SketchCADGAN: A generative approach for completing partially drawn query sketches of engineering shapes to enhance retrieval system performance

First Author Given Name Surname\textsuperscript{a,*}, Second Author Given Name Surname\textsuperscript{b,1}

\textsuperscript{a}Address, City, Postcode, Country
\textsuperscript{b}Address, City, Postcode, Country

ARTICLE INFO

Article history:
Received April 12, 2023

Keywords: Engineering shapes, 3D CAD models, Search and retrieval, Generative Adversarial Networks (GANs), Sketch Completion, Sketch Inpainting, Image Inpainting, CAD model sketches, Incomplete sketches.

ABSTRACT

Retrieval systems are commonly used to find relevant data in large datasets. In engineering, these systems are useful for locating specific engineering shapes in a large dataset of engineering components. When end users want to search for a shape, they prefer a two-dimensional (2D) sketch over a three-dimensional (3D) object. However, users lacking domain knowledge may struggle to generate a complete query sketch and provide a partially completed sketch instead. Retrieving relevant information from partially drawn sketches is difficult because they may have missing edges, partially drawn circles, holes, ovals, etc. Most retrieval systems compare the similarity between the query and items in the database, so incomplete sketches may be ineffective in finding the relevant information.

To address this problem with incomplete sketches, we propose a new generative adversarial network called SketchCADGAN. This network uses a two-stage cascaded architecture, with the first network attempting to predict a CAD model image from an incomplete sketch and the second network using the CAD model image to predict a completed sketch. Both networks are trained together adversarially. Our approach is proven more effective than other advanced techniques through qualitative and quantitative comparisons. Furthermore, we present the results of the retrieval system using both partially drawn and completed sketches, and demonstrate that incorporating completed sketches from the suggested cascaded GAN architecture results in improved retrieval performance.

© 2023 Elsevier B.V. All rights reserved.

1. Introduction

Advancements in the field of computer networks and the invention of the World Wide Web \cite{1} in the 90s led to the large availability of data on the internet. Given this large amount of data, finding relevant information for the given input query became a key topic of interest. Based on a user query, information retrieval systems are employed to find relevant information from documents, text, web pages, etc. The book on information retrieval by Kowalski et al. \cite{2} focuses on an overview of retrieval systems that mainly use text queries. Grossman et al. \cite{3} demonstrated various retrieval algorithms for finding pertinent documents to user queries; in addition, their run-time performance was evaluated and compared. Initially, the main mode of the query was text, owing to users’ desire to find relevant information from the web using text queries, which led to the research and development of efficient search engines such as \cite{4,5}. Although useful, text-based queries became impractical to retrieve other forms of data like images, videos, audio, etc.
(recent developments like CLIP [6] model can correlate image semantics with text, leading to state-of-the-art performance on retrieval and classification tasks). This was mainly due to no correlation between text queries and Image semantics. As a result, content-based retrieval systems were created, which deal with non-text queries like images, video, and audio and retrieve relevant images from databases based on those image queries.

Various approaches have been proposed for Content-Based Image Retrieval (CBIR). Query by Image and Video Content (QBIC) system [7] used features like shape, color, and texture and compared these features with features of images already present in the database and computed a similarity-based ranking. Andritsos et al. [8] proposed a recursive HSV-Space segmentation technique to locate important colour areas in the image and convert them into a vector feature which was used for ranking similar images. Similar systems have also been developed for video retrieval. By segmenting video clips in shorter frames, matching histograms of frame images [9] developed a video parsing, indexing, and retrieval framework.

3D models of generic shapes like chairs, planes, etc., are used significantly in the domain of computer graphics. 3D models for engineering domains are vastly different like pistons, cylinders, etc. Retrieval of relevant 3D CAD models becomes crucial when retrieving relevant shapes from a large database. Text-based techniques like those mentioned in [10] cannot be applied to 3D models since such models don’t have semantic pairing between the images and text queries. Text-queries can be applied to 3D model retrieval using CLIP [6] based model if there exists an annotated dataset pairing texts with CAD models, which, according to our knowledge, is not available for engineering as well as for generic shapes. Image-based retrieval techniques like [7,8] can be applied to 3D models, but this will usually perform poorly since these techniques compute similarity based on the histogram of the query image and the images in the database. For images of 3D shapes, the color distribution is uniform; hence, using histograms for similarity indexing instead of shape information like bends, holes, curves etc., will give poor results.

Directly using the 3D model as a query is also not feasible since this requires domain knowledge [10]. Using a 2D representative sketch for a 3D CAD model is a feasible alternative since drawing sketches of 3D objects is much easier than handling 3D CAD models directly as queries [11]. Methods like [12,13] use multiple views of 3D models expecting one of the views will match with that of the query sketch and perform ranking based on local features present, which are generated by techniques like SIFT [14], GALIF [15]. The view-based approach proposed in [10] used Siamese networks [17] to generate feature vectors which were then compared to create a ranking of relevant 3D models from a database.

Significant research has already been done for retrieval of 3D models in the domain of generic shapes, mainly due to the availability of open-source 3D model datasets like PSB [18]. ModelNet [19]. Research on retrieval of 3D models of engineering shapes is limited, mainly due to the unavailability of large-scale datasets for 3D CAD models of engineering shapes. A dataset of 800 CAD models of engineering shapes was proposed in Engineering Shapes Benchmark (ESB) [20]. Recently large-scale datasets like ABC [21], with a million CAD models of engineering shapes and Mechanical Components Benchmark (MCB) [22] with 58000 CAD models and CADNET [23] have been published. Recently, a sketch dataset of 801 CAD models from ESB dataset was proposed in SketchCleanNet [24]. This dataset was further used to improve the performance of the retrieval system in CADSketchNet [25].

A retrieval system’s performance is highly dependent on the quality of the query it’s provided with. Computer-generated sketch queries of ESB [20] and MCB [22] CAD models, proposed in [25] have defects like missing lines, overdrawn lines, missing portions, mesh lines, etc., which reduces the performance of a retrieval system. SketchCleanNet [24] tried to rectify this computer-generated query sketches using a Fully Convolutional Network [26].

It is important to note that engineering shapes are an important part of any design workflow. For analyzing these engineering shapes, they need to be retrieved from a large database where they are stored. Manually searching for particular engineering shapes will be time-consuming and becomes even more complex if the database has many engineering shapes. Hence, building a retrieval system for CAD models of engineering shapes is crucial.

It is highly possible that the end user may not have domain knowledge about engineering shapes to be retrieved, although they may have a partial idea of how the sketch of the CAD model might be. In such cases, the user may provide a representative partially drawn sketch as a query to the retrieval system. This sketch will lack important features, such as lines, circles, holes, ovals, slots, bends, and other inner salient features. Retrieving relevant engineering shapes from such partially drawn query sketches is highly unlikely. This is mainly due to shape dissimilarity between the input incomplete query sketch and the relevant engineering shape in the database. In such cases, retrieval performance will be largely affected, as shown in Section 6.3.

This paper introduces a new type of Generative Adversarial Network (GAN) [26] called SketchCADGAN. It uses a two-stage cascaded approach to fill in missing parts of a sketch provided as input query. The first stage of the network uses a generator network to predict an image of a CAD model from the incomplete sketch, while the second stage generator network takes the output of the first stage network i.e., a completed CAD model image and attempts to convert it into a completed sketch. Both stages of the network are trained adversarially. In addition, the paper proposes two unique local discriminators that assist the cascaded generators in completing incomplete areas in the partially drawn input sketch. Further, in sections 6.1 and 6.3, we offer sufficient qualitative and quantitative evidence to demonstrate the significance of the auxiliary task of completing CAD model images, which enhances the performance of sketch completion. Moreover the use of two separate discriminators for each stage, which focuses on a 160 × 160 patch around the missing portion improves the inner salient features generated in the completed sketch as well as completed CAD model image.
The key contributions of the paper are:

- Perhaps the first network for completing incomplete query sketches of engineering shapes.
- A novel two-stage generator architecture, where the auxiliary task of CAD model image completion in the first stage improves sketch completion in the second stage.
- Rather than using the complete prediction from the generator, the discriminator in SketchCADGAN utilizes a 160 × 160 patch around the missing portion, which generates more salient inner and outer features.
- A novel training strategy involving two cascaded generators trained adversarially with two separate discriminators for each stage.

2. Related Works

There has been limited research conducted on the topic of sketch completion even on sketches of generic shapes. To the best of our knowledge, no research has been done on the completion of sketches for engineering shapes.

**Sketch completion of vector sketches:** SketchRNN [28] utilized a method based on Variational Auto-Encoder (VAE) for completing sketches, where the encoder was a bidirectional Recurrent Neural Network (RNN) that took a sequence of three pen states and x, y offsets as input. These were fed into the encoder network sequentially, and an Auto-Regressive RNN predicted future pen states for sketch completion. Sketchpix2seq [29] replaced the RNN encoder with a Convolutional Neural Network (CNN), taking pixel images as input and still generating a sequence of pen states as output. SketchLattice [30] followed a similar approach but used a lattice representation. It’s worth noting that all of these methods require sequential data with pen states, making them unsuitable for completing pixel sketches.

**Sketch completion of pixel sketches:** According to relevant sources, SketchGAN [31] is the only network that specifically addresses the issue of completing pixel sketches. SketchGAN uses a three-stage cascaded network architecture based on a Conditional-GAN [32], which treats sketch completion as a sequential task. Additionally, SketchGAN demonstrated that performing an auxiliary sketch recognition task can improve the performance of the sketch completion task.

**Image inpainting methods:** Sketches lack the texture and colour distribution of images. This means that filling in missing parts of sketches with image inpainting or completion methods will not work well. This is due to the fact that these methods rely on neighbouring pixels to predict missing ones, but sketches have a sparse pixel distribution with many white pixels, resulting in incomplete or blank areas, as shown in Figure 1 and evident from the quantitative comparison in Tables 1 and 2.

We include a few of the image-inpainting works that can be applied to sketches here for completeness. Bertalmio et al. [33] used the concept of vorticity and gradients from the field of fluid dynamics to inpaint images.

The Context Encoder [34] used GANs [27] to predict missing information in the centre part of the image. Yang et al. [35] improved this by removing blurriness from the reconstructed image.

GLCIC [36] used local and global discriminators instead of a single discriminator, resulting in more realistic inpainting results. Similar work with modifications in network architecture was proposed by Demir et al. [37]. Shift-Net [38] combined traditional copy-paste approach, with learning-based approach using CNN, to predict missing pixels.

Generative Image Inpainting [39] used a two-stage coarse-to-fine network with contextual attention to produce non-blurry and realistic inpainting results.

3. Dataset Generation

The sketch completion network requires an incomplete sketch as input and it is expected to predict a completed sketch as output. Thus, a dataset containing pairs of incomplete and corresponding complete sketches is necessary for using learning-based techniques for sketch completion. Unfortunately, there is no such paired dataset available for CAD model sketches, so we need to generate incomplete sketches by randomly masking a portion of a pre-existing, completed hand-drawn sketch. Additionally, there are no available sketch datasets for ABC [40] and MCB [22] CAD models except for the ESB [20] dataset proposed in SketchCleanNet [24]. Therefore, we first need to create a dataset of complete CAD model sketches, which will then be randomly masked to simulate incomplete input sketches that will be fed into SketchCADGAN.
4. Network Architecture

4.1. Generative Adversarial Networks (GANs)

GAN \[27\] uses two adversarial models, a generative model generator(G) and a classifier discriminator(D), and are trained using an adversarial loss proposed in \[27\]. This adversarial loss makes G generate realistic samples. Conditional-GANs (C-GANs) \[48\], unlike GANs \[27\] are conditioned on auxiliary information like class labels, image masks \[49\], etc. This conditionality adds an extra constraint over how the generated samples should be.

Fig. 2. CAD model snapshots (row 1) and corresponding sketches (row 2) generated by finetuning PhotoSketching \[43\] followed by sketch simplification \[44\], (row 3) represents masked sketches that act as input to SketchCADGAN of SketchCADGAN, we chose to train it on sketches from three different CAD model datasets: ESB \[20\], MCB \[22\], and ABC \[40\]. SketchCleanNet \[24\] provided hand-drawn sketches of ESB \[20\] dataset CAD model images; therefore we train SketchCADGAN for ESB \[20\] dataset using dataset proposed in SketchCleanNet \[24\]. CADSketchNet \[25\] presented a dataset of sketches for CAD model images from the MCB dataset, which was generated using the Canny \[45\] edge-detection algorithm. However, since the user query for SketchCADGAN training is a hand-drawn sketch, this dataset is not suitable.

There is currently no paired dataset of CAD-model images and corresponding hand-drawn sketches available for the ABC dataset, to the best of our knowledge. While recent deep learning-based edge-detection methods \[42\], \[46\], \[41\] could be used to generate sketches for CAD models from both the MCB and ABC datasets, they detect unnecessary details such as mesh lines and do not accurately represent the qualities of a hand-drawn sketch. To address this, we decided to draw sketches for the MCB dataset manually. In order to generate CAD model images for the MCB dataset, a snapshot of each CAD model was taken from an angle that shows the maximum information present in the CAD model. Since the MCB dataset contains 58,000 images of different CAD models, drawing all of them manually is highly time-consuming. Therefore, we selected representative CAD model images from all 68 classes present in the MCB dataset. A group of students with domain knowledge of mechanical components were tasked with selecting representative images. Additionally, they were asked to ensure that the images they select capture most of the variations present within the class and across the whole dataset. Finally, we generate a dataset of 534 hand-drawn MCB sketches by tracing CAD model images using a tablet device. Figure 1 shows few examples, where the sketch traced is better than other techniques, such as edge detection \[41\], \[42\], or computer generated \[25\].

Training GANs for smaller datasets requires a lot of fine-tuning to achieve stable performance. If the number of training examples is less, then there can be problems like mode collapse \[27\]: the discriminator might become really good and detect fake images easily, leading to bad generator performance. Therefore, for large CAD model datasets like MCB and ABC, training SketchCADGAN with just 534 sketches would lead to poor performance on test data. Hence, we provide an alternative method for generating more paired training examples using the same 534 hand-drawn sketches. We use the same network proposed by Lips et al. \[43\] with the input being the 434 CAD model snapshots and ground-truths being the corresponding hand-drawn sketch. We keep 100 images for testing the performance of the proposed approach on unseen data. We use pre-trained weights from \[43\] to fine-tune the network instead of training it from scratch, as we observed that performance is much better with pre-trained weights. Figure 2 shows the CAD model images in row 1 and corresponding predicted sketches in row 2. The first four columns demonstrate results on ABC \[40\] dataset, while the last 4 columns demonstrate results on MCB \[22\] dataset. It is to be noted that results shown in Figure 2 are on unseen data and were not used to finetune Photosketching with sketch simplification. Row 3 in Figure 2 represents the masked input sketches that act as input to our SketchCADGAN network. Further, by using the proposed approach of finetuning Photosketching with sketch simplification, we generate 10000 sketches for the ABC dataset with 10000 corresponding CAD model images. For training SketchCADGAN on the ABC dataset, we use 8000 sketches and keep 2000 sketches for testing. Similarly, for training SketchCADGAN on the MCB dataset, we generate 9905 training examples across 68 classes and keep 4007 images for testing.
Fig. 3. SketchCADGAN network architecture consists of a cascaded generator module and a discriminator module, generator module consists of two cascaded generators while the discriminator module consists of two separate discriminators.

Fig. 4. White patches generated at location of missing regions by image-inpainting method Deep-FillV2 [50](column 2) and sketch completion method SketchGAN [31](column 3).

4.2. SketchCADGAN Architecture

SketchCADGAN aims to predict missing details in partially drawn input sketches. However, identifying plausible details in the missing region is a difficult task, even for humans with domain knowledge. Pix2Pix [51] and similar image-to-image translation networks are effective at converting sketches into colored images because of the significant contrast between the input (sketch) and output (colored image), allowing the generator to create more realistic images that can deceive the discriminator. However, when it comes to completing sketches, these networks tend to produce white patches in the missing areas, as shown in Figure 4. (See Section 6 for more information). This is because the input incomplete sketch and the complete sketch are identical except for the missing portion, which results in the generator copying the input as output to deceive the discriminator. As a consequence, the generator’s gradients become weaker during updates leading to the white-patched prediction as seen in Figures 4 and 6. Image-inpainting methods such as GLCIC [36] use texture from neighbouring regions to predict the missing areas. However, since sketches lack the texture present in colored images, the resulting predictions from these methods can appear blurry, as illustrated in column 3 of Fig. 7.

The SketchCADGAN architecture addresses the limitations of image-to-image translation networks and image inpainting methods. It uses a novel two-stage cascaded strategy (Figure 3), where the first stage predicts a completed CAD model image from an incomplete sketch, which is then sent to the second stage to predict missing portions of the incomplete sketch. The auxiliary task of predicting the completed CAD model image in the first stage helps in predicting the missing portions present in the input incomplete sketch as shown in Figure 5.

We use two separate discriminators (shown in Figure 3) for each stage of the generator block, which are trained alongside the generator networks. Image inpainting methods like GLCIC [36] and Generative image inpainting [39] use local and global discriminators to generate consistent local textures. Using local and global discriminators for non-textured images like sketch results in a uniform white patch. This is mainly because there is no sufficient texture distribution available in the neighborhood of the missing portion. This suppresses salient features that are present in the CAD model sketches as shown in column 5 of Figure 6. Therefore, instead of separating local and global discriminators, we crop a 160 x 160 patch around the missing portion from the output of the respective stage. This patch acts as input to the discriminator model. Both of our discriminator architectures follow similar architecture as PatchGANs [51]. Cropping a larger patch near the missing portion does the job of local and global discriminator without producing uniform white textures.

The key points separating SketchCADGAN from other
image-to-image translation and image inpainting methods are:

- SketchCADGAN uses a novel two-stage cascading strategy, where CAD model completion in the first stage improves sketch completion in the second stage. The addition of CAD model image completion in the first stage solves the copy-paste problem present in image-to-image translation networks like Pix2Pix [51].

- Concept of local and global discriminators [50] works well for textured images, but this strategy leads to uniform white patches in textureless images. SketchCADGAN uses separate discriminators for each stage. Each discriminator takes 160 x 160 patch surrounding the missing portion as input. Therefore, local and global consistency is maintained with a single discriminator instead of two separate discriminators per stage.

**Generator Module Architecture:** The generator module in the network architecture diagram (Figure 2) comprises two linked generators. Both of these generators have the same architecture. Each generator has an encoder-decoder style configuration, where the encoder part consists of 8 down-convolution layers. Except for the first Down-Convolution, the remaining seven down-convolution layers are followed by InstanceNorm [52] and Leaky-Rectified Linear Unit (LeakyReLU) [53] activation with a negative slope of 0.2. Instance normalization [52] was preferred over batch normalization [54] for SketchCADGAN because it offered greater stability during training, and the use of LeakyReLU in the encoder layer was inspired by Pix2Pix.

The generator’s decoder part has eight Up-Convolutions, which are Transpose-Convolutions [55]. Following each Up-Convolution operation, there is a concatenation with the corresponding Down-Convolution block found in the encoder part of the network. This concatenation is similar to U-net [56]. After concatenation, the output is passed through Rectified Linear Unit(ReLU) activation. After eight Up-Convolutions and corresponding concatenations, the final output goes through a hyperbolic-tangent(Tanh) activation function. A kernel size of 4x4, a stride of 2x2, and zero-padding of 1x1 are used for both the encoder and decoder parts of the cascaded generator network architecture.

**Discriminator Module Architecture:** The input to the cascaded generator block is an image of an incomplete sketch of size 256 x 256. This image is first transformed into a complete CAD model image and further into a complete CAD model sketch (as shown in Fig. 3). The respective outputs from each stage of the generator block are sent to separate discriminators. We pre-process the output from each stage of the generator, where we crop a 160 x 160 patch neighbouring the missing portion during training. This gives the benefit of maintaining separate local and global discriminators like [50] into a single discriminator. It also solves the problem of uniform texture generated by the local discriminator. Each discriminator in this module consists of four down-convolution operations. Except for the first and the last down-convolution, the remaining down-convolution operations are followed by instance normalization and RELU activation. The final output is of size 38 x 38. The effective receptive field size of the output layer is 34 x 34, i.e., each position in 38 x 38 output represents a corresponding 34 x 34 patch in an input cropped image of size 160 x 160. After testing different sizes of receptive fields using a different number of down-convolution layers, we found performance with a receptive field size of 34 x 34 more stable. Except for the last down-convolution, others use a kernel size of 4 x 4, stride of 2 x 2 and padding of 1 x 1. The last down-convolution uses a kernel size of 4 x 4, stride of 1 x 1 and padding of 1 x 1. The output is further passed through a final sigmoid activation to bring all the outputs in 38 x 38 grid to the range 0 to 1.

5. Training Strategy, Loss function, Mask shape and Size details

5.1. Mask size and shape details for training

Since there exists no dataset that pairs an incomplete sketch with a complete sketch, therefore we randomly mask different regions in sketches that were generated using the method explained in section 3. During training, we specifically keep a single rectangular masked region with sizes ranging between 96 and 128 along the width and height dimensions i.e. during training, the minimum masked region size is 96 x 96 while the maximum masked region size is 128 x 128.

Let Y represent the initial unmasked sketch from a dataset of ESB, ABC, or MCB, and M represent a random rectangular binary mask matrix, which has the same size of 256 x 256 as Y with pixels in the missing region set to 1 while rest of the pixels set to 0; then our input incomplete sketch can be obtained using the equation.

\[ X = Y - Y \circ M + M \] (1)

5.2. Loss function Details

Let \( G_1 \) and \( G_2 \) represent two generators in the Generator block module. Let \( \theta \) and \( \phi \) be their respective parameters, and let \( D_1 \) and \( D_2 \) represent two discriminators in the Discriminator block module. Let \( \omega \) and \( \gamma \) be their respective parameters.

It is important to note that we only condition the generator of SketchCADGAN by passing an incomplete input sketch, but the discriminator is unconditioned, i.e., we don’t condition the discriminator on the generator’s input. This is mainly done since conditioning the discriminator on the generator’s input makes it easier for the discriminator to identify fake images leading to poor performance while training.

Let \( \hat{Z} \) represent the ground truth auxiliary CAD model sketch, then the forward pass of SketchCADGAN is as follows.

\[ \hat{Z} = G_1(X) \] (2)

\[ \hat{Y} = G_2((\hat{Z} + 1)/2) \] (3)

We use two loss function objectives for SketchCADGAN training, namely reconstruction loss and adversarial loss. The reconstruction loss is the standard L1 norm or Mean Absolute
Error between generator prediction and ground truth, while adversarial loss treats the generator and discriminator as adversaries. The L1 loss for outputs of $G_1$ and $G_2$ is as follows:

$$Rec_1 = ||\hat{Z} - Z||_1$$

(4)

$$Rec_2 = ||\hat{Y} - Y||_1$$

(5)

The adversarial loss for generators $G_1$ and $G_2$ is as follows:

$$G_{adv1} = -\sum \log(D_1(\hat{Z}))$$

(6)

$$G_{adv2} = -\sum \log(D_2(\hat{Y}))$$

(7)

The adversarial loss for discriminators $D_1$ and $D_2$ is as follows:

$$D_{adv1} = -\sum \log(D_1(Z)) + \log(1 - D_1(\hat{Z}))$$

(8)

$$D_{adv2} = -\sum \log(D_2(Y)) + \log(1 - D_2(\hat{Y}))$$

(9)

The final objectives for Generator and Discriminators are as follows:

$$G = \min_{\theta, \phi} G_{adv1} + G_{adv2} + \lambda_1 * Rec_1 + \lambda_2 * Rec_2$$

(10)

$$D = \min_{\omega, \gamma} \lambda D_1 * D_{adv1} + \lambda D_2 * D_{adv2}$$

(11)

where we set hyperparameters $\lambda_1 = \lambda_2 = 100$ and $\lambda D_1 = \lambda D_2 = 0.5$. Equations (10) and (11) are the final minimization objectives for generator module $G$ and discriminator module $D$ respectively.

5.3. Training Strategy

GANs can become unstable during training due to adversarial training; if the discriminator becomes too good at identifying completed images from the generator, then the generator’s performance will become poor, and vice-versa. Therefore, we provide a specific strategy for training SketchCADGAN. We train the generator module first for $t$ iterations using only the reconstruction losses mentioned in equations (4) and (5). This is done to avoid discriminators becoming too good at the start of the training itself. The rest of the training strategy is explained in pseudocode [1]. We set $T = 30000, t = 1000$ iterations for ESB [20] dataset training and $T = 100000, t = 20000$ for ABC [40] and MCB [22] dataset training. We use adam optimizer for training generator as well as the discriminator module with a learning rate of 0.0002 and momentum parameters $\beta_1 = 0.5$ and $\beta_2 = 0.999$. The training time for ESB [20] dataset is 1.2 hours, and 8 hours for MCB [22] and ABC dataset [40]. The training of SketchCADGAN is carried out using Pytorch 1.10 using a batch size of 8 on a machine with NVIDIA RTX 3080Ti GPU, 128GB RAM, i9-10900F processor.

6. Results

6.1. Ablation study for Cascaded Generator Module

Figure 5 displays the results of an ablation study that emphasizes the importance of using CAD model image completion as an additional task when conducting sketch completion. The second column shows the output of a generator model trained only for sketch completion using L1 loss and without performing auxiliary CAD model image completion. This resulted in blurry predictions since the network tried to minimize the average L1 loss over the entire missing region. The third column shows the output of a network that was trained only for sketch completion using adversarial loss without performing auxiliary CAD model image completion. This network faced the problem of mode collapse, as shown in rows three to six of Figure 5. This happened due to the absence of reconstruction loss like L1 loss. When sketch completion was done directly by combining L1 loss and adversarial loss without performing image completion, it resulted in a copy-paste problem, where the input was directly copied as the output. This is shown in Column 4 of Figure 5. The fifth column shows the results of the SketchCADGAN network, which was trained using a loss function that combines both sketch completion and image completion tasks (equations 10 and 11). The learned shapes are similar to those in the ground truths (Column 6) but are not exactly the same. This demonstrates the ability of SketchCADGAN to generate new, similar-looking shapes. The auxiliary CAD image completion results in the first stage of SketchCADGAN are shown in (Column 7), while the ground truth CAD model images are shown in (Column 8).

6.2. Ablation study for Discriminator Module

Image inpainting methods like GLCIC [36] and generative image inpainting [39] use global and local discriminators to determine whether the generated image is real or fake. The local discriminator ensures that the generated image is consistent with the overall image semantics, which is important for textured images. However, for images without texture, like sketches, most of the neighborhood pixels around the miss-

Algorithm 1: Training Strategy for SketchCADGAN

for $Y, Z$ in data do

1. Loop over data

2. $X \leftarrow Y \cdot Y \cdot M \cdot M$ → generate masked image

3. $\hat{Z} \leftarrow G_1(X)$ → Stage-1 image completion

4. $\hat{Y} \leftarrow G_2(\hat{Z} + 1)/2$ → Stage-2 sketch completion

if iteration < $t$ then

1. $\nabla_1 \leftarrow Rec_1.backward()$ → backpropagation for $G_1$

2. $\nabla_2 \leftarrow Rec_2.backward()$ → backpropagation for $G_1$ and $G_2$

3. $\theta \leftarrow \theta - \alpha * (\nabla_1 + \nabla_2)$ → Update $G_1$

4. $\phi \leftarrow \phi - \alpha * (\nabla_2)$ → Update $G_2$

else

1. $\nabla_D \leftarrow D.backward()$ → Backpropagation for $D_1$ and $D_2$ using loss mentioned in equation (11)

2. $\omega \leftarrow \omega - \alpha * \nabla_D$ → Update $D_1$

3. $\phi \leftarrow \phi - \alpha * \nabla_D$ → Update $D_2$

4. $\nabla_G \leftarrow G.backward()$ → Backpropagation for $G_1$ and $G_2$ using loss mentioned in equation (10)

5. $\theta \leftarrow \theta - \alpha * \nabla_G$ → Update $G_1$

6. $\phi \leftarrow \phi - \alpha * \nabla_G$ → Update $G_2$
Fig. 5. Ablation study showing the importance of cascaded generator module of SketchCADGAN, first two rows demonstrate results on ESB [20] dataset sketches, next two rows on ABC [40] dataset sketches and last two rows on MCB [22] dataset sketches.

6.3. Qualitative and quantitative comparison with other image inpainting and sketch completion methods.

We evaluate the effectiveness of SketchCADGAN by comparing it to other state-of-the-art methods, [36, 39, 38, 51, 31], both qualitatively and quantitatively. However, since our SketchCADGAN network is a generative network, its output may not be exactly the same as the ground truth. Therefore, using pixel-wise similarity metrics such as mean squared error or mean absolute error may not be the best approach to judge the quality of our method. Instead, we train a classification network using original complete sketches, specifically the ESB and MCB datasets, with the number of neurons in the last fully connected layer adjusted to the number of classes in the ESB and MCB datasets, which are 42 and 68, respectively. We train the network with 641 ESB sketches and reserve 160 images for testing. For the MCB dataset, we train with 8000 sketches and test the network on 2000 sketches. In order to compare the performance...
Fig. 6. Ablation study showing importance of Discriminator module of SketchCADGAN over local and global discriminators from image-inpainting methods, the results presented above are on MCB sketches

of SketchCADGAN with image-inpainting, sketch completion, and image-to-image translation networks, we pass the model predictions from respective methods to the by predictions. A good completion network should produce similar shapes to the ground truths, leading to higher accuracy, precision, and F1-score when evaluated using an already trained VGG network on the completed sketches. Figure 7 makes a qualitative comparison of sketch completion results achieved by different image-inpainting [36, 39, 38, 50, 58], image-to-image translation [51], and sketch completion methods [31], it can be seen from 7 that SketchCADGAN can generate similar shapes(column 8) when compared to ground truth(column 9). Due to the generative nature of SketchCADGAN, the generated shapes are not exactly the same as the ground truth, which shows the generative ability of SketchCADGAN. The first two rows in Figure 7 show results on ESB dataset sketches, rows 3 to 6 show results on MCB dataset sketches, while the last 4 rows show results on the ABC dataset. Table 1 shows the quantitative performance of different models on ESB [20] sketches, Table 2 shows a similar comparison on MCB [22] dataset sketches, since the VGG [59] classification model was trained on already complete ground truth sketches, a good accuracy, precision, and F1-score in Tables 1 and 2 by a sketch completion method would signify a better sketch completion, therefore from Tables 1 and 2 it’s clear that SketchCADGAN is able to do much better sketch completion when compared with other methods. It’s important to note we only provide qualitative results for ABC [40] dataset sketches since it does not have class labels for any of the CAD models. Therefore a classification network can not be trained for ABC dataset sketches. The results for SketchGAN [31] are our implementation according to the details mentioned in SketchGAN. This is due to no official implementation of SketchGAN available on the internet, SketchGAN may work well for smaller missing regions, but it’s clearly unable to generate any shapes, as shown in column 7 of Figure 7.

6.4. Retrieval performance

CADSketchNet [25] proposed a method to retrieve relevant CAD models based on computer-generated query sketches; we use the same approach presented in CADSketchNet to create a retrieval system that retrieves relevant CAD models from the database based on hand-drawn query sketches. Specifically, we train the siamese network presented in CADSketchNet with Resnet-18 backbone separately on ESB [24] as well as MCB hand-drawn sketches and corresponding CAD model images. We don’t train on retrieval system on ABC [40] sketches since the ABC dataset is an unclassified dataset, and training
Fig. 7. Qualitative comparison of SketchCADGAN (column 8) with image inpainting methods like Shift-Net [38] (column 2), GLCIC [36] (column 3), Deep-Fill V2 [50] (column 4), Partial-Convolution [58] (column 5), image-to-image translation method like Pix2Pix [51] (column 4) and sketch completion method [31] (column 7)
Table 1. Comparison of the VGG classification network’s performance on test sketches from the ESB [20] dataset, using various image-inpainting, sketch-completion, and image-to-image translation networks to generate completed sketches, it is apparent that SketchCADGAN exhibits the highest levels of accuracy, precision, and recall when compared to other methods. The baseline accuracy represents the accuracy of the original complete ground truth sketches.

<table>
<thead>
<tr>
<th>Models</th>
<th>Accuracy</th>
<th>Precision</th>
<th>F1-Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline (original unmasked sketch)</td>
<td>0.8310</td>
<td>0.7429</td>
<td>0.8251</td>
</tr>
<tr>
<td>GLCIC [36]</td>
<td>0.5535</td>
<td>0.4379</td>
<td>0.5830</td>
</tr>
<tr>
<td>SketchGAN [31]</td>
<td>0.5369</td>
<td>0.3774</td>
<td>0.5042</td>
</tr>
<tr>
<td>Pix2Pix [51]</td>
<td>0.5136</td>
<td>0.3188</td>
<td>0.2820</td>
</tr>
<tr>
<td>Shift-Net [38]</td>
<td>0.6132</td>
<td>0.5642</td>
<td>0.6156</td>
</tr>
<tr>
<td>Deep-FillV2 [50]</td>
<td>0.4991</td>
<td>0.2160</td>
<td>0.3771</td>
</tr>
<tr>
<td>Partial Convolution [58]</td>
<td>0.4469</td>
<td>0.1967</td>
<td>0.3583</td>
</tr>
<tr>
<td>SketchCADGAN</td>
<td>0.7132</td>
<td>0.6651</td>
<td>0.6938</td>
</tr>
</tbody>
</table>

Table 2. Comparison of the VGG classification network’s performance on test sketches from the MCB [22] dataset, using various image-inpainting, sketch-completion, and image-to-image translation networks to generate completed sketches, it is apparent that SketchCADGAN exhibits the highest levels of accuracy, precision, and recall when compared to other methods. The baseline accuracy represents the accuracy of the original complete ground truth sketches.

<table>
<thead>
<tr>
<th>Models</th>
<th>Accuracy</th>
<th>Precision</th>
<th>F1-Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline (original unmasked sketch)</td>
<td>0.9146</td>
<td>0.8429</td>
<td>0.8911</td>
</tr>
<tr>
<td>GLCIC [36]</td>
<td>0.5813</td>
<td>0.4585</td>
<td>0.6145</td>
</tr>
<tr>
<td>SketchGAN [31]</td>
<td>0.5219</td>
<td>0.2994</td>
<td>0.3697</td>
</tr>
<tr>
<td>Pix2Pix [51]</td>
<td>0.5127</td>
<td>0.3587</td>
<td>0.3123</td>
</tr>
<tr>
<td>Shift-Net [38]</td>
<td>0.7279</td>
<td>0.6576</td>
<td>0.7116</td>
</tr>
<tr>
<td>Deep-FillV2 [50]</td>
<td>0.5317</td>
<td>0.3198</td>
<td>0.4965</td>
</tr>
<tr>
<td>Partial Convolution [58]</td>
<td>0.4869</td>
<td>0.2967</td>
<td>0.2696</td>
</tr>
<tr>
<td>SketchCADGAN</td>
<td>0.8372</td>
<td>0.7510</td>
<td>0.8014</td>
</tr>
</tbody>
</table>

Tables 3 and 4 represent the average top 10 accuracy of the retrieval system by querying sketches generated by different image-inpainting, sketch completion and image-to-image translation methods. Top 10 accuracy represents average number of correct results in among top 10 retrieved results. It is evident from the retrieval accuracy of SketchCADGAN that it is able to provide better query sketches compared to other methods, hence proving that sketch completion results by SketchCADGAN are better than rest of the methods shown in the Tables 3 and 4.

Table 3. Top 10 retrieval accuracy for query sketches generated by different methods on ESB dataset, completed sketches generated by SketchCADGAN act as a better query compared to other image inpainting, sketch completion and image-to-image translation methods.

<table>
<thead>
<tr>
<th>Method</th>
<th>Top 10 retrieval accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline (trained on ground truth complete sketches)</td>
<td>0.94</td>
</tr>
<tr>
<td>GLCIC [36]</td>
<td>0.64</td>
</tr>
<tr>
<td>SketchGAN [31]</td>
<td>0.58</td>
</tr>
<tr>
<td>Pix2Pix [51]</td>
<td>0.55</td>
</tr>
<tr>
<td>Shift-Net [38]</td>
<td>0.75</td>
</tr>
<tr>
<td>Deep-FillV2 [50]</td>
<td>0.51</td>
</tr>
<tr>
<td>Partial Convolution [58]</td>
<td>0.46</td>
</tr>
<tr>
<td>SketchCADGAN</td>
<td>0.86</td>
</tr>
</tbody>
</table>

Table 4. Top 10 retrieval accuracy for query sketches generated by different methods on MCB dataset, completed sketches generated by SketchCADGAN act as a better query compared to other image inpainting, sketch completion and image-to-image translation methods.

<table>
<thead>
<tr>
<th>Method</th>
<th>Top 10 retrieval accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline (trained on ground truth complete sketches)</td>
<td>0.97</td>
</tr>
<tr>
<td>GLCIC [36]</td>
<td>0.47</td>
</tr>
<tr>
<td>SketchGAN [31]</td>
<td>0.52</td>
</tr>
<tr>
<td>Pix2Pix [51]</td>
<td>0.49</td>
</tr>
<tr>
<td>Shift-Net [38]</td>
<td>0.65</td>
</tr>
<tr>
<td>Deep-FillV2 [50]</td>
<td>0.41</td>
</tr>
<tr>
<td>Partial Convolution [58]</td>
<td>0.35</td>
</tr>
<tr>
<td>SketchCADGAN</td>
<td>0.83</td>
</tr>
</tbody>
</table>

7. Conclusion

Retrieving relevant CAD models from a database using sketch queries is easier for an end user. However, the end user may input only a partially drawn sketch query as input to the system. These partially drawn sketches need to be completed in order to get better retrieval results from an existing retrieval method.
system. In this paper, we presented a generative adversarial network to predict missing portions present in incomplete query sketches of CAD model sketches using a novel two-stage cascaded generator module. The first stage did an auxiliary task of CAD model image completion, while the second stage converted the completed CAD model image into a CAD model sketch. This novel approach has been demonstrated to help improve sketch completion significantly. Furthermore, instead of using local and global discriminators, we used only a single discriminator for each stage of cascaded generator module. Also, instead of passing the entire prediction by the generator to discriminator, using a 160 × 160 patch surrounding the missing region generated much better inner and outer salient features. This overall approach was also shown to be effective using ablation studies. A detailed quantitative and qualitative comparison of SketchCADGAN with other methods showed that SketchCADGAN significantly outperformed other methods.

References


