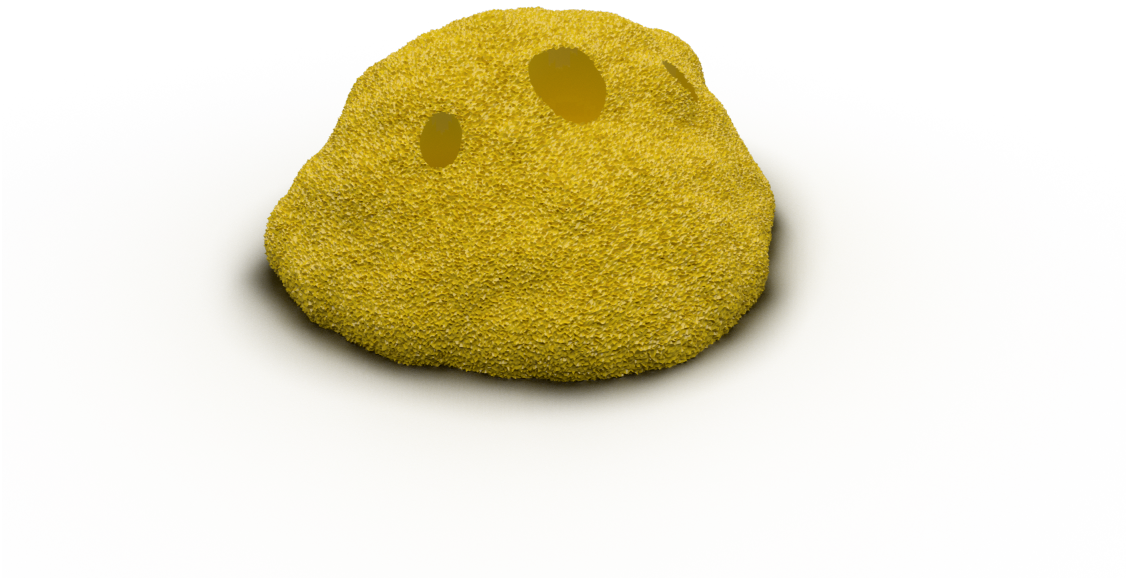
CHALMERS



Generative Design and Topological Optimization of Climbing Holds

Development of a Climbing Hold Generation Pipeline

Bachelor's Thesis in Electrical Engineering

DEPARTMENT OF ELECTRICAL ENGINEERING

CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2026
www.chalmers.se

Bachelor's Thesis 2026

Generative Design and Topological Optimization of Climbing Holds

Development of a Climbing Hold Generation Pipeline

Linus Andersson

Karin Cognell

Artur Rekstad

Klara Strandberg

Thea Söderstjerna

Adrian Warg

CHALMERS

Department of Electrical Engineering
Division of System and Control Engineering
EENX16-26-12
Chalmers University of Technology
Gothenburg, Sweden 2026

Generative Design and Topological Optimization of Climbing Holds
Development of a Climbing Hold Generation Pipeline
Linus Andersson, Karin Cognell, Artur Rekstad, Klara Strandberg, Thea Söderstjerna & Adrian Warg

Supervisors: Rikard Karlsson, Systems and Control Engineering
Sabino Francesco Roselli, System and Control Engineering
Examiner: Petter Falkman, System and Control Engineering

Bachelor's thesis 2026
Department of Electrical Engineering
Division of System and Control Engineering
EENX16-26-12
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Climbing hold geometry generated by the proposed pipeline, visualised as a rendered STL mesh.

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2026

Generative Design and Topological Optimization of Climbing Holds
Development of a Climbing Hold Generation Pipeline
Linus Andersson, Karin Cognell, Artur Rekstad, Klara Strandberg, Thea Söderstjerna & Adrian Warg Department of Electrical engineering
Chalmers University of Technology

Abstract

The design and development of climbing holds has traditionally required significant time and hands-on effort, relying heavily on manual modeling, iterative refinement and expert knowledge. This thesis primarily investigates whether generative design, topology optimization and additive manufacturing methods can be effectively applied to the development of climbing holds. A secondary objective was to use these findings to establish a structured pipeline for climbing holds generation.
A dataset of 3D-scanned climbing holds was created, representing predefined hold categories to enable unsupervised learning of geometric features. Two independent generative methods were developed and evaluated to assess their capability to generate novel within-category climbing hold geometries. These methods produced new hold designs as point clouds, which were subsequently reconstructed as CAD models for further refinement. The generated designs were topology optimized to improve material efficiency while maintaining structural integrity. Additional CAD refinement was performed to ensure manufacturability, validate tolerances and apply final surface textures.
The results demonstrate that these design and manufacturing methods can be successfully adapted for climbing hold development, while also highlighting their potential to reduce many of the limitations associated with traditional design practices. In addition, this thesis establishes a functional pipeline that can serve as a foundation for future development in AI-assisted climbing hold design. This research contributes to the broader field of AI-assisted product development by demonstrating the feasibility of combining machine learning with engineering optimization in a specialized design context. Future work should focus on expanding the dataset, improving generative precision and exploring the commercial viability of the methodology.

Keywords: Generative design, topology optimization, climbing holds, additive manufacturing, computational design, parametric modeling, finite element analysis, product development.

Acknowledgements

We would like to thank our supervisors, Rikard Karlsson and Sabino Francesco Roselli at the division of system and control engineering, for their guidance and support throughout the project. We would also like to thank our examiner Petter Falkman for his valuable feedback.

We would like to extend our gratitude to Klätterlabbet for providing access to their climbing holds for 3D scanning and the route setters who participated in our interviews and shared their professional insight on climbing hold design.

Lastly we would also like to thank CASE-labbet for the access of their tools such as filament 3D printers, SLS printer and 3D scanner.

Linus Andersson, Karin Cognell, Artur Rekstad
Klara Strandberg, Thea Söderstjerna, Adrian Warg
Gothenburg, May 2026

List of Acronyms

Below is the list of acronyms that have been used throughout this thesis listed in alphabetical order:

AE	Autoencoder
AM	Additive Manufacturing
ANN	Artificial Neural Network
CAD	Computer Aided Design
CNF	Continuous Normalizing Flow
EMD	Earth Mover's Distance
FAISS	Facebook AI Similarity Search
FEA	Finite Element Analysis
FDM	Fused Deposition Modeling
FPS	Farthest Point Sampling
GUI	Graphical User Interface
IDW	Inverse-Distance Weightning
KL	Kullback–Leibler
k-NN	k-Nearest-Neighbour
LRL	Latent Representation Learning
LLM	Large Language Model
ML	Machine Learning
MLP	Multilayer Perceptron
ReLU	Rectified Linear Unit
SA	Set Abstraction
SDG	Sustainable Development Goals
SIMP	Solid Isotropic Material with Penalization
SLS	Selective Laser Sintering
TO	Topological Optimization
VAE	Variational Autoenconder
WSL	Windows Subsystem for Linux

Nomenclature

Indices

e	Element index in topology optimization
i, j	Point indices in point cloud sets
k	Number of nearest neighbours in similarity search

Sets

A, B	Point sets used in distance metric definitions
X	Input point set
X_d	Reconstructed point set

Parameters

A	Load-bearing cross-sectional area of climbing hold
B	Batch size in neural network training
E_0	Elastic modulus of solid material
F_{design}	Required static design load
$F_{\text{design,dynamic}}$	Required dynamic design load
FoS	Factor of safety used in load calculations
N	Number of points in a point cloud
p	Penalization factor in the SIMP method
r	Radius used in PointNet++ ball query grouping
$V_{\max}$	Prescribed maximum material volume
v_e	Volume of element e
β	Weighting factor for the KL-divergence term in VAE training
ϵ	Small numerical constant used for stability in inverse-distance weighting
σ_{UTS}	Ultimate tensile strength of the material
σ_s	Yield strength of the material
θ	Trainable neural network parameters

Variables

b	Bias vector
b_1, b_2	Bias vectors in encoder and decoder layers
d_i	Distance associated with neighbour i in inverse-distance weighting
d_e	Element nodal displacement vector
f	Global nodal force vector
f_e	Element nodal force vector
F_{ultimate}	Ultimate load capacity
F_{dynamic}	Estimated dynamic load
k_e	Element stiffness matrix
u	Global nodal displacement vector
u_i	Two-dimensional grid point used in the folding decoder
v_i	Latent vector of neighbour i
v_{blend}	Blended latent vector
w_i	Inverse-distance interpolation weight for neighbour i
W	Weight matrix
W_1, W_2	Weight matrices in encoder and decoder layers
x_i	Point i in input point set X
x_j	Point j in reconstructed point set X_d
z	Latent vector
z_{cond}	Conditioned latent vector
ϵ	Random noise vector sampled from $\mathcal{N}(0, I)$
μ	Mean vector of the latent distribution
ρ_e	Density variable of element e, where $0 \leq \rho_e \leq 1$
σ	Standard deviation vector of the latent distribution
σ_e	Equivalent stress
σ_{VM}	von Mises equivalent stress
$\sigma_1, \sigma_2, \sigma_3$	Principal stresses
ε_x	Normal strain in the x-direction

Functions

$f(\cdot)$	Nonlinear activation function used in the encoder
$g(\cdot)$	Activation function used in the decoder
$\text{ReLU}(x)$	Rectified linear unit activation function, $\max(0, x)$
$J_{AE}(\theta)$	Autoencoder loss function
$L(X, X_d)$	Reconstruction loss between point sets X and X_d
$CH(A, B)$	Directed Chamfer distance from point set A to point set B
$d_{CH}(A, B)$	Symmetric Chamfer distance between point sets A and B
$d_X(\cdot, \cdot)$	Underlying distance metric
EMD	Earth Mover's Distance
$K(\rho)$	Stiffness matrix as a function of density ρ
$J(\rho)$	Compliance objective function
$E_e(\rho_e)$	Element elastic modulus as a function of density ρ_e

Definitions

User	The operator of the pipeline, namely climbing hold manufacturers and route setters who generate new hold designs
External user	Climbers at Klätterlabbet
Anchor	Dataset hold selected as the reference geometry
Grip region	Region of the hold surface where external load is applied

Contents

List of Figures

List of Tables

1
Introduction

Indoor climbing is a sport in which climbing routes of varying difficulty and technical demands are created using climbing holds. Through texture and shape variations, holds can emulate features of natural rock. However, modern indoor climbing has evolved into its own discipline, where hold design often introduces movements and interactions that differ significantly from outdoor climbing.

Traditional climbing holds are typically produced through manual sculpting processes, often involving hand-shaping foam prototypes [1, 2]. Although this approach offers creative flexibility, design research suggests that manual design processes may be susceptible to design fixation, potentially limiting geometric diversity over time [3, 4]. An interview conducted with a former routesetter indicated that hold designs often follow recurring patterns and established geometries, suggesting the presence of design fixation within traditional climbing hold development processes [5]. Furthermore, the manual sculpting workflow is labor-intensive and may involve material waste during prototyping and mold fabrication, which can limit production efficiency and scalability [1, 6].

Modern digital manufacturing and design methods, such as additive manufacturing [7] and machine-learning-based generative design [8], offer opportunities to explore more complex and varied hold geometries. Despite these technological advancements, the application of generative design methods within the climbing hold industry remains largely unexplored. In particular, it is not yet known whether generative design, topology optimization, and additive manufacturing can be meaningfully integrated into a structured pipeline for climbing hold production.

Although these methods have not yet been widely applied in climbing hold design, related work exists that explores individual components of the approach. For example, procedural point cloud and mesh editing techniques have been applied in urban planning, where Blender is used to convert point clouds into physical models [9]. In addition, shape generation through prompt-based or machine learning-driven systems have been extensively studied and applied in various product design projects [7, 10]. Machine learning-driven approaches have similarly proven valuable in manufacturing contexts, both for reducing time and labor cost in product development [6] and for the optimization and generation of new design solutions [11].

In high-stakes industries that exhibit limitations in design variation, such as aerospace and automotive engineering, generative design has been used to break the recurring

design pattern [12, 13]. Furthermore, generative design and topology optimization have been shown to reduce material consumption, thereby lowering CO_2 emissions in engineering and building applications [14]. This demonstrates the potential of the individual methods in creating complex and diverse geometries, yet whether they can be integrated into a coherent pipeline for climbing hold production remains an open question.

1.1 Purpose

This project investigates the applicability of generative design, topology optimization and additive manufacturing as methods for climbing hold development. The primary aim is to examine whether these methods can produce geometrically diverse and structurally valid climbing hold geometries and whether topology optimization can reduce material usage while maintaining sufficient load-bearing capacity. Geometric diversity is quantified using normalized Chamfer distance. This was evaluated through a distribution curve and anchor drift. Anchor drift defined as the difference in Chamfer similarity between the generated hold´s nearest neighbor in the full dataset and its designated anchor. Material efficiency is assessed through volumetric reduction following optimization. Manufacturing is carried out using selective laser sintering (SLS) with PA12 nylon powder. As a secondary objective, the project explores the extent to which these methods can be integrated into a cohesive pipeline for climbing hold production, positioning generative AI as a supportive tool that augments rather than replaces traditional design processes.

1.2 Limitations

1. No standardized practical testing will be conducted on the holds, instead structural validation will be performed using simulations and simplified models.
2. No large dataset will be created during this project, instead a smaller one will be used as a proof of concept.

1.3 Research questions

1. To what extent do the generated climbing holds exhibit categorically distinguishable characteristics across hold categories in external user testing?
2. How well do the generated climbing holds satisfy the defined structural load requirements?
3. To what degree does a Retrieval-based generative methodology produce geometrically novel outputs, as measured by Chamfer distance between each generated hold and its designated anchor?
4. To what extent does a Latent Representation Learning methodology produce geometrically novel outputs, as measured by Chamfer distance between each generated hold and its nearest neighbor in the dataset?

5. Can a pipeline be created by integrating generative design, topology optimization and additive manufacturing to automate the production of climbing holds?

2

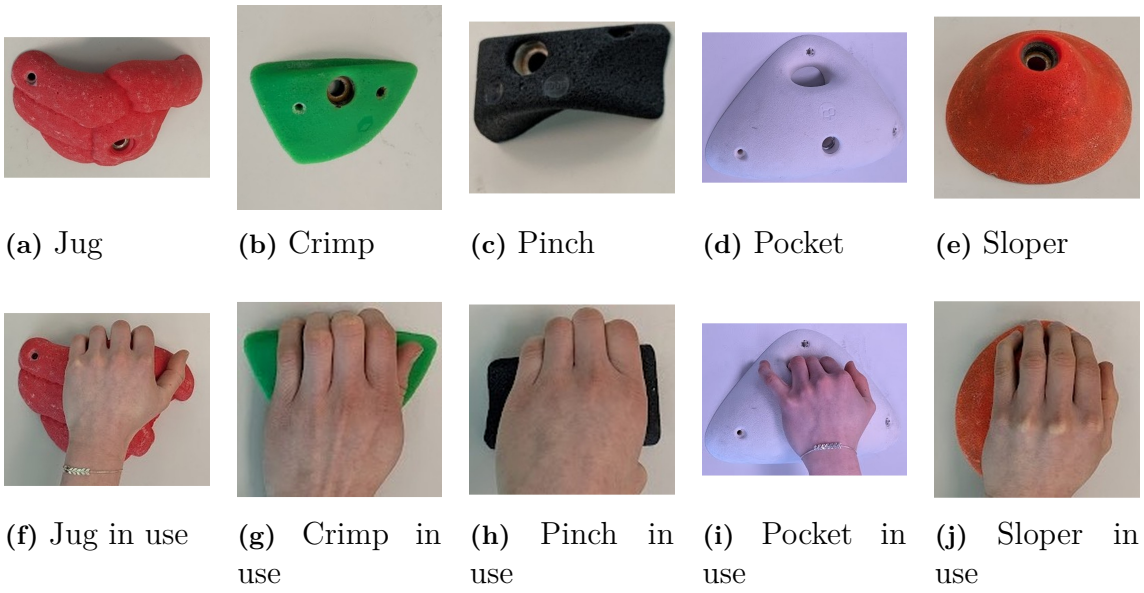
Theory

This chapter introduces the theoretical background relevant to the project. It covers climbing hold design principles, additive manufacturing, machine learning, similarity search, distance metrics, generative design, topology optimization and structural loading considerations.

2.1 Climbing hold design

Normally when creating climbing holds, an original design is carved out of high-density foam by hand. A mold is then made by pouring silicone over the foam shape. Lastly the mold is filled with polyurethane and the cured cast is sanded down to create the final hold [1, 2]. Although this approach can produce high-quality and engaging designs, it may lead to repetitive geometries over time due to design fixation [3, 4]. Design fixation refers to a phenomenon where designers creativity is limited because of excessive dependence on preexisting designs [4].

The occurrence of design fixation in the context of indoor climbing is indicated by how climbing hold designers historically imitate rock surfaces [15] and how external stimuli risk constraining ideation [16]. According to interviews, it's found that hold sizes and styles shift over time in response to emerging trends [5]. Furthermore its noted that holds can sometimes appear similar across brands. While not framed as a limitation, this trend-responsive development may indicate periods of convergence around particular shapes and stylistic directions.



(a) Jug **(b)** Crimp **(c)** Pinch **(d)** Pocket **(e)** Sloper

(f) Jug in use **(g)** Crimp in use **(h)** Pinch in use **(i)** Pocket in use **(j)** Sloper in use

Figure 2.1: Examples of climbing hold types and their typical use.

Climbing holds are commonly categorized into distinct types based on their geometric characteristics and corresponding gripping mechanics. The generalization for the hold types is based on classifications from manufacturers [17, 18]. A jug is generally characterized by a relatively large graspable surface with a pronounced positive edge or lip, like figure 2.1a. The geometry allows the fingers to wrap around the hold, enabling a full-hand grip with substantial support as seen in figure 2.1f. Crimps are categorically defined as a small hold with a thin, positive edge that primarily supports the fingertip contact as seen in figure 2.1b and 2.1g. A hold is categorized as a pinch if it has opposing graspable surfaces that require compression between the fingers and the thumb. The geometry is commonly elongated or prismatic, facilitating two-sided force application across the hold's lateral faces as seen in figure 2.1c and 2.1h. If a hold has one or more cavities or holes that accommodate one or more fingers, as seen in figure 2.1d and 2.1i, it is categorized as a pocket. A sloper is a hold with a mainly smooth, curved surface and no distinct positive edge. The geometry is typically convex or mildly concave and successful use relies primarily on friction and open-hand contact, as seen in figures 2.1e and 2.1j.

2.2 3D scanning and point cloud acquisition

A common type of 3D scanner is a structured light scanner which works by projecting a precisely calibrated light pattern onto an object. The reflection of the distorted pattern captures the object's shape and texture [19]. However, the accuracy of a structured light 3D scanner is challenged by reflective and dark materials. Reflective surfaces such as acrylic can cause overexposure as light from the projector is reflected back. Black surfaces have the opposite issue, as the scanner can recognize it as a void due to light absorption [19, 20]. A third obstacle involves geometries with holes in it, since the scanner cannot capture internal surfaces [19]. These material and geometric characteristics are frequently present in climbing holds, which may affect

scanning quality 2.2.

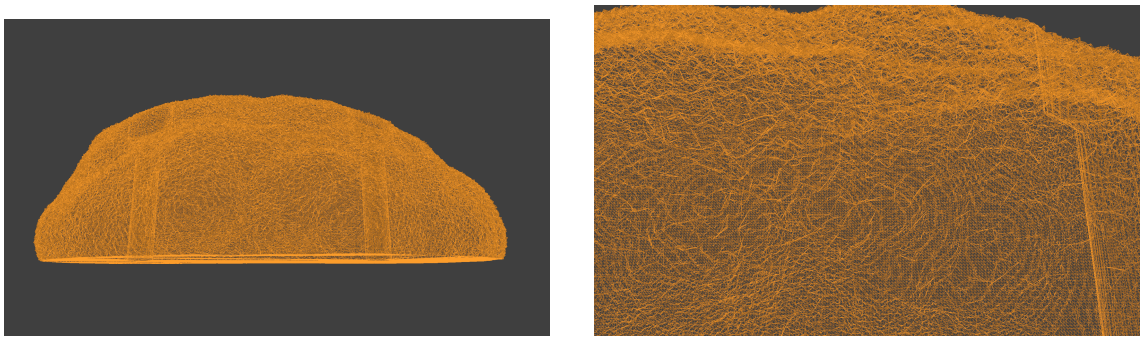
(a) Black dualtext hold.

(b) Hold with deep holes.

Figure 2.2: Example of two types of holds from Klätterlabbet that might be hard to accurately scan using a 3D scanner.

The many shots acquired from 3D scanning usually require some processing to create a coherent model. This typically involves the removal of background noise in the snapshots for a scan, subsequently followed by a global registration [21]. Global registration aligns the snapshots for a scanned object to a coherent representation, increasing the accuracy and giving it a clear visualization. This allows the scanned object to easily be exported in file formats such as PLY and STL.

The data generated by the Artec EVA scanner can be exported in formats such as STL and PLY-files [22]. These formats are among the most commonly used in 3D scanning applications [23]. In the STL format, a geometry is represented as a collection of triangular surface elements as shown in figure 2.3. Each triangle is defined by three vertices with associated x-, y- and z-coordinates, together with a normal vector indicating the outward-facing direction of the surface. The representation is purely surface-based, meaning that no explicit volumetric information is stored [24]. PLY-files support vertex coloring using RGB, transparency and generally store data in better quality than STL-files [23]. The major advantage is not needing to store the points in a mesh of triangles, instead PLY-files can store points as a point cloud.



(a) STL triangle mesh **(b)** Closeup view of the mesh

Figure 2.3: Two views of the STL mesh side by side

2.3 Additive manufacturing

The process of creating a physical product from a 3D CAD model by adding material layer-by-layer is called additive manufacturing (AM). This method enables the fabrication of complex geometries that are difficult to achieve using formative or subtractive techniques [7]. AM has traditionally been used for rapid prototyping [7], particularly through filament-based 3D printing (FDM), an extrusion-based technology in which thermoplastic material is forced through a heated nozzle and deposited layer by layer [25]. However, in recent years it has also emerged as a viable method for final-stage manufacturing [26].

Another form of additive manufacturing is Selective Laser Sintering (SLS). In this process, a thin layer of powder (usually 0.075-0.1 mm thick [27]) is spread on a building platform, the powder is then sintered using laser beams [28]. Once a layer is completed, the build platform is lowered by the thickness of one layer and a fresh layer of powder is applied. During the process, the unmelted powder serves as support for the structure. Enclosed cavities require escape holes to allow removal of trapped powder after printing. The most commonly used powder in these types of printers is nylon [27]. Commercial SLS systems typically impose minimum wall thickness constraints. For example, the Formlabs Fuse 1 specifies a minimum vertical wall thickness of 1.5 mm [29]. SLS PA12 components with wall thicknesses of 2.5–3.0 mm exhibit an average ultimate strength of approximately 43 MPa [30].

2.4 Machine Learning

Machine learning (ML) is a branch of artificial intelligence that enables computer systems to identify patterns and make predictions or decisions based on data without being explicitly programmed for every task. Instead of relying solely on predefined rules, machine learning models learn relationships from historical data and use this knowledge to generalize to new, unseen data [31].

In design and engineering applications, machine learning can be used to optimize, automate, and support different stages of the design process using existing data [8]. In engineering and generative design applications, machine learning has become increasingly important for analyzing and generating complex geometric structures.

This section focuses on neural networks, with particular emphasis on autoencoders. These models learn compact latent representations of input data, enabling reconstruction, interpolation, and generation.

2.4.1 Neural Network

Neural networks are computational models capable of learning complex mappings between input and output data through layered nonlinear transformations. They are widely used in machine learning for tasks such as classification, regression and representative learning [32].

For three-dimensional geometric data, neural networks can extract features directly from representations such as point clouds, enabling compact encoding and reconstruction of shapes. This makes them suitable for generative design applications [10].

Autoencoders

An autoencoder (AE) is a type of artificial neural network designed to learn compact representations of unlabeled data using unsupervised learning. Its primary objective is to learn an efficient encoding of the input by compressing it into a lower-dimensional latent space and subsequently reconstructing the original input from this compressed form [33].

A conventional autoencoder consists of two main components: an encoder and a decoder, as shown in figure 2.4. Together, they form a three-layer architecture comprising an input layer, a hidden layer (latent space) and an output layer [34]. The encoder maps the input data to a latent representation, while the decoder reconstructs the input from this latent space.

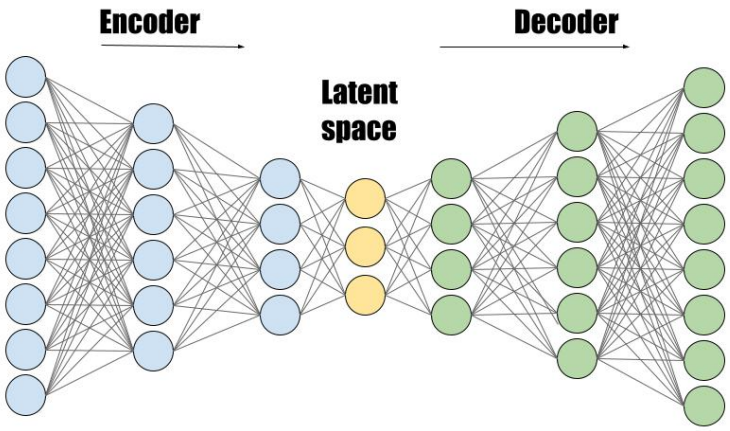


Figure 2.4: Autoencoder

A variational autoencoder (VAE) is a type of autoencoder that shares a similar structure with a traditional model, but differs in how it represents the latent space. Instead of mapping an input to a single point, a VAE maps the input to a probability distribution. Consequently, the VAE outputs two variables, a mean vector (μ) and a standard deviation vector (σ), which together define a Gaussian distribution [35]. This allows each input to be represented as a region in latent space rather than a fixed point, allowing latent variables to be sampled. To enable backpropagation, VAE:s use the reparameterization trick. A latent vector variable z is sampled as

$$z = \mu + \sigma * \epsilon \tag{2.1}$$

where $\epsilon \sim N(0, I)$ and $N(0, I)$ denotes the distribution. Overall, this formula allows the VAE to learn continuous latent spaces while still supporting efficient training [36].

Encoder

The encoder extracts the major features of the input data by performing a nonlinear transformation that typically reduces dimensionality. It maps the input to a latent representation in a new feature space.

The input layer and the hidden layer form an encoder. Given the original input data X, the encoded representation z is obtained by 2.2.

$$z = f(\cdot) = f(W_1 X + b_1) \tag{2.2}$$

where W_1 is the weight matrix, b_1 is the bias vector and $f(\cdot)$ is a nonlinear activation function [34]. The latent vector variable z represents a compressed feature embedding of the input data.

Decoder

The decoder maps the latent representation back to the original input space and attempts to reconstruct the original representation. The reconstructed output X^d is given as

$$X^d = g(\cdot) = g(W_2 z + b_2), \tag{2.3}$$

where W_2 is the weight matrix between the hidden layer and the output layer, b_2 is the bias vector, and $g(\cdot)$ is an activation function [34].

The objective of the decoder is to approximate the identity mapping by minimizing the reconstruction error between the original input X and the reconstructed X^d.

Multilayer perceptron

A multilayer perceptron (MLP) is a class of feedforward artificial neural network composed of multiple fully connected layers. Each layer performs a linear transformation of its input, followed by a nonlinear activation function, enabling the network to learn complex mappings between input and output spaces. A single layer can be expressed as a linear transformation with weights and biases, followed by a nonlinear function [37].

By stacking multiple such layers, the MLP can approximate complex nonlinear functions, making it a versatile model for learning complex relationships in high-dimensional data.

PointNet++ encoder

The PointNet++ encoder captures geometric structures of point clouds through hierarchical feature learning, aggregating local neighborhoods into increasingly abstract representations [38].

PointNet++ consists of three Set Abstraction (SA) levels, each performing sampling, grouping and feature extraction. In the sampling stage centroids are selected using Farthest Point Sampling (FPS). FPS iteratively picks the point most distant from all previously selected centroids. The grouping stage consists of local region that are selected via ball query where all the regions points are within radius r. In the final stage, a shared MLP is applied to all regions taking relative coordinates and any existing features as input, then max-pooling across the region to extract one feature vector per centroid. [38]

Folding decoder

A folding decoder reconstructs a point cloud by applying a learned deformation, referred to as a folding operation on the latent representation. The process begins with a fixed set of points sampled from a 2D grid embedded in $\mathbb{R}^2$ where each grid point $u_i = (x_i, y_i)$ does not initially correspond to the target object. Consequently,

the decoder learns the function

$$(u_i, \theta) : \mathbb{R}^2 \rightarrow \mathbb{R}^3, \tag{2.4}$$

where θ denotes the latent code. Altogether, the function maps each 2D grid point to a 3D coordinate, which deforms the accurate grid into the geometry of the target point cloud [39].

Training autoencoder

The autoencoder is trained by optimizing its parameters to minimize the reconstruction error between the input point cloud X and the reconstructed output X_d [34]. For point cloud data, this reconstruction error is commonly measured using Chamfer distance [40], introduced in section 2.4.3.

The corresponding objective function is defined as

$$J_{AE}(\theta) = L(X, X_d) \tag{2.5}$$

where θ represents the model parameters and $L(X, X_d)$ denotes the Chamfer distance between the input and reconstructed point sets.

The training process follows an unsupervised learning method and therefore does not require labeled data. By minimizing the reconstruction loss, the autoencoder learns a latent representation that captures the underlying geometric structure of the input data [34].

ReLU

In the encoder and decoder formulations (Equations 2.2 and 2.3), the nonlinear activation functions $f(\cdot)$ and $g(\cdot)$ are implemented using the rectified linear unit (ReLU).

The ReLU is one of the most widely used activation functions in artificial neural networks [41, 42]. It is defined as

$$\text{ReLU}(x) = \max(0, x) \tag{2.6}$$

or equivalently

$$\text{ReLU}(x) = \begin{cases} 0, & x < 0 \\ x, & x \geq 0 \end{cases}$$

The main advantage of ReLU is its computational simplicity, making it efficient to evaluate during both forward and back propagation [41].

2.4.2 Similarity Search

Similarity search refers to the process of identifying data points that are closest to a given query in a defined feature space. In many applications, data is represented as high-dimensional vectors, and similarity is measured using a distance metric such

as Euclidean distance [43].

A common approach is k-nearest neighbour (k-NN) search, where the objective is to retrieve the k closest vectors to a query point [43]. Similarity search is a fundamental operation in tasks such as information retrieval, clustering, and generative modeling, where identifying structurally or semantically similar samples is required.

Facebook AI Similarity Search (FAISS) is an open-source library for efficient similarity search and clustering of high-dimensional vectors. It is designed to handle large-scale datasets and supports approximate nearest neighbor search [44]. FAISS provides a range of indexing methods that enable efficient querying by reducing the computational complexity compared to brute-force search. These methods allow a trade-off between search accuracy and speed [44].

2.4.3 Distance equations

Distance metrics are used to quantify the similarity between data representations. In the context of geometric data such as point clouds, distance measures must operate on unordered sets of points. Two commonly used metrics are Chamfer distance and Earth Mover's Distance (EMD), both of which are widely applied in 3D shape analysis and generative modeling.

The Chamfer distance equation is a commonly used measure of dissimilarity between two point sets. It is defined by computing, for each point in one set, the distance to its nearest neighbour in the other set and summing these distances.

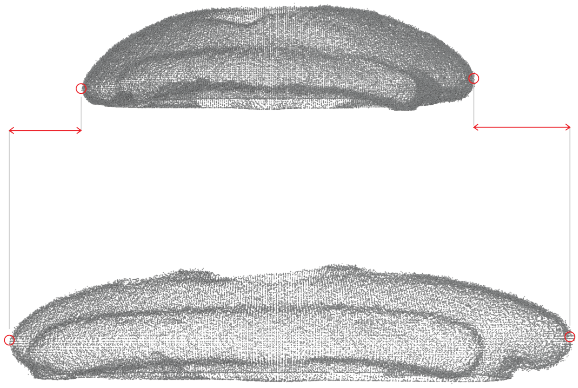


Figure 2.5: One-sided Chamfer Distance showing nearest-neighbor matching from the lower to the upper point cloud on dissimilar shapes.

Formally, for two point sets $A, B \subset \mathbb{R}^d$, the Chamfer distance from A to B is defined as

$$CH(A, B) = \sum_{a \in A} \min_{b \in B} d_X(a, b), \tag{2.7}$$

where d_X denotes the underlying distance metric [40].

A symmetric form of Chamfer distance is defined as

$$d_{\mathrm{CH}}(A, B) = \frac{1}{|A|} \sum_{a \in A} \min_{b \in B} \|a - b\|_2 + \frac{1}{|B|} \sum_{b \in B} \min_{a \in A} \|b - a\|_2. \tag{2.8}$$

The symmetric form averages both directed distances, ensuring the measure is invariant to the ordering of both point sets.

The Chamfer distance is widely used in applications involving point clouds, including machine learning. It is often employed as a computationally efficient approximation of more complex distance measures, such as the Earth Mover's Distance [40].

The Earth Mover's Distance (EMD) is a measure of dissimilarity between two distributions, defined as the minimum amount of work required to transform one distribution into another. It is formulated as an optimal transport problem, where mass is transported between elements of the two distributions [45].

Let one distribution be represented as suppliers and the other as consumers. The cost of transporting mass between elements is given by a ground distance c_{ij}. The optimal flow f_{ij} is obtained by minimizing the total transportation cost:

$$\min \sum_i \sum_j c_{ij} f_{ij} \tag{2.9}$$

subject to constraints on non-negativity and supply-demand consistency.

The EMD is then defined as the normalized total cost:

$$EMD = \frac{\sum_i \sum_j c_{ij} f_{ij}}{\sum_i \sum_j f_{ij}} \tag{2.10}$$

where f_{ij} denotes the optimal flow and c_{ij} the ground distance between elements [45].

2.4.4 Interpolation and blending

Interpolation is used to construct a new representation from a set of neighbouring samples. A common approach is to compute a weighted combination of vectors, where the weights depend on the distance to a reference point. Interpolation is performed using inverse-distance weighting (IDW) [46], where vectors closer to the reference point contribute more strongly to the result.

Given a set of vectors $\mathbf{v}_i$ and corresponding distances d_i, the weights are defined weighting:

$$w_i = \frac{1}{(d_i + \varepsilon)^p} \tag{2.11}$$

where p controls the influence of distance and ε is a small constant for numerical stability.

The interpolated vector is then computed as a normalized weighted sum:

$$\mathbf{v}_{\text{blend}} = \frac{\sum_{i=1}^{k} w_i \mathbf{v}_i}{\sum_{i=1}^{k} w_i} \tag{2.12}$$

where k is the number of retrieved neighbors, $\mathbf{v}_i$ is the latent vector of neighbor i, and d_i is its Chamfer distance to the anchor. The weight w_i is computed as the inverse of the distance raised to a power of p.

This formulation ensures that vectors closer to the reference point contribute more strongly to the resulting interpolation. This relies on the assumption that geometrically similar shapes are mapped to nearby points in the latent space, enabling meaningful interpolation between latent representations [47].

2.4.5 Generative design

Generative design is an iterative process in which multiple solutions can be created using an algorithmic or rule-based approach [48]. A generative system ensures fulfillment of the assigned requirements among all permutations and modifications of defined design parameters.

This section introduces the theoretical background of three different generative design approaches: Latent Representation Learning, Retrieval-based generation, and PointFlow.

Latent Representation Learning

Artificial Neural Networks (ANNs) are computational models capable of learning complex mappings from data through hierarchical feature extraction [49]. In the context of three-dimensional geometric data, neural networks can be trained to learn latent representations of shapes, capturing the underlying structure of geometric variation [47].

Architectures specifically developed for unordered point sets, such as *PointNet++*, enable the extraction of geometric features directly from point coordinates without requiring conversion into mesh representations [38]. Operating directly on point sets, these architectures achieve higher geometric fidelity while reducing preprocessing complexity compared to voxel or mesh-based representations [38]. Point cloud data is typically normalized prior to network training, involving scaling and centering of geometric models within a common reference frame.

Neural networks intended for geometric reconstruction are optimized by minimizing a reconstruction loss over point clouds, through which the model learns a latent representation of the geometric variation present in the training data [47]. Well-structured latent spaces have been shown to support both interpolation between known geometries and the generation of novel shapes, achieved by sampling in latent space and decoding the resulting vectors into point cloud representations [47].

The generative capability of such models follows from the structure of the learned latent space, by sampling from regions of this space and passing the samples through a trained decoder, previously unseen geometries can be produced that remain consistent with the distributional characteristics of the training data.

Retrieval-based generation

Retrieval-based generation constitutes a data-driven generative framework in which new designs are constructed through the selection and modification of reference samples from an existing dataset. Based on a structured interpretation of the user's input, the retrieval stage selects one or more references from the dataset [50]. Initially, candidate data are filtered using metadata [51]. Subsequently, a similarity search is performed within the filtered subset using geometric embeddings to identify the most relevant samples. The retrieved data serve as anchors for the modification and embedding recombination [50], ensuring that generated designs remain consistent with the existing dataset while allowing controlled variation.

Similarity search in high-dimensional latent spaces relies on established approximate nearest neighbor algorithms. Given a latent vector, the retrieval step returns k nearest neighboring samples in the latent space [52], enabling efficient identification of geometrically similar reference holds.

Following the retrieval stage, the selected reference data is combined by blending their latent vectors [47]. The resulting blended representation can then be decoded

to create new data.

PointFlow

PointFlow is a probabilistic generative model for 3D point clouds that formulates shape generation as a distribution of distributions. Specifically, it learns a two level hierarchy structure in which the first level models the distribution of shapes and the second level models the distribution of points given a shape. This formulation enables the model to both sample new shapes and generate an arbitrary number of points from each shape [53].

Each level of the hierarchy is parameterized using continuous normalizing flows (CNFs), which define invertible transformations from simple prior distributions to complex target distributions. The invertibility of CNFs permits exact likelihood computation via the change of variables formula, thereby enabling training within a variational inference framework [53].

PointFlow has been validated on objects such as chairs and airplanes, where objects share consistent topological characteristics while exhibiting substantial geometric variation [53]. The results demonstrate the model's ability to capture structural differences between shapes and fine grained local surface details.

2.5 Point Cloud to CAD

Point clouds generated by structured light 3D scanning are often imprecise and risk including noise [19]. When generating new point clouds from these scans, noise risks being amplified, which can hurt the overall dimensional accuracy of the resulting part [54].

Constraint driven CAD programs rely on exact geometric and topological data, making the direct use of noisy and unstructured point clouds difficult without first reconstructing them into parametric geometry [55, 56]. Even when possible, this process requires precise input data [57], creating the need for complex preprocessing while not guaranteeing dimensional accuracy [58]. This stems from the constraint solver algorithm becoming increasingly complex with noisy or low fidelity data, resulting in less accurate solutions [59].

In comparison, Blender is a data driven geometry engine that can directly process point clouds as coordinate based vertices, without requiring a constraint solver. The interpreted data is largely identical to the imported data, with the exception of rounding and downsampling. Blender's geometry node system enables non destructive editing of point cloud data [9], allowing conversion into manufacturable geometry [60].

To produce a solid, closed model from a point cloud, a volumetric reconstruction approach is used. Each point contributes to a volumetric density field, using the

Blender node *Points to Volume*, which defines regions of space as inside or outside the object. This defines a continuous implicit model rather than relying on explicit connectivity [61, 62]. A key advantage of this method is that connectivity is not created directly between neighboring points. Instead, it forms naturally when an isosurface is extracted from the volume. Because the volume represents occupancy rather than a mesh, it is more resistant to noise, gaps, and uneven sampling, which are common characteristics of structured light point clouds [63].

2.6 Structural loading principles

Both static and dynamic loads should be considered. Therefore, the structural design must ensure sufficient load-bearing capacity under tensile and bending stresses while maintaining an appropriate safety margin.

The component's load-bearing capacity can be estimated from its material strength and cross-sectional area. The ultimate force is given by

$$F_{\text{ultimate}} = \sigma_{\text{UTS}} \cdot A \tag{2.13}$$

where σ_{UTS} is the ultimate strength of the material and A is the load-bearing cross-sectional area.

To ensure safe usage, a safety factor (FoS) is usually applied:

$$F_{\text{design}} = \frac{F_{\text{ultimate}}}{\text{FoS}} \tag{2.14}$$

The safety factor accounts for uncertainty in loading conditions, geometric assumptions, and material variability.

Static body weight is exceeded by dynamic forces due to acceleration during movement. Experimental studies show that forces acting on climbing holds are time-varying and increase under dynamic conditions such as rapid movements or falls [64]. Therefore, structural design must consider amplified force scenarios when defining load requirements.

2.7 Topology optimization

Topology optimization is a computational design method used to determine an efficient distribution of material within a prescribed design domain. In structural applications, the aim is commonly to minimize material usage while maintaining sufficient stiffness and load-carrying capacity [65]. This requires the geometry to be represented in a discrete form, the structural response to be evaluated numerically, and the material layout to be updated according to an optimization objective.

In this section, voxel-based geometry representation is first introduced as a way of discretizing three-dimensional objects into solid and void regions. Ray casting

is then described as a method for classifying whether points in the voxel grid lie inside or outside the original geometry. The section further presents density-based topology optimization, where each finite element is assigned a continuous density variable that controls the local material stiffness. The SIMP method is used to penalize intermediate densities and promote a clear solid-void material distribution. Finally, the finite element method is introduced as the numerical framework used to compute displacements, stresses, and structural stiffness during the optimization process.

2.7.1 Voxel representation

Voxelization is the process of converting a continuous geometric object into a discrete representation on a three-dimensional grid, where each cell, called a voxel, is analogous to a pixel in two dimensions. In binary voxelization, each voxel takes one of two values, representing either void or solid depending on whether the voxel belongs to the object or not [66].

Connectivity describes how voxels are linked to one another and is important for defining topology and continuity in the discrete model. In three dimensions, connectivity is commonly defined as 6-connectivity (shared faces), 18-connectivity (shared faces and edges), and 26-connectivity (shared faces, edges, and corners). The choice of connectivity affects how components are identified and how geometric continuity is interpreted [66].

An important artifact in voxelization is tunneling, which occurs when diagonally adjacent voxels appear connected without sharing a face. This may create artificial gaps or unintended paths through the structure. Tunneling is more likely when using 18- or 26-connectivity and is avoided by enforcing 6-connectivity, which ensures face-to-face connections and more physically meaningful continuity [66].

To obtain a voxel representation from a surface mesh, each voxel center must be classified as either inside or outside the geometry. One common approach for performing inside-outside classification is ray casting. In this approach, a ray is cast from a query point and the number of intersections between the ray and the surface is counted. An odd number of intersections indicates that the point lies inside the geometry, while an even number indicates that it lies outside [67].

2.7.2 Density-based topology optimization

The design domain is discretized into finite elements, where each element is assigned a density variable $\rho_e \in [0, 1]$ [68]. A value of $\rho_e = 1$ represents solid material, while $\rho_e = 0$ represents void. Intermediate values correspond to partially filled elements, which are typically penalized to obtain a clear solid–void design [68].

A commonly used approach is the Solid Isotropic Material with Penalization (SIMP) method, which relates the elastic modulus E_e of an element to its density ρ_e as

$$E_e(\rho_e) = \rho_e^p E_0 \qquad p \geq 1 \tag{2.15}$$

where E_0 is the elastic modulus of the solid material and p is a penalization factor [68, 69, 70]. Higher values of p discourage intermediate densities and promote a discrete solid–void solution.

The most common objective in topology optimization is the minimization of compliance, which corresponds to maximizing structural stiffness:

$$J(\rho) = \mathbf{f}^T\mathbf{u} = \mathbf{u}^T\mathbf{K}(\rho)\mathbf{u} \tag{2.16}$$

where $\mathbf{K}$ is the global stiffness matrix, $\mathbf{u}$ is the displacement vector, and $\mathbf{f}$ is the force vector [68, 71].

The optimization is subject to a volume constraint

$$\sum_e \rho_e v_e \leq V_{\max}, \qquad 0 \leq \rho_e \leq 1 \tag{2.17}$$

where v_e is the volume of element e and $V_{\max}$ is the prescribed maximum material volume [68, 71].

The structural response is evaluated using the finite element method, leading to the equilibrium equation

$$\mathbf{K}(\rho)\mathbf{u} = \mathbf{f} \tag{2.18}$$

The topology optimization problem can therefore be written as

$$\begin{aligned} \min_{\rho} \quad & J(\rho) \\ \text{subject to} \quad & \mathbf{K}(\rho)\mathbf{u} = \mathbf{f} \\ & \sum_e \rho_e v_e \leq V_{\max} \\ & 0 \leq \rho_e \leq 1 \end{aligned} \tag{2.19}$$

.

2.7.3 Finite element method basics

The finite element method (FEM) is a numerical technique used to approximate the response of a continuous body by replacing it with a finite number of interconnected elements. These elements are connected at nodes, where the primary unknowns, typically displacements in structural analysis, are defined [72].

Within each element, the displacement field is approximated by interpolation functions expressed in terms of the nodal values. In this way, a continuous field is represented by a piecewise approximation over the discretized domain [72].

For three-dimensional problems, common element types include tetrahedral and

hexahedral elements [72]. The mechanical behavior of each element is described by an element stiffness relation of the form

$$\mathbf{f}_e = \mathbf{k}_e \mathbf{d}_e \tag{2.20}$$

where $\mathbf{f}_e$ is the element nodal force vector, $\mathbf{k}_e$ is the element stiffness matrix, and $\mathbf{d}_e$ is the vector of element nodal degrees of freedom [72].

The element equations are assembled into a global system of equilibrium equations (2.18). After application of boundary conditions, the unknown nodal displacements can be solved.

In one-dimensional form, the strain-displacement relation may be written as

$$\epsilon_x = \frac{du}{dx} \tag{2.21}$$

and for a linear elastic material the stress-strain relation is given by Hooke's law,

$$\sigma_x = E\epsilon_x \tag{2.22}$$

where E is the modulus of elasticity [72]. More generally, strains are obtained from the displacement field, and stresses are subsequently determined from the constitutive relation.

The equivalent stress σ_e is a scalar measure of the multiaxial stress state, combining all stress components into a single value. A commonly used approximation is the von Mises equivalent stress, defined as

$$\sigma_e^{\mathrm{VM}} = \sqrt{\frac{1}{2}(\sigma_1 - \sigma_2)^2 + \frac{1}{2}(\sigma_2 - \sigma_3)^2 + \frac{1}{2}(\sigma_3 - \sigma_1)^2} \tag{2.23}$$

where σ_1, σ_2, and σ_3 are the principal stresses [73].
The von Mises yield criterion is then expressed as

$$f = \sigma_e - \sigma_s \tag{2.24}$$

where σ_s is the material yield strength. As long as $\sigma_e < \sigma_s$, the material response is assumed to be linear elastic and follows Hooke's law (2.22). When $\sigma_e \geq \sigma_s$, yielding occurs and permanent (plastic) deformation may develop [73].

The theoretical foundations presented in this chapter form the basis for the methods described in Chapter 3.

3
Methods

This chapter describes the methods used to develop and evaluate the proposed climbing hold generation pipeline. The workflow integrates multiple stages, including dataset creation through 3D scanning, generative design using machine learning, geometric processing, additive manufacturing and topology optimization. The proposed workflow is divided into sub-parts and visualized in figure 3.1.

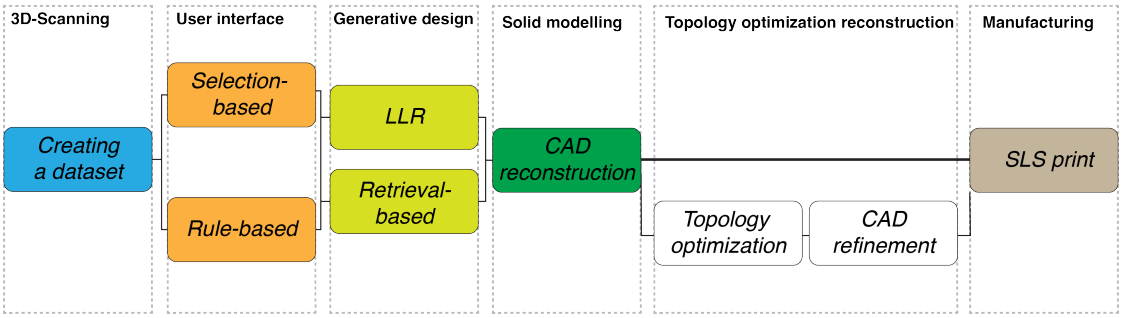


Figure 3.1: Pipeline

The pipeline is available in a GitHub repository, and instructions on how to use it are provided in the *README.md*. The project structure and further implementation details are described more thoroughly in *https://github.com/theaa20/EENX16_VT26_12A* [74].

3.1 Creating a dataset

To create the dataset, several preprocessing steps were carried out. Climbing holds were collected from two sources: 200 from an external database and roughly 100 climbing holds were scanned using the Artec Eva scanner. The holds were then manually classified into their respective hold categories (jug, crimp, pinch, pocket, sloper) based on subjective geometric assessment. Although additional hold categories exist, these represent the most common classifications used in climbing hold design.

Following categorization, the STL files of the climbing holds were converted to PLY format using the software *CloudCompare*. To improve reproducibility and efficiency, the conversion process of all the STL files was automated through a bash script. For more efficient computational processing, the normalized PLY files containing 2048

points were subsequently converted into NPY files, which is NumPy's native binary file format for storing array-based data [75]. This format enables faster loading and processing of point cloud data within Python-based machine learning workflows while preserving the geometric information of each climbing hold [76]. Furthermore, a point count of 2048 point were chosen as a trade-off between fidelity and computational cost.

After conversion, geometric dimensions were extracted from the resulting point clouds using a Python-based processing script. The computed measurements were subsequently exported to a CSV file.

3D scanning

3D scanning was used to create a dataset for the training of the generative design model. The Artec EVA 3D scanner, with an accuracy of approximately 0.1 mm according to the producer [22], was selected due to its suitability for capturing high-resolution geometric data.

The scanning process was conducted by two operators. One operated the scanner, while the other rotated the hold using a turntable to ensure that all surfaces were captured. Multiple scans were performed from different angles to cover the entire geometry of each hold. A total of 100 holds were scanned, with an even distribution across the different categories to ensure a balanced dataset for model training.

After scanning, the data was processed using Artec Studio 20, a proprietary scan processing software. Noise, such as background points and stray reflections, was removed to clean the scan. The individual scans were then aligned and merged into a single 3D model using global registration (explained in section 2.1). Remaining gaps in the mesh were subsequently filled to produce a continuous surface suitable for further analysis.

Several challenges were encountered during scanning. Holds with dark or reflective surfaces were more difficult to capture accurately, occasionally resulting in incomplete geometry. In addition, holds with deep cavities or other challenging geometries that could not be reliably reconstructed were excluded from the dataset. Minor movements of the scanning frame occasionally caused misalignment between scans, requiring careful adjustment during the registration process. Overall, the combination of precise scanning, careful rotation of the holds, and thorough post-processing ensured a high-quality 3D data set suitable for subsequent analysis.

3.2 User interface

This section describes the developed user interface for the generative pipeline. Two interaction methods were implemented: a text-based interface and a selection-based interface, both within the same web application. During the development process,

several frameworks for implementing the interface were evaluated.

The user interface was initially developed in Python using the built-in library Tkinter. Tkinter was selected due to its ease of use but was limited in its design flexibility and did not support interactive 3D plot rotation. Therefore, alternative approaches for user interface development were explored. One option involved the use of PyQt, another Python-based framework for user interface creation. Alternatively, the interface could be implemented using HTML. Both PyQt and HTML were evaluated due to their support for interactive 3D plots, but HTML was ultimately selected due to its greater design flexibility.

Creating the website

To enable user interaction with the generative system, a locally hosted HTML-based website was developed. This approach was chosen to provide a lightweight and accessible interface without requiring a dedicated backend or server infrastructure. By running locally, the interface can directly interact with the generative pipeline while maintaining low latency and ease of deployment.

The website serves as the main interaction layer between the user and the generative system, enabling the specification of design parameters and initiating the generation process. It was designed to support both text-based and selection-based input methods, allowing flexibility in how users define desired climbing hold characteristics.

The two interface types are illustrated in Appendix C, figure C.1 and figure C.2. The text-based interface allows natural language input through a text field, accompanied by example prompts that guide the user in specifying valid inputs. The selection-based interface provides predefined parameter choices using sliders. A generate button initiates the pipeline and upon completion, the resulting hold is displayed on the canvas. The interface also includes a *"Save hold"* button, which allows the generated hold to be stored as an STL file in the *outputs/saved_holds/* directory.

In the text-based interface, the natural language input is processed through a rule-based mapping system that converts climbing-specific keywords into structured design parameters, including category, size, depth, and texture. A parameter fallback mechanism ensures that unspecified values, such as size, are automatically assigned default values. The resulting structured specification is then passed to the generative pipeline for point cloud generation.

In contrast, the selection-based interface allows users to specify design parameters through predefined options instead of using natural language input. This approach proved to be a more controlled input method, reducing ambiguity and ensuring that all required parameters are explicitly defined. The selected parameters are then passed to the generative pipeline for point cloud generation.

3.3 Generative design process

This section describes the generative methods used in the pipeline. Three approaches were initially considered: Pointflow, a Retrieval-based method and a Latent Representation Learning method. PointFlow was evaluated first but could not be implemented due to technical constraints, as described below. The two remaining approaches were successfully implemented and are described in sections 3.3.1 and 3.3.2.

PointFlow was installed using Windows Subsystem for Linux (WSL), as native Windows support was not feasible. The installation was also conducted within a virtual environment using Miniconda, since PointFlow requires Python 3.6, which is incompatible with the system-wide Python installation. Several dependency conflicts were encountered, which required Conda and multiple Pytorch-related packages to be downgraded before most libraries could be installed. The installation was ultimately abandoned after it was found that PointFlow requires a version of the CUDA toolkit that WSL does not support. Due to time constraints, migrating to a native Linux environment was not feasible and the approach was therefore dropped.

3.3.1 Retrieval-based generation

The Retrieval-based generation produces novel climbing hold geometries by blending latent representations of geometrically similar holds. The process is conditioned on user-specified parameters, namely category and size, which are extracted from the user interface and used to filter a CSV-based metadata catalogue. An anchor hold is then selected at random from the filtered subset using uniform sampling. Additionally, the sampling procedure ensures that the same hold is not selected in consecutive iterations when more than one candidate exists in the filtered subset.

The anchor geometry is encoded into a latent vector via the trained autoencoder encoder, described in section 2.4.1. A similarity search is subsequently performed in latent space using FAISS [44], retrieving the k nearest neighbours to the anchor from the filtered subset. If the filtered subset is fewer than k, one of the filter parameters is relaxed. The parameters are relaxed in the order of depth followed by the size. Geometric similarity is evaluated using the Chamfer distance, which measures the mean nearest-neighbour distances between two point sets. The retrieved latent vectors are combined through a distance-weighted average, described in section 2.4.4, and the resulting blended vector is decoded into a new point cloud representation of the generated hold.

An autoencoder was implemented, consisting of an encoder and a decoder trained jointly to minimize a point cloud reconstruction loss. Prior to training, point clouds were normalized to ensure consistent scale and positioning across samples, thereby reducing unwanted geometric variance during training. The dataset was partitioned into training and validation subsets using an 80/20 split, balancing the need for sufficient training data with model evaluation. Figure 3.2 illustrates a schematic

overview of the autoencoder architecture.

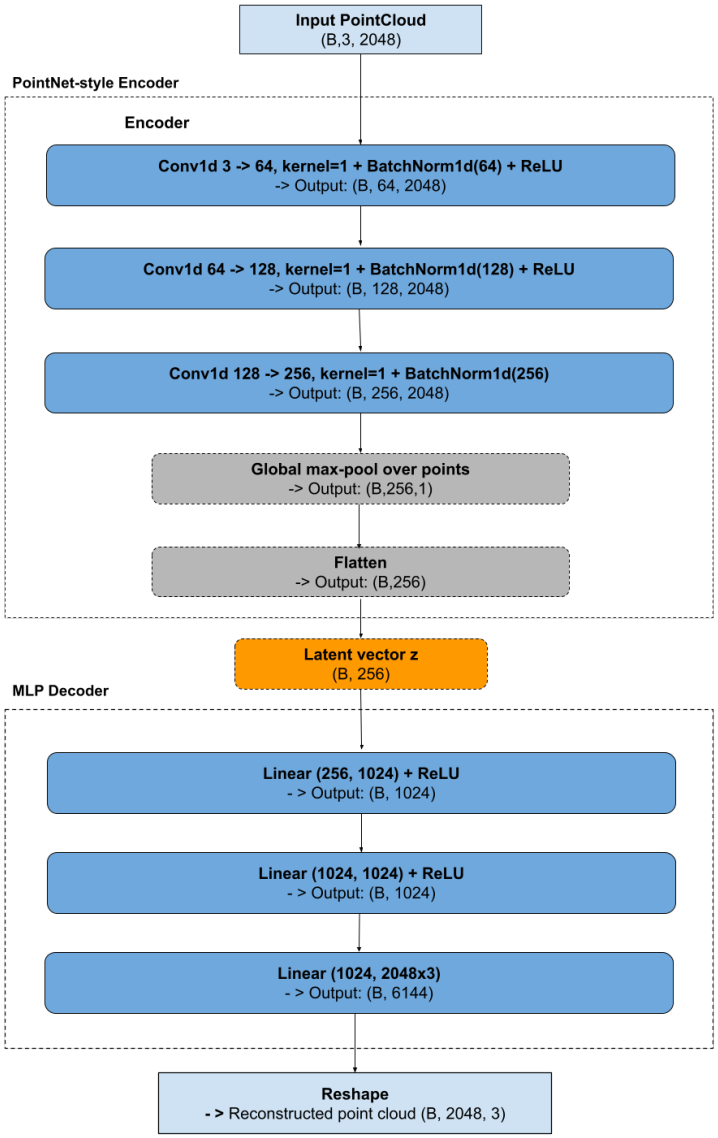


Figure 3.2: A conventional autoencoder

The encoder is a feedforward neural network with layers implemented as 1D convolutions with kernel size 1, which are applied to each point independently. Feature dimensions are progressively increased and a global max-pooling operation is used to aggregate point-wise features into a permutation-invariant latent representation. This results in a compact latent vector of dimension 256.

The decoder consists of a fully connected multilayer perceptron (MLP) that maps the latent vector back to a point cloud representation. The output is reshaped to match the original point cloud dimensions.

To train the model, the Chamfer distance between the input point cloud and the

reconstructed output is minimized. This loss function encourages the decoder to generate reconstructions that closely resemble the original geometry. Upon convergence, the trained model weights were saved for use during inference.

3.3.2 Latent Representation Learning

The Latent Representation Learning approach revolved around generating climbing hold geometries based on the information acquired from a compressed probabilistic latent space. The latent space consisted of climbing hold shapes derived from 3D point cloud data. The generation was guided by user-defined parameters through the GUI (Appendix C, figure C.2). These parameters enabled the model to sample new shapes conditioned on a desired hold category. The encoder compressed the input point cloud into a probability distribution within the latent space, parametrized by a mean and a variance vector. The decoder then reconstructed a point cloud from a sample drawn from this distribution. The hold category specified in the CSV metadata file was used during the creation of a learned vector. This vector was concatenated with the latent sample before decoding, allowing the model to generate category-specific holds.

The autoencoder consisted of two parts, an encoder based on Pointnet++ and a decoder that implemented with a FoldingNet structure. Below is a figure of the autoencoder's construction blocks (figure 3.3).

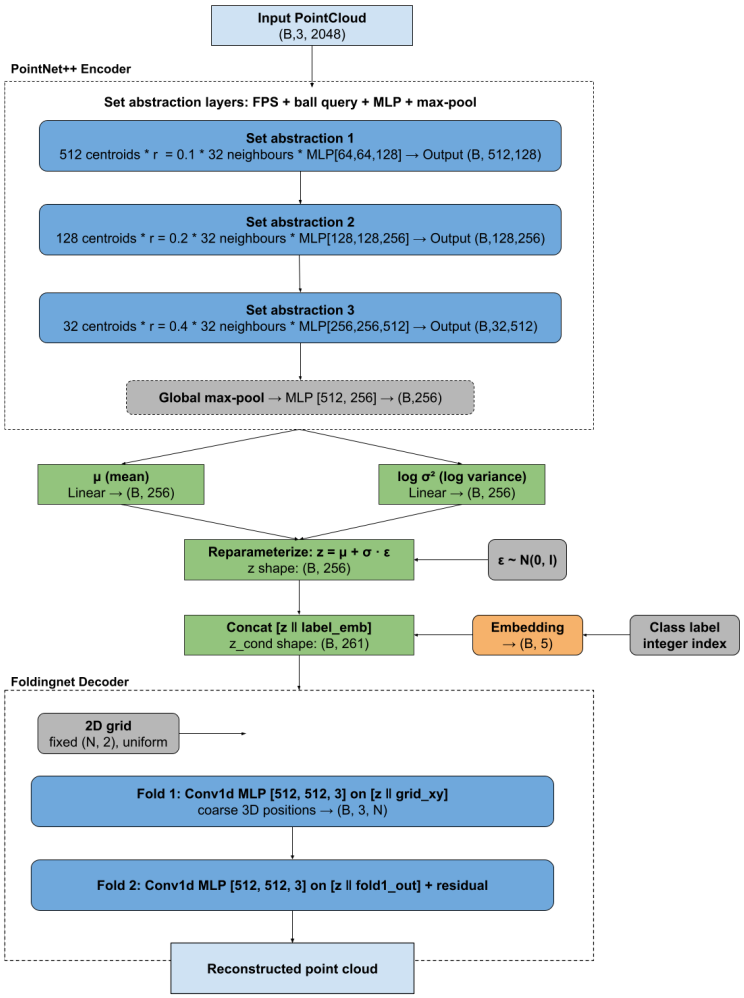


Figure 3.3: Variational autoencoder

The encoder was based on the PointNet++ architecture and processed point clouds of shape (B, 3, 2048). It consisted of three hierarchical set abstraction (SA) layers, each reduced the number of points which increased feature representation. In each SA layer, Farthest Point Sampling (FPS) was used to select representative centroids. For each centroid, local neighborhoods were constructed using ball query with a predefined radius. These local regions were processed using shared MLP 2.4.1, followed by max-pooling to extract local features. The configuration of the three set abstraction levels is summarized in table 3.1.

Level	Centroids	Radius	Inputs	Output	What it captures
SA1	512	0.10	(xyz only)	128	Fine local details
SA2	128	0.20	128	256	Medium structures
SA3	32	0.40	256	512	Global shape

Table 3.1: Set abstraction levels for the Latent Representation Learning Encoder

The first abstraction layer sampled 512 centroids with a radius of 0.10 and learned 128 dimensional features from spatial coordinates, capturing fine local details, such as edges and curves. The second layer reduced the representation to 128 centroids with a radius of 0.20, producing 256-dimensional features that denoted medium level structures such as surfaces. The third layer further abstracted the point cloud into 32 centroids with a radius of 0.40 and generated 512-dimensional features, capturing global shape information.

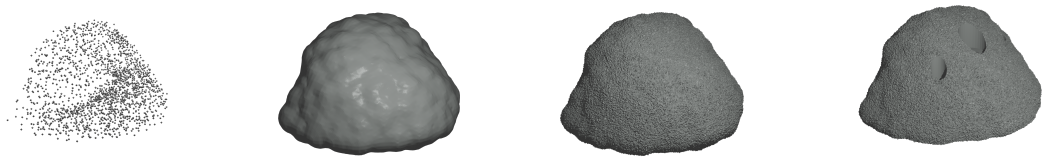
Following the hierarchical feature extraction, a global max pool was used to create a single global descriptive vector. Continuously, this vector was passed though MLP with layer dimensions [512,256] which produced a latent representation of size (B,256). The latent space was parametrized using two parallel linear layers, the mean (μ) and the logarithm of the variance ($\log \sigma^2$), each with the dimension of 256. A latent vector z was then sampled using the reparameterization trick 2.1. To incorporate conditional information, a learned class label embedding was concatenated with z which formed a conditioned latent representation z_{cond}.

The decoder followed the FoldingNet structure. A fixed two-dimensional grid of shape (N,2) was first generated and uniformly sampled. The conditioned latent vector z_{cond} was then concatenated to each grid point. The reconstruction process consisted of two sequential folding stages. Each stage was implemented as a Conv1d MLP[512,512,3] which mapped the concatenated input to three-dimensional coordinates. The second folding stage further refined these coordinates by incorporating a residual connection from the first stage output. The final output of the decoder was a reconstructed point cloud of shape (B, 3, N).

The model was trained on the point clouds of both the BHToolset [77] and the scanned holds across all five categories. This dataset was split into 80/20 for training respectively testing. Underrepresented hold types were augmented by application of left-mirroring, uniform scale jitter between 85–115%, as well as small positional noise at 1% of bounding box scale. The KL divergence keeps the latent distribution close to a standard normal. A loss function combining the Chamfer distance 2.4.3 and the KL divergence was created by weighting them by a parameter called beta. In an attempt to control variability, the beta was set to 0.001.

3.4 Point Cloud to CAD

The generated point cloud is then exported as a PLY file containing the x, y, z coordinates of points on the model surface, and imported into Blender as unconnected vertices, as shown in figure 3.4.



(a) Point cloud **(b)** Solid mesh **(c)** Displace texture **(d)** Boolean operations

Figure 3.4: Generalized blender pipeline

A Geometry Nodes pipeline is then applied to convert the point cloud into a manufacturable mesh. The vertices are first converted to points using the *Mesh to Points* node, then passed to *Points to Volume* to produce a volumetric density field, and finally converted to a surface mesh using the *Volume to Mesh* node. The voxel size and point radius are set relative to the bounding box size of the point cloud. For pocket holds, a smaller point radius is used to better preserve the concave hole geometry, while a lower threshold value is applied during mesh extraction to maintain surface detail.

Subsequently, a voxel mesh is applied to ensure that the resulting mesh is fully watertight before further processing. The surface texture is then applied using a CLOUDS displacement modifier. Three levels are supported: smooth, medium, and rough, each with distinct noise scale and displacement strength values.

The bottom face of the hold is flattened using a Boolean difference operation with a cube primitive. The cut depth is set to 35% of the model height for most hold categories, and 60% for pocket holds, which require a deeper cut due to the concave geometry. After flattening, the model is centered at the origin and scaled to the target size derived from a CSV file containing measured dimensions of the scanned holds. A maximum size limit is applied to ensure the model remains within manufacturing constraints. Vertex coordinates are manipulated directly rather than using Blender's *transform_apply* operator, which was found to be unreliable in headless mode.

Bolt and screw holes are placed automatically based on the hold size and category. For most categories, a *bmesh* query identifies vertices on the flat bottom face and uses their bounding box to determine hole placement. For pocket holds, the model bounding box is used directly. Holes are placed as Boolean difference operations using cylinder primitives. The number and type of holes depend on the target size, holds larger than 150mm in width receive a central M10 bolt and two M5 screws. Holds between 100mm-150mm receive two M5 screws. Holds smaller than 100mm receive a single central M10 bolt. Each hole includes a counterbore, positioned at 60% of the model height from the bottom. The finished model is then exported as an STL file for later stages.

3.5 Load case definition and design requirements

The static load requirement was derived from the mechanical properties of PA12 components manufactured by the SLS printer 2.3.

A representative minimum load-bearing cross-sectional area near the bolt connection was estimated to be 50 mm^2. Therefore, the ultimate load capacity was calculated as 2.13

$$F_{\text{ultimate}} = 43 \times 10^6 \text{ Pa} \cdot 50 \times 10^{-6} \text{ m}^2 = 2150 \text{ N}.$$

A safety factor of 1.5 was applied, this resulted in a required static design load of 2.14

$$F_{\text{design}} \approx 1.43 \text{ kN}.$$

Dynamic loading was estimated by multiplying the static body weight by a factor of two to account for the amplified forces experienced during dynamic movements, consistent with experimental observations of dynamic forces in climbing [64]. To ensure safety across different weight classes, the 99th percentile male body mass of 109.68 kg [78] was used.

$$F_{\text{dynamic}} = 109.68 \text{ kg} \cdot 9.81 \text{ m/s}^2 \cdot 2 \approx 2.15 \text{ kN}.$$

Hence, the required dynamic design load was set to

$$F_{\text{design,dynamic}} \geq 2.2 \text{ kN}.$$

Table 3.2: Structural and manufacturability requirements for climbing hold design

Category	**Requirement**	**Quantified target**
Structural	Static load-bearing capacity	$\geq 1.43\,\text{kN}$
Structural	Dynamic load resistance	$\geq 2.2\,\text{kN}$
Manufacturability	Minimum vertical wall thickness (SLS, PA12)	$\geq 1.5\,\text{mm}$

3.6 Topology optimization framework

This section presents the MATLAB-based framework used to topology optimize the climbing holds. The workflow includes voxelization of the input STL geometry, detection of interior holes, definition of passive and free design regions, and assignment of load and support conditions. The optimization is performed using a density-based method with SIMP interpolation, where voxel densities are updated iteratively based on finite element compliance sensitivities while satisfying a prescribed volume constraint.

After optimization, the density field is thresholded into a binary solid-void model

and post-processed to ensure structural continuity and manufacturability. The final geometry is then validated using voxel-based finite element analysis, where the displacement field, stress distribution, von Mises equivalent stress, and safety factor are evaluated under the prescribed loading conditions.

3.6.1 Input geometry handling

The input geometry was provided as an STL surface mesh and converted into a voxelized structure through ray-casting as explained in section 2.7.1. An example of a voxelized hold with voxel resolution 24×30×20 can be seen in figure 3.5.

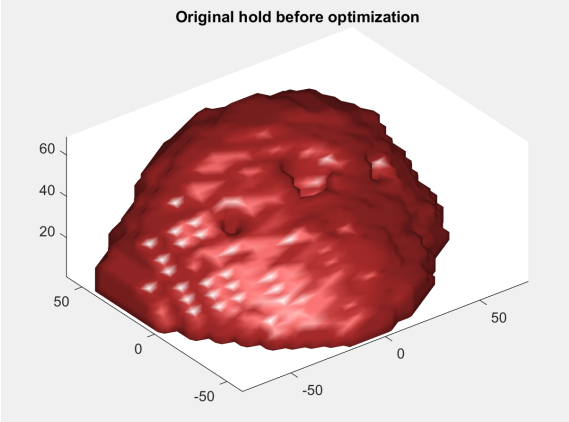


Figure 3.5: Figure of a voxelized hold in MATLAB.

The voxel grid was generated from the bounding box of the geometry with a small padding in each coordinate direction. For the optimized hold that was printed, a resolution of 160×100×58 voxels was used.

3.6.2 Hole detection

Interior voids are identified before optimization so that they remain empty throughout the design process. The voxel model is processed slice by slice in the z-direction as shown in figure 3.6. In each slice, connected void regions are identified, and components touching the outer slice boundary are discarded as they correspond to the outside area rather than holes. Connected regions in each slice are identified using 4-connectivity, which is the 2D analogue of 6-connectivity (Section 2.7.1), meaning that voxels are considered connected only through shared edges. This reduces the risk of tunneling artifacts (Section 2.7.1). The remaining regions are treated as hole candidates if they satisfy simple geometric filtering criteria based on area and compactness.

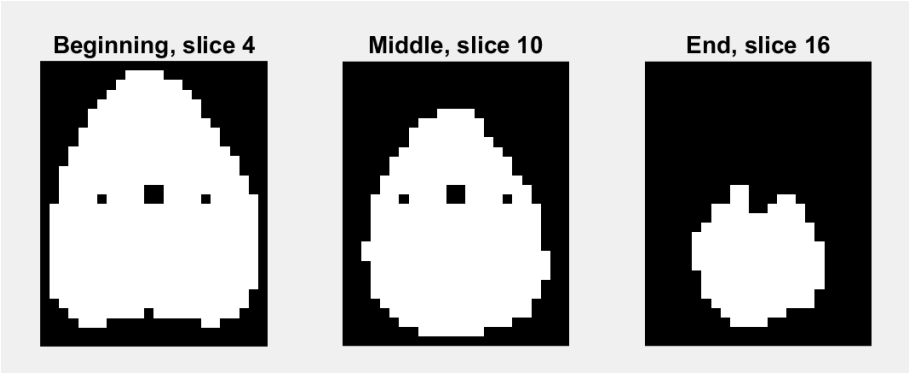


Figure 3.6: Three samples of slices that the code runs through to find boltholes.

The centroids of the retained slice-wise candidates are then grouped into three clusters. Within each cluster, one candidate per slice is selected and combined into a three-dimensional hole label field. Finally, connected-component cleanup is applied so that only the largest connected part of each detected hole is retained. In three dimensions, hole candidates are further refined using 6-connectivity, where voxels are connected through shared faces. This ensures that only physically continuous void regions are retained.

3.6.3 Design masks

The voxel domain is divided into passive void, passive solid, and free regions. The passive void region corresponds to the detected interior holes and is constrained to remain empty during optimization. The passive solid region contains material that must be preserved. This includes a ring of voxels around the detected holes and an outer shell extracted from the surface of the geometry. The shell is extended inward by a prescribed number of layers in order to preserve the exterior shape. For the SLS-printed hold that was topologically optimized, a wall thickness of four voxels was used. The back face is excluded from this shell so that material may be redistributed there during optimization. The remaining voxels inside the geometry that are neither passive void nor passive solid are treated as free design variables.

3.6.4 Load and support setup

The optimization is performed using multiple load cases applied on a prescribed grip region. A rectangular region is defined in physical coordinates and intersected with the voxelized geometry. Only voxels lying on the exterior surface within this region are used for load application.

Table 3.3: Load case definitions for the FEM analysis of the TO hold

LC	Description	Exposed surfaces	Force
1	Pull out from the wall, in the positive z-direction	4,795	1,430 N
2	Vertical downward pull, in the negative y-direction	933	2,200 N
3	Sideward force, in the positive x-direction	2,189	300 N

Three load cases are considered. These represent loading in the negative y-direction, positive z-direction, and the positive x-direction. For each load case, the total load is distributed equally over the nodes of the loaded surface elements. For the printed hold, the loads where defined as stated in table 3.3

Support conditions are defined based on the voxels surrounding the detected hole regions. Voxels adjacent to the identified holes are marked as support regions, and the corresponding nodes are fixed in the finite element analysis.

3.6.5 Optimization procedure

The optimization is based on the density method explained in section 2.7.2 in which each voxel is assigned a density value between 0 and 1 representing the local material amount. The initial density field is set from the prescribed volume fraction within the interior of the geometry. Passive void and passive solid constraints are then enforced so that fixed empty and fixed solid regions remain unchanged throughout the optimization. A density filter is applied to the free design region in order to smooth the density field and control the minimum feature size. The stiffness of each voxel is interpolated from its density using a penalized SIMP-type relation, where low-density voxels contribute very little stiffness.

For each iteration, the global stiffness matrix is assembled and the equilibrium equations are solved for all load cases. The element strain energy is then used to compute the compliance and its sensitivity with respect to the density variables. The sensitivities are filtered and used in an optimality-criteria update scheme, while the prescribed volume fraction of 45% for the free design region is enforced through a bisection search on the Lagrange multiplier. The iterations continue until the maximum density change between two successive iterations falls below a prescribed tolerance or the maximum number of iterations is reached.

3.6.6 Post-processing

After optimization, the density field is converted into a binary solid-void model by thresholding values above or below 0,5. Additional post-processing steps are applied to ensure a physically meaningful and manufacturable structure. First, only material connected to the support region is retained, removing disconnected components

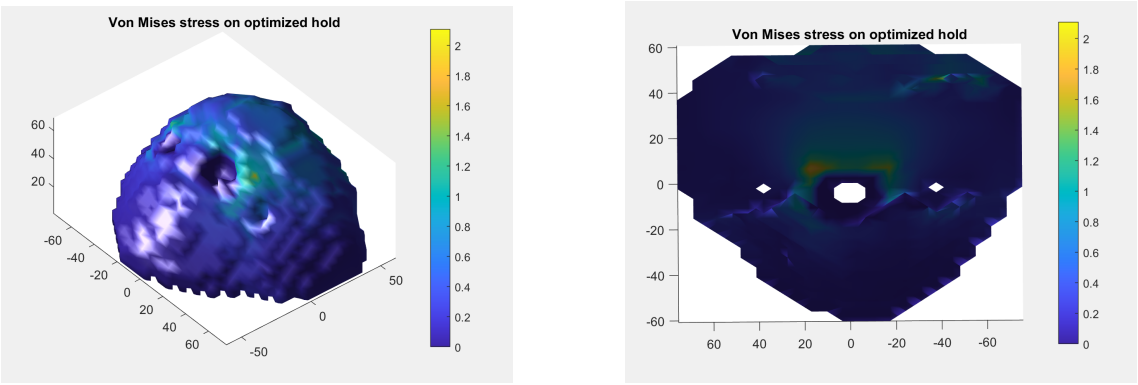
that do not contribute to load transfer. Subsequently, isolated material clusters are eliminated by keeping only the largest connected component. Finally, enclosed voids are removed so that internal cavities not connected to the exterior are filled.

These operations are performed using connectivity-based filtering of the voxel model and ensure that the resulting geometry is structurally continuous and suitable for fabrication. Different connectivity definitions are used depending on the purpose of the operation. For enforcing physically meaningful load paths connected to the support region, 6-connectivity is applied, ensuring that only face-connected voxels are considered connected. For geometric filtering, such as removal of small disconnected regions, 26-connectivity is used to avoid unnecessary fragmentation of the structure (see section 2.7.1). The resulting binary voxel model is then used for visualization and STL export through an isosurface reconstruction.

3.6.7 Validation

After post-processing, the optimized solid-void model is evaluated using a voxel-based finite element analysis explained in section 2.7.3. The final model is interpreted as a structured mesh of eight-node hexahedral elements, where each solid voxel corresponds to one finite element. The element dimensions are obtained from the spacing of the voxel-center coordinates in the x-, y-, and z-directions. Material properties are assigned to all solid elements, while void elements are assigned zero stiffness and therefore do not contribute to the global stiffness matrix.

Support conditions are applied to the material surrounding the detected holes. External loads are applied on a prescribed grip region located on the exterior surface of the model. Only exposed element faces in the load direction are loaded, and the total load is distributed over the corresponding face nodes. The displacement field is obtained by solving the global equilibrium equations. From the element displacements, strains are computed at Gauss integration points using the strain–displacement matrix. Stresses are then obtained from the constitutive relation for linear elasticity.



(a) The hold obliquely from above.

(b) The hold straight from behind.

Figure 3.7: Figures showing the von Mises stress on the topologically optimized hold.

The stress state is evaluated using the von Mises equivalent stress σ_e derived from equation 2.23, the result can be seen in figure 3.7. A safety factor is defined as

$$\text{SF} = \frac{\sigma_s}{\sigma_e} \tag{3.1}$$

where σ_s is set to 40 MPa [79]. Although originally derived for metals, the von Mises stress is used here as an engineering approximation under the assumption of linear elastic material behavior.

3.7 Additive Manufacturing

Prototyping was employed in the project to enable rapid and cost-effective testing of pipeline iterations. Fused Deposition Modeling (FDM) (Section 2.3) was selected as the manufacturing method, as it is an additive manufacturing (AM) technique that uses low-cost materials while allowing for fast production.

The use of an AM method was particularly important, as it enabled the identification of manufacturing-specific challenges early in the design process. For example, the layer-by-layer fabrication approach used in both FDM and Selective Laser Sintering (SLS) handles sharp edges and complex geometries in a similar manner. This made it possible to explore and evaluate surface textures using FDM printing, while still maintaining relevance for potential future production using other AM techniques.

The final manufacturing was carried out using SLS, as it provides significantly better mechanical properties compared to FDM printing. The holds were first printed and subsequently sandblasted to remove residual powder and sharp edges.

3.8 Evaluation and testing

Below, the methods for evaluating and testing the proposed pipeline, as well as the manufactured holds are explained and motivated.

3.8.1 Structural Testing of SLS-printed Holds

To evaluate the structural integrity of the climbing holds and validate their safety, a practical test was performed using four SLS-printed holds. The purpose of the test was to identify potential failure points under representative loading conditions.

The non-optimized holds consisted of two small holds and one large jug and were tested to represent the extremes of size. The smallest holds were tested first, based on the assumption that their reduced size would make them more susceptible to failure. The largest hold was subsequently tested. Larger holds, particularly jugs, introduce increased leverage at the mounting screws, which resulted in higher forces at the attachment points. All three non-optimized holds were subjected to a dynamic load of 105kg. The topologically optimized hold, a large sloper, was subjected to approximately 80kg.

Testing across this range of hold types and sizes ensured that a range of mechanical stresses relevant to practical climbing scenarios was considered, providing a basis for assessing hold safety. This assessment is further supported by analytical calculations presented in section 3.6.7.

3.8.2 Similarity Testing

To evaluate the similarity between the generated and scanned holds, the normalized Chamfer distance was used. This metric computes the average distance from each point in one point cloud to the nearest point in the other point cloud, and vice versa, thereby providing a quantitative measure of geometric similarity. Chamfer distance is commonly used to compare point clouds in applications requiring high geometric precision, such as the 3D holds considered in this study [80]. For completeness, the results were also evaluated using the Earth Mover's Distance (EMD), shown in Figure 4.1e, which provides an alternative measure of geometric similarity by considering the minimal cost of transforming one point cloud into another.

Chamfer distance is sensitive to the number and spatial distribution of points, meaning that differences in point density between scanned and generated data can influence the results [81]. In addition, the metric primarily captures geometric proximity and is less sensitive to topological discrepancies, such as small holes, missing features, or disconnected surfaces. Consequently, two point clouds may exhibit a low Chamfer distance despite differences in fine structural details.

Generated holds may include novel but structurally valid features, which can increase the Chamfer distance without necessarily indicating lower quality or usability. For the Retrieval Based method the Chamfer distance was complemented with anchor drift δ, defined as

$$\delta = s_{\text{nearest}} - s_{\text{anchor}} \tag{3.2}$$

where s_{nearest} is the Chamfer similarity between the generated hold and its closest match in the full dataset, and s_{anchor} is the Chamfer similarity between the gener-

ated hold and its designated anchor. Both similarity scores are derived from the normalized Chamfer distance d_{CH} as

$$s = \max\Big(0,\ \Big(1 - d_{\mathrm{CH}}\Big) \times 100\Big) \tag{3.3}$$

where d_{CH} is the symmetric Chamfer distance between two normalized point clouds, defined in equation 2.8. A drift value of zero indicates that the generated hold remained closest to its original anchor. A positive value indicates that the generated geometry became more similar to a different sample within the dataset than to its designated anchor, suggesting that the output drifted from its intended design reference. Drawing on nearest-neighbor similarity as an established diagnostic in point cloud evaluation [47, 82], this measure provides an indicator of geometric deviation relative to its anchor.

For the LRL method, the Chamfer distance from the generated hold to every hold in the dataset was calculated using similarity search (Section 2.4.2). The Chamfer distances from each nearest neighbor were converted into a similarity percentage, where higher values indicated that the hold was geometrically consistent with its nearest neighbor. To further assess reconstruction quality, each hold was encoded and decoded before the similarity search. This was done both deterministically, only using the latent mean μ, and stochastically, by sampling z around μ using the reparameterization trick (Equation 2.1). Higher similarity percentage indicated that the reconstructed point cloud was geometrically closer to the original input hold.

4

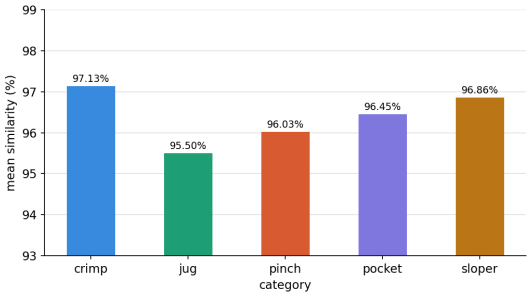
Results

The following chapter presents the results obtained from the previously described methodology. The chapter includes similarity evaluation, topology optimization results, and physical testing results. The similarity was conducted to quantify how closely the generated climbing holds resemble existing geometries while still introducing variation. The evaluation was conducted separately for the two generative approaches: Retrieval-based generation and latent representation learning.

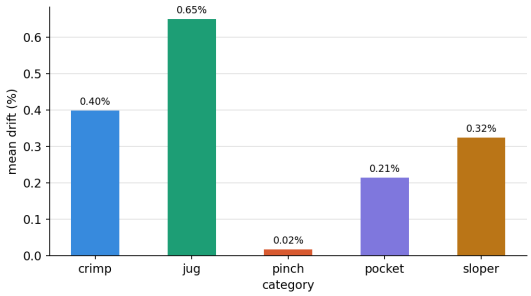
The topology optimization results were evaluated using the von Mises stress (explained in 2.7.3) analysis to identify how the applied loads were distributed throughout the material and to determine the structural performance of the holds. In addition, physical tests were conducted on both optimized and non-optimized holds to evaluate their ability to sustain real-world forces. Finally, several generated holds were used to create a climbing route at Klätterlabbet (Appendix D, figure D.1), where external users were given the opportunity to evaluate and categorize the holds based on their climbing experience.

4.1 Retrieval-based similarity evaluation

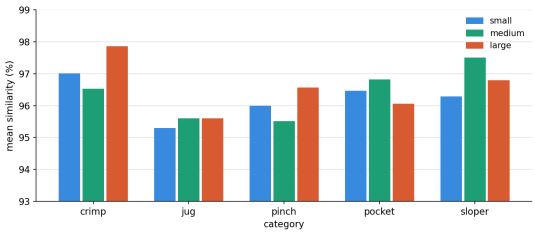
The Retrieval-based generation was evaluated across all five hold categories and three size classes, producing a total of 45 generated point clouds. Each generated hold was compared to its corresponding anchor using the normalized Chamfer distance, as explained in section 2.4.3, between point clouds and additionally evaluated using anchor drift to assess whether the hold remained closest to its designated anchor across the whole dataset.



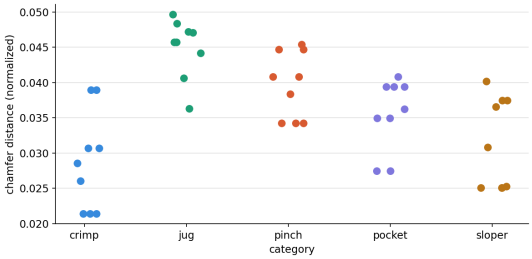
(a) Mean similarity between generated holds and their respective anchors, grouped by hold category.



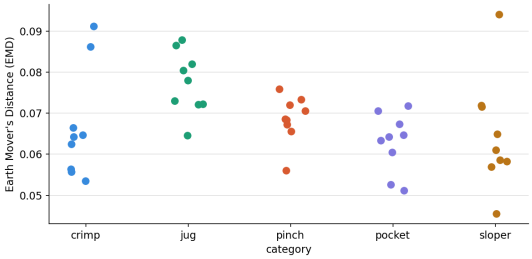
(b) Mean anchor drift by category, zero is the minimum value, positive values indicate that the output has drifted from its intended reference.



(c) Mean similarity to anchor broken down by both category and size class.



(d) Distribution of normalised Chamfer distances across all 45 evaluation runs, grouped by category.



(e) Distribution of normalised EMD distances across all 45 evaluation runs, grouped by category.

Figure 4.1: Summary of Retrieval-based similarity evaluation results.

Anchor similarity was high across all evaluated categories, ranging from 95.50% for jugs to 97.13% for crimps. Slopers and pockets achieved slightly lower mean similarities of 96.86% and 96.45% respectively, with pinches at 96.03%. Jugs exhibited both the lowest mean similarity and the highest within-category variance, with individual runs ranging from 94.69% to 96.37%.

Anchor drift was near zero across all categories, which indicates that the generated holds remained geometrically similar to their designated anchor rather than converging toward other dataset samples. Jugs had the highest mean anchor drift

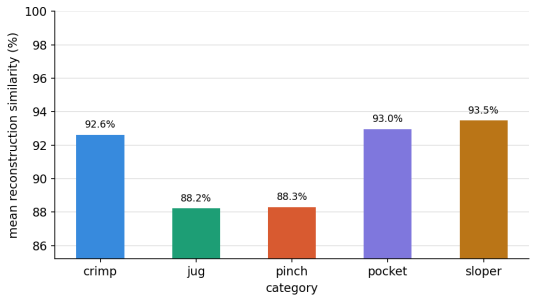
at 0.65%, followed by crimps at 0.40%. Pockets and pinches showed a much lower drift, with pinches as low as 0.02%.

When broken down by size class, mean similarity was generally stable across small, medium and large holds. The most notable exception was crimps, where mean similarity varied by more than 1% between the medium and large size classes. This may reflect a low number of large crimps in the dataset, which reduces the variety of available anchors and consequently constrains the novelty of generated outputs in that subgroup.

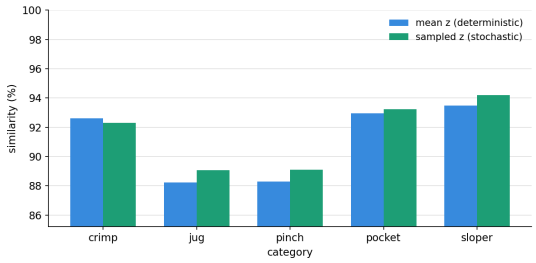
EMD results supported the Chamfer distance findings. Jugs showed the highest mean EMD of 0.0075, further supporting the observation that this category exhibits greater geometric variance than the others.

4.2 Latent Representation Learning similarity evaluation

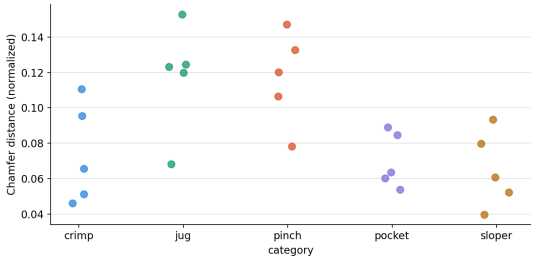
The Latent Representative Learning was also evaluated across all five hold categories and three size classes by encoding each hold into a latent vector and reconstructing it for comparison with the original.



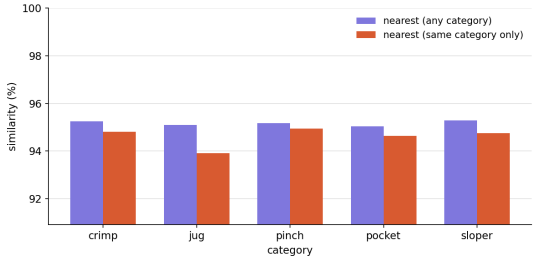
(a) Mean reconstruction similarity per category, decoded from a deterministic latent mean μ.



(b) Comparison of reconstruction similarity between deterministic mean-z decoding and stochastic sampled-z, per category.



(c) Distribution of individual chamfer distance per category for the reconstruction evaluation.



(d) Mean nearest-neighbor similarity of generated holds against the full dataset (purple) and within the correct category only (red).

Figure 4.2: Summary of LRL similarity evaluation results.

Figure 4.2a indicates consistently high reconstruction performance across all categories, with a mean similarity ranging from between 88.2%-93.5%. Slopers achieved the highest similarity 93.5%, while jugs performed worst at 88.2%. The narrow range suggest that the model generalizes well across different hold types.

Reconstruction quality remains stable when comparing deterministic decoding using the mean latent vector (z) with stochastic sampling 4.2b. Sampling marginally outperforms mean decoding across all categories, indicating that the latent space is well structured and can be sampled without affecting output quality.

Variation within categories is further illustrated by normalized Chamfer distance in figure 4.2c shows that there is notable differences within-category variance. Jugs, pinches and crimps exhibit the largest spread, whereas slopers and pockets show relatively low variance which suggests more consistent reconstruction within that category.

Generated samples also closely resemble the training data, achieving nearest-neighbor similarities of approximately 95% overall (figure 4.2d). Similarity within the correct category is slightly lower, approximately 93.9% to 95%, indicating that generated

shapes are both realistic and category consistent.

4.3 Topology optimization

The result in table 4.1 was obtained from a MATLAB based script that uses a SIMP based approach for topological optimization.

Parameter	Value
Geometry & Mesh	
Voxel grid size	$160 \times 100 \times 58$
Voxel size (mm)	$dx = 0.994,\ dy = 0.980,\ dz = 1.027$
Solid voxels (initial)	383530
Material Distribution	
Final solid voxels	251471
Removed material	132059 (34.4%)
Material kept	65.6%
Optimization	
Final objective value	0.7262
Volume fraction (free)	0.550
Iterations	40
Validation Results	
Max displacement	0.2858 mm
Max von Mises stress	23.32 MPa
Yield stress	40.0 MPa
Safety factor	1.72
Critical load case	2
Critical location (mm)	(4.5, 11.3, 35.4)
Conclusion	
Structural requirement	PASS (FoS ≥ 1.0)

Table 4.1: Summary of topology optimization and validation results.

Evaluation of the von Mises stress showed that the critical load case corresponded to the applied downward load of 2200 N. The highest stress concentration was located above the main bolt hole. The resulting safety factor was 1.72, indicating that the computed equivalent stress remained below the assumed yield strength of the material.

4.4 Physical tests

Three of the SLS-printed holds were subjected to a dynamic load corresponding to 105 kg. These non-optimised holds, two small holds and one large jug, passed with-

out showing any visible signs of deformation or structural failure, confirming that the standard pipeline output meets the defined dynamic load requirement of 2.2 kN. The topologically optimised hold exhibited noticeable flexing at approximately 80 kg (785 N), after which testing was stopped on safety grounds. This indicates an inconsistency with the FEA-predicted safety factor of 1.72.

4.5 External user testing

External user testing was conducted to evaluate how well the generated outputs correlate to their respective grip categories. Participants were shown the holds and asked to fill out the perceived category without being told the intended category.

Table 4.2: Retrieval-based Method Results

Hold Type	Size	Answered	Correct	% Correct
Sloper	Small	9	1	11%
Crimp	Small	9	7	78%
Crimp	Large	9	6	67%
Sloper	Large	9	9	100%
Pinch	Medium	9	3	33%
Crimp	Small	9	9	100%
Pinch	Large	9	2	22%
Jug	Small	9	1	11%
Jug	Large	7	5	71%
Sloper	Large	6	0	0%
Jug	Large	7	6	86%
Overall				**52.7%**

The Retrieval-based method achieved an overall recognition rate of 52.7%. Performance varied considerably with respect to category and size. Crimps were the most consistently recognizable at a mean score of 81.6%.

Table 4.3: LRL Method Results

Hold Type	Size	Answered	Correct	% Correct
Crimp	Small	8	5	63%
Jug	Large	9	1	11%
Pinch	Large	9	6	67%
Crimp	Large	9	9	100%
Jug	Medium	9	0	0%
Overall				**48%**

The LRL method achieved an overall recognition rate of 48%. Crimps were the most easily recognized category, whereas jugs proved considerably more difficult to

identify, with only 0% and 11% of the medium and large jugs correctly classified, respectively.

5
Discussion

This study investigated the application of generative design to climbing hold development through two distinct approaches: a Latent Representative Learning generative model and a Retrieval-based latent blending method, both complemented by topology optimization for structural assessment. A third approach, PointFlow, was initially considered but was not pursued due to the time investment required relative to the expected improvement over the two implemented methods. Additionally, PointFlow has been validated on object categories such as chairs and airplanes, and it was unclear whether it would transfer effectively to the more irregular geometry of climbing holds.

The generative design results suggest that both approaches, Retrieval-based and LRL, were capable of producing geometrically coherent and structurally plausible designs. However, their effectiveness must be interpreted in relation to the dataset, evaluation methods, and the limitations of simulation-based validation. Pockets generally produced promising results in 4.2 and 4.1, but the generated hold for both methods exhibited the characteristics of a sloper 2.1e. This may be explained by their surface geometry and latent representation similarities to slopers.

Additionally, there was dissimilarities within the pocket category, such as differences in the number, size, and placement of cavities, which may have introduced excessive geometric variability for the defining pocket characteristics to be consistently preserved in the generated outputs. For the Retrieval-based algorithm, this issue could potentially have been mitigated by selecting more geometrically similar pocket holds or by using a larger dataset containing more distinct subcategories within the pocket category. However, this would likely have reduced the geometric diversity of the generated pockets, resulting in outputs that were more similar to the existing designs they were based on. In contrast, the LRL approach may have been capable of generating more varied, yet still recognizable, pocket geometries if trained on a sufficiently large and representative dataset.

The key distinction between the two approaches lies in their respective generation process. The LRL method enables unconstrained exploration of the latent space through sampling and decoding, allowing for the generation of geometries that are not tied to specific instances in the dataset. This supports a broader range of design variations. However, the results in 4.2 indicate that this flexibility is associated with increased variability in output quality. In particular, the inconsistent category precision and instances of misclassification suggest that the latent space does not fully

separate hold types. This limitation is likely due to the restricted size of the dataset and overfitting, which may restrict the model's ability to capture the diversity of commercially available climbing holds. While the folding decoder promotes smooth reconstruction of the point cloud, it may limit the decoder's ability to recreate complex topologies and sharp geometric features. It could be a reason why holds with more irregular shapes experience loss of local detail or oversmoothing.

In contrast, the Retrieval-based approach provides a more controlled method for generating new geometries. By retrieving similar samples and interpolating between them in latent space, the method constrains outputs to remain close to existing designs. This appears to result in more consistent and stable geometries, which may be advantageous in contexts where reliability and usability are prioritized. However, this constraint also limits the degree of novelty, as the generated designs remain bounded by the characteristics of the input data. The comparison between the two approaches highlights a trade-off between exploration and control, where increased flexibility may reduce stability.

One of the motivations for this work was to explore whether generative methods can help address design fixation. The results suggest that both approaches can contribute to expanding the accessible design space. The LRL enables the generation of new geometries through latent sampling, while the blending method allows for controlled variation of existing forms. Together, these approaches reduce reliance on direct replication and support a more exploratory design process. However, this effect appears to be dependent on the dataset, as both methods are influenced by the distribution and diversity of the training data. As a result, limitations in the dataset may still constrain the range of generated designs.

The similarity results for the retrieval-based method indicate that generated holds generally remain closely aligned with their assigned anchors across all categories, with anchor drift consistently low overall. The highest drift is observed for jugs, likely due to their greater geometric variability in the dataset. In contrast, pinches show the lowest drift, reflecting their relatively uniform and constrained geometry. This geometry results in a tighter geometric cluster within the dataset, which reduces the likelihood that a generated hold will land closer to a neighboring sample than to its own anchor. Despite these category-dependent differences, drift remains low across all classes, indicating controlled within-category generation that stays geometrically close to the reference shape. This balance between similarity and creativity is important for ensuring that the generated designs remain recognizable and usable. At the same time, these metrics primarily capture geometric relationships and do not directly reflect functional performance. The usability of climbing holds depends on factors such as grip comfort, friction, and interaction with the human hand, which are not fully represented in the current evaluation framework.

From a structural perspective, topology optimization and subsequent voxel-based finite element analysis provide an approximate evaluation of material distribution under prescribed loading conditions. This introduces an engineering-based filtering

step in the generative pipeline, linking geometric generation to structural feasibility. However, the analysis relies on several simplifying assumptions.

The structural model assumes linear elastic material behavior and employs a voxel-based discretization with hexahedral elements. Support conditions are approximated by fixing regions surrounding the bolt holes, while external loads are applied to a predefined grip region. These simplifications do not fully represent real-world loading scenarios, which may involve complex contact conditions and dynamic forces during climbing. The grip and support regions are approximated and do not accurately model a hold in use.

The definition of support conditions represents a limitation of the current implementation. Although the intention was to include both the bolt hole and all screw holes as support regions, the code only consistently identifies the bolt hole and partially one screw hole. This indicates that the hole detection and support assignment are not fully robust. However, since the primary load transfer occurs through the bolt connection, and the screw holes mainly serve to prevent rotation, the simplified support representation is considered acceptable for the purpose of relative structural evaluation. Nevertheless, a more consistent and automated support definition would be required for improved accuracy.

Furthermore, structural evaluation is based on the von Mises effective stress, which is traditionally derived for ductile metals. Although used here as an engineering approximation, the applicability to polymer materials such as PA12 is limited, particularly with respect to nonlinear and rate-dependent behavior. As a result, the computed safety factors should be interpreted as indicative rather than predictive.

Additional post-processing steps, including the removal of disconnected material and enclosed voids, further modify the optimized structures. While these operations improve manufacturability and structural continuity, they introduce deviations from the purely optimized topology.

Overall, the structural analysis provides a useful relative measure for comparing generated designs, but its predictive accuracy is constrained by modeling assumptions and the absence of experimental validation. Although the topologically optimized hold that was SLS-printed did hold up structurally, the material showed flexibility. A possible reason for this is that the printed hold was optimized using a force application designed for a jug, even though the hold was used as a pinch. This introduces structural instability as the material was not redistributed to withstand a pinching force from both above and below. Another possible explanation is that the von Mises stress is developed primarily for metals, meaning that it might overestimate the material stiffness.

A limitation of the proposed workflow for topological optimization is the lack of full automation in the definition of loading conditions. The grip region and corresponding load directions are manually specified and are dependent on the specific

geometry and intended use of the climbing hold. While the pipeline successfully processes a given input once these parameters are defined, the selection of appropriate load cases is not trivially generalizable. This introduces a degree of user dependency and limits the extent to which the method can be fully automated. Consequently, although the study demonstrates that generative design and topology optimization can be integrated into a functional pipeline, additional work is required to automate the identification of physically meaningful load and support conditions.

In terms of efficiency, the results suggest that the generative approaches can support faster exploration of design alternatives compared to traditional manual workflows. The ability to generate multiple geometries may reduce the need for iterative sculpting and enable broader consideration of possible solutions. At the same time, the implementation of such a pipeline introduces additional complexity, including data preparation, model training, and parameter tuning. This indicates that while efficiency may be improved in the conceptual phase, it may be offset by increased technical requirements.

Several limitations influence how the results can be interpreted. The relatively small dataset may restrict the representational capacity of the model and limit the diversity of generated outputs. The evaluation methods rely primarily on geometric metrics and limited user feedback, which do not fully capture functional and ergonomic performance. Additionally, the absence of extensive physical validation constrains the ability to assess real-world structural behavior. These factors suggest that the findings should be viewed as indicative rather than definitive.

A further limitation concerns the structure and reliability of the dataset itself. The categorization of climbing holds may have introduced inaccuracies in the dataset due to potential misclassifications. Some holds exhibit characteristics of multiple categories, making distinct classification difficult. Additionally, the categorization was not performed by domain experts, which increases the risk of incorrect labeling due to limited or inconsistent understanding of classification criteria. This may affect the model's ability to make the distinctions between categories.

For the generated and printed holds, there is a possibility that they exhibit characteristics associated with categories other than the one specified in the input. In addition, hold size appears to significantly influence category recognition. Smaller holds were more frequently misclassified and showed a tendency to be perceived as crimps. This suggests that reduced size can make important geometric features harder to distinguish. This was especially apparent for jugs and slopers, where smaller versions were significantly less recognizable. In general, larger holds were identified more accurately, likely because their defining features were more visually pronounced. One exception was a large sloper generated through the Retrieval-based method that performed poorly in user evaluation. This may be explained by it being small, which made it possible to use as a pinch and thus changed the classification. These findings suggest that some classification errors may be influenced not only by generative performance but also by hold size and testing conditions.

Overall, the results suggest that both the LRL method and the blending approach offer useful but distinct capabilities for generative design in this context. The LRL supports exploration of new geometries, while the blending method provides greater control and consistency. This reflects a broader trade-off between creativity and reliability in generative design systems. However, the extent to which these approaches can be applied in practice remains dependent on factors such as dataset size, evaluation methods, and physical validation.

6
Conclusion

The external user testing showed that both generative methods were capable of producing holds that represented several categories, though with varying levels of recognizability. The retrieval-based approach generally maintained clearer category distinctions, while the LRL method showed greater variability. Recognition performance was stronger for certain categories, particularly crimps, while others were less consistently defined. The results also indicate that factors such as size and dataset limitations influenced how effectively category specific features were reconstructed. Overall, both methods demonstrated the potential to generate categorically distinct climbing holds but consistency varied depending on generation approach and category.

The SLS-printed non-optimized holds showed no signs of failure when subjected to a load corresponding to 105 kg, indicating that the holds would likely remain structurally reliable for the target load defined in 3.5. The safety factor of the topologically optimized hold, presented in 4.1, exceeded the formal requirement of FoS > 1. This indicates that the numerical model predicted acceptable structural performance under the prescribed loading conditions. However, the optimized hold exhibited noticeable flexing during physical testing . This suggests that the simplified structural model and von Mises based validation did not fully capture the real mechanical behavior of the printed hold, particularly with respect to stiffness and material behavior.

The evaluation of geometries generated with the Retrieval-based method, based on Chamfer distance and anchor drift, indicates that the outputs maintain a level of consistency with their prompted categories while introducing variation. It was also shown that large crimps fared worse at producing novel outputs, explained by the limited number of large crimp samples in the dataset.

Holds generated by the LRL method achieved high nearest-neighbor similarity, indicating that outputs were geometrically close to existing holds while not being identical. The within-category similarity indicates that the holds remained within its intended category while showing a degree of novelty.

The results demonstrate that generative design and additive manufacturing can be successfully integrated into a functional pipeline for climbing hold development. Although the TO workflow was automated end-to-end after manual definition of grip region and load cases, it was not fully automated or incorporated into the generative

pipeline.

Future Work

While the pipeline shows clear promise as an alternative to traditional modeling, it is constrained by dataset size, load case definitions, and the absence of ergonomic evaluation. Future work should include expanding the dataset and increasing representation across hold sizes and categories. Further evaluation of the autoencoder training process, such as the number of epochs, batch size and beta value, could improve generation quality.

Additional work is needed to automate the topological optimization workflow by investigating the possibility of automated identification of grip regions and application of forces. Incorporating ergonomic evaluation focusing on standardized user testing, hand size and grip strength could further support usability development. Additionally, testing with different AM methods could improve both surface properties and structural performance.

Finally, an economic assessment of scanning holds, manufacturing and lifespan compared to traditional methods would be necessary to evaluate the commercial feasibility of the pipeline.

7
AI usage

For this project AI, or more specifically LLMs, have been utilized for grammar checking and verifying the academic validity of sentence structure. The Chalmers guideline around AI usage for thesis work have been followed and no text has been directly generated. Furthermore, it has been used for explaining programming bugs, and for exploring possible implementation approaches. It has also been used to suggest sources, however, all sources used in the thesis have been manually reviewed and verified for validity. Moreover, it has been used for citation conversion into BibTeX.

Bibliography

[1] Climbing Business Journal, "The hidden industry of hold manufacturing," [Online]. Available: https://climbingbusinessjournal.com/the-hidden-industry-of-hold-manufacturing/, 2016, accessed: Feb. 5, 2026.

[2] D. Keating, "What are climbing holds made of?" 2023, accessed: Apr. 28, 2026. [Online]. Available: https://climbinghouse.com/what-are-holds-made-of-materials/.

[3] D. G. Jansson and S. M. Smith, "Design fixation," *Design Studies*, vol. 12, no. 1, pp. 3–11, 1991.

[4] R. J. Youmans and T. Arciszewski, "Design fixation: Classifications and modern methods of prevention," *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, vol. 28, no. 2, pp. 129–137, 2014.

[5] Anonymous former professional route setter, "Interview a," 2026, private communication, Feb. 2026. Transcript available in Appendix A.

[6] S. Kumar, T. Gopi, N. Harikeerthana, M. K. Gupta, V. Gaur, G. M. Krolczyk, and C. S. Wu, "Machine learning techniques in additive manufacturing: A state of the art review on design, processes and production control," *Journal of Intelligent Manufacturing*, vol. 34, no. 1, pp. 21–55, 2023.

[7] M. Trovato, L. Belluomo, M. Bici, M. Prist, F. Campana, and P. Cicconi, "Machine learning in design for additive manufacturing: A state-of-the-art discussion for a support tool in product design lifecycle," *The International Journal of Advanced Manufacturing Technology*, vol. 137, pp. 2157–2180, 2025.

[8] S. K. Sim and A. H. B. Duffy, "A foundation for machine learning in design," *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, vol. 12, no. 2, pp. 193–209, 1998.

[9] G. Gorup, Lesar, M. Marolt, and C. Bohak, "Procedural point cloud and mesh editing for urban planning using blender," *Land*, vol. 14, no. 4, p. 815, 2025.

[10] Q.-C. Xu, T.-J. Mu, and Y.-L. Yang, "A survey of deep learning-based 3d shape generation," *Computational Visual Media*, vol. 9, no. 3, pp. 407–442, 2023.

[11] Y. Li, G. Lei, G. Bramerdorfer, S. Peng, X. Sun, and J. Zhu, "Machine learning for design optimization of electromagnetic devices: Recent developments and future directions," *Applied Sciences*, vol. 11, no. 4, 2021, art. no. 1627.

[12] Y. Tong, M. Luo, S. Ren, Z. Zhang, C. Xing, and Z. Du, "A rapidly structured aircraft concept design method based on generative artificial intelligence," *Chinese Journal of Aeronautics*, vol. 38, no. 10, p. 103629, 2025.

[13] C. Shinde and D. Garikapati, "Gen AI in automotive: Applications, challenges, and opportunities with a case study on In-Vehicle experience," SAE International, SAE Technical Paper 2026-01-0153, 2025.

[14] F. Alsakka, A. Haddad, F. Ezzedine, G. Salami, M. Dabaghi, and F. Hamzeh, "Generative design for more economical and environmentally sustainable reinforced concrete structures," *Journal of Cleaner Production*, vol. 387, p. 135829, 2023.

[15] F. K. Fuss, Y. Weizman, G. Niegl, and A. Tan, "Climbers' perception of hold surface properties: Roughness versus slip resistance," *Frontiers in Psychology*, vol. 11, 2020.

[16] L. A. Vasconcelos, C. Cardoso, M. Saaksjarvi, C.-C. Chen, and N. Crilly, "Inspiration and fixation: The influences of example designs and system properties in idea generation," *Journal of Mechanical Design*, vol. 139, 2016.

[17] ICP Climbing Walls, "Holdsnbsp;101: Types of climbing holds," 2024, resource article, published Mar. 27, 2024. [Online]. Available: https://climbicp.com/resource-hub/holds-101-types-of-climbing-holds/.

[18] Uncarved Block Pty Ltd, "Climbing hold type glossary: Understand different climbing holds," 2026, climbing hold glossary resource. [Online]. Available: https://www.uncarvedblock.com.au/page/glossary.

[19] Artec 3D, "How does structured-light 3d scanning work?" 2022, accessed: Jan. 30, 2026. [Online]. Available: https://www.artec3d.com/learning-center/structured-light-3d-scanning.

[20] R. Williams, T. Thompson, C. Orr, and G. Taylor, "Developing a 3d strategy: Pipelines and recommendations for 3d structured light scanning of archaeological artefacts," *Digital Applications in Archaeology and Cultural Heritage*, vol. 33, p. e00338, 2024.

[21] Artec 3D, "Artec studio 20: Professional 3d data capture and processing," 2026, accessed: May 5, 2026. [Online]. Available: https://www.artec3d.com/3d-software/artec-studio.

[22] ——, "Artec eva technical specification sheet," product brochure. Accessed: Apr. 30, 2026. [Online]. Available: https://cdn.artec3d.com/pdf/Artec3D-Eva.pdf.

[23] H. T.-P. Doan, K. D. Nguyen, Y.-M. Lin, G. Hong, M.-L. Hsu, and D.-H. Wang, "Comparison of stereolithography (stl) and polygon file format (ply) for intraoral scans: From chairside to archive," *Journal of Dentistry*, vol. 165, p. 106253, 2026.

[24] C. K. Chua, G. K. Jacob, and T. Mei, "Interface between cad and rapid prototyping systems. part 1: A study of existing interfaces," *International Journal of Advanced Manufacturing Technology*, vol. 13, pp. 566–570, 1997.

[25] I. Gibson, D. Rosen, and B. Stucker, "Extrusion-based systems," in *Additive Manufacturing Technologies: 3D Printing, Rapid Prototyping, and Direct Digital Manufacturing*. New York, NY: Springer, 2015, pp. 147–173.

[26] K. Kanishka and B. Acherjee, "Revolutionizing manufacturing: A comprehensive overview of additive manufacturing processes, materials, developments, and challenges," *Journal of Manufacturing Processes*, vol. 107, pp. 574–619, 2023.

[27] I. Gibson, D. Rosen, and B. Stucker, "Powder bed fusion processes," in *Additive Manufacturing Technologies: 3D Printing, Rapid Prototyping, and Direct Digital Manufacturing*. Springer, 2015, pp. 107–145.

[28] E. M. Sefene, "State-of-the-art of selective laser melting process: A comprehensive review," *Journal of Manufacturing Systems*, vol. 63, pp. 250–274, 2022.

[29] Formlabs, Inc., *Fuse 1 User Guide*, 2025, accessed: Feb. 17, 2026. [Online]. Available: https://support.formlabs.com.

[30] A. G. Rodriguez, E. E. Mora, M. A. Velasco, and C. A. N. Tovar, "Mechanical properties of polyamide 12 manufactured by means of sls: Influence of wall thickness and build direction," *Materials Research Express*, vol. 10, no. 10, p. 105304, 2023.

[31] V. Piuri, S. Raj, A. Genovese, and R. Srivastava, Eds., *Trends in Deep Learning Methodologies: Algorithms, Applications, and Systems*. Academic Press, 2021.

[32] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, [Online]. Available: http://www.deeplearningbook.org.

[33] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[34] P. Li, Y. Pei, and J. Li, "A comprehensive survey on design and application of autoencoder in deep learning," *Applied Soft Computing*, vol. 138, p. 110176, 2023.

[35] W.-N. Hsu, Y. Zhang, and J. Glass, "Unsupervised domain adaptation for robust speech recognition via variational autoencoder-based data augmentation," 2017. [Online]. Available: https://arxiv.org/abs/1707.06265

[36] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," 2022. [Online]. Available: https://arxiv.org/abs/1312.6114

[37] K. Y. Chan, B. Abu-Salih, R. Qaddoura, A. M. Al-Zoubi, V. Palade, D.-S. Pham, J. D. Ser, and K. Muhammad, "Deep neural networks in the cloud: Review, applications, challenges and research directions," *Neurocomputing*, vol. 545, p. 126327, 2023.

[38] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, vol. 30, 2017.

[39] Y. S. D. T. Y. Yang, C. Feng, "Foldingnet: Point cloud auto-encoder via deep grid deformation," *arXiv*, pp. 1–5, 2018.

[40] A. Bakshi, P. Indyk, R. Jayaram, S. Silwal, and E. Waingarten, "A near-linear time algorithm for the chamfer distance," 2023, arXiv:2307.03043 [cs.DS]. [Online]. Available: https://arxiv.org/abs/2307.03043.

[41] G. Luo, X. Wang, W. Zhao, S. Tao, and Z. Tang, "Relu neural networks and their training," *Mathematics*, vol. 14, no. 1, p. 39, 2026.

[42] X. Yu, J. Sun, H. Yang, and Q. Zhang, "Self-gated rectified linear unit for performance improvement of deep neural networks," *ICT Express*, vol. 7, pp. 369–373, 2021.

[43] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*. Springer, 2009.

[44] M. Douze, A. Guzhva, C. Deng, J. Johnson, G. Szilvasy, P.-E. Mazaré, M. Lomeli, L. Hosseini, and H. Jégou, "The faiss library," 2024, arXiv:2401.08281 [cs.LG]. [Online]. Available: https://arxiv.org/abs/2401.08281.

[45] Y. Rubner, C. Tomasi, and L. J. Guibas, "A metric for distributions with applications to image databases," in *Proc. Sixth Int. Conf. Computer Vision (ICCV)*, 1998, pp. 59–66.

[46] G. Y. Lu and D. W. Wong, "An adaptive inverse-distance weighting spatial interpolation technique," *Computers & Geosciences*, vol. 34, no. 9, pp. 1044–1055, 2008.

[47] P. Achlioptas, O. Diamanti, I. Mitliagkas, and L. Guibas, "Learning representations and generative models for 3d point clouds," 2018, arXiv:1707.02392 [cs.LG]. [Online]. Available: https://arxiv.org/abs/1707.02392.

[48] I. Caetano, L. Santos, and A. L. ao, "Computational design in architecture: Defining parametric, generative, and algorithmic design," *Frontiers of Architectural Research*, vol. 9, no. 2, pp. 287–300, 2020.

[49] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[50] T. B. Hashimoto, K. Guu, Y. Oren, and P. Liang, "A retrieve-and-edit framework for predicting structured outputs," 2018, arXiv:1812.01194 [stat.ML]. [Online]. Available: https://arxiv.org/abs/1812.01194.

[51] M. Li, X. Yan, B. Lu, Y. Zhang, J. Cheng, and C. Ma, "Attribute filtering in approximate nearest neighbor search: An in-depth experimental study," 2025, arXiv:2508.16263 [cs.DB]. [Online]. Available: https://arxiv.org/abs/2508.16263.

[52] J. Johnson, M. Douze, and H. Jégou, "Billion-scale similarity search with gpus," 2017, arXiv:1702.08734 [cs.CV]. [Online]. Available: https://arxiv.org/abs/1702.08734.

[53] G. Yang, X. Huang, Z. Hao, M.-Y. Liu, S. Belongie, and B. Hariharan, "Pointflow: 3d point cloud generation with continuous normalizing flows," *IEEE*, 2019.

[54] C. Arnold, L. Riß, J. Hey, and R. Schweyen, "Dimensional accuracy of different three-dimensional printing models as a function of varying the printing parameters," *Materials*, vol. 17, no. 14, p. 3616, 2024.

[55] Y. Liu, A. Obukhov, and J. Wegner, "Point2cad: Reverse engineering cad models from 3d point clouds," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024, pp. 3763–3772.

[56] X. Wang *et al.*, "Frame points attention convolution for deep learning on point cloud," *Scientific Reports*, 2025.

[57] N. Abreu, A. Pinto, A. Matos, and M. Pires, "Procedural point cloud modelling in scan-to-bim and scan-vs-bim applications: A review," *ISPRS International Journal of Geo-Information*, vol. 12, no. 7, p. 260, 2023.

[58] J. Lu, Y. Wang, Y. Wu, H. Li, Y. Shi, and F. Ning, "An autoregressive framework for reconstructing editable parametric computer-aided design models from point clouds," *Engineering Applications of Artificial Intelligence*, vol. 163, p. 113107, 2026.

[59] D. Michelucci and S. Foufou, "Geometric constraint solving: The witness configuration method," *Computer-Aided Design*, vol. 38, no. 4, pp. 284–299, 2006, symposium on Solid and Physical Modeling 2005.

[60] Blender Foundation, "Geometry nodes," 2026, blender Manual. [Online]. Available: https://docs.blender.org/manual/en/latest/modeling/geometry_nodes/index.html. Accessed: Feb. 5, 2026.

[61] B. Curless and M. Levoy, "A volumetric method for building complex models from range images," in *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '96)*, 1996, pp. 303–312.

[62] M. Kazhdan, M. Bolitho, and H. Hoppe, "Poisson surface reconstruction," in *Proceedings of the Eurographics Symposium on Geometry Processing*, 2006, pp. 61–70.

[63] M. Berger, A. Tagliasacchi, L. Seversky, P. Alliez, G. Guennebaud, J. Levine, A. Sharf, and C. Silva, "A survey of surface reconstruction from point clouds," *Computer Graphics Forum*, vol. 36, 2017.

[64] A. Nikonov, M. Nagode, J. Klemenc, and I. Emri, "Effect of weight on dynamic ropes' responses in indoor climbing," *Proceedings of the Institution of Mechanical Engineers, Part P: Journal of Sports Engineering and Technology*, 2024. [Online]. Available: https://doi.org/10.1177/17543371241265997

[65] I. P. Rosinha, K. V. Gernaey, J. M. Woodley, and U. Krühne, "Topology optimization for biocatalytic microreactor configurations," in *12th International Symposium on Process Systems Engineering and 25th European Symposium on Computer Aided Process Engineering*, ser. Computer Aided Chemical Engineering. Elsevier, 2015, vol. 37, pp. 1463–1468.

[66] M. Aleksandrov, S. Zlatanova, and D. J. Heslop, "Voxelisation algorithms and data structures: A review," *Sensors*, vol. 21, no. 24, 2021, art. no. 8241.

[67] S. Cui, S. Zhang, X. Chen, Z. Pang, X. Fu, and X. Zhang, "Point-in-polyhedra test with direct handling of degeneracies," *Geo-spatial Information Science*, vol. 14, no. 2, pp. 91–97, 2011.

[68] T. Zegard and G. H. Paulino, "Bridging topology optimization and additive manufacturing," *Structural and Multidisciplinary Optimization*, vol. 53, no. 1, pp. 175–192, 2016.

[69] M. P. Bendsoe, "Optimal shape design as a material distribution problem," *Structural Optimization*, vol. 1, pp. 193–202, 1989.

[70] M. Zhou and G. I. N. Rozvany, "The coc algorithm, part ii: Topological, geometrical and generalized shape optimization," *Computer Methods in Applied Mechanics and Engineering*, vol. 89, pp. 309–336, 1991.

[71] J. Li, W. Yao, and Y. S. et al., "An integrated simp-emsfem approach for concurrent multiscale optimization of compliance and stress in topology design," *Structural and Multidisciplinary Optimization*, vol. 68, p. 223, 2025.

[72] D. L. Logan, *A First Course in the Finite Element Method*, 4th ed. Thomson Canada Ltd., 2007, pp. 7–15.

[73] A. A. Roostaei and H. Jahed, *Cyclic Plasticity of Metals: Modeling Fundamentals and Applications*. Elsevier, 2022.

[74] L. Andersson, K. Cognell, A. Rekstad, K. Strandberg, T. Söderstjerna, and A. Warg, "Climbing hold generation pipeline," GitHub repository, 2026, available on: https://github.com/theaa20/EENX16_VT26_12A.

[75] NumPy Developers, "numpy.lib.format — numpy v2.5.dev0 manual," 2026, [Online]. Available:

https://numpy.org/devdocs/reference/generated/numpy.lib.format.html. Accessed: May 10, 2026.

[76] S. van der Walt, S. C. Colbert, and G. Varoquaux, "The numpy array: A structure for efficient numerical computation," *Computing in Science & Engineering*, vol. 13, no. 2, pp. 22–30, Mar. 2011.

[77] J. Sparte, "Bhtoolset," GitHub repository, 2023, accessed: Feb. 17, 2026. [Online]. Available: https://github.com/JeremSparte/BHToolset.

[78] L. Hanson, L. Sperling, G. Gard, S. Ipsen, and C. O. Vergara, "Swedish anthropometrics for product and workplace design," *Applied Ergonomics*, vol. 40, no. 4, 2009.

[79] Plastshop.se, "Technical data sheet: Pa 12," 2021, published: Sep. 5, 2021. Accessed: May 8, 2026. [Online]. Available: www.teamlenom.se.

[80] P. Arcano-Bea, A. García-Fischer, P. P. Gómez-González, F. Zayas-Gato, J. L. Calvo-Rolle, and H. Quintián, "Deep learning-based 3d reconstruction for defect detection in shipbuilding sub-assemblies," *Sensors (Basel)*, vol. 26, no. 2, p. 660, 2026.

[81] M. Lapenna, F. Faglioni, K. Chand, B. Hejazi, R. Fioresi, and G. Bruno, "Chamfer distance for non-linear registration of triply periodic minimal surface lattices," *Additive Manufacturing Letters*, vol. 14, p. 100299, 2025.

[82] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao, "3d shapenets: A deep representation for volumetric shapes," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 1912–1920.

A

Interview with former professional route setter

Date: 10th Feb 2026
Role: Climbing gym employee and former professional route setter
Format: Semi-structured interview

Excerpt

Q: Finns det några problem med greppen ni har idag?
A: Ja, kvaliteten är inte alltid bra. Fel i färgen, de skär sig vilket resulterar i att vi måste skicka tillbaka dem.
Q: Många upplevs liknande, upplever du det också?
A: Inte mycket men ibland. Det ändras också mycket med åren när det kommer nya stilar. Exempelvis så har storlekarna blivit större allt eftersom.
Q: Som i comp-klättring?
A: Ja precis, gamla grepp är ju oftast mycket mindre. Sen finns det strukturväggar.
Q: Är det ett problem när de är lika?
A: Ja det kan det vara. Vissa är duktiga på att snappa upp trender som Euroholds, ett annat märke är också X-Cult.
Q: Vilka parametrar hade du velat prompta i ett promptbaserat system?
A:

- Färg är viktigt för att de ska bli ett set
- Grad (svårighetsgrad)
- Vinkel på greppet
- Distanser mellan greppen
- Storlek på greppen
- Vissa regler, t.ex. inga pockets på grad 4–5

B

Interview with professional route setter

Date: Feb 20th 2026
Role: Climbing gym employee and professional route setter
Format: Semi-structured interview

Excerpt

Q: Finns det några problem med greppen ni har idag?
A: Det finns inte så mycket frihet hos ledbyggaren, utan mest hos greppföretagen. Vi kan lösa det genom att producera grepp "in house". Vi hade kunnat fixa mindre grepp själva och köpa större grepp av tillverkarna. Jag hade velat ha möjlighet att skapa egna grepp och fixa nya om något går sönder. Vi hade också sparat pengar på att kunna göra egna grepp, vilket är bra ur ett företagsperspektiv. Fördelen är att det kommer inte krävas någon designutbildning, utan man behöver bara förståelse för vilket grepp som behövs. Frågetecknet gäller mest färgerna, kan man göra olika färger? Det är en stor grej att kunna göra det. Måste man isåfall städa ut skrivaren mellan? Jag hade velat kunna städa ur den på max 1-2 timmar. .
Q: Många upplevs liknande, upplever du det också?
A: Ja det finns basformer, vilket inte är ett stort problem. De finns av anledning, de är snälla mot händerna. Men det är kul med variation, bara att basformerna är en viktig grundpelare.
Q: Är det ett problem när de är lika?
A: Många tillverkare har en stor repertoar av grepp, man väljer tillverkare utifrån kvalité. Finns inte ett underskott av tillverkare. Det är ett högt slitage på greppen, så kvalité är den största faktorn. Core håller länge, efter ett dussin år är de fortfarande lika bra.
Q: Vilka parametrar hade du velat prompta vid ett prompting baserat system?
A: Incut variabel hade varit bra, det gör stor skillnad för greppet.
Q: Vid ett eventuellt user interface, hur hade du velat ha det?
A: Clickar boxar är enklare. För det tydligare, tar bort ett lager i modellen. Om man har en chatt får man en känsla av störrre kontroll, fast det egentligen är otydligare. Enkelhet. Om textbaserat innebär mer frihet hade det varit najs. Men han tror inte det kommer sluta så.

C

User interface

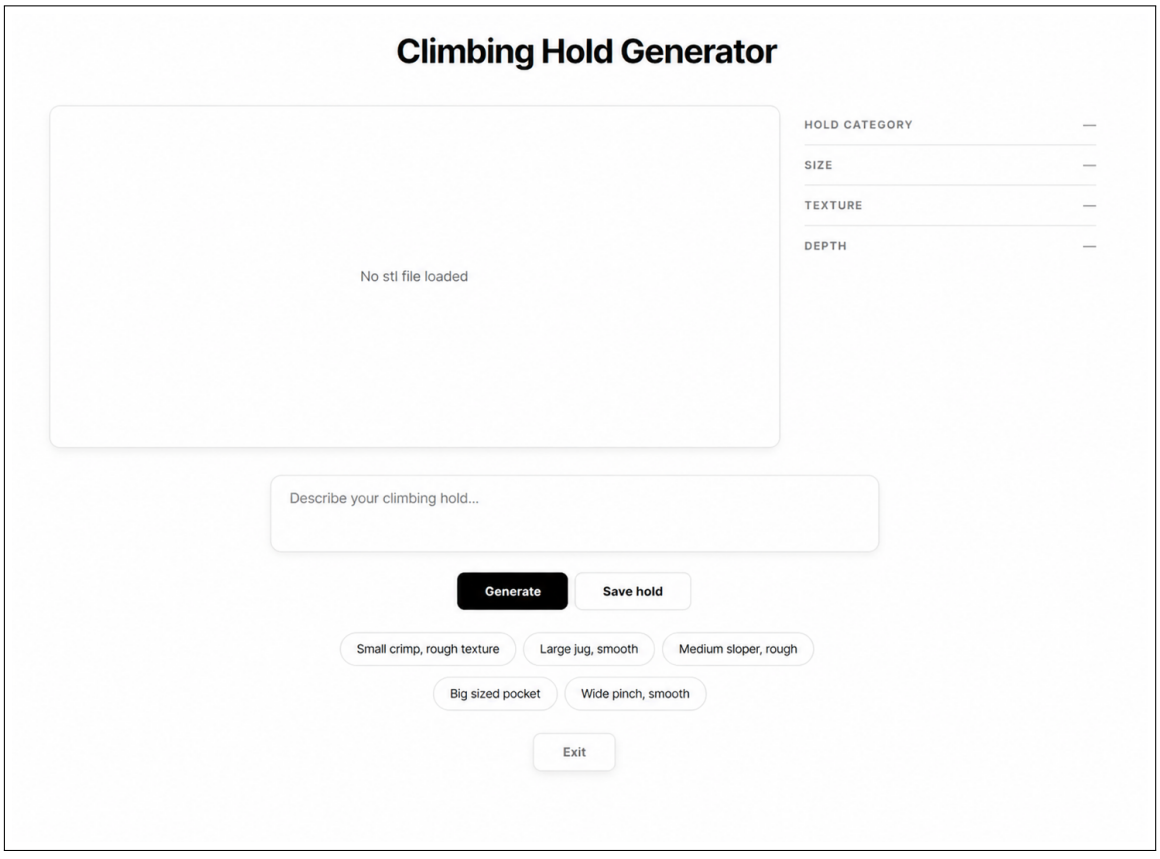


Figure C.1: The text-based UI

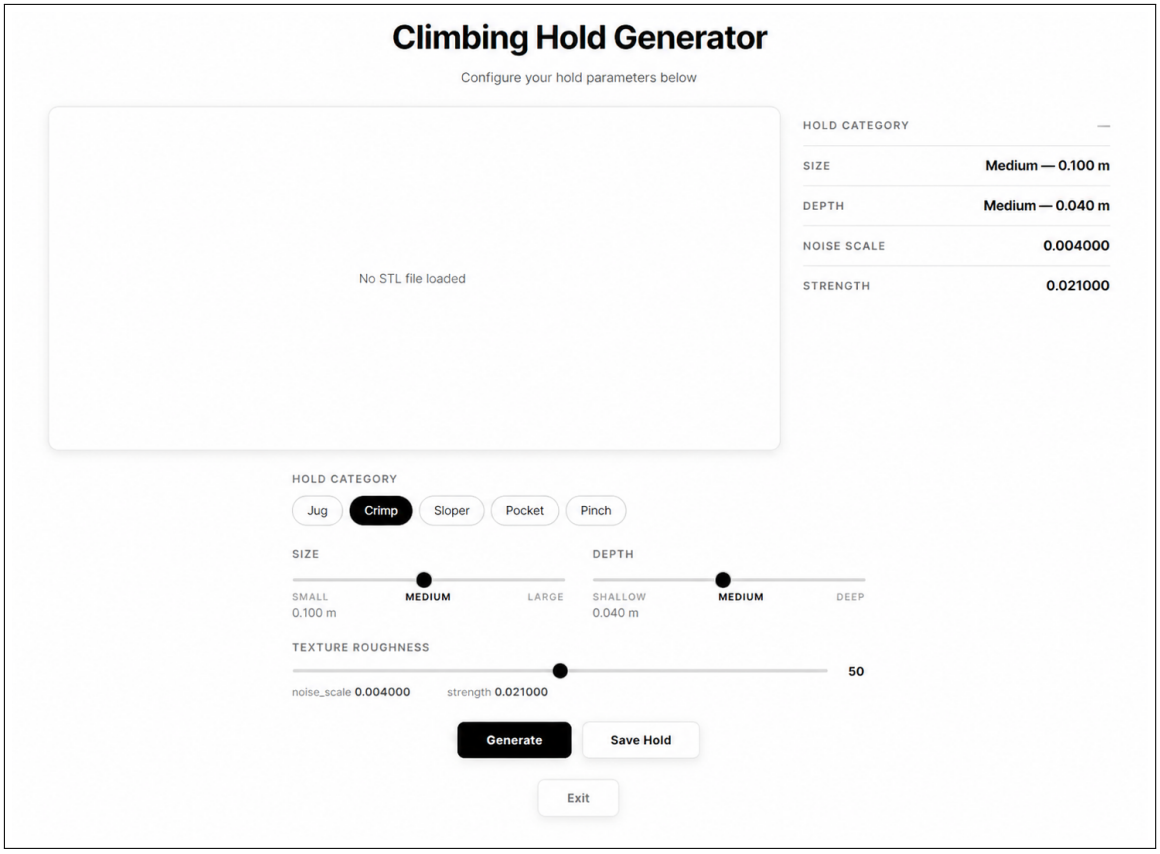


Figure C.2: The selection-based UI

D

Climbing route

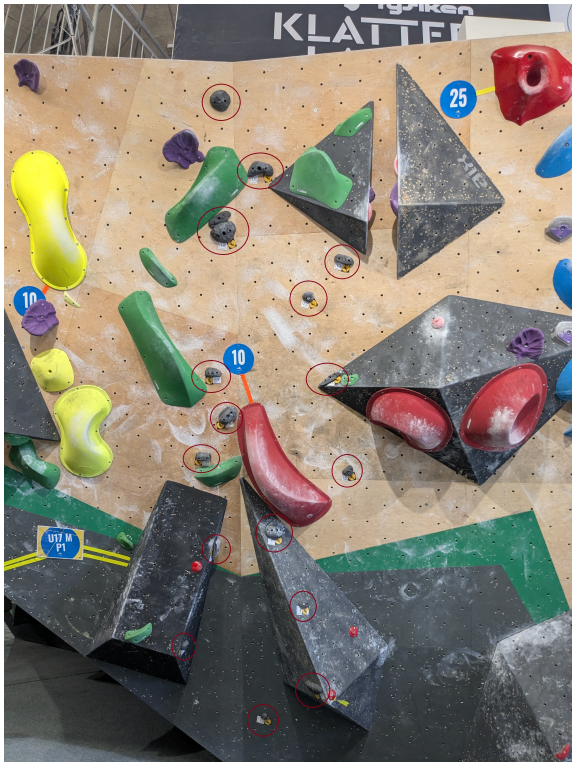


Figure D.1: A route with the generated holds

DEPARTMENT OF ELECTRICAL ENGINEERING
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS