RESEARCH ARTICLE

Generating 3D texture models of vessel pipes using 2D texture transferred by object recognition

Min-Ji Kim, Kyung-Ho Lee*, Young-Soo Han, Jaejoon Lee and Byungwook Nam

1 DSME Information & Consulting Co. Ltd., CAD development department, Geoje, Gyeongsangnam-do 53302, South Korea and 2 Department for Naval Architecture & Ocean Engineering, INHA University, 22212 Incheon, South Korea

Abstract

Research and development of smart vessels has progressed significantly in recent years, and ships have become high-value technology-intensive resources. These ships entail high production costs and long-life cycles. Thus, modernized technical design, professional training, and aggressive maintenance are important factors in the efficient management of ships. With the continuing digital revolution, the industrial shipbuilding applicability of augmented reality (AR) and virtual reality (VR) technologies as well as related 3D system modeling and processes has increased. However, resolving the differences between AR/VR and real-world models remains burdensome. This problem is particularly evident when mapping various texture characteristics to virtual objects. To mitigate the burden and improve the performance of such technologies, it is necessary to directly define various texture characteristics or to express them using expensive equipment. The use of deep-learning-based CycleGAN, however, has gained attention as a method of learning and automatically mapping real-object textures. Thus, we seek to use CycleGAN to improve the immersive capacities of AR/VR models and to reduce production costs for shipbuilding. However, when applying CycleGAN’s textures to pipe structures, the performance is insufficient for direct application to industrial piping networks. Therefore, this study investigates an improved CycleGAN algorithm that can be specifically applied to the shipbuilding industry by combining a modified object-recognition algorithm with a double normalization method. Thus, we demonstrate that basic knowledge on the production of AR industrial pipe models can be applied to virtual models through machine learning to deliver low-cost and high-quality textures. Our results provide an on-ramp for future CycleGAN studies related to the shipbuilding industry.

Keywords: augmented reality; CycleGAN; ResNet; normalization; texture; 3D model

List of Symbols

- $L_{GAN}$: Generative adversarial loss
- $L_{CC}$: Cycle-consistency loss (reconstruct loss)
- $D_A$ or $D_B$: Discriminator A or B
- $G$: A function that translates from A to B
- $F$: A function that translates from B to A
- $A \sim P_{data(A)}$: Sampling data (A)
- $B \sim P_{data(B)}$: Sampling data (B)
- $\lambda$: Weighted value set by a user
- $L_{Const}$: Cycle-consistency loss (reconstruct loss)
- $d$: Difference value of data comparison

Received: 25 March 2020; Revised: 1 November 2020; Accepted: 3 November 2020

© The Author(s) 2021. Published by Oxford University Press on behalf of the Society for Computational Design and Engineering. This is an Open Access article distributed under the terms of the Creative Commons Attribution-NonCommercial License (http://creativecommons.org/licenses/by-nc/4.0/), which permits non-commercial re-use, distribution, and reproduction in any medium, provided the original work is properly cited. For commercial re-use, please contact journals.permissions@oup.com
Generating 3D texture

1. Introduction

1.1. Research background

As the research and development of smart vessels progresses, ships are becoming high-value technology-intensive resources. This entails high production costs and long-life cycles. Thus, early design as well as professional training and maintenance during production stages are important factors in efficiently managing a ship’s life cycle (Kim, Lee, Son, & Han, 2012; Montt, Kasinski, & Pietrzak, 2018). Recently, the shipbuilding industry has reduced the production costs of ships by applying interference checks and digital mock-ups as well as using 3D augmented reality (AR) and virtual reality (VR) models prior to the production stages (Kim, Park, & Ko, 2018; Cebolero & Sanchez, 2019). These technologies have also been increasingly considered for use in the operational stages (e.g. AR/VR maintenance support, training/education simulation, and navigation information systems; Hikida, Fukuto, Numano, Inaoka, & Nakato, 2010; Jakub & Andrej, 2018; Lee, Lee, & Nam, 2020). Many shipyards have attempted to reduce production costs by studying the installation positions and orders of pipes, enhancing AR pipe models using mobile devices prior to production, and applying these to actual manufacturing. The German Fraunhofer–Gesellschaft Research Institute studied the suitability of AR-matching technology for measuring the dimensions of installation sites before production by using a head-mounted AR helmet (Fraga-Lamas, Fernandez-carames, Blanco-Novoa, & Vilar-montesinos, 2018; 3Dmaritim, 2019). The Korea Register of Shipping developed a ship life-cycle-management system (SeaTrust_SLM) that manages the entire shipbuilding life cycle. They have also been developing AR/VR-related content, including 3D model-based inspection simulators and crew-training tools (Kil, Son, & Lee, 2018).

AR/VR technologies using 3D models allow interactions in extended space, and they connect physical and virtual spaces beyond the natural environment using a limited amount of information. This technology best meets the basic human need to communicate in multiple directions via channels that do not exist physically but can be experienced virtually (Ceruti, Marzocca, Liverani, & Bil, 2019). Therefore, AR/VR has been recently applied to the shipbuilding industry to ensure the immersion and presence of workers and to minimize uncertainty factors that may arise during production. For example, when digital twins (i.e. digital replicas of living or nonliving physical entities) and internet of things technologies are fused, the information collected from real objects can be shared, and the corrosion or failure of shipping equipment as measured by sensors can be visualized as objects in a virtual world. Thus, by using a 3D model with realistic textures for educational simulations, students and engineers can gain an immediate understanding (Chen, 2007; Sadik, 2018; Ke, Xing, Zhang, & Zuo, 2019).

AR/VR technologies provide users with a more vivid sense of the world. Although it is still considered a viewing experience, the information presentation is neither real nor a projection on a screen. The true AR/VR environment allows human interaction with virtual content and maximizes immersion in ways never before seen. Additionally, the scope of recent immersion experiences has expanded not only to the entertainment realm, but also to manufacturing industries, as mentioned (Fukuda, Yokoi, Yabuki, & Motamedi, 2019). For example, remote locations can be virtually visited for field-like experiences, and a variety of industrial accident cases can be visualized without real losses. Furthermore, pre-production products can be visualized, resulting in virtual contact with customers. In particular, this has significant advantages in the order-share production approach to high-value-added industries.

To make the simulations more realistic and immersive, considerable amounts of time and money must be invested into developing visual features (e.g. textures). However, because the input cost for creating realistic textures is high, and its applicability in industrial tasks is low, insufficient effort and research have been dedicated. Therefore, the 3D-model textures used in industry are different from those of the actual objects. They are typically monochrome and do not account for complex lighting, which is quite heterogeneous in reality. Thus, representing realistic textures is an effective method of increasing the utility of 3D models.

In particular, if the AR model is to be applied to an industry, production costs must be considered. This is exacerbated by the requirement of most texture and mapping activities to be carried out by hand or by using hands-on equipment. Such manual work is expensive, as is the high-specification equipment. These factors make implementation difficult in any industry.

The left-hand side of Fig. 1 displays the process by which the general AR is shown to a user. The process consists of recognizing and tracking an object, rendering it, and displaying it. During this process, the goal is to automatically map the texture generated by using a generative adversarial network (GAN) that has learned the actual texture in advance (i.e. Domain B) and applying it to the AR piping model (i.e. Domain A; Bernardini, Martin, & Rushmeier, 2001; Ovcharenko, Kazei, Peter, & Alkhalifah, 2019). The GAN structure, which uses unsupervised learning, consists of a generator and a discriminator. In this study, we add a direct-skip connection and double normalization to the GAN to learn the actual images and to enhance image recognition performance. The image recognition module serves to separate objects (i.e. pipes) from the background, and the discriminator compares the visualized object with the actual pipe and proceeds with learning. Finally, this is visualized to the user. In summary, this study proposes a deep-learning-based texture-transfer method that learns a number of textural images that do not require separate human and material resources and automatically maps them to 3D AR piping models.

2. Related Works

Previous studies on texturing were conducted using a variety of methods. Most of them sought to reconstruct textures using manual operations that directly defined textures or obtained textural information from real objects using hardware (Gaty, Ecker, & Bethge, 2016). These methods were inefficient, because they required specialized models that directly manipulated image characteristics. Recently, deep-neural-network-based GANs that extract and learn the characteristics of model feature spaces have gained considerable attention. The basic structure of GAN can be seen in Fig. 2. Using noise, the Generator generates a fake image and the fake image is tested by the discriminator that learned the actual image to see if it is similar to the actual image.
2.1. Existing texturing methods

Texturing-based studies have generally focused on rapid mapping using high-quality textures to elaborate fabricate texture images (Yang, Ohtake, & Suzuki, 2020). For this purpose, Alshawabkeh and Haala (2005) produced a real-texture model that applied lens distortion and color correction via a 3D laser scanner using light detection and ranging. Lininger (2012) proposed a method to extract similar photo textures from an image database and applied it to match the coordinates of real and 3D models. Laycock, Ryder & Day (2007) promulgated a method of omitting redundant texture maps by considering the characteristics of compressed data, and Heindl, Akkaladevi and Bauer (2016) proposed a study to reduce memory usage by decomposing high-quality color images into low-dimensional images by using a red–green–blue sensor. Buyukdemircioglu, Kocaman & Isikdag (2018) semantically converted data using 3D scanners for an entire city and textured them. This allowed the creation of realistic textures by adding parameters, such as building heights and topographical characteristics to the existing texture information. Because these studies were conducted based on the premise that people or equipment directly participated in the process of creating and texturing realistic objects, many researchers have focused on temporal reduction and texture sophistication.

Another study was conducted that investigated how the style of a texture image could be transferred to the desired image. Unlike the methods described above, this method did not require people or equipment to create textures. It only required texture images. In particular, the previous studies focused on creating photorealistic textures, whereas texturing via this method was mainly used to produce nonphotorealistic textures by dealing with various specific expressive styles, including color maps, cartoons, and drawings. Hertzmann, Jacobs, Oliver, Curless & Salesin (2001) proposed an image-processing framework that analogized images around adjacent pixels to assess painting styles. Harrison (2001) synthesized textures by repeatedly adding to and inspecting pixels of output images. Welsh, Ashikhmin & Mueller (2002) transferred the color values from color images to grayscale images by matching luminance and texture information. However, these methods required individual filters and depended on nonparametric
techniques, which required the manipulation of pixel expressions at each iteration.

### 2.2. GAN-based texturing method

Unlike previous methods that focused on nonparametric image processing, the method using deep-neural-network-based GANs can enhance the performance of the learning result in a more direct manner by sharing common goals using a parameter technique (Cao, Zhou, Zhang, & Yu, 2017; Deng & Yu, 2014). The GAN is a neural network model wherein two neural networks (i.e. generator and discriminator) learn the distribution of data through repetitive competitions, generating virtual data similar to learned data (Goodfellow, 2016; Goodfellow, Bengio, & Courville, 2016). GANs are used mainly for conversions between different domains and are most commonly used to convert images or increase resolutions (Choi, 2017; Liu, Breuel, & Kautz, 2017; Huang, Yu, & Wang, 2018). In particular, the style that the user intends to recreate is learned, and new objects in that style can then be created.

Image-to-image translation using GANs is realized by learning two domains (i.e. A and B) and by applying the style of Domain B to the form of Domain A. Therefore, this study aims to achieve domain adaptation to map the texture of the actual image using the target domain image (Choi, 2017; Zhu, Park, Isola, & Efros, 2016). Additionally, semisupervised GANs (SGAN) provide an improved discriminator that enables the use of softmax activation functions to perform classification despite false judgments of data (Chavdarova & Fleuret, 2017). CycleGANs are models that seek to maintain their form and convert only the style of a specific object by adding cycle-consistency loss to the structure of the basic GAN. The structure of DiscoGAN is similar to that of the CycleGAN, but is designed to be more sensitive to errors through its use of the mean square error (Kim, Cha, Kim, Lee, & Kim, 2017; Zhu, Park, Isola, & Efros, 2016). Finally, StarGAN learns how the constructor receives both the model’s image and domain information as input and translates the input image using the target domain image (Choi et al., 2019). This aims to perform image transformations of multiple domains using one generator and one discriminator.

In the end, using the image-to-image translation of Table 1, the pix2pix and SGAN methods have low utility in the industry because of their limitations of dataset labeling. Furthermore, StarGAN does not align with the aims of this study, which are to convert only pipes, because it generates several domains. Therefore, the performances of CycleGAN and DiscoGAN, which are relatively unrestrained in the dataset problem and coincide with the aim of this study, are further compared and analyzed.

### 3. Methodology

#### 3.1. CycleGAN

CycleGAN is an image-translation algorithm that converts one image into another of the desired domain only using a specific part of the image via an unpaired dataset like Fig. 4. CycleGAN is structured into two generators (G_{ab} and G_{ba}) and two discriminators (D_a and D_b).

Equations (1)–(4) represent the CycleGAN. Equation (1) describes the process of converting from Domain A to B using the G function. The input value acquired from data sampling is fed into a log function corresponding to each domain, and the loss value is calculated using a discriminator. Similarly, equation (2) represents the process of converting from Domain B to A using the F function, which is the inverse function of G. This can be seen as the same process as equation (1). Besides, CycleGAN adds cycle-consistency loss to the image only for the object determined across different domains. This is given by equation (3), which represents a loss that occurs when the unpaired dataset is converted from Domain A to B and then restored to A. Limits are applied to the process of equation (4) by considering the processes of both conversion and restoration. The purpose function of the final CycleGAN that combines the above three equations can be expressed as equation (4). The lambda value of the right-hand term is set to 10, which is the weight value for the cycle-consistency loss. This value is the loss value to be restored to the original, citing the lambda value (Zhu et al., 2017). Finally, all losses of CycleGAN use a least-square GAN loss that has a more stable performance than the existing cross-entropy loss. This is because the least-square GAN loss ensures that the virtual generated data, such as real data, are more similar to real data (Mao et al., 2017).

The CycleGAN generator is an algorithm frequently used for image recognition. It uses a residual neural network (ResNet) structure and aims to separate only specific regions of interest from the background. ResNet has a disadvantage in that the network is deep, complicated, and consumes much time for learning. However, it is capable of processing high resolutions. Thus, it can generate higher quality images than other algorithms. Moreover, the discriminator uses a 70 × 70 patch to distinguish the image produced by the generator from the actual image. Because identification using the patch employs correlations between the peripheral pixels, it is possible to construct a more effective discriminator than with the pixel-by-pixel discrimination process.

\[
L_{\text{GAN}}(G, D_B, A, B) = E_{(b \sim P_{\text{data}})}[\log D_B(b)] + E_{(a \sim P_{\text{data}})}[\log(1 - D_B(G(A)))] \quad (1)
\]

\[
L_{\text{GAN}}(F, D_A, B, A) = E_{(a \sim P_{\text{data}})}[\log D_A(A)] + E_{(b \sim P_{\text{data}})}[\log(1 - D_A(G(B)))] \quad (2)
\]

\[
L_{\text{L1}}(G, F) = E_{(a \sim P_{\text{data}})}[(|F(G(A)) - A|)_1] + E_{(b \sim P_{\text{data}})}[(|G(F(B)) - B|)_1] \quad (3)
\]

Final objective function:

\[
L(G, F, D_A, D_B) = L_{\text{GAN}}(G, D_B, A, B) + L_{\text{GAN}}(F, D_A, B, A) + \lambda L_{\text{L1}}(G, F) \quad (4)
\]

#### 3.2. DiscoGAN

DiscoGAN shares the same ultimate purpose as CycleGAN and has an almost identical network structure and loss function. However, in CycleGAN, the L1 is used to obtain the absolute value of the difference, whereas DiscoGAN is more sensitive to error because it uses L2 (i.e. the mean squared error), which squares the difference.
Table 1: Comparison of GAN models.

| Number | Dataset  | Labeling | Feature                                      |
|--------|----------|----------|----------------------------------------------|
| Pix2pix | Paired   | √        | Loss1 per pixel + cGAN                       |
| SGAN   | Semi     | -        | Improved discriminator + softmax             |
| CycleGAN | Unpaired |          | Cycle-consistency reconstruction (least-square GAN) |
| DiscoGAN | Unpaired |          | Cycle-consistency reconstruction (mean square error) |
| StarGAN | Unpaired |          | CycleGAN + condition                         |

![Figure 3: Others based on GAN.](image)

Equations (5)–(9) represent DiscoGAN. First, DiscoGAN generates a fake domain, A, using functions G and F, similar to the cycle-consistency loss of CycleGAN. Domain A, thus generated, calculates the difference from the real A in equation (5). Equation (6) describes the loss value that is generated as the data of the fake domain, A, discriminated by the discriminator B. Equation (7) is the loss value generated by the generator, \( G_{AB} \), derived by adding equations (5) and (6). Finally, equation (8) adds the loss value generated through \( G_{AB} \) and \( D_B \) to derive the final loss value.

DiscoGAN has the advantage of a freer conversion, because the amount of information lost through the layers increases when using U-Net rather than ResNet as the generator. Consequently, unlike CycleGAN, where the background and object form are nearly completely maintained, DiscoGAN rarely maintains form and texture. Because the discriminator uses a \( 94 \times 94 \) patch, the discrimination is based on a larger area than CycleGAN, which uses a \( 70 \times 70 \) patch. Thus, the accuracy is lower. Paradoxically, the conversion performance improves, even as the accuracy reduces.

\[
L_{ConstA} (G, F) = d (F \cdot G(A), A) . (5)
\]

\[
L_{GAN} (G, F, A, B) = - E_{(A \sim Pdata(A))} [log D_B (G(A))] . (6)
\]

\[
L_{G,B} = L_{ConstA} (G, F) + L_{GAN} (G, D_B, A,B) . (7)
\]

\[
L_{DB} = -E_{(B \sim Pdata(B))} [log D_B (G(B))] - E_{(A \sim Pdata(A))} [log (1-D_B (G(A)))] . (8)
\]
Generating 3D texture

3.3. Comparison of texturing algorithms

The result of mapping the actual texture to the AR 3D model using CycleGAN and DiscoGAN is shown as Fig. 5 (Kim, Lee, & Kang, 2019). For a comparison of the two representative image transformation models, identical datasets and only simple models were used. The visual evaluations of the results of the two algorithms are almost identical, and the peak signal-to-noise ratio (PSNR) values of CycleGAN and DiscoGAN showed no significant difference. The higher the PSNR is, the higher the image resolution. However, because DiscoGAN transformed styles across objects outside of the background, it was necessary to preprocess the background of the dataset. Therefore, it is unsuitable to apply to industrial applications, and it is easier to apply CycleGAN, which provides appropriate texturing, even in unprocessed images. Therefore, in this study, automatic texturing is studied using the structure of CycleGAN. However, the AR pipes used in the shipbuilding industry are augmented from reality regardless of background, and the shapes of the pipes are very diverse. Therefore, AR pipes should be robust against background changes, and texturing should be performed for all elements of the pipe. Furthermore, because there are insufficient data caused by the characteristics of the industry, it is necessary to study algorithms that can recognize the target object in asymmetric datasets and separate completely the background and object.

3.4. Object separated from background

In this study, we propose to map the texture of Domain B onto a specific object of Domain A, which is an unpaired dataset. Domain A has the form of a virtual pipe, and Domain B has the texture of a real pipe. CycleGAN uses ResNet to recognize images but does not show satisfactory results when applying these existing CycleGANs as they are (He, Zhang, Ren, & Sun, 2015). Therefore, the original model is modified to use direct-skip connection and double normalization to enhance the results. The proposed method seeks to restructure the image recognition deep-learning network to fit the purpose, that is, to separate and generate specific objects from the background and map the learned textures using the given methods. Therefore, in this study, the classification performance was improved using a ResNet with direct-skip connection applied to the generator of CycleGAN. Furthermore, two normalizations inside ResNet were developed by combining the two normalizations and by adding appropriate weights to improve the object–background separation performance.

3.4.1. Direct ResNet

The deep-learning network for image recognition is a Convolutional Neural Network (CNN)-type architecture that extracts the characteristics of images via multiple layers. Generally, deep-learning models improve their performance as the model gets deeper: the deeper the layer, the more characteristics that can be extracted from the image. This property increases the learning parameter. However, increasing the depth of the layers increases the problem of the vanishing gradient, whereby the backpropagation slope value disappears. Therefore, ResNet is designed with a residual block to solve this problem. ResNet is a network that connects each layer via a skip connection through simple operations, using an identity function to bind each layer into a residual block, unlike the existing method seen in the left-hand side of Fig. 6. This connection method can avoid the vanishing gradient problem, because the value of the upper layer is fully propagated without change, despite it being a very deep layer.

Generally, ResNet shows notably higher performance and faster learning speeds than recognition models. However, when applied to pipes, the target dataset of this study, it gives poor performance. This pipe dataset is very complex and varied in shape. All data that bend like an elbow and do not have a constant thickness, such as a flange, should be considered. Thus, the feature-map information used in the first layer was not propagated to the last one, and the deformation of the image form was intensified accordingly. Usually, ResNet mitigates the problem of the disappearing gradient to some extent with backpropagation, as shown in Fig. 7. However, this study attempts to improve the results by adding a direct-skip connection to the existing ResNet to add more weight. Thus, it seeks to maintain the shape of the image and change only the texture by amplifying the effect of the learning parameter of the first layer to that of the final layer. Thus, to add the first layer’s parameter to the final layer, the two layers must be arranged in parallel and made into the form of a residual block using the identity function. Then, one more layer is added after the existing final layer. This is referred to as the direct-skip connection. It is thus an intuitive method that directly connects the parameter value extracted by the first layer to the final one. Therefore, as shown in Fig. 7, the feature map of the first layer and the feature map of the last layer can be seen in the form of an additional layer.

3.4.2. Double normalization

The internal covariate shift problem, in which the input distribution of each layer of the network varies as the layer of the model becomes deeper, also occurs. The coarser the layers, the more the weight changes accumulate, and thus, the greater the width of the value change. During learning, the problem is solved by normalizing this value. The normalization stage is placed directly in front of the activation function and plays the role of stabilizing the overall learning by directly normalizing the input value. There are several types of normalization, including batch, instance, and layer (Wu & He, 2018). These are distinguished by the units that are normalized, as shown in Fig. 8. Batch normalization is commonly used for recognition, and instance normalization is used to erase the existing image in image-to-image translation processes, such as style transfers.

The most widely used domain conversions in existing image-to-image translation methods are the first, whereby the image is converted into a specific style, and the second, whereby the shape is converted only to a specific object with similar shapes and an identical background between the two datasets. The first case converts the entire image. The second case retains the same background and a similar shape for the specific object. Therefore, both cases are suitable for instance normalization, which does not require a significant object-recognition performance and often features limited data. On the other hand, the dataset of this study has different backgrounds and object shapes between the two domain images. The comparison of datasets for the three cases can be seen in Fig. 9.

To solve these problems, this study employs a method of dividing normalization according to the background and specific objects, as shown in Fig. 10 (Nam & Kim, 2018). Thus, we propose a double normalization process that combines the two methods, using the batch norm to maximally maintain the background and shape of the object and the instance norm to erase the texture of the object. Equations (10) and (11) are the formulations for
4. Experimental Methods and Evaluation

4.1. Experimental method

The data used in this study are from Case 3 of Fig. 9. The dataset corresponding to Domain A is a virtual pipe, and Domain B is set as a real pipe. The following datasets were composed of frame-by-frame captures of the augmented virtual 3D-pipe model and the image of the actual pipe. CycleGAN was used as the background, and objects were separated through the neural network, including the unpaired data set. The background of the data was limited to the pipe model photographed outdoors. Additionally, to increase the amount of data applied in the learning process, the images were cropped and rotated by 90° to increase the number of datasets. Data of Domains A and B were adjusted to square sizes of 256 × 256 pixels, owing to the graphical-processing-unit data limit, and the total dataset was divided into 1000 pieces. Learning and verification data were divided into a 9:1 ratio.

The same dataset was used to compare the results of the basic CycleGAN model with the modified model. The model was constructed by transforming the learning technique described above one by one. The configuration for each model can be seen in Fig. 12. In the basic CycleGAN model, Algorithm 1 added double normalization to CycleGAN, and Algorithm 2 added double normalization and direct ResNet to CycleGAN. Because the variables were not considered beyond the aforementioned learning technique, the optimization technique was set as the Adam optimizer, and the learning rate was set to 0.0002. Furthermore, the batch size was 1 and was iterated only 2000 times. All three algorithms were designed to recognize objects that are pipes, which show the output of separating objects from the background. The explanation of each model can be seen in Table 2.

4.2. Qualitative evaluation

The results of the experiment can be seen in Fig. 13. The first result is a virtual pipe in AR, and the second result is the consequence of applying the actual texture to the virtual pipe using the basic CycleGAN. The third result is from Algorithm 1, and the fourth is from Algorithm 2. When the results were confirmed after the learning was completed, they were found to be close to the actual texture, reflecting the contrast between the fourth result and that of the basic CycleGAN. Therefore, meaningful results were confirmed.

The model using batch and instance normalization and the model using direct ResNet on the norm (i.e. Algorithm 2) both produced results superior to those of the basic CycleGAN. Cases 1, 6, 7, and 8 (i.e., CycleGAN and Algorithm 1) did not maintain the form of the object, whereas Algorithm 2 maintained the form and reflected the actual texture. In Case 2, all three models retained the color of the background and the shape of the object. In particular, Algorithm 2 completely detected the background letters. Cases 3 and 7 were converted into completely different colors in CycleGAN, and in Algorithm 1, the color of the background was maintained, except the lattice pattern. However, the results of Algorithm 3 fully reflected the actual model. In Cases...
Figure 7: Direct-skip connection.

Figure 8: Types of normalization.

Figure 9: Comparison of dataset types.
4, 9, and 10, the results of Algorithm 2 reflected the contrast and texture as the actual texture. Finally, Cases 3, 9, and 11, which can be called a complex model, showed notable results compared with other results. The shape and texture of the actual pipe were most similar in Algorithm 2, and the results of distinguishing objects from backgrounds were derived from various types of models. Thus, the direct ResNet and the double normalization were shown to have a positive effect on improving object-recognition performance and changing the appropriate textures. Thus, it is confirmed that it not only learns the characteristics of images, but it also better distinguishes the objects for learning purposes.

4.3. Quantitative evaluation

Figure 14 shows the final loss value of each model after learning that it is completed. It is generally difficult to quantitatively evaluate GAN-based learning results. GANs generate
Table 2: Explanation the model structure of three algorithms.

| Name | CycleGAN | Algorithm 1 | Algorithm 2 |
|------|----------|-------------|-------------|
| Generator | ResNet | ResNet | Direct ResNet |
| Normalization | Instance normalization | Double normalization (Batch + Instance) | Double normalization (Batch + Instance) |
| Learning time | 11 H | 11 H | 12 H |
| Remarks | Basic CycleGAN | - | - |
| Common Dataset | Train: Test = 900:100 | 100 | - |
| Learning rate | 0.002 | - | - |
| Batch size | 1 | - | - |
| Iteration | 20 000 | - | - |
| Etc | Leaky ReLU/Adam optimizer | - | - |

Figure 13: Results of the three algorithms.

results that do not exist in reality. However, because the learning proceeds in the direction of reducing the loss created by competition between the generator and the discriminator in the whole model, evaluating the trend of the loss can be part of a quantitative evaluation. The final loss value of CycleGAN was 0.25, and that of Algorithm 1 was 0.22, which was about 20%. In particular, in the case of Algorithm 2, the value of Algorithm 1 decreased by 68%, compared with the final loss value of 0.08, which showed visible performance improvement. Additionally, the overall graph trend showed the most stable decrease with Algorithm 2, compared with CycleGAN and Algorithm 1.
Table 3: Matching rate between average color value of the generated image and original image.

| Name | CycleGAN | Algorithm 1 | Algorithm 2 |
|------|----------|-------------|-------------|
| Case ① | 73.0%    | 81.2%       | 87.2%       |
| Case ② | 71.0%    | 70.0%       | 73.0%       |
| Case ③ | 50.1%    | 77.0%       | 81.9%       |
| Case ④ | 40.4%    | 60.3%       | 93.0%       |
| Case ⑤ | 45.5%    | 45.0%       | 86.6%       |
| Case ⑥ | 66.2%    | 90.2%       | 96.1%       |
| Case ⑦ | 56.0%    | 68.3%       | 90.1%       |
| Case ⑧ | 39.2%    | 56.5%       | 90.6%       |
| Case ⑨ | 81.5%    | 77.0%       | 96.7%       |
| Case ⑩ | 65.2%    | 58.3%       | 92.8%       |
| Case ⑪ | 70.6%    | 68.4%       | 78.4%       |

To additionally ensure the reliability of the color, the average values for the color components of the object, apart from the background, were calculated, and their consistency with the original pipe image was evaluated (equation 12). Table 3 summarizes the matching rate per case for each algorithm. Additionally, the highest rate of agreement was found in Algorithm 2, similar to the previous evaluation, which indicates that it was most similar to the original.

Matching rate (%) = \( \frac{\mu_{\text{Generated Image}}}{\mu_{\text{Real Image}}} \)  

5. Conclusions and Future Scope

Generally, when using AR pipe models in the shipbuilding and marine industry, realistic and sophisticated textures must be created for the immersion of a worker in the simulation. In the past, these tasks were usually performed manually, which is difficult because of low productivity. More economical texturing could be achieved by automating the process of generating and mapping sophisticated textures. The AR texturing process using the deep-learning model is thus considered a firm basis for generating low-cost high-quality textures in the future. Notably, the method that employs a GAN showed that the texture learned on the virtual pipe model can be applied by learning the texture of the actual pipe.

Furthermore, this study derived the image-conversion results that were improved through multiple models developed from the existing CycleGAN. The results of each model were evaluated and analysed. The applicability of the AR model for automatic texturing was suggested. Particularly, if the hardware recognizing the light source were to be combined using the proposed method, the shape of the object in image would be recognized, and the light contrast would be rendered automatically according to the angle of shape. The proposed method recognized the shape of the object in the image and automatically rendered the light contrast according to the angle of the shape. This saved physical resources following the production of the AR model, and it made the object look more realistic. This experiment was designed to texture a pipe using a single color. However, the results are expected to be applicable to various objects.

As a result of the assumptions and processes required, our methods were not enough to apply in the complete pipe model. Because the real-time lighting was not reflected, and the resolution was 256 × 256, it remained limited in model size and in real-time AR environment. Moreover, because it could not take the complete form of the model, it is necessary to preserve this form completely and to replace only the texture. It was also difficult to judge various shapes and colors of pipes with one learning session.

Thus, it will likely take a long time to repeat the number of learning sessions until the entire model is fully generated. For this problem, the generator should consider the process of classifying and learning a plurality of objects in a single photo by receiving additional information when learning the image of the actual pipe. Then, the discriminator should consider a solution that distinguishes multiple objects.

Acknowledgement

This work was supported by INHA University Research Grant (2020) and the Technology Innovation Program (No. 20002772, Development of the smart manufacturing collaboration system for the innovation of pipe and steel outfit and block logistics in the shipbuilding and marine) funded by the Ministry of Trade, Industry & Energy (MOTIE, Korea).
6. Conflict of interest statement

None declared.

References

Alshawabkeh, Y., & Haala, N. (2005). Automatic multi-image photo-texturing of complex 3D scenes. *International Archives of Photogrammetry and Remote Sensing, 34*, 1–6.

Bernardini, F., Martin, I. M., & Rushmeier, H. (2001). High-quality texture reconstruction from multiple scans. *IEEE Transactions on Visualization and Computer Graphics, 7*(4), 318–332.

Buyukdemircioglu, M., Kocaman, S., & Isikdag, U. (2018). Semi-automatic 3D city model generation from large-format aerial images. *ISPRS International Journal of GeoInformation, 7*(9), 339.

Cao, Y., Zhou, Z., Zhang, W., & Yu, Y. (2017). Unsupervised diverse colorization via generative adversarial networks, in *Machine Learning and Knowledge Discovery in Databases* (ECML PKDD 2017), 10534, 151–166.

Cebollero, A., & Sanchez, L. (2019). Virtual reality empowered design. In *International Conference on Computer Applications in Shipbuilding 2017*.

Ceruti, A., Marzocca, P., Liverani, A., & Bil, C. (2019). Maintenance in aeronautics in an Industry 4.0 context: The role of augmented reality and additive manufacturing. *Journal of Computational Design and Engineering, 6*(4), 516–526.

Chavdarova, T., & Fleuret, F. (2017). SGAN: An alternative training of generative adversarial networks, in *IEEE Computer Vision and Pattern Recognition*, (pp. 9407–9415).

Chen, Y. (2007). 3D texture mapping for rapid manufacturing. *Computer-Aided Design and Applications, 4*(6), 761–771.

Choi, Y., Choi, M., Kim, M., Ha, J. W., Kim, S., & Choo, J. (2018). StarGAN: Unified generative adversarial networks for multi-domain image-to-image translation, in *IEEE Conference on Computer Vision and Pattern Recognition*, (pp. 8789–8797).

Deng, L., & Yu, D. (2014). Deep learning: Methods and application. *Foundations and Trends in Signal Processing, 7*(3–4), 197–387.

Fraga-Lamas, P., Fernandez-carames, T. M., Blanco-Novoa, O., & Vilar-montesinos, M. (2018). A review on industrial augmented reality systems for the industry 4.0 shipyard. *IEEE Access, 6*, 13358–13375.

Fukuda, T., Yokoi, K., Yabuki, N., & Motamedi, A. (2019). An indoor thermal environment design system for renovation using augmented reality. *Journal of Computational Design and Engineering, 6*(2), 179–188.

Gatys, L. A., Ecker, A. S., & Bethge, M. (2016). A neural algorithm of artistic style, *Journal of Vision, 16* (12), 326.

Goodfellow, I. (2016). Generative Adversarial Networks. In *NIPS 2014*.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep learning. MIT Press.

Harrison, P. (2001). A non-hierarchical procedure for re-synthesis of complex texture. In *Conference Proceedings: The Ninth International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision 2001: University of West Bohemia, Plzen, Czech Republic* (pp. 190–197).

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Deep residual learning for image recognition. In 2016 IEEE Conference on Computer Vision and Pattern Recognition, (pp. 770–778).

Heindl, C., Akkaladevi, S. C., & Bauer, H. (2016). Photorealistic texturing of human busts reconstruction. In *Seventh International Conference on 3D Body Scanning Technologies* (pp. 225–230).

Hertzmann, A., Jacobs, C. E., Oliver, N., Curless, B., & Salesin, D. H. (2001). Image analogies. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques* (pp. 327–340).

Hikida, K., Fukuto, J., Numano, M., Inouka, T., & Nakato, M. (2010). Development of a visual lookout support system with head-up display. *The Journal of Japan Institute of Navigation, 122*, 7–13.

Huang, H., Yu, P. S., & Wang, C. (2018). An introduction to image synthesis with generative adversarial nets. preprint arXiv:1803.04469.

Isola, P., Zhu, J. Y., Zhou, T., & Efros, A. A. (2016). Image-to-image translation with conditional adversarial networks. In *Computer Vision and Pattern Recognition (CVPR 2017)*.

Jakub, W., & Andrzej, L. (2018). Ship information systems using smartglasses technology. *Scientific Journal of Silesian University of Technology (Series Transport)*, 100, 211–222.

Ke, S., Xing, F., Zhang, Z., & Zuo, Y. (2019). An enhanced interaction framework based on VR, AR and MR in Digital Twin. *Procedia CIRP, 83*, 753–758.

Kil, W., Son, M., & Lee, J. (2018). Development of VR ship environment for the educational training of ship survey. *Journal of Ship and Ocean Technology, 220*, 361–369.

Kim, D., Park, K. H., & Ko, K. H. (2018). Development of an AR-based method for augmentation of 3D CAD data onto a real ship block image. *Computer-Aided Design, 98*, 1–11.

Kim, M., Lee, K., & Kang, H. (2019). A comparison study on creation of texture for visualization with realistic virtual model in augmented reality. *Korean Journal of Computational Design and Engineering, 24*(2), 169–179.

Kim, S., Lee, J., Son, G., & Han, E. (2012). Conceptual design of product & asset Lifecycle management system for marine structures during middle of life. *Journal of Ocean Engineering and Technology, 26*(2), 58–67.

Kim, T., Cha, M., Kim, H., Lee, J. K., & Kim, J. (2017). Learning to discover cross-domain relations with generative adversarial networks. preprint arXiv:1703.05192.

Laycock, R. G., Ryder, G. D. G., & Day, A. M. (2007). Automatic generation, texturing and population of a reflective real-time urban environment. *Computers and Graphics, 31*(4), 625–635.

Lee, W., Lee, K., Lee, J., & Nam, N. (2020). Registration method for maintenance-work support based on augmented-reality-model generation from drawing data. *Journal of Computational Design and Engineering, 7*(6), 775–787.

Liniger, S. (2012). Automatic method for photo texturing geolocated 3D models from geolocated imagery. US Pat., 1.

Liu, M. Y., Breuel, T, & Kautz, J. (2017). Unsupervised image-to-image translation networks. In *Computer Vision and Pattern Recognition (NIPS 2017)*.

Mao, X., Li, Q., Xie, H., Lau, R., Wang, Z., & Smolley, S. P. (2017). Image-to-image translation with conditional adversarial networks. In *Conference on Computer Vision and Pattern Recognition (CVPR 2017)*.

Montwitt, A., Kasinsk, J., & Pietrzak, K. (2018). Importance of key phases of the ship manufacturing system for efficient vessel life cycle management. *Pocedia Manufacturing, 19*, 34–41.

Nam, H., & Kim, H. E. (2018). Batch-instance normalization for adaptively style-invariant neural networks. In Advances in Neural Information Processing Systems 31 (NIPS 2018).

Odena, A. (2016). Semi-Supervised learning with generative adversarial networks, in *Machine Learning* (ICML 2016).

Ovcharenko, O., Kazei, V., Peter, D., & Alkhalifah, T. (2019). Style transfer for generation of realistically textured subsurface models. In *SeG Technical Program Explorations Abstracts* (pp. 2393–2397).
Saddik, A. (2018). Digital Twins: The convergence of multimedia technologies. IEEE MultiMedia, 25(2), 87–92.
3Dmaritim. https://www.3dmaritim.de/ Accesssed 30 March 2019.
Welsh, T., Ashikhmin, M., & Mueller, K. (2002). Transferring color to greyscale images. In SIGGRAPH 2002(pp. 277–280).
Wu, Y., & He, K. (2018). Group normalization. In Computer Vision and Pattern Recognition (ECCV 2018).
Yang, Y., Ohtake, Y., & Suzuki, H. (2020). Mesh processing for improved perceptual quality of 3D printed relief. Journal of Computational Design and Engineering, 8(1), 1-10. https://doi.org/10.1093/jcde/qwaa065.
Zhu, J. Y., Park, T., Isola, P., & Efros, A. A. (2017). Unpaired image-to-image translation using cycle-consistent adversarial networks. In Computer Vision and Pattern Recognition (ICCV 2017).