Computers & Graphics 107 (2022) 73–83

Contents lists available at ScienceDirect

Computers & Graphics

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

Special Section on 3DOR 2022

SketchCleanNet — A deep learning approach to the enhancement and correction of query sketches for a 3D CAD model retrieval system

Bharadwaj Manda [a,*], Prasad Pralhad Kendre [a], Subhrajit Dey [b], Ramanathan Muthuganapathy [a]

[a] Advanced Geometric Computing Lab, Department of Engineering Design, Indian Institute of Technology Madras, Chennai - 600036, India
[b] Department of Electrical Engineering, Jadavpur University, Jadavpur, Kolkata - 700032, India

A R T I C L E   I N F O

Article history:
Received 29 March 2022
Received in revised form 2 July 2022
Accepted 4 July 2022
Available online 8 July 2022

Keywords:
Search and retrieval
Engineering models
Sketch enhancement
Mechanical components
CAD models
Sketch-based retrieval

A B S T R A C T

Search and retrieval remains a major research topic in several domains, including computer graphics, computer vision, engineering design, etc. A search engine requires primarily an input search query and a database of items to search from. In engineering, which is the primary context of this paper, the database consists of 3D CAD models, such as washers, pistons, connecting rods, etc. A query from a user is typically in the form of a sketch, which attempts to capture the details of a 3D model. However, sketches have certain typical defects such as gaps, over-drawn portions (multi-strokes), etc. Since the retrieved results are only as good as the input query, sketches need cleaning-up and enhancement for better retrieval results.

In this paper, a deep learning approach is proposed to improve or clean the query sketches. Initially, sketches from various categories are analysed in order to understand the many possible defects that may occur. A dataset of cleaned-up or enhanced query sketches is then created based on an understanding of these defects. Consequently, an end-to-end training of a deep neural network is carried out in order to provide a mapping between the defective and the clean sketches. This network takes the defective query sketch as the input and generates a clean or an enhanced query sketch. Qualitative and quantitative comparisons of the proposed approach with other state-of-the-art techniques show that the proposed approach is effective. The results of the search engine are reported using both the defective and enhanced query sketches, and it is shown that using the enhanced query sketches from the developed approach yields improved search results.

1. Introduction

The use of search engines has become commonplace in daily life. The primary driver for developing efficient search engines is the explosion of publicly available data, combined with the user's need to find relevant information from a large collection of data items [1]. This has led to a rapid growth in the development and usage of search engines across a multitude of application domains, including library management systems, demographic data, internet-based information retrieval, enterprise search, and so on. The user first characterizes his/her information need in the form of an input query and expects to find information that is most relevant to the query [2]. Traditionally, input queries to the search engine have been in the form of text (string of characters). This has been an effective way for users to provide their queries, since most retrieval systems focused on text data such as documents, personnel records, news articles, etc.

However, the process of search and retrieval becomes challenging in the domain of visual data, especially 3D models, because it is difficult to characterize what a human being sees and perceives in the form of a text query [3]. As a result, content-based retrieval systems have been developed, which make use of the visual/shape properties of the 3D models as opposed to the traditional text query [4,5]. Among the available query options, a sketch-based query is the most efficient since it is natural for the user and easy to learn [6]. The usage of a sketch-based query is also shown to be very intuitive and convenient for the user than describing the 3D object by a set of descriptor rules or using the 3D model itself [7,8].

The increased availability of large-scale datasets, paired with greater computational capabilities, has accelerated the deployment of deep learning solutions to a wide range of research problems, including three-dimensional shape search (3DSS), which is a key problem in the engineering design process. The Princeton 3D model search engine [9], PROBADO3D search engine [10] and the

* Corresponding author.
E-mail addresses: mvlkbcse@gmail.com (B. Manda), pkendre619@gmail.com (P.P. Kendre), subhrajitdey.agt@gmail.com (S. Dey), mraman@iitm.ac.in (R. Muthuganapathy).

https://doi.org/10.1016/j.cag.2022.07.006
0097-8493/© 2022 Elsevier Ltd. All rights reserved.

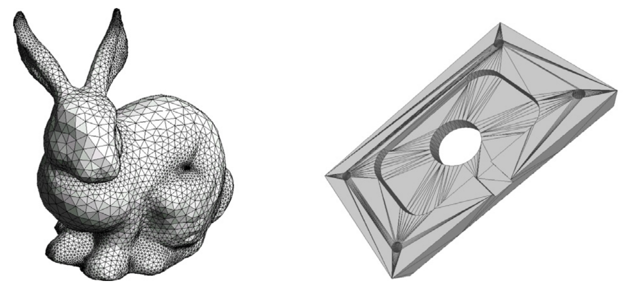

(a) Generic 3D Shapes or Graphical Models (b) 3D Engineering shapes or CAD models

Fig. 1. Distinction between a Graphical Model and a CAD Model.

CAD model search engine proposed by [11] are all well-known examples of 3DSS. [12] contributed the earliest benchmark dataset of sketches corresponding to each 3D model in the Princeton Shape Benchmark (PSB) [13]. Consequently, a few instances of the Shape Retrieval Contest (SHREC) such as [14] and [15] established large-scale benchmarks for sketch data for 3D shapes, in addition to many approaches for sketch-based retrieval. These datasets and approaches, however, are solely aimed at generic 3D shape data or graphical representations and do not include engineering CAD model data.

Collecting the right information for product design is a time-consuming process. The product development process in the modern day is fast-paced, in order to keep up with rising customer demands. As a result, the development of new product designs is increasingly reliant on reference products or existing designs [16–18]. Consequently, efficient search and retrieval of relevant product designs becomes crucial for tasks such as automated feature recognition of CAD models [19], extracting components for CAD assembly [20], computing geometric similarity of CAD parts [21] and so forth.

In product life cycle (PLC) analysis and management, the entire cycle starts with design. A user's intent of the design is best captured using a sketch of the model than a user drawing a 3D model. It is also perhaps easier to hand sketch the design intent than to use a 2D drawing tool. Once a sketch is drawn, its corresponding 3D model may be retrieved using a search and retrieval of CAD models. However, it is always not an easy task to provide a perfect sketch and often it needs cleanup to be used in further downstream applications.

Despite the technological advances, the progress in the field of developing efficient search and retrieval systems for 3D CAD models has been slow. This is largely attributed to the inherent challenges associated with CAD models of engineering parts and shapes, as well as the proprietary nature of such CAD datasets [22]. An engineering components database typically includes 3D CAD models such as washers, pistons, connecting rods, gear parts, and so forth. Figs. 1(a) and 1(b) illustrate a generic 3D shape and an engineering CAD model, respectively. Engineering CAD models are distinct from regular 3D shapes in the following ways [23]:

- CAD model tessellations are typically sparse
- CAD models contain sharp changes in curvature while generic 3D shapes are usually smooth
- Machining features such as holes, pockets and slots are present in 3D engineering shapes

In the case of engineering models, the individuals involved must have substantial domain experience and skill. Hence, the available sketch datasets for engineering CAD models are very few. The CADSketchNet dataset [24] introduced a benchmark sketch dataset for engineering components. Query sketches for each CAD model from the Engineering Shape Benchmark (ESB) [23] as well as the Mechanical Components Benchmark (MCB) [25] have been provided. However, sketches have certain typical defects such as missing lines, over-drawn portions (multi-strokes), etc. Such sketches are referred to as rough sketches or defect sketches in this paper (see Fig. 2 for examples). Since the retrieved results of a search engine are only as good as the input query, these sketches need cleaning-up and enhancement for better retrieval results. Henceforth, in this paper, "clean sketches" refers to those sketches in which such defects are either absent or greatly reduced. It also refers to the increased visual appeal of the images as compared to the "defect sketches". It may be noted that the sketches provided in CADSketchNet [24] contain certain defects and do not provide any clean sketches as ground truth. Hence, there is a need to first create a database of training examples that can be used to train a deep learning model. It was also established in CADSketchNet that CAD models need separate attention as opposed to using the techniques developed for regular (also sometimes termed as graphical) models.

This paper, therefore, presents a deep learning approach to clean or enhance the query sketches. A dataset of enhanced query sketches is first created based on an understanding of the possible defects in the query sketches. Following this, a deep neural network architecture (SketchCleanNet) is proposed, that processes the input query sketch and generates a clean/enhanced query sketch. An end-to-end training of SketchCleanNet is carried out in order to provide a mapping between the defective and clean sketches. The proposed approach is also compared, both qualitatively and quantitatively, with other state-of-the-art techniques, and is shown to be more effective. Using the enhanced query sketches yields better search results than using the query sketches with defects, as shown in Fig. 2. The key contributions of the paper are:

- The first learning-based strategy to clean the rough query sketches of 3D CAD models, to the best of the authors knowledge
- Introduces SketchCleanNet — an end-to-end image translation scheme to understand the mapping between rough sketches and clean query images
- A novel scheme to calculate the loss based on a weighted combination of pixel probabilities using Kernel Density Estimation and Weighted Cross entropy Loss
- Dataset Contribution: The resulting enhanced query sketch dataset will be made available publicly

The paper is organized as follows: Section 2 discusses the related works. The dataset preparation is presented in Section 3 followed by the proposed methodology in Section 4. A report of the results followed by a discussion on the state-of-the-art comparison is provided in Section 5. Section 6 discusses the potential limitations and future work, followed by a conclusion (Section 7).

2. Related works

There have been very few studies that have focused on sketch cleanup and enhancement as a research topic in and of itself. There is little available research on the enhancement of query sketches of 3D CAD models, not to mention the available literature on learning-based approaches to such a problem. As a result, related works pertaining to algorithm-based sketch cleanup of image data and generic 3D shapes are discussed. Additionally, a few deep learning based edge detection techniques are also discussed since they aim at generating enhanced sketches from the input image(s). A summary of the available 3D CAD model datasets is also presented.

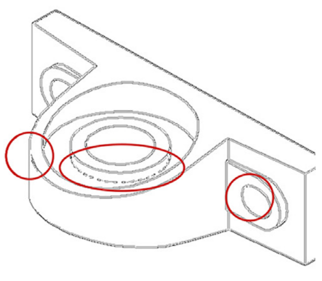
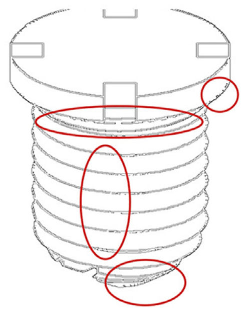
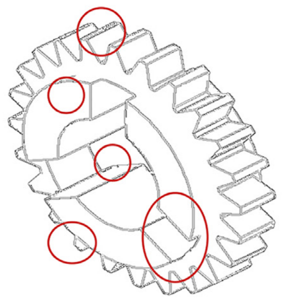
Incomplete portions Dotted lines	Incomplete portions Dotted lines Duplicate lines	Incomplete portions Dotted lines Duplicate lines Extra/unwanted lines
		

Fig. 2. Illustrating the various types of defects that could be present in a query sketch.

2.1. Algorithms for sketch simplification

[26] were one of the first to present an algorithm for cleaning rough sketches. An input vector drawing is simplified by clustering input strokes based on the geometry, replacing each cluster with a single representative line/curve. Consequently, many works such as [27–29] were proposed, that follow a similar line of thought by classifying or clustering the input strokes and replacing each cluster with a single line/curve. The work presented by [30] focused on simplifying raster inputs. [31,32] also focus on cleaning rough raster sketches using Delaunay triangulation and Bézier splines respectively. [33] provides a benchmark for rough sketch cleanup by comparing many state-of-the-art techniques.

[34] have introduced a neural network based approach to cleaning rough sketches, where a series of convolutional layers are used. A new dataset that contains 68 pairs of rough and simplified sketches (obtained from 5 artists) is used to train the Fully Convolutional Network. The authors attempt to better their sketch simplification by proposing an unsupervised Generative Adversarial Network (GAN) for sketch simplification [35], in an attempt to overcome the limited availability of annotated training pairs in their previous work. All these methods, however, only aim at simplifying the sketches of generic shapes and objects, and do not focus on the 3D shapes of engineering parts and components.

2.2. Edge detection techniques

Edge-detection techniques have been traditionally used for several computer vision tasks. A survey of classical edge detectors is provided in [36]. With the advent of neural networks, many sophisticated edge detection techniques have been proposed. An effective edge detection procedure could potentially serve as a sketch generation algorithm. Holistically Nested Edge Detection (HED) was one of the earlier methods to use a neural network approach for an end-to-end edge detection system [37]. DeepEdge [38] proposes a multi-scale bifurcated deep network, consisting of two independently trained branches. While one branch attempts to detect contours, the second branch is optimized to depict the fraction of human annotators agreeing to the presence of a contour.

[39] have introduced a Scale Enhancement Module to their convolutional neural net (CNN), that effectively increases the size of the receptive fields of network neurons. [40] attempt to build a robust CNN model (DexiNed) for edge detection, inspired by both HED and Xception [41] networks. A large dataset with edge annotations is also introduced here. However, the dataset consists only of objects from the generic object categories. [42] proposed the idea of neural contours, wherein a neural network is trained to generate line drawings of 3D models. The proposed approach yields state-of-the-art drawings, owing to its complex neural network pipeline, but at a heavy computational cost.

2.3. Available sketch datasets of common objects

[12] provided one of the first sketch datasets, containing hand-drawn sketches corresponding to every 3D model in the Princeton Shape Benchmark [13]. A few instances of the Shape Retrieval Contest (SHREC) have also provided benchmark sketch datasets [14,15]. The QuickDraw dataset [43] provides a collection of vector drawings from an online game, where users have to provide rough sketches in less than 20 s. The OpenSketch dataset [44] contains annotated product design sketches, including a detailed study of stroke pressure and drawing time etc. The dataset, however, contains only a limited number of sketches (107) across 12 categories. The ProSketch3D [45] consists of 1500 sketches of 3D models across 500 object categories, taken from ShapeNet [46].

2.4. Datasets of 3D CAD models

The Engineering Shape Benchmark (ESB) [23] was one of the earlier datasets of 3D CAD models, with 801 CAD models across 42 classes. The CADNET dataset [47] combines the ESB dataset and the National Design Repository (NDR) [48] and augments them with additional CAD models, resulting in 3317 CAD models across 43 classes. [49] have introduced the ABC dataset with a million CAD models. However, it is difficult to make use of this dataset for tasks such as automated classification and retrieval, because the ground truth information such as category labels is not available in the dataset. The Mechanical Components Benchmark (MCB) [25] is the latest benchmark dataset of 3D CAD models, containing 58,696 CAD models across 68 categories. These datasets only provide the 3D CAD model data and do not contain any sketch information.

A sketch dataset for 3D CAD models is proposed by [50], consisting of 2148 CAD models and six corresponding views of each model. However, this dataset is proprietary and is not available. [51] introduce SketchGraphs, a sketch-dataset for CAD models based on the design workflow as opposed to the model shape and geometry. These datasets are not useful in developing a deep learning based search engine for 3D CAD models. The recently proposed CADSketchNet dataset [24], contains 801 hand-drawn sketches for every CAD model in the ESB. Additionally, a

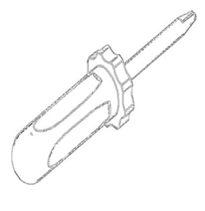
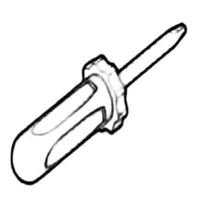
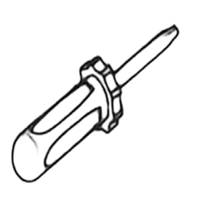
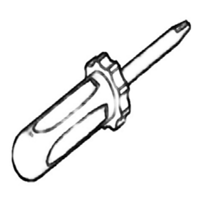
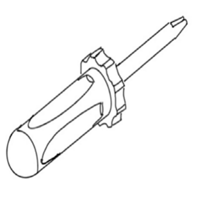
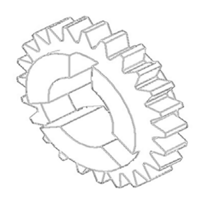
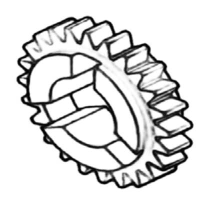
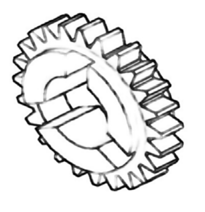
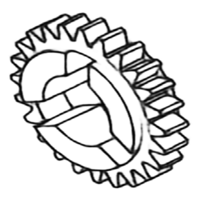
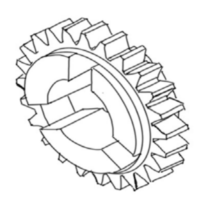
Computer generated sketches from CADSketchNet	Computer generated sketches using BDCN	Computer generated sketches using HNED	Computer generated sketches using DexiNed	Hand-drawn sketches by tracing object boundaries
				
				

Fig. 3. Hand-drawn sketches by tracing object boundaries provide the best ground truth for training. Bi-Directional Cascade Network (BDCN) for edge detection [52]; HNED — Holistically-Nested Edge Detection [53]; DexiNed — Dense Extreme InceptionNet [54].

weighted combination of the Canny edge detection algorithm and Gaussian Blurring is proposed as a sketch-generation algorithm, which is used to obtain 58,696 computer-generated sketches for every CAD model in the MCB. This dataset is both large-scale and well-annotated, making it a suitable choice for developing deep learning based solutions.

3. Dataset preparation

Given a rough sketch of a 3D CAD model as an input, the goal of this research is to build a deep learning model that can generate clean or enhanced query sketches. The dataset needed to train such a deep learning model, should contain training pairs as follows: (X, Y) — where X is the rough sketch of the 3D CAD model (which is to be cleaned), and Y is the clean image (ground truth). A significant number of such training examples are needed, in order for the deep learning model to effectively learn the mapping between the images. In the absence of any such dataset, a new dataset is to be constructed from scratch.

For this purpose, the computer-generated sketches from the CADSketchNet dataset are used as the rough sketches (X). While the proposed sketch-generation algorithm in CADSketchNet [24] is automatically able to produce computer-generated query sketches for 3D CAD models, the resulting sketches have a few defects. A brief overview of various sketches in the dataset indicates the presence of certain defect types, such as missing partial input lines, presence of duplicate lines, unwanted mesh lines in sketches, extra lines that are not present in input etc. Since the retrieved results of a search engine are only as good as the input query, these sketches need cleaning-up or enhancement in order to obtain better retrieval results. These sketches, therefore, serve as excellent candidate images for X.

The ground truth Y (clean images) of the corresponding sketches are not available in the dataset. Edge detection techniques result in images that can potentially be used as sketches. Hence, some of the popular neural network based edge detectors are experimented with, to obtain potential candidate images for Y. After due experimentation, it has been found that none of the networks are able to provide clean sketch images (see Fig. 3). Hence, we manually generate training pairs by hand-drawing sketches by tracing the object boundaries for every CAD model. This method results in clean and sharp sketch images, which are then used as the ground truth image Y. It is important to note that the 'traced' sketches are obtained only to be used as ground truth images for training and not as a substitute for the query images provided to the search engine. It should also be noted that the ground truth is obtained merely by tracing the object boundaries from the CAD model images and not "drawing" new sketches. Additionally, it is made sure that the obtained ground truth are free from potential noise and errors by validating them by a taking a majority vote among a group of volunteers with knowledge of CAD and drawing. This avoids the potential issue of worsening the input sketch by introducing an intermediate sketch enhancement module.

Since it is not possible to manually obtain the 'traced' sketches for a large dataset such as MCB, training pairs are generated only for the 801 CAD models in ESB. It should be noted here that the X here contains 801 computer-generated sketches of ESB, using the sketch-generation algorithm proposed by [24]. These are not the 801 hand-drawn sketches from CADSketchNet. As discussed earlier in this section, although [24] proposed an algorithm to obtain computer-generated query sketches for 3D CAD models, these sketches have a few defects. These defective query images of the ESB are hence used as candidate images for X and the corresponding traced sketches as the clean images for Y(ground truth). This dataset with 801 pairs is split into 632 (train set) and 169 (test set) respectively. This is the standard 80-20 split ratio as per the Pareto principle. The dataset can be found at https://github.com/bharadwaj-manda/SketchCleanNet.

4. Network architecture

The problem at hand involves an end-to-end training, an attempt to map the relationship between the two images — the rough sketch X and the clean sketch Y. A traditional fully connected network cannot be used since it does not handle image inputs. A convolutional neural network (CNN) can process image inputs, but the fully connected layers at the end are meant for classification tasks. The suitable choice of network architecture for an end-to-end image translation problem would be the use of a Fully Convolutional Network (FCN). The characteristic features of an FCN are: (1) the absence of any fully connected layers; (2) the use of up-sampling or de-convolution layers; and (3)

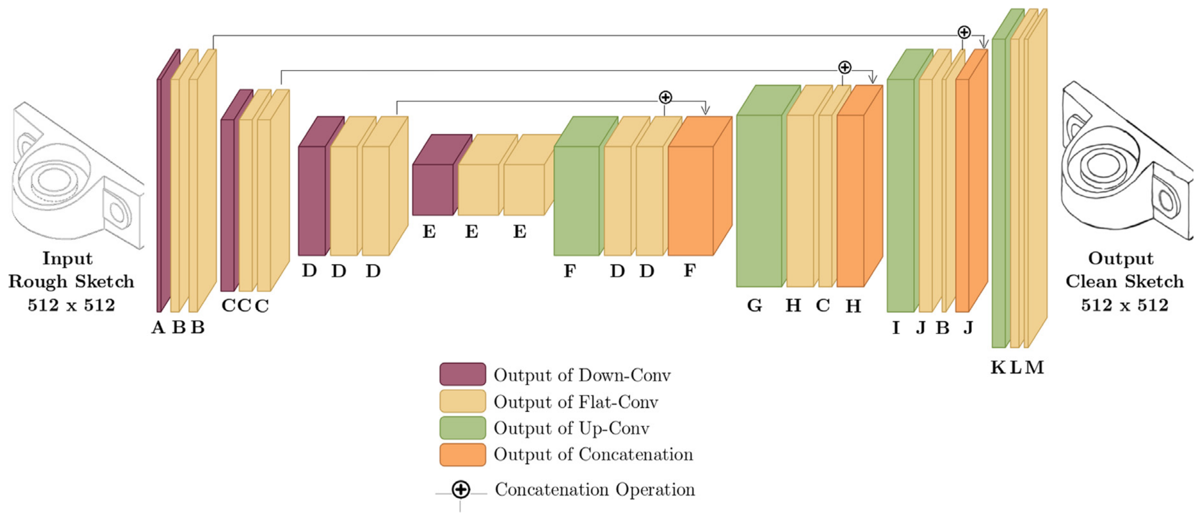


Fig. 4. Proposed Fully Convolutional Network Architecture called 'SketchCleanNet'. The dimensions at every layer are **A:** 32 × 256 × 256; **B:** 64 × 256 × 256; **C:** 128 × 128 × 128; **D:** 256 × 64 × 64; **E:** 512 × 32 × 32; **F:** 512 × 64 × 64; **G:** 512 × 128 × 128; **H:** 256 × 128 × 128; **I:** 256 × 256 × 256; **J:** 128 × 256 × 256; **K:** 128 × 512 × 512; **L:** 64 × 512 × 512; **M:** 32 × 512 × 512.

Table 1
Comparing The effect of skip connections on the network performance across different metrics. Using skip connections helps to obtain a better reconstruction of the input image. MSE — Mean Squared Error; BDCN Loss — [39]; PSNR — Peak Signal-to-Noise Ratio; SSIM — Structured Similarity Index [55]; ↑ — greater value for the metric indicates higher similarity; ↓ — lesser value for the metric indicates higher similarity.

	MSE ↓	L1 ↓	BDCN Loss ↓	PSNR ↑	SSIM ↑
With skip connections	**0.0149**	**0.026**	**0.9024**	**18.86**	**0.9208**
Without skip connection	0.0171	0.0281	0.9827	18.29	0.9101

the presence of an encoder–decoder network, resulting in an hourglass-shaped architecture.

The proposed network architecture for the current task is shown in Fig. 4. We name this network 'SketchCleanNet' (referred to as SCNet henceforth). The network consists of four types of operations: down-convolution, flat-convolution, up-convolution, and concatenation using skip connections. Down-convolution is the standard convolution operation that decreases the input size by a pre-determined factor. The flat-convolution operation performs the standard convolution operation while maintaining the input size. Up-convolution operations are used in the decoder portion of the network, which merely attempts to up-sample the data to a higher dimension. The network also contains skip connections, which are mainly used to avoid the issue of vanishing gradient and to aid the reconstruction of input structure in the output images. Concatenating a few layers in the decoder part with skip connections from the encoder portion aids in reconstructing the image to the original dimension. Table 1 shows the effect of skip connections on the network performance.

The key highlights of the proposed network architecture are:

- The absence of pooling layers: Ideally, a pooling operation is expected to extract only useful information and discard irrelevant details. However, this might not be the case every time. There is always a scope for losing useful information. For the current scenario, since the fraction of pixels in the input sketch that contain lines/curves is already small, the pooling layers are not used in the proposed network. This has been verified by experimentation, wherein the network that contained pooling layers could not provide an accurate reconstruction of the input image.
- Substituting the use of pooling layers with down-convolution layers: Convolution layers contain trainable parameters as opposed to a pooling layer. Therefore, the advantages of substituting the pooling operation with down-convolution layers are two-fold: (1) No information is lost as a result of pooling; (2) Aids in training the network to obtain a better mapping between the input and output images.
- The use of flat-convolution layers to avoid shrinking the image dimension each time a filter is applied from the convolution operation. This is mainly done to keep the network deep. Otherwise, the input image dimensions would quickly reduce, which drastically reduces the number of convolution layers that can be used in the network architecture.

4.1. Implementation details

The current choice of network architecture for SCNet (SCNet) is made after due experimentation with respect to the number of layers, kernel sizes for convolutions, and so forth. The choices of such hyper-parameters are based on heuristics and practical training strategies such as [56,57]. The network is trained (on the training set that contains 632 training pairs) for 200 epochs, and uses the Adam optimization algorithm [58]. A learning rate of $3e{-}04$ and a batch size of 8 are used. ReLU is used for activation in all layers. A kernel size of (3,3) is used in all convolutional layers. The stride for every down-convolution layer is (2,2) with padding = 1; the flat-convolution layer is (1,1) with padding = 1. A scale of 2 is used in each up-convolution layer.

4.2. Choice of loss function

While the standard cross-entropy loss function is typically used for image classification tasks, it does not yield good results in the current scenario. This is because the distribution of edge/non-edge pixels in a typical sketch image is substantially skewed — at least 90% of the ground truth is white pixels. The Bi-Directional Cascade Network (BDCN) for edge detection [52] uses a simple strategy to counter this, which essentially calculates the number of positive and negative pixels (with edge and without edge). Every pixel that has a value below a certain threshold is

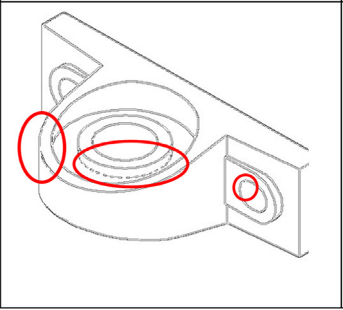
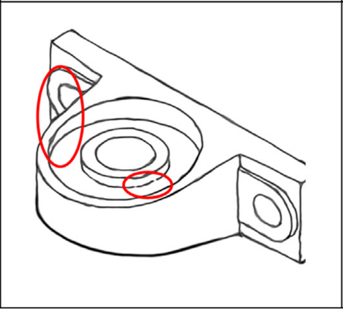
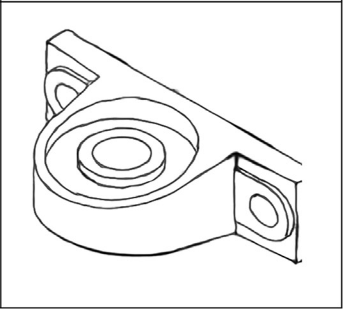
Input Rough Sketch	Result using Loss (Eq 1)	Result using custom loss (Eq 6)
		

Fig. 5. Using the custom loss function yields cleaner and sharper sketch images. The network output using the loss from Eq. (1) still contains gaps and blurred regions, which are resolved using the proposed loss function.

Table 2
The effect of various for λ_1 and λ_2. Using the values of 0.8 and 0.2 gives the best performance across different metrics. MSE — Mean Squared Error; BDCN Loss — [39]; PSNR — Peak Signal-to-Noise Ratio; SSIM — Structured Similarity Index [55]; ↑ — greater value for the metric indicates higher similarity; ↓ — lesser value for the metric indicates higher similarity.

λ_1	λ_1	MSE ↓	L1 ↓	BDCN Loss ↓	PSNR ↑	SSIM ↑
0.8	**0.2**	**0.0149**	0.026	**0.9024**	**18.86**	**0.9208**
0.7	0.3	0.0167	0.0276	1.1299	18.37	0.9188
0.85	0.15	0.0155	0.0264	1.0264	18.74	0.9192
0.5	0.5	0.0153	**0.0257**	0.9689	18.79	0.9202
1	0	0.0151	0.0258	0.9526	18.82	0.9199

considered as a positive pixel (Y_+) and other pixels are considered negative pixels (Y_-). The consequent loss function is defined as:

$$\mathcal{L}_1 = -\alpha \sum_{j \in Y_-} \log\left(1 - \hat{y}_j\right) - \beta \sum_{j \in Y_+} \log\left(\hat{y}_j\right) \tag{1}$$

where $\alpha = \lambda \cdot |Y_+| \,/\, (|Y_+| + |Y_-|)$ and $\beta = |Y_-| \,/\, (|Y_+| + |Y_-|)$ are the relative weights for the positive and negative pixels, with λ being a hyper-parameter to control the weight of positive over negative samples.

DeepHist [59] proposes the idea of a Kernel Density Estimation (KDE) function. The main idea here is to represent the image as a histogram, where the height of each bin in the histogram represents the probability of a range of pixels being present in that bin. For instance, the outputs of SCNet are single-channel sketch images which have a large number of white pixels. So the bin corresponding to the group of white pixels will be taller in the histogram, thus indicating a high probability of the presence of white pixels.

For training SCNet, we use a loss function that is similar to the BDCN loss function described in Eq. (1), while also integrating the idea of KDE. The BDCN loss calculates the weights of positive and negative pixels based on a hyper-parameter threshold. This is substituted by the use of KDE. The kernel used for the current KDE implementation is given by,

$$k(z) = \frac{1}{\sqrt{2\pi}\sigma} \times \exp\left(-\frac{z^2}{2\sigma^2}\right), \tag{2}$$

where σ is the variance of the distribution.

The KDE function is given by,

$$\hat{f}(g) = \frac{1}{N} \sum_{x \in \Omega} k(I(x) - g), \tag{3}$$

where N is the number of pixels in the ground truth, $I(x)$ is the intensity of the pixel x and Ω is the set of all pixels in the ground truth image. The method used to calculate the weights for the positive and negative pixels is as follows. All pixels of the ground truth image are in the range [0,1]. If we have K bins, each bin will be of the length $\frac{1}{K}$. The probability of a pixel g falling in the kth bin (B_k) is given by,

$$\mathrm{P}\left(g \in B_k\right) = \int_{B_k} \hat{f}(g) \cdot dg \tag{4}$$

The bin with the maximum probability (P_{max}) of containing the pixel g is determined. This process is repeated for every pixel in the ground truth image and these probability values are then used as weights in the loss function. The expression for the loss function using Kernel Density Estimation is given by,

$$\mathcal{L}_2 = -\sum_{g} P_{max_g} \times y_g \log\left(\hat{y}_g\right) \tag{5}$$

The custom scheme to calculate losses at the final layer of the network uses a weighted combination of the BDCN loss (Eq. (1)) and the loss using KDE (Eq. (5)).

$$\mathcal{L} = \lambda_1 \mathcal{L}_1 + \lambda_2 \mathcal{L}_2, \tag{6}$$

where λ_1, λ_2 are hyperparameters. The values for λ_1, λ_2 used in the current implementation are 0.8 and 0.2 respectively, identified after due experimentation. Table 2 shows the effect of λ_1 and λ_2 for a few combinations. Using the custom loss function shows a significant improvement in results over using the BDCN loss alone (see Fig. 5).

4.3. Coding framework and system configuration

For implementing the neural network, Python3 with PyTorch is used. OpenCV library is used for all image-based implementations. All the implementations are carried out on a system running Ubuntu 20.04 Operating System. The system has an Intel Core i7-8700K CPU with 64 GB RAM and an NVIDIA RTX 2080Ti GPU with 12 GB RAM.

5. Results and discussion

The results of the proposed SCNet are reported in this section, followed by a discussion on the comparison of results with other state-of-the-art approaches and on other sketch datasets. All the reported results are on the test set, containing 169 images.

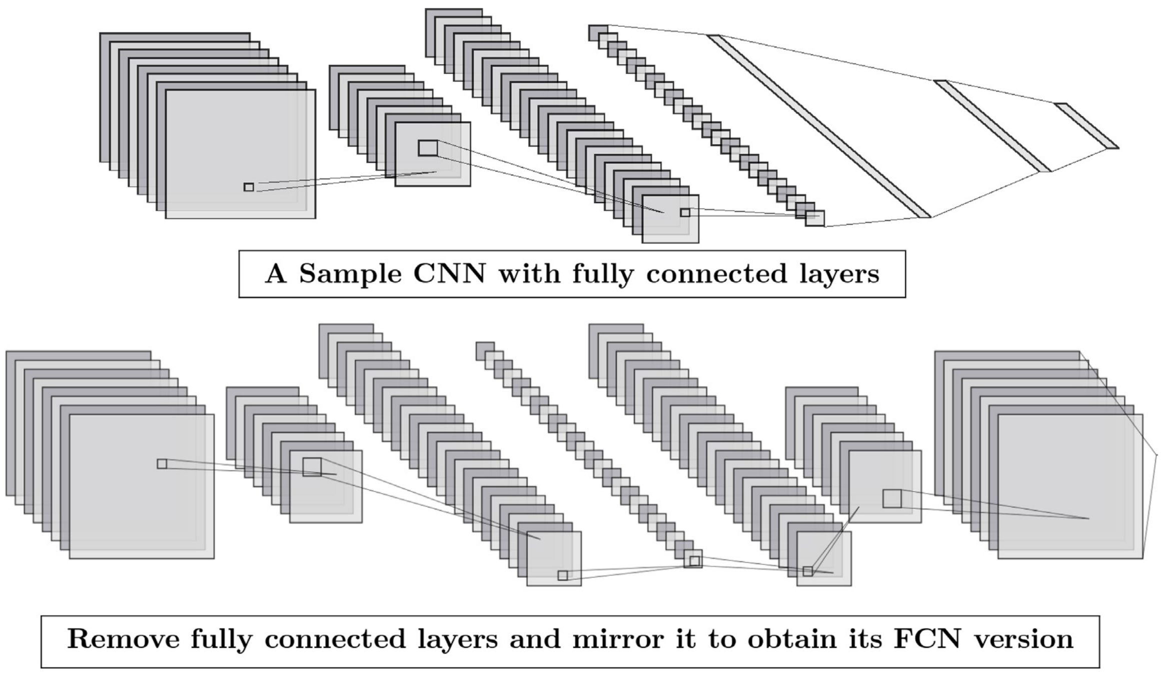

A Sample CNN with fully connected layers
Remove fully connected layers and mirror it to obtain its FCN version

Fig. 6. Method used to obtain the fully convolutional version of any sample CNN with dense layers at the end.

Table 3
Similarity results when comparing the outputs of various networks with ground truth (on the test set). SCNet gives the closest results. MSE — Mean Squared Error; BDCN Loss — [39]; PSNR — Peak Signal-to-Noise Ratio; SSIM — Structured Similarity Index [55]; ↑ — greater value for the metric indicates higher similarity; ↓ — lesser value for the metric indicates higher similarity.

Network	MSE ↓	L1 loss ↓	BDCN Loss ↓	PSNR ↑	SSIM ↑
FC-VGGNet	0.020	0.0373	1.5225	16.90	0.8962
FC-ResNet	0.016	0.0307	1.1245	17.80	0.9115
DenseNet	0.050	0.1012	4.4420	13.00	0.8129
Simo-Serra et al. [34]	0.0173	0.0287	1.2270	17.61	0.9153
GCN	0.020	0.039	1.6354	16.62	0.8901
SCNet	**0.0149**	**0.0260**	**0.9024**	**18.24**	**0.9208**

5.1. Comparing results with other networks

Only a few state-of-the-art FCN architectures exist in the literature, such as the Global Convolutional Network (GCN) proposed by [60]. Even so, these networks are mostly aimed towards image segmentation and not directly on sketch enhancement. Hence, we make use of a few popular ImageNet architectures, by removing the fully connected layers and mirroring them. This results in a fully convolutional version of these networks. This process is summarized in Fig. 6 for a sample CNN.

Some sample visualizations of the results can be seen in Fig. 7. The fully convolutional version of VGGNet [61] does not perform well and results in images that are blurred, The outputs also contain many smudges and defects of the input rough sketch are not fully removed. Similar characteristics are observed for the images generated by the fully convolutional version of ResNet [62], although the images are a lot sharper. GCN performs very badly, probably because the architecture is mainly suited for image segmentation as opposed to a dedicated sketch enhancement task. It is easy to observe that the images generated by SCNet are a lot sharper with most of the input defects removed. They also give a more natural feel, bearing a closer resemblance to hand-drawn sketches.

In an attempt to quantify the visual results obtained by these methods, the output images from each of the above networks are compared for similarity with the ground truth clean sketches. Table 3 reports the similarity values obtained by the networks using various similarity metrics. From the table, it can be seen once again that the SCNet provides the clean sketches which are closest to the ground truth for every tested metric of similarity.

5.2. Results of SCNet on sketch datasets of CAD models

SCNet has been trained on the rough and clean image pairs of the ESB dataset. The rough sketches of ESB were obtained from the CADSketchNet dataset. CADSketchNet also contains query sketches for every CAD model in the MCB dataset. SCNet could not be trained on these since it was difficult to obtain the ground truth clean sketches. Nonetheless, the trained SCNet can now be tested on the MCB sketches and the performance can be evaluated. A random sample of 1000 sketches of the MCB models is chosen from CADSketchNet and are fed to SCNet. A few sample results of the cleaned sketches are shown in Figs. 8 and 9. It can be seen that a majority of the defects are removed from the input sketches, and the network produces good results.

5.3. Results of SCNet on other sketch datasets

The performance of SCNet is also analysed on inputs from other sketch datasets such as OpenSketch [44] and the sketches corresponding to the 3D shapes of common objects from the Princeton Shape Benchmark [12]. It is to be noted here that these datasets contain sketches of generic 3D shapes and not of 3D engineering parts and components. This experiment, therefore, serves as a cross-domain test of the performance of SCNet. Fig. 10 shows some sample results.

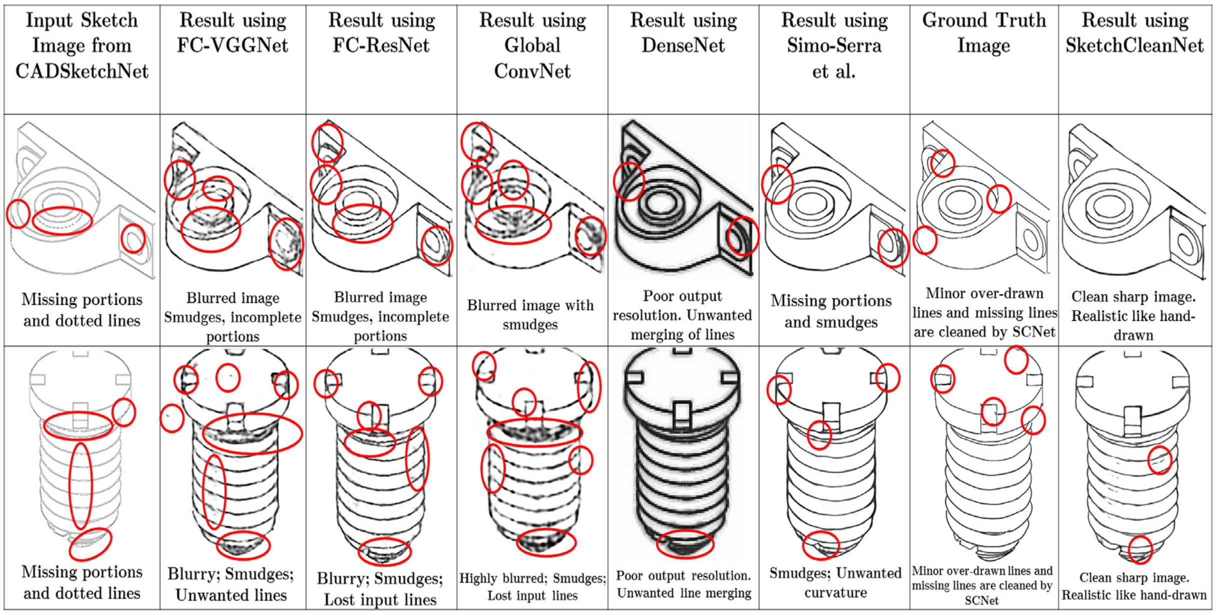

Input Sketch Image from CADSketchNet
Result using FC-VGGNet
Result using FC-ResNet
Result using Global ConvNet
Result using DenseNet
Result using Simo-Serra et al.
Ground Truth Image
Result using SketchCleanNet
Missing portions and dotted lines
Blurred image Smudges, incomplete portions
Blurred image Smudges, incomplete portions
Blurred image with smudges
Poor output resolution. Unwanted merging of lines
Missing portions and smudges
Minor over-drawn lines and missing lines are cleaned by SCNet
Clean sharp image. Realistic like hand-drawn
Missing portions and dotted lines
Blurry; Smudges; Unwanted lines
Blurry; Smudges; Lost input lines
Highly blurred; Smudges; Lost input lines
Poor output resolution. Unwanted line merging
Smudges; Unwanted curvature
Minor over-drawn lines and missing lines are cleaned by SCNet
Clean sharp image. Realistic like hand-drawn

Fig. 7. The proposed SCNet Architecture gives the best results. FCVGGNet, FCResNet — fully convolutional versions of VGGNet [61] and ResNet [62] respectively, obtained from process described in Fig. 6; Global ConvNet - [60]; DenseNet [63]; Simo-Serra et al. [34].

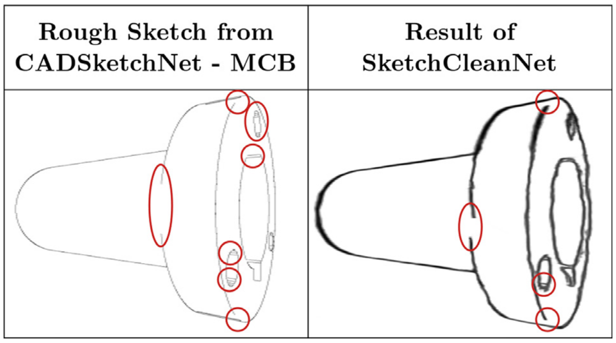


Fig. 8. Result of SCNet on MCB — Sample 1. Most defects are removed from the input sketches.

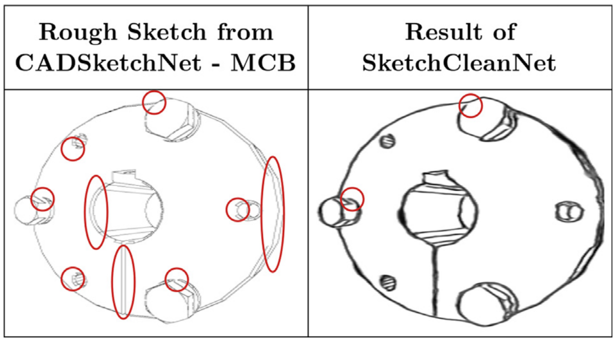


Fig. 9. Result of SCNet on MCB — Sample 2. Most defects are removed from the input sketches.

For sketches from the OpenSketch dataset, minor defect corrections such as completion of missing lines, removing unwanted lines, etc. are achieved. The first sample result from Fig. 10 shows that certain multi-stroke portions are simplified. However, most sketches from the OpenSketch dataset contain images with heavy multi-stroke regions, and no improvement is observed for such images. This is probably because the network treats these lines as separate input lines/curves rather than multiple strokes for the same line/curve (see sample result in the second row).

Most sketches of the PSB models are already simplified, have continuous lines and curves in the input, and do not require any major corrections. The results shown in Fig. 10 show only minor improvements, such as correction of zig-zag lines to uniform curves etc. However, there is some unwanted blurring of some portions of the image, as seen in both the sample results. This is because the PSB sketches already contain clear and sharp input lines/curves to begin with. Using SCNet, which is trained on rough sketch inputs with multiple defects (specific to engineering components and not regular 3D shapes), might be unnecessary for such cases.

5.4. Quantitative comparison of results by using enhanced query sketches for retrieval

The ultimate test for evaluating the outputs generated by SCNet is done by actually verifying if the cleaned sketches provide better retrieval results as compared to the sketches with defects. A deep learning based search engine for 3D CAD models has been presented along with the CADSketchNet dataset by [24], which is modelled using a Siamese Network. It is also reported there, that a Siamese Network using ResNet18 for both CNN pipelines yielded the best results. A search engine with the same configuration is used in this paper as well, to compare the retrieval results when trained using (1) sketches from CADSketchNet and (2) SCNet.

The results of retrieval using both the sets of query sketches are reported in Table 4. For search and retrieval, the value for recall is simply the probability that a relevant data item is retrieved by the query, while precision denotes what fraction of the retrieved data items are relevant to the query. Clearly, higher values of precision and recall indicate better retrieval performance. In addition, a comparison of the top *k*-accuracy value is also reported. If 8 out of the 10 most similar results belong to the same class as the query, then the top 10-accuracy value is 80%. In Table 4, the values reported against these metrics are the average values obtained over all categories of 3D CAD models in the database. These values show that using the cleaned sketches generated by SCNet gives better retrieval performance, for each

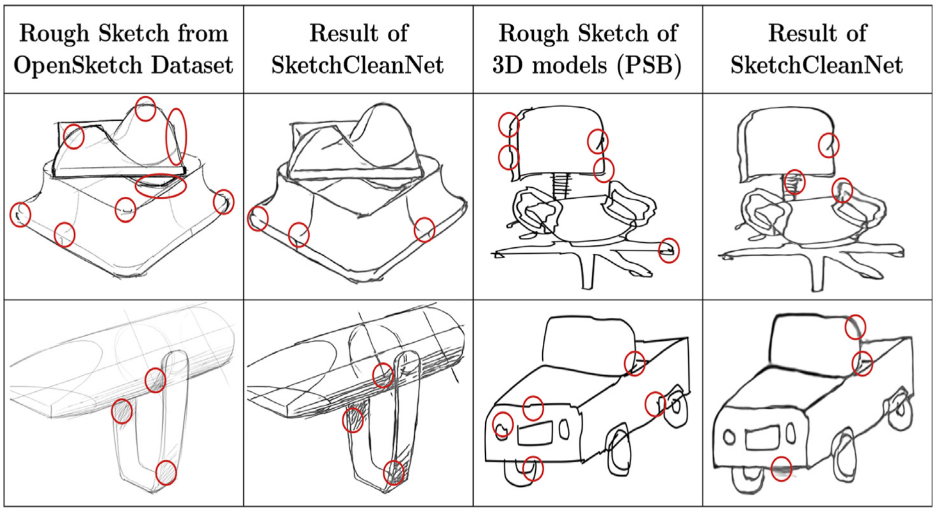

Rough Sketch from OpenSketch Dataset
Result of SketchCleanNet
Rough Sketch of 3D models (PSB)
Result of SketchCleanNet

Fig. 10. Results of SCNet on sketches from OpenSketch and models of PSB. Results on OpenSketch: Image1 — simplification of multi-stroke regions; Image2 — no correction as multiple strokes are being treated as distinct input lines. Results on sketches of PSB: Unwanted blurring of some portions of the images since the input sketches already contain clear, sharp stokes.

Table 4
Clean sketches generated by SCNet give better retrieval performance when compared to the defect sketches from CADSketchNet and sketches generated by other networks.

	CADSketchNet	VGGNet	ResNet	GCNet	DenseNet	Simo-Serra	SCNet
Top k accuracy	94.23	92.51	95.79	92.23	91.51	96.05	96.43
Precision	0.9375	0.91667	0.92308	0.87500	0.88889	0.90000	0.9412
Recall	0.4545	0.3818	0.42222	0.34705	0.33889	0.43529	0.4705
Mean Retrieval Time	1.38E−05	1.42E−05	1.38E−05	1.49E−05	1.39E−05	1.40E−05	1.39E−05

of the evaluation metrics. This demonstrates that the retrieved results of a search engine are only as good as the input query, and also justifies the need for a dedicated query sketch enhancement module in an image-based search engine.

6. Limitations and future work

The current work focuses on the enhancement and correction of query sketches, aimed at improving the retrieval performance of a 3D CAD model search engine. Hence, the network is trained specifically on sketches of engineering shapes and parts. Due to this, the network will not always translate well to sketches from other domains, such as the sketches of regular shapes (3D models of common objects). Obtaining sketches from multiple contexts and augmenting the existing dataset to build unified model that works across domains is an interesting future work.

The primary goal of the paper is to provide better query images for a CAD model search engine, and that goal has been achieved. However, a detailed analysis and characterization of the various types of defects present in the query sketches, and developing image enhancement solutions specific to the type of defect is worthy of further investigation. Augmenting the dataset with sketches that contain varying degrees of noise and defects is also an interesting future work that helps in building a robust network. In addition, the current approach only considers query sketches from a single orientation of the 3D model. This is because the end-user of a search engine would typically expect a search result using a single query image. However, the proposed methodology could also be extended to process multi-view sketch queries, i.e, sketches of a CAD model from multiple orientations as opposed to a single query sketch.

7. Conclusion

The retrieved results of a search engine are only as good as the input query. Typical query sketches of 3D CAD models contain various types of defects, such as missing portions, duplicate lines, the presence of unwanted mesh lines, etc. This paper, therefore, presents SketchCleanNet — a deep learning approach to enhance and correct such rough query sketches of 3D CAD models. The network is trained using a ground truth dataset of clean query sketches, and this dataset will be made available publicly along with the enhanced query images generated by the network. A detailed comparison of the results of SCNet with other architectures is reported, with the proposed architecture significantly outperforming other approaches. Input sketches from sketch datasets of generic 3D shapes are also used to test the performance of the network. Finally, the generated clean sketches from the network are then used as query images to train a deep learning based search engine for 3D CAD models. A quantitative comparison of retrieval results demonstrates that utilizing the query images from SCNet yields significantly better results for retrieval, while also justifying the need for a dedicated query sketch enhancement module in an image-based search engine.

CRediT authorship contribution statement

Bharadwaj Manda: Conceptualization, Methodology, Software, Validation, Data curation, Writing – original draft, Visualization. **Prasad Pralhad Kendre:** Conceptualization, Methodology, Software, Validation, Data curation. **Subhrajit Dey:** Methodology, Software, Validation, Data curation. **Ramanathan Muthuganapathy:** Conceptualization, Resources, Writing – review & editing, Supervision, Project administration.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

Thanks are due to the teams of the ESB and the MCB datasets for making their data publicly available. Thanks are also due to many users who have contributed to our CADSketchNet dataset.

References

[1] Baeza-Yates R, Ribeiro-Neto B, et al. Modern information retrieval, Vol. 463. ACM press New York; 1999.
[2] Schütze H, Manning CD, Raghavan P. Introduction to information retrieval, Vol. 39. Cambridge University Press Cambridge; 2008.
[3] Funkhouser T, Min P, Kazhdan M, Chen J, Halderman A, Dobkin D, et al. A search engine for 3D models. ACM Trans Graph 2003;22(1):83–105. http://dx.doi.org/10.1145/588272.588279.
[4] Tangelder JWH, Veltkamp RC. A survey of content based 3D shape retrieval methods. In: Proceedings shape modeling applications, 2004. 2004, p. 145–56. http://dx.doi.org/10.1109/SMI.2004.1314502.
[5] Bustos B, Keim D, Saupe D, Schreck T. Content-based 3D object retrieval. IEEE Comput Graph Appl 2007;27(4):22–7. http://dx.doi.org/10.1109/MCG.2007.80.
[6] Lee J, Funkhouser T. Sketch-based search and composition of 3D models. In: Proceedings of the fifth eurographics conference on sketch-based interfaces and modeling. SBM'08, Goslar, DEU: Eurographics Association; 2008, p. 97–104.
[7] Shin H, Igarashi T. Magic canvas: Interactive design of a 3-D scene prototype from freehand sketches. In: Proceedings of graphics interface 2007. GI '07, New York, NY, USA: Association for Computing Machinery; 2007, p. 63–70. http://dx.doi.org/10.1145/1268517.1268530.
[8] Aono M, Iwabuchi H. 3D shape retrieval from a 2D image as query. In: Proceedings of the 2012 Asia pacific signal and information processing association annual summit and conference. 2012, p. 1–10.
[9] Min P. A 3D model search engine (Ph.D. thesis), Princeton University; 2004.
[10] Blümel I, Berndt R, Ochmann S, Vock R, Wessel R. PROBADO3D-indexing and searching 3D CAD databases: Supporting planning through content-based indexing and 3D shape retrieval. Des Decis Support Syst 2010.
[11] Li W, Mac G, Tsoutsos NG, Gupta N, Karri R. Computer aided design (CAD) model search and retrieval using frequency domain file conversion. Addit Manuf 2020;36:101554.
[12] Eitz M, Richter R, Boubekeur T, Hildebrand K, Alexa M. Sketch-based shape retrieval. ACM Trans Graph 2012;31:31:1–31:10.
[13] Shilane P, Min P, Kazhdan M, Funkhouser T. The princeton shape benchmark. In: Proceedings shape modeling applications, 2004. 2004, p. 167–78. http://dx.doi.org/10.1109/SMI.2004.1314504.
[14] Li B, Lu Y, Godil A, Schreck T, Aono M, Johan H, et al. SHREC'13 track: Large scale sketch-based 3D shape retrieval. In: Castellani U, Schreck T, Biasotti S, Pratikakis I, Godil A, Veltkamp R, editors. Eurographics workshop on 3D object retrieval. The Eurographics Association; 2013, http://dx.doi.org/10.2312/3DOR/3DOR13/089-096.
[15] Li B, Lu Y, Li C, Godil A, Schreck T, Aono M, et al. Extended large scale sketch-based 3D shape retrieval. In: Bustos B, Tabia H, Vandeborre J-P, Veltkamp R, editors. Eurographics workshop on 3D object retrieval. The Eurographics Association; 2014, http://dx.doi.org/10.2312/3dor.20141058.
[16] Leizerowicz W, Bilgic T, Lin J, Fox MS. Collaborative design using WWW. In: Proceedings of WET-ICE. 1996.
[17] Funkhouser T, Kazhdan M, Shilane P, Min P, Kiefer W, Tal A, et al. Modeling by example. ACM Trans Graph 2004;23(3):652–63. http://dx.doi.org/10.1145/1015706.1015775.
[18] Albers A, Behrendt M, Klingler S, Reiß N, Bursac N. Agile product engineering through continuous validation in PGE–Product generation engineering. Des Sci 2017;3.
[19] Neb A, Briki I, Schoenhof R. Development of a neural network to recognize standards and features from 3D CAD models. Procedia CIRP 2020;93:1429–34.
[20] Lupinetti K, Giannini F, Monti M, Pernot JP. Automatic extraction of assembly component relationships for assembly model retrieval. Procedia CIRP 2016;50:472–7.
[21] Bickel S, Sauer C, Schleich B, Wartzack S. Comparing CAD part models for geometrical similarity: A concept using machine learning algorithms. Procedia CIRP 2021;96:133–8.
[22] Qin Fw, Li Ly, Gao Sm, Yang Xl, Chen X. A deep learning approach to the classification of 3D CAD models. J Zhejiang Univ Sci C 2014;15(2):91–106. http://dx.doi.org/10.1631/jzus.C1300185.
[23] Jayanti S, Kalyanaraman Y, Iyer N, Ramani K. Developing an engineering shape benchmark for CAD models. Comput Aided Des 2006;38(9):939–53. http://dx.doi.org/10.1016/j.cad.2006.06.007, Shape Similarity Detection and Search for CAD/CAE Applications.
[24] Manda B, Dhayarkar S, Mitheran S, Viekash V, Muthuganapathy R. 'CADSketchNet' - An annotated sketch dataset for 3D CAD model retrieval with deep neural networks. Comput Graph 2021;99:100–13. http://dx.doi.org/10.1016/j.cag.2021.07.001, URL https://www.sciencedirect.com/science/article/pii/S0097849321001333.
[25] Kim S, Chi H-g, Hu X, Huang Q, Ramani K. A large-scale annotated mechanical components benchmark for classification and retrieval tasks with deep neural networks. In: Proceedings of 16th European conference on computer vision. 2020.
[26] Barla P, Thollot J, Sillion FX. Geometric clustering for line drawing simplification. In: ACM SIGGRAPH 2005 Sketches. SIGGRAPH '05, New York, NY, USA: Association for Computing Machinery; 2005, p. 96–es. http://dx.doi.org/10.1145/1187112.1187227.
[27] Liu X, Wong TT, Heng PA. Closure-aware sketch simplification. ACM Trans. Graph. 2015;34(6). http://dx.doi.org/10.1145/2816795.2818067.
[28] Liu C, Rosales E, Sheffer A. StrokeAggregator: Consolidating raw sketches into artist-intended curve drawings. ACM Trans Graph 2018;37(4). http://dx.doi.org/10.1145/3197517.3201314.
[29] Ogawa T, Matsui Y, Yamasaki T, Aizawa K. Sketch simplification by classifying strokes. In: 2016 23rd International conference on pattern recognition. 2016, p. 1065–70. http://dx.doi.org/10.1109/ICPR.2016.7899777.
[30] Noris G, Hornung A, Sumner RW, Simmons M, Gross M. Topology-driven vectorization of clean line drawings. ACM Trans Graph 2013;32(1). http://dx.doi.org/10.1145/2421636.2421640.
[31] Parakkat AD, Pundarikaksha UB, Muthuganapathy R. A Delaunay triangulation based approach for cleaning rough sketches. Comput Graph 2018;74:171–81.
[32] Donati L, Cesano S, Prati A. A complete hand-drawn sketch vectorization framework. Multimedia Tools Appl 2019;78(14):19083–113.
[33] Yan C, Vanderhaeghe D, Gingold Y. A benchmark for rough sketch cleanup. ACM Trans Graph 2020;39(6). http://dx.doi.org/10.1145/3414685.3417784.
[34] Simo-Serra E, Iizuka S, Sasaki K, Ishikawa H. Learning to simplify: Fully convolutional networks for rough sketch cleanup. ACM Trans Graph 2016;35(4). http://dx.doi.org/10.1145/2897824.2925972.
[35] Simo-Serra E, Iizuka S, Ishikawa H. Mastering sketching: Adversarial augmentation for structured prediction. ACM Trans Graph 2018;37(1). http://dx.doi.org/10.1145/3132703.
[36] Ziou D, Tabbone S, et al. Edge detection techniques-an overview. Pattern Recognit Image Anal C/C Raspoznavaniye Obrazov I Anal Izobrazhenii 1998;8:537–59.
[37] Xie S, Tu Z. Holistically-nested edge detection. In: 2015 IEEE international conference on computer vision. 2015, p. 1395–403. http://dx.doi.org/10.1109/ICCV.2015.164.
[38] Bertasius G, Shi J, Torresani L. Deepedge: A multi-scale bifurcated deep network for top-down contour detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2015.
[39] He J, Zhang S, Yang M, Shan Y, Huang T. Bi-directional cascade network for perceptual edge detection. In: 2019 IEEE/CVF conference on computer vision and pattern recognition. 2019, p. 3823–32. http://dx.doi.org/10.1109/CVPR.2019.00395.
[40] Soria X, Riba E, Sappa A. Dense extreme inception network: Towards a robust CNN model for edge detection. In: 2020 IEEE winter conference on applications of computer vision. 2020, p. 1912–21. http://dx.doi.org/10.1109/WACV45572.2020.9093290.
[41] Chollet F. Xception: deep learning with depthwise separable convolutions (2016). 2016, arXiv preprint arXiv:1610.02357.
[42] Liu D, Nabail M, Hertzmann A, Kalogerakis E. Neural contours: Learning to draw lines from 3D shapes. In: 2020 IEEE/CVF conference on computer vision and pattern recognition. (CVPR), 2020, p. 5427–35. http://dx.doi.org/10.1109/CVPR42600.2020.00547.
[43] Ha D, Eck D. A neural representation of sketch drawings. 2017, arXiv preprint arXiv:1704.03477.
[44] Gryaditskaya Y, Sypesteyn M, Hoftijzer JW, Pont S, Durand F, Bousseau A. OpenSketch: A Richly-annotated dataset of product design sketches. ACM Trans Graph (Proc. SIGGRAPH Asia) 2019;38.
[45] Zhong Y, Qi Y, Gryaditskaya Y, Zhang H, Song YZ. Towards practical sketch-based 3D shape generation: The role of professional sketches. IEEE Trans Circuits Syst Video Technol 2020;1. http://dx.doi.org/10.1109/TCSVT.2020.3040900.
[46] Wu Z, Song S, Khosla A, Yu F, Zhang L, Tang X, et al. 3D ShapeNets: A deep representation for volumetric shapes. In: 2015 IEEE conference on computer vision and pattern recognition. 2015, p. 1912–20. http://dx.doi.org/10.1109/CVPR.2015.7298801.
[47] Manda B, Bhaskare P, Muthuganapathy R. A convolutional neural network approach to the classification of engineering models. IEEE Access 2021;1. http://dx.doi.org/10.1109/ACCESS.2021.3055826.
[48] Bespalov D, Ip CY, Regli WC, Shaffer J. Benchmarking CAD search techniques. In: Proceedings of the 2005 ACM symposium on solid and physical modeling. SPM '05, New York, NY, USA: ACM; 2005, p. 275–86. http://dx.doi.org/10.1145/1060244.1060275.

[49] Koch S, Matveev A, Jiang Z, Williams F, Artemov A, Burnaev E, et al. Abc: A big CAD model dataset for geometric deep learning. In: The IEEE conference on computer vision and pattern recognition. 2019.
[50] Qin F, Gao S, Yang X, Bai J, hong Zhao Q. A sketch-based semantic retrieval approach for 3D CAD models. Appl Math-A J Chin Univ 2017;32:27–52.
[51] Seff A, Ovadia Y, Zhou W, Adams R. Sketchgraphs: A large-scale dataset for modeling relational geometry in computer-aided design. 2020, arXiv, arXiv:2007.08506.
[52] He J, Zhang S, Yang M, Shan Y, Huang T. BDCN: Bi-directional cascade network for perceptual edge detection. IEEE Trans Pattern Anal Mach Intell 2020;1. http://dx.doi.org/10.1109/TPAMI.2020.3007074.
[53] Xie S, Tu Z. Holistically-nested edge detection. In: 2015 IEEE international conference on computer vision. 2015, p. 1395–403. http://dx.doi.org/10.1109/ICCV.2015.164.
[54] Soria X, Riba E, Sappa A. Dense extreme inception network: Towards a robust CNN model for edge detection. In: 2020 IEEE winter conference on applications of computer vision. Los Alamitos, CA, USA: IEEE Computer Society; 2020, p. 1912–21. http://dx.doi.org/10.1109/WACV45572.2020.9093290, URL https://doi.ieeecomputersociety.org/10.1109/WACV45572.2020.9093290.
[55] Wang Z, Bovik A, Sheikh H, Simoncelli E. Image quality assessment: From error visibility to structural similarity. IEEE Trans Image Process 2004;13(4):600–12. http://dx.doi.org/10.1109/TIP.2003.819861.
[56] Goodfellow I, Bengio Y, Courville A. Deep learning, chapter - Practical methodology. MIT Press; 2016, http://www.deeplearningbook.org.
[57] Bengio Y. Practical recommendations for gradient-based training of deep architectures. 2012, CoRR abs/1206.5533.
[58] Kingma DP, Ba J. Adam: A method for stochastic optimization. In: International conference on learning representations. 2015, arXiv:1412.6980.
[59] Avi-Aharon M, Arbelle A, Raviv TR. DeepHist: Differentiable joint and color histogram layers for image-to-image translation. 2020, arXiv preprint arXiv:2005.03995.
[60] Peng C, Zhang X, Yu G, Luo G, Sun J. Large kernel matters – improve semantic segmentation by global convolutional network. In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2017.
[61] Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition. 2014, CoRR abs/1409.1556, arXiv:1409.1556.
[62] He K, Zhang X, Ren S, Sun J. Deep residual learning for image recognition. In: 2016 IEEE conference on computer vision and pattern recognition. 2016, p. 770–8.
[63] Huang G, Liu Z, Van Der Maaten L, Weinberger KQ. Densely connected convolutional networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2017, p. 4700–8.