Learning to Optimize Shapes

Meta-Optimization and Implicit Neural Fields

Arselane Hadj Slimane · **Grégoire Allaire**

Centre de Mathématiques Appliquées (CMAP), CNRS

École Polytechnique, Institut Polytechnique de Paris, Palaiseau, France

arselane.hadj-slimane@polytechnique.edu — gregoire.allaire@polytechnique.edu

1. Shape Optimization Problem

Formulation

Shape optimization follows the standard domain-based setting [1, 2].

$$\min_{\Omega \in \mathcal{U}_{ad}} J(\Omega) = \mathcal{J}(\Omega, u(\Omega))$$

subject to the PDE constraint $\mathcal{A}_\Omega(u) = f$ in Ω, with admissible set:

$$\mathcal{U}_{ad} := \{\Omega : \Omega = (\mathrm{Id} + \theta)(\Omega_0),\ \theta \in \Theta_{ad}\}$$

$\Theta_{ad} \subset W^{1,\infty}(\mathbb{R}^d, \mathbb{R}^d)$ — topology-preserving deformations.

Key features:

- Optimization variable is the *domain* Ω
- Constraints are PDEs posed on Ω
- Highly nonlinear and infinite-dimensional

Shape Derivative

Directional derivative of J at Ω in direction θ:

$$dJ(\Omega)(\theta) = \lim_{\tau \to 0} \frac{J((\mathrm{Id} + \tau\theta)(\Omega)) - J(\Omega)}{\tau}$$

Structure theorem: depends only on $\theta \cdot n$ on $\partial\Omega$.

Riesz representation: there exists $\nabla J(\Omega) \in H^1(\Omega)^d$ such that $dJ(\Omega)(\theta) = (\nabla J(\Omega), \theta)_{H^1}$.

Shape deformation

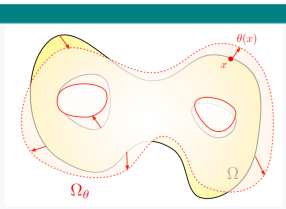


Figure: Shape deformation via the flow of a vector field θ

Level Set Method

Implicit shape representation on a reference box $D \subset \mathbb{R}^d$:

$$\Omega = \{x \in D : \phi(x) < 0\}, \quad \phi(x) = \begin{cases} -\mathrm{dist}(x, \partial\Omega) & x \in \Omega \\ +\mathrm{dist}(x, \partial\Omega) & x \notin \Omega \end{cases}$$

Shape evolution via the Hamilton–Jacobi equation:

$$\partial_t \phi + v|\nabla\phi| = 0, \quad v = \nabla J(\Omega) \cdot n_\Omega$$

2. Implicit Neural Fields for Shape Representation

Implicit Neural Field and VAE Learning

An **implicit neural field (INF)** represents a shape as the zero level set of a continuous function:

$$f_\theta : \mathbb{R}^2 \times \mathbb{R}^d \to \mathbb{R}, \qquad \Omega(z) = \{(x, y) \mid f_\theta((x, y), z) < 0\}.$$

Here, $z \in \mathbb{R}^d$ is a latent vector encoding the shape, and f_θ is trained to approximate the signed distance function (SDF) [4].

To find the latent vectors we consider a Variational Auto-Encoder (VAE) framework where an encoder E_ϕ maps SDF samples on a grid to a latent distribution:

$$\{((x_j, y_j), s_j)\}_{j=1}^M \xrightarrow{E_\phi} (\mu(S), \log\sigma^2(S)) \in \mathbb{R}^d \times \mathbb{R}^d$$

with latent code sampled via the reparameterization trick:

$$z(S) = \mu(S) + \sigma(S) \odot \varepsilon, \qquad \varepsilon \sim \mathcal{N}(0, I).$$

The training objective is the negative ELBO:

$$\mathcal{L}(\theta, \phi) = \underbrace{\frac{1}{MN} \sum_{i,j} |\hat{s}_{ij} - s_{ij}|^2}_{\text{Reconstruction}} + \underbrace{\frac{\beta}{N} \sum_i D_{\mathrm{KL}}(q_\phi(z_i \mid S_i) \,\|\, \mathcal{N}(0, I))}_{\text{KL regularization}}$$

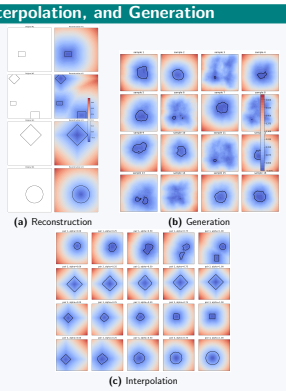
with the closed-form Gaussian KL divergence:

$$D_{\mathrm{KL}} = \tfrac{1}{2} \textstyle\sum_{k=1}^d (\mu_k^2 + \sigma_k^2 - \log\sigma_k^2 - 1)\,.$$

β controls the fidelity versus latent-space regularity trade-off.

The learned latent space provides a compact parametrization of complex geometries, in a significantly lower dimension than the full level-set. Thus, the latent code z offers a new, resolutions-free representation of the shapes. The geomtery of the latent space can also be controlled or designed during training to either encourage a certain behavior or act as a learned preconditionner. In practice, so far, we regularize z with simple priors (e.g., ℓ_2) and augment training with noise to improve robustness to out-of-distribution initializations.

Examples: Reconstruction, Interpolation, and Generation



(a) Reconstruction **(b)** Generation

(c) Interpolation

Figure: Examples of shape encoding/decoding in the learned latent space

Latent Reduced-Order Model

The INF induces a **nonlinear reduced-order model** for shapes. Shape optimization reformulated as:

$$\min_{z \in \mathbb{R}^d} J(f_\theta(\cdot; z))$$

Advantages:

- Low-dimensional latent space instead of infinite-dimensional shape space
- Implicit geometric regularization from the decoder
- Smooth z-paths $\Rightarrow$ smooth shape deformations
- Can learn a Riemannian metric capturing shape-relevant variations

This approach can be used to leverage the generalization capabilities of neural network and generative models to explore new solutions for shape optimization problems, condition the search space and, possibly, learn topology changes that happen in level-set methods for shape optimization.

3. Meta Shape Optimization

General Meta-Optimization Framework

Inner optimization with learned update rule Ψ_ϕ:

$$\rho_{t+1} = \rho_t + \Psi_\phi(\nabla J(\rho_t)), \qquad t = 0, \dots, T-1$$

This learned-update perspective follows the meta-optimization literature [3].

Unrolled meta-objective over a distribution $\mathcal{D}$ of problems:

$$\min_\phi \ \mathcal{L}(\phi) = \mathbb{E}_{\xi \sim \mathcal{D}} \left[\sum_{t=0}^T w_t\, J_\xi(\rho_t) \right]$$

Note: ρ_t depends on ϕ through the unrolled iterations.

Target Problem: Density Optimization

$$\min_{\rho \in \mathcal{U}_{ad}} \ J(\rho) = \tfrac{1}{2} \textstyle\int_\Omega |u - u_0|^2$$
$$\text{s.t.} \quad -\nabla \cdot (\rho \nabla u) + \alpha u = f, \quad \rho\, \partial_n u|_{\partial\Omega} = g$$

Projected gradient iteration with $\mathcal{U}_{ad} = \{\rho_{min} \le \rho \le \rho_{max}\}$:

$$\rho_{t+1} = \mathrm{proj}_{\mathcal{U}_{ad}}(\rho_t - \gamma\, \delta\rho_t), \qquad J'(\rho_t) \cdot w = \int_\Omega w\, \nabla u_t \cdot \nabla p_t$$

Learned Anisotropic Regularization

Classical H^1 regularization:

$$-\epsilon^2 \Delta \delta\rho_t + \delta\rho_t = J'(\rho_t)$$

Proposed learned anisotropic regularization:

$$-\epsilon^2 \nabla \cdot \big(H(\rho_t) \nabla \delta\rho_t\big) + \delta\rho_t = J'(\rho_t)$$

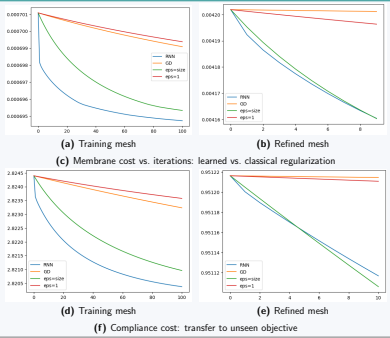
$H(\rho_t)$ positive definite and data-driven: **learned local preconditioning**.

Local-by-triangle neural architecture:

$$H(\rho_t)|_\tau = \mathrm{RNN}\big((\rho_t, u_t, p_t, J'(\rho_t))|_\tau,\ \phi\big)$$

The same network is shared by all mesh triangles, making the learned regularization parameter scalable across meshes. In practice we parameterize H to guarantee positive-definiteness and train the local network end-to-end.

Results



(a) Training mesh **(b)** Refined mesh

(c) Membrane cost vs. iterations: learned vs. classical regularization

(d) Training mesh **(e)** Refined mesh

(f) Compliance cost: transfer to unseen objective

Geometric Meta Shape Optimization

For geometric shape optimization, the shape derivative has a boundary structure:

$$J'(\Omega)[\theta n_\Omega] = \int_{\partial\Omega} g(u, p)\, \theta, \qquad \theta \in \Theta_{ad}$$

Classical H^1 regularization as a convolution with a Green kernel:

$$\nabla J(\Omega) = K_\epsilon * g(u, p)$$

A CNN generalizes this with a learned compact kernel:

$$\nabla J(\Omega) = \mathrm{CNN}_\phi(g(u, p))$$

Training meta-objective (unrolled full shape optimization):

$$\min_\phi \ \mathbb{E}_{\Omega_0, \mathrm{PDE}} \left[\sum_{t=0}^T w_t J(\Omega_t) \right], \quad \Omega_{t+1} = \big(\mathrm{id} + \tau_t\, \mathrm{CNN}_\phi(g(u_t, p_t))\big)(\Omega_t)$$

The learned CNN update generalizes classical smoothing kernels. We can also consider to train this regularizer to project the updates into geometric constraints while accelerating convergence. Provided the training set is diverse enough, the learned regularizer could generalize across different PDE constraints and objective functions, reducing the need for hand-tuning and improving robustness.

Computational Challenges

- No direct GPU–CPU communication pipeline
- FEM assembly in PyTorch is expensive and not highly optimized
- Full shape optimization loop required for each objective evaluation
- Structural mismatch: *local* predictor vs. *global* objective

References

G. Allaire. *Conception optimale de structures*. Springer, 2007.

G. Allaire, F. Jouve, and A.-M. Toader. Structural optimization using sensitivity analysis and a level-set method. *J. Comput. Phys.*, 194(1):363–393, 2004.

T. Chen, X. Chen, W. Chen, H. Heaton, J. Liu, Z. Wang, and W. Yin. Learning to optimize: A primer and a benchmark. *JMLR*, 23(189):1–59, 2022.

J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove. DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation. *CVPR*, 2019.