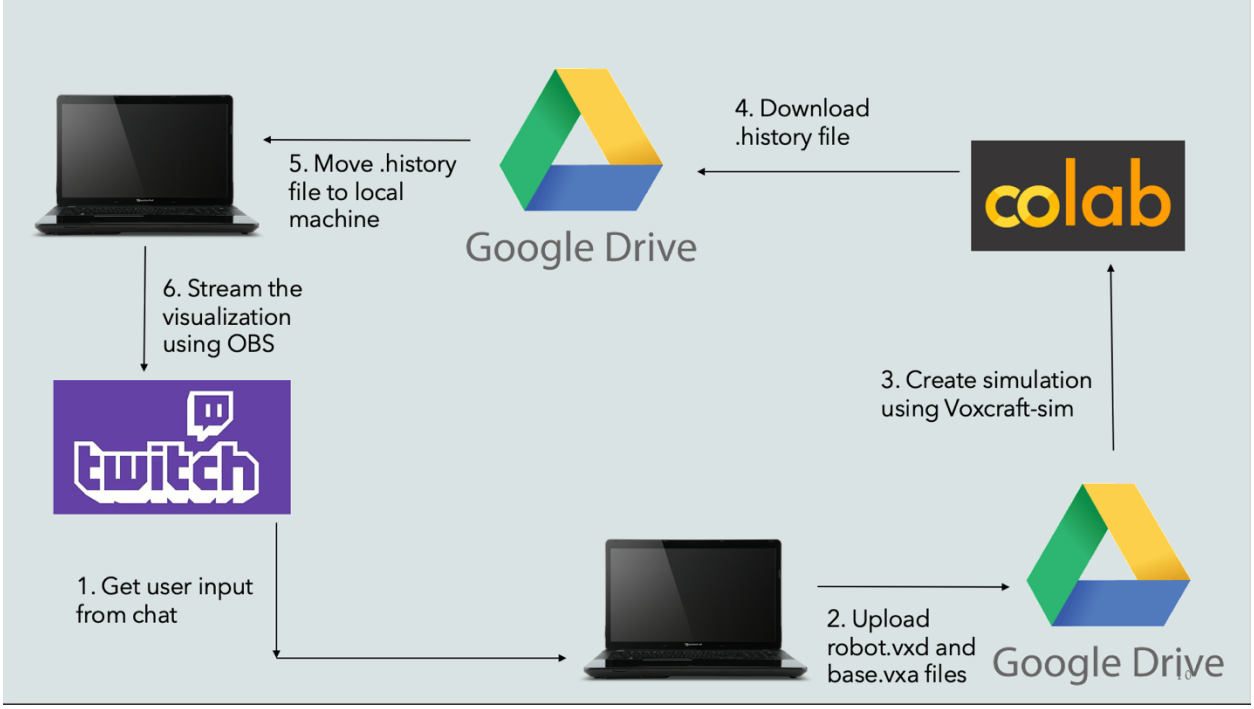

Figure 1: Summary of Crowd Deployment

2.2.1. Getting user input from Twitch and storing it in the local machine

To get the user input from the crowd to build simulations of soft robots, I created a Twitch chatbot using node.js. Upon entering a valid input, users get a response from the Twitch chatbot indicating that their input has been accepted and will be digested and that the simulation will be displayed shortly. The Twitch chatbot would then store these user inputs in a text file on the local machine.

2.2.2. Uploading robot.vxd and base.vxa files to Google Drive

I created a python program that then automatically reads in those user inputs and, using the Julia programming language and voxcraft.jl framework, it creates robot.vxd and base.vxa files, which are necessary for creating the simulations of soft robots. The python program then automatically uploads these two files to a Google Drive account. To do so I just used the Google Drive desktop app. The Drive app can be mounted onto the local machine and upon doing so, it acts like a regular folder on the machine and automatically syncs with the Google Drive cloud account. I went for this method to upload the two files to Google Drive because it does not require using Google's APIs and makes it easier for anyone to replicate this process without needing to have preexisting knowledge of APIs.

2.2.3. Moving the files from Google Drive to Colab

I then created a python program on Colab that automatically mounts Google Drive onto the Google Colab notebook. After importing Voxcraft-sim, Colab creates a simulation using the robot.vxd and base.vxa files. This results in the creation of a .history file, which is required to visualize the simulation of the soft robot created by the users.

2.2.4. Moving .history file from Colab to Google Drive

The python program then automatically moves the .history file back to Google Drive.

2.2.5. Moving .history file from Google Drive to Laptop/Local Machine

The python program then automatically moves the .history file from Google Drive back to the local machine and the .history file is then visualized using Voxcraft-viz. The visualization shows the voxelbot created by the users in action.

2.2.6. Streaming the simulation on Twitch

The visualization of the simulated soft robots created by the user is then streamed to Twitch using OBS (Open Broadcaster Software)[5].

I automated this entire loop using a python program and it takes approximately thirty seconds from getting user input to displaying the simulation to the users on Twitch.

3. Discussion and Results

3.1 Limitations of Google Colab

Google Colab notebooks run by connecting to virtual machines with a maximum lifetime of 12 hours, after which the notebook will have to be reconnected to the virtual machine again. The maximum lifetime can be extended to 24 hours by subscribing to Colab Pro, but that defeats the purpose of this research. Google Colab also enforces usage limits on GPU usage and automatically disconnects from runtime if the connected GPU remains idle for a couple of hours or if users exceed GPU usage limits which are dynamically assigned. These limits are not disclosed by Google for security reasons and to prevent people from misusing Google Colab. If users exceed the usage limit, Google enforces a cool-down period before the user can connect to the GPUs again. This cool-down period can last from a few minutes to a few hours. However, if users remain well within the limits posed by Google, then they would probably not face any such limitations and issues.

For my research, to mitigate these limitations, I firstly created two accounts and alternated between them if either of the accounts exceed the usage limit. To prevent my program from

[5] https://obsproject.com/

running idle and disconnecting from the GPU, I created a placeholder voxel at x, y and z coordinates: 15,15,0 and created a Twitch chatbot that refreshes this voxel every 30 minutes.

3.2 Analyzing user-generated data

We streamed on Twitch for 8 days with an average stream time of just a little more than 6 hours every day. Over these eight days, we were able to garner 100 unique views and get chats from 15 unique chatters and creation of 10 different soft robots. Of the 10 robots, 3 of them were collectively created by two or more people whereas the remaining 7 were created by individual users on Twitch. I anonymized the usernames of the users for privacy reasons.

Table 1: YouTube links to the soft robots that were collectively and individually created.

Soft robots that were collectively created by users.	https://youtu.be/0mzC8K3Y__w
Soft robots that were individually created by users.	https://youtu.be/-05tMgS7nmI

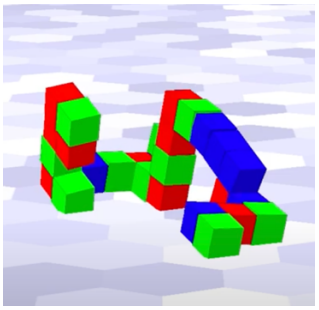

Fig. 2a) collective1

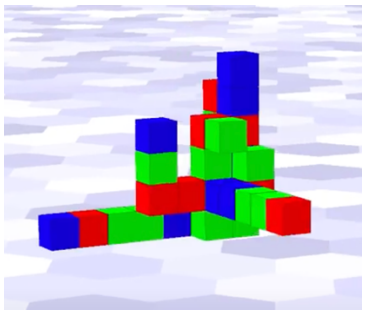

Fig. 2b) collective2

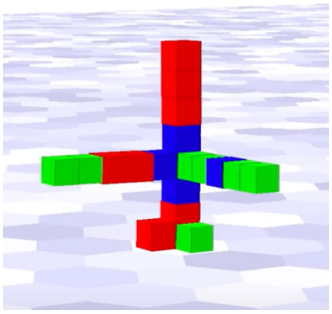

Fig. 2c) collective3

Figure 2: Collectively created robots. The soft robot shown in Fig. 2a) was collectively created by two people and it took them approximately 34.68 minutes to create the robot. The soft robot shown in Fig. 2b) was collectively created by three people and it took them 18.97 minutes to create the robot. The soft robot shown in Fig. 2c) was collectively created by two people and it took them 52.27 minutes to create the robot.

There was a vast gap between the distribution of contributions to create the robots for both Fig. 2a) and 2b) with one of the two people contributing to create a little more than 90% of the robot shown in Fig. 2a) and with one of the three individuals creating a little more than 70% of the robot shown in Fig. 2b). The distribution of contribution for creating the robot shown in Fig. 2c) was much more uniform with each of the two people contributing nearly equally.

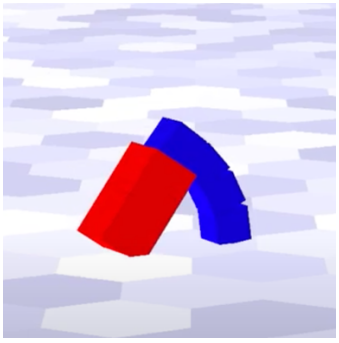

Fig. 3a) Individual1

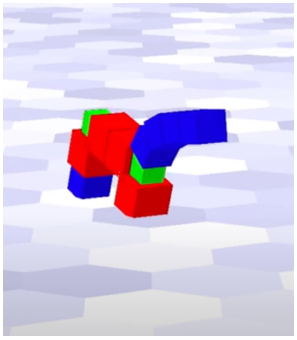

Fig. 3b) Individual2

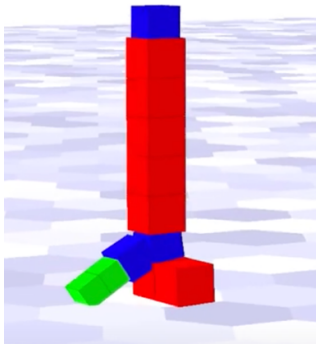

Fig. 3c) Individual3

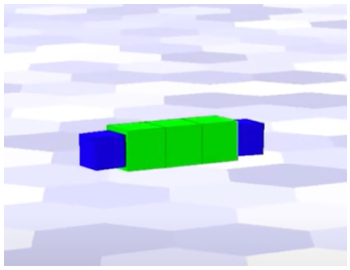

Fig. 3d) Individual4

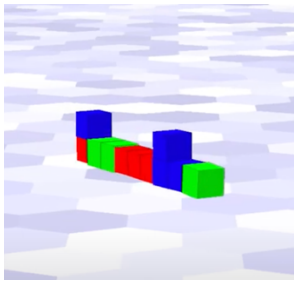

Fig. 3e). Individual5

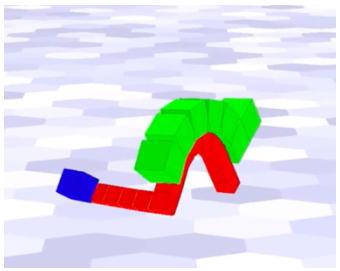

Fig. 3f) Individual6

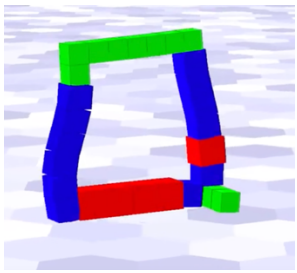

Fig. 3g) Individual3

Figure 3: Individually created robots. The soft robot shown in Fig. 3a) was created in 10.5 minutes. The soft robot shown in Fig. 3b) was created in 11.7 minutes. The soft robot shown in Fig. 3d) was created in 2 minutes. The soft robot shown in Fig. 3e) was created in 20.47 minutes. The soft robot shown in Fig. 3f) was created in 48.23 minutes. The soft robot shown in Fig. 3g) was created in 24.2 minutes.

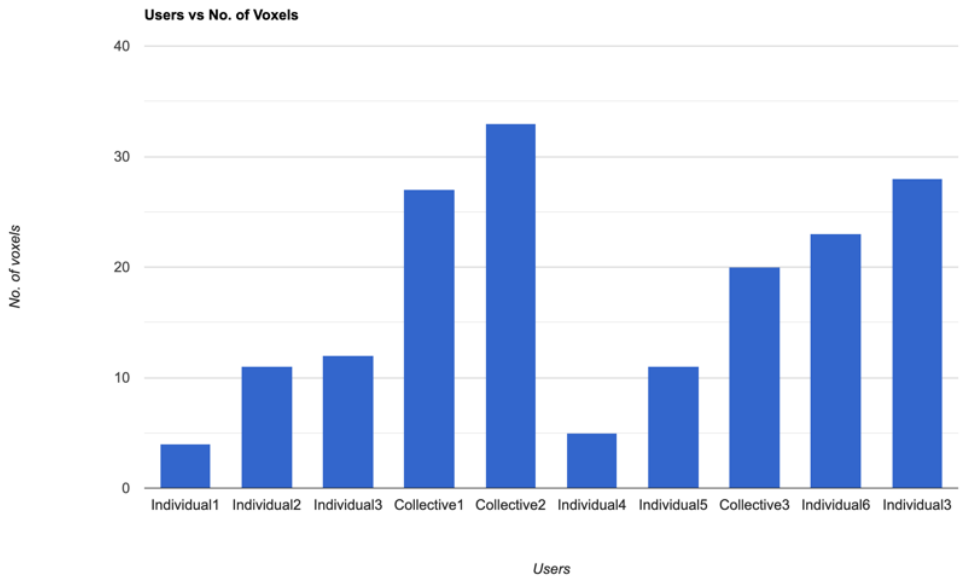

Figure 4: Number of voxels added by users to create their soft robots

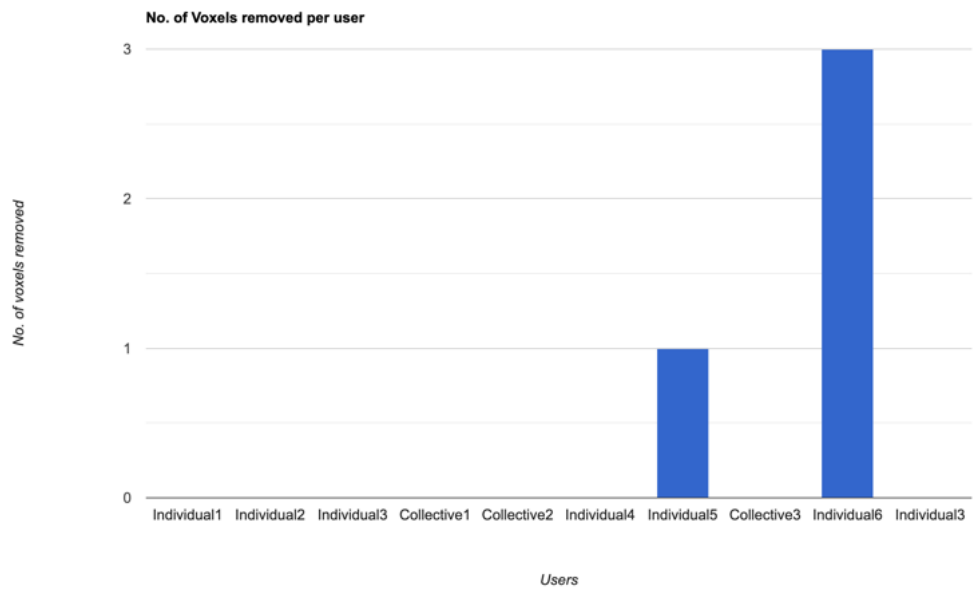

Figure 5: Number of voxels removed by users

From the Fig. 4 we can see that there seems to be a correlation between the number of users creating the robots and the number of voxels used, with users who collectively created the soft robots on average, using more voxels to create the simulation of robots. The two cases where the individuals used a similar number of voxels to those used collectively by two or more people were because both the users had already interacted with Twitch Plays Simulated Soft Robotics (TPSS) to create robots previously and were familiar with the system. As you can see from Fig. 4, individual 3 had created a robot twice and individual 6 was involved in the creation of two of the collectively created robots. So, it seems like they were already familiar with the system and as a result, used more voxels when they used TPSS again. While Fig. 5 shows that only a small amount of the users used the functionality to remove voxels. Due to the small amount of data that we had collected, we weren't able to definitely say as to why this was the case. So, we could only predict that improving the user interface design and making our program more intuitive would likely lead to more users using the functionality.

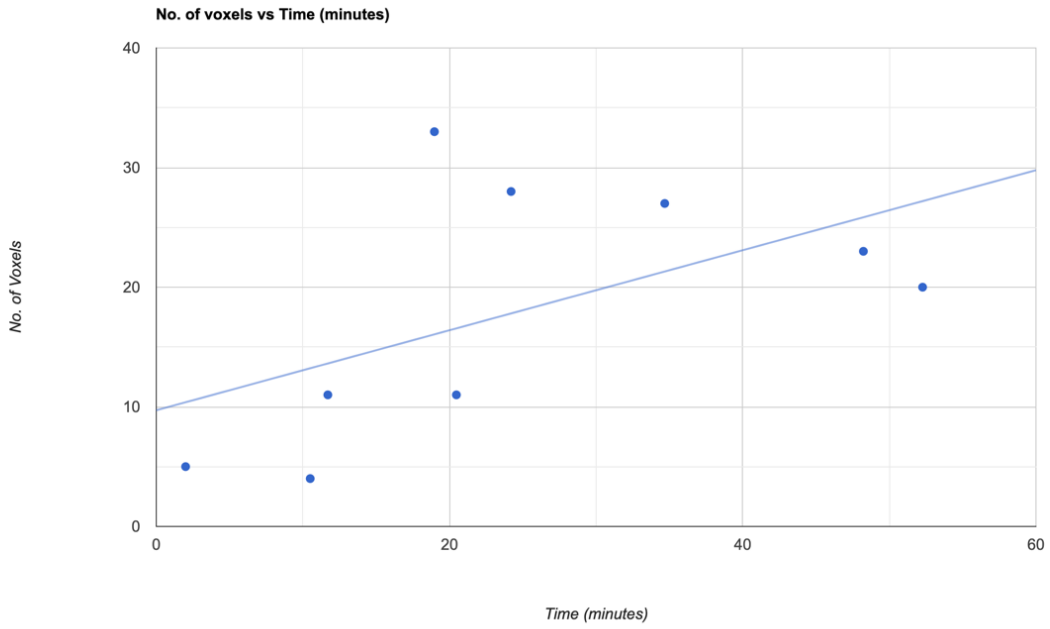

Figure 6: No. of voxels used to create robot vs time spent creating robot

Fig. 6 also seems to depict somewhat of a correlation between time spent creating the robots and the number of voxels used. So, it seems like the more voxels that the users used, the more time they spent creating the soft robot.

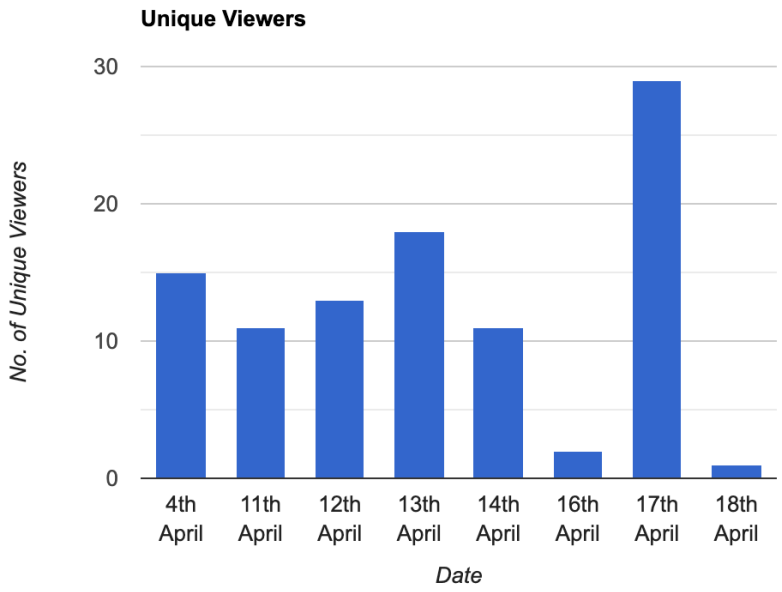

Figure 7: Unique Viewers

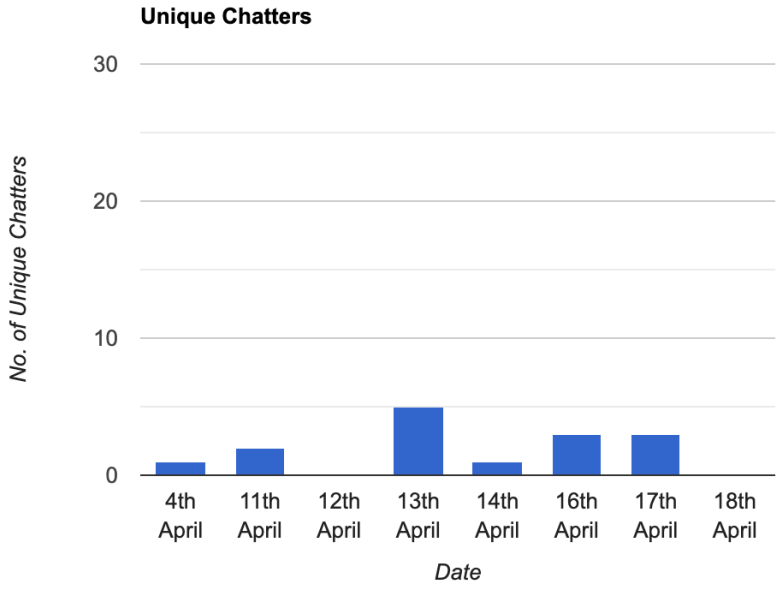

Figure 8: Unique Chatters

Fig. 7 and 8 show that a lot of people just looked at TPSS and only a small minority of people participated in chat to create simulations of soft robots. From Fig. 7 and 8, the implication that I came up with was that we could potentially entice more people to participate by improving the user interface design and by making TPSS more intuitive.

4. Conclusion and Future Work

Here we have demonstrated that it is possible to crowdsource the creation of soft robots and have done so successfully for the first time ever. We have also demonstrated that it is possible for the crowd to collectively create soft robots using Twitch Plays Simulated Soft Robotics.

Future work will involve scaling up Twitch Plays Simulated Soft Robotics by creating a tutorial so that we can have more people using the tutorials to create their own versions of Twitch Plays Simulated Soft Robotics and stream it on Twitch. As a result of which we will be crowdsourcing the crowdsourcing of simulated soft robotics. Additionally, we also wish to integrate all the different pieces of Twitch Plays Simulated Soft Robotics into a Colab notebook such that any user can just click run on the Colab notebook and be able to successfully run and stream Twitch Plays Simulated Soft Robotics on Twitch. By doing so, we not only hope to expand the search space but also get more citizen scientists and non-experts to get involved and contribute their unique, creative and diverse ideas to the field of soft robotics. Additionally, from this research, we found out that, although we had a lot of unique viewers, only a small minority of them participated to create simulations of soft robots. So, we wish to investigate further into user interface design, user engagement, user retention and how to make Twitch plays simulated soft robotics intuitive enough to entice more viewers to participate and create simulations of soft robots. My recommendation to do so is to perform A/B testing using different user interface designs based on the principles of Human-Computer Interaction, to further investigate the intuitiveness of TPSS. And try to figure out what elements, technology and user interface designs would make TPSS more intuitive for non-experts to use and create simulations of soft robots by performing quantitative tests against various functional and non-functional requirements along with randomized the allocation of users to each version.

For future iterations of Twitch Plays Simulated Soft Robotics, we are planning to get user-inputted fitness functions or mutation rates to evolve and train soft robots rather than getting x, y, and z coordinates as inputs from users. Other possibilities that can be explored include, getting user inputs and using natural language processing to help users build simulations of soft robots rather than having them enter x, y, and z coordinates. Finally, we also wish to explore open-endedness and creative differences in the symmetry, functionality, complexity, and morphology between simulated soft robots created by humans and those that are automatically generated and evolved.

References:

Anetsberger, J., & Bongard, J. (2016). Robots can ground crowd-proposed symbols by forming theories of group mind. Proceedings of the Artificial Life Conference 2016, 684– 691. https://doi.org/10.7551/978-0-262-33936-0-ch109

Beberg, A. L., Ensign, D. L., Jayachandran, G., Khaliq, S., & Pande, V. S. (2009). Folding@home: Lessons from eight years of volunteer distributed computing. *2009 IEEE International Symposium on Parallel & Distributed Processing*, 1–8. https://doi.org/10.1109/IPDPS.2009.5160922

Bisong, E. (2019). Google Colaboratory. In: Building Machine Learning and Deep Learning Models on Google Cloud Platform. Apress, Berkeley, CA. https://doi.org/10.1007/978-1-4842-4470-8_7

David Matthews, Josh Bongard; July 13–18, 2020. "Crowd grounding: finding semantic and behavioral alignment through human robot interaction." Proceedings of the *ALIFE 2020: The 2020 Conference on Artificial Life*. *ALIFE 2020: The 2020 Conference on Artificial Life*. Online. (pp. pp. 148-156). ASME. https://doi.org/10.1162/isal_a_00317

Lichten, C.A., Ioppolo, B., d'Angelo, C., Simmons, R., & Jones, M.M. (2018). Citizen Science: Crowdsourcing for Research.

Secretan, J., Beato, N., D'Ambrosio, D. B., Rodriguez, A., Campbell, A., Folsom-Kovarik, J. T., & Stanley, K. O. (2011). Picbreeder: A Case Study in Collaborative Evolutionary Exploration of Design Space. *Evolutionary Computation*, *19*(3), 373–403. https://doi.org/10.1162/EVCO_a_00030

Sida Liu, Matthews, D., Kriegman, S., & Bongard, J. (2020). *Voxcraft-sim, a GPU-accelerated voxel-based physics engine* (3.0) [Computer software]. Zenodo. https://doi.org/10.5281/ZENODO.3835152

Tang, J. C., Cebrian, M., Giacobe, N. A., Kim, H.-W., Kim, T., & Wickert, D. "Beaker." (2011). Reflecting on the DARPA Red Balloon Challenge. *Communications of the ACM*, *54*(4), 78–85. https://doi.org/10.1145/1924421.1924441

Uhlmann, E. L., Ebersole, C. R., Chartier, C. R., Errington, T. M., Kidwell, M. C., Lai, C. K., McCarthy, R. J., Riegelman, A., Silberzahn, R., & Nosek, B. A. (2019). Scientific Utopia III: Crowdsourcing Science. Perspectives on Psychological Science, 14(5), 711–733. https://doi.org/10.1177/1745691619850561