

PeriSim: A Simulator for Optimizing Peristaltic Table Control

Ross M. McKenzie, Jamie O. Roberts, Mohammed E. Sayed, and Adam A. Stokes*

Peristaltic conveyance can be used for the sorting and transport of delicate and nonrigid objects such as meat or soft fruit. The non-linearity and stochastic behavior of peristaltic systems make them difficult to control. Optimizing controllers using machine learning represents a promising path to effective peristaltic control but currently, there is no suitable simulated model of a peristaltic table in which to run these optimizations. A simple, simulated model of a peristaltic conveyor that can be used for optimizing peristaltic control on a variety of peristaltic tables is presented. This simulator is demonstrated through a limited control problem evaluated on our real-world system that is built for peristaltic conveyance. This simulator is available as the python package PeriSim so that it can be used by the robotics community for peristaltic control development.

been created and used for simple tasks.^[2–6] The interest in peristaltic motion comes from the limitations of existing 2D conveyance systems seen in the industry. Industrial 2D conveyance relies on rollers or wheels that push against rigid, flat surfaces^[7,8] and is not suitable for soft, slippery, or round objects. These objects can be manipulated with peristaltic motion, allowing for the automation of new areas such as offal harvesting.^[1]

A major barrier to the adoption of peristaltic systems in industry is in their control. Peristaltic waves are challenging to control because they are nonlinear, temporal combinations of the movements of multiple cells within the table.^[9] The effect of the waves on objects is also highly dependent

on the physical characteristics, velocity, and orientation of the objects. The non-linearity of the surface to object interactions makes it difficult to accurately predict the precise effect that a passing peristaltic wave will have on an object.

1. Introduction

1.1. Peristaltic Motion Can be Used to Sort Delicate and Soft Objects

Peristaltic tables consist of a number of shape-changing cells that connect to form a deformable 3D surface.^[1] These cells are actuated linearly to deform the surface allowing for the automated control of gradients across the surface, as shown in **Figure 1**. Objects that slide or roll can be moved by keeping the gradient at the position of the object biased in the direction of desired travel. This control is usually achieved through a wave-like pattern of cell actuation called a peristaltic wave. A number of hardware implementations of peristaltic systems have already

1.2. Peristaltic Control Can be Optimized in Simulation


Stommel and Xu presented a probabilistic automaton for high-level peristaltic control.^[9,10] Their peristaltic probabilistic automaton uses input from the sensors of the table to determine a state and has a number of high-level wave patterns as actions. The probabilistic result of each state-action pair is determined through multiple trial runs of the system. This approach is independent of the hardware of the system and the object being manipulated, which makes it widely applicable. This approach requires a simulated model of the system as the state-action space can be very large and can take a prohibitive amount of time to explore.

The probabilistic automata approach requires that high-level actions are determined in advance to reduce the size of the state-action space. These high-level actions can be seen as a subset of the larger problem of peristaltic control as they have the same challenges in modeling caused by the nonlinear nature of peristaltic motion. Optimizing these actions provides a foundation for higher-level control and can provide valuable lessons for optimization at this higher level.

Probabilistic automaton control has been demonstrated on an inverse caterpillar motion table by Deng et al.^[11] This table moves an object by lifting it with deformable cells and then moving the point of contact between the cell and the object in the plane of the table. As this type of motion does not rely on objects rolling or sliding, simulations of this table will be mechanically simpler and less stochastic than a simulation of peristaltic motion.

R. M. McKenzie, J. O. Roberts, Dr. M. E. Sayed, Dr. A. A. Stokes
School of Engineering
Institute for Integrated Micro and Nano Systems
Scottish Microelectronics Centre
The University of Edinburgh
Alexander Crum Brown Road, King's Buildings, Edinburgh EH9 3FF, UK
E-mail: adam.stokes@ed.ac.uk

R. M. McKenzie, J. O. Roberts
EPSRC CDT in Robotics and Autonomous Systems
Edinburgh Centre for Robotics
Edinburgh, UK

 The ORCID identification number(s) for the author(s) of this article can be found under <https://doi.org/10.1002/aisy.201900070>.

© 2019 The Authors. Published by WILEY-VCH Verlag GmbH & Co. KGaA, Weinheim. This is an open access article under the terms of the Creative Commons Attribution License, which permits use, distribution and reproduction in any medium, provided the original work is properly cited.

DOI: 10.1002/aisy.201900070

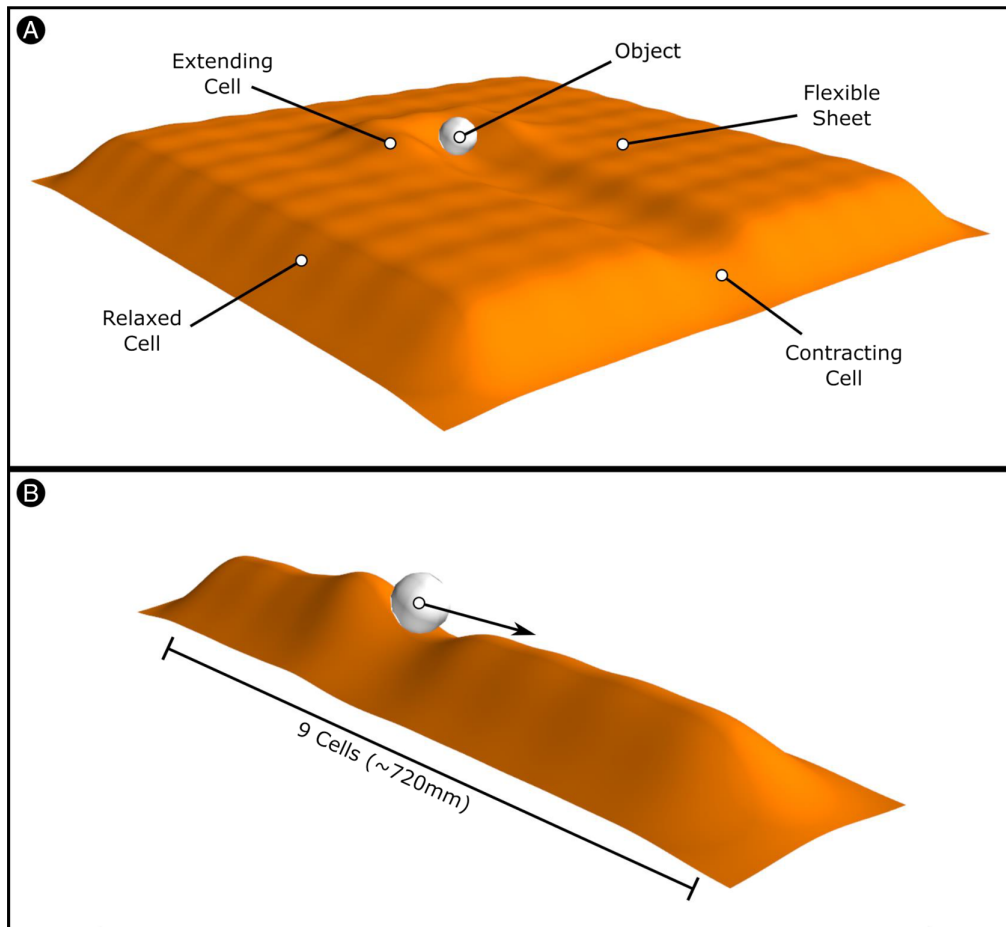


Figure 1. Simulated peristaltic tables. a) A 2D peristaltic table capable of moving an object in the plane of the table. b) The simulated 1D peristaltic conveyor used to optimize peristaltic waves for conveying the object from one end of the conveyor to the other in as short a time as possible.

1.3. Genetic Algorithms Are Ideal for Optimizing Peristaltic Waves

A basic high-level action for a peristaltic table is a single, constant amplitude wave. These waves are created through a series of cells executing a cyclic motion with a delay between their start times. This delay can be the same, for a constant velocity wave, or different to have a wave with acceleration. The exact series of delays to best move an object is highly dependent on the object and peristaltic table hardware. Genetic algorithms have been used successfully for optimizing both high-level^[12] and low-level^[13] robotic behavior. Cheney et al. demonstrated that genetic algorithms can successfully harness soft material properties to produce effective behavior.^[14] This work suggests that a genetic algorithm approach could work for optimizing peristaltic control due to the shared theme of independent, nonrigid components operating together to produce useful work. Genetic algorithms present a simple optimization method to test the feasibility of optimized peristaltic control. This optimization will require a simulated peristaltic table to explore the state space in a reasonable time. This simulation would be imperfect as the nonlinear nature of the system that creates the need for probabilistic automaton control also makes modeling the system in simulation difficult.

1.4. Radical Envelope of Noise to Overcome the Reality Gap

Optimizations will often exploit aspects of a simulation that are different or missing in the real world.^[15] Examples of aspects that often change when moved to the real world include minor variance in actions that are perfectly repeatable in simulation, the frictional behavior of sliding surfaces changing from the simulated model, or internal computation, taking time rather than being “instant” as in the simulation. The lack of, or difference between, these elements can cause reduced performance when that system is used in the real world. This effect is called “the reality gap.” The reality gap can be reduced by learning an offset from simulation to the real world.^[16,17] It is also possible to design a system in such a way that a small section of it can be trained in the real world to overcome the differences.^[18] These methods offer a more directed approach to close the reality gap; however, they require major modifications to the genetic algorithm approach.

The reality gap can also be narrowed through altering the simulation itself. Adding noise to sensor data and uncertainty to the results of actions in a simulation makes controllers that are optimized in that simulation more robust in the real world.^[15] However, choosing the correct level of noise is

necessary for this approach and it still requires a well-validated simulation to work.

An improvement on noisy simulations is to design a simulator around the radical envelope-of-noise hypothesis.^[19] This hypothesis requires that each simulation parameter or aspect is randomly varied between each simulation run. By randomly changing the parameters that control the dynamics of our simulations between runs, we can create controllers that are robust to changes in the underlying behavior of their reality. These controllers can, therefore, be optimized in a simplistic simulator and still work in the real world. The simulator we use must still model all of the key features of the task and system we optimize but can be less complex and rely on less real-world validation.

2. Experimental Section

2.1. Simulation of a Peristaltic Table

2.1.1. Modeling Peristaltic Cells and Objects on the Table

We modeled a peristaltic table as a flexible and extensible surface in the horizontal (XY) plane that was vertically (Z) perturbed by peristaltic cells. Each peristaltic cell was modeled as a Gaussian disturbance on the surface. The amplitude of the Gaussian was determined by the height of the cell. The effect that a peristaltic cell (p) has on the height (h) of the surface at the position of each piece of cargo is

$$h_p = Ae^{-\frac{(\vec{c}-\vec{b}_p)^2}{2\sigma^2}} \quad (1)$$

where A is the height of the peristaltic cell, \vec{b}_p is the XY position vector of the peristaltic cell, and \vec{c} is the XY position vector of the cargo. The term σ is a constant controlling the width of the effect from each peristaltic cell and is the same for all cells. We chose to refer to σ as the standard deviation as it contributes to the shape of the Gaussian disturbance of the cell in the same way that the standard deviation contributes to the shape of a normal probability distribution.

From Equation (1), the effect that each of the peristaltic cells has on the gradient at the position of an object on the surface is

$$\frac{dh_p}{d\vec{c}} = -\frac{(\vec{c}-\vec{b}_p)}{\sigma^2} Ae^{-\frac{(\vec{c}-\vec{b}_p)^2}{2\sigma^2}} \quad (2)$$

The total gradient at the position of an object is a sum over all of the peristaltic cells.

The magnitude of the vertical force applied between the cargo object and the surface (F_v) is

$$F_v = -(mg + F_{act}) \quad (3)$$

where m is the mass of the object, g is the strength of gravity in the simulator, and F_{act} is the magnitude of the force produced by actuating peristaltic cells beneath the object. The value of F_{act} is 0 in all cases other than when the nearest cell to the object actuated within the last t_{act} seconds. The term t_{act} represents the duration of a movement between the actuation states of a cell and is a constant.

The vector angle between the gradient of the surface and the horizontal in the XZ and YZ planes ($\vec{\theta}$) can be calculated as

$$\vec{\theta} = \tan^{-1}\left(\sum_p \frac{dh_p}{d\vec{c}}\right) \quad (4)$$

Geometrically, the angle between the vertical force and the line normal to the plane of the surface is equal to $\vec{\theta}$, shown in **Figure 2**. The magnitude of the reaction force of the surface on the object in XZ and YZ planes (\vec{F}_r) is equal and opposite to the component of the vertical force acting normal to the surface in those planes. Therefore, using Equation (1),

$$\vec{F}_r = (mg + F_{act}) \cos(\vec{\theta}) \quad (5)$$

Geometrically, the angle between the reaction force and vertical is also equal to $\vec{\theta}$, shown in **Figure 2**. The acceleration of the object in the horizontal plane (\vec{a}) can be calculated in an element-wise manner using

$$\vec{a}_i = \frac{\vec{F}_r \sin(\vec{\theta}_i)}{m} - D\vec{v}_i \quad (6)$$

where D is the constant drag factor and \vec{v} is the velocity of the object.

Each simulation step proceeds in the following order: For each object, 1) calculate the gradient at the position of the object; 2) calculate the acceleration; 3) add one timestep of acceleration to the velocity; 4) add one timestep of velocity to the position.

The objects in our simulator were modeled as point masses and as such, their physical volumes did not interact with the surface or with other objects. This behavior is suitable for modeling other peristaltic tables as from our research into peristaltic table development (see Section 1.1); most systems do not rely on

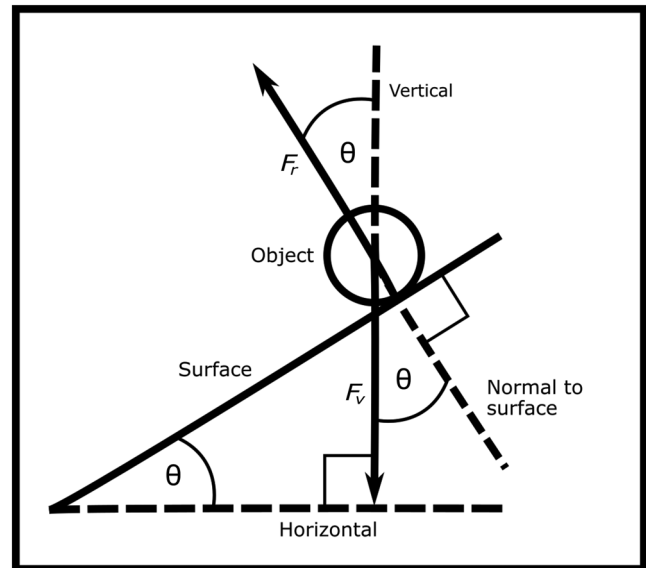


Figure 2. Schematic of an object on the surface of a peristaltic table. The term θ is the angle between the gradient of the surface and horizontal, F_v is the vertical force applied between the cargo object and the surface, and F_r is the reaction force of the surface on the object.

object-surface interactions other than moving along gradients and do not intentionally bring multiple objects into contact with each other. Our simulator used PyTorch tensors for most of the calculations it performed. PyTorch is a deep-learning python library that provides access to high-performance tensor operations.^[20] The PyTorch tensors we used in our simulator could contain information about all the objects or cells on the table, allowing calculations involving every cell and every object to be completed with one-tensor operation without looping over variables.

While the default values of the parameters of the simulator were tuned to model the hardware we used for testing, the parameters were easy to access and change. Other peristaltic table researchers can tune these parameters to model their own systems in our simulator without needing to change the source code.

2.1.2. Radical Envelope of Noise

We implemented the radical envelope of noise in this simulator by a perturbing every base parameter randomly at the beginning of each simulation run. For simulation run (i), a parameter (ω_i) has a default value (ω_0) and a range around that default value in which it may lie in each simulation. We referred to this range as the noise level (N). At the start of each simulation run, each parameter was calculated such that

$$\omega_i = \omega_0 + rN\omega_0 \quad (7)$$

where r is a random number picked from a uniform distribution between -1 and 1 . The noise level therefore could have a significant effect on the behavior of the simulation created at each run.

2.2. Design of the Peristaltic Conveyor

To test our simulation, we designed a peristaltic conveyor based on Linbot actuators and in a similar design to our previously demonstrated peristaltic conveyor.^[6] Linbots are modular robots that function as the cells in a peristaltic system. They are capable of actuating linearly, tactile sensing, and communicating with one another. To test our simulator, we arranged nine Linbots in a row to make a one dimensional peristaltic table, or peristaltic conveyor. We enclosed the conveyor with walls on both sides to stop objects from falling off. We used a sheet of acetate on top of the cells for the surface of our peristaltic conveyor. The experimental setup is detailed in **Figure 3**. We then created a model of the conveyor in our simulator with a 9×1 array of peristaltic cells and tuned the parameters of the cells to match the behavior of Linbots. We used the same control architecture for the simulated model and real-world system. This architecture is detailed in Section 2.3.1

2.2.1. Simulator Calibration with Real-World Data

We set the parameters of the simulator to match our real-world system. As the simulator is designed to be simplistic, it uses parameters that do not have direct real-world versions or that cannot be easily measured, such as the standard deviation of the Gaussians modeling the peristaltic cells. To set these values, we compared the behavior of simulation runs against the real-world experiments.

We ran our real-world system with an identical known delay for each Linbot and filmed the results. We used a circle detection algorithm from OpenCV^[21] to track the position and velocity of

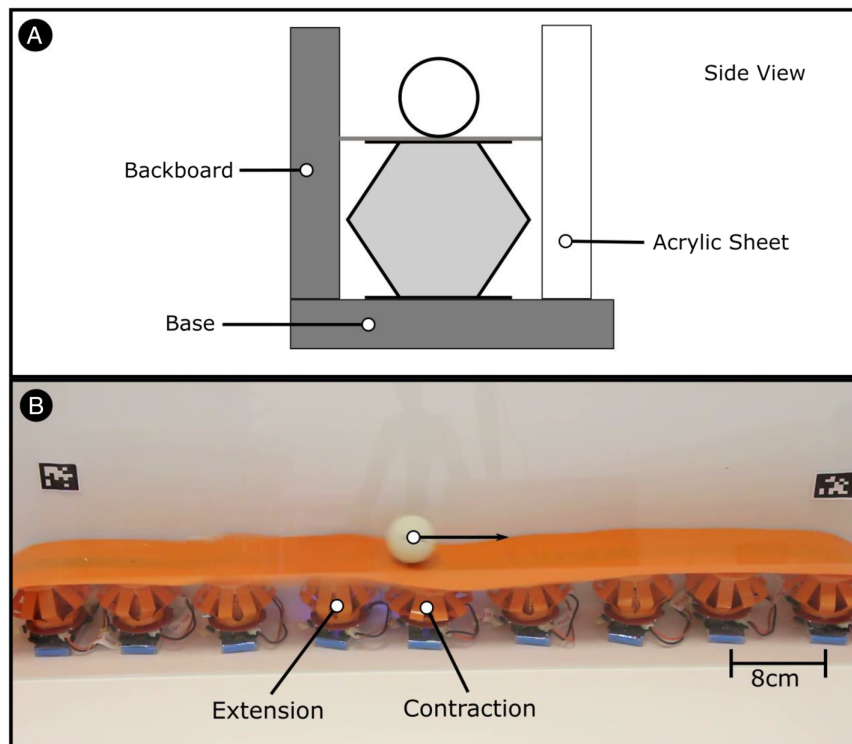


Figure 3. System overview. A) A side view schematic of the experimental setup. B) The peristaltic conveyor during an experiment.

the ball and compared the results to those of the same experiment in the simulator. We then calibrated the parameters of the simulator to produce the most realistic results possible. The simulator runs still showed unrealistic effects caused by the simplicity of the simulator, but the results were close enough to allow us to optimize robust controllers.

2.3. Design of the Control

2.3.1. Genetic Controller

We designed a state machine controller that has an identical structure for each peristaltic cell but each has an individual

control variable. The nine control variables were stored in a genetic code that is used to define a collective wide controller. Each variable was between zero and one. Each variable described the time, in seconds, that the respective peristaltic cell will remain in a contracted or extended state. The position of a gene describing the delay of a peristaltic cell in the genetic code was determined by the position of the corresponding cell in the conveyor. The state machine for each cell and the related extensions are shown in **Figure 4**. A simulated model using this controller to create a constant velocity peristaltic wave is shown in Video S1, Supporting Information. The control variables used in Video S1, Supporting Information, are all equal to 0.3.

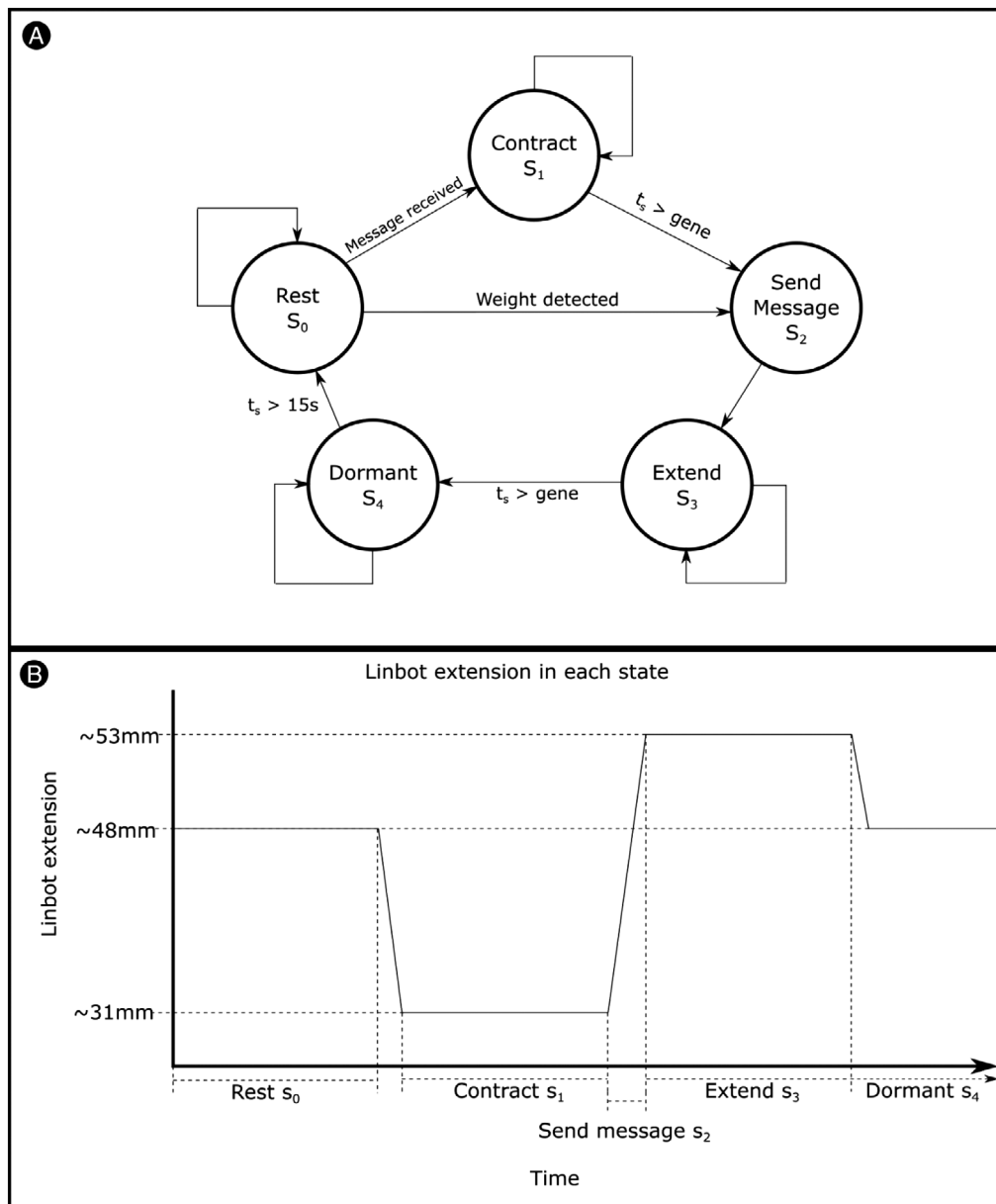


Figure 4. System overview. A) A schematic showing the state machine controller for each peristaltic cell with delays related to the genetic code. Unlabeled links represent “else” actions. B) The extension of the peristaltic cell in each state.

2.3.2. Optimizing the Controller

We generated a population of random genetic codes to use as controllers for the simulated conveyor. Each time a controller was run a fitness value was calculated and stored. During the optimization process, the fitness of each controller was averaged over ten runs in a simulation.

The fitness (f) is calculated using

$$f = -x_{\min} - 2t \quad (8)$$

where t is the time taken for the task to finish with a maximum of 10 s, and x_{\min} is the minimum distance to the end of the conveyor reached during the test; these variables are clarified in Figure 5A. We doubled t to increase evolutionary competition in more successful controllers, see below.

We optimized the fitness of the controllers via a genetic algorithm approach in simulation. We used a genetic algorithm closely aligned with previous approaches,^[22] which is demonstrated in Figure 5B. A population of controllers was randomly created and tested. We then created a new population based on members of the previous population. The probability of a controller being selected to be used for the creation of a member of the next generation is proportional to the relative fitness of the controller compared with the rest of the population. Once chosen, the selected controller underwent a crossover with a different controller selected with the same rule. The two resulting controllers then had a small chance for each gene to be mutated to create a new genetic code used for a controller in the next generation. Crossovers split the codes of two parent controllers and then created a child code by combining the first section of one parent code with the second of another and vice versa for a second child code. Mutations involved changing the value of one or more randomly selected genes in a child code within a small envelope (<0.1).

During testing, we discovered that controllers that only moved the object a short distance along the conveyor had large x_{\min} values compared to the maximum value of t . This large difference led to more successful controllers having small differences in relative fitness when compared to the minimum fitness value of the population. This narrowing of comparative fitness values reduced the selection pressure on successful controllers and reduced the exploitation of different successful strategies. We rectified this imbalance by doubling t in our fitness calculations.

For our optimization, we used 50 generations with a population of 1000 per generation.

2.4. Sampling from Noise Levels

Building higher levels of noise into the parameters of a simulator led to the optimization of controllers to work for a wider range of different parameters, making them more likely to work when moved to real-world robotic system. These controllers, however, could not exploit their environment to complete a task as much as if lower noise levels were used in the simulator. This trade-off means that controllers from a variety of noise levels need to be sampled to find the best controller. These noise levels were a percentage by which each base parameter could be perturbed at each simulation run and that the acceleration of the object could be

perturbed in each simulation timestep. We varied the noise levels from 0% to 50% in steps of 5%.

2.5. Design of the Physical Experiments to Evaluate the Optimized Controllers

To determine the best real-world controller, we tested the best controllers from each optimization. We selected the top five controllers from each optimization and programmed the control variables into our peristaltic conveyor. We then tested each selected controller by placing a ball on the right-hand half of the first cell and recording the resulting peristaltic conveyance. We calculated the fitness of each controller using Equation (7) We repeated the test ten times for each selected controller and averaged the measured fitness values.

The data collected during the experiment was in the form of videos and so we needed to extract the necessary features using computer vision. We developed a tool to evaluate the peristaltic controllers using Python with OpenCV and AprilTag libraries.^[23] The AprilTags served as the static and reliable start and finish positions with which to evaluate each peristaltic conveyance. We placed the AprilTags above the first and last cells of the system. AprilTag detection is robust, reliable, and is handled by the supporting libraries. We used the OpenCV implementation of the Hough Circle detector^[24] to detect the ball in each frame. We tuned the Hough Circle detector parameters for each video via an interactive script that saved manual changes in parameters to file. We later used the parameters in an automated script for controller evaluation. We developed our evaluation tool to be able to detect when a run is finished, classify whether the run is a success, and record the time and distance information from each run. A screenshot of our evaluation tool is shown in Figure 5A.

3. Results and Discussion

3.1. Using PeriSim

We made our simulator available as a python package called PeriSim, which can be installed via `pip install perisim`. Full instructions and the source code are available at: <https://github.com/stokesresearchgroup/perisim>. PeriSim provides a python object that stores the state of a peristaltic table and calculates the next state. The user provides variables for the initial state of the system and controls the system via writing a controller object or by using the PeriSim object functions to control the cells and updates. PeriSim can also create 3D visualizations of the simulator state via `mlab`;^[25] examples are shown in Figure 1 `mlab` is a python script interface of the scientific data visualization application Mayavi.^[26] These visualizations can be created from one-function call and changed between static and animated via function arguments.

3.2. Performance of the Optimized Controller

We successfully used our PeriSim package to optimize a peristaltic wave controller on the simulated model, shown in Figure 1B. The optimized controller successfully conveyed a ball across the real-world test system in all ten runs. This controller was evolved

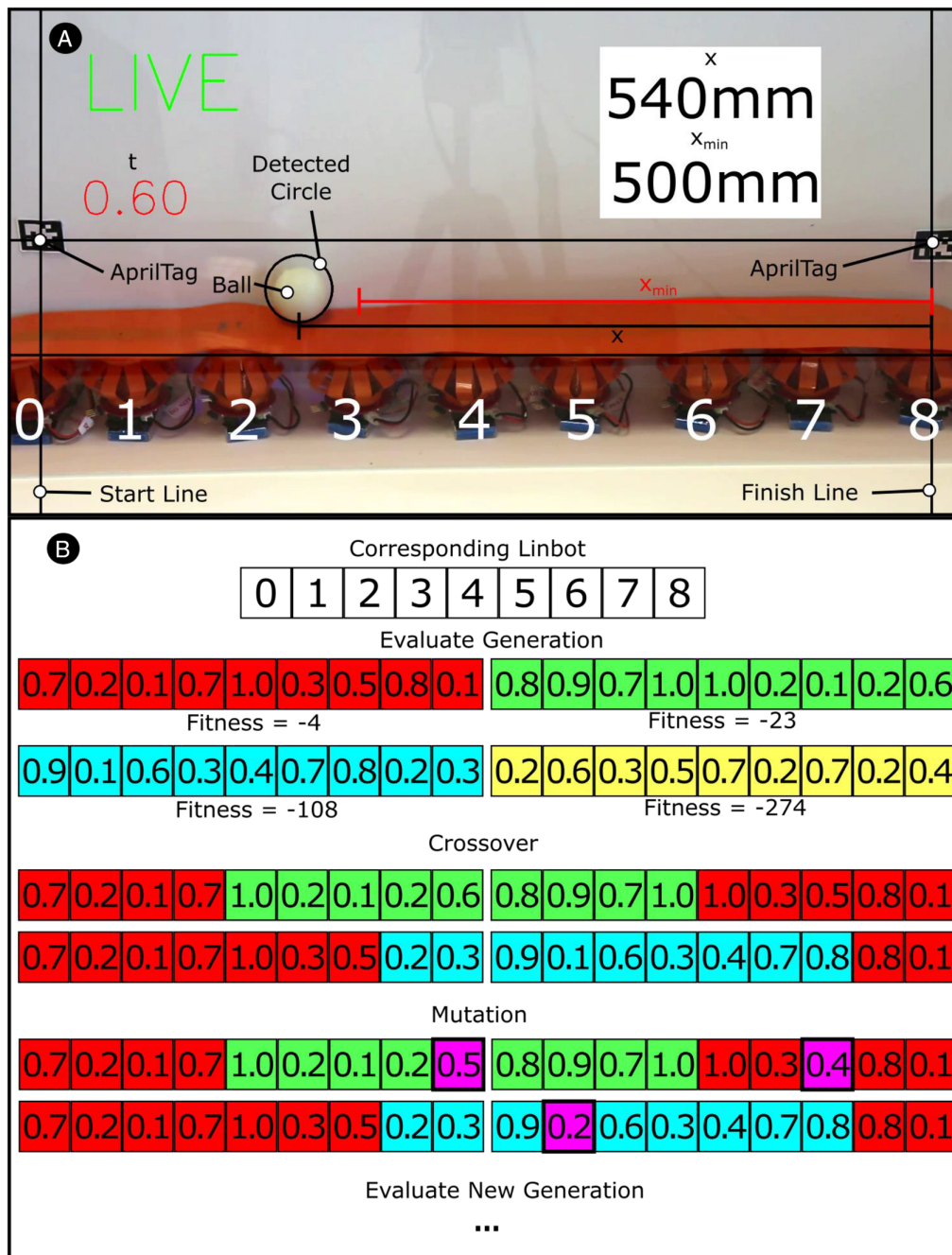


Figure 5. Schematic of genetic algorithm approach. A) A screenshot of data gathering software with annotated information about fitness, f , where $f = -x_{\min} - 2t$. The term t is the time taken for the task to finish with a maximum of 10 s, and x_{\min} is the minimum distance to the end of the conveyor reached during the test. B) A diagram of a single optimization step of the genetic algorithm. The probability that a genetic code is selected to create part of the next generation is proportional to the fitness of the controller based on the code.

with PeriSim set to a 35% noise level and has an average conveyance time of 1.8 s and so an average fitness of -1.8 . The three controllers with the highest real-world fitness are detailed in **Table 1** along with the controller that produced the lowest average conveyance time in successful runs. Video S2, Supporting Information, shows the controller with the highest fitness being used on the peristaltic conveyor. Video S2, Supporting

Information, shows how the control variables are optimized such that the wave pushes the object effectively off the edge.

There were perturbances in the starting position of the ball as well as a chaotic interaction between the ball, the surface, and the sides of the conveyor. These random factors led to every real-world run behaving differently and we did not include them in PeriSim. The performance of the evolved controllers, despite

Table 1. The three selected controllers with the highest fitness (\uparrow) and the controller with the lowest average time of successful runs (\downarrow). The value of interest for each controller is highlighted in bold.

Simulation noise [%]	Simulation fitness	Real-world fitness	Average time for successful runs [s]	Successful run proportion [%]
35	-5.25	-3.61 \uparrow	1.80	100
35	-5.22	-4.42 \uparrow	2.21	100
10	-4.74	-28.8 \uparrow	1.66	90
30	-4.62	-127.2	1.23 \downarrow	50

these random factors, shows that the controllers are robust to effects that were not explicitly modeled in the simulator.

3.3. Scope for Development

3.3.1. Using PeriSim to Optimize 2D Peristaltic Conveyance

Our simulator can be used for optimizing 2D peristaltic behavior. The degree to which real-world perturbances can disturb the controller will be increased, and it is likely that a different level of noise will be optimal for evolving a controller.

3.3.2. Object Volumes and Shapes

PeriSim can be extended to better model real-world effects. These effects could include the objects no longer being modeled as point masses but instead as masses with a volume that can interact with one another. Improving our simulator in this way also opens up the chance to have objects with different shapes that have more complex interactions with the surface medium. The extension to different shapes will allow for testing that objects can be moved by an existing peristaltic system or aid with the design of new peristaltic systems by modeling the gradients that will be required to roll or slide the objects the system is being designed for. Modeling this extra factor will increase the computational power used by our simulator and so will need to be important to the task being modeled.

3.3.3. Vibrational Sorting

The effect of high-frequency actuation on objects could be added to PeriSim. This would require adding firmness and density characteristics to objects and approximating the effect that these values have when an object is moved down a vibrating slope. Modeling high-frequency actuations will allow for vibrational sorting within PeriSim.

4. Conclusions

Peristaltic tables represent a promising area of research in advanced intelligent systems for the sorting of soft and delicate objects. The control of these tables remains a challenge due to each table being made of a large number of soft, stretchy cells. Machine learning represents an opportunity to create controllers for peristaltic tables, but simulated environments are necessary

for the optimization of these controllers. We presented a novel simulator, PeriSim, that models the fundamental behavior of a peristaltic table using tensor-based mathematics. We used a radical envelope of noise to account for the inherent differences between simulators and the real world when optimizing controllers in PeriSim.

We demonstrated the use of PeriSim for an optimization task using genetic algorithms. This optimization required a large number of simulations to produce optimal results, which was facilitated by PeriSim only modeling fundamental aspects of a peristaltic table. The optimized controller works in the real world despite the simplicity of our simulator.

PeriSim will be of use to anyone exploring the use of peristaltic sorting or conveyance. The radical envelope of noise we used in our simulator will help others produce real-world optimized controllers while avoiding a large reality gap.

Supporting Information

Supporting Information is available from the Wiley Online Library or from the author.

Acknowledgements

This study was supported by the EPSRC via the Robotarium Capital Equipment, and CDT Capital Equipment Grants (EP/L016834/1), and the University of Edinburgh and Heriot-Watt University CDT in Robotics and Autonomous Systems. A.A.S. acknowledges the support from the EPSRC ORCA Hub (EP/R026173/1).

Conflict of Interest

The authors declare no conflict of interest.

Keywords

object conveyance, modular robotics, peristaltic systems, sorting

Received: July 3, 2019

Revised: September 18, 2019

Published online:

- [1] M. Stommel, W. L. Xu, P. P. K. Lim, B. Kadmiry, in *Robot Intelligence Technology and Applications 3: Results from the 3rd International Conference on Robot Intelligence Technology and Applications*, (Eds: J. Kim, W. Yang, J. Jo, P. Sincak, H. Myung, Springer International Publishing, Cham **2015**, pp. 605–615.
- [2] S.-R. Kim, D.-Y. Lee, J.-S. Koh, K.-J. Cho, in *2016 IEEE Int. Conf. Robotics Automation*, IEEE, Piscataway, NJ **2016**.
- [3] A. A. Stanley, A. M. Okamura, *IEEE Trans. Haptics*, **2015**, 8, 20.
- [4] P. Schoessler, D. Windham, D. Leithinger, S. Follmer, H. Ishii, in *The ACM Symp. User Interface Software Technology*, Charlotte, NC, USA **2015**.
- [5] R. Hashem, B. Smith, D. Browne, W. Xu, M. Stommel, in *IEEE 14th Int. Workshop Advanced Motion Control*, IEEE, Piscataway, NJ **2016**.
- [6] R. M. McKenzie, M. E. Sayed, M. P. Nemitz, B. W. Flynn, A. A. Stokes, *Soft Robotics*, **2019**, 6, 195.

- [7] C. Uriarte, H. Thamer, M. Freitag, K.-D. Thoben, *Logistics J.: Proc.* **2016**, https://doi.org/10.2195/lj_Proc_uriarte_de_201605_01.
- [8] Intralox, Activated Roller Belt (ARB), **2019**.
- [9] M. Stommel, W. L. Xu, in *IEEE Int. Conf. Mechatronics*, IEEE, Piscataway, NJ **2015**.
- [10] M. Stommel, W. Xu, *IEEE Trans. Autom. Sci. Eng.* **2016**, *13*, 858
- [11] Z. Deng, M. Stommel, W. Xu, *IEEE/ASME Trans. Mechatron.* **2016**, *21*, 1702.
- [12] T. W. Manikas, K. Ashenayi, R. L. Wainwright, *IEEE Instrum. Meas. Mag.* **2007**, *10*, 26.
- [13] E. Yoshida, S. Murata, A. Kamimura, K. Tomita, H. Kurokawa, S. Kokaji, *J. Robot. Mechatron.* **2003**, *15*, 227.
- [14] N. Cheney, R. MacCurdy, J. Clune, H. Lipson, in *Proc. Genetic Evolutionary Computation Conf.*, **2013**.
- [15] N. Jakobi, P. Husbands, I. Harvey, presented at *Proc. Third European Conf. Artificial Life*, Granada, Spain, June, **1995**.
- [16] A. Shrivastava, T. Pfister, O. Tuzel, J. Susskind, W. Wang, R. Webb, in *IEEE Conf. Computer Vision Pattern Recognition*, IEEE, Piscataway, NJ **2016**.
- [17] P. Abbeel, M. Quigley, A. Y. Ng, *Proc. 23rd Int. Conf. Machine Learning*, ACM, New York, NY, **2006**.
- [18] K. Bousmalis, G. Trigeorgis, N. Silberman, D. Krishnan, D. Erhan, Domain Separation Networks, *Proc. 30th Int. Conf. Neural Information Processing Systems*, Barcelona, Spain **2016**.
- [19] N. Jakobi, *Adapt. Behav.* **1997**, *6*, 325.
- [20] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga and A. Lerer, *31st Conf. Neural Information Processing Systems*, Long Beach, CA, USA **2017**.
- [21] G. Bradski, *Dr. Dobb's J. Softw. Tools* **2000**, *120*, 122.
- [22] K.-F. Man, W. K.-S. Tang, S. Kwong, *IEEE Trans. Industr. Electron.* **1996**, *43*, 519.
- [23] J. Wang, E. Olson, in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots Systems*, IEEE, Piscataway, NJ **2016**.
- [24] J. Illingworth, J. Kittler, *IEEE Trans. Pattern Anal. Mach. Intell.* **1987**, *9*, 690.
- [25] Mayavi, mlab: Python scripting for 3D plotting, Mayavi, [Online]. <https://docs.enthought.com/mayavi/mayavi/mlab.html>. (accessed: September 2019).
- [26] P. Ramachandran, G. Varoquaux, *Comput. Sci. Eng.* **2011**, *13*, 40.