Manufacturable Shape Reconstruction using Neural Networks

Longlai Qiu
Supervisor: Pierre Yvernay, Prof. Pascal Fua
CVLab, École Polytechnique Fédérale de Lausanne
longlai.qiu@epfl.ch

***Abstract*—The use of neural networks over the last decade have been employed in many industries and have had a significant influence in day to day life. One area in which its use is being recently explored is within the realm of manufacturing for topology optimization (TO). Over the last years, there has been a surge in the use of additive manufacturing (AM) processes for greater design freedom in a product's lifecycle. However, the reliance of AM have specific drawbacks such as computational time, need for human intervention and lack of economy of scale. As a result, we introduce the use of neural networks to address such limitations and to provide greater freedom in the selection of manufacturing processes such as casting, forging and machining. We utilize a two step network that trains on a 2D dataset followed by an optimization network in attempt to ensure that the reconstructed shape is manufacturable. We implicitly represent the shape's boundary using a signed distance function which allows us to represent a vast array of different shapes. Finally, we demonstrate qualititative results for each manufacturing constraint as outlined in this report.**

I. Introduction

Deep learning has made a profound impact on many industries including medical imaging, robotics and linguistics. One area in which neural networks are recently being explored is within the field of manufacturing. The manufacturing process begins with the product design and material selection which is then modified based on the desired manufacturing method to achieve the final product.

Traditional manufacturing methodologies have a long history of encompassing different processes including casting, extrusion, forging and many more. More recently, the use of additive manufacturing (AM), more commonly known as 3D printing, has given rise to a new field of possible manufactured parts. 3D printing constructs three-dimensional objects from a digital computer-aided design (CAD) model by depositing material layer by layer. This contrasts traditional methods which are coined as *subtractive manufacturing* and has given engineers greater freedom from the initial design phase. Due to the nature of material deposition, AM has enabled production of complex geometries that would not be feasible with the use of subtractive manufacturing and is subject to much fewer geometric constraints than its counterpart.

The first 3D printer was invented by Charles W. Hull in the mid 1980s [1]. Although, it wasn't until the 2000s when 3D printing made major breakthroughs in manufacturing of goods. Simultaneously, the emergence of AM has brought further attention to the use of topology optimization during design iteration. Topology optimization (TO) is a mathematical method that optimizes material layout for given design constraints and an objective function criteria to maximize or minimize. As a result, AM is able to take advantage of the more efficient designs that TO can offer.

However, there are still areas of improvement that exist for both TO and AM. Firstly, the designs that TO can offer may still be difficult to manufacture even with the freedom provided by AM. This requires human intervention and failure to check for manufacturability of the TO design prior to production may compromise build quality and efficiency. Secondly, for large scale TO tasks, the requirement to iteratively solve millions of equilibrium points remains a computational challenge [2]. Finally, AM's economy of scale is subpar when compared to traditional manufacturing methods [3] and is thus limited to low production quantities.

In this report, we address the limitations in both TO and AM. Specifically, we utilize a fully connected neural network for shape reconstruction of geometries that are manufacturable under traditional processes. We build upon the work outlined in DeepSDF [4], but for 2D generative modelling of shapes instead. The work utilizes the concept of Signed Distance Functions (SDF) to represent surfaces of shapes by a continuous field. An SDF is a function that states for a given point, returns the distance of the point to the closest surface boundary. The sign of the SDF indicates whether it is inside (-) or outside (+) from the surface. This is further depicted in Figure 1.

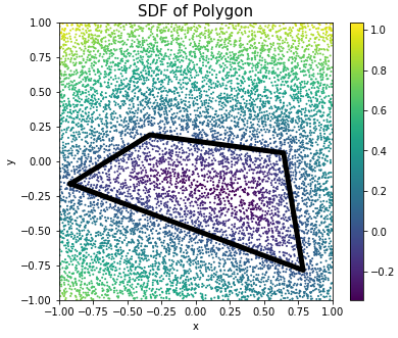


Fig. 1: SDF of points sampled from a random uniform distribution.

Similarly to the work in DeepSDF, we formulate 2D shapes

using an implicit representation and employ a learning based method using an auto-decoder. The SDF model is trained on a dataset of randomly generated shapes and take as input 2D points and their corresponding SDF values for each shape. The output of the model consists of a latent vector feature space that maps to a distribution of various shapes in 2D. Henceforth, we showcase an optimization network that is able to generate shapes that are not only optimal, but ideally also manufacturable under traditional processes by penalizing the network directly.

More concisely, the main contributions of this project are three-fold:

1) We perform extensive literature review on constraints that exist within traditional manufacturing methods.
2) We formulate the work outlined in DeepSDF onto our 2D training dataset.
3) We apply manufacturing constraints to an optimization network in attempt to guarantee that the final rendered shapes are manufacturable.

The report is organized as follows. In Section II, we discuss literature surrounding shape optimization under traditional manufacturing processes, namely; casting, extrusion and machining. In Section III, we briefly discuss relevant work related to shape and representation learning. In Section IV, we discuss the training dataset in addition to necessary data preperation steps. In Sections V and VI, we discuss the neural network model used in DeepSDF and in more detail our optimization network, respectively. Finally, in Sections VII and VIII, we discuss experimental results and present our conclusions alongside future work.

II. Literature Review

Topology optimzation is a relatively new and expanding area of research within the field of structural mechanics. Among the many techniques that appear in literature, the two most prevalent TO methods are the Solid Isotropic Material with Penalization (SIMP) technique and the Evolutionary Structural Optimization (ESO) technique. A common TO example is the cantilever beam shown in Figure 2.

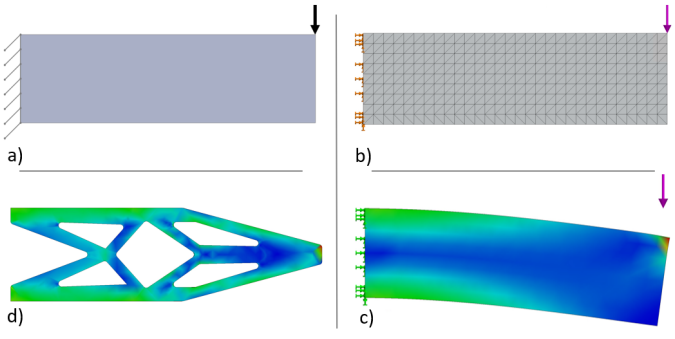


Fig. 2: Typical topology optimization process starting with a a) defined problem, b) meshed surfaces, c) structural analysis and d) resultant optimal shape.

The SIMP method is based on varying material density within a predefined domain to determine the optimal structure [5]. The predefined domain is discretized into elements for which finite element analysis (FEA) is to be applied onto the geometry. Typically, FEA is used to reveal regions of high stress on the part and this allows the optimization procedure to seek the optimal density of each element under some sort of optimality criteria. This is an iterative process allowing various configurations of the initial shape until final convergence.

The ESO method is another similar approach that is reliant on FEA [6]. After the stress distribution of the geometry is found using FEA, material of the structure is progressively removed according to a rejection criteria. This is a relatively simple and straight forward approach in which the rejection criteria can be a ratio of the discovered maximum stress value. Again, this is repeated until steady state is achieved. This technique is frequently used in CAD packages such as ANSYS [7].

In the following subsections, we will further expand on literature related to topology optimization for manufacturing methods of interest. Further, we highlight key formulas that are relevant to our level-set method.

A. *Casting*

Casting is a fabrication process in which molten liquid (typically metal) is poured into a cavity formed by molds [8]. After a period of time, the liquid will cure and solidify to take its final shape defined by the cavity design. The desired product is achieved after removal of the mold. As such, the design of the desired part should possess a geometry such that removal of the mold is feasible without the need of breaking it. This soft requirement is referred to as the molding constraint.

Different topology optimization methods have been proposed to address the molding constraint in casting. Within the SIMP framework, [9] introduces a penalization scheme that favors higher densities at lower regions of the structure. A more in depth review of such topology optimization methods can be found in [10].

In this report, we will focus on the molding constraint using the level-set method for TO. While previous work in [11], [12] have also focused on using level-sets, the proposed method only focuses on initial shapes that are castable from the start. We showcase later that this is not necessary when using a neural network. Instead, we formulate our problem using SDF and penalize the optimization network using a series of constraints in attempt to achieve an optimal geometry that is also castable.

For the setting of our problem, we recall that our goal is to optimize a shape $\Omega \subset \mathbb{R}^N$, where $N = 2$ for our 2D scenario. We introduce D as the bounded domain which contains all admissable shapes such that $\Omega \subset D$. Further, we define a linear isotropic elastic material for our shape with Hooke's law A, which is a positive definite fourth-order tensor. The displacement field u is the unique solution to the linearized elastic system, where $e(u)$ is the strain tensor, also defined as the gradient of u. A typical objective function for the

optimization problem is to minimize compliance $J(\Omega)$ during loading and is the following:

$$J(\Omega) = \int_{\Omega} Ae(u) \cdot e(u) dx \tag{1}$$

A common shape optimization problem can be formulated as:

$$\inf_{\Omega \in U_{ad}} J(\Omega) \tag{2}$$

where U_{ad} is the set of all admissable shapes that satisfy the volumetric (area for 2D problems) constraint $0 < V < |D|$. However, the existing problem falls short in converging to an optimal solution and requires further restrictions by imposing additional constraints to the problem. One such way to advect the geometry is to rely on Hadamard shape differentiation. The mathematical formulations will not be discussed and can be explored further in [8], [13].

While several molding constraints exist within literature, we will focus on the most criterion; the removal of molds. Several important terminologies are needed to be explained before proceeding. First, the draw direction $\vec{d}$ refers to the direction in which the mold(s) is separated from the parting surface. The parting surface is the plane in which molds come together into contact to form the desired cavity of the part. We show an example of two identical parts in Figure 3 that is both castable and non-castable due to the direction of $\vec{d}$.

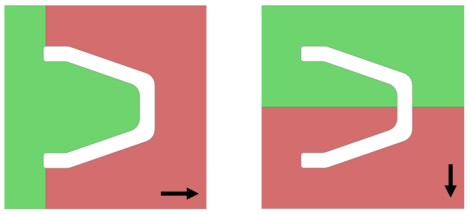


Fig. 3: Castable (left) vs. non-castable (right) process.

Apart from process choices related to draw direction and parting surface, meticulous design details are to be avoided to ensure castability of a part. Specifically, the avoidance of undercuts and internal voids are necessary. An undercut is an indentation of the surface boundary $\partial\Omega$ that prohibits ejection of the molds. Simultaneously, an internal void is a casting defect that results in an undesired porous region of the geometry. Both defects are depicted in Figure 4.

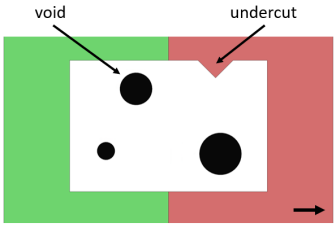


Fig. 4: Defects present in casting process.

The boundary surface $\partial\Omega$ can be divided into m number of parts Γ_i, such that $i = \{1, ..., m\}$, and can be drawn in direction $\vec{d_i}$. Hence, an imposing draw direction constrained can be defined as:

$$\boxed{\vec{d_i} \cdot \vec{n}(x) \geq 0, \forall x \in \Gamma_i} \tag{3}$$

where $\vec{n}(x)$ is the external unit normal at $x \in \partial\Omega$. We see in Figure 4 that the presence of such defects do not satisfy the constraint described in Equation (3). Despite the simplicity of such constraint, there are two major drawbacks that exist. First, the optimization problem requires the initial shape to be castable from the start, thus avoiding shapes that inhibit uncercuts or voids. Second, there does not exist an advection velcotiy that is normal to the parting surface, hence not all deformable shapes are achievable. The advection velocity is the process of transporting a substance or quantity governed by a vector field [8]. For example, a limitation present in casting is the inability for a shape to expand in the direction that is normal to the parting surface. To address this concern, a more generalized draw direction constraint is given in Equation (4) with the formal definition of the $SDF(\cdot)$ shown in Equation (5).

$$\boxed{SDF(x + \zeta \vec{d_i}) \geq 0, \forall x \in \Gamma_i, \forall \zeta \in [0, dist(x, \partial D)]} \tag{4}$$

$$\boxed{SDF(x) = \begin{cases} -dist(x, \partial\Omega) & \text{if } x \in \Omega \\ 0 & \text{if } x \in \partial\Omega \\ dist(x, \partial\Omega) & \text{if } x \in (\mathbb{R}^N / \bar{\Omega}) \end{cases}} \tag{5}$$

In simple terms, Equation (4) states that starting from a point on boundary $\partial\Omega$, we travel along direction $\vec{d_i}$ and it should hold that we no longer encounter any part of the original shape. For our optimization network, we will address later on that there is no need to use the generalized form of Equation (4), and can instead use the simpler form of Equation (3). One notable advantage to using the simpler form, is that we can formulate a penalization function as follows:

$$\boxed{P(\Omega) = \int_{\partial\Omega} [\min(\vec{d} \cdot \vec{n}(x), 0)]^2 dx} \tag{6}$$

Again, the shape differentiation procedure will not be discussed in the context of this report and can be found in [8]. We will introduce a similar penalization function to our optimization network in a later section.

B. Extrusion

Extrusion is a bulk metal forming process in which a metal billet is forced under pressure through a die to reduce its cross sectional area [14]. In extrusion, the contact forces between the billet and die exhibit high regions of stress in order to deform the part to take shape defined by the die. An extrusion manufacturing process is demonstrated in Figure 5.

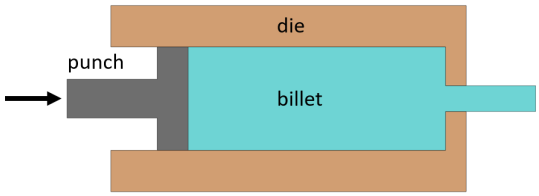


Fig. 5: Extrusion process.

The most important criterion related to extrusion is the requirement of a constant cross section along the extrusion direction [15]. The work in [16] introduces a projection technique to identify undesirable solutions during optimization. The necessary steps are as follows:

1) Map the relation between design variables τ and pseudo-densities ρ domains.
2) Project τ onto ρ.
3) Perform sensitivity analysis of ρ in relation to τ to determine desired design variables.

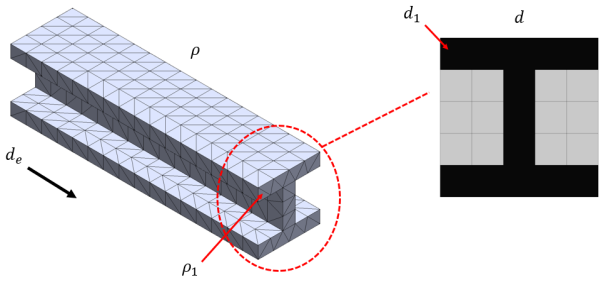


Fig. 6: Extrusion constraint design domain (left) and design variables (right).

In regards to extrusion, the original geometry Ω to be extruded is mapped along a reference plane that is normal to the extrusion direction. The mapping relation identifies which elements in the pseudo-density domain ρ are influenced by design variables τ_i. Given the need for a constant cross section throughout the geometry, the design variables τ_i for the model is the cross section area. All elements contained in the extrusion direction are to be included within the design domain and projected into pseudo-densities. An example of this phenomenon is given in Figure 6. Several mapping techniques are addressed in [16] depending on the manufacturing method. One example for mapping the pseudo-density domain for a symmetry constraint is to mirror the design variables along a plane as shown in Figure 7. Another example shown in Figure 8 takes each design variable τ_i and protrudes along +X as the extrusion direction.

The second step of the process adopts a function that projects τ_i to ρ_j during each iteration of the TO process. This is found using a differentiable q-norm to find the maximum value for a set of design variables τ_i and defined in Equation (7). In other words, after mapping the relation between τ and ρ, we are only interested in the largest value in the set of variables τ_i for projecting onto ρ_j. Thus, we convert τ_i to ρ_j

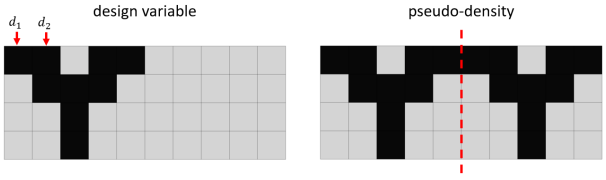


Fig. 7: Mapping design variable to pseudo-density for symmetry constraint.

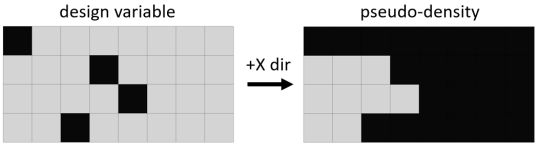


Fig. 8: Mapping design variable to pseudo-density along $+x$ extrusion direction.

at each iteration to evaluate the objective and constraint values of the optimization problem.

$$\rho_j = \left(\sum_{i \in \Omega} \tau_i^q \right)^{1/q}, q > 0 \tag{7}$$

Recall that another approach in defining the optimization problem for extrusion that differs Equation (2) in using pseudo-densities can be formulated given objective function $W(\rho)$ as follows:

$$\begin{aligned} &\min\ W(\rho) \\ &\text{subject to } g_i(\rho) \leq \bar{g}_i \\ &\qquad \rho_j = \rho_{j+1} = ... = \rho_M \end{aligned} \tag{8}$$

where $g_i(\rho)$ is some i-th constraint bounded by upper limit $\bar{g}_i$. The second constraint states that all pseudo-densisties must be equal for a constant cross section to exist for extrusion. Once ρ is found for a single iteration, the final step is to perform sensitivity analysis to find a new set of design variables τ_i. Specifically, we differentiate the objective function with respect to the design variables as shown in Equation (9). We then update both the design variables, and hence associated pseudo-densitites after a single iteration and repeat this process until convergence.

$$\frac{\partial W(\rho)}{\partial \tau_i} = \sum_{j \in \Omega} \frac{\partial W(\rho)}{\partial \rho_j} \frac{\partial \rho_j}{\partial \tau_i} \tag{9}$$

Recall from Equation (7) that the differentiable q-norm function can be solved analytically using:

$$\frac{\partial \rho_j}{\partial \tau_i} = \tau_i^{q-1} \left(\sum_{i \in \Omega} \tau_i^q \right)^{1/q-1}, q > 0 \tag{10}$$

In summary, the methodology outlined in [16] first maps the relation between τ and ρ depending on the manufacturing method. For extrusion, τ is the cross sectional area and is

converted into equivalent finite elements ρ using Equation (7). The ρ values are solved for at each iteration of the optimization problem outlined in Equation (8). Further, the parameters τ and ρ of the optimization problem are updated using sensitivity analysis in Equation (9). This evaluates the influence of the parameters on the overall objective function.

We will show later on that discretization of the working domain into pseudo-densities is not necessary when working with level-sets. Instead, we apply a uniform cross-sectional constraint using a similar approach as Equation (3). We state that along extrusion direction $\vec{d_e}$, there cannot exist a non-zero component along the surface boundary as shown in Equation (11).

$$\boxed{\vec{d_e} \cdot \vec{n}(x) = 0, \forall x \in \partial\Omega/\partial D} \tag{11}$$

Again, we can apply a similar penalization scheme as in Equation (6) as follows:

$$\boxed{P(\Omega) = \int_{\partial\Omega/\partial D} (\vec{d_e} \cdot \vec{n}(x))^2 dx} \tag{12}$$

C. *Machining*

Machining is a broad manufacturing term used to describe removal of material from a workpiece using a variety of tools to achieve the desired structure. Some common examples include cutting, drilling and milling [17]. Generally speaking, machining is a process that follows casting or extrusion as a post-processor in order to achieve better dimensional accuracy and geometrical tolerances. However, drawbacks of the process include higher capital, labor and manual methods than other operations. Fortunately, with the aid of computer numerical control (CNC), many of the challenges have been alleviated. CNC is the automated control and actuation of machining tools by following a series of coded programs. In this subsection, we discuss topology constraints specific to CNC machining.

In [18], a minimum hole constraint for machining is shown in Equation (13) where discretized elements of the geometry are used. Given G as the desired hole size, s_k is the exterior surface area of element k with max M number of elements. l^p is the average depth of the hole size and G^* is the desired minimium hole size. In a TO problem setting, the constraint of Equation (13) is to be defined in Equation (8).

$$G = \frac{\sum_{k=1}^{M} s_k \times x_k^e}{l^p} \geq G^* \tag{13}$$

While such formulation is relevant within the TO community, our work is reliant on the level-set method in which Equation (13) will not work for our setting. Moreover, constraints in relation to maximum thickness, minimum thickness and minimum member distance are introduced in [19] and can similarly be applied to our work.

The maximum thickness constraint in Equation (14) states that a shape Ω has a smaller thickness than d_{max} if the absolute value of the SDF for a particular point $x \in \Omega$ is less than $d_{max}/2$. Note that the $SDF(x)$ is negative for points in domain Ω. Concurrently, its penalization function is given in Equation (15).

$$\boxed{SDF(x) \geq -d_{max}/2, \forall x \in \Omega} \tag{14}$$

$$\boxed{P(\Omega) = \int_{\Omega} [\min(0, SDF(x) + d_{max}/2)]^2 dx} \tag{15}$$

The minimum thickness constraint along with its corresponding penalization function are given in Equations (16) and (17), respectively. Simply put, an offset distance d_{off} is defined in relation to a minimum thickness d_{min} by $0 \leq d_{off} \leq d_{min}$. The minimum thickness constraint states that starting at $x \in \partial\Omega$, we can travel along $-\vec{n}(x)$ up to d_{min} and still stay within shape Ω. It's important to highlight that solely setting $d_{off} = d_{min}$ is not sufficient as the constraint fails to detect any voids that do not belong part of shape which violate the constraint.

$$\boxed{SDF(x - d_{off}\vec{n}(x)) \leq 0, \forall x \in \partial\Omega, \forall d_{off} \in [0, d_{min}]} \tag{16}$$

$$\boxed{P(\Omega) = \int_{\Omega} \int_{0}^{d_{min}} [\max(0, SDF(x - \zeta\vec{n}(x)))]^2 d\zeta dx} \tag{17}$$

The minimum member distance is similarly defined as the minimum thickness condition, except the normal direction is reversed and d_{mmd} is denoted as the minimum member distance between two structural members. In the event of a member surrounding a hole, this formulation is equivalent to defining a minimum hole size. The following equation reveals the minimum member distance constraint along with its penalty function.

$$\boxed{SDF(x + d_{off}\vec{n}(x)) \leq 0, \forall x \in \partial\Omega, \forall d_{off} \in [0, d_{mmd}]} \tag{18}$$

$$\boxed{P(\Omega) = \int_{\Omega} \int_{0}^{d_{mmd}} [\max(0, SDF(x + \zeta\vec{n}(x)))]^2 d\zeta dx} \tag{19}$$

III. Related Work

A. *Shape Learning*

Shape reconstruction learning techniques can be largely categorized into three groups; point-based, mesh-based and voxel-based [4].

1) Point-based: A point cloud is a series of data points that represent a 3D (or 2D) object with a set of (x, y, z) coordinates. Registration of such points are usually extracted using laser scanning techniques such as LiDAR sensors. PointNet in [20] is an approach that learns using point clouds. However, the described geometry fails to accurately represent watertight surfaces.

2) Mesh-based: Many approaches surrounding mesh-based representation exist for shape learning. The work in [21] relies on poly-cube mapping for optimization. However, it is limited to certain mesh configurations. Other methods [22], [23] propose learning parameterization of surfaces by morphing multi-planes. While the use of deep networks have been utilized to address the shortcomings of reconstruction quality induced by paremeterization algorithms, the final produced shape fails to generate closed surfaces [22].

3) Voxel-based: A voxel represents a single sample or data point within a 3D space by using a technique similar to convolution, a technique commonly found within the 2D imaging community. The most simple example of a voxel-based learning method for shapes is a binary occupancy grid. This approach discritizes and scans the environment to determine if the 3D volume is found (occupied/not occupied) within a specific region. One limitation of this approach is the inability to highlight finer resolutions [24], [25]. It is also possible to use voxel-based learning to represent signed distance functions as in [26], [27]. However, such approach is computationally expensive and memory intensive.

B. *Generative Modelling*

Existing representation learning approaches aim at identifying and compressing a set of features to express entire data as a whole. A more comprehensive review is found in [28].

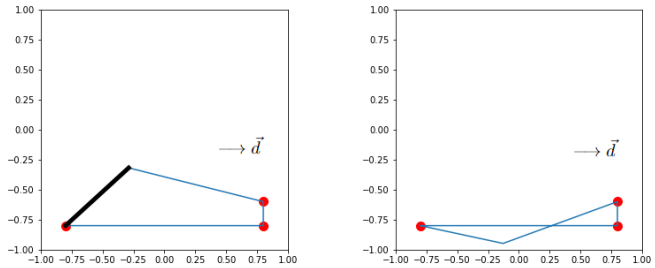
1) Generative Adversial Networks: GANs learn a set of compressed features by training a discriminator adversally against a generator [4]. In [29], a GAN is trained to generate an object using voxel representation. A downside of such approach is that GANs are unstable in certain circumstances.

2) Autoencoders: Autoencoders work by compressing the input data into a latent-space representation and then using a decoder to reconstruct the output using this representation. This technique has been largely explored as a representation learning method in literature [22], [30]. More recently, variational autoencoders have been used in [31], in which the latent-space is perturbed with Gaussian noise for regularization.

3) Autodecoders: Autodecoders work by training only the decoder portion of the autoencoder network to simultaneously optimize the latent vectors for each data point and decoder weights. The work in [32] applies this technique towards deep architectures.

IV. Dataset

We work with 100 shapes for our training data, in which 50 are completely randomly generated, while the remaining consists of what we refer as constraint *satisfying shapes*. We introduce the set of *satisfying shapes* with the rationale that the network should learn, but not overfit to the desired geometry. The *satisfying shapes* are intended to be shapes that are manufacturable under casting and do not violate such constraints mentioned in Section II-A. We generate these shapes by fixing three vertices of a four-sided polygon. The remaining vertex is sampled randomly and as a result, we estimate that upwards of 10% of the *satisfying shapes* are in fact *violating shapes*. The randomly generated shapes are five-sided polygons such that the training set is more diverse for the network to learn an array of different shape types. Examples of the training shapes are shown in Figure 9.



(a) Satisfying shape (b) Violating shape

Fig. 9: Example of shapes generated for dataset. Three vertices shown as red points are fixed and the final vertex is sampled randomly. Note in a), the black edge is intended to be the parting surface such that only a single mold is needed to be removed during casting.

From the gathered shapes as ground truths, we sample 200,000 uniformly random points in a confined [-1, 1] region for both (x, y) and compute each random point's respective SDF value. We identify that the SDF of points that are far away from the surface edge are of little value in the final reconstruction. As a result, we sample more aggressively from points near the boundary by using a more principled approach inspired from [33].

We employ an exponential weighting metric as in [33] and use a rejection criteria for samples which violate Equation (20) against a uniform distribution U. The β parameter ranging from $0 \rightarrow \infty$ determines how aggresive to sample near the surface. Figure 10 illustrates this behaviour.

$$w(x) = e^{-\beta|SDF(x)|} > x \sim U(0,1) \tag{20}$$

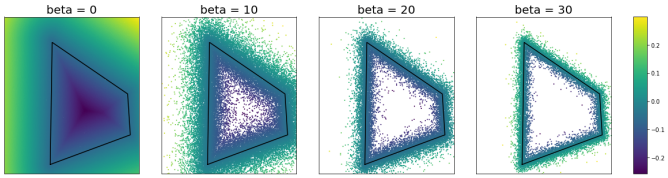


Fig. 10: Exponential sampling of points near surface by altering β parameter.

In our work we use $\beta = 30$. The generated number of samples are hence reduced from 200,000 to $\approx$ 10,000 after employing the weighting criteria. The final samples are saved as `npz` files for future processing.

V. Model

The details of the neural network architecture is further discussed in [4]. In summary, we train a decoder model

f_θ, as an approximator of the SDF for a given sample, on a dataset $S := \{x, SDF(x)\}$ where x are the sample coordinates. The network parameters θ of $f_\theta(x)$ are updated using backpropogation after computing the L_1 loss function:

$$\mathcal{L}(f_\theta(x), SDF(x)) = |f_\theta(x) - SDF(x)| \tag{21}$$

Rather than training a network for a single shape, it is more suitable for f_θ to learn a series of different shapes by embedding a latent vector z into the model. As such, the latent vector can represent a desired shape and the network can now be described as:

$$f_\theta(z_i, x) \approx SDF(x) \tag{22}$$

Given that each training shape S_i in dataset S is paired with a latent vector z_i, the posterior of z_i for a given S_i can be written as follows:

$$p_\theta(z_i|S_i) = p(z_i) \prod_{x_j \in S_i} p_\theta(SDF(x_j)|z_i; x_j) \tag{23}$$

The prior distribution $p_\theta(z_i)$ is formulated as a zero-mean multivariate-Gaussian with a spherical covariance $\sigma^2 I$ [4]. The likelihood $p_\theta(SDF(x_j)|z_i; x_j)$ can be represented using the L_1 function expressed in Equation (21) and takes the form:

$$p_\theta(SDF(x_j)|z_i; x_j) = \exp(-\mathcal{L}(f_\theta(z_i, x_j), SDF(x_j)) \tag{24}$$

During training, the joint log posterior in Equation (26) over all training shapes is maximized with respect to individual latent vectors $\{z_i\}_{i=1}^N$ and model paramters θ [4].

$$\arg\min_{\theta, \{z_i\}_{i=1}^N} \sum_{i=1}^{N} \left(\sum_{j=1}^{K} \mathcal{L}(f_\theta(z_i, x_j), SDF(x_j)) + \frac{1}{\sigma^2} \|z_i\|_2^2 \right) \tag{25}$$

After training, we can fix θ and learn a latent vector for an unobserved shape S_i using the Maximum-a-Posterior estimation as in Equation (26). This formulation is crucial for SDF samples of arbitrary sizes because the loss gradient with respect to z can be solved separately.

$$\hat{z} = \arg\min_{x} \sum_{x_j \in S} \mathcal{L}(f_\theta(z_i, x_j), SDF(x_j)) + \frac{1}{\sigma^2} \|z_i\|_2^2 \tag{26}$$

It is important to note that we do not test on unobserved shapes throughout this report and instead focus solely on the training shapes. We use the same parameters as in [4] except work with a smaller network of four fully connected layers with 128 dimensions each. We train for 3000 epochs and perform batch gradient descent rather than mini-batch gradient descent as in [4]. Further, we use a latent vector size of 32, an eighth of the original work, to represent a single shape. The surface can be implicitly represented as the zero iso-surface of $f_\theta(x)$ using marching squares algorithm [34]. An example of the reconstructed geometry from training is shown in Figure 11. Henceforth, we will work with the following reconstructed shape throughout optimization. The training loss of the entire dataset is shown in Figure 12.

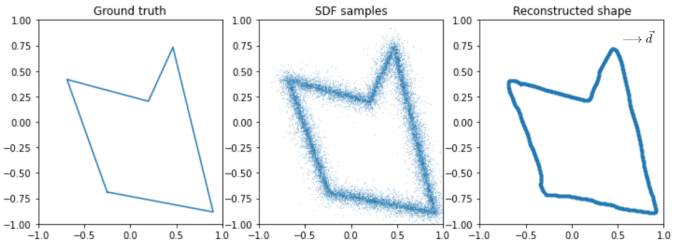


Fig. 11: Reconstructed geometry rendered after training the neural network.

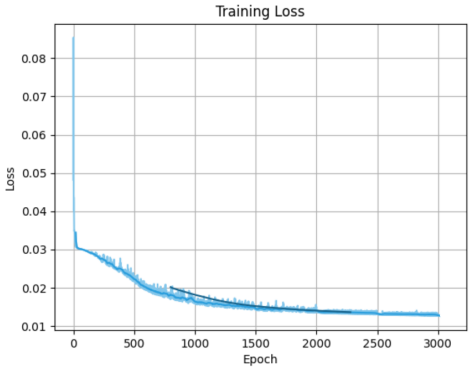


Fig. 12: Loss plot during training.

VI. Optimization

In this section, we will discuss the optimization steps performed to achieve manufacturability with casting from the reconstructed shape. We solve the following 2D optimization problem in minimizing the area of the shape $A(\Omega)$ in the bounded domain D outlined below:

$$\begin{aligned} &\min\ A(\Omega) \\ &\text{subject to } SDF(g(x)) \leq 0, \forall g(x) \in D \\ &\vec{d} \cdot \vec{n}(x) \geq 0, \forall x \in \partial\Omega \end{aligned} \tag{27}$$

where $g(x)$ are geometric points that are to be included in the shape and $\vec{d} \cdot \vec{n}(x) \geq 0$ constraints the presence of undercuts or internal voids.

Recall that the use of an autodecoder in Section III-B allows us to freeze the model weights from training and optimize the latent vectors through backpropogation. For the single shape example in Figure 11, we will work with a single latent vector and use the same model parameters outlined in Section V. We construct three new loss functions for the objective function, geometric constraint and draw direction constraint shown in Equations (28), (29) and (30), respectively. We also introduce a fourth loss function in Equation (31) which penalizes the network directly in the presence of an

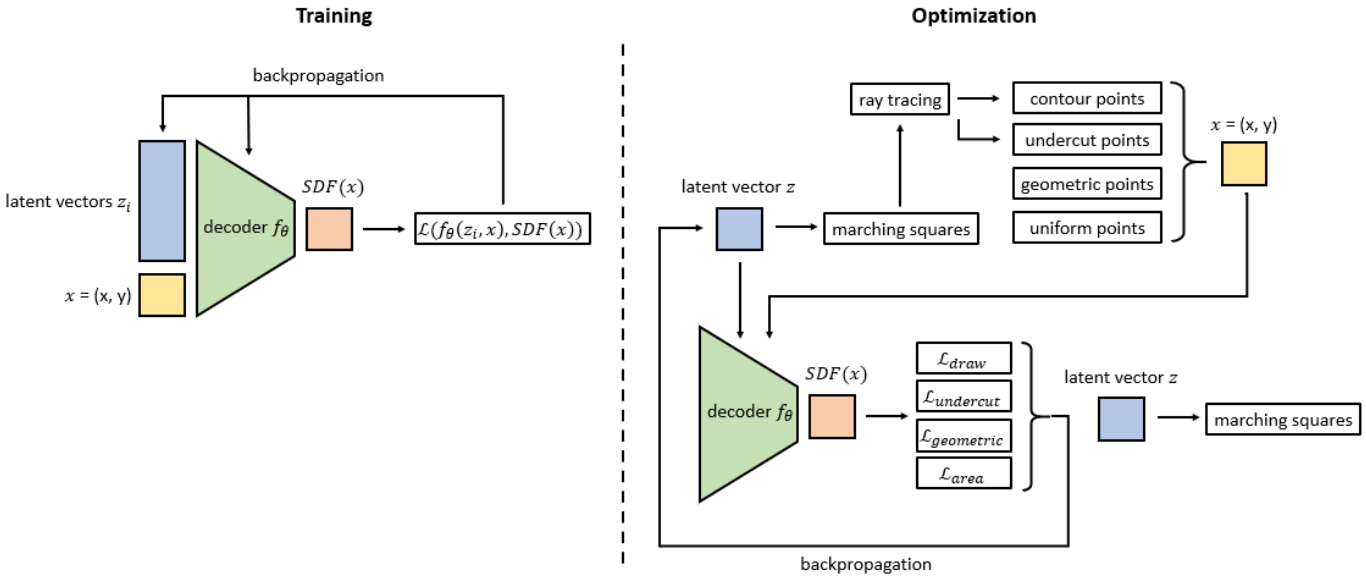


Fig. 13: Training and optimization network.

undercut and serves a similar purpose or goal as the draw direction constraint. The undercut loss function is analogous to the geometric constraint, however the sampled points are obtained differently. The constant λ_i determines the severity of the network penalization.

$$\mathcal{L}_{area} = max(0, -\lambda_1 SDF(x)), \forall x \sim U(-1, 1) \quad (28)$$

$$\mathcal{L}_{geometric} = max(0, \lambda_2 SDF(x), \forall x \in D \quad (29)$$

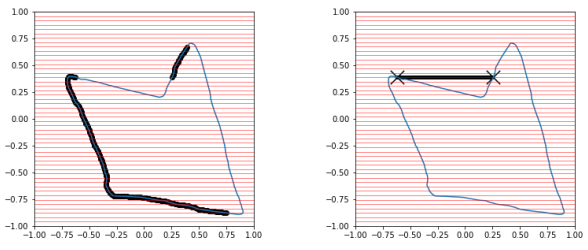
$$\mathcal{L}_{draw} = max(0, -\lambda_3(\vec{d} \cdot \vec{n}(x))), \forall x \in \partial\Omega \quad (30)$$

$$\mathcal{L}_{undercut} = max(0, \pm\lambda_4 SDF(x), \forall x \sim U_{undercut} \quad (31)$$

The loss function for area minimization states that for a given uniform sample in the domain $[-1, 1]$, we penalize the network if the SDF of such value is negative, indicating that it lies inside the shape's contour, thus minimizing the shape's area such that the point lies on or oustide of the surface. The geometric loss states the opposite in which the network is penalized if the SDF of the desired geometric point x is positive, indicating that the point lies outside of the surface. For each iteration, we extract the contour (zero iso-surface) of $f_\theta(x)$ in order to obtain the points that lie on the object's surface. We can compute the normals analytically by calculating $\partial f_\theta(x)/\partial x$ via backpropogation for each $x \in \partial\Omega$ and thus obtain the loss for the draw direction constraint. However, a concern exists in which the dot product between the draw direction and normal of specific contours will be negative despite the surface not violating the constraint. A simple example that illustrates this phenomenon is the normal of the leftmost edge of a rectangle subject to the draw direction in Figures 9 and 11.

To address this problem, we raycast within the bounded domain and along the draw direction to find contours that should not violate the constraint. By finding the first point of intersection between the cast ray and contour, we know not to include the normals of the contour in our loss function. The presence of an undercut would still be accounted for in the loss function, as the cast ray would not intersect the region at first encounter as shown in Figure 14. Note that the discovered surface resembles the parting surface and is the interface in which the two molds come together.

The direct undercut penalization also utilizes raycasting, however does not require the need for further surface normal calculations. In addition to using the cast rays for finding the parting surface, it simultaneously attempts to discover any undercut regions. By casting a ray along the draw direction, we know that a perfectly castable shape should have exactly two points of intersection (assuming the ray does not rest perfectly on a contour). In other words, any cast rays that intersect the shape more than two times, would indicate an undercut. We sample $x \sim U_{undercut}$ along the cast ray which intersect the undercut region and penalize the network similarly as the geometric constraint in order to include the sampled points within the shape. The constant λ is positive for $+SDF$ values and negative otherwise. This method is illustrated in Figure 14.



(a) Parting surface (b) Undercut

Fig. 14: Raycasting along $+x$ direction to find parting surface and points to sample along undercut contour.

In summary, at each iteration we compute a series of loss values depending on the objective function and constraints. For each loss value calculated, we backpropogate the loss function with respect to the input latent vectors. Once the loss values for all objective function and constraints are found for a single interation, we proceed with a gradient descent update solely on the latent vectors. The overall two-step pipeline consisting of training and optimization is shown in Figure 13.

VII. Results

The following subsections outline the results obtained from optimizing the shape in Figure 11. Note that in the original work of [4], the reconstruction accuracy is explicitly benchmarked against current state of the art methods. This is not feasible for our scenario as we have no ground truth labels for the reconstructed part after optimization. Instead, we showcase qualititative results of the final rendered geometry for each given constraint. We optimize for 3000 epochs or until convergence is reached.

A. *Area Minimization*

For the objective function, area minimization, we sample 100 uniform points in the bounded domain and apply a constant of $\lambda_1 = -5$. The network reaches convergence and the results are shown in Figure 15:

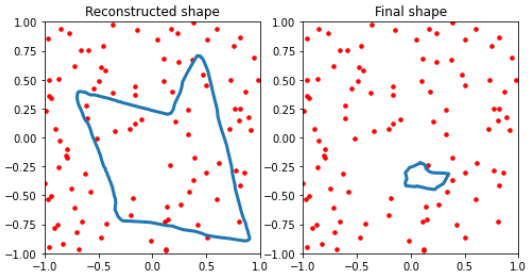


Fig. 15: Shape after training vs. shape after optimization with area minimization constraint.

B. *Geometric*

The geometric constraint uses constant $\lambda_2 = 10$ and penalizes the network with four geometric points bounded at the limits of $[\pm 0.7, \pm 0.7]$. Note that the penalization is only applied to geometric points that lie outside of the contour, hence points that are within the geometry are acceptable. The network reaches convergence and the results are shown in Figure 16.

C. *Draw Direction*

The draw direction constraint uses constant $\lambda_3 = -2$ and is only penalized for the shaded red region shown in Figure 17. After optimizing, there still exists a smaller undercut as the network never reaches convergence. We speculate that optimizing the network for greater number of epochs or tuning the learning rate may alleviate this issue.

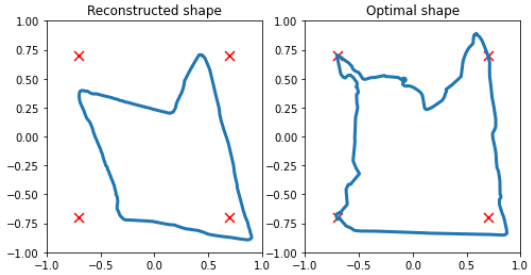


Fig. 16: Shape after training vs. shape after optimization with geometric constraint.

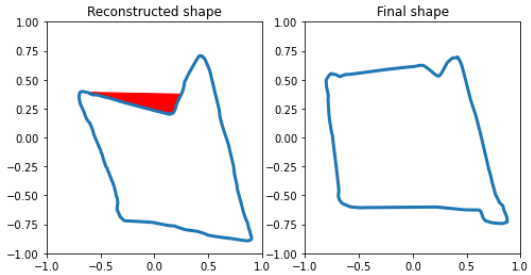


Fig. 17: Shape after training vs. shape after optimization with draw direction constraint.

D. *Undercut*

The undercut constraint uses constant $\lambda_4 = 5, -5$ for $+SDF$ and $-SDF$, respectively. Similar to the draw direction constraint, we sample 20 uniform points along the horizontal edge of the red region to fill the undercut void. We have tried sampling for greater number of points, but observe less optimal results. In the final shape of Figure 18, there exists a small protrusion in the top left corner. We believe that increasing the contour resolution will help address this concern such that the undercut region is found at a higher $+y$ value during raytracing. We have also tried addressing this issue by using a greater negative λ constant, but found it to be less effective.

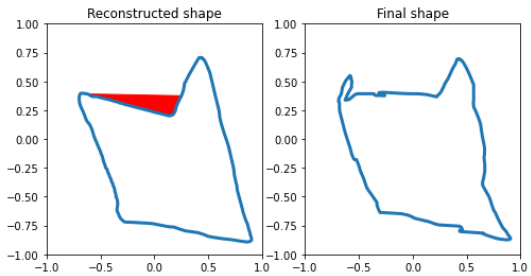


Fig. 18: Shape after training vs. shape after optimization with undercut constraint.

E. *Combined Results*

We show a series of multi-objective results in Figure 19. Namely, from left to right, the individual plots are for constraints: a) area and geometric, b) geometric and undercut and

c) area, geometric and undercut. λ_i values were updated such that greater penalization and importance are placed onto the geometric and undercut constraints while imposing a softer area minimization constraint. The before and after λ_i values are highlighted in Table I. Further, it should be noted that converge was never reached for any of the trials. In addition, the network was only ran for 500 epochs for b) and c) due to GPU memory issues. This remain a bottleneck throughout the project and will be addressed in the proceeding section.

TABLE I: Before and after λ_i values

λ_i	**Before**	**After**
1	-5	-0.05
2	10	100
3	-2	-
4	± 10	± 100

It can be observed that the network in a) is attempting to minimize the area while simultaneously incorporating the geometric point at $(-0.7, 7)$ inside its boundary. However, the irregularity of the reconstructed shape remains a concern. We expect this to occur as a result from training such a large network on only a few shapes. In the original work in [4], the authors use a dataset that is several magnitudes larger than ours, despite having only a slightly more complex network.

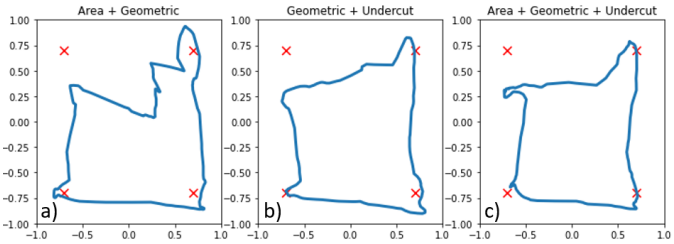


Fig. 19: Multi-objective results for a) area and geometric, b) geometric and undercut and c) area, geometric and undercut.

In b), we observe that the network has made a successful attempt in filling the undercut region present in Figure 18. In c), we impose our objective function to minimize area, and notice that the undercut contour is more horizontal than in b). Further, it can be observed visually that the area of the shape in c) is less than that of b).

VIII. Conclusion

In this project we employ a series of manufacturing constraints to an optimization network in attempt to reconstruct manufacturable shapes under traditional processes. We use the model outlined in [4] and implement a two step pipeline with an optimization network that follows after training. We highlight results of the rendered shapes from using individual constraints, as well as a combination of them. While we demonstrate the overall concept in this report, we believe there still exists future steps that can be considered for improved results.

Firstly, like in many machine learning problems, computation was a bottleneck present throughout the project. Given more compute power will enable the network to train for greater epochs, thus potentially reaching convergence. Further, more compute will enable finer contour resolutions and allow greater freedom in points to sample. Secondly, the λ parameters used in the individual loss functions were chosen arbitrarily and future work should consider a more robust method for hyperparameter selection. Additionally, cross validation should be used for tuning the model parameters during training as selection of parameters consumed a considerable amount of time. Thirdly, the final rendered shape during training was not as desirable as expected. Ideally, the produced shape should be as close to an exact match as the ground truth as possible, but this was not observed. There also exists an unexplained behaviour in the loss plot during the initial epochs. Fourthly, training using a smaller and more simplified network will have prevented overfitting that was observed in the results. Finally, the method for detecting undercut regions may not entirely be robust to edge cases for the rendered irregular shape. This may explain why the results obtained from the undercut constraint are not as desirable as expected.

Some areas that can be further explored or to be built upon the work in this project include the use of a larger and more complex dataset such that the network is able to generalize onto more shapes. Additionally, we work mainly on constraints related to casting, however other processes should also be investigated. Finally, while we focus on 2D shapes for simplicity, it would be more applicable to apply the proposed work onto 3D shapes.

IX. Acknowledgements

I would like to thank Pierre Yvernay for his guidance and mentorship throughout the project. I would also like to thank Prof. Pascal Fua for the opportunity to work on the project.

REFERENCES

[1] C. W. Hull, "The birth of 3d printing," *Research-Technology Management*, vol. 58, no. 6, pp. 25–30, 2015.

[2] H. Wang, J. Liu, and G. Wen, "Achieving large-scale or high-resolution topology optimization based on modified beso and xefm," 2019.

[3] C. Weller, R. Kleer, and F. T. Piller, "Economic implications of 3d printing: Market structure models in light of additive manufacturing revisited," *International Journal of Production Economics*, vol. 164, pp. 43 – 56, 2015.

[4] J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove, "Deepsdf: Learning continuous signed distance functions for shape representation," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[5] 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, K. V. Gernaey, J. K. Huusom, and R. Gani, Eds. Elsevier, 2015, vol. 37, pp. 1463 – 1468.

[6] Y. Xie and G. Steven, "A simple evolutionary procedure for structural optimization," *Computers and Structures*, vol. 49, no. 5, pp. 885 – 896, 1993.

[7] ANSYS. Ansys fluent - cfd software — ansys.

[8] G. Allaire, F. Jouve, and G. Michailidis, "Molding direction constraints in structural optimization via a level-set method," in *Variational Analysis and Aerospace Engineering*. Springer, 2016, pp. 1–39.

[9] M. Zhou, R. Fleury, Y.-K. Shyy, H. Thomas, and J. Brennan, *Progress in Topology Optimization with Manufacturing Constraints*.

[10] G. G. Lothar Harzheim, "A review of optimization of cast parts using topology optimization," *Structural and Multidisciplinary Optimization*, vol. 31, pp. 388 – 399, 2006.

[11] Q. Xia, T. Shi, M. Wang, and S. Liu, "A level set based method for the optimization of cast part," *Structural and Multidisciplinary Optimization*, vol. 41, pp. 735–747, 2010.

[12] Q. Xia, T. Shi, M. Y. Wang, and S. Liu, "Simultaneous optimization of cast part and parting direction using level set method," *Struct. Multidiscip. Optim.*, vol. 44, no. 6, p. 751–759, Dec. 2011.

[13] F. Murat and J. Simon, "Etude de problème d'optimal design," in *Proceedings of the 7th IFIP Conference on Optimization Techniques: Modeling and Optimization in the Service of Man, Part 2*. Berlin, Heidelberg: Springer-Verlag, 1975, p. 54–62.

[14] R. Ranjan Yadav, Y. Dewang, J. Raghuwanshi, and V. Sharma, "Finite element analysis of extrusion process using aluminum alloy," *Materials Today: Proceedings*, vol. 24, pp. 500 – 509, 2020, international Conference on Advances in Materials and Manufacturing Applications, IConAMMA 2018, 16th -18th August, 2018, India.

[15] M. Bendsøe, N. Olhoff, and O. Sigmund, Eds., *Topological design optimization of structures, machines and materials: Status and perspectives*. Kluwer/Springer, 2006.

[16] S. L. Vatanabe, T. N. Lippi, C. R. de Lima, G. H. Paulino, and E. C. Silva, "Topology optimization with manufacturing constraints: A unified projection-based approach," *Advances in Engineering Software*, vol. 100, pp. 97 – 112, 2016.

[17] S. Chaturvedi, "Introduction to machine and machine tools," 04 2015.

[18] K.-T. Zuo, L.-P. Chen, Y.-Q. Zhang, and J. Yang, "Manufacturing- and machining-based topology optimization," *The International Journal of Advanced Manufacturing Technology*, vol. 27, no. 5, pp. 531–536, Jan 2006.

[19] G. Allaire, F. Jouve, and G. Michailidis, "Thickness control in structural optimization via a level set method," *Structural and Multidisciplinary Optimization*, vol. 53, no. 6, pp. 1349–1382, Jun 2016.

[20] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," *arXiv preprint arXiv:1612.00593*, 2016.

[21] P. Baqué, E. Remelli, F. Fleuret, and P. Fua, "Geodesic convolutional shape optimization," 2018.

[22] T. Groueix, M. Fisher, V. G. Kim, B. C. Russell, and M. Aubry, "Atlasnet: A papier-mâché approach to learning 3d surface generation," 2018.

[23] H. Ben-Hamu, H. Maron, I. Kezurer, G. Avineri, and Y. Lipman, "Multi-chart generative surface modeling," *ACM Transactions on Graphics*, vol. 37, no. 6, p. 1–15, Jan 2019. [Online]. Available: http://dx.doi.org/10.1145/3272127.3275052

[24] C. B. Choy, D. Xu, J. Gwak, K. Chen, and S. Savarese, "3d-r2n2: A unified approach for single and multi-view 3d object reconstruction," 2016.

[25] Zhirong Wu, S. Song, A. Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and J. Xiao, "3d shapenets: A deep representation for volumetric shapes," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 1912–1920.

[26] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohi, J. Shotton, S. Hodges, and A. Fitzgibbon, "Kinectfusion: Real-time dense surface mapping and tracking," in *2011 10th IEEE International Symposium on Mixed and Augmented Reality*, 2011, pp. 127–136.

[27] B. Curless and M. Levoy, "A volumetric method for building complex models from range images," in *SIGGRAPH '96*, 1996.

[28] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013.

[29] J. Wu, C. Zhang, T. Xue, W. T. Freeman, and J. B. Tenenbaum, "Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling," in *Advances in Neural Information Processing Systems*, 2016, pp. 82–90.

[30] T. Bagautdinov, C. Wu, J. Saragih, P. Fua, and Y. Sheikh, "Modeling facial geometry using compositional vaes," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 3877–3886.

[31] M. Bloesch, J. Czarnowski, R. Clark, S. Leutenegger, and A. J. Davison, "Codeslam - learning a compact, optimisable representation for dense visual slam," 2019.

[32] P. Bojanowski, A. Joulin, D. Lopez-Paz, and A. Szlam, "Optimizing the latent space of generative networks," 2019.

[33] T. Davies, D. Nowrouzezahrai, and A. Jacobson, "Overfit neural networks as a compact shape representation," 2020.

[34] W. E. Lorensen and H. E. Cline, "Marching cubes: A high resolution 3d surface construction algorithm," *SIGGRAPH Comput. Graph.*, vol. 21, no. 4, p. 163–169, Aug. 1987. [Online]. Available: https://doi.org/10.1145/37402.37422