

Sol LeRoot: a Stigmergic Implementation for Distributed Drawing

Sara Falcone
sfalcone@mit.edu
MIT Media Lab

Yonghwan Kim
yonghwankim@gsd.harvard.edu
Harvard GSD M.Des Technology

Abstract . This paper investigates using robot primitives to create drawings in the style of Sol LeWitt. The drawings are done using an off the shelf drawing robot, Root with custom appendages developed by the authors. The robots have no global reference frame and a narrow view of their environment, only being able to see a 3.5” wide row of color underneath the robot. As the robots draw, they stigmergicly augment their environment, affecting the path and decisions of future robots.

1. Inspiration from Sol LeWitt

Sol LeWitt is an american artist known for his contributions to conceptual and minimalist art during the 20th century. Sol LeWitt is particularly well known for a set of over a thousand Wall Drawings, for which he outlined directions to enable their creation and re-creation. These works were titled with a unique ID, and the description of the work comprise the algorithm for its creation. By decomposing

these algorithms meant for human interpretation into algorithms interpretable by robots, and re-creatable using robotic primitives. We aim to create drawings in the style of the Wall Drawings using a multi-robot system. We chose to focus on the drawing shown in **Figure 1**, Wall Drawing #797, #1099 and #876. They were selected for their visual variation, diversity in algorithmic representation and because they are iconic of the artist and this series.

From an art theory perspective this robotic implementation dramatically changes the meaning of the Wall Drawings. One of the contributions of Sol LeWitts work to conceptual art is his implementation where he, as the artist, does not paint or draw the work. In 1971 Sol LeWitt noted that “each person draws a line differently and each person understands words differently” [1]. This variation between individuals and imperfections of human execution resonate throughout his work.



Figure 1. Wall Drawings by Sol LeWitt numbered left to right #797, #1099 and #876

2. Stigmergic multi-robot inspiration

The work of Deneubourg et al. [2] shows an initial study of ant stigmergy. They show that the ant highways are the result of many ants pheromone secretion collecting in high traffic areas. In their experiment with Argentine ant, Deneubourg builds two bridge that separates a food source from a nest. One bridge is twice as long as another, and thus ant who take that path take twice as long to return. As pheromones evaporate in time, the shorter trail collected more pheromone, thus attracting more ants, creating a positive feedback loop. This seminal research introduced a natural method of path optimization that allows the ants to consistently determine the shortest route with very minimal environmental awareness, using only stigmergy.

Marco Dorigo [3, 4] extended Deneubourg et al.'s research with computer simulations to demonstrate effectiveness of an algorithm based on the pheromone trails. Dorigo et al focus on one of the most popular NP-complete problems, the Traveling Salesman Problem in which a person must find the shortest route to visit a given number of cities. For just 15 cities there are billions of route possibilities. The algorithm written by Dorigo et al. robustly optimized the short route for the Traveling Salesman. This research shows the possibility of application of stigmergy mechanism on a real-world problem in computer simulation level.

C. Ronald Kube and Hong Zhang [3, 5] further extend ants stigmergy study into a physical level. In their study, a swarm robots push an illuminated circular box toward a light. These robots act like ant swarms found in nature, which can displace large prey that cannot be carried by a single ant. The most intriguing aspect of this research is that each agent performs their deploying process without communicating with the other agents. They solely use a global communication mechanism

based on stigmergy. This research proves the possibility of the application of stigmergy on a real-world problem in the physical world.

The concept of stigmergy is pivotal to Sol LeWitt's work as previously drawn elements impact the behavior of the drawer and determine the location of future elements. For example, the instructions for Sol LeWitt's Wall Drawing #797 is as follows [6]:

"The first drafter has a black marker and makes an irregular horizontal line near the top of the wall. Then the second drafter tries to copy it (without touching it) using a red marker. The third drafter does the same, using a yellow marker. The fourth drafter does the same using a blue marker. Then the second drafter followed by the third and fourth copies the last line drawn until the bottom of the wall is reached."

Without enough environmental awareness to determine the color and location of the previous line, it would be impossible to create this work. It is interesting to note that the offset line is not a 1:1 offset line but includes interpretation of the next agent as well as noise, because of these imperfections the resulting drawing vary greatly. Results varying this noise level can be seen in works that digitally recreate the Wall Drawings [7-9]

3. Materials and Methods

In order to recreate the wall drawing the robot has two basic affordances: it needs to be able to draw and needs some visual, environmental awareness. There are several pre-existing systems that meet these affordances [10-12]. Due to ease of access we chose to explore the implementation with a Root by Root Robotics [10]. This is an off the shelf product with a

variety of on-board sensors as see below in **Figure 2** copied from the robots' instruction manual. These sensors are read locally on the robot and transmitted via GATT for BLE to a user interface to program and control the robot.

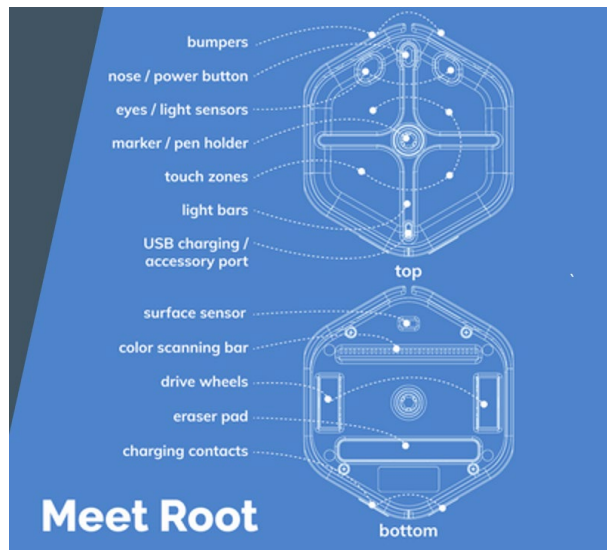


Figure 2. Graphic from the Root instruction manual labeling all its sensors. Note the marker/pen holder in the center of the robot and the color scanning bar in front.

The robot comes with an educational iOS app to teach users basics of coding while interfacing with the robot. This app, Root Coding [13], has three levels allowing users with different experience levels to interact with the robot via a graphical block interface, a graphical computational block interface or a textual coding interface. The wall drawings were deconstructed into four robot primitives: perimeter definition, edge following, color fill and pattern fill. All of the behaviors proposed were demonstrated independently using the iOS app, however limitations of the app prevented nested functions, which necessitated direct control of the robot with Python via BLE [14-15]. For example, **Figure 3** shows part of pattern fill behavior programmed with the iOS app. As the robot needs to stay within the perimeter it needs to respond to visual cues while executing the main function of drawing dashed lines, which is not possible to code with this platform.

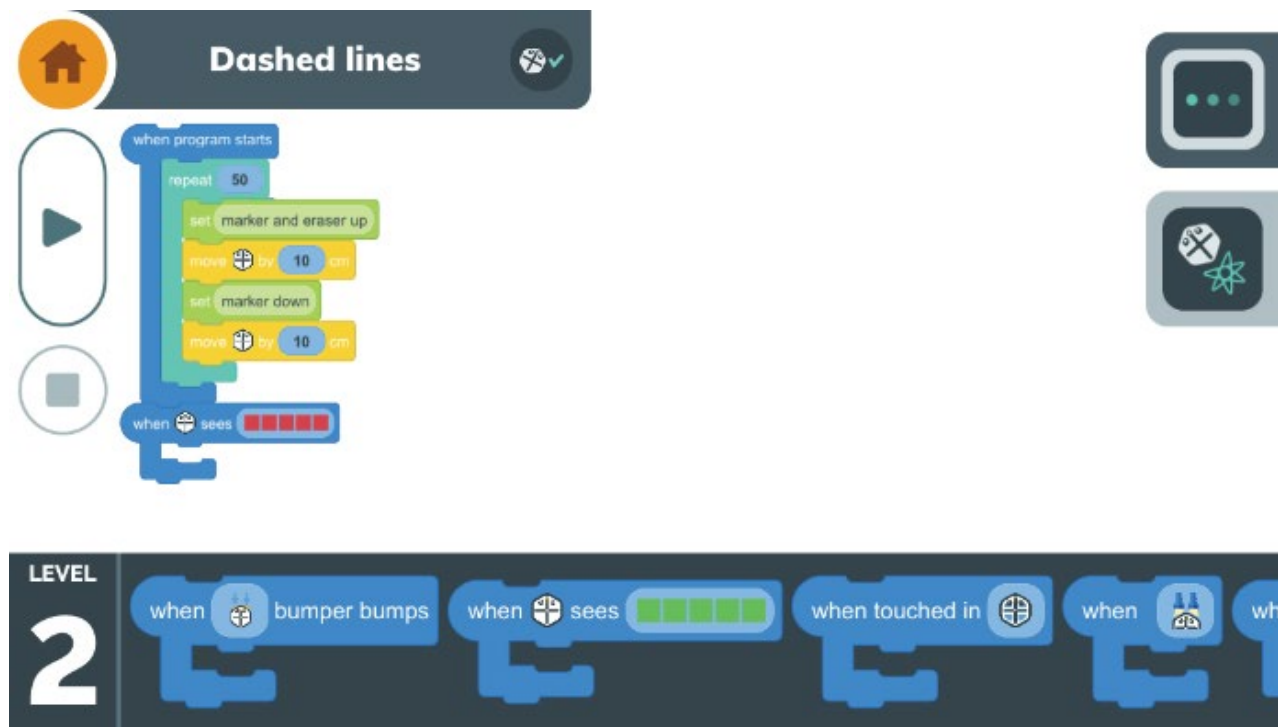


Figure 3. Screenshot of iOS app [13]

3.1 Perimeter definition

Perimeter definition was easily accomplished via several methods. As the perimeter is the first thing to be drawn on the page the robot does not need to respond to previous work and therefore is the only function we could fully utilize the iOS app for. Previous work by Root Robotics also wrote an open source waypoint following script. Pre-existing RootPy functions [16] allowed us to steer and drive the robot and control the state of the pen. We found the speed of the robot less intuitive to control, and the width of the pen was insufficient to clearly define a boundary the robot could respond to. Though several adjacent lines could clearly define the boundary for the robot, this was time consuming to draw and challenging to maintain consistency as the Bluetooth intermittently dropped packets or became disconnected. This led to the development of a custom appendage for the Root which enabled painting of wide stripes. This appendage clipped onto magnetic receptors on the top of the robot and fit into the marker holder to utilize its actuation. It should be noted that the linkage reverses the effects of the pen up and down function, i.e. when the robot is told to draw it lowers the pen, and which raises the ink pad off preventing the robot from painting.



Figure 4. Root with a wide appendage for painting, referred to as the Zamboni

After the perimeter is drawn the subsequent robots need to execute their drawing behaviors

only within the perimeter. Thus, they need to identify when they have encountered a perimeter and turn to remain within the region. As the Root only distinguishes between 5 colors (red, green, blue, black and white), and black was often misread as blue or green, red was chosen to define the perimeter. The Root natively sends a packet each time a sensor reading changes with the type of sensor and new sensor data. We wrote logic, which can be seen in the appendix, that interprets the new data and send commands to the robot causing it to turn and stay within the perimeter.

3.2 Color fill

Color fill is implemented by driving forward with the pen down within the bounds of the perimeter. Note the pen is not wide enough to fill this area within a reasonable amount of time. This test was performed on white vinyl and the wheels turning on the surface partially erased the previously drawn lines, this created the shadowy inner white square.



Figure 5. Initial test with perimeter functions.

3.4 Pattern fill

Pattern fill requires the pen to oscillate up and down at regular intervals while the robot drives within the perimeter. This was accomplished by running a separate thread to keep time independent of the rest of the robot operation. It can be seen below that several of the dashes are longer. Though we didn't prove the source of this issue, it was noted that the longer the robot drew, the less consistently the dashes were drawn. As the robot sends new data whenever a sensor reads a new value, the more lines in the drawing area the more the robots color sensor readings change, and the more packets are sent. It is suspected that this increase in traffic is causing the errors.

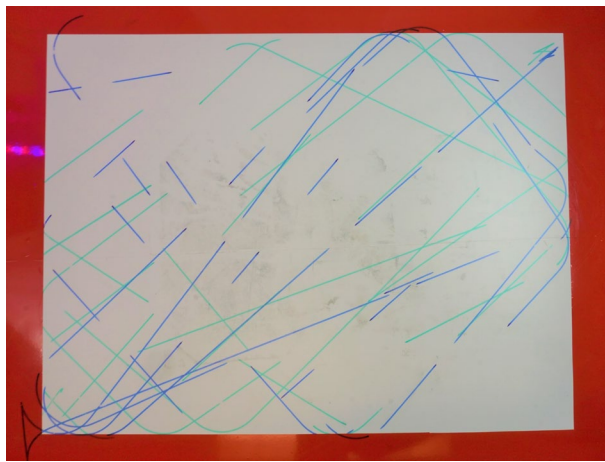


Figure 6. Initial test with perimeter functions and pattern filling dashes

3.5 Edge following

Line following is one of the main functions of this project and is also a common function implemented in introductory courses. However, implementing edge following function with the Root robot was a bit trickier than expected since due to unanticipated challenges communicating with the hardware. First, we make the edge following function inside of RootDevice class which have characteristic_value_updated function and update the sensor_data by using flags called edge_following_enabled. Once the

flag set to True, while implementing the root_robot, it automatically parses the data collected by sensors to the edge following function in real time. After solving the issue of updating real-time sensor value, we implement the function by analyzing the distance between lines and color sensors.

The Root robot has 32 color sensor which can distinguish RGB and black color. The values of 32 sensors are reduced into 16 values. To implement the edge following function, we set one color sensor for the line and compare the distance between the line and blank region and the Root robot turn itself to manage the color sensor of the center to be aligned with the line. After making the initial edge following function, we meme Sol LeWitt's multiple offset line seen in **Figure 1**.

However, the result was not as we expected since the edge following function can only perceive a single line not multiple lines. As a result, the Root robot made amplification of the wavy line for the first few steps and start malfunction after the overlapping of multiple lines transcends the capacity of multiple sensors.

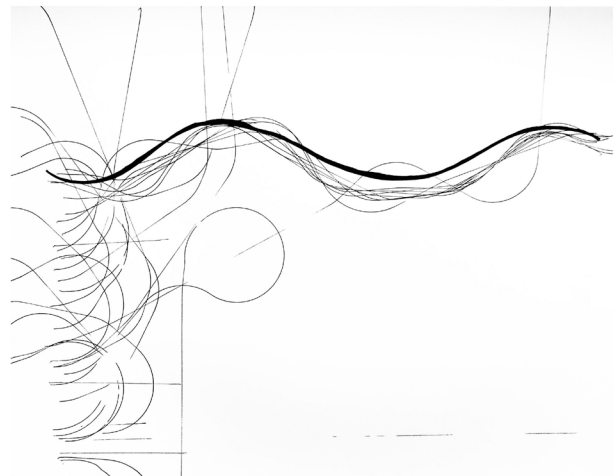


Figure 7. Initial test with edge following function

To fix this problem a customized device was made to allow lines to be offset from the center of the robot, shown in **Figure 8**. This tool needs

to isolate the existing line within the color sensor and draw a new line out of the range of color sensors. After making the customized tool for offset line, we implement the edge following function and it successfully isolated the existing lines and the newly drawn line.



Figure 8. Root with appendage to draw offset lines

4. Evaluation and Conclusion

This demonstration is evaluated visually comparing the works of the robots to the original Wall Drawings selected, #797, #1099 and #876. Though it was not expected that the robots would recreate identical representations of the work, a metric of success for this project is creating images that clearly reference the originals. This demonstrates the translation between human readable instructions and robot primitives. Several tests demonstrated the capacity of the robots to execute commands and draw images following the robot primitives previously outlined, and the intent of the primitive is clear, however the results vary dramatically from the original work, as can be seen in **Figure 9** and **Figure 10**.

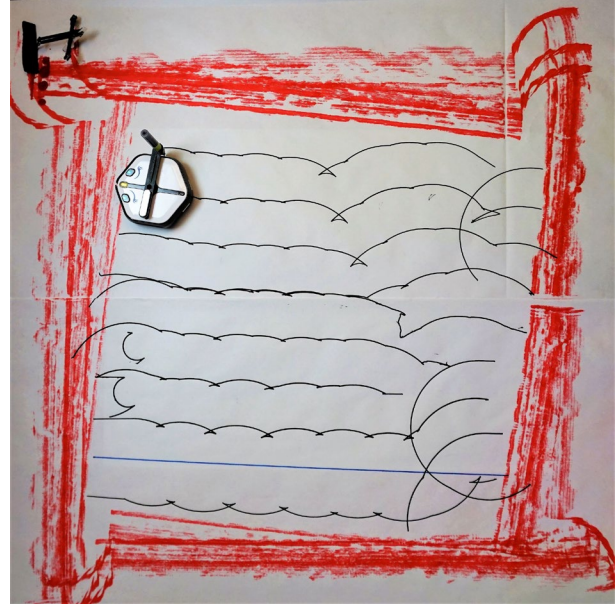


Figure 9. Drawing r797. The perimeter was painted with the Root Zamboni. The lines were filled in using the edge following function and the offset pen appendage.

A significant difference between human implementation and robot implementation can be attributed to the difference in scale. Humans can control a marker on a millimeter scale, and the Wall Drawings spanned meters of wall space, whereas the resolution of the robot is on the order of 5-10 centimeters and our drawings were constrained to one meter by one meter. The difference in resolution between the human and robotic implementation is estimated to be about 35 times higher for the human implementation, making the robot drawing look very crude in comparison.

The robot is also very sensitive to its environment, and with such low-level environmental awareness it is difficult to detect when it is off course and avoid problems. For instance, wrinkles in the paper the robot drew on caused it to veer off course or get stuck. The Bluetooth connection was also very sensitive. Multiple robots could not operate concurrently, and the Bluetooth connection was noticeably spottier when there were more Bluetooth devices available.

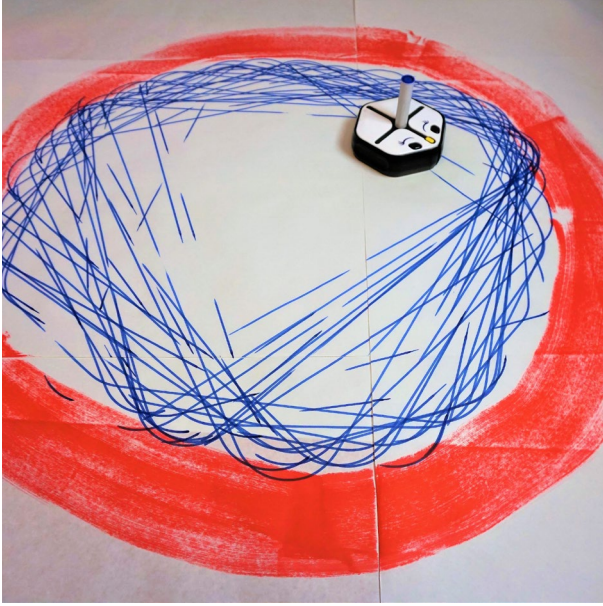


Figure 10. Drawing r1099. The perimeter was painted manually, and the lines are filled in using with the patterning function.

The limitations in drawing area and course motions of the robot also caused the robot to overlap and repeat paths previously followed, which obfuscate the dashed patterning.

5. Future work

This paper focuses on the application of ant's stigmergy to the artworks of Sol LeWitt. This is a unique and protruding way of subjugating and applying science based computational research on artworks. However, the stigmergy is not the only source of the communication method for the swarm insects. Ant's also frequently used reciprocal local communication for their foraging and it is evident that there are some critical regions and edge cases that stigmergic communication cannot cover as we seen in our experiment.

In the future research, we will trigger ourselves for the mixed use of stigmergy and local communication which have more profound coverage on the interpretation of each agent's behavior and how this mixed use of the global and local communication affects the overall effects of interpretation and warping of

information. The key issue will be how to sieve the data collected by Root sensors and how to make a meaning pattern and how to bipartite the transmission of those data to stigmergy and local communication. The pattern data transmission can be implemented in two way, 1st: address_addresable way and 2nd: context addressable way. Each method has pros and cons for the actual usage, but context addressable way seems more intriguing for us.

In another aspect, the main purpose of researching on the swarm robotic behavior is mimicking the language learning and communication mechanism of nature swarm animals. To deeply understand and subjugate their mechanism we need to punctiliously dissect the structure of the abstract language learning process of those swarm animals. In other words, how they interpret the meaning of syntaxes, which were inscribed by other agents, with sparse data collected. To accomplish this purpose, borrowing concepts from Joshua Tenenbaum et al's [17] paper, "How to Grow a Mind: Statistics, and Abstraction", seems useful. In Tenenbaum et al's paper, they use a stochastic generative model that structure the causal process in the words.

$$P(h|d) = P(d|h)P(h) / \sum_{h'} P(d|h')P(h') \mu P(d|h)P(h)$$

When the learner is faced with sparse data, the initial set of data triggers the chain and boiling up the process each chain is actuated below the surface of perception and Bayesian-like stochastic sieve sort those chains and trimmed out based on the causality. (referenced from my prev. Assignment for MIT 6.S081)

By analyzing the swarm robotics stigmergic behavior in the context of Tenenbaum et al's Bayesian chain learning will invoke some interesting result that cannot be achieved by general approach done previously and give some practical cornerstone for the future

research of the abstract language learning of swarm robots.

To recapitulate the discourse above, for the future research, we need to clearly define and analyze what is the key structure of the swarm insects language. Thanks to Deneubourg et al. [1], we already know that the stigmergy is text and the local communication is sound. However, we still need to punctilious figure out what is the structure and syntax of those stigmergy and local communication. By implementing future research of mixed-used stigmergy and local communication with root robot, we expect to build a headstone of swarm insects' transformational syntax.

Also, investigation on the application of practical research will be meaningful to grow the overall boundary of this swarm insect-robot language research. For instance, the application of this research to the Amazon shipment distribution will help them to utilize and improve the efficiency of their shipping methods. Also, the data collected by agents will be commercially used for analyzing customer tendency and focusing on the warp information will give meaningful data why that distortion happens. Another application will be the massive urban transportation. The swarm robots and insects can be metaphorically replaced with cars and other urban transportation methods. To deeply analyze the actual outcome of the application of the stigmergy and local communication is a bit over the scope of this paper but it might be interesting to see how to reduce the problem of urban transportation to swarm insects architecture. Lastly, this research also applicable to the smart cities problem. The key issue in smart cities problem is how to reduce the cost of the sensors that collect data from the city and how to smartly make patterns and transmit those collected data to other agent and organization. The juxtaposition of ants nest to human city and dialect the structure of that two

architecture may provide a meaningful data to computer scientists, architects, and urban designers.

References

- [1] A. Searle, "Second thoughts," *the Guardian*, 07-Dec-2006. [Online]. Available: <http://www.theguardian.com/culture/2006/dec/07/2>. [Accessed: 12-Dec-2018].
- [2] S. Goss, S. Aron, J. L. Deneubourg, and J. M. Pasteels, "Self-organized shortcuts in the Argentine ant," *Naturwissenschaften*, vol. 76, no. 12, pp. 579–581, Dec. 1989.
- [3] E. Bonabeau and G. Théraulaz, "Swarm Smarts," *Scientific American sp*, vol. 18, no. 1, pp. 40–47, Feb. 2008.
- [4] M. Dorigo and L. M. Gambardella, "Ant colony system: a cooperative learning approach to the traveling salesman problem," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 53–66, Apr. 1997.
- [5] C. R. Kube and Hong Zhang, "Collective Robotics: From Social Insects to Robots," *Adaptive Behavior*, vol. 2, no. 2, pp. 189–218, Sep. 1993.
- [6] "Wall Drawing 797 | MASS MoCA." [Online]. Available: <https://massmoca.org/event/walldrawing797/>. [Accessed: 12-Dec-2018].
- [7] "Programmatic Sol LeWitt Wall Drawing." [Online]. Available: <http://blog.benwiener.com/programming/art/2017/08/24/le Witt.html>. [Accessed: 12-Dec-2018].

- [8] “Analogue Coding Processes: Translation and Transmission – Sol LeWitt: ‘Wall Drawing 797’ | Ben Morrissey’s MEDA102 (computational media) Blo.” [Online]. Available:
<https://benmorisseymeda102computationalmedia.wordpress.com/2014/08/21/analogue-coding-processes-translation-and-transmission-sol-lewitt-wall-drawing-797/>. [Accessed: 12-Dec-2018].
- [9] “Sol Lewitt Line Drawing – Physical Computing.” [Online]. Available:
<http://graysonearle.com/edu/physcom/sol-lewitt-line-drawing/>
- [10] “Root Robotics.” [Online]. Available:
<https://rootrobotics.com/>. [Accessed: 12-Dec-2018].
- [11] “Scribit robot could soon be drawing on your walls.” [Online]. Available:
<https://newatlas.com/scribit-wall-drawing-robot/54057/>. [Accessed: 12-Dec-2018].
- [12] “Mirobot - the DIY WiFi robot for children by Ben Pirt — Kickstarter.” [Online]. Available:
<https://www.kickstarter.com/projects/bjpirt/mi-robot-the-diy-wifi-robot-for-children>. [Accessed: 12-Dec-2018].
- [13] “Root Coding,” *App Store*. [Online]. Available:
<https://itunes.apple.com/us/app/root-coding/id1153070718?mt=8>. [Accessed: 12-Dec-2018].
- [14] *Root Robot Bluetooth Low Energy Protocol Documentation: RootRobotics/root-robot-ble-protocol*. Root Robotics, 2018.
- [15] *Bluetooth GATT SDK for Python. Contribute to getsenic/gatt-python*

development by creating an account on GitHub. Senic GmbH, 2018.

[16] C. Anderson, *Python control of Root Robot via BLE. Contribute to zlite/PyRoot development by creating an account on GitHub*. 2018.

[17] J. B. Tenenbaum, C. Kemp, T. L. Griffiths, and N. D. Goodman, “How to Grow a Mind: Statistics, Structure, and Abstraction,” *Science*, vol. 331, no. 6022, pp. 1279–1285, Mar. 2011.

Author Contributions

The authors of this paper co-developed the idea and implementation strategy for the project. They worked together to produce the visual content and prepare the manuscript. Yonghwan Kim focused on implementing the edge following function. Sara Falcone designed the Root appendages and implemented the perimeter and pattern fill functions.

Video Link :

Root multiline following
<https://youtu.be/h9NoDtw8VmM>

Root edge following
<https://youtu.be/kwqVtF-CISE>

Root painter test circle
https://youtu.be/L276xcI_x-w

Root gone wild
<https://youtu.be/R6uyIQomohs>

Root within perimeter
<https://youtu.be/RIbZDw12SdM>

Source code:

<https://gitlab.cba.mit.edu/falcone/solleroot>.
 git