Optimising Soft Robot Designs through an Integrated Environment

Syed Ismail Ahmad and Helge Wurdemann

***Abstract*— Simulation-driven optimisation is increasingly utilised in the design and control development of soft robots and actuators. However, setting up such an optimisation pipeline is complex, often requiring the integration of multiple software tools and algorithms, which can compromise robustness.**

To address these challenges, we propose a single integrated environment that allows for the flexible description of soft robots and actuators. Our system robustly handles geometry generation, meshing, simulation, and optimisation.

We achieve this by using implicit geometry shape functions and voxelisation to create tetrahedral meshes, followed by Extended Position-Based Dynamics (XPBD) to simulate soft materials. As XPBD lacks physical constants, we use an evolutionary optimisation algorithm to calibrate simulation parameters to real-world behaviour and assess how geometry and voxel count affect simulation accuracy. Once calibrated, we find these parameters enable accurate simulations of more complex geometries.

Finally, we validate the effectiveness of our integrated environment by optimising a cylindrical soft actuator, demonstrating its potential as an optimisation platform for the field of soft robotics.

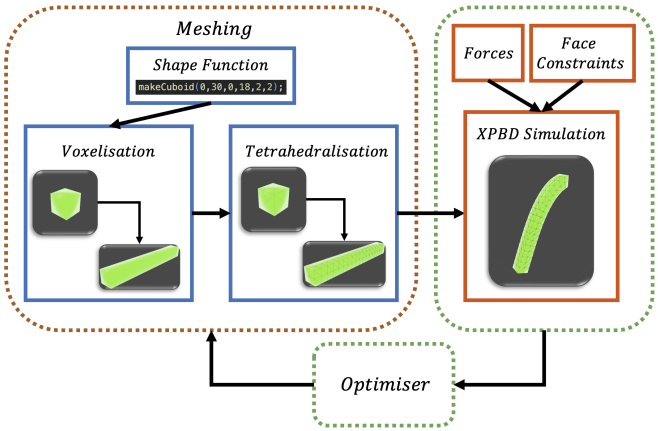

Fig. 1. Overview of the soft robot optimisation pipeline: Geometry is defined via a shape function, followed by voxelisation and tetrahedralisation. XPBD simulation applies forces and constraints, providing data for the optimiser to iteratively refine parameters, feeding back into meshing and simulation for optimal performance.

I. Introduction

Soft robotics has emerged as a rapidly growing field due to its significant advantages across numerous applications [1], [2]. Soft robots can navigate challenging environments [3]–[5], enhance safety in human-machine interactions [6], and achieve complex actuation with inherent simplicity [7]–[9]. These capabilities position soft robotics as a promising avenue for the future of robotic systems.

However, the design process for soft robots is highly iterative and requires multiple prototypes to be manufactured and tested [10]. To mitigate this, virtual methods have been increasingly employed to simulate and optimise soft robot performance prior to manufacturing [11], [12]. This approach accelerates the design process and reduces the number of iterations required to achieve high-performance designs. Current simulation approaches utilise commercial solvers like ANSYS, ALTAIR, ABAQUS, or COMSOL. These solvers employ Finite Element Analysis (FEA) techniques to solve a global stiffness matrix based on boundary conditions and input volumetric meshes, predicting the expected displacement of the soft robot or actuator.

Integrating these simulation tools into an optimisation loop necessitates coupling them with geometry generation, meshing, and optimisation algorithms. Typically, the original geometry is stored and modified in a parametric Boundary Representation (B-Rep) format. A meshing algorithm then converts the B-Rep geometry into a volumetric mesh, which is passed to the simulation engine for boundary condition application and simulation.

Most Computer-Aided Design (CAD) software, such as SOLIDWORKS, CATIA, and open-source kernels like Open Cascade Technology (OCCT), utilise the B-Rep format. Meshing is often handled within a separate software like GMSH, and the simulation is conducted in yet another tool. This fragmented pipeline poses challenges: creating a simulation workflow is complex and often brittle, especially with intricate geometries. Failures can occur during geometry updates within the B-Rep format or due to errors in meshing and boundary condition application.

Recent literature reflects efforts to optimise soft actuators and robots with these pipelines. For example, [13] optimised a bending soft actuator using genetic algorithms by integrating ABAQUS for geometry generation and simulation with MATLAB for optimisation. Similarly, [14] optimised a soft finger using a pipeline that integrates an OCCT backend for geometry generation, GMSH for meshing, SOFA [15] for simulation, and Optuna [16] for optimisation.

With all of these advancements, a common pattern emerges: optimisation pipelines rely on multiple software tools for geometry design, meshing, simulation, and optimisation. This approach introduces brittleness within the pipeline, as multiple geometry type conversions are required for each algorithm. This issue is especially pronounced when working with complex geometries. It also means the barrier

This work was supported by the Department of Mechanical Engineering, University College London, London, UK.

Syed Ismail Ahmad and Helge Wurdemann are with the Department of Mechanical Engineering, University College London, UK. ucemsia@ucl.ac.uk h.wurdemann@ucl.ac.uk

to entry for engineers looking to optimise their soft robot and actuator designs is high, as a pipeline must first be created [17]–[19].

Therefore, there is a pressing need to generate soft robot and actuator volumetric meshes, simulate, and optimise them in a cohesive manner within a single, robust environment. To effectively search this design space, fast simulation methods are essential due to the increased number of simulations required. Several frameworks aim to reduce computational costs in simulating soft robots. The SOFA framework integrates FEA with coarse grids to enable real-time control system interactions [20], [21]. For finer meshes, model order reduction techniques using proper orthogonal decomposition have been applied [22]. However, these require offline simulations tailored to specific shape types, making them infeasible for optimisation loops involving diverse geometries.

In computer graphics, Position-Based Dynamics (PBD) [23] offers a robust approach to deformable simulation by directly manipulating particle positions and enforcing physical constraints, contrasting traditional methods that solve for forces and accelerations. Extended Position-Based Dynamics (XPBD) [24] further improves this by making constraint stiffness time-step independent. While PBD and XPBD have been applied in surgical soft tissue simulation [25] and soft robot simulation with strain energy constraints and model order reduction [26], their potential for simulating soft robots and actuators remains underexplored.

We address these challenges by proposing an integrated environment that enables robust volumetric mesh generation and rapid simulation, thereby facilitating the optimisation of soft robots and actuators. Our contributions are:

1) **A direct uniform meshing algorithm**: We introduce a method to rapidly produce tetrahedral meshes from simple shape functions in a robust, unbreakable manner. This eliminates dependencies on complex B-Rep geometry and meshing software, enhancing reliability and simplifying the pipeline.
2) **An XPBD simulation framework**: We develop an XPBD-based simulator and material calibration process that leverages the uniform meshing algorithm to efficiently simulate soft robots. Its simplicity and efficiency make it suitable for optimisation loops.
3) **An integrated optimisation environment**: We provide an environment that seamlessly embeds optimisation algorithms for the design of soft actuators. This enables exploration of novel designs beyond traditional parametric limitations, unlocking new possibilities in soft robotics.

Link to repository:
https://anonymous.4open.science/r/VoxSoft

II. Direct Uniform Meshing Algorithm

We create meshes directly by leveraging voxelisation to define material presence at points in 3D space. This binary representation, where each point either contains material or not, eliminates geometry computation errors and simplifies material placement. To facilitate this process, we developed two libraries:

1) **Low-Level Voxel Positioning Library**: Allows for the placement of specific voxels within the 3D space.
2) **High-Level Implicit Shape Function Library**: Enables generation of primitive and periodic shapes to build complex soft actuators and robots.

The implicit shape function generates a 3D surface. We use this to determine whether a point lies inside or outside the shape. The low-level voxel positioning library then fills voxels within these surfaces to create a voxelised solid body (we denote a solid body as $\mathcal{B} \in A$, where A represents the set of all solid bodies, and non-solid shell bodies as $\mathcal{B} \in B$, where B represents the set of all shell bodies). Finally, for the mesh setup, a tetrahedralisation step is performed that converts the individual voxels into a tetrahedral mesh by splitting each voxel into five separate tetrahedra.

To ensure the simulation runs as a single object rather than a multitude of individual tetrahedralised voxels, the vertex coordinates are stitched together. This ensures adjacent voxels share the same vertices.

A. Voxel Positioning Based on an Implicit Shape Function

The shape function $s(\mathbf{x})$ is represented as an implicit function, which defines a surface as the set of points that satisfy a certain equation. It can be written for a three-dimensional space as:

$$s(x, y, z) = 0, \tag{1}$$

where $(x, y, z) \in \mathbb{R}^3$ are the coordinates of a point in 3D space. The surface of the shape is defined by the set of points for which this equation holds true.

A voxel grid is then used to subdivide the three-dimensional space into cubic elements. Each voxel represents a discrete sample point within the space. In the context of the shape function, each voxel in the grid stores the value of the shape function $s(x, y, z)$ at the center of that voxel.

Sampling and Processing Implicit Geometry

To sample and process the shape function in the voxel grid, the following steps are taken:

1) **Define the Voxel Grid:** Establish the dimensions of the grid, $N_x \times N_y \times N_z$, and the spacing between two adjacent voxels, also known as the scale factor μ.
2) **Evaluate the Shape Function:** For each voxel centered at (x_i, y_j, z_k), compute the value of the shape function $s(x_i, y_j, z_k)$.
3) **Thresholding:** Determine if the voxel is inside, outside, or on the surface of the shape by comparing the function value to 0 or a shell thickness ϵ for a shell object. Specifically, the voxel is considered to be:
   - **On the surface** if $s(x_i, y_j, z_k) = 0$.
   - **Inside the shape** if $s(x_i, y_j, z_k) < 0$.
   - **Outside the shape** if $s(x_i, y_j, z_k) > 0$.

   For a shell object (denoted as $\mathcal{B} \in B$), the classification is as follows:

- **Inside the shell** if $|s(x_i, y_j, z_k)| < \frac{\epsilon}{2}$.
- **Outside the shell** if $|s(x_i, y_j, z_k)| \geq \frac{\epsilon}{2}$.

Algorithm 1 details the process of sampling each point in a voxel grid to determine the classification of each voxel.

B. Tetrahedralisation of Positioned Voxels

Once the voxel grid has been created, each voxel is split into five tetrahedra. This is required because the XPBD volume constraint is formulated for tetrahedra, and this allows us to capture shear movement of the elements. The four outer tetrahedra are congruent, creating a final regular tetrahedron at the core.

To capture this geometry, four separate rectangular arrays are created:

1) A vertex position ID array $\mathbf{z}$, containing the vector positions of all the individual vertices in the mesh.
2) A tetrahedral array $\mathbf{f}$, containing the position IDs of vertices which make up each tetrahedron within the mesh, used for maintaining the volumetric constraints within the XPBD simulation.
3) An edge array $\mathbf{e}$, containing the position IDs of vertices which are connected to each other; this is used for maintaining the distance constraint between edges within the XPBD simulation.
4) A surface triangle array $\mathbf{s}$, containing the position IDs of vertices on the surface of a voxel that make a triangular plane; this is used to render the mesh.

For each one of the voxels, new vertex positions are added to the array based on the position of a specific voxel G_i. Similarly, the tetrahedral, edge, and surface triangle arrays are also updated.

The process of integrating a new voxel mesh into the overall mesh structure can be generalised as follows. Starting with the initial array C for a voxel at the origin $(i, j, k) = (0, 0, 0)$, we generate array components D for each new voxel. These components are then concatenated with C to create the updated array C':

$$C' = \begin{bmatrix} C \\ D \end{bmatrix}, \quad C = [a_{ij}]_{m \times n}, \quad D = [b_{ij}]_{p \times n}, \tag{2}$$

$$b_{pn} = \begin{cases} 8G_i a_{mn} & \text{if } \alpha = \mathbf{z}, \\ 5G_i a_{mn} & \text{if } \alpha = \mathbf{f}, \\ 18G_i a_{mn} & \text{if } \alpha = \mathbf{e}, \\ 16G_i a_{mn} & \text{if } \alpha = \mathbf{s}. \end{cases} \tag{3}$$

C. Mesh Vertex Stitching Algorithm

The tetrahedralisation step generates a mesh of individual tetrahedralised voxels. In order for the object to be simulated as a single body rather than multiple separate voxels, the mesh must be stitched together where voxels are adjacent to one another. To stitch the mesh together, we remove duplicate vertices from the geometry arrays. This is accomplished by identifying vertices that share the same position and merging them. This ensures that the mesh forms a continuous, connected structure. Algorithm 2 outlines the steps involved to do this in a computationally efficient manner.

By iteratively merging duplicate vertices and updating the geometry arrays, the algorithm stitches the mesh together, ensuring that adjacent voxels share vertices and the mesh forms a single, connected body suitable for the XPBD simulation.

III. Simulation and Material Calibration

Extended Position-Based Dynamics (XPBD) models an object as a series of particles connected by constraints, with each particle defined by its mass m_i, position $\mathbf{p}_i$, and velocity $\mathbf{v}_i$, in this case positioned at the vertices of a tetrahedral mesh. Edge distance constraints maintain fixed distances between particles, while volume constraints preserve the volume of each tetrahedron. XPBD iteratively adjusts particle positions to new positions $\mathbf{q}_i$ in order to satisfy these constraints, using semi-implicit time integration: particles move based on velocities and external forces

Algorithm 1 Implicit Shape Function Voxel Positioning

Input: Voxel grid G with dimensions $N_x \times N_y \times N_z$, shell thickness ϵ, scale factor μ
Output: Positioned voxels for the implicit shape function
for $i = 1$ to N_x **do**
  for $j = 1$ to N_y **do**
    for $k = 1$ to N_z **do**
      Compute $s(i, j, k)$ at voxel center $(\mu i, \mu j, \mu k)$
      if $\mathcal{B} \in B$ **then**
        if $|s(i, j, k)| < \frac{\epsilon}{2}$ **then**
          Position voxel at $(\mu i, \mu j, \mu k)$
        end if
      else if $s(i, j, k) < 0$ **then**
        Position voxel at $(\mu i, \mu j, \mu k)$
      end if
    end for
  end for
end for

Algorithm 2 Mesh Vertex Stitching

Input: Vertex array $\mathbf{z}$, tetrahedral array $\mathbf{f}$, edge array $\mathbf{e}$, surface triangle array $\mathbf{s}$
Output: Updated geometry arrays with merged vertices
Initialise an empty mapping for unique vertex positions
for each vertex index i in $\mathbf{z}$ **do**
  position$_i \leftarrow \mathbf{z}[i]$
  if *position*$_i$ is not in the mapping **then**
    Add *position*$_i$ to the mapping with index i
  else
    $j \leftarrow$ index of existing vertex at *position*$_i$
    Update indices in $\mathbf{f}$, $\mathbf{e}$, and $\mathbf{s}$: replace i with j
  end if
end for

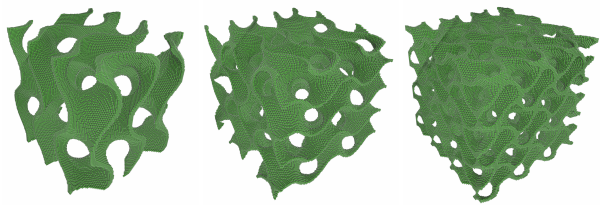

Fig. 2. A tetrahedral mesh generated for a gyroid structure to evaluate the robustness of voxelisation, meshing, and vertex stitching processes.

$\mathbf{f}_{\text{ext}}$, followed by constraint corrections, thereby enhancing stability, especially for stiff objects.

However, material compliance $\alpha_{(e,v)}$ within XPBD is a non-physical parameter affected by mesh density and time step size. Therefore for high-accuracy soft robotics applications, tuning the simulation with specific time steps and compliance values is essential to achieve precise results.

A. Material Calibration

To calibrate simulation parameters with real-world materials, a beam deflection test was devised. Beams made from three materials (EcoFlex 00-20, 00-30, and 00-50), each measuring 10 mm in width and height and 100 mm in length, were tested. In three separate experiments, the beams were clamped 10 mm from one end, leaving 90 mm exposed, and displacements due to gravity were recorded at 10 mm intervals along the length.

An XPBD simulation was run to model the experiment, with each voxel representing a 5x5x5mm volume of material. The simulation was run in tandem with the genetic optimisation given in Algorithm 4 that adjusted edge and volume compliance to align simulated displacements with real-world data. An exponentially decaying fitness function f was applied to minimise the error between simulated and actual displacements.

$$f = e^S \tag{4}$$

where:

$$S = k \sum_{i=0}^{8} \|\vec{r}i - \vec{r}i, \text{real}\| \tag{5}$$

$$\vec{r}i = (x_i, y_i, z_i), \quad \vec{r}i, \text{real} = (x_{i,\text{real}}, y_{i,\text{real}}, z_{i,\text{real}})$$

- $\vec{r}_i$ is the simulated displacement vector at position i.
- $\vec{r}_{i,\text{real}}$ is the real displacement vector at position i.
- k is the decay coefficient (-10).

1) Genetic Algorithm Implementation: The genetic algorithm evolves a population of candidate solutions, each represented by a DNA sequence of fixed length n.

The algorithm operates with a population size (N) of 30 and a DNA length (n) of 2. It selects 2 elite individuals (E) in each generation with a mutation rate (m) of 0.05. When running the genetic algorithm with function f the objective becomes to minimise the error between the simulated and real beam displacements. Thus, through this optimisation we encode the material compliance characteristics into the tetrahedralised voxel.

Using the genetic algorithm, we achieved average errors between simulation and reality for EcoFlex 00-50, 00-30, and 00-20 of 3.69%, 4.23%, and 6.71%, respectively. This corresponds to a maximum displacement error of 2.14mm, 3.26mm and 5.72mm for the beam across its length.

To analyse encoding sensitivity to voxel count and geometry, we conducted six experiments using Ecoflex 00-30 beams of varying sizes (shown in Figure 4). The first

Algorithm 3 Extended Position Based Dynamics Algorithm

for all particles i **do**
  initialise $\mathbf{p}_i = \mathbf{p}_i^0, \mathbf{v}_i = \mathbf{v}_i^0, w_i = 1/m_i$, $\alpha_{e,v} = \frac{1}{k_{e,v}}$
end for
loop
  for all particles i **do**
    $\mathbf{v}_i \leftarrow \mathbf{v}_i + \Delta t w_i \mathbf{f}_{\text{ext}}(\mathbf{p}_i, \mathbf{p}_N)$
  end for
  dampVelocities($\mathbf{v}_1, \ldots, \mathbf{v}_N$)
  for all particles i **do**
    $\mathbf{q}_i \leftarrow \mathbf{p}_i + \Delta t \mathbf{v}_i$
  end for
  for solverIterations times **do**
    projectConstraints($C_1, \ldots, C_N, \mathbf{q}_1, \ldots, \mathbf{q}_N, \alpha_{e,v}$)
  end for
  for all particles i **do**
    $\mathbf{v}_i \leftarrow (\mathbf{q}_i - \mathbf{p}_i)/\Delta t$
    $\mathbf{p}_i \leftarrow \mathbf{q}_i$
  end for
  velocityUpdate($\mathbf{v}_1, \ldots, \mathbf{v}_N$)
end loop

Algorithm 4 Genetic Algorithm

Initialise population Population with N random individuals
Set generation counter $G \leftarrow 1$
repeat
  for each individual DNA_i in Population **do**
    Compute fitness $f_i = {e_i}^S$
  end for
  Sort Population in descending order of fitness
  Copy top E elite individuals to New Population
  for $i = E + 1$ to N **do**
    Select parents DNA_a, DNA_b using roulette wheel selection approach
    Create offspring $\text{DNA}_{\text{child}}$ via crossover
    Mutate $\text{DNA}_{\text{child}}$ with mutation rate m
    Add $\text{DNA}_{\text{child}}$ to New Population
  end for
  Replace Population with New Population
  Increment generation $G \leftarrow G + 1$
until termination condition is met ($G = 50$)

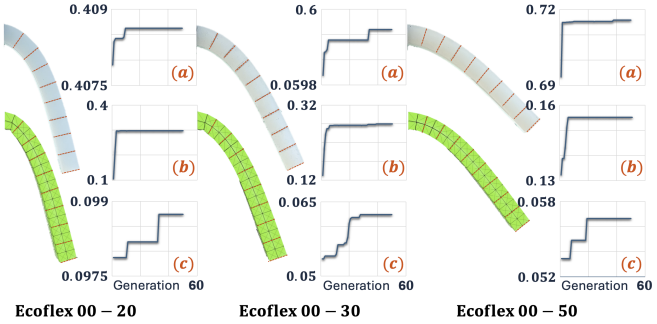

Fig. 3. Beam deflection under gravity within an XPBD simulation compared to real world experiment. (a) Shows the fitness function evolution, (b) the edge compliance evolution and (c) the volume compliance evolution for 50 generations of the Genetic Algorithm.

beam (a) retained the original dimensions (10×10×100mm), while the width of beam (b) and height of beam (c) were separately doubled to 20mm. These experiments were then repeated, but with the beam length halved to 50mm (d, e, f). Node displacement under gravity was recorded at 10mm increments. With a constant voxel scale factor μ, voxel count varied with geometry, but local mesh density and pattern remained unchanged. We also conducted all simulations with the optimised parameters given in Table I.

We found that doubling the voxel count in beams (b) and (c) increased the average error from 4.23% to 5.53% and 12.76%, respectively. Conversely, reducing the voxel count in beams (d) and (e) lowered the error to 3.68% and 2.41%. Notably, beam (f), like beam (c), where height was increased, showed a greater accuracy drop than width expansion, with an average error of 6.56%. Thereby indicating accuracy of an XPBD approach being linked to the force application direction in relation to the mesh.

IV. Optimising a Soft Actuator using the Integrated Environment

To evaluate the system as a whole, a cylindrical soft actuator was optimised to maximise displacement under fixed pressure (4 kPa) within specific geometric constraints using our integrated environment. The optimised values were then used to manufacture a real actuator in order to validate the effectiveness of the integrated environment.

A. Optimisation and Simulation Protocol

A parametric shape function characterised the cylindrical actuator, with variables for radius, height, and wall thickness. Geometric constraints set maximums for radius (5 voxels, each 5x5x5 mm), height (7 voxels, 35 mm), and wall thickness (3 voxels, 15 mm). A brute-force optimisation searched the design space, enabled by the discrete voxel-based geometry (105 combinations in 210s on an ARM64 Apple M3 Max processor) using EcoFlex 00-30 compliance values from Table I. The optimal configuration for maximum displacement was found at radius = 5 voxels, height = 7 voxels, and wall thickness = 3 voxels. A physical actuator with these dimensions was then produced and tested for validation.

TABLE I

Optimised XPBD simulation variables for different material types.

Material Type	Edge Compliance	Volume Compliance
Ecoflex 00-20	0.29939	0.01761
Ecoflex 00-30	0.27877	0.09300
Ecoflex 00-50	0.14575	0.05815

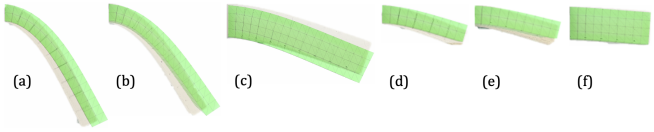

Fig. 4. Overlay of real and simulated displacements for the 6 beams tested.

B. Soft Actuator Manufacturing and Experimental Protocol

To manufacture the actuator, 3D printed moulds were created using an ELEGOO MARS 3 PRO printer and ELEGOO ABS-LIKE 2.0 RESIN (GREY). The actuator was split into two moulded sections, the first being the lower cap and actuator walls and the second being the upper cap. Two part EcoFlex 00-30 was then mixed by equal mass and vacuum degassed before being poured into each mould. Once cured the two sections were bonded using WACKER ELASTOSIL E41. Finally 3D printed upper and lower caps (ELEGOO ABS-LIKE 2.0 RESIN) were bonded to the actuator also using ELASTOSIL E41, with the top cap also including an air inlet port.

The actuator was then mounted to a table surface and pressurised air was introduced via the air inlet port. This was done with a HYUNDAI HY5508 compressor and a pressure regulator with an embedded closed loop controller. The displacement of the actuator was then measured using a vernier calliper. The experiment was repeated 10 times and the average displacement was then taken.

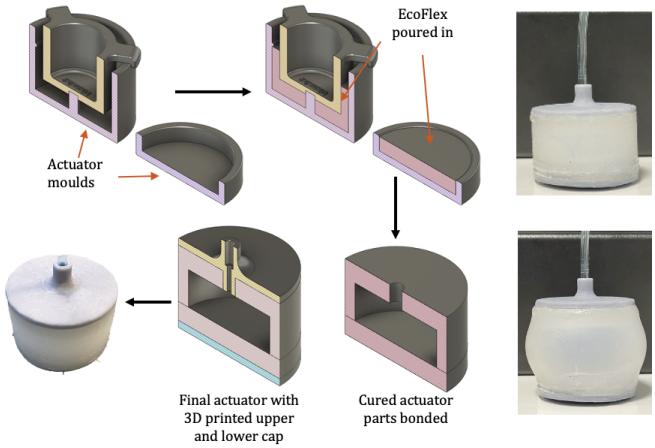

Fig. 5. Left: Manufacturing process of the soft actuator. Right: Actuator in pressurised and non-pressurised states.

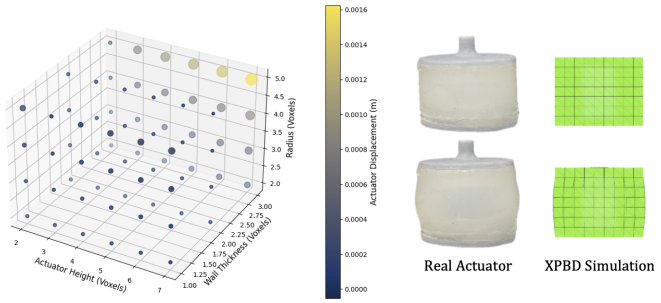

Fig. 6. 4D plot of cylindrical soft actuator optimisation, showing displacement as a function of radius, height, and wall thickness under 4 kPa. Colour denotes displacement magnitude, highlighting the optimal design. Right: XPBD simulation mesh and real-world test.

We found that the displacement prediction was 1.63mm and the real actuator displaced 2.3mm, thereby representing a 0.67mm discrepancy between the simulated and experimental displacements. As the possible displacement of an actuator of this type is relatively small, the discrepancy could be attributed to a number of factors such as manufacturing errors, the voxel definition not accurately capturing the geometry used and errors within the material calibration process. Further work is required to fully understand how best to ensure accuracy with an environment of this nature.

V. Conclusion

This paper presented an integrated environment for soft robotic design, combining geometry generation, voxel-based meshing, XPBD simulation, and parameter optimisation. Calibration against real-world beam bending tests demonstrated its feasibility, with reasonable accuracy across varying geometries. The optimisation and fabrication of a soft actuator further validated its effectiveness, despite minor discrepancies between simulated and experimental results. Future work will refine material calibration, enhance voxel representation of complex geometries, and extend the framework to more intricate actuators and soft robots. A comparative study with established FEA packages will also assess XPBD's computational efficiency and accuracy in soft robotic applications.

References

[1] D. Rus and M. T. Tolley, "Design, fabrication and control of soft robots," *Nature*, vol. 521, no. 7553, pp. 467–475, May 2015.

[2] C. Laschi, B. Mazzolai, and M. Cianchetti, "Soft robotics: Technologies and systems pushing the boundaries of robot abilities," *Sci. Robot.*, vol. 1, no. 1, p. eaah3690, 2016.

[3] R. F. Shepherd, F. Ilievski, W. Choi, S. A. Morin, A. A. Stokes, A. D. Mazzeo, X. Chen, M. Wang, and G. M. Whitesides, "Multigait soft robot," *Proceedings of the National Academy of Sciences*, vol. 108, no. 51, pp. 20 400–20 403, 2011.

[4] E. Almanzor, F. Ye, J. Shi, T. G. Thuruthel, H. A. Wurdemann, and F. Iida, "Static shape control of soft continuum robots using deep visual inverse kinematic models," *IEEE Transactions on Robotics*, vol. 39, no. 4, pp. 2973–2988, 2023.

[5] A. Ataka, P. Qi, A. Shiva, A. Shafti, H. Wurdemann, H. Liu, and K. Althoefer, "Real-time pose estimation and obstacle avoidance for multi-segment continuum manipulator in dynamic environments," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016, pp. 2827–2832.

[6] G. Agarwal, M. A. Robertson, H. Sonar, and J. Paik, "Design and computational modeling of a modular, compliant robotic assembly for human lumbar unit and spinal cord assistance," *Scientific Reports*, vol. 7, 12 2017.

[7] H. A. Wurdemann, A. Stilli, and K. Althoefer, "Lecture notes in computer science: An antagonistic actuation technique for simultaneous stiffness and position control," in *Intelligent Robotics and Applications*, H. Liu, N. Kubota, X. Zhu, and R. Dillmann, Eds. Cham: Springer International Publishing, 2015, pp. 164–174.

[8] F. Giorgio-Serchi, A. Arienti, and C. Laschi, "Underwater soft-bodied pulsed-jet thrusters: Actuator modeling and performance profiling," *International Journal of Robotics Research*, vol. 35, pp. 1395–1416, 9 2016.

[9] Q. Qi, Y. Teng, and X. Li, "Design and characteristic study of a pneumatically actuated earthworm-like soft robot," in *2015 International Conference on Fluid Power and Mechatronics (FPM)*, 2015, pp. 435–439.

[10] D. Trivedi, C. D. Rahn, W. M. Kier, and I. D. Walker, "Soft robotics: Biological inspiration, state of the art, and future research," *Applied Bionics and Biomechanics*, vol. 5, no. 3, p. 520417, 2008.

[11] O. Goury and C. Duriez, "Fast, generic, and reliable control and simulation of soft robots using model order reduction," *IEEE Transactions on Robotics*, vol. 34, no. 6, pp. 1565–1576, 2018.

[12] P. Polygerinos, Z. Wang, J. T. B. Overvelde, K. C. Galloway, R. J. Wood, K. Bertoldi, and C. J. Walsh, "Modeling of soft fiber-reinforced bending actuators," *IEEE Transactions on Robotics*, vol. 31, no. 3, pp. 778–789, 2015.

[13] G. Runge, J. Peters, and A. Raatz, "Design optimization of soft pneumatic actuators using genetic algorithms," in *IEEE International Conference on Robotics and Biomimetics*, 2017, pp. 393–400.

[14] S. E. Navarro, T. Navez, O. Goury, L. Molina, and C. Duriez, "An open source design optimization toolbox evaluated on a soft finger," pp. 6044–6051, 2023.

[15] F. Faure, C. Duriez, H. Delingette, J. Allard, B. Gilles, S. Marchesseau, H. Talbot, H. Courtecuisse, G. Bousquet, I. Peterlik *et al.*, "Sofa: A multi-model framework for interactive physical simulation," in *Soft tissue biomechanical modeling for computer assisted surgery*. Springer, 2012, pp. 283–321.

[16] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," in *The 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019, pp. 2623–2631.

[17] D. R. Ellis, M. P. Venter, and G. Venter, "Computational design for inflated shape of a modular soft robotic actuator," in *IEEE International Conference on Soft Robotics*, 2019, pp. 7–12.

[18] *Evolution of morphology through sculpting in a voxel based robot*, ser. Proceedings of the 2023 Artificial Life Conference, 2021.

[19] M. P. Venter and I. J. Joubert, "Generative design of soft robot actuators using esp," *Mathematical and Computational Applications*, vol. 28, p. 53, 4 2023.

[20] C. Duriez, "Control of elastic soft robots based on real-time finite element method," in *IEEE international conference on robotics and automation*. IEEE, 2013, pp. 3982–3987.

[21] P. Chaillou, J. Shi, A. Kruszewski, I. Fournier, H. A. Wurdemann, and C. Duriez, "Reduced finite element modelling and closed-loop control of pneumatic-driven soft continuum robots," in *2023 IEEE International Conference on Soft Robotics (RoboSoft)*, 2023, pp. 1–8.

[22] O. Goury and C. Duriez, "Fast, generic, and reliable control and simulation of soft robots using model order reduction," *IEEE Transactions on Robotics*, vol. 34, no. 6, pp. 1565–1576, 2018.

[23] M. Müller, B. Heidelberger, M. Hennix, and J. Ratcliff, "Position based dynamics," *J. Vis. Comun. Image Represent.*, vol. 18, no. 2, p. 109–118, 2007.

[24] M. Macklin, M. Müller, and N. Chentanez, "Xpbd: position-based simulation of compliant constrained dynamics," in *International Conference on Motion in Games*, ser. MIG '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 49–54.

[25] M. Camara, E. Mayer, A. Darzi, and P. Pratt, "Soft tissue deformation for surgical simulation: a position-based dynamics approach," *International Journal of Computer Assisted Radiology and Surgery*, vol. 11, no. 6, pp. 919–928, 2016.

[26] H. Peng, N. Li, D. Jiang, and F. Li, "Soft robot fast simulation via reduced order extended position based dynamics," *Robotics and Autonomous Systems*, vol. 175, p. 104650, 2024.