Sensors and Materials, Vol. 34, No. 2 (2022) 515–522
MYU Tokyo

S & M 2824

Image-to-image Translation via Contour-consistency Networks

Hsiang-Ying Wang,[1] Hsin-Chun Lin,[2] Chih-Hsien Hsia,[3*]
Natnuntnita Siriphockpirom,[2] Hsien-I Lin,[1] and Yung-Yao Chen[2**]

[1]Graduate Institute of Automation Technology, National Taipei University of Technology,
No. 1, Sec. 3, Zhongxiao E. Rd., Taipei City 106, Taiwan
[2]Department of Electronic and Computer Engineering, National Taiwan University of Science and Technology,
No. 43, Sec. 4, Keelung Rd., Taipei City 106, Taiwan
[3]Department of Computer Science and Information Engineering, National Ilan University,
No. 1, Sec. 1, Shennong Rd., Yilan City, Yilan County 260, Taiwan

(Received June 30, 2021; accepted October 6, 2021)

Keywords: image-to-image translation, contour-consistency networks, inconsistency problem, attention feature map

In this paper, a novel framework for image-to-image translation, in which contour-consistency networks are used to solve the problem of inconsistency between the contours of generated and original images, is proposed. The objective of this study was to address the lack of an adequate training set. At the generator end, the original map is sampled by an encoder to obtain the encoder feature map; the attention feature map is then obtained using the attention module. Using the attention feature map, the decoder can ascertain where more conversions are required. The mechanism at the discriminator end is similar to that at the generator end. The map is sampled through an encoder to obtain the encoder feature map and then converted into the attention feature map. Finally, the map is classified by the classifier as real or fake. Experimental results demonstrate the effectiveness of the proposed method.

1. Introduction

In unmanned robot and self-driving car applications,[1–4] labeling data is one of the most time-consuming tasks. For instance, when daytime and nighttime data are required, data must be collected during different periods and then staff must be assigned to label the data. Marking data is labor-intensive. Usually, the data available for training a deep learning model are inadequate; as such, convergence of the training process is poor. To address this problem, we propose a method for improving the similarity of transformed images; an already labeled database is converted into different domain textures to obtain differing domain style information for data augmentation.

With the progress of deep learning, several related areas of study have emerged, such as computer vision and speech recognition. In computer vision, image-to-image translation is vital. Goodfellow *et al.*[5] proposed a method called the generative adversarial network (GAN) for

*Corresponding author: e-mail: chhsia625@gmail.com
**Corresponding author: e-mail: yungyaochen@mail.ntust.edu.tw
https://doi.org/10.18494/SAM3493

ISSN 0914-4935

generating real-world images. Since then, the derivation of image-to-image translation and image restoration, processing, and coloring have been popular research topics.

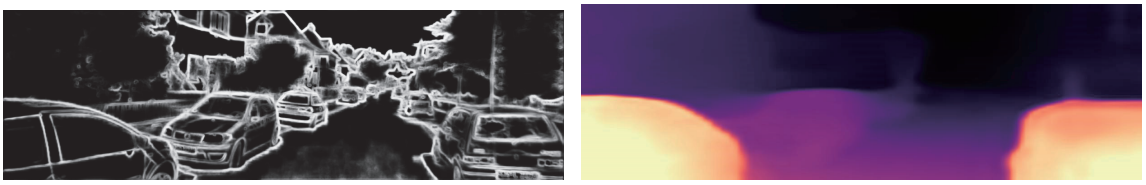
Image-to-image translation can be performed by two methods: paired and unpaired training. The paired training method is a supervised learning method and requires paired images. For example, the method proposed by Isola *et al.*[6] requires the preparation of a database of paired images. However, the preparation of paired image data is difficult for some tasks, such as converting the season of landscape photographs.

The unpaired training method is an unsupervised learning method because it does not require the preparation of paired images. GAN loss alone is known to be insufficient and often leads to inadequate training and generates inferior images. Therefore, in 2017, Zhu *et al.*[7] proposed CycleGAN, which is a cycle consistency approach, to determine whether the images reconstructed from the target domain back to the source domain are the same. The cycle consistency approach helps ensure that specific features are unchanged after the transformation.

Using the cycle consistency approach, we analyzed the dataset of the video game Grand Theft Auto (GTA) version 5 and the Cityscapes Dataset and converted the computer-generated images into real-world images. Our objective was to preserve the original road structure and convert it into real-world texture information. As shown in Fig. 1, we discovered two crucial features that are related to street structure and can improve the stability of the generated images. (1) Edge information: the similarity of the edges can be calculated to determine whether the edges of the images are the same before and after the conversion. (2) Depth information: the size and proximity of the objects are often neglected when performing object recognition for self-driving vehicles.

2. Proposed Contour Consistency Network

For image-to-image translation, we propose a contour consistency network as a generic network that ensures the consistency of the image profile after conversion (Fig. 2). First, the original domain image is passed through the generator to obtain a photograph of the target domain. Then, the original and generated images are passed through the edge detection and depth estimation networks to obtain an edge map and a depth map, respectively. The loss is then calculated for the target domain photograph; the contour loss calculated using the contour consistency network is added to the original loss function to obtain superior results, with the function proposed by Chen *et al.*[8] being used for training.



(a) (b)

Fig. 1. (Color online) (a) Edge detection map and (b) depth estimation map.

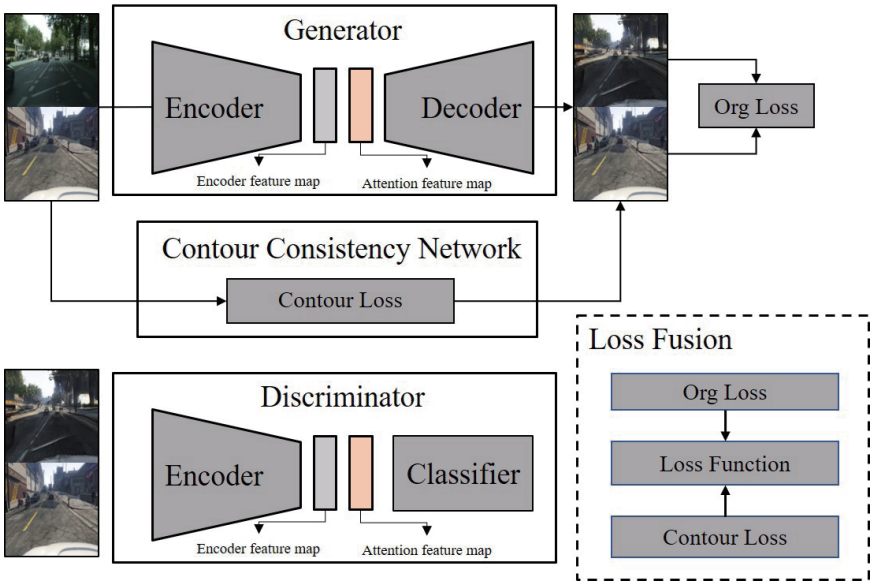


Fig. 2. (Color online) Architecture of the proposed contour consistency network.

2.1 Generator

The generator and discriminator used in this study are based on the unsupervised generative attentional networks with adaptive layer-instance normalization for image-to-image translation (U-GAT-IT)[9] approach. However, we integrate the attention module (which helps the model ascertain where more conversions are required) and the adaptive layer-instance normalization (Ada-LIN) method (which enables the model to maintain a favorable transformation style). In the Ada-LIN method, layer normalization helps retain more wind features whereas instance normalization helps retain more content features, thus enabling the model to learn adaptively. In summary, the model generation effect can be improved by combining it with the contour consistency network proposed in this paper.

Generator: First, the original map is sampled by the encoder to obtain the encoder feature map; then, the attention feature map is obtained using the attention module. Using the attention feature map, the decoder can ascertain where more conversions are required.

Discriminator: The structure of the discriminator is similar to that of the generator. The original map is sampled by the encoder to produce the encoder feature map, which is then converted into the attention feature map. Finally, it is classified as real or fake by the classifier.

2.2 Contour consistency network

To solve the problem of the contour of an image being dramatically altered after the conversion, a contour consistency network architecture is proposed for improving the conversion of computer-generated street views into real street views. As illustrated in Fig. 3, the original image and transformed image are passed through the edge detection and depth estimation branches, respectively, and their similarity is calculated.

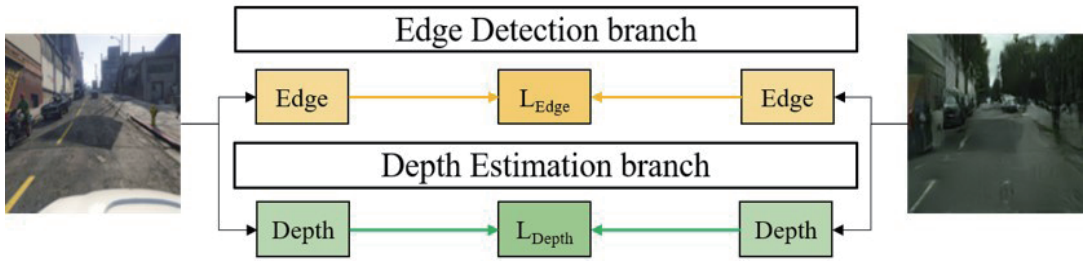


Fig. 3. (Color online) Detailed structure of the contour consistency network.

Edge detection branch: This branch uses edge detection to solve the image contour consistency problem. Edge detection offers several advantages: edge detection can be used to calculate the edges of objects and backgrounds in the image, and by analyzing the edges between the foreground and background, whether or not the texture after conversion is acceptable can be ascertained. We used the holistically nested edge detection (HED)[10] method, which is a modified edge detection algorithm based on the VGG16 architecture. This deep-learning-based algorithm yields better results than the traditional edge detection algorithm for generator-generated images because such images are usually blurry and traditional deep learning is less effective in determining edges. Therefore, in this branch, two edge maps are obtained from the original image and the transformed image obtained through HED, and the L1 loss algorithm is employed to calculate the similarity.

Depth estimation branch: This branch uses a depth map, generated through depth estimation, to improve the consistency of image contours. The depth map, representing the depth information, is applied to our contour consistency network. Even if objects overlap in the depth map, the depths of the objects can be calculated to determine the contours. The Monodepth2[11] algorithm is used to compute the original image and transformed image, and the L1 loss function is used to compute the similarity.

The losses from the two branches are added to obtain the contour loss, which can be expressed as follows:

$$L_{contour} = \frac{1}{N}\sum_{x\sim Xs}\left|Edge(x) - Edge(G(x))\right| + \frac{1}{N}\sum_{x\sim Xs}\left|Depth(x) - Depth(G(x))\right|, \quad (1)$$

where x is the number of pixels in the original image, $G(x)$ is the image converted by the generator, $Edge()$ is the edge map generated through edge detection, and $Depth()$ is the depth map generated through depth estimation; the results are added to obtain $L_{contour}$.

2.3 Loss function

The original U-GAT-IT has four parameters: adversarial loss, cycle loss, identity loss, and cam loss. Their definitions are given briefly as follows. Adversarial loss: This L_{lsgan} value predicted by the least-squares GAN (LSGAN) is used to calculate the distributions of the image before and after the conversion. Cycle loss: This L_{cycle} value predicted by the LSGAN is used to

ensure that the images converted into the target domain can be converted back into the source domain. Identity loss: This $L_{identity}$ value predicted by the LSGAN is used to ensure the color distribution of the image before and after the conversion. Cam loss: This L_{cam} value predicted by the LSGAN is used to ascertain which areas require improvement before and after the conversion or where the biggest difference between the two domains is. Contour loss: This $L_{Contour}$ value predicted by the LSGAN is used to calculate the contour difference between the images before and after conversion. These four parameters are summed to determine the final loss.

$$L = \lambda_1 L_{lsgan} + \lambda_2 L_{cycle} + \lambda_3 L_{identity} + \lambda_4 L_{cam} + \lambda_5 L_{Contour} \tag{2}$$

To adjust the parameters used in this study, we employed the ratios presented in the original paper [U-GAT-IT[9]], $\lambda_1 = 1$, $\lambda_2 = 10$, $\lambda_3 = 10$, $\lambda_4 = 1000$, and contour loss $\lambda_5 = 1$, in our experiments.

3. Experiments

3.1 Dataset

In this section, we introduce the two datasets used for experiments and demonstrations. We used GTA datasets to perform stylistic conversions with Cityscapes; these two datasets are often used in studies on image-to-image translation.

GTA datasets: Richter *et al.*[12] extracted images from the open-world game, Grand Theft Auto, to create a dataset that comprises virtual city street scenes. This dataset contains 24966 images, each of which has a pixel-level semantic annotation; the dataset is therefore popular for semantic segmentation. In the present study, 2500 images were used for training and 100 for testing.

Cityscapes datasets: Cordts *et al.*[13] collected street view data from various cities and created a dataset comprising 25000 images, including 5000 finely labeled images and 20000 roughly labeled images. This dataset is often used in semantic segmentation tasks, similar to the GTA dataset. In the present study, 2993 images were used as training data and 100 images were used as test data.

These two datasets were used because both show street scenes and include complexities found in real scenes; thus, these datasets can be used to evaluate the effectiveness of the proposed model (Fig. 4).

(a) (b)

Fig. 4. (Color online) (a) GTA dataset sketch map and (b) Cityscapes dataset sketch map.

3.2 System performance evaluation

The model used in this experiment is based on the extension of the U-GAT-IT and was implemented using PyTorch with the standard initial values. The contour consistency network proposed herein was used for the training. VGG16 was the base network of the HED detection model and was pretrained using the Berkeley Segmentation Dataset 500.[14] For depth estimation, the Monodepth2 model was pretrained using the KITTI dataset, and the image resolution was set to 300000 iterations for each experiment.

To demonstrate the feasibility of our method, we employed the inception score (IS),[15] Frechet inception distance (FID),[16] and learned perceptual image patch similarity (LPIPS)[17] as evaluation indexes because these can be used to evaluate image diversity and quality. The IS was primarily used to assess the quality of the two indicators and image diversity. The higher the IS, the higher the monetary worth. The FID is also used to assess image quality and diversity. The lower the FID value, the better. The LPIPS is the difference between the features extracted by different network architectures; the smaller the value, the greater the similarity.

The results presented in Tables 1 and 2 demonstrate the effectiveness of the proposed method. We analyzed different network architectures and performed ablation experiments with the original U-GAT-IT, U-GAT-IT with edge loss, U-GAT-IT with depth loss, and our proposed approach with contour loss. The GTA to Cityscapes and Cityscapes to GTA conversions were analyzed. Table 3 shows the results of the conversion from GTA to Cityscapes. The conversion yielded different results in two directions. However, the addition of depth loss caused a considerable increase in the FID and a notable decrease in quality. For the Cityscapes to GTA conversion (Table 4), considerable improvement was observed when using only a single

Table 1
GTA to Cityscapes evaluation with IS, FID, and LPIPS.

Method	IS	FID	LPIPS
U-GAT-IT	2.16	166.404	0.4386
U-GAT-IT + edge loss	2.36	169.880	0.4396
U-GAT-IT + depth loss	2.17	177.852	0.4376
Proposed	2.36	169.880	0.4322

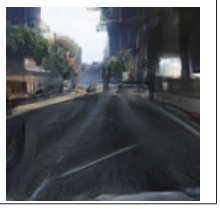
Table 2
Cityscapes to GTA evaluation with IS, FID, and LPIPS.

Method	IS	FID	LPIPS
U-GAT-IT	1.79	115.163	0.4582
U-GAT-IT + edge loss	1.73	112.285	0.4618
U-GAT-IT + depth loss	1.73	112.285	0.4601
Proposed	1.83	110.827	0.4548

Table 3
(Color online) GTA to Cityscapes conversion.

Input	U-GAT-IT	With depth loss	With edge loss	Proposed
				

Table 4
(Color online) Cityscapes to GTA conversion.

Input	U-GAT-IT	With depth loss	With edge loss	Proposed
				

particular loss. However, regardless of the task used, the proposed method yielded more comprehensive results.

4. Conclusion

In this study, we proposed a network for solving the problem of inconsistency between the contours of a generated image and the original image, resulting in generated images that are more similar to the original images. Edge detection and depth estimation are performed in a modular manner; thus, applying the proposed approach to all image-to-image translation conversions is straightforward. The model is used only in the training phase and hence does not increase the performance burden during actual testing. The contributions of this study can be summarized as follows: (1) we proposed a contour consistency network that minimizes the alteration of the transformed image profile; (2) we proposed two feature types that can be used to calculate image contours—edge features and depth features; and (3) the proposed method can generate images of different domains while retaining the original structure, thus solving the problem of time-consuming data collection and annotation. In future studies, we will determine suitable network models for different domain conversions. In addition, we will integrate different consistency networks into our current system.

References

1 C. Masuzaki, I. Yamamoto, A. Morinaga, K. Sadano, Y. Kai, T. Nakano, and Y. Kato: Sens. Mater. **33** (2021) 907. https://doi.org/10.18494/SAM.2021.3223
2 Y. Kawamura, J. Tahara, T. Kato, S. Fujii, S. Baba, and M. Koike: Sens. Mater. **33** (2021) 883. https://doi.org/10.18494/SAM.2021.3216
3 L. Wang, R. Li, J. Sun, H. S. Seah, C. K. Quah, L. Zhao, and B. Tandianus: Sens. Mater. **32** (2020) 1245. https://doi.org/10.18494/SAM.2020.2549
4 Y. X. Lin, D. S. Tan, Y. Y. Chen, C. C. Huang, and K. L. Hua: IEEE MultiMedia **27** (2020) 44. https://doi.org/10.1109/MMUL.2020.3008529
5 I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio: arXiv:1406.2661 (2014).
6 P. Isola, J. Zhu, T. Zhou, and A. A. Efros: 2017 IEEE Conf. Computer Vision and Pattern Recognition (2017) 5967–5976. https://doi.org/10.1109/CVPR.2017.632
7 J. Zhu, T. Park, P. Isola, and A. A. Efros: 2017 IEEE Conf. Computer Vision (2017) 2242–2251. https://doi.org/10.1109/ICCV.2017.244
8 Y. Chen, G. Li, S. Jhong, P. Chen, C. Tsai, and P. Chen: Sens. Mater. **32** (2020) 3157. https://doi.org/10.18494/SAM.2020.2838

9 J. Kim, M. Kim, H. Kang, and K. Lee: arXiv:1907.10830 (2019).
10 S. Xie and Z. Tu: 2015 IEEE Conf. Computer Vision (2015) 1395–1403. https://doi.org/10.1109/ICCV.2015.164
11 C. Godard, O. M. Aodha, M. Firman, and G. Brostow: 2019 IEEE Conf. Computer Vision (2019) 3827–3837. https://doi.org/10.1109/ICCV.2019.00393
12 S. R. Richter, V. Vineet, S. Roth, and V. Koltun: Eur. Conf. Computer Vision (2016) 102–118. https://doi.org/10.1007/978-3-319-46475-6_7
13 M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele: 2016 IEEE Conf. Computer Vision and Pattern Recognition (2016) 3213–3223. https://doi.org/10.1109/CVPR.2016.350
14 P. Arbeláez, M. Maire, C. Fowlkes, and J. Malik: IEEE Trans. Pattern Anal. Mach. Intell. **33** (2011) 898. https://doi.org/10.1109/TPAMI.2010.161
15 T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen: arXiv:1606.03498 (2016).
16 M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter: arXiv:1706.08500 (2017).
17 R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang: 2018 IEEE Conf. Computer Vision and Pattern Recognition (2018) 586-595. https://doi.org/10.1109/CVPR.2018.00068