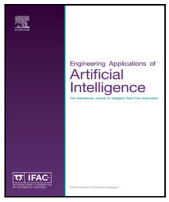
Engineering Applications of Artificial Intelligence 123 (2023) 106482

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

journal homepage: www.elsevier.com/locate/engappai



Implicit Neural Representations of Sheet Stamping Geometries with Small-Scale Features

Hamid Reza Attar [a], Alistair Foster [b], Nan Li [a,*]

[a] Dyson School of Design Engineering, Imperial College London, London SW7 2DB, UK
[b] Impression Technologies Ltd, Coventry CV5 9PF, UK

ARTICLE INFO

Keywords:
Stamping
Machine learning
Implicit neural representation
Signed Distance Field (SDF)
Geometry
Optimisation

ABSTRACT

Geometric deep learning models, like Convolutional Neural Networks (CNNs), show promise as surrogate models for predicting sheet stamping manufacturability but lack design variables essential for inverse problems like geometric optimisation. Recent developments in deep learning have enabled geometry generation from compact latent spaces that are suitable for optimisation. However, current methods do not accurately model small-scale geometric features that are crucial for stamping performance. This study proposes a new deep learning-based method to address this limitation and generate detailed stamping geometries for optimisation. Specifically, neural networks are trained to generate Signed Distance Fields (SDFs) for stamping geometries, where the zero-level-set of each SDF implicitly represents the generated geometry. A new training approach is proposed for generating SDFs of stamping geometries, which involves supervising geometric properties of the SDFs. A novel loss function is introduced that directly acts on the zero-level-set and places high emphasis on learning small-scale features. This approach is compared with the state-of-the-art approach DeepSDF by Park et al. (2019), which explicitly supervises SDF values using ground truth data. The geometry generation performance of networks trained using both approaches is evaluated quantitatively and qualitatively. The results demonstrate significantly greater geometric accuracy with the proposed approach, which can faithfully generate small-scale features. Further analysis of the new approach reveals an organised learned latent space and varying the network input generates high-quality geometries from this space. By integrating with CNN-based manufacturability surrogate models by Attar et al. (2021), this work could enable the first-ever manufacturability-constrained optimisation of arbitrary sheet stamping geometries, potentially reducing geometry design time and cost.

1. Introduction

1.1. Industrial drive and research motivation

The design of stamping geometries is of central importance to their functional performance and manufacturability. For example, consider the stamping of a deep drawn electric vehicle battery container as presented in Attar et al. (2021a). Tight designed corner radii allow for a reduction of unusable internal volume but may impose large thinning defects and fractures during stamping (Zhou et al., 2014). In addition, the latest stamping processes, such as Hot Form Quench (HFQ®) for hot stamping high strength aluminium alloys (Lin et al., 2008), have the potential to widen the scope of manufacturability and therefore create structures with superior functional performance. These benefits are realised through enabling complex components with challenging geometric features to be produced from high specific strength and stiffness alloys. For example, Politis et al. (2016) successfully stamped a door inner component with a 200 mm draw depth feature from AA6082 using the HFQ process. Thus optimising stamping geometries is vital for leveraging the design capabilities of these stamping processes while meeting stringent manufacturability criteria.

The design of geometric features such as fillet radii is highly influential in dictating manufacturing performance of stamping geometries. For example, Attar et al. (2021a) showed that a small change in die and punch fillet radii produces a large change in sheet thinning during stamping. Despite their strong influence, fillet radii are small-scale geometric features when considered in the context of the global component scale. For example, automotive door inner components could be as wide as 1500 mm (Politis et al., 2016) and the latest stamping processes (Mohamed et al., 2012) can enable fillet radii of 10 mm or smaller. Therefore, representing stamping geometries to allow accurate manufacturing performance evaluation and thus enable effective optimisation demands high quality representations of small-scale features. State-of-the-art approaches for predicting and optimising stamping performance and representing geometry are now reviewed.

* Corresponding author.
E-mail address: n.li09@imperial.ac.uk (N. Li).

https://doi.org/10.1016/j.engappai.2023.106482
Received 17 December 2022; Received in revised form 6 May 2023; Accepted 16 May 2023
Available online xxxx

1.2. Advancements in predicting manufacturability and optimising sheet stamping geometries

Physics-based Finite Element (FE) simulations in conjunction with advanced material models have been shown capable of predicting the stamping performance of designed geometries with high accuracy (Politis et al., 2016; El Fakir et al., 2014; Mohamed et al., 2015), and are widely used in industry. For example, Politis et al. (2016) used a temperature and strain rate dependent material model accurately predict the thinning behaviour of a hot stamped aluminium alloy door panel. However, the computational cost of these simulations limits extensive design explorations. To this end, the simulator would have to be run by a process expert each time a design engineer wishes to make a geometry change and this makes the design development process slow and costly.

To reduce computational effort, Surrogate-Based Optimisation (SBO) techniques can be used (Naceur et al., 2008; Hu et al., 2017; Zhou et al., 2013). These techniques employ surrogate models which are efficient, data-driven numerical approximates of expensive simulations. The development of these models requires first collecting input–output observations from offline simulation runs and then training a model to fit these observations. Once sufficiently trained, numerical optimisation can be performed on the efficient surrogate model as opposed to the expensive simulator. Xiao et al. (2016) used SBO to optimise stamping speed, blank holder force, friction and blank temperature to reduce thinning defects induced by stamping an automotive floor component under HFQ conditions. Similarly, Zhou et al. (2013) used SBO to optimise blank holder force and stamping speed to reduce thinning and minimise springback in a anti-collision side beam component under HFQ conditions.

However, available literature for SBO for stamping processes focuses on optimisation of processing parameters while changes to the geometry are rarely considered, as identified by Zimmerling et al. (2020) and Attar et al. (2021b). This focus means that existing surrogate models are inherently geometry specific and therefore even slight geometric changes made by a designer invalidates their output predictions. Once invalidated, an entirely new surrogate model construction campaign is required, which consists of discarding old data, collecting new data and model retraining.

In contrast to processing parameters, stamping geometries comprise of significantly more parameters, particularly for complex geometries with many Computer-Aided Design (CAD) dimensions. In addition, they are frequently evolving during design phases and are defined according to uncommon parameterisation schemes. These challenges have recently brought to light advanced surrogate models that borrow deep-learning-based modelling techniques to accept non-parametric geometrical inputs. For example, Pfaff et al. (2021) use graph neural networks in conjunction with mesh-based data representations to predict the dynamics of different physical systems, such as deformable plates and fluid flows. Zimmerling et al. (2019) developed a Convolutional Neural Network (CNN) to predict textile draping results for variable doubly curved geometries. Attar et al. (2021b,c) and Zhou et al. (2022) developed CNNs to predict thinning distributions after stamping variable deep drawn geometries. Therefore, these surrogate models appear promising for SBO of stamping geometries and thus capitalising on stamping process capabilities and reductions in geometry design development time.

However, surrogate models with geometric inputs (e.g., meshes or images) are only capable of forward predictions for variable geometries. This limitation means that designers would need to implement a trial-and-error approach since there is no backward feedback or guidance towards optimum geometric designs. Moreover, since these models take non-parametric representations of geometries as inputs, there are no parameters available to serve as design variables for optimisation of stamping geometries. This drawback currently prevents surrogate models with geometric inputs from being used in numerical optimisation settings. Therefore, there is a need for an optimisation-friendly representation of stamping geometries which, when combined with these surrogate models, enables optimisation of arbitrary sheet stamping geometries.

1.3. Computer methods for geometric representation

In CAD modelling software, e.g., SolidWorks, geometric representations such as curves and surfaces are used to create 3D models of objects. These geometric representations are constructed using mathematical equations and algorithms that define the shape, size, and orientation of the object being modelled. Curves are used to define the shape of an object in 3D space. Examples of curves include lines, circles, ellipses, and splines. Surfaces are created by combining multiple curves, and they define the outer boundary of an object. These curves are modelled as explicit functions and the parameters of these functions can be manipulated in an optimisation setting. For example, Li et al. (2020) constructed and optimised addendum surfaces for sheet metal stamping using parametric curves. However, since explicit functions define parametric splines, this representation is significantly restricted to pre-defined topologies. Further, these functions lack expressivity, may be difficult to mathematically formulate for complicated geometries and do not allow for optimisation between different parameterisation schemes.

Modern techniques are available for generating realistic images-based representations of geometries from compact, optimisation-friendly representations. A popular approach is to use generative models such as Variational Auto-Encoders (VAEs) (Wang et al., 2020a; Higgins et al., 2017) and Generative Adversarial Networks (GANs) and their variants (Goodfellow et al., 2014; Mirza and Osindero, 2014; Radford et al., 2016). VAEs are trained to replicate variants of the original input but are prone to blurred reconstructed images. On the other hand, GANs learn deep embeddings of target data by training image generating decoders adversarially against discriminators. Once trained, these networks can generate realistic images of objects and scenes which are visually indistinguishable from their training data distributions (Gayon-Lombardo et al., 2020). However, the training of GANs is notoriously unstable (Arjovsky et al., 2017; Gulrajani et al., 2017) and difficult to extend to 3D geometric representations (Wu et al., 2016), e.g., high resolution 3D CAD geometries for designers. These major limitations make both VAEs and GANs unsuitable for practical stamping applications in industrial settings.

Several other computer methods for 3D geometric representation are commonly used in computer graphics and scientific visualisation applications. These methods can be broadly categorised into *explicit* and *implicit* representations, as shown in Fig. 1. Explicit representations are defined by a set of geometric primitives, such as vertices, edges, and faces, which describe the boundaries of objects in space. Explicit representations can include voxels, which divide space into a regular grid and assign values to each cell to represent the presence or absence of material, as well as point clouds, which represent a collection of points in space that approximate the shape of an object. These explicit representations have been state-of-the-art approaches for 3D geometry modelling in computer graphics, engineering simulations, and related fields for many years (Druc et al., 2022; Kohar et al., 2021). However, they have certain limitations that can make them less than ideal for certain tasks. For example, point clouds can suffer from noise and irregular sampling, while voxel grids offer a more regular representation but can suffer from high memory requirements and limited resolution. Importantly, voxel-based or point cloud-based representations may not be suitable for modelling key small-scale features in stamping geometries, such as fillet radii, since they require a high voxel or point density to accurately capture fine details while increasing the voxel or point density may lead to large datasets and computationally expensive processing times.

Meshes are a promising type of explicit representation, which define the shape of an object by connecting vertices with edges to form

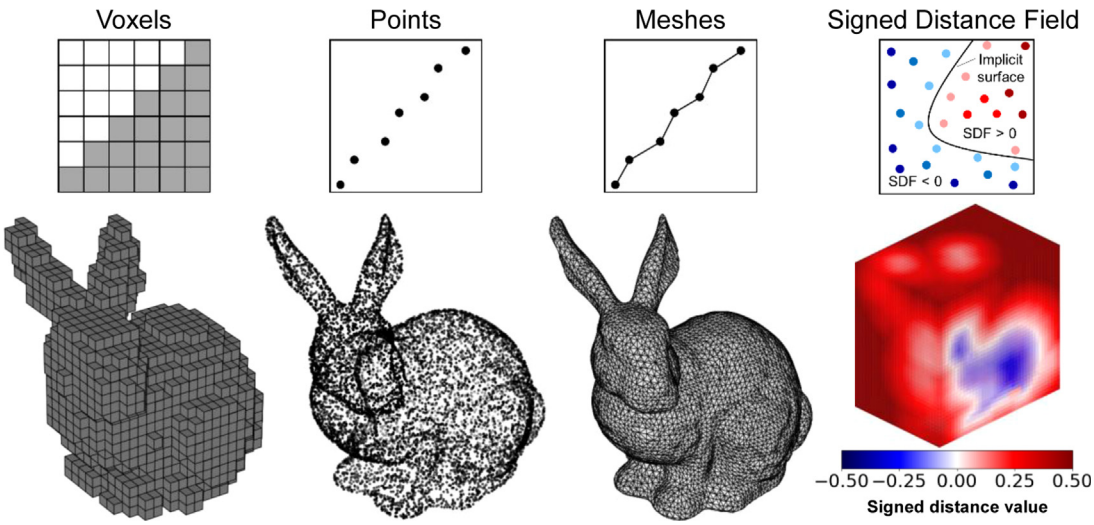


Fig. 1. Common geometric representations applied to the Stanford bunny. Voxels, points and meshes are examples of explicit representations while signed distance fields are implicit.

polygons that define the surface. By using fine mesh resolutions, high frequency geometric details on single shapes can be well represented, as seen from the examples provided by Sorkine et al. (2003). In attempts to extend these mesh-based representations to multiple arbitrary shapes, various works proposed representing meshes using data-driven 3D learning approaches, such as neural networks (Gupta and Chandraker, 2020; Jiang et al., 2020; Wang et al., 2018). These networks represent classes of similar shapes by optimising/deforming initial template meshes. In this context, Wang et al. (2018) proposed a network that generates a mesh from an image of an object by deforming the vertices of an initial spherical template mesh. Similarly, Baque et al. (2018) deformed mesh vertices via a poly-cube mapping algorithm for shape optimisation. However, although meshes are capable of representing high frequency details, the results of mesh-based optimisations lose these details and often result in poorly deformed meshes. This phenomenon arises because the optimisation is based on deforming an initial predefined topology and therefore cannot handle large topology changes well, as mentioned by Peng et al. (2021) and Park et al. (2019).

In contrast, implicit representations define shapes as level sets of 3D functions defined over voxelised grids (Peng et al., 2021; Liao et al., 2018; Osher and Paragios, 1999), and these level sets are then extracted or rendered into explicit meshes or images (Liao et al., 2018; Lorensen and Cline, 1987). These representations easily allow large changes in topology as they remove the need to deform initial meshes. For example, Remelli et al. (2020) changed the topology of genus 0 shapes into genus 1 by manipulating the underlying 3D function. However, operating on voxelised grids leads to large memory requirements and is therefore limited to low resolutions only. This limitation has been removed in recent years with the introduction of *implicit neural representations* which represent shapes as level sets of occupancy fields (Mescheder et al., 2019) or Signed Distance Fields (SDFs) (Park et al., 2019; Liu et al., 2020; Yang et al., 2021) that are generated from neural networks. These networks learn compact latent representations which are then decoded into their underlying 3D fields and these compact representations are suitable for numerical optimisation.

Implicit neural representations have gained tremendous popularity in the computer graphics and vision communities for modelling 3D shapes due to their expressiveness and flexibility that is not limited by resolution (Park et al., 2019). These representations appear promising as they enable large topology changes in an optimisation setting (Peng et al., 2021). Despite their advantages, current research on implicit neural representations has been focused on scenarios where only the visual shape quality on a global scale is important. For instance, they have been successfully used for modelling sofas, chairs in 3D scenes for computer games and virtual reality (Park et al., 2019; Chabra et al., 2020), and for modelling 3D human poses (Atzmon and Lipman, 2020; Gropp et al., 2020). However, these representations have not been utilised to model geometries for a scenario like sheet stamping of 3D components, where small-scale features play a critical role in determining functional performance. Thus, there is a clear research gap in exploring the use of implicit neural representations for modelling small-scale features in stamping geometries.

1.4. Novelty and main contribution of this paper

In summary, while the latest surrogate models have shown promise in efficiently assessing the manufacturability of stamping geometries without the need for costly FE simulations, they lack a geometric representation that is suitable for numerical optimisation. This drawback limits their use in numerical optimisation of stamping geometries. Meanwhile, powerful implicit neural representations have recently emerged and enable expressive geometry changes during optimisation. However, despite the success of these representations in computer vision applications, their effectiveness in representing stamping geometries has not yet been explored. This is a crucial gap in the literature since small-scale details often dictate the manufacturing performance of stamped components. Therefore, it is important to investigate the potential of these representations for stamping applications. The main contributions of this study are listed below:

- Development of a deep learning-based method for generating 3D stamping geometries using SDF-based implicit neural representations, providing a novel and flexible approach for representing such geometries.
- Design of a new loss function to encourage high geometric accuracy of small-scale features, addressing a crucial challenge in accurately representing stamping geometries.
- Proposal of systematic quantitative and qualitative evaluation methods to assess the accuracy of the generated stamping geometries, providing a comprehensive and rigorous analysis of the method's effectiveness.
- Evaluation of the new method's effectiveness at training neural networks to learn SDFs of stamping geometries with small-scale features compared with the state-of-the-art DeepSDF method (Park et al., 2019).
- Analysis of the learned continuous geometric latent space and smooth interpolation between geometries from different parameterisation schemes achievable for the first time. This analysis

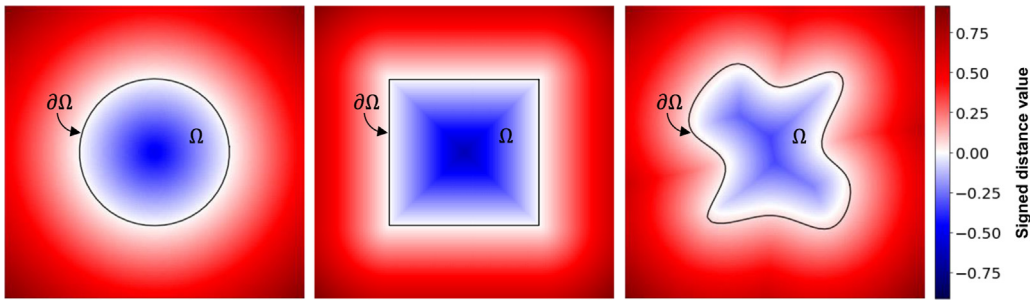


Fig. 2. 2D examples of SDFs for a circle, a square and an arbitrary shape. Black solid lines illustrate shape boundaries which are implicitly represented by the zero-level-set of the SDF.

demonstrates the new possibilities that have been opened up for exploring and designing complex stamping geometries in an optimisation-friendly way.

2. Overview of the proposed geometry generator

An overview of the proposed platform geometry generator is presented in this section. The concept of Signed Distance Fields (SDFs) are first introduced since SDFs are an important part of the non-parametric modelling strategy proposed in this paper. Details on a neural network model proposed for modelling these SDFs is then given.

2.1. Introduction to Signed Distance Fields (SDFs)

SDFs are widely used in the computer vision and graphics communities to model a variety of shapes. Some examples of shapes modelled by SDFs include complex shaped 3D objects, and scenery in computer game environments. This work proposes using SDF to model stamping die geometries.

2.1.1. Definition of SDFs

An SDF for a shape with internal domain Ω is defined as a scalar field s where the magnitude of a point anywhere in the field represents the distance d to the shape boundary $\partial\Omega$. d is defined here as the Euclidian distance to the closest point that is sampled on the continuous shape boundary $\partial\Omega$. The sign denotes whether the point is inside or outside Ω; by convention, points inside take a negative sign (−) and points outside take a positive sign (+) (Park et al., 2019), as expressed in Eq. (1)

$$s(\boldsymbol{x}) = \begin{cases} 0, & \boldsymbol{x} \in \partial\Omega \\ -d(\boldsymbol{x}), & \boldsymbol{x} \in \Omega \text{ and } \boldsymbol{x} \notin \partial\Omega \\ d(\boldsymbol{x}), & \boldsymbol{x} \notin \Omega \text{ and } \boldsymbol{x} \notin \partial\Omega \end{cases} \tag{1}$$

where $s(\boldsymbol{x}) \in \mathbb{R}$ is a scalar signed distance value of a point $\boldsymbol{x} \in \mathbb{R}^n$ for an n-dimensional SDF. The shape boundary $\partial\Omega$ is implicitly represented by the zero-level-set of the continuous SDF. The classification space ((−) or (+)) is explicitly represented by all other non-zero-level-sets.

Fig. 2 shows 2D examples of SDFs for three different shapes: a circle, a square and an arbitrary shape. The zero-level-set of these SDFs are highlighted by the black solid lines and these would be surfaces in the 3D case. The figure illustrates that SDFs can be used to implicitly represent a range of shapes irrespective of their geometric complexity.

2.1.2. Geometric properties of SDFs

The shape representative nature of SDFs leads to two noteworthy geometric properties. **Property 1:** if the shape domain Ω is a subset of an n-dimensional Euclidian space $\mathbb{R}^n$ with shape boundary $\partial\Omega$, then the SDF is differentiable everywhere and its gradient satisfies the solution to the Eikonal partial differential equation, expressed in Eq. (2)

$$\left\|\nabla_{\boldsymbol{x}} s(\boldsymbol{x})\right\| = 1 \tag{2}$$

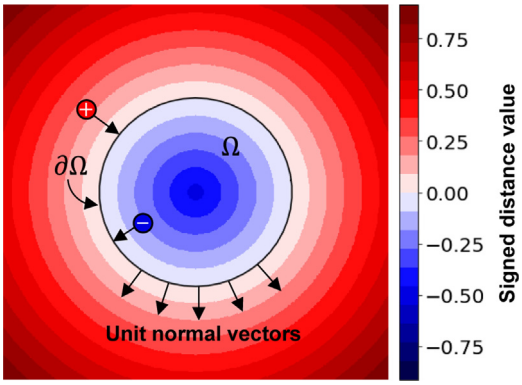


Fig. 3. An illustration of a 2D SDF for a circle with labelled characteristic features for the SDF. Although the field is continuous, the colourmap is shown as discrete to highlight equidistant contour lines. Black solid line illustrates the shape boundary $\partial\Omega$ which is implicitly represented by the zero-level-set of the SDF.

where the $\nabla_{\boldsymbol{x}}$ operator denotes the spatial gradient of $s(\boldsymbol{x})$ with respect to the coordinates of point $\boldsymbol{x}$. To interpret this property, Fig. 3 shows a 2D SDF for a circle shape but plotted with a discrete colourmap. The equidistant contour lines can be seen as a visualisation of constant spatial gradient. **Property 2:** the spatial gradients of the SDF at the shape boundary $\partial\Omega$ align with the outward normal vector field $\mathcal{N}$. This property arises since the gradient vector points in the direction of steepest descent (i.e., normal to each contour line). Furthermore, since Property 1 holds true everywhere (including at the shape boundary $\partial\Omega$), these surface normal vectors have unit magnitude. Property 2 can be mathematically expressed as in Eq. (3).

$$\nabla_{\boldsymbol{x}} s(\boldsymbol{x})\big|_{\partial\Omega} = \mathcal{N}(\boldsymbol{x}) \tag{3}$$

Fig. 3 shows the outward unit normal vectors at the shape boundary, where only a limited number are drawn for illustration. When combining Properties 1 and 2, the continuous SDF can be interpreted as a differentiable extension of the unit normal vector field. These properties will be used later in this paper to effectively train a neural network to learn a range of SDFs for a class of stamping geometries and with high accuracy.

2.2. Modelling SDFs of stamping geometries

As mentioned previously, SDFs were used in this work to model stamping die geometries to enable geometry optimisation. However analytically pre-computing and storing a large library of SDFs for various candidate geometries is neither feasible nor useful for optimisation. Instead, a model that can represent a wide range of geometries, discover similarities between them and store these similarities into a latent (i.e., characteristic) space is required.

To achieve this requirement, a neural network model for generating volumetric SDFs of different 3D geometries within entire geometry

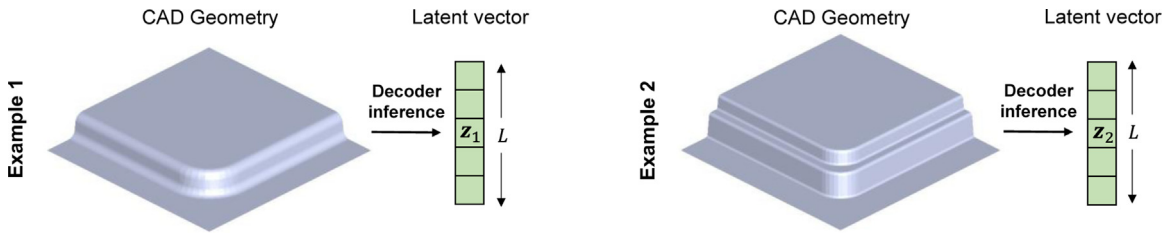


Fig. 4. CAD geometries are first converted into low dimensional latent vectors using the decoder inference process.

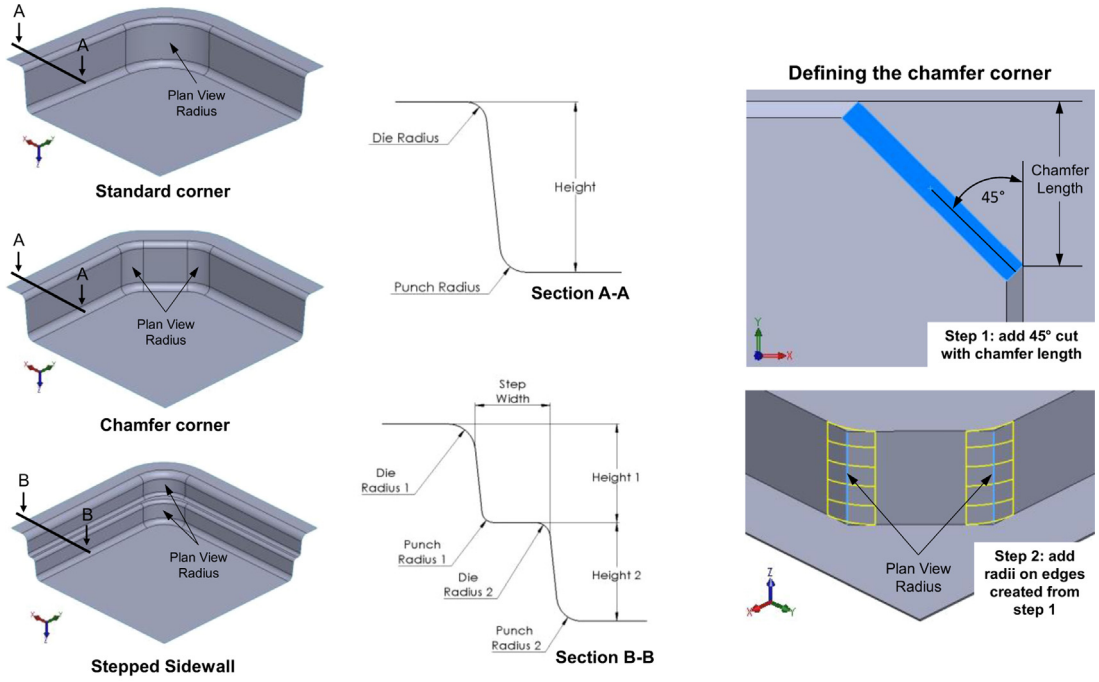


Fig. 5. Definition of the three box corner subclasses used in this study: standard corners, chamfer corners and stepped sidewalls. Figure highlights (a) parameterisation of CAD geometries with section views and (b) the two step process used to create chamfer corners.

classes is proposed. This concept was first introduced by Park et al. (2019) for generic shapes. For stamping applications, the trained network can be considered as a compact function that generates SDFs of common classes of components, while only explicitly storing the parameters of the neural network. For example, a network can be trained to generate SDFs for thousands of variants of A-Pillars, B-Pillars, Battery Boxes, or similar broad classes of component geometries. In this work, box corner geometries were considered for demonstration. Network training details are given in Section 3.

To effectively work with CAD models of stamping die geometries in a deep learning environment, these geometries must first be mapped into a suitable form. Here, each CAD geometry was represented as a latent vector $\boldsymbol{z} \in \mathbb{R}^L$ as illustrated in Fig. 4. These vectors were obtained using a process introduced as *decoder inference*, which will be detailed in Section 3. In essence, these latent vectors became compact characteristic representations of CAD geometries and were agnostic to geometric complexities.

2.3. Creating datasets of geometries

2.3.1. Geometry definitions

Deep drawn box corners were considered as the high level geometry class of interest in this study. These corners are common limiting design features and are found on a range of rectangular or square component designs, e.g., door inners or battery boxes (Politis et al., 2016; Zhu et al., 2021). Due to their symmetry, quarter boxes were modelled. Half-lengths of 500 mm and 6° draft angles were used. Blank shapes were defined in accordance with Attar et al. (2021b).

To showcase the non-parametric advantage of the SDF based geometric modelling approach proposed in this study (see Section 2.1), a mix of three geometry subclasses were considered. These subclasses were named standard corners, chamfer corners and stepped sidewalls and could only be described by adopting three different CAD parameterisation schemes. These parameterisations are shown in Fig. 5(a) and further clarification for how chamfer corners were defined is shown in Fig. 5(b). Here, the tool geometries were determined by surface offsets from designed components, and a constant offset of 1.15× blank thickness was used.

2.3.2. Design of experiments

To create datasets of geometries, a Design of Experiments (DoE) was conducted, and variants of the geometries introduced in above were generated. The Latin Hypercube (LHC) DoE technique was used since it is a popular sampling strategy for deterministic computer simulations (Bonte et al., 2007). Using this technique, an approximately uniform distribution of samples within the design space was obtained while reoccurrences were avoided.

Independent datasets for neural network training and testing were created. Three different LHC runs were conducted for each dataset type, one for each geometry subclass. The training data consisted of 400

Table 1
Considered parameters, ranges, and DoE constraints for each parameterisation scheme for the different geometry subclasses.

Parameterisation scheme 1: Standard corners			
Symbol	Description	Bounds (mm)	DoE constraints
r_{die}	Die radius	5–25	None
r_{punch}	Punch radius	7.3–27.3	
r_{plan}	Plan view radius	60–120	
H	Height	60–120	
Parameterisation scheme 2: Chamfer corners			
Symbol	Description	Bounds (mm)	DoE constraints
r_{die}	Die radius	5–25	$1.7C - R_{plan} + 5.4 > 0$
r_{punch}	Punch radius	7.3–27.3	
r_{plan}	Plan view radius	60–250	
C	Chamfer length	60–140	
H	Height	60–120	
Parameterisation scheme 3: Stepped sidewalls			
Symbol	Description	Bounds (mm)	DoE constraints
$r_{die,1}$	Die radius 1	5–20	$SW - r_{punch,1} - r_{die,2} - \epsilon > 0$
$r_{die,2}$	Die radius 2	5–20	$H_1 - r_{die,1} - r_{punch,1} - \epsilon > 0$
$r_{punch,1}$	Punch radius 1	7.3–22.3	$H_2 - r_{die,2} - r_{punch,2} - \epsilon > 0$
$r_{punch,2}$	Punch radius 2	7.3–22.3	$120 - H_1 - H_2 \geq 0$
r_{plan}	Plan view radius	60–120	$\epsilon = 5$ mm, arbitrary compensation for draft angle
H_1	Height 1	30–100	
H_2	Height 2	30–100	
SW	Step width	10–50	

samples for each subclass, and the testing data consisted of 100 samples for each subclass. Since three geometry subclasses were considered, the collocated training dataset consisted of 1200 samples and the collocated testing dataset consisted of 300 samples.

The LHC sampling criterion followed the optimisation technique of maximising the minimum distance between points (MathWorks, 2021). During this maximisation process, empirically determined linear inequality constraints were imposed. The purpose of these constraints was to preserve geometric integrity during CAD model generation (Ramnath et al., 2020, 2019). These constraints, along with the considered parameters and ranges for the LHC uniform distributions, are listed in Table 1. The parameters here correspond to those labelled in Fig. 5. The CAD model generation was automated due to the relatively large number of samples in the datasets. The automation was achieved using parametric CAD models along with the Visual Basic for Applications (VBA) programming language in SolidWorks.

The data generation process was relatively cheap since the proposed model was trained using data from CAD software only. This is unlike other machine learning tasks such as surrogate modelling (Attar et al., 2021b) where data labels are often generated from simulations or physical experiments which could be expensive and thus the dataset can be prone to imbalance due to data scarcity. Consequently, the issue of dataset imbalance could be avoided here by generating plenty of CAD models, using a simple design of experiments scheme, such as LHC. As long as a sufficient number of samples are generated, this approach can ensure that the training dataset is balanced, enabling the model to learn from a representative, efficiently curated, range of examples.

2.4. Neural network architectures

An Auto-Decoder (Park et al., 2019) network f_{θ_1} with network parameters θ_1, was required to approximate an SDF s_i for a stamping geometry indexed by i, given its latent vector (i.e., for its CAD geometry) z_i. This formulation is denoted in Eq. (4) for a query point $\boldsymbol{x} \in \mathbb{R}^3$, and is valid for all points within the considered SDF volume.

$$f_{\theta_1}\left(\boldsymbol{z}_i, \boldsymbol{x}\right) \approx s_i\left(\boldsymbol{x}\right), \forall \boldsymbol{x} \tag{4}$$

Three potential Auto-Decoder architectures for f_{θ_1} were trialled in this study, and these are shown in Fig. 6. These architectures were inspired by the result presented by Park et al. in that increasing the number of skip connections and network depth improved regression accuracy on their dataset (Park et al., 2019).

The Auto-Decoders used here were based on multi-layer perception architectures. The spatial coordinates of a query point $\boldsymbol{x}$ in ambient 3D space were first concatenated with a given latent vector. This concatenated vector was decoded by an Auto-Decoder into a scalar SDF value at the input 3D query point. Performing this forward pass on a full grid of points $\boldsymbol{X}$ would generate the entire SDF $f_{\theta_1}\left(\boldsymbol{z}_i, \boldsymbol{X}\right)$ over that grid. This generated SDF would be conditioned on the latent vector for the geometry indexed by i. Therefore, changing the latent vector (i.e., in an optimisation setting) would change the generated SDF, and this phenomenon can be exploited for optimisation of stamping geometries.

Based on the three architectures in Fig. 6, five different variants of Auto-Decoders were designed and these are summarised in Table 2. The models were trained and their performances were compared and presented in Section 5.

The advantage of adopting an Auto-Decoder is that its formulation is valid for input point clouds of an arbitrary size and distribution. This means that network training and inference can occur with a large number of points densely sampled near to the geometry surface while sparsely sampled far from the surface to allow for better learning performance (see Sections 3.1.2 and 3.2.2). In addition, a uniform grid can also be accepted as input to produce a continuous volumetric SDF, which encompasses the entire stamping die geometry. Further, this grid can be sampled at arbitrary resolution, which is useful if a particular geometry contains small local features which demand high resolutions. These are major advantages over more traditional Auto-Encoders and variants thereof Wang et al. (2020b), since their encoders would expect inputs that are similar to the training data.

2.5. Explicit geometry surface extraction

It was discussed how signed distance fields (SDFs) can be effective at *implicitly* modelling arbitrary shape boundaries and surfaces as their zero-level-sets. However, several stamping applications demand *explicit* surface representations, such as meshes for CAD modelling or FE stamping simulations (El Fakir et al., 2014), and images as inputs to image-based surrogate models (Attar et al., 2021b). Combining the benefits of SDFs with explicit representations requires the use of Marching Cubes (Lorensen and Cline, 1987). Marching Cubes is an algorithm for creating a triangle mesh based representation of a level-set from an implicit field. Since the zero-level-set of SDFs here are designed to implicitly represent the surface geometry, running Marching Cubes on

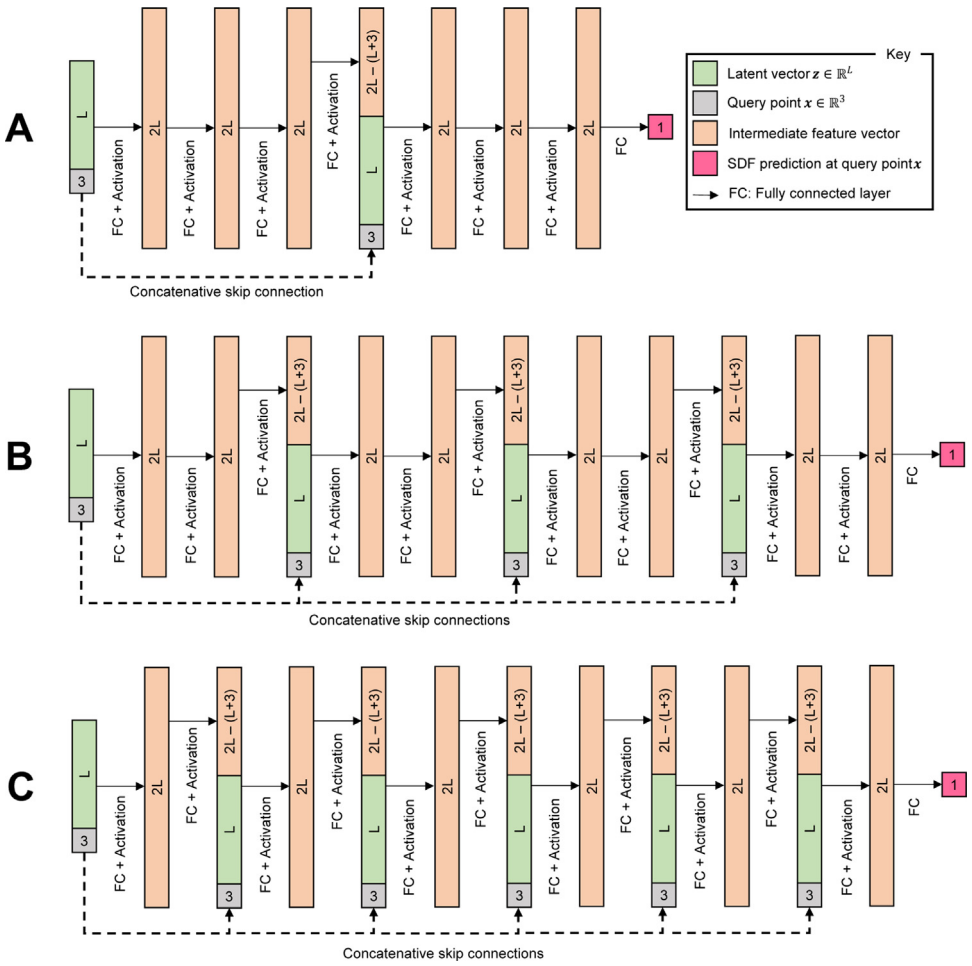


Fig. 6. Three Auto-Decoder architectures considered in this study. Input latent vectors were concatenated with the (x, y, z) coordinates of a 3D query point and the concatenated vector was decoded into an SDF value at the query point by the network. The SDF value was conditioned on the latent vector. L and the numbers denote vector lengths.

Table 2
Variants of Auto-Decoders used in this study, based on the architectures from Fig. 6 and training approaches introduced in Section 3.

Network name	Architecture	Latent vector length L	Activation function	Training approach
Explicit Net 1	A	128	ReLU	Explicit
Explicit Net 2	A	32	ReLU	
Explicit Net 3	B	128	ReLU	
Explicit Net 4	C	128	ReLU	
Implicit Net	A	128	SoftPlus	Implicit

the SDF would extract the implicit surface by converting it to an explicit triangular mesh. The process of generating a volumetric SDF from a latent vector and then extracting its zero-level-set using Marching Cubes is illustrated in Fig. 7. Further details of Marching Cubes can be found in Appendix.

3. Training approaches for learning SDFs of stamping geometries

Inspired by recent computer vision literature on implicit neural representations (Park et al., 2019; Remelli et al., 2020; Gropp et al., 2020; Guillard et al., 2021; Sitzmann et al., 2020), two different approaches for learning SDFs of stamping geometries using neural networks were developed. The first approach focuses on explicitly supervising the network predictions using ground truth SDFs of stamping geometries, which follows the state-of-the-art approach DeepSDF (Park et al., 2019). The second approach exploits key geometric properties of SDFs to learn these SDFs implicitly. The latter approach does not require ground truths and promotes increased accuracy about the zero level-set, which includes around small-scale geometric features.

Previous work has been focused on learning SDFs for watertight surfaces, for example in Park et al. (2019) and Yang et al. (2021). However, stamping die geometries are not watertight since they have bounding edges, and this creates a barrier in representing these geometries using SDFs. To overcome this barrier, a new data pre-treatment method is first introduced here. Given a die geometry, which contains geometric features of a component to be stamped, its top surface was first extracted and exported as an STL mesh file, as shown in Fig. 8(a). The mesh vertices were scaled such that the die edge was of unit length. The scaled mesh was then subsampled into a point cloud and the subsampling was done using the Trimesh Python library (Dawson-Haggerty, 2021). The configuration of the 3D points for training data depended on the approaches used for learning SDFs, and these approaches are detailed in the following subsections.

Since SDF values on the surface are expected to be zero, the surrounding volume was also sampled in order to learn a metric SDF. An example sampled point set is shown in Fig. 8(b), and each point is given an SDF value for demonstration. In what follows, neural networks were

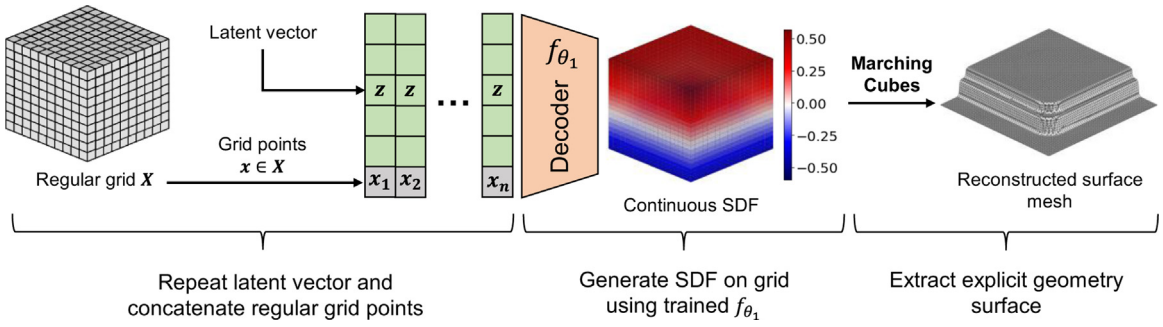


Fig. 7. Illustration of using an Auto-Decoder to generate an SDF for the geometry encoded in z and then extracting that geometry using Marching Cubes.

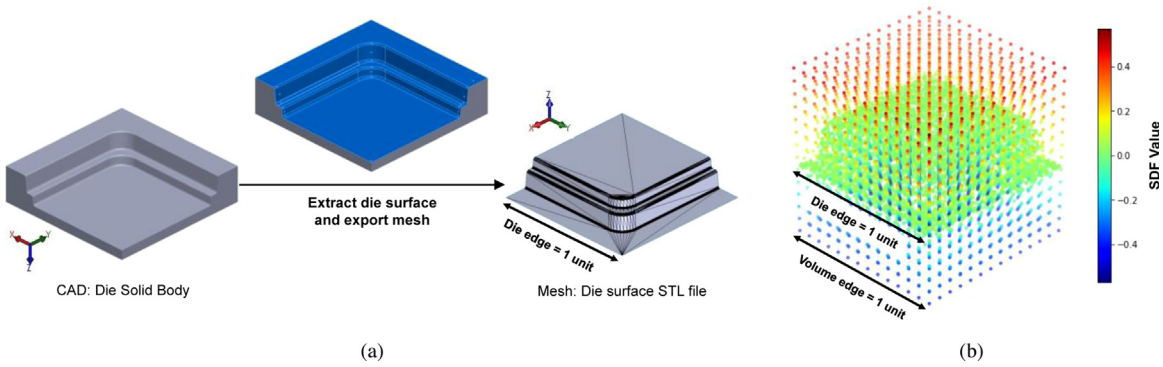


Fig. 8. Example data-pre-treatment (a) the top surface of a die solid body is extracted and exported as an STL mesh and (b) points are sampled on this surface and within the surrounding cubic volume. Crucially, die edge length equals the cubic volume edge length.

employed to predict the SDF values of these points within the cubic volume of unit edge length for different geometries.

Because the surface had zero thickness, points above the surface had positive values and points below the surface had negative values. To make this formulation possible, the surrounding volume was dimensioned such that its side length was equal to a selected characteristic die edge length, as seen in Fig. 8(b). The geometry was positioned in the centre of the volume. This positioning allowed approximately equal amounts of positive and negative SDF values to be predicted by the networks, which has been reported to support training performance (Park et al., 2019).

3.1. Explicit learning approach: regression of SDF values

3.1.1. Approach description

The state-of-the-art approach to modelling 3D SDFs using neural networks is based on learning regressors via explicit supervision of predicted SDF values using ground truths at training time (Park et al., 2019; Remelli et al., 2020; Guillard et al., 2021). This commonly accepted approach was adopted here to understand to what extent it could be used to model SDFs of geometries for sheet stamping applications. The approach involved a two-step data preparation process; first sampling points and then computing ground truth SDF values for each of these points. This process is now detailed below.

3.1.2. Data preparation

The first step was to discretise each geometry sample and its surrounding volume into a point set $\boldsymbol{K}$ which contained points in $\mathbb{R}^3$. A breakdown of the adopted point sampling strategy is demonstrated in Fig. 9 for one geometry. Points were sampled more aggressively in the space near the surface to capture a more detailed SDF around the surface. To do this effectively, 3000 points were first sampled exactly on the geometry surface, where an SDF value of zero is expected. Then, two noisy point sets were further generated, one with low noise and the other with high noise. These sets were generated by perturbing the positions of the surface points by zero mean Gaussian distributions with low and high standard deviations respectively, as shown in Fig. 9. To cover the remainder of the domain, a uniform unit cube with 3000 points was generated. These point sets were collated which resulted in $|\boldsymbol{K}| = 12\,000$ total points for one geometry sample. This point sampling process was repeated for all geometries in the training and testing datasets.

The second step was to analytically compute the SDF values at each of the subsampled points for all geometry samples. For each geometry, the surface points were assigned an SDF value of zero, and their position vectors were stored in a K-dimensional tree (KD-tree). For each off surface point $\boldsymbol{x}_o$, its nearest surface point $\boldsymbol{x}_s$ was identified by querying the KD-tree, and the distance between them was computed by taking the Euclidian norm of their difference. The sign was determined by the sign of the dot product between the normal vector of the surface point $\boldsymbol{n}_s$ and their difference, i.e., sign $\left(\boldsymbol{n}_s \cdot \left(\boldsymbol{x}_o - \boldsymbol{x}_s\right)\right)$.

The positions of the 3D points were taken as network inputs and their analytical SDF values were taken as ground truth targets for this learning approach. The dataset D^i for the ith geometry is summarised by the set shown in Eq. (5).

$$D^i = \left\{\left(\boldsymbol{x}_j, s_j\right) : s_j = \mathrm{SDF}^i\left(\boldsymbol{x}_j\right)\right\} \tag{5}$$

where for each point $\boldsymbol{x}_j$ there was a corresponding s_j. The complete dataset $D^{i^N_{i=1}}$ consisted of N geometry samples represented with analytical signed distance functions $\mathrm{SDF}^{i^N_{i=1}}$ evaluated at each subsampled point $\boldsymbol{x}_j$.

3.1.3. Training and inference

To supervise the SDF predictions explicitly, the clamped L_1 loss function was used as suggested by Park et al. (2019) and is shown in Eq. (6).

$$\mathcal{L}\left(f_{\theta_1}(\boldsymbol{z}, \boldsymbol{x}), s\right) = \left|\mathrm{clamp}\left(f_{\theta_1}(\boldsymbol{z}, \boldsymbol{x}), \delta\right) - \mathrm{clamp}\left(s, \delta\right)\right| \tag{6}$$

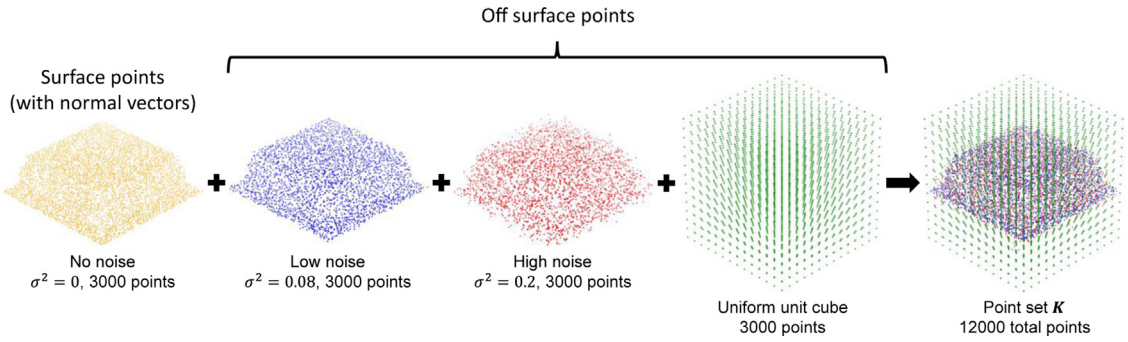


Fig. 9. Breakdown of point subsampling strategy used for explicit learning of SDFs. A total point set is collected, which consists of both points on and off the surface. Here, σ denotes the standard deviation of a zero mean Gaussian distribution added as noise.

In this equation, s and $f_{\theta_1}(\boldsymbol{z}, \boldsymbol{x})$ are the ground truth and predicted SDF values at point $\boldsymbol{x}$ respectively. The function $\text{clamp}(\varepsilon, \delta) := \min(\delta, \max(-\delta, \varepsilon))$ introduces the parameter $\delta > 0$, known as the truncation distance, to control the distance from the surface over which the network is expected to learn the SDF. In this study, $\delta = 0.05$ was used, and this value was inspired from ablation studies performed by Park et al. (2019). This loss function concentrated the networks learning capability on details close to the zero-level set of the SDF.

Using this loss function, an objective function to be minimised at training time was formulated in terms of all geometry samples in a training batch with batch size B, show in Eq. (7).

$$\begin{aligned} &\arg\min_{\theta_1, \{\boldsymbol{z}_i\}_{i=1}^{B}} \frac{1}{B} \sum_{i=1}^{B} \mathcal{L}^{E}_{training}\left(\theta_1, \boldsymbol{z}_i\right) \\ &= \arg\min_{\theta_1, \{\boldsymbol{z}_i\}_{i=1}^{B}} \frac{1}{B} \sum_{i=1}^{B} \left(\frac{1}{|\boldsymbol{K}|} \sum_{\boldsymbol{x}_j \in \boldsymbol{K}} \mathcal{L}\left(f_{\theta_1}\left(\boldsymbol{z}_i, \boldsymbol{x}_j\right), s_j\right) + \lambda \left\|\boldsymbol{z}_i\right\|_2 \right) \end{aligned} \tag{7}$$

where the superscript E denotes this explicit learning approach described. The minimisation was performed with respect to both the network parameters θ_1 and the latent vectors for geometries in the training batch $\{\boldsymbol{z}_i\}_{i=1}^{B}$. The addition of the regularisation term weighted by λ ensured that the latent vector magnitudes were concentrated near the origin. This concentration, which has been reported to help training and inference convergence (Park et al., 2019), enabled similar shapes to have similar latent vectors. As recommended by Park et al. (2019), all latent vectors were initialised from a zero mean Gaussian distribution with low standard deviation $N(0, 0.01^2)$ to further ensure that similar shapes were represented by similar latent vectors.

Eq. (7) was used to train f_{θ_1} to determine the network parameters θ_1 using the explicit learning approach. Minimising Eq. (7) with respect to both θ_1 and $\{\boldsymbol{z}_i\}_{i=1}^{B}$ at training time enabled the latent vector of an unseen geometry to be inferred using the trained decoder f_{θ_1}. At inference time, the network parameters θ_1 were fixed and an optimum latent vector for an unseen geometry was inferred through an iterative optimisation process. This decoder inference process involved minimising the expression shown in Eq. (8) and was conducted iteratively by backpropagating through the trained decoder at each iteration.

$$\arg\min_{\boldsymbol{z}} \mathcal{L}^{E}_{inference}(\boldsymbol{z}) = \arg\min_{\boldsymbol{z}} \frac{1}{|\boldsymbol{K}|} \sum_{\boldsymbol{x}_j \in \boldsymbol{K}} \mathcal{L}\left(f_{\theta_1}\left(\boldsymbol{z}, \boldsymbol{x}_j\right), s_j\right) + \lambda \|\boldsymbol{z}\|_2 \tag{8}$$

A graphical representation for the iterative decoder inference process using this explicit learning approach is shown in Fig. 10. Given an unseen geometry, a point set K which contained points in $\mathbb{R}^3$ was sampled and ground truth SDF values at these points were pre-computed in the same way as was done for the training data. Next, a random vector $\boldsymbol{z}_0 \sim N(0, 0.01^2)$ of length L was sampled from a zero mean Gaussian distribution and this was the initial latent vector to be optimised. For each iteration, the latent vector $\boldsymbol{z}$ was repeated for $|\boldsymbol{K}|$ instances, and the coordinates of each 3D point were concatenated to each instance to form a tensor of size $(L+3) \times |\boldsymbol{K}|$. This tensor was passed through the trained decoder f_{θ_1} to predict SDF values at each of the points. A comparison with ground truth SDF values was performed and the loss $\mathcal{L}^{E}_{inference}(\boldsymbol{z})$ was computed following Eq. (8). Using the gradient of this loss, the latent vector $\boldsymbol{z}$ was iteratively updated by the Adam optimiser to minimise the loss until convergence. At convergence, an optimum latent vector $\boldsymbol{z}^*$ was obtained, which best described the unseen geometry using this approach.

3.2. Implicit learning approach: supervising geometric properties of SDFs

3.2.1. Approach description

The implicit learning approach did not require explicit supervision of SDF values at training time. Instead, the key idea was to learn SDFs implicitly by exploiting and supervising their geometric properties. By learning the intrinsic relationship between the coordinates of points in $\mathbb{R}^3$ and the geometric properties that all ground truth SDFs obey by definition, this approach was expected to provide superior performance over learning a naive regressor (i.e., explicit approach). This approach was inspired by recent work in learning high quality scene and 3D objects reconstruction, such as rooms, tables and chairs with thin features, for computer vision applications (Yang et al., 2021; Gropp et al., 2020; Sitzmann et al., 2020).

Recall the definition and geometric properties of SDFs from Section 2.1. By definition, the SDF values of points that lie exactly on the geometry surface are zero. The magnitude of spatial gradient vectors of SDFs are equal to one and these vectors at the geometry surface align with the surface normals. Together, the definition and properties describe attributes of SDFs that can be classified into two categories: off the geometry surface and on the surface. The dataset and loss function for this approach were prepared by considering these two attribute categories.

3.2.2. Data preparation

To prepare the dataset for this approach, points in $\mathbb{R}^3$ were subsampled on and around all geometry samples and this step was similar to the one used for the explicit learning approach. However, here the off surface and on surface points were kept separate and stored as two point sets to facilitate learning SDF attributes for these two categories. A breakdown of this point subsampling strategy is shown in Fig. 11 for one geometry sample. 9000 total off surface points were subsampled according to the figure, and an additional 9000 surface points were subsampled and stored separately. The surface points also came with normal vectors as part of the subsampling process. This point subsampling process was repeated for all geometries in the training and testing datasets. The deviation from the attributes of SDFs was penalised solely based on the predicted SDF values. Therefore, this approach did not need ground truth SDF samples and can be considered as self-supervised. Details on how this self-supervised approach was employed are provided in the following.

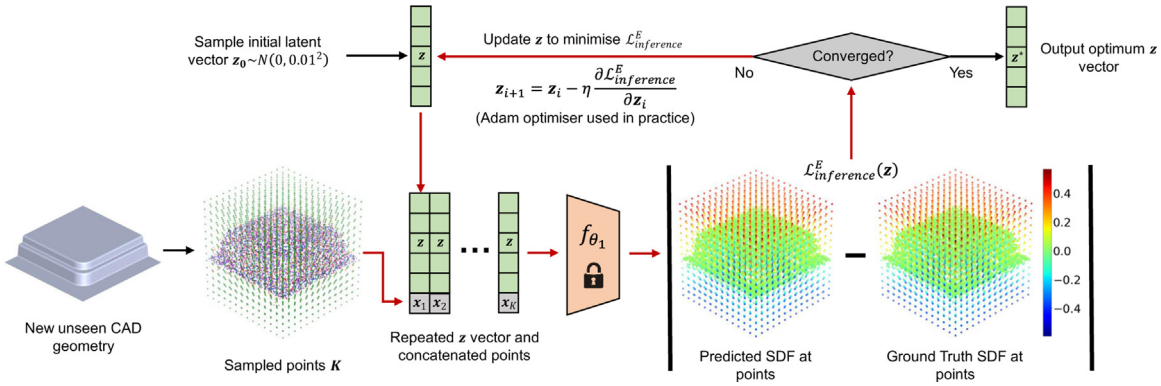


Fig. 10. Decoder inference process to obtain a latent vector representation of an unseen geometry using a trained network f_{θ_1} which was trained using the explicit learning approach. Red arrows occur once per inference iteration.

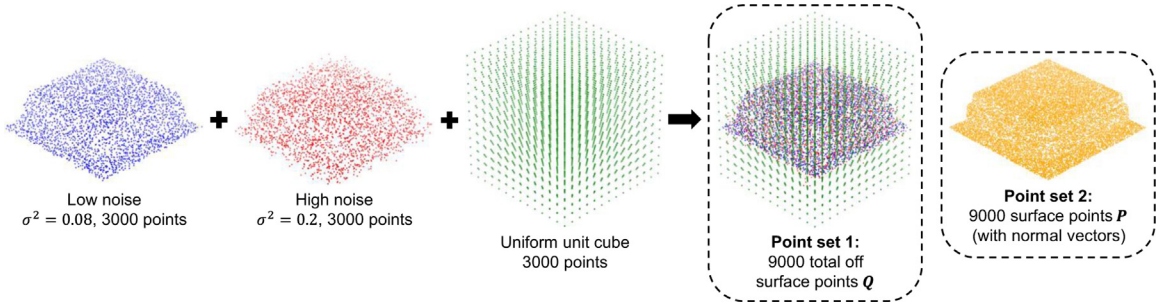


Fig. 11. Breakdown of point subsampling strategy used for implicit learning of SDFs. Two point sets were sampled: off surface points, and surface points with surface normal vectors. Here, σ denotes the standard deviation of a zero mean Gaussian distribution added as noise for off surface points close to the surface.

3.2.3. Training and inference

To effectively learn the attributes of SDFs, a new loss function to be minimised was cast in terms of the aforementioned point categories, surface, and off surface points, as shown in Eq. (9). As before, the loss was formulated in terms of all geometry samples in a training set batch with batch size B. The minimisation was performed with respect to both the network parameters θ_1 and the latent vectors for geometries in the training batch $\{z_i\}_{i=1}^{B}$. The three terms in Eq. (9) are separately explained in detail in the following paragraphs.

$$\begin{aligned}&\underset{\theta_1,\{z_i\}_{i=1}^{B}}{\arg\min}\frac{1}{B}\sum_{i=1}^{B}\mathcal{L}^{I}_{training}\left(\theta_1,z_i\right)\\&=\underset{\theta_1,\{z_i\}_{i=1}^{B}}{\arg\min}\frac{1}{B}\sum_{i=1}^{B}\left(\mathcal{L}_{surface}\left(\theta_1,z_i\right)+\mathcal{L}_{offsurface}\left(\theta_1,z_i\right)+\mathcal{L}_{reg}\left(z_i\right)\right)\end{aligned} \tag{9}$$

The first term in Eq. (9) contained a loss concerned with the behaviour of the predicted SDF at the geometry surface $\mathcal{L}_{surface}\left(\theta_1,z_i\right)$, and this term is shown in Eq. (10).

$$\begin{aligned}\mathcal{L}_{surface}\left(\theta_1,\boldsymbol{z_i}\right)=&\frac{1}{|\boldsymbol{P}|}\sum_{\boldsymbol{x}_j\in\boldsymbol{P}}\Big(\lambda_1\left|f_{\theta_1}\left(z_i,\boldsymbol{x}_j\right)\right|\\&+\lambda_2\alpha_j\left\|\nabla_{\boldsymbol{x}_j}f_{\theta_1}\left(z_i,\boldsymbol{x}_j\right)-\boldsymbol{n}_j\right\|_2\Big)\end{aligned} \tag{10}$$

This surface loss ensured supervision of surface attributes by penalising deviations from them. This loss was formulated on the surface point set $\boldsymbol{P}$ and contained two terms, weighted by the scalars λ_1 and λ_2 respectively. Minimising the first term ensured that the magnitude of an SDF prediction $f_{\theta_1}\left(z_i,\boldsymbol{x}_j\right)$ for point $\boldsymbol{x}_j\in\boldsymbol{P}$ on the surface of the geometry encoded in z_i tended toward zero, which satisfies the definition of SDFs. The second term ensured that the spatial gradient vector of the predicted SDF at point $\boldsymbol{x}_j\in\boldsymbol{P}$ on the surface $\nabla_{\boldsymbol{x}_j}f_{\theta_1}\left(z_i,\boldsymbol{x}_j\right)$ aligned with the unit normal vector at that point $\boldsymbol{n}_j$. The L_2 norm $\|\cdot\|_2$ was taken between the two to allow alignment in both vector direction and magnitude.

So far, the importance of certain geometric features on manufacturing performance has not been considered during network training. For stamping applications, fillet radii are geometric features which are highly influential in determining manufacturing performance, as explained in the introduction. These features are small in scale when considered in the context of the global component geometry. This means that a surface subsampled with an oriented point cloud as shown in Point set 2 in Fig. 11 would have relatively few points on these small radii regions.

To address the imbalance and consider small local geometric features in Eq. (10), an additional scaling factor α_j was introduced to operate pointwise. This scaling factor was designed to provide greater weight to small-scale regions on the geometry surface where accurate reconstructions are essential, such as fillet radii. The value of α_j was set to a scalar $\beta>1$ if the conditions in Eq. (11) were met for point $\boldsymbol{x}_j$.

$$\alpha_j=\begin{cases}\beta, & \frac{1}{n_p}\sum_{k=1}^{n_p}\cos^{-1}\left(\boldsymbol{n}_j\cdot\boldsymbol{n}_k\right)\geq\tau\\1, & \frac{1}{n_p}\sum_{k=1}^{n_p}\cos^{-1}\left(\boldsymbol{n}_j\cdot\boldsymbol{n}_k\right)<\tau\end{cases} \tag{11}$$

In this expression, $\boldsymbol{n}_j$ is the unit normal vector of point $\boldsymbol{x}_j$ and $\left(\boldsymbol{n}_j\cdot\boldsymbol{n}_k\right)$ represents the dot product between this vector and the unit normal vector of the kth nearest point to $\boldsymbol{x}_j$. This dot product represents the cosine of the angle between the two vectors. Using the dot product, the angle between the normal of every point and the normals of n_p nearest points was calculated. The average of these angles was taken to compute a single angle that represented the overall "sharpness" of

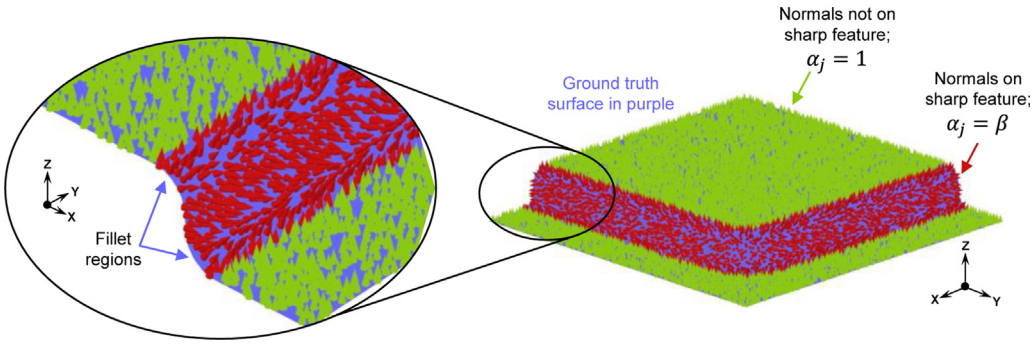


Fig. 12. Cone plots showing surface unit normal vectors pointing away from the surface. Cones are stemming from every subsampled surface point. Normals on flat surfaces in green, aligning with a uniform direction. Normals on fillet radii and sidewall regions in red, corresponding to small-scale features.

the area surrounding each point. This value was then compared with a threshold angle τ, and if greater than the threshold, the point was considered to belong to a small-scale feature. In this study, $\tau = 2°$ and $\beta = 3$ were used. These settings meant that the second term in Eq. (10) was scaled by a factor of 3 for points $\boldsymbol{x}_j \in \boldsymbol{P}$ that belonged to small-scale features. Fig. 12 graphically illustrates the regions detected for scaling. The cones represent the surface normal vectors at each subsampled point on the geometry surface. The red zones show the identified fillet and sidewall regions that are scaled by Eq. (11), while the green zones indicate unscaled surfaces with uniformly aligned normals. This scaling strategy emphasises the alignment of normals on locally curved areas, such as radii and sidewalls.

The second term in Eq. (9) contained a loss concerned with the behaviour of the predicted SDF away from the geometry surface $\mathcal{L}_{off\,surface}\left(\theta_1, \boldsymbol{z}_i\right)$, shown in Eq. (12).

$$\mathcal{L}_{off\,surface}\left(\theta_1, \boldsymbol{z}_i\right) = \frac{\lambda_3}{|\boldsymbol{Q}|} \sum_{\boldsymbol{x}_k \in \boldsymbol{Q}} \left(\left\| \nabla_{\boldsymbol{x}_k} f_{\theta_1}\left(\boldsymbol{z}_i, \boldsymbol{x}_k\right) \right\|_2 - 1 \right)^2 \tag{12}$$

Minimising this term ensured that the spatial gradient vector of the predicted SDF at point $\boldsymbol{x}_k \in \boldsymbol{Q}$ away from the surface $\nabla_{\boldsymbol{x}_k} f_{\theta_1}\left(\boldsymbol{z}_i, \boldsymbol{x}_k\right)$ had a unit magnitude, which satisfied the solution to the Eikonal equation (see Eq. (2)). The magnitude of the spatial gradient vector was computed by taking the L_2 norm $\|\cdot\|_2$.

The final term in Eq. (9) was a regularisation term that was added for the same reasons mentioned in Section 3.1.3, shown in Eq. (13). The λ terms in the presented equations denoted scalar weights that were manually tuned to balance the magnitudes of each of the terms that make up Eq. (9).

$$\mathcal{L}_{reg}\left(\boldsymbol{z}_i\right) = \lambda_4 \left\| \boldsymbol{z}_i \right\|_2 \tag{13}$$

Eq. (9) was used to train f_{θ_1} to determine the network parameters θ_1 using the implicit learning approach. At inference time, similar to the explicit learning approach, the network parameters θ_1 were fixed and an optimum latent vector for an unseen geometry was inferred through an iterative optimisation process. This decoder inference process involved minimising the expression shown in Eq. (14) and was conducted iteratively by backpropagating through the trained decoder at each iteration.

$$\begin{aligned} \arg\min_{\boldsymbol{z}} \mathcal{L}^{I}_{inference}\left(\boldsymbol{z}\right) = \arg\min_{\boldsymbol{z}} \frac{1}{|\boldsymbol{P}|} \sum_{\boldsymbol{x}_j \in \boldsymbol{P}} \Big(& \lambda_1 \left| f_{\theta_1}\left(\boldsymbol{z}, \boldsymbol{x}_j\right) \right| \\ & + \lambda_2 \alpha_j \left\| \nabla_{\boldsymbol{x}_j} f_{\theta_1}\left(\boldsymbol{z}, \boldsymbol{x}_j\right) - \boldsymbol{n}_j \right\|_2 \Big) \\ + \frac{\lambda_3}{|\boldsymbol{Q}|} \sum_{\boldsymbol{x}_k \in \boldsymbol{Q}} & \left(\left\| \nabla_{\boldsymbol{x}_k} f_{\theta_1}\left(\boldsymbol{z}, \boldsymbol{x}_k\right) \right\|_2 - 1 \right)^2 + \lambda_4 \left\| \boldsymbol{z} \right\|_2 \end{aligned} \tag{14}$$

A graphical representation for the iterative decoder inference process using this implicit learning approach is shown in Fig. 13. An unseen geometry was first sampled into the two point sets, as explained above for the training data. Similar to the explicit learning approach, a random vector $\boldsymbol{z}_0 \sim N\left(0, 0.01^2\right)$ of length L was sampled from a zero mean Gaussian distribution and this was the initial latent vector to be optimised. For each iteration, and for each of the two point sets, the latent vector $\boldsymbol{z}$ was repeated for as many instances as there were points in each point set. The coordinates of each 3D point were concatenated to each instance to form tensors of size $(L+3)\times|\boldsymbol{P}|$ and $(L+3)\times|\boldsymbol{Q}|$ for the surface and off surface point sets respectively. These tensors were passed through the trained decoder f_{θ_1} and SDF values were predicted at each point in the two point sets. The spatial gradients of these predictions were then taken as necessary and the loss function $\mathcal{L}^{I}_{inference}\left(\boldsymbol{z}\right)$ in Eq. (14) was computed. Using the gradient of this loss, the latent vector $\boldsymbol{z}$ was iteratively updated by the Adam optimiser to minimise the loss until convergence. At convergence, an optimum latent vector $\boldsymbol{z}^*$ was obtained, which best described the unseen geometry using this approach.

4. Design considerations for training and training history

4.1. Training settings

The Auto-Decoders f_{θ_1} introduced in Section 2.4 were trained using both training approaches introduced in Section 3. For the explicit learning approach, the ReLU activation function was used as recommended by Park et al. (2019). For the implicit learning approach, the SoftPlus activation function was used since it is more differentiable than the commonly used ReLU function. This enhanced differentiability was conjectured to be useful for the implicit learning approach since computing the loss function first involved computing gradients of the predicted SDFs. These gradients were computed via backpropagation through f_{θ_1} using automatic differentiation (Pytorch, 2021).

For both approaches, the network parameters θ_1 and latent vectors $\left\{\boldsymbol{z}_i\right\}_{i=1}^{B}$ were iteratively updated during training and these updates occurred after each training batch. During this iterative process, the optimiser sought to find the combinations of θ_1 and $\left\{\boldsymbol{z}_i\right\}_{i=1}^{B}$ that best minimised the loss function for each training approach. In this way, the networks were able to learn compact functions that could predict SDFs of families of box corner geometries. The networks were trained in the PyTorch framework using the commonly recommended Adam optimiser (Kingma and Ba, 2015) with default beta parameters of $\beta_1 = 0.9$ and $\beta_2 = 0.999$. An initial learning rate of 1×10^{-4} was used and this was halved every 500 epochs up to a minimum of 5×10^{-6}. These learning rates were the same for both θ_1 and $\left\{\boldsymbol{z}_i\right\}_{i=1}^{B}$ updates. All parameters in the loss functions for both approaches were determined empirically and are summarised in Table 3.

4.2. Dataset ordering

Gradient updates by the Adam optimiser at training time occurred after each training batch, and each training batch was presented to the networks sequentially during training. Since geometries from different subclasses were considered (see Section 2.3.1), careful attention was

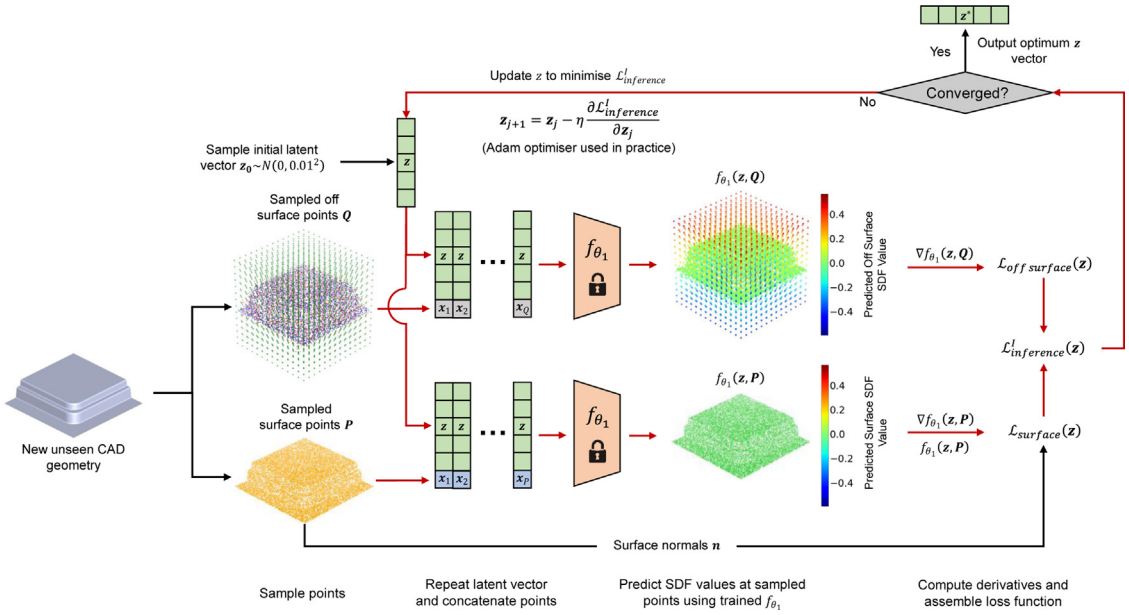


Fig. 13. Decoder inference process to obtain a latent vector representation of an unseen geometry using a network f_{θ_1} using the implicit learning approach. Red arrows occur once per inference iteration.

Table 3
Empirically determined loss function parameters for explicit and implicit training approaches.

Training approach	Parameter	Symbol	Value
Explicit	SDF truncation distance	δ	0.05
	Latent vector regularisation loss term weight	λ	1×10^{-4}
Implicit	Surface SDF loss term weight	λ_1	3
	Normals loss term weight	λ_2	0.7
	Eikonal loss term weight	λ_3	0.9
	Latent vector regularisation loss term weight	λ_4	1×10^{-4}
	Point weight for normals loss term	β	3
	Threshold angle between normals (°)	τ	2

paid to ordering the data samples in the training dataset. The training data was ordered such that each batch contained an equal number of geometry samples from each geometry subclass. This was achieved by periodically arranging samples from each geometry subclass in the dataset, as shown in Fig. 14. Further, since there were 3 subclasses in total, the batch size which chosen to be a multiple of 3. Here, a batch size of $B = 21$ geometries was selected. This selection ensured gradient updates contained information on 7 random (random because of Latin Hypercube sampling) samples from each of the 3 subclasses. This batch size allowed learning SDFs from all subclasses uniformly while being small enough to efficiently fit in GPU memory.

4.3. Training history

Fig. 15 shows the training loss histories for both the explicit and implicit learning training approaches. The curve in Fig. 15(a) belongs to Explicit Net 1 (see Table 2) and is representative of all networks trained using the explicit learning approach. The steady declines in the loss values provide evidence of good training stability for both approaches. Training for both approaches was left to run for 24 h and was run on an NVIDIA P100 GPU in the Google Colab environment. In this timeframe, the explicit learning approach completed 8000 epochs (i.e., 8000 complete iterations of the training dataset), while the implicit approach completed 3800 epochs. The implicit approach completed less epochs in the same 24 h training timeframe because spatial gradients of the SDF predictions had to be calculated before assembling the loss function at each batch iteration, as explained earlier. However, convergence was seen to occur at approximately 1500 epochs for the implicit approach, while the explicit approach took approximately 7000 epochs. A further study on the effect of the number of subsampling points and number of geometries in the training dataset on the SDF generation accuracy and training time was outside the scope of this work. Nevertheless, it is expected that significant training speeds ups could be achieved by using fewer subsampling points and geometries but may compromise performance accuracy.

5. Evaluation of shape representation performance

After successfully training the Auto-Decoder networks f_{θ_1} using both of the introduced training approaches, their performances were evaluated quantitatively and qualitatively.

5.1. Quantitative evaluation

For the quantitative evaluation, similarity metrics were used to measure the difference between geometries reconstructed from f_{θ_1} and the ground truth geometries exported from CAD software. These metrics were inspired by recent literature on implicit neural representations (Park et al., 2019; Peng et al., 2021; Gropp et al., 2020) and were Chamfer Distance, Hausdorff Distance and Mesh Similarity, and are defined below.

5.1.1. Performance metrics definition

The distance metrics, Chamfer and Hausdorff, required the continuous geometry surfaces to be first discretised into point sets $\boldsymbol{P}_{GT}$ and $\boldsymbol{P}_R$ for ground truth and reconstructed geometries respectively. The Mesh Similarity metric required $\boldsymbol{P}_{GT}$ and the reconstructed surface mesh $\mathcal{M}$. To generate $\boldsymbol{P}_{GT}$, points were sampled on the surface of

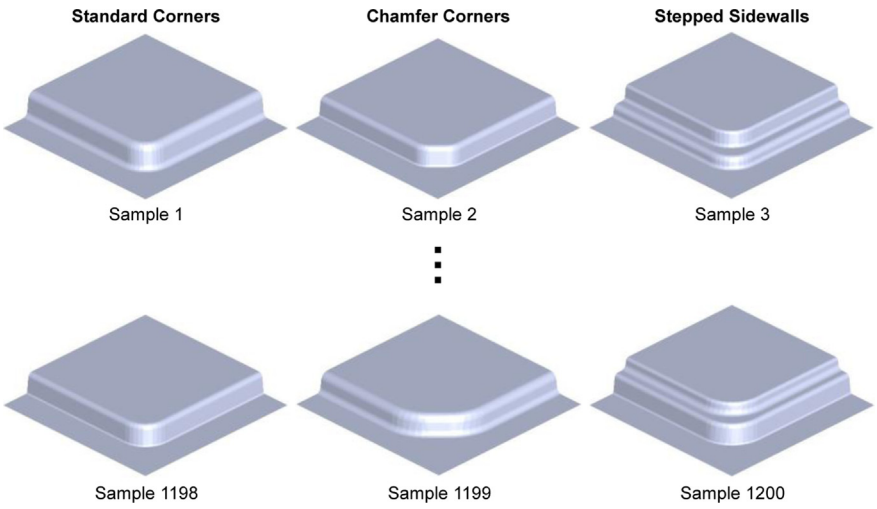
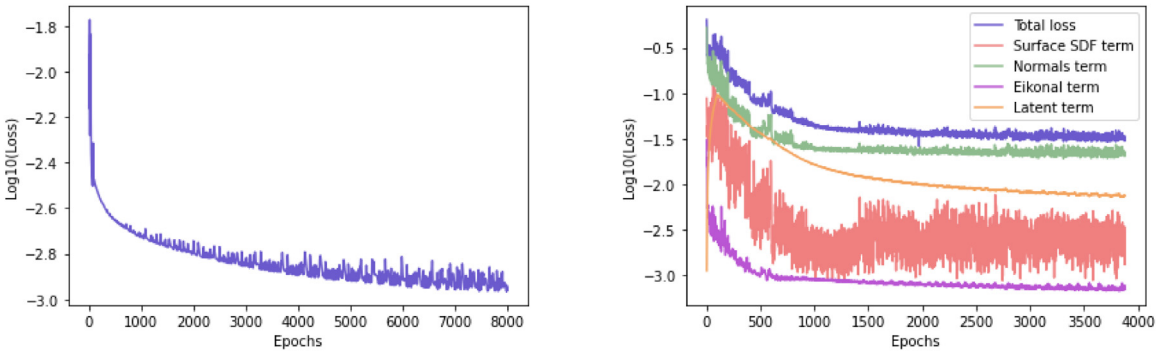


Fig. 14. Ordering of geometries in the training dataset; random geometries from all 3 geometry subclasses were arranged periodically. Note that CAD geometries are shown here for clarity whereas the training dataset contained point clouds sampled from these geometries.



(a) Explicit learning approach (b) Implicit learning approach

Fig. 15. Loss histories for the proposed neural networks trained using the two learning approaches. The plots show (a) the history of Eq. (7) and (b) the history of Eq. (9). Since the loss of the implicit learning approach is more complex, histories of each loss term that made up the total loss are also presented. To better visualise meaningful decreases in the loss function terms, the Log10 of the losses were computed and plotted.

the STL mesh file that was exported from CAD software. To generate $\boldsymbol{P}_R$, a forward pass was first required through f_{θ_1} to generate the continuous SDF. Then, the marching cubes algorithm was used to extract the reconstructed surface mesh $\mathcal{M}$. Finally, points were sampled on $\mathcal{M}$. A generous 30,000 points were sampled for both $|\boldsymbol{P}_{GT}|$ and $|\boldsymbol{P}_R|$ to effectively approximate the continuous surfaces. The point set generating process is depicted in the black rectangles in Fig. 16, and what is needed for computing each similarity metric is shown in the green rectangle.

a. Chamfer distance

The Chamfer Distance measured the difference between reconstruction and ground truth surfaces by considering the distances of all sampled points. Given the two pre-computed point sets $\boldsymbol{P}_{GT}$ and $\boldsymbol{P}_R$, this metric was computed by taking the sum of the squared distances between each point in one set and its nearest in the other set using Eq. (15).

$$d_C\left(\boldsymbol{P}_{GT}, \boldsymbol{P}_R\right) = \sum_{\boldsymbol{x}_i \in \boldsymbol{P}_{GT}} \min_{\boldsymbol{x}_j \in \boldsymbol{P}_R} \left\|\boldsymbol{x}_i - \boldsymbol{x}_j\right\|_2^2 + \sum_{\boldsymbol{x}_j \in \boldsymbol{P}_R} \min_{\boldsymbol{x}_i \in \boldsymbol{P}_{GT}} \left\|\boldsymbol{x}_i - \boldsymbol{x}_j\right\|_2^2 \quad (15)$$

The two terms being summed are each the one sided chamfer distances and summing ensures the distance is symmetric $d_C\left(\boldsymbol{P}_{GT}, \boldsymbol{P}_R\right) = d_C\left(\boldsymbol{P}_R, \boldsymbol{P}_{GT}\right)$. The nearest distances were computed efficiently by the use of a KD-tree. The lower the Chamfer Distance, the better the reconstruction performance.

b. Hausdorff distance

The Hausdorff Distance measured the maximum distance between reconstructions and ground truth surfaces. Given the two pre-computed point sets $\boldsymbol{P}_{GT}$ and $\boldsymbol{P}_R$, this metric was computed by taking the maximum distance between any point in one set and its nearest in the other set using Eq. (16).

$$d_H\left(\boldsymbol{P}_{GT}, \boldsymbol{P}_R\right) = \max\left(\max_{\boldsymbol{x}_i \in \boldsymbol{P}_{GT}} \min_{\boldsymbol{x}_j \in \boldsymbol{P}_R} \left\|\boldsymbol{x}_i - \boldsymbol{x}_j\right\|_2, \max_{\boldsymbol{x}_j \in \boldsymbol{P}_R} \min_{\boldsymbol{x}_i \in \boldsymbol{P}_{GT}} \left\|\boldsymbol{x}_i - \boldsymbol{x}_j\right\|_2\right) \quad (16)$$

The two terms in the bracket are each the one sided Hausdorff distances and taking the $\max(\cdot)$ of both ensures the distance is symmetric $d_H\left(\boldsymbol{P}_{GT}, \boldsymbol{P}_R\right) = d_H\left(\boldsymbol{P}_R, \boldsymbol{P}_{GT}\right)$. This metric was computed using functionality from the Scipy Python library. The lower the Hausdorff Distance, the better the reconstruction performance.

c. Mesh similarity

The Mesh Similarity measured the accuracy of normals from the reconstructed mesh $\mathcal{M}$. It was defined as the mean dot product between the normals of points in $\boldsymbol{P}_{GT}$, which were treated as references, and the normals of the nearest faces of $\mathcal{M}$. This metric was computed by Eq. (17)

$$\text{Mesh. sim}\left(\boldsymbol{P}_{GT}, \mathcal{M}\right) = \frac{1}{\left|\boldsymbol{P}_{GT}\right|} \sum_{\boldsymbol{x}_i \in \boldsymbol{P}_{GT}} \max\left(\boldsymbol{n}_{F_i} \cdot \boldsymbol{n}_{\boldsymbol{x}_i}, -\boldsymbol{n}_{F_i} \cdot \boldsymbol{n}_{\boldsymbol{x}_i}\right) \quad (17)$$

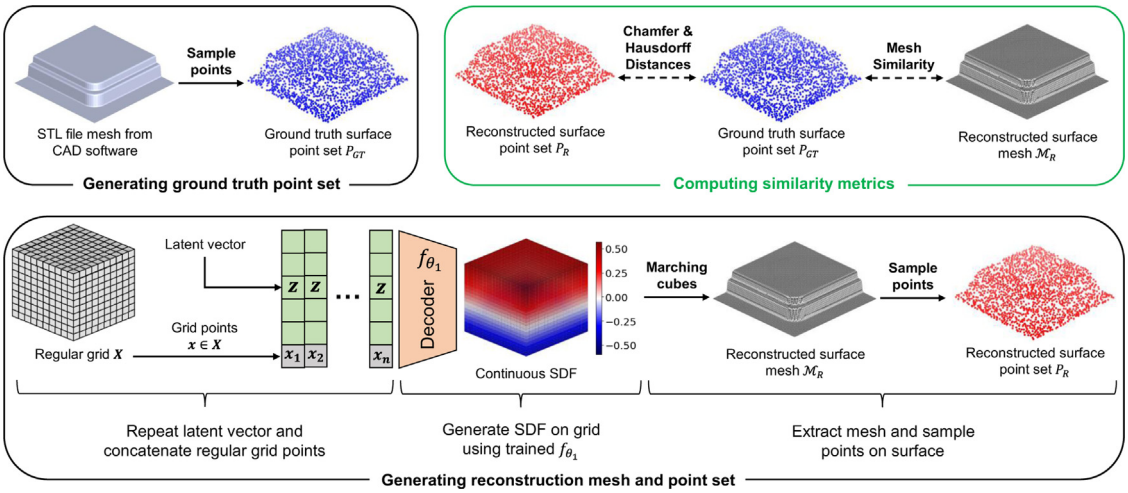


Fig. 16. Generating data (black rectangles) to compute similarity metrics (green rectangle). This process was repeated for all geometries in the training and testing datasets.

Table 4
Mean values ± standard deviations of selected values from Fig. 18.

Auto-Decoder name	Chamfer distance ≤ 1.5e−4 (Black dashed line in Fig. 18(a))	Chamfer distance ≤ 2.5e−4 (Black dotted line in Fig. 18(a))	Normals similarity ≥ 0.9 (Black dotted line in Fig. 18(b))
Explicit Net 1	0.19 ± 0.19	0.69 ± 0.20	0.93 ± 0.03
Explicit Net 2	0.40 ± 0.16	0.84 ± 0.07	0.93 ± 0.05
Explicit Net 3	0.11 ± 0.16	0.44 ± 0.32	0.94 ± 0.03
Explicit Net 4	0.25 ± 0.18	0.74 ± 0.21	0.95 ± 0.04
Implicit Net	0.85 ± 0.03	1.00 ± 0.02	1.00 ± 0.01

where F_i is the nearest face of $\mathcal{M}$ to point $\boldsymbol{x}_i \in \boldsymbol{P}_{GT}$, and each $\boldsymbol{n} \in \mathbb{R}^3$ is a unit normal vector. The nearest faces were efficiently computed using functionality from the Trimesh Python library (Dawson-Haggerty, 2021). Further, recall that $\mathcal{M}$ was extracted from the generated SDF using the Marching Cubes algorithm (see Fig. 16). This algorithm does not guarantee oriented normals (Park et al., 2019) on $\mathcal{M}$ faces. To allow for misorientations due to Marching Cubes, the $\max(\cdot)$ dot product using the normal generated from Marching Cubes $\boldsymbol{n}_{F_i}$ and its flipped counterpart $-\boldsymbol{n}_{F_i}$ was computed prior to the summation, as seen in Eq. (17). The mesh similarity metric ranged from 0 to 1, where 1 meant perfectly matching normals.

5.1.2. Quantitative evaluation results

The aforementioned metrics were computed across all geometries in the training and testing datasets, and for all Auto-Decoder configurations. This computation resulted in metric values for each geometry and these values were collected as distributions and are visualised the violin plots in Fig. 17.

Several noteworthy conclusions can be drawn from Fig. 17. Excellent agreement between performance on training and testing datasets for all Auto-Decoder configurations can be seen when comparing Figures (a) and (b). This agreement provides evidence for the inexistence of overfitting and suggests the Auto-Decoders did indeed correctly learn the geometry space. No clear gain in performances can be seen between Auto-Decoders trained using the explicit learning approach (Explicit Nets 1–4). Although, Explicit Net 2 performed marginally better in terms of Chamfer Distance but was on par with its counterparts for the other two metrics. On the other hand, Implicit Net performed significantly better than all Explicit Nets across all metrics. This vast improvement in performance suggests the implicit learning approach was highly effective at learning stamping geometries when compared to the explicit learning approach. This result is further unpacked in the following.

Fig. 18 quantifies the fraction of surface coverage that meets performance metrics ρ between reconstructions and ground truths across all geometry samples in the testing dataset. In Figure (a), ρ represents unsummed Chamfer Distance and is plotted on the X-axis. Plotted on the Y-axis is the fraction of points $\boldsymbol{x} \in \boldsymbol{P}_R$ that satisfy $\overrightarrow{d_C}(\boldsymbol{x}, \boldsymbol{P}_{GT}) \leq \rho$, where $\overrightarrow{d_C}(\boldsymbol{x}, \boldsymbol{P}_{GT})$ is the one sided Chamfer Distance from a point $\boldsymbol{x} \in \boldsymbol{P}_R$ to $\boldsymbol{P}_{GT}$, i.e., the elements being summed in the second term of Eq. (15). In Figure (b) ρ represents unsummed Normals Similarity and is plotted on the X-axis. Plotted on the Y-axis is the fraction of faces $F \in \mathcal{M}$ that satisfy $\max(\boldsymbol{n}_F \cdot \boldsymbol{n}_{\boldsymbol{x}}, -\boldsymbol{n}_F \cdot \boldsymbol{n}_{\boldsymbol{x}}) \geq \rho$ and here $\boldsymbol{x}$ is the closest point in $\boldsymbol{P}_{GT}$ to face F.

Once again, Implicit Net significantly outperformed all Explicit Net variants in both performance metrics. The plots in Fig. 18 show that Implicit Net had higher mean values and tighter standard deviation bands across all X-axis values. These outcomes suggest consistently high performance across all unseen geometries in the testing dataset. Results at the black dashed and dotted lines in Fig. 18 are reported in Table 4 for further quantitative evaluation.

The excellent performance of Implicit Net on the testing dataset and across all metrics suggests that this network could generate realistic geometries. A qualitative evaluation was next performed to further investigate this suggestion.

5.2. Qualitative evaluation

To interpret the results of the quantitative evaluation, a qualitative evaluation was performed by visualising reconstructed surface meshes $\mathcal{M}$ obtained from the trained Auto-Decoder networks. Fig. 19 compares surface meshes $\mathcal{M}$ reconstructed from Explicit Net 1 and Implicit Net with points sampled from the ground truth CAD surfaces $\boldsymbol{P}_{GT}$. The figure shows that although the surfaces from Explicit Net were mostly in agreement with the ground truth points, there were differences in local areas, such as radii. Some of these differences are indicated by the arrows in Fig. 19(a). On the other hand, excellent qualitative correspondence was found between the surfaces from Implicit Net and the ground truth points, seen in Fig. 19(b).

For further qualitative evaluation, Fig. 20 compares 2D image projections of three surfaces which were reconstructed from Explicit Nets 1–2 and Implicit Net with 2D projections of their CAD ground truth

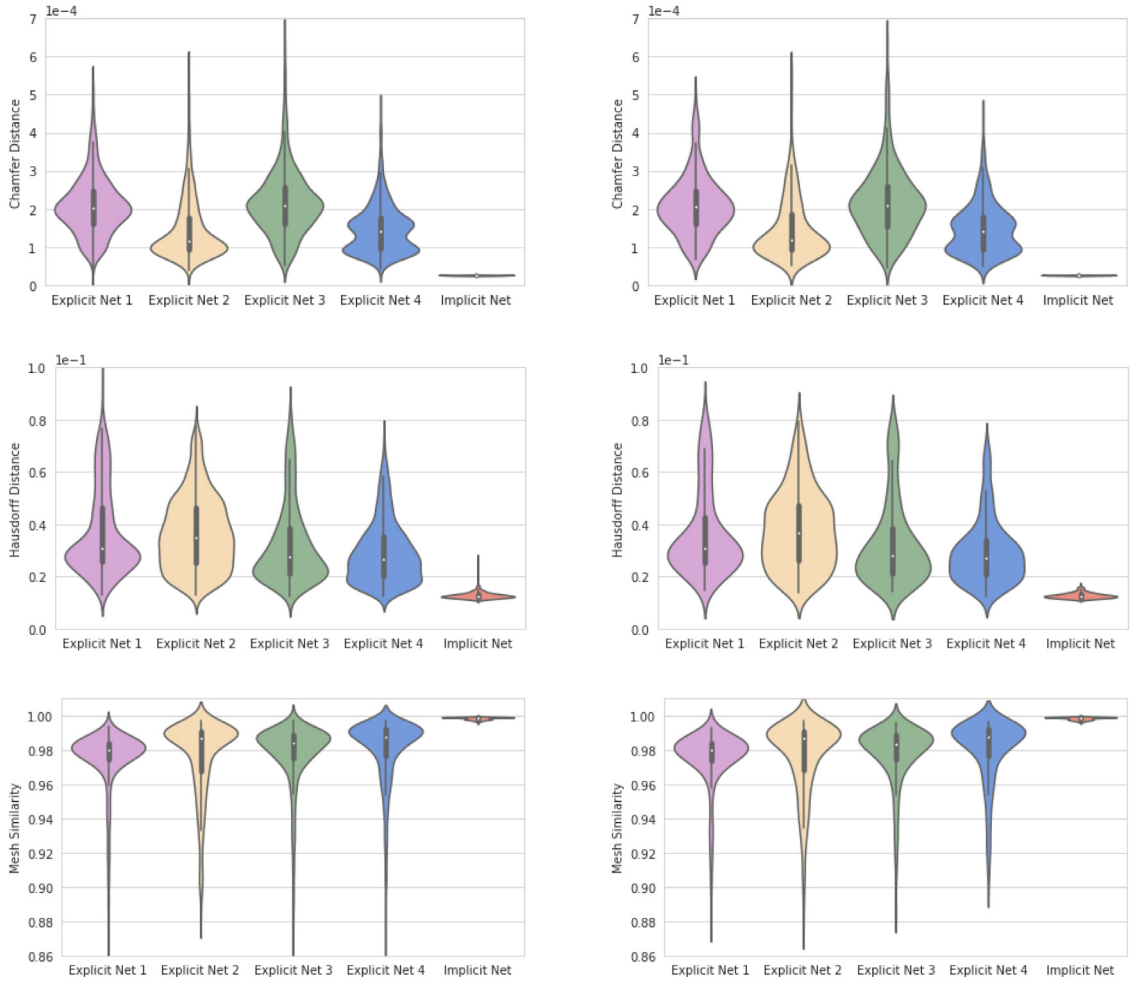


Fig. 17. Distributions of Chamfer Distance (top row), Hausdorff Distance (middle row) and Mesh Similarity (bottom row) for networks trained using the explicit learning approach (Explicit Nets 1–4) and implicit learning approach (Implicit Net).

counterparts. It was found that all the networks did indeed reconstruct geometries well on the global scale. The height values agreed between reconstructions and ground truths, and the reconstructions did replicate box corner geometries. However, this figure further confirms that reconstructions from Explicit Nets 1–2 failed to capture local radii features, as seen from their difference images. In contrast, near indistinguishable images are seen when comparing the reconstructions of Implicit Net with ground truths.

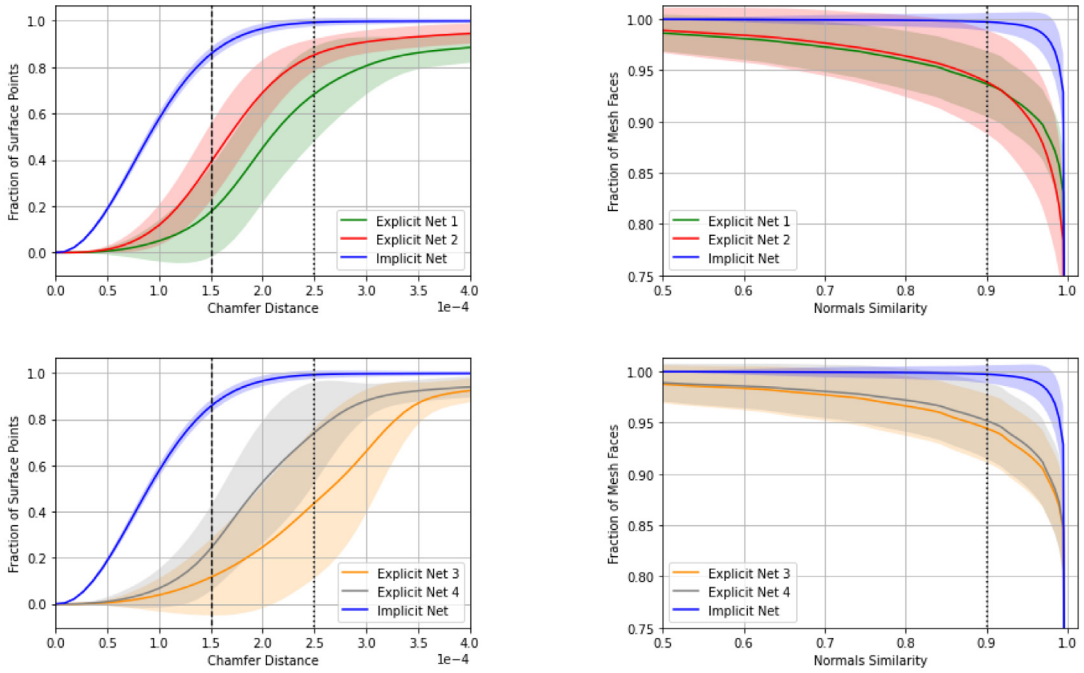
The impressive performance achieved by Implicit Net can be attributed to its implicit learning approach, as described in Section 3.2, which supervises the geometric properties of the generated SDFs instead of the SDF value itself. In particular, the loss function used in the training of the model contains a term that operates on the zero-level-set of the SDF directly, as shown in Eq. (10), which corresponds to the surface geometry of the shape. By supervising the surface geometry, the model is able to capture local geometric features more accurately, resulting in more structurally sound geometries. Additionally, the surface normal scaling factor α_j in Eq. (10) further promotes the accurate reconstruction of local geometric features on the surface.

One the other hand, the conventional approach for training a model for shape generation using SDFs is to supervise the SDF value at each point in space with respect to the ground truth SDF, as used by DeepSDF (Park et al., 2019) and DeepMesh (Guillard et al., 2021) to name a few. This training technique is a form of explicit learning because the model explicitly learns a function that maps an input point to its corresponding SDF value, as described in Section 3.1. However, this approach has limitations when it comes to capturing local geometric features of the surface, such as sharp edges, corners, and ridges. This is because the SDF value alone does not contain sufficient information about the local geometry of the surface. Therefore, explicit learning can only promote SDF regression in the vicinity of the surface, rather than directly on it.

Further, Attar et al. (2021b) and Attar et al. (2021a) have demonstrated that local radii features play a crucial role in determining the manufacturability of sheet stamping. Optimising these features to meet manufacturing constraints (Attar et al., 2021a; Horton et al., 2020) is essential and requires high-quality representations. As the Explicit Nets were unable to reconstruct these critical features, this approach was found unsuitable for representing sheet stamping geometries for geometry optimisation. In contrast, the Implicit Nets provided high-quality reconstructions of local features and are recommended for representing sheet stamping geometries for geometry optimisation, especially in situations where small-scale features are important.

6. Continuity of the learnt geometric latent space

The latent space refers to the learnt low dimensional space that contained the inferred latent vectors $\boldsymbol{z} \in \mathbb{R}^{128}$ for the considered stamping geometries. The term *low dimensionality* here is used in relation to high dimensional data representations, such as images or meshes. In



(a) One sided Chamfer Distance from reconstruction to ground truth (b) Similarity of normals between reconstruction and ground truth

Fig. 18. Testing dataset surface coverage versus performance metrics for various Auto-Decoder networks: Explicit Nets 1–2 and Implicit Net (top row), and Explicit Nets 3–4 and Implicit Net (bottom row). Solid curves represent means and shaded bands represent standard deviations across all samples in the testing dataset.

the case of an image, the dimensionality is equal to the total number of pixels (e.g., $256 \times 256 = 65{,}536$ pixels). In the case of a mesh, the dimensionality is equal to the total degrees of freedom of mesh vertices (e.g., 10,000 vertices $\times 3$ orthogonal directions $= 30{,}000$ degrees of freedom).

In this section, the continuity of the latent space learnt by Implicit Net is presented. As mentioned by Wang et al. (2020a), when continuity is combined with low dimensionality, different vectorised directions in the latent space encode geometrically meaningful embeddings. These embeddings could be leveraged for explorative geometric optimisations.

6.1. Organisation of the latent space

The learnt latent space was found to be organised according to the natural similarity between different geometries. To visualise this organised space, three-dimensional Principal Component Analysis (PCA) was performed on latent vectors inferred from geometries in the training and testing datasets. Following PCA, the principal components of the latent vectors were obtained, and this reduced the space from $\mathbb{R}^{128}$ to $\mathbb{R}^3$. These principal components are plotted in Fig. 21, where it was found that geometries from the same subclass cluster together in the latent space. This clustering can be interpreted as the latent space being organised in terms of global geometric features. A mild overlap was found between red and green points, which correspond to geometries that were similar to both standard and chamfer corners (e.g., chamfer corners with small chamfer lengths). In contrast, the blue points, which corresponded to stepped sidewalls, were further away because their stepped feature made them dissimilar to the other geometries.

To further show the organisation of the latent space, the three coloured clusters in Fig. 21 are plotted separately in Fig. 22. In each plot, selected local geometric features (i.e., height, radius, or similar) defined in Fig. 5(a) are superimposed as colours. The smooth transitions between different colours show that the latent space was further organised in terms of these local geometric features in addition to global features.

In summary, it was found that the latent space encoded meaningful and well organised geometric information on a global and local scale. Consequently, the proximity of two points in the latent space provided a measure of similarity between two geometries represented by these points. This organisation suggests that smooth geometric changes are possible by travelling along vectorised directions in the latent space. These smooth changes are demonstrated by latent space interpolation in the following subsection.

6.2. Interpolation in the latent space

To evaluate the learnt latent space further, geometries generated by interpolating between latent vectors are presented in Fig. 23. In Figure (a) the geometries shown in blue were generated from latent vectors that were averaged from those of the adjacent grey test set geometries. It is evident that these generated geometries naturally combine common geometric features of their adjacent test set constituents. In Figure (b), the geometries shown in blue were generated from latent vectors that were linear combinations of those of the two grey test set geometries, $\boldsymbol{z}_A$ and $\boldsymbol{z}_B$, according to Eq. (18).

$$\boldsymbol{z}_\alpha = (1-\alpha)\,\boldsymbol{z}_A + \alpha \boldsymbol{z}_B \tag{18}$$

Here, $\boldsymbol{z}_\alpha$ is the latent vector at the value of the scalar $\alpha \in [0, 1]$. The smooth changes between the geometries in Figure (b) demonstrate that robust free morphing of sheet stamping geometries was possible by exploring the learnt latent space. As a separate note, notice the well captured sharp fillet radii in all geometries shown.

7. Limitations and potential solutions

The proposed model for generating sheet stamping geometries with small-scale features has shown promise, but it is important to acknowledge potential limitations and ways to address them. One major limitation is the lack of generalisability to unseen geometry classes due to the model being trained on a specific class of stamping geometries. To increase the practicality of the proposed approach for manufacturing

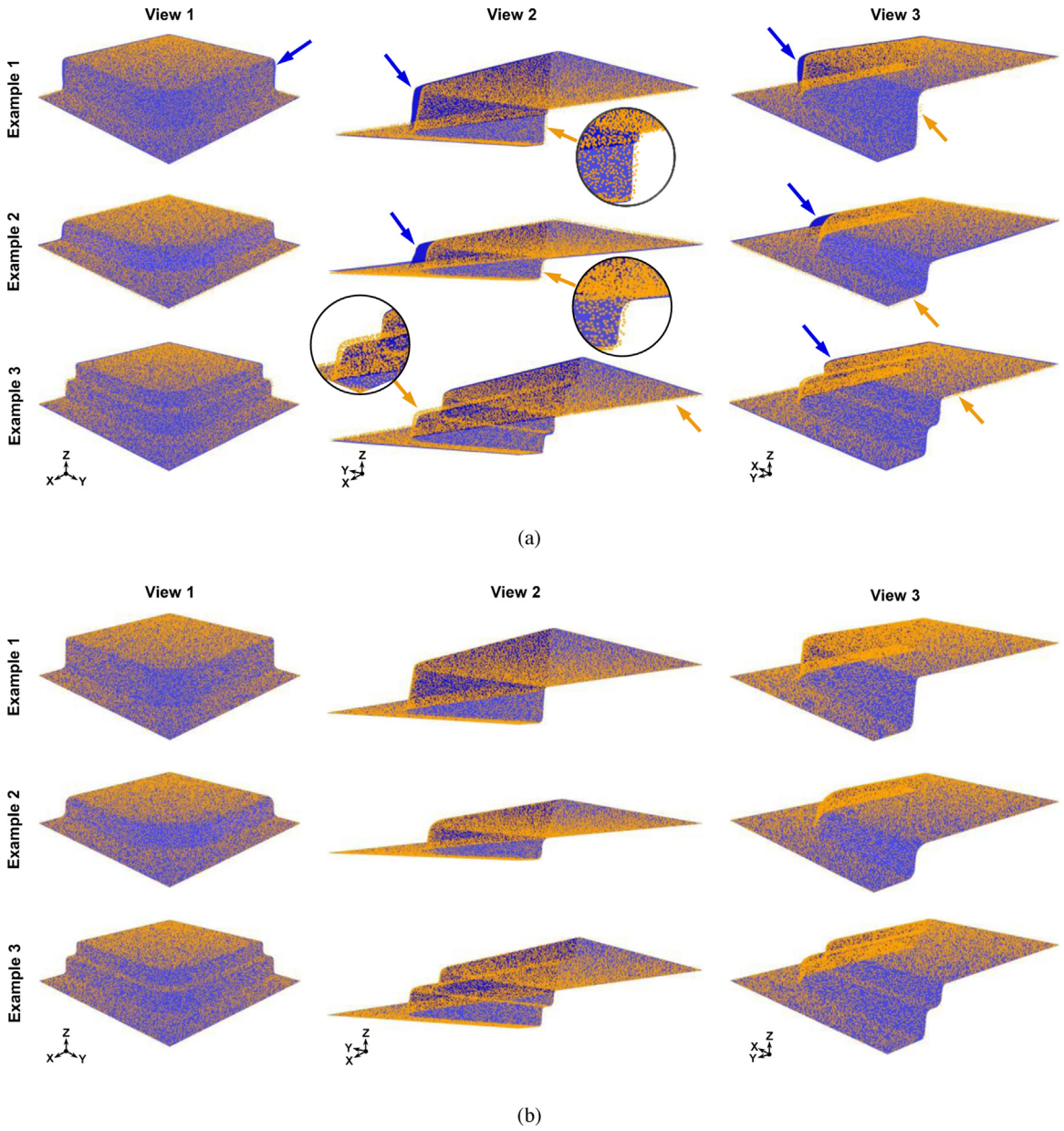


Fig. 19. Comparison between reconstructed surface meshes $\mathcal{M}$ (purple surfaces) and ground truth surface points $\boldsymbol{P}_{GT}$ (orange points) for three representative geometries from the testing dataset. Reconstructions generated using (a) Explicit Net 1 and (b) Implicit Net. Arrows in (a) indicate zones of misalignment between $\mathcal{M}$ and $\boldsymbol{P}_{GT}$. All images taken with the perspective camera projection type for best figure clarity.

applications, researchers and engineers can wisely utilise the approach by training separate geometry models on different broad geometry classes, such as boxes, door panels, pillars, and beams. Specifically, for sheet stamping applications, there are only a limited number of common geometry classes typically used in the industry, making it feasible to train separate models for each class. By using this strategy, a single model does not need to learn a latent space shared across vastly different geometry classes, e.g., between boxes and beams. Consequently, the requirement for a universally generalisable model can be removed, leading to generating higher quality stamping geometries of a specific class, thereby increasing its practicality for manufacturing applications. The proposed approach also requires significant computational resources, which can limit accessibility for some researchers and engineers. This limitation can be mitigated by investing in GPU-enabled workstations or utilising online cloud-based resources. Additionally, the method generates geometries without considering manufacturing constraints, which limits their practicality. This limitation can be addressed by incorporating manufacturing-specific constraints, such as stamping process feasibility to guide the geometry generation. The integration of manufacturability criteria will be explored in future work by using recently developed manufacturability surrogate models (Attar et al., 2021b) to generate manufacturable geometries, as detailed in Section 8.

8. Conclusion and broader impact

A novel deep learning method has been introduced for creating high-quality implicit neural representations of 3D sheet stamping geometries using Signed Distance Fields (SDFs). The research conclusions can be summarised as follows:

- An innovative implicit learning approach to generate SDFs of sheet stamping geometries was proposed, based on supervising the geometric properties of SDFs. Additionally, the state-of-the-art DeepSDF (Park et al., 2019) approach in literature, which involves explicit learning and supervision of SDF values, was

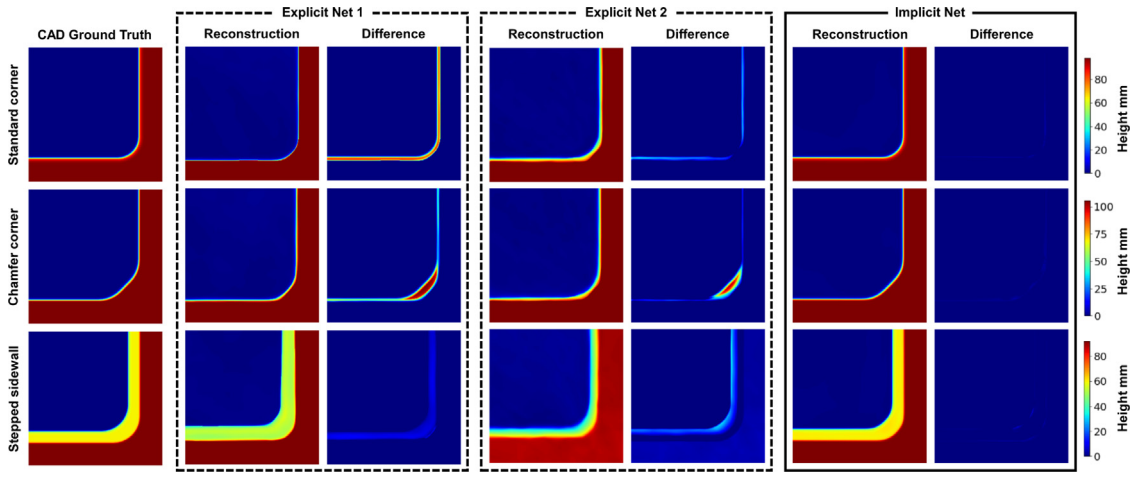


Fig. 20. Comparison between 2D projected height maps of reconstructed surfaces from Explicit Nets 1–2 (dashed line rectangles) and Implicit Net (solid line rectangle) with CAD ground truths for three representative geometries from the testing dataset. Colourbar limits set equal across columns for comparison between reconstruction methods.

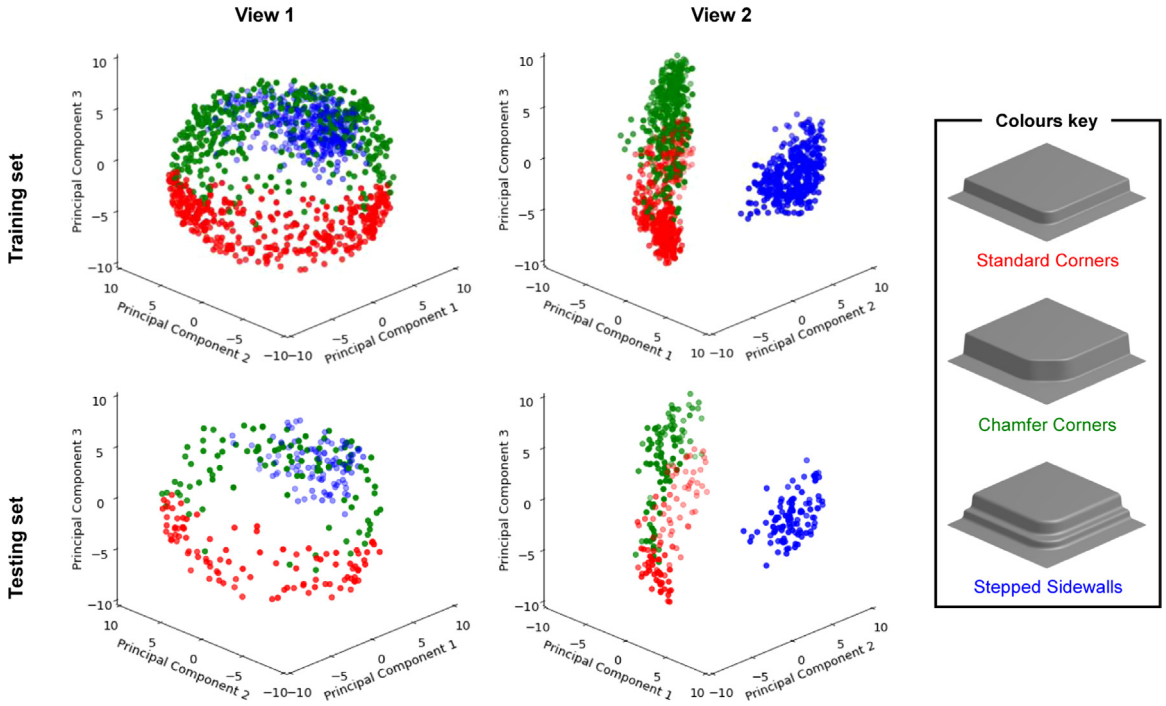


Fig. 21. Three dimensional PCA of the learnt latent space. Coloured clusters denote points belonging to different geometry subclasses, and therefore global geometric features, and are labelled in the colours key. Views selected for best figure clarity.

investigated for its ability to accurately reconstruct small-scale geometric features.

- Using the two approaches, neural network models were trained as compact, differentiable functions capable of generating a range of sheet stamping geometries, which were captured in the trained model parameters.
- Comprehensive evaluations were conducted on models trained using both approaches, including quantitative and qualitative analyses. Results showed that the implicit approach outperformed the explicit approach in terms of geometric accuracy, as demonstrated by the Chamfer distance, Hausdorff distance, and mesh similarity metrics. The explicit approach failed to capture small-scale features critical to the manufacturing performance of sheet stamping geometries, while the implicit approach performed well in this regard. Therefore, adopting the implicit approach is recommended for generating stamping geometries.
- The continuity of the geometric latent space learnt from the implicit approach was analysed, revealing that similar geometries formed distinct clusters within the latent space representing global features, such as stepped sidewalls and chamfer corners. It was found that each cluster was further organised based on local geometric features, such as height or radius, indicating the possibility of smooth and detailed geometric changes by exploring this space. This organised and meaningful latent space was demonstrated through latent space interpolation, which resulted in smooth transitions between geometry subclasses.

Future work will leverage the continuity of the latent space, as well as the differentiable nature of the trained networks, to perform optimisation of 3D stamping geometries that is agnostic to CAD parameterisation scheme. Integrating the proposed model as a geometry generator with the previously developed manufacturability surrogate

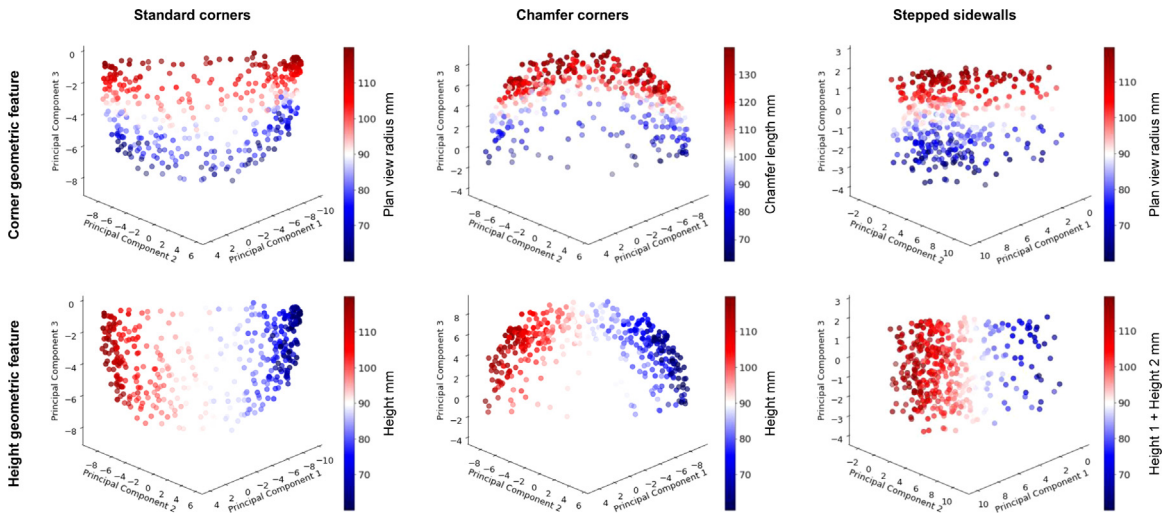


Fig. 22. Three dimensional PCA of the learnt latent space where each cluster in Fig. 21 is plotted here individually. The colour bars represent values of a local geometric feature. The geometric features follow those defined in Fig. 5(a). Views selected for best figure clarity.

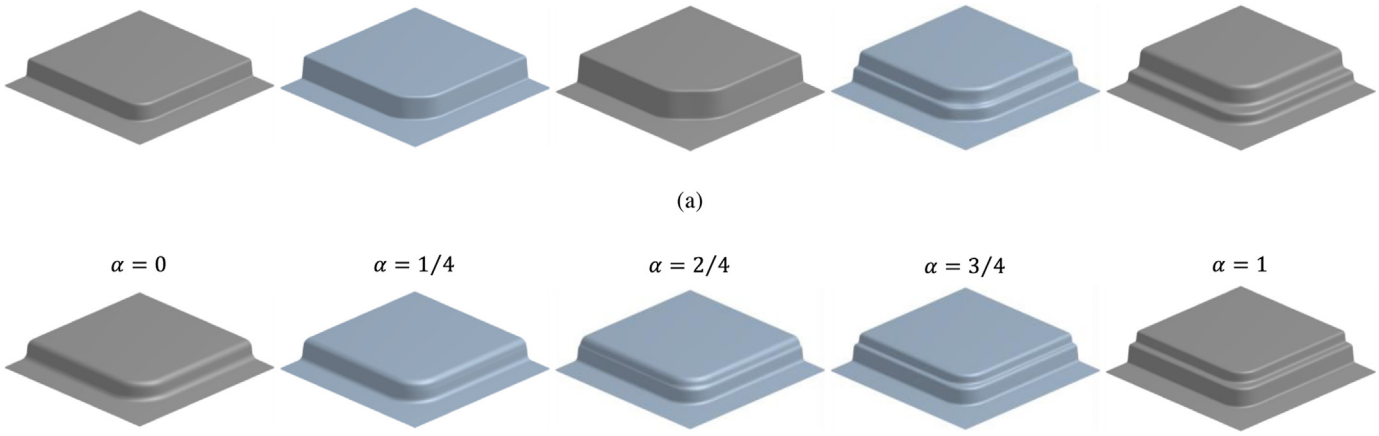


Fig. 23. Reconstructions of test set samples (in grey) and generated geometries (in blue). In (a), the generated geometries were decoded from latent vectors that were averaged from those of adjacent grey geometries. In (b), the generated geometries were decoded from latent vectors that were obtained by linearly interpolating between those of the grey geometries.

models (Attar et al., 2021b) could enable the creation of a closed-loop optimisation platform for sheet stamping processes. The generator model can be used to create virtual geometries, which can be evaluated for their manufacturing performance using the surrogate models. The surrogate models provide predictions of the performance of the virtual geometries, which can be used to iteratively update the inputs into the generator model to optimise the geometry for better manufacturability. The integration of the generator model and the surrogate models in a closed loop can create a digital twin for real-time virtual optimisation of geometries for sheet stamping processes. The use of such a digital twin can eliminate the need for costly FE simulation iterations or physical stamping trials, making the optimisation process faster and more efficient.

Furthermore, the implicit neural representations and methodologies developed in this study can be extended to other application scenarios in the manufacturing sector. For example, the proposed approach could be used to represent the design of arbitrary blank shapes for optimisation, which is critical in many sheet forming processes. Additionally, the model could be adapted to combine material properties and process parameters as system inputs for holistic process optimisation. Finally, the approach could be applied to other sheet forming processes (Zheng et al., 2018), including various fixed die forming (e.g., deep drawing, hydroforming, superplastic forming) and incremental sheet forming with flexible dies where 3D shape optimisation may be applicable in real-time.

In addition to manufacturing scenarios, the methodology proposed here can be applied to any scenarios requiring design of 3D surfaces with critical small-scale features. For example, the framework can be used in concurrently optimising both global design and small-scale local features that are critical to certain performance indicators, such as local stiffeners in safety-critical components (e.g., car B-Pillars) or small-scale surface design features affecting the aerodynamic performance of car exteriors, aircraft panels, or boat hull surfaces.

CRediT authorship contribution statement

Hamid Reza Attar: Conceptualization, Methodology, Data curation, Formal analysis, Writing – original draft, Visualisation. **Alistair Foster:** Supervision, Writing – review & editing. **Nan Li:** Supervision, Conceptualization, Formal analysis, Writing – review & editing, Funding acquisition.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Hamid Reza Attar reports financial support was provided by Engineering and Physical Sciences Research Council.

Data availability

The data that supports this study is available from the corresponding author upon reasonable request.

Acknowledgements

The authors thank Impression Technologies Ltd for funding support, the UK EPSRC for the CASE conversion DTP training grant (EP/R513052/1), and UKRI for the Impact Acceleration Accounts grant. Software from ESI Group is also gratefully acknowledged. HFQ® is a registered trademark of Impression Technologies Ltd. For the purpose of open access, the author has applied a Creative Commons Attribution (CC BY) license to any Author Accepted Manuscript version arising.

Appendix. Brief explanation of marching cubes

Combining the benefits of SDFs with explicit representations requires the use of Marching Cubes (Lorensen and Cline, 1987) as mentioned in Section 2.5. Marching Cubes is an algorithm for creating a triangle mesh based representation of a level-set from an implicit field. Since the zero-level-set of SDFs here are designed to implicitly represent the surface geometry, running Marching Cubes on the SDF would extract the implicit surface by converting it to an explicit triangular mesh. This application of Marching Cubes works by iterating over a uniform 3D grid of cubes imposed over a region of the SDF and searching for its zero-level-set. For each cube in the 3D grid, the algorithm evaluates the sign of the SDF points at the cube vertices. If all 8 cube vertices of the cube have the same sign, then the cube is entirely above or below the zero-level-set and no triangles are created. Otherwise, triangles are created which make up mesh faces. There are 15 unique cases of triangle and mesh vertex combinations and these are determined by a lookup table shown in Fig. A.1, where a search is performed using the SDF sign at each of the 8 cube vertices. The final extracted mesh is a combination of the triangles and mesh vertices from each cube in the 3D grid. For more details on Marching Cubes, the reader is referred to the original paper (Lorensen and Cline, 1987).

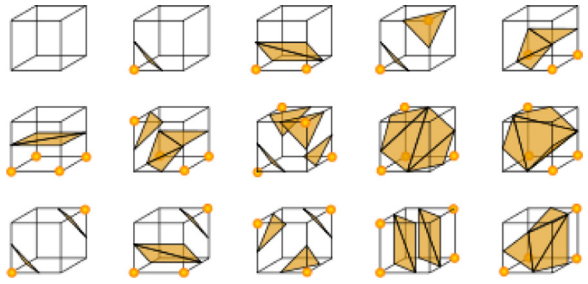


Fig. A.1. A graphical representation of the triangulated cubes lookup table used by the Marching Cubes algorithm. Yellow highlighted cube vertices have an opposite sign to non-highlighted cube vertices. There are 15 unique combinations in total.
Source: Figure adapted from Liao et al. (2018).

References

Arjovsky, M., Chintala, S., Bottou, L., 2017. Wasserstein GaN. ArXiv.
Attar, H.R., Li, N., Foster, A., 2021a. A new design guideline development strategy for aluminium alloy corners formed through cold and hot stamping processes. Mater. Des. 207, http://dx.doi.org/10.1016/j.matdes.2021.109856.
Attar, H.R., Zhou, H., Foster, A., Li, N., 2021b. Rapid feasibility assessment of components to be formed through hot stamping : A deep learning approach. J. Manuf. Process. 68, 1650–1671. http://dx.doi.org/10.1016/j.jmapro.2021.06.011.
Attar, H.R., Zhou, H., Li, N., 2021c. Deformation and thinning field prediction for HFQ® formed panel components using convolutional neural networks. IOP Conf. Ser. Mater. Sci. Eng. 1157, http://dx.doi.org/10.1088/1757-899X/1157/1/012079.
Atzmon, M., Lipman, Y., 2020. SAL: Sign agnostic learning of shapes from raw data. In: Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.. pp. 2562–2571. http://dx.doi.org/10.1109/CVPR42600.2020.00264.
Baque, P., Remelli, E., Fleuret, F., Fua, P., 2018. Geodesic convolutional shape optimization. In: 35th Int. Conf. Mach. Learn. ICML 2018. Vol. 2. pp. 797–809.
Bonte, M.H., van den Boogaard, A., Huétink, J., 2007. A metamodel based optimisation algorithm for metal forming processes. In: Adv. Methods Mater. Form.. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 55–72. http://dx.doi.org/10.1007/3-540-69845-0_4.
Chabra, R., Lenssen, J.E., Ilg, E., Schmidt, T., Straub, J., Lovegrove, S., Newcombe, R., 2020. Deep local shapes: Learning local SDF priors for detailed 3D reconstruction. In: Eur. Conf. Comput. Vis.. pp. 608–625. http://dx.doi.org/10.1007/978-3-030-58526-6_36.
Dawson-Haggerty, 2021. Trimesh. https://trimsh.org/.
Druc, S., Balu, A., Wooldridge, P., Krishnamurthy, A., Sarkar, S., 2022. Concept activation vectors for generating user-defined 3D shapes. ArXiv. 2993–3000. http://dx.doi.org/10.48550/arXiv.2205.02102.
El Fakir, O., Wang, L., Balint, D., Dear, J.P., Lin, J., Dean, T.A., 2014. Numerical study of the solution heat treatment, forming, and in-die quenching (HFQ) process on AA5754. Int. J. Mach. Tools Manuf. 87, 39–48. http://dx.doi.org/10.1016/j.ijmachtools.2014.07.008.
Gayon-Lombardo, A., Mosser, L., Brandon, N.P., Cooper, S.J., 2020. Pores for thought: The use of generative adversarial networks for the stochastic reconstruction of 3D multi-phase electrode microstructures with periodic boundaries. NPJ Comput. Mater. http://arxiv.org/abs/2003.11632.
Goodfellow, I.J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y., 2014. Generative adversarial nets. In: Adv. Neural Inf. Process. Syst., Vol. 3. pp. 2672–2680.
Gropp, A., Yariv, L., Haim, N., Atzmon, M., Lipman, Y., 2020. Implicit geometric regularization for learning shapes. In: 37th Int. Conf. Mach. Learn. ICML. https://dl.acm.org/doi/abs/10.5555/3524938.3525293.
Guillard, B., Remelli, E., Lukoianov, A., Richter, S., Bagautdinov, T., Baque, P., Fua, P., 2021. DeepMesh: Differentiable iso-surface extraction. ArXiv. http://arxiv.org/abs/2106.11795.
Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., Courville, A., 2017. Improved training of wasserstein GANs. ArXiv.
Gupta, K., Chandraker, M., 2020. Neural mesh flow: 3D manifold mesh generation via diffeomorphic flows. ArXiv.
Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M., Mohamed, S., Lerchner, A., 2017. B-VAE: Learning basic visual concepts with a constrained variational framework. In: 5th Int. Conf. Learn. Represent. ICLR 2017 - Conf. Track Proc.. pp. 1–22.
Horton, P.M., Allwood, J.M., Cleaver, C., Nagy-Sochacki, A., 2020. An experimental analysis of the relationship between the corner, die and punch radii in forming isolated flanged shrink corners from Al 5251. J. Mater Process. Technol. 278, http://dx.doi.org/10.1016/j.jmatprotec.2019.116486.
Hu, W., Fan, Y., Lei, C., Enying, L., 2017. Sheet metal forming optimization by using surrogate modeling techniques. Chin. J. Mech. Eng. (Engl. Ed.) 30, 22–36. http://dx.doi.org/10.3901/CJME.2016.1020.123.
Jiang, C., Huang, J., Tagliasacchi, A., Guibas, L., 2020. ShapeFlow: Learnable deformations among 3D shapes. In: Adv. Neural Inf. Process. Syst..
Kingma, D.P., Ba, J.L., 2015. Adam: A method for stochastic optimization. ArXiv. arXiv:1412.6980v9.
Kohar, C.P., Greve, L., Eller, T.K., Connolly, D.S., Inal, K., 2021. A machine learning framework for accelerating the design process using CAE simulations: An application to finite element analysis in structural crashworthiness. Comput. Methods Appl. Mech. Engrg. 385, 114008. http://dx.doi.org/10.1016/j.cma.2021.114008.
Li, J., Ning, T., Xi, P., Hu, B., Wang, T., Yang, J., 2020. Smoothing parametric design of addendum surfaces for sheet metal forming. Chin. J. Mech. Eng. (Engl. Ed.) 33, http://dx.doi.org/10.1186/s10033-019-0425-8.
Liao, Y., Donne, S., Geiger, A., 2018. Deep marching cubes: Learning explicit surface representations. In: Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.. pp. 2916–2925. http://dx.doi.org/10.1109/CVPR.2018.00308.
Lin, J., Dean, A., Trevor, Garrett, P., Richard, Foster, D., Alistair, 2008. Process for forming aluminium alloy sheet component. WO2008059242A2.
Liu, S., Zhang, Y., Peng, S., Shi, B., Pollefeys, M., Cui, Z., 2020. DIST: Rendering deep implicit signed distance function with differentiable sphere tracing. In: Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.. pp. 2016–2025. http://dx.doi.org/10.1109/CVPR42600.2020.00209.
Lorensen, W.E., Cline, H.E., 1987. Marching cubes: A high resolution 3D surface construction algorithm. In: Proc. 14th Annu. Conf. Comput. Graph. Interact. Tech. SIGGRAPH 1987. Vol. 21. pp. 163–169. http://dx.doi.org/10.1145/37401.37422.
MathWorks, 2021. lhsdesign. https://uk.mathworks.com/help/stats/lhsdesign.html.
Mescheder, L., Oechsle, M., Niemeyer, M., Nowozin, S., Geiger, A., 2019. Occupancy networks: Learning 3D reconstruction in function space. In: Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.. http://dx.doi.org/10.1109/CVPR.2019.00459.
Mirza, M., Osindero, S., 2014. Conditional generative adversarial nets. http://arxiv.org/abs/1411.1784.
Mohamed, M.S., Foster, A.D., Lin, J., Balint, D.S., Dean, T.A., 2012. Investigation of deformation and failure features in hot stamping of AA6082: Experimentation and modelling. Int. J. Mach. Tools Manuf. 53, 27–38. http://dx.doi.org/10.1016/j.ijmachtools.2011.07.005.

Mohamed, M., Li, N., Wang, L., El Fakir, O., Lin, J., Dean, T., Dear, J., 2015. An investigation of a new 2D CDM model in predicting failure in HFQing of an automotive panel. In: MATEC Web Conf.. http://dx.doi.org/10.1051/matecconf/20152105011.

Naceur, H., Ben-Elechi, S., Batoz, J.L., Knopf-Lenoir, C., 2008. Response surface methodology for the rapid design of aluminum sheet metal forming parameters. Mater. Des. 29, 781–790. http://dx.doi.org/10.1016/j.matdes.2007.01.018.

Osher, S., Paragios, N., 1999. Geometric Level Set Methods in Imaging, Vision, and Graphics. Springer, http://dx.doi.org/10.1007/b97541.

Park, J.J., Florence, P., Straub, J., Newcombe, R., Lovegrove, S., 2019. DeepSDF: Learning continuous signed distance functions for shape representation. http://dx.doi.org/10.48550/arXiv.1901.05103, ArXiv.

Peng, S., Max Jiang, C., Liao, Y., Niemeyer, M., Pollefeys, M., Geiger, A., 2021. Shape as points: A differentiable Poisson solver. ArXiv. http://arxiv.org/abs/2106.03452.

Pfaff, T., Fortunato, M., Sanchez-Gonzalez, A., Battaglia, P.W., 2021. Learning mesh-based simulation with graph networks. ArXiv. http://arxiv.org/abs/2010.03409.

Politis, D.J., Li, N., Wang, L., Lin, J., Foster, A.D., Szegda, D., 2016. Prediction of thinning behavior for complex-shaped, lightweight alloy panels formed through a hot stamping process. pp. 395–401. http://dx.doi.org/10.1142/9789813140622_0065.

Pytorch, 2021. Automatic differentiation package: Autograd. https://pytorch.org/docs/stable/autograd.

Radford, A., Metz, L., Chintala, S., 2016. Unsupervised representation learning with deep convolutional generative adversarial networks. In: 4th Int. Conf. Learn. Represent. ICLR 2016 - Conf. Track Proc.. pp. 1–16.

Ramnath, S., Haghighi, P., Kim, J.H., Detwiler, D., Berry, M., Shah, J.J., Aulig, N., Wollstadt, P., Menzel, S., 2019. Automatically generating 60, 000 CAD variants for big data applications. In: 39th Comput. Inf. Eng. Conf. American Society of Mechanical Engineers. http://dx.doi.org/10.1115/DETC2019-97378.

Ramnath, S., Haghighi, P., Ma, J., Shah, J.J., Detwiler, D., 2020. Design science meets data science: Curating large design datasets for engineered artifacts. In: 40th Comput. Inf. Eng. Conf. American Society of Mechanical Engineers. http://dx.doi.org/10.1115/DETC2020-22377.

Remelli, E., Lukoianov, A., Richter, S.R., Guillard, B., Bagautdinov, T., Baque, P., Fua, P., 2020. MeshSDF: Differentiable iso-surface extraction. ArXiv. http://arxiv.org/abs/2006.03997.

Sitzmann, V., Martel, J.N.P., Bergman, A.W., Lindell, D.B., Wetzstein, G., 2020. Implicit neural representations with periodic activation functions. ArXiv. http://arxiv.org/abs/2006.09661.

Sorkine, O., Cohen-Or, D., Toldeo, S., 2003. High-pass quantization for mesh encoding. In: Eurographics Symp. Geom. Process.. pp. 42–51. http://dx.doi.org/10.2312/SGP/SGP03/042-051.

Wang, L., Chan, Y.C., Ahmed, F., Liu, Z., Zhu, P., Chen, W., 2020a. Deep generative modeling for mechanistic-based learning and design of metamaterial systems. Comput. Methods Appl. Mech. Engrg. 372, http://dx.doi.org/10.1016/j.cma.2020.113377.

Wang, J., Zeng, Y., Jiang, X., Wang, H., Li, E., Li, G., 2020b. Variational auto-encoder based approximate bayesian computation uncertian inverse method for sheet metal forming problem.

Wang, N., Zhang, Y., Li, Z., Fu, Y., Liu, W., Jiang, Y.G., 2018. Pixel2Mesh: Generating 3D mesh models from single RGB images. In: Proc. Eur. Conf. Comput. Vis.. http://dx.doi.org/10.1007/978-3-030-01252-6_4.

Wu, J., Zhang, C., Xue, T., Freeman, W.T., Tenenbaum, J.B., 2016. Learning a probabilistic latent space of object shapes via 3D generative-adversarial modeling. In: Adv. Neural Inf. Process. Syst.. pp. 82–90.

Xiao, W., Wang, B., Zhou, J., Ma, W., Yang, L., 2016. Optimization of aluminium sheet hot stamping process using a multi-objective stochastic approach. Eng. Optim. 48, 2173–2189. http://dx.doi.org/10.1080/0305215X.2016.1163483.

Yang, M., Wen, Y., Chen, W., Chen, Y., Jia, K., 2021. Deep optimized priors for 3D shape modeling and reconstruction. In: Cvpr. pp. 3269–3278, http://arxiv.org/abs/2012.07241.

Zheng, K., Politis, D.J., Wang, L., Lin, J., 2018. A review on forming techniques for manufacturing lightweight complex—shaped aluminium panel components. Int. J. Lightweight Mater. Manuf. 1, 55–80. http://dx.doi.org/10.1016/j.ijlmm.2018.03.006.

Zhou, J., Wang, B., Lin, J., Fu, L., 2013. Optimization of an aluminum alloy anti-collision side beam hot stamping process using a multi-objective genetic algorithm. Arch. Civ. Mech. Eng. 13, 401–411. http://dx.doi.org/10.1016/j.acme.2013.01.008.

Zhou, J., Wang, B.Y., Lin, J.G., Fu, L., Ma, W.Y., 2014. Forming defects in aluminum alloy hot stamping of side-door impact beam. Trans. Nonferrous Met. Soc. China (Engl. Ed.) 24, 3611–3620. http://dx.doi.org/10.1016/S1003-6326(14)63506-8.

Zhou, H., Xu, Q., Nie, Z., Li, N., 2022. A study on using image-based machine learning methods to develop surrogate models of stamp forming simulations. J. Manuf. Sci. Eng. 144, 1–15. http://dx.doi.org/10.1115/1.4051604.

Zhu, M., Lim, Y.C., Liu, X., Cai, Z., Dhawan, S., Gao, H., Politis, D.J., 2021. Numerical forming limit prediction for the optimisation of initial blank shape in hot stamping of AA7075. Int. J. Lightweight Mater. Manuf. 4, http://dx.doi.org/10.1016/j.ijlmm.2020.12.006.

Zimmerling, C., Poppe, C., Kärger, L., 2020. Estimating optimum process parameters in textile draping of variable part geometries - A reinforcement learning approach. Procedia Manuf. 47, 847–854. http://dx.doi.org/10.1016/j.promfg.2020.04.263.

Zimmerling, C., Trippe, D., Fengler, B., Kärger, L., 2019. An approach for rapid prediction of textile draping results for variable composite component geometries using deep neural networks. In: AIP Conf. Proc., Vol. 2113. http://dx.doi.org/10.1063/1.5112512.