TorsioSquid: A squid-inspired underwater bot

Naman Khetan and Krishna Chaitanya Myneni

IIT (BHU) Varanasi

Abstract

The TorsioSquid is a bio-inspired robotic system that mimics the propulsion of squids through innovative shape-morphing, combining compactness and lightweight design with an efficient mass ejection mechanism. Its unique propulsion is achieved through a synergy of torsional buckling [1] and Bowden cables. **Weighing 800 g** and standing **540 mm tall**, TorsioSquid achieves a notable **mass ejection rate of 0.4 kg/sec**, contributing to significant thrust force.

Introduction

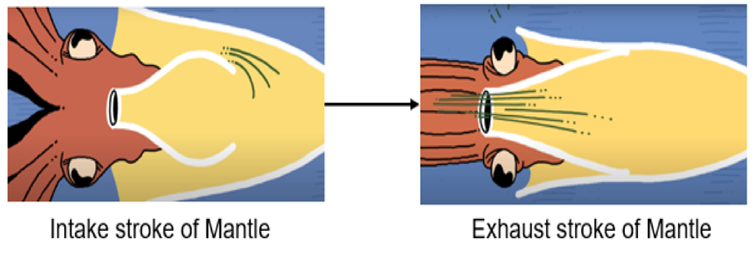
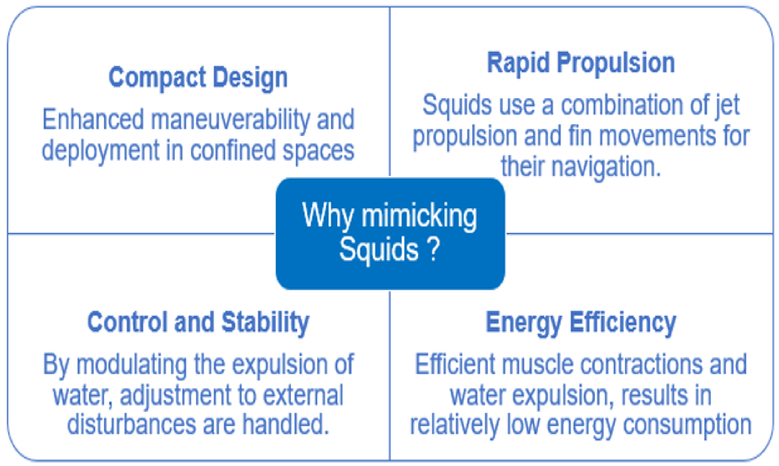


Figure 1. Working of mantle

Underwater soft robotics is an emerging field due to the increased need for underwater exploration and surveillance, which requires blending with natural habitats. Although previous work has been done mimicking fishes and octopuses, only a few have tried to imitate the mantle movement of squid. Mimicking the mantle's propulsion mechanism could lead to a compact design, which we have presented in this paper. In previous similar works, rigid components like motors and gears have been used [2], often failing in cyclic loads. Thus, our design prioritizes compactness, lightweight construction, and softer components.



References

1. Harnessing Mechanical Instabilities in the Development of an Efficient Soft Pump for an Artificial Heart Ventricle Simulator. IEEE/ASME Transactions on Mechatronics. PP. 1-11. 10.1109/TMECH.2023.3345529.
2. Caleb Christianson et al. 2020 Bioinspir. Biomim. 16 016014

Methodology

Key to TorsioSquid's actuation is a unique combination of torsional buckling and Bowden cables. Bowden cables arranged helically around the shell have one end connected to the linear actuator and the other fixed to the apex. As the actuator retracts, it pulls on the cables, inducing axial and tangential forces, causing controlled buckling of the shell. This mimics the squid's mantle movement, generating thrust for propulsion.

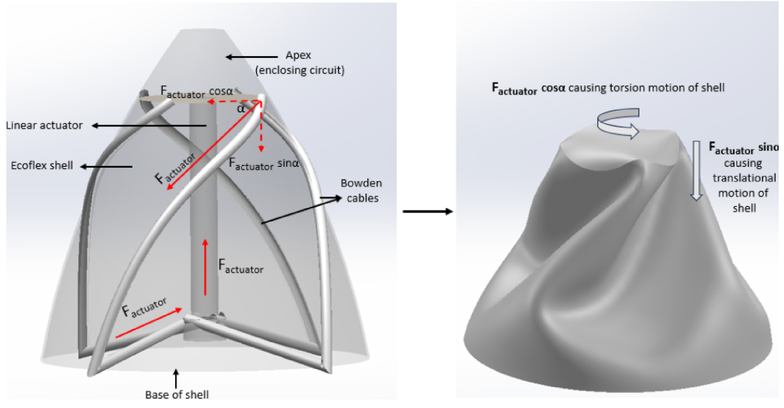


Figure 2. FBD and working of the shell during backstroke of linear actuator

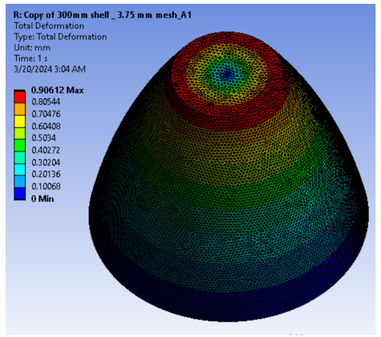
Equations and Boundary conditions

1. Thrust force = Exit velocity × Mass ejection rate
2. Mass ejection rate = Density of water × $\widetilde{V}$
3. Translation load [1] = $(A + w) \cdot (\sin\alpha_f - \sin\alpha_0) = 37$ mm
4. Rotational load [1] = (Helix parameter) $\cdot (\alpha_f - \alpha_0) = 45$ degrees

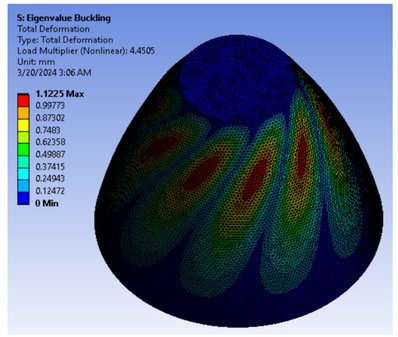
Where

$\widetilde{V}$ = Volumetric deformation
A = Major axis = 298 mm
w = Shell thickness = 2 mm
α_f = Final helix angle = 56.25 degrees
α_0 = Initial helix angle = 45 degrees
Helix parameter = $4 \times n = 4$
n = number of helix turns around the semielliptical surface.

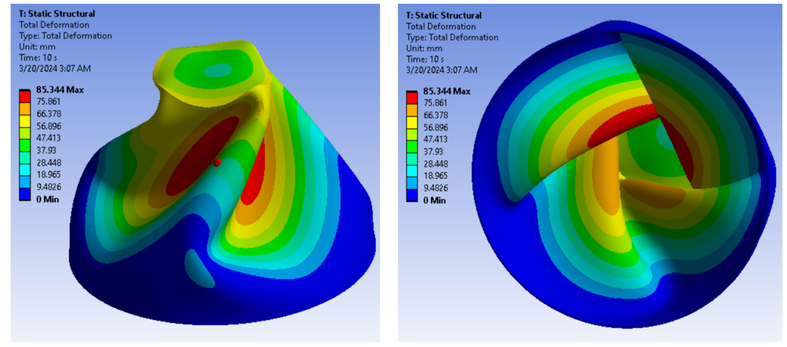
Simulation and Analysis



Analysis A1



Analysis A2



Analysis A3

Figure 3. Nonlinear buckling and eigenvalue buckling to find ΔV

- **Location of boundary conditions** - The base of the shell has been kept fixed. For simplicity, the apex shell has been removed, and the translational and rotational loads have been applied at the enclosing disc of the apex from where the shaft of the linear actuator is mounted.
- **Analysis A1**: 1-degree rotation has been applied to give Pre-stress to the shell.
- **Analysis A2**: Eigen buckling analysis to create geometrical imperfections in the shell.
- **Analysis A3**: Translational and rotational load are applied in steps of t = 10 secs on the geometrically imperfect shell to generate folds in the semi-ellipsoidal shape, replicating the volumetric deformed state
- **Obtaining volumetric reduction** - Importing the volumetric deformed state in Analysis A3 of the shell in Solidworks and comparing it with the initial one gives us the volumetric deformation Δ V

Usage of the nozzle to increase the exit velocity

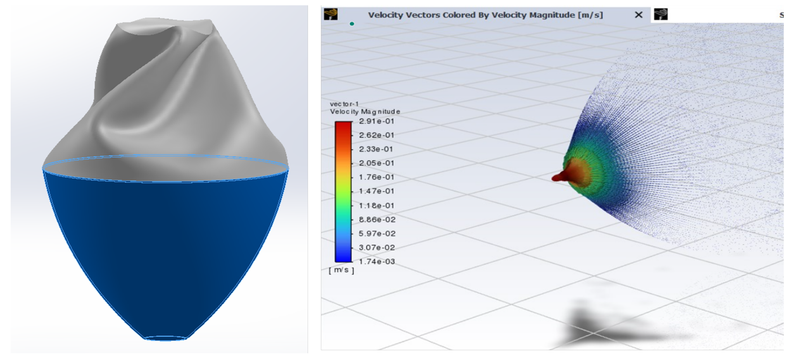


Figure 4. A semi-ellipsoidal-shaped nozzle with an exit diameter of 50 mm can increase the exit velocity, thus increasing the thrust force.

Results and Conclusion

Δ V (cm^3)	Mass ejection rate (Kg/sec)	Exit Velocity (cm/sec)	Thrust Force (N)	Acceleration (ms^{-2})
4077.032	0.4077	20.76	0.0846	0.105

We calculate the mesh convergence error to verify our simulation results. We get a **0.7 % mesh convergence error**, which gives our simulation results reasonable accuracy.