Intelligent Automation & Soft Computing
DOI:10.32604/iasc.2022.019979
Article

Tech Science Press

Unsupervised Semantic Segmentation Method of User Interface Component of Games

Shinjin Kang[1] and Jongin Choi[2,*]

[1]School of Games, Hongik University, Sejong, 30016, Korea
[2]Department of Digital Media, Seoul Women's University, Seoul, 01797, Korea
[*]Corresponding Author: Jongin Choi. Email: funtech@swu.ac.kr
Received: 04 May 2021; Accepted: 02 July 2021

Abstract: The game user interface (UI) provides a large volume of information necessary to analyze the game screen. The availability of such information can be functional in vision-based machine learning algorithms. With this, there will be an enhancement in the application power of vision deep learning neural networks. Therefore, this paper proposes a game UI segmentation technique based on unsupervised learning. We developed synthetic labeling created on the game engine, image-to-image translation and segmented UI components in the game. The network learned in this manner can segment the target UI area in the target game regardless of the location of the corresponding component. The proposed method can help interpret game screens without applying data augmentation. Also, as this scheme is an unsupervised technique, it has the advantage of not requiring paring data. Our methodology can help researchers who need to extract semantic information from game image data. It can also be used for UI prototyping in the game industry.

Keywords: Object segmentation; UI segmentation; game user interface; deep learning; generative adversarial network

1 Introduction

In the gaming industry, various deep learning-based algorithms are used in game development to enhance the features of the game and to improve the response to each player's actions dynamically. Specifically, vision-based algorithms are applied to reinforcement learning [1] and level generation field [2–4], which show their functionality and capability. Hence, these techniques can achieve the target learning goal while using only pure pixel images from the game. However, since they do not utilize in-game parameters, it is not possible to achieve a more specialized learning goal for the game, which makes it only applicable to manageable action games or puzzle games.

On the game screen, the user interface (UI) provides essential information that helps the player navigate, continue and accomplish goals during gameplay. In a convolutional neural network (CNN), which uses only pure pixel information on the screen, if the information provided by the UI is extracted and used separately as an additional input value, then there will be a considerable improvement in the learning efficiency of the deep

learning network. For this reason, UI components such as buttons, image icons and gauge bars must be effectively segmented on the game screen with the aim of analyzing the corresponding image only.

In general, if the information obtained from the UI is presented in the form of an application programming interface (API) in game development, then game researchers can easily access and use it in various research fields. However, games are commercial software that prevents external access to these data to protect the contents from numerous hacking activities. Therefore, researchers have to develop separate tools and acquire these data by themselves [5]. These become barriers to the development of deep learning research in game development. Recently, we developed a tool that connects the recent vision-based object segmentation technology to the app used in the data mining field, detects the main UI area in the general app screen and then distinguishes the UI image from it [6]. Since the developed technique can extract semantic information from game screenshots, further studies can be conducted using this knowledge. In addition, it helps to automatically generate the data set necessary for image tag labeling, outlier detection and highlight scene generation. However, since the supervised learning technique was applied using the paired game screen and segmentation labeling image, the pairing data could not be obtained and this serves as a limitation in the study.

Taking this into account, we extended our study to show that UI segmentation is possible with image-to-image translation without paring data. The proposed method in this paper has the advantage of supporting large-scale machine learning research related to game data by increasing the data accessibility of individual researchers. In contrast to existing research, our research has the following characteristics:

- Semi-Automatic Segmentation Data Generation: A technique that approximates the UI area from the app screen and semi-automatically generates a large-scale dataset required for machine learning.
- Image-to-Image Translation Network for UI Segmentation: Deep learning network that translates from original game screen image to UI segmented image based on unsupervised learning.

2 Previous Works

2.1 Segmentation Using Game Data

There were various attempts made to increase the efficiency of segmentation using game data. Moreover, there is a need for an increase in human resources to produce large-scale label data at the pixel level, which results in considerable costs. Richter et al. [7] proposed a method to quickly generate an accurate label map in pixels on an image extracted from a computer game. They verified their proposed method by generating a dense pixel-level label for 30,000 images produced by an open world computer game. Meanwhile, Chen et al. [8] employed a recognition approach to estimate affordance, unlike the traditional method mainly used for the autonomous driving method. Consequently, they proposed input image mapping to several key recognition indicators related to the affordance of road and traffic conditions for driving. It provides a complete set of descriptions for the relevant scene so that a simple controller can navigate autonomously. To prove this, they trained a deep CNN using long-period driving images extracted from a video game and showed the possibility of an autonomous driving system in a virtual environment. Taylor et al. [9] provided a surveillance simulation test environment that could be used by anyone, based on Object Video Virtual Video, a commercial game engine. It simulates various video streams synchronized in numerous camera configurations in a virtual environment where humans and vehicles controlled by a computer or player are present. Moreover, Marin et al. [10] suggested a method that pedestrian images learned in an arbitrary scenario operated appropriately for real pedestrian detection. After recording the training sequence in such a situation, they determined a shape-based pedestrian classifier. Vazquez et al. [11] was built for the same purpose as the study conducted by Marin et al. [10]. To accomplish their objectives, they designed a domain adaptation framework called V-AYLA.

By collecting several samples of pedestrians in the real world and then combining them with several examples in the virtual world, they tested a method of training a pedestrian classifier that operated in the target domain.

In this study, we applied object segmentation methods to several gameplay screens. The proposed algorithm first segments the UI that contains the most meaningful content within the game. This study acknowledges the works of Liao et al. [12], which obtained the necessary data from the gameplay data and Richter et al. [7], which proposed a methodology for detecting game objects.

2.2 Segmentation Methods

Instance segmentation independently masks each instance of an object contained in an image at the pixel level [13,14]. Object detection and instance segmentation are highly interrelated. In object detection, researchers use a bounding box to detect each object instance of an image with a label for classification. Meanwhile, instance segmentation takes this one step forward and applies a segmentation mask to each instance of an object. The classification of the recently developed CNN-based instance segmentation models is based on their principal functions. Girshick et al. [15] proposed a novel design for instance segmentation called simultaneous detection and segmentation (SDS) [14], which followed the architecture of the object detector and was a four-stage instance segmentation model. Thus far, CNN-based models have used only the last layer functional map for classification, detection and segmentation. In addition, they proposed the concept of hypercolumn [16] using information from some or all middle functional maps of the network for better instance segmentation. The authors added the notion of hypercolumn to the SDS and the modified network achieved better segmentation accuracy. Various detection algorithms such as region based convolutional neural network (R-CNN), spatial pyramid pooling in deep convolutional networks (SPPnet) [17] and Fast R-CNN [18] used a two-stage network for object detection. The first stage detects object proposals using the selective search algorithm [19] and the second stage classifies them using another CNN-based classifier. Multibox [20,21], Deepbox [22] and Edgebox [23] used a CNN-based proposal generation method for object detection. Faster R-CNN [24] generated box proposals using a CNN-based region proposal network (RPN). However, this proposal generation mode uses the bounding box and instance segmentation model. Instance segmentation algorithms such as SDS and hypercolumn used multi-scale combinatorial grouping (MCG) [25] to generate region proposals. By using CNN-based region proposal network (RPN) as Faster R-CNN, Deep Mask [26] also generated region proposals for the model to be trained end-to-end. Previous object detection and instance segmentation modules such as [22–24] used methods such as selective search, MCG, constrained parametric min-cuts (CPMC) [27] and RPN. Dai et al. [28], however, did not follow the conventional way of using the pipeline network and did not use the external mask proposal method. Instead, they used a cascading network to incorporate the functions of different CNN layers for instance segmentation. SDS, deep mask and hypercolumn used feature maps at the top layer of the network to detect object instances leading to coarse object mask generation. In [29–32], introducing skip connection is shown to be more effective for semantic segmentation than instance segmentation owing to the reduction of the roughness of the mask. Pinheiro et al. [33] used the model to generate an approximate functional map by employing CNN and then adjusted the model to obtain an accurate instance segmentation mask in pixels using a reinforcement model. In [34–37], the researchers used context information and low-level features in CNN in various ways for better segmentation. Additionally, Zagoruko et al. [32] incorporated skip connection, foveal structure and integral loss in Fast R-CNN [18] for better segmentation.

In the traditional CNN, images with the same attributes but different context information obtain the same classification score. Previous models, especially fully convolutional network (FCN), used a single score map for semantic segmentation. For example, in segmentation, the model should allow the same image pixel of different instances with different context information to be segmented individually. Dai et al. [38]

incorporated the concept of relative position into FCN to differentiate multiple instances of an object by assembling a set of small score maps calculated at different relative positions of the object. Furthermore, Li et al. [39] extended the concept of [38] and introduced two different position detection score maps. SDS, hypercolumn, convolutional feature masking (CFM) [40], multi-task network cascade (MNC) [28] and multi path net [32] prevented the model from being a learnable end-to-end by using two different sub-networks for object detection and segmentation. Furthermore, works of Liang et al. [41] and Liu et al [42] expand instance segmentation by grouping or clustering FCN score maps with enormous post-processing, while Li et al. [39] introduced a joint formula of classification and division masking sub-networks in an efficient manner. Various researches [43–46] used a semantic segmentation model, whereas Mask R-CNN [47] extends Faster R-CNN, an object detection model, by adding a binary mask prediction branch for instance segmentation. In the work of Huang et al. [48], they qualitatively learned the predicted mask by injecting a network block into Mask R-CNN and proposed the Mask Scoring R-CNN. Recently, Kirillov et al. [49] used point-based rendering in Mask R-CNN and created a cutting-edge instance segmentation model. These researches [50,51] introduced a direction function that predicted different instances of a specific object. Uhrig et al. [50] employed a template matching method with distinct characteristics to extract the center of an instance and Chen et al. [51] obtained an instance by following the assembly process of [38,39]. References [16,44,52] used the function of forming a middle layer for better performance. Additionally, Liu et al. [53] used the concept of function propagation from the lower-level to the highest-level and built a state-of-the-art model based on Mask R-CNN. Newell et al. [54] used a novel method of using CNN along with association embedding for joint detection and grouping to process instance segmentation. Object detection using the sliding window approach provided a segmentation step and successful tasks such as Faster R-CNN, Mask R-CNN (e.g., SSD) [55] and RetinaNet [56] without using a quality improvement stage. The sliding window approach is widely used for object detection, but not for instance segmentation. Chen et al. [57] introduced dense instance segmentation to reduce this difference presented in Tensor Mask. With the numerous segmentation methods presented, we have applied the Pix2Pix method as the segmentation algorithm, which has recently shown stable performance in the paired dataset. This has the advantage of fast and stable segmentation without a separate masking task.

3 Methods

This study aims to develop an accurately semantic segmentation of the UI area on the game screen. The UI area must be labeled with a specific color to accomplish this goal. Acquiring such paired labeling data in large quantities requires a very high manual cost. Simple puzzle and arcade games do not have a large number of UI components in the game but have a large size of UI components and these components are separated from each other, making them relatively easy to label. However, in case of in-game genres such as MMORPG, there are dozens of small-sized UI components on the screen and as they are adjacent to each other, it is expensive to label them by hand. The size of the data is related to the accuracy of the deep learning network, therefore, making it difficult to develop an accurate semantic segmentation network if a large volume of data cannot be acquired. To solve this problem, we developed a tool that automatically detects UI components on the game screen. Through this, it is possible to obtain an image in which only the UI area is automatically labeled in a game screen. However, this tool must have a separate UI area and can only be applied to games that do not have an alpha value; hence, it cannot detect the UI area in all games. Therefore, labeling data can be acquired only in some of the target games. In this study, we tried to verify the functionality and capability of this tool when the image-to-image translation technique was applied to overcome the limitations. If we construct a large amount of labeling data with our tool and use it as target data for unsupervised learning, our tool has the advantage of translating source data with complex game UI components that cannot be labeled, to obtain semantic segmentation images. Fig. 1 shows our unsupervised learning concept.

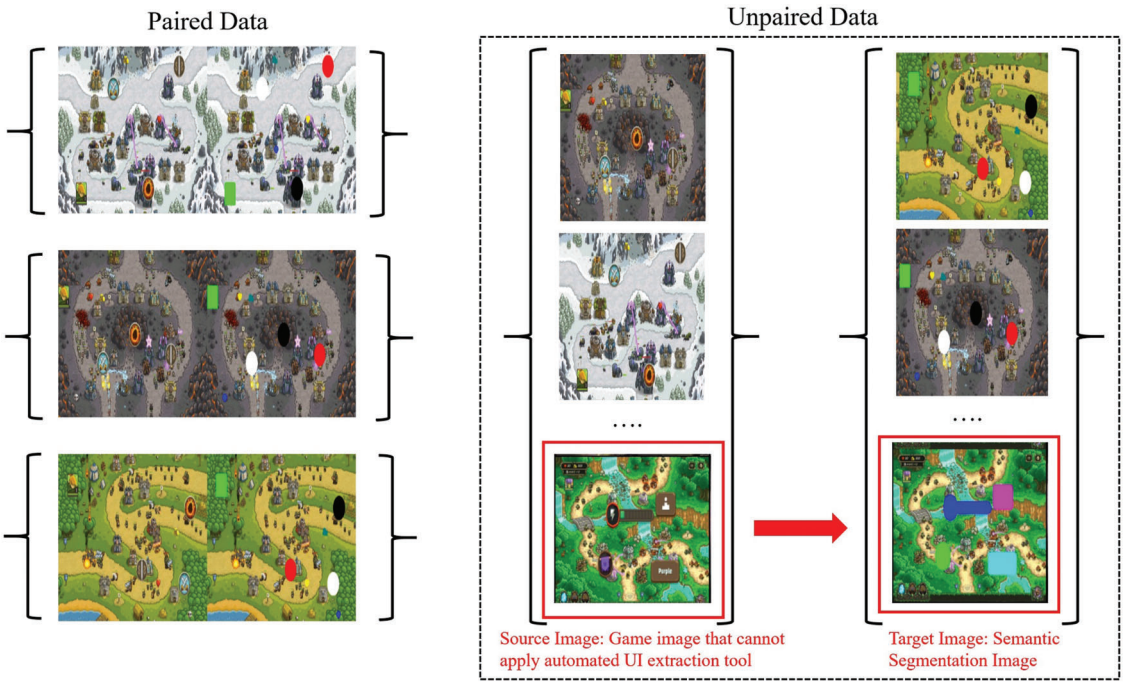

Figure 1: Proposed unsupervised learning concept

3.1 Paired UI Data Synthesis from Screenshots

To effectively generate image data, in this study, we automatically generated one million synthetic UI images using a commercial game engine. Using this, we automatically extracted the UI using three images with the same UI but different backgrounds on the game screen. The algorithm used for automatic UI extraction is as follows:

1. Inputting three game screenshots with the same UI but different backgrounds
2. Comparing the RGB values of all pixels in an image and removing pixels whose difference is higher than the threshold
3. Extracting the intersection of the remaining pixels by applying method 2 to screenshots 1 and 2, screenshots 1 and 3 and screenshots 2 and 3
4. UI segmentation and color designation using the Flood Fill algorithm
5. Filtering the small sized UI by regarding it as noise

In the experiment, the value of the threshold between 10% and 20% was appropriate. If the threshold is too small, the UI is not extracted properly. If it is too large, considerable noise are mixed. Some parts were not automatically filtered during the noise filtering process; thus, some manual intervention would be required. The UI within 10×10 pixels was regarded as noise and the extracted UI was designated as magenta. Magenta is a color that is not used to a large extent in the game; thus it is appropriate to be used in experiments. Next, we captured dozens of game images without the UI to be used as a background. Generally, the UI exists at the edge of the screen; thus, we selectively captured the inner part of the screen. After randomly selecting one of the captured background images using the aforementioned process, we randomly placed all the extracted UIs on the selected image. To increase the efficiency of the experiment, all UIs were processed such that they do not overlap. The synthetic UI data for the experiment were constructed in 256×256 size. Fig. 2a shows image data produced by randomly placing original UIs, while Fig. 2b shows image data produced by

coloring the placed UIs as magenta. By randomly placing all UIs not to overlap each other on an arbitrarily selected background, we automatically generated approximately 10,000 sheets of experimental image data in 10 minutes.

(a) (b)

Figure 2: Generated UI image data from the proposed tool, (a) colored UI, (b) original UI

3.2 Network

Our goal is to infer the UI area when we receive a general game screen as an input. UI components generally have a rectangular or circular shape on the screen. When the network recognizes the characteristics of these shapes effectively, it can exclude other image elements (e.g., game characters, game backgrounds) in the game screen and detect only the UI area. This means that the image-to-image translation network that we are going to use must be an extremely specialized network for shape recognition. For this, we used the UGATIT network model [58] that showed good performance in image-to-image translation as the baseline network.

The UGATIT network features an auxiliary classifier and AdaLIN; these two features of the UGATIT network have a significant advantage in shape modification, compared with existing image-to-image models. This study aims to develop a network that specializes in recognition of UI component. The features of UI component are considerably important, compared with other image transform domains. If proper learning is not performed, particularly with regard to the shape and area of the UIs, a large error occurs in the cognitive aspects, regardless of how well the other background areas are learned. Therefore, stable shape changes are necessary when UI-specific information is used by the network.

To achieve this, we incorporated an additional feature detail - geometry features of UI component. These details were obtained from a pre-trained network and it generated one loss for training the UGATIT Network. Geometry feature network is a network that extract features of basic shapes such as triangles, circles and squares. This is to take advantage of the fact that UI components such as buttons, window windows and icons are mainly composed of basic shapes. For the Geometry feature network learning, we created a paired synthetic image with only basic shapes colored on various random backgrounds. This paired synthetic image was classified as VGG16. For training, a total of 20,000 256×256 images were created. At this time, there are six categories—square, rectangle, circle, ellipse, round bar and round rectangle. After learning, the full connect layer was excluded from VGG and only the feature encoder network was used. The learned geometry feature network converts only the basic geometrical features detected in the

image to a 1D 256-dimensional vector when a specific image is entered and can be quantified as a loss value through the calculation of the cosine similarity between the converted vectors. Fig. 3 illustrates an example of the training dataset.

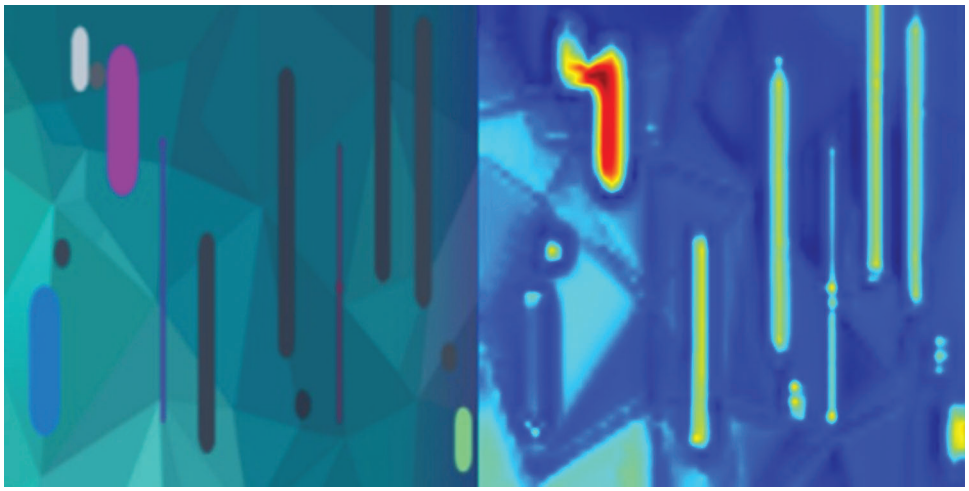

Figure 3: Synthetic dataset for geometry feature network training (round bar) and its detected features in feature network

The proposed network design is shown in Fig. 4. Let $x_s \in X_s$ and $x_t \in X_t$ represent samples from the source and target domains, respectively. Furthermore, let $G_1(x_s)$ and $G_2(x_t)$ represent the translated source and target domains, respectively. Our proposed model consists of two generators ($G_1(x_s)$ and $G_2(x_t)$), two discriminators ($D_1(G_1(x_s))$ and $D_2(G_2(x_t))$) and two feature extractors $F_1(x_s, x_t)$. $G_1(x_s)$ creates an image that fits the target style based on the GAN framework and $G_2(x_t)$ is used for cycle consistency. The discriminators D_1 and D_2 distinguish between real and fake translated images. The feature extractor F_1 provides a loss values to the CycleGAN framework to facilitate shape transformation. The final loss function of our model can be written as the loss of L_{total}.

$$arg \min_{G_1 G_2} \max_{D_1 D_2} L_{total}(G_1,\ G_2,\ D_1,\ D_2,\ F_1) \tag{1}$$

L_{total} comprises five loss terms: L_{lsgan}, L_{cycle}, $L_{identity}$, L_{cam}, and $L_{geometry}$. The adversarial loss L_{lsgan} is employed to match the distribution of the translated images to the target image distribution. The cycle loss L_{cycle} is applied for a cycle consistency constraint to the generator. The identity loss $L_{identity}$ is used to ensure that the color distributions of the input and output images are similar. These three losses are calculated using G_1, G_2, D_1 and D_2 with the traditional GAN framework. These terms are described in detail in [58,59]. L_{cam} uses information from the auxiliary classifiers to determine the differences between two domains [60].

The additional feature loss Lgeometry is the difference in 256-dimensional cosine similarity between the input image and the generated image. This value shows how similar the input image and the generated image are in terms of basic figure shape. Therefore, when creating a segmentation image, it induces the creation of similar images in terms of shape appearance as much as possible. $L_{geometry}$ loss can be applied differently according to the visual characteristics of each game by adjusting weight value α. Fig. 5 shows the network performance experiment results. When $L_{geometry}$ loss is applied, it is shown that the accuracy of segmentation increases in UIs with many shapes such as rectangles and circles. However, if the UI within

the screen to be applied is not classified into a simple geometry shape and the images are connected to each other, the effect of applying $L_{geometry}$ loss is weak.

$$L_{total} = L_{lsgan}(G_1,\ G_2,\ D_1,\ D_2) + L_{cycle}(G_1,\ G_2,\ D_1,\ D_2) + L_{identity}(G_1,\ G_2,\ D_1,\ D_2) + L_{cam}(G_1,\ G_2,\ D_1,\ D_2) + L_{geometry}(F_1,\ G_1,\ G_2) \tag{2}$$

$$L_{identity}(F_1) = \alpha cosine - similarity(F_1, G_1, G_2) \tag{3}$$

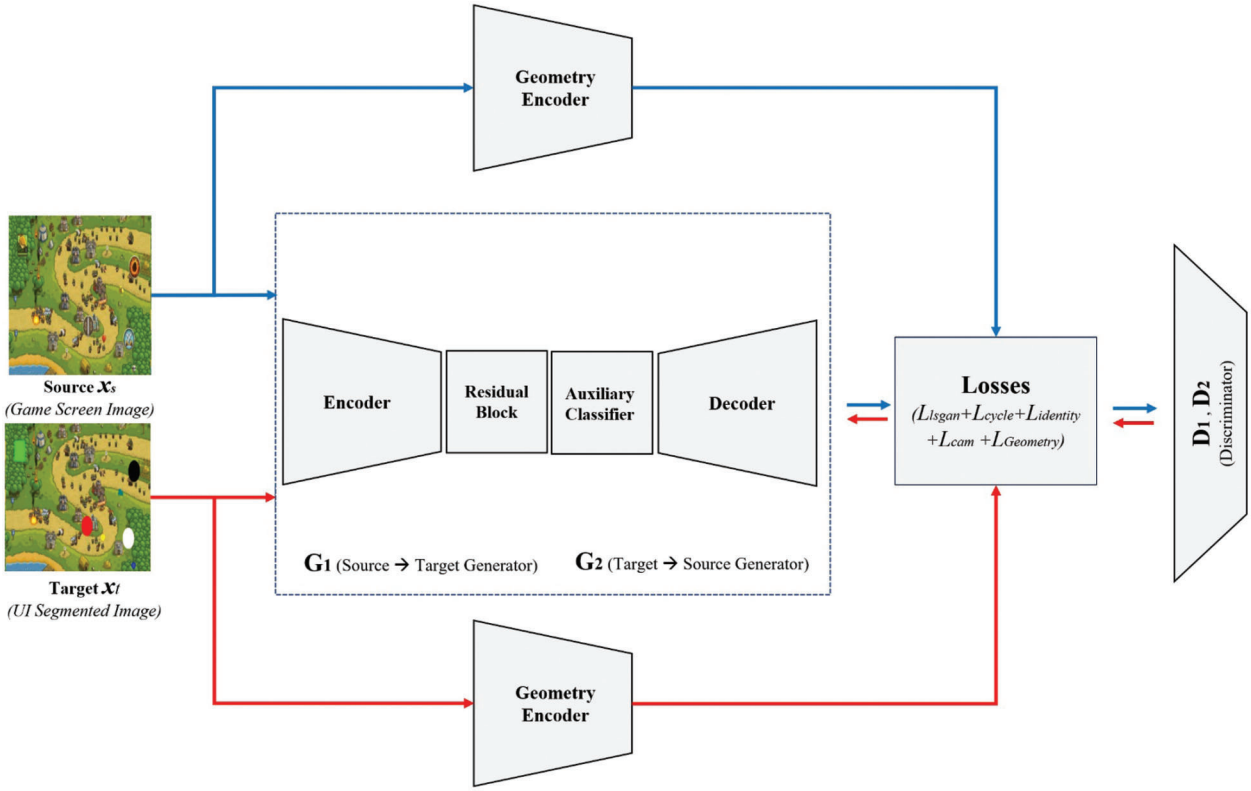

Figure 4: Proposed network architecture

4 Experimental Results

To verify the usefulness of this system, we tested it in three commercial games (Kingdom Rush, Iron Marine and Blade and Soul). The reason for choosing such a commercialized game was to confirm the practicality of the proposed technique when applied to an actual game. The three games differ in UI complexity. The segmentation number was 7 for Kingdom Rush, 8 for Iron Marine and 30 for Blade and Soul. These numbers can be interpreted as the numbers of UI groups that are geographically isolated from each other by the floor fill algorithm in the tool we use. We first created a simple screen capture program and captured screenshots of 5,000 images for each game. These screenshots were labeled with UI through the automatic UI extractor we suggested. The labeled image result is shown in Fig. 6.

The training took place for each game. All images were resized to 256 × 256 for training. For optimization, we set the maximum number of iterations to 200, the learning rate as 0.001 and its decay rate as 20% per 5 iterations. We used the SGD optimizer for training with the batch size = 8.

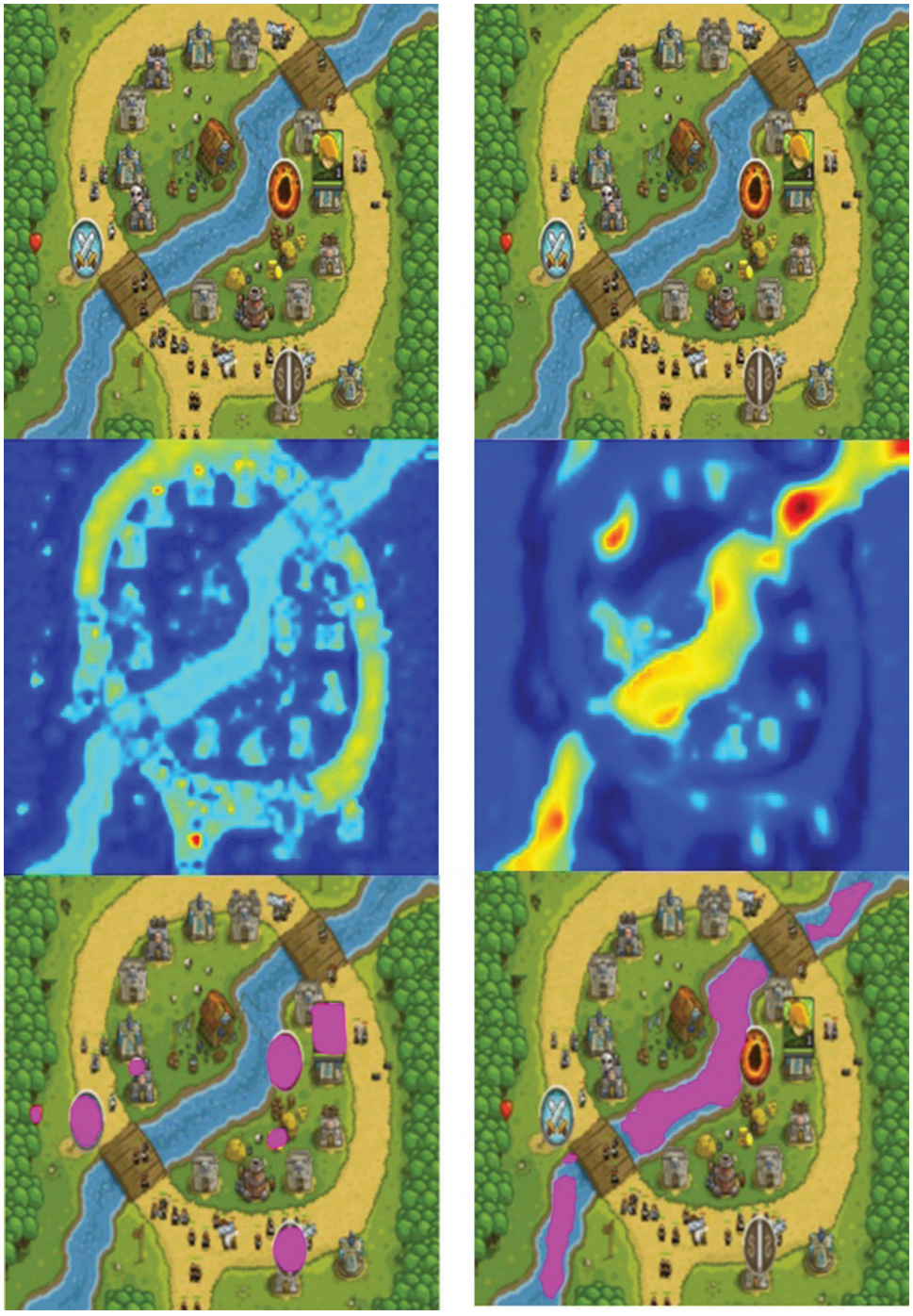

Figure 5: Network performance experiment results (left: segmentation result with $L_{geometry}$, right: segmentation result without $L_{geometry}$)

The NVIDIA RTX Titan GPU took approximately two days to perform a single training. The images resulting from creating segmentation images with the test set of each game using the trained network model are shown in Figs. 7–9. The ratio of the training, validation and test data set size was set at 6:2:2.

As this experiment recognizes the UI area in units of objects, instance segmentation of each UI unit module was not performed.

Figure 6: Auto-labeled training data set for each game

Fig. 7 shows the results of applying our system to Kingdom Rush. The tower defense game is a game in which a tower is placed in a designated location on the screen to defeat the attacking enemies. As this game is provided on a smart device and is based on monitor tapping, all UIs have a wider space between each other to facilitate touch. Furthermore, it has an area of relatively larger size to facilitate the recognition of finger-sized touches. The proposed network detects UI elements divided into 7 with a probability of more than 80%.

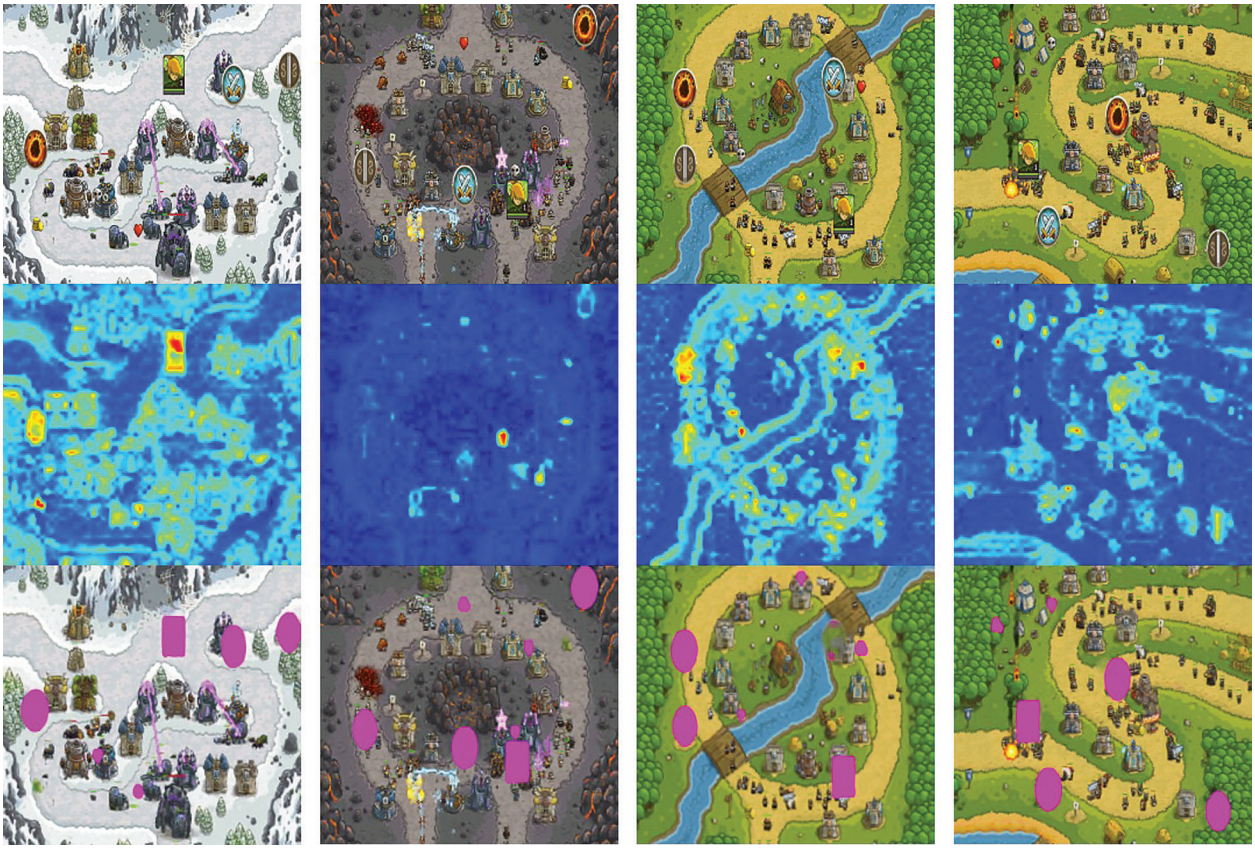

Figure 7: Result of Kingdom Rush UI segmentation

Fig. 8 shows the results of applying our system to Iron Marine. It is a tower defense game created in the same game development as Kingdom Rush. Unlike Kingdom Rush, where only towers are placed within a fixed screen, Iron Marine requires a user interface to move the screen and move the character. Because of this, the UI screen is more complicated and the UI modules are not separated and attached, making labeling relatively difficult. Our system was able to detect the UI area with 73.5% accuracy. These results show that our method works stably even with the unsupervised learning technique in games with approximately 7 - 8 touch-based interface components.

Fig. 9 shows the results of applying our system to PC MMORPG Blade and Soul. Blade and Soul is an action-based PC MMORPG game with complex types of information displayed on the screen where players interact by learning them. Therefore, unlike in the case of Kingdom Rush, the chat window, map screen and character information UI are further displayed on the screen. Moreover, the space between the UI components is smaller because the mouse, which allows fine control, is used as the primary interface. In this experiment, we attempted to confirm whether the system could automatically detect these complex UI components. The system automatically generated synthetic data segmented into 30 groups and trained the network through this. When applying the data trained in this manner to the original Blade and Soul, we could accurately detect the corresponding UI area with a pixel accuracy of 64.5%. The learning process of the network is stable although the number of the UI components detected is larger than that of Kingdom Rush. Unlike Kingdom Rush, in Blade and Soul, the UI components are relatively smaller and have high image details. Therefore, if the corresponding UI component has a similar color and shape to

the background image, it may be difficult for the network to detect it. However, our network demonstrates its capability to segment the UI components.

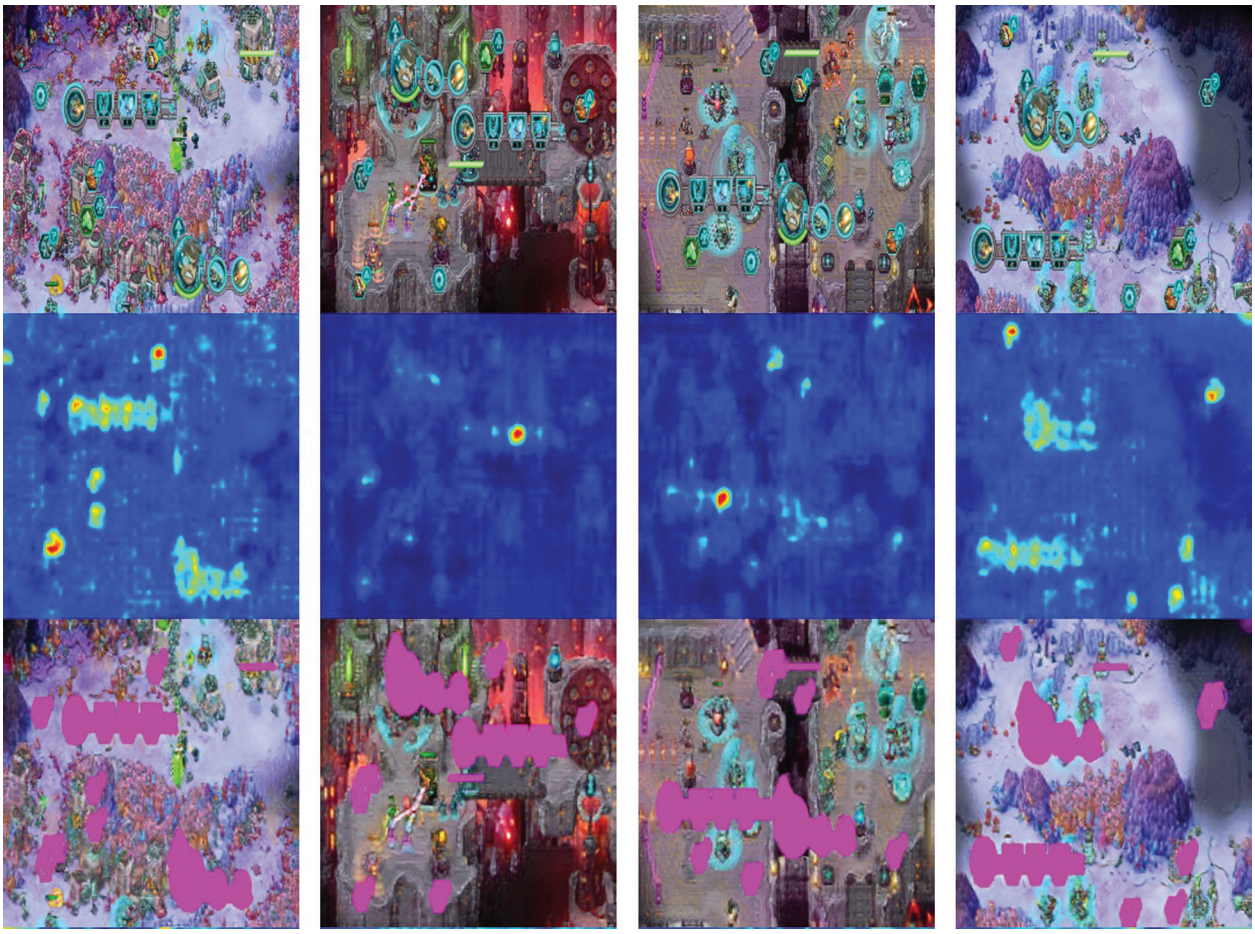

Figure 8: Result of Iron marine UI segmentation

Tab. 1 shows the overall segmentation performance of our network. First, we reduced the size of UI screens of the three games to 256 × 256; we then checked the number of segmented UI components, the size of the most basic UI button and the approximate spacing between each UI component and defined approximate complexity from this. We measured pixel accuracy, mean accuracy, mean IU and frequency weighted IU values for Kingdom Rush, Iron Marine and Blade and Soul. Pixel accuracy values were 0.816, 0.735 and 0.645, respectively. It is established that, overall, pixel accuracy reduced, being inversely proportional to the number of UI component classes we intend to classify. Since Kingdom Rush, a mobile game, is based on a touch-based UI, our system could automatically extract all UI components. However, the number of the UI components detected by the system for Blade and Soul and MMORPG games is approximately 65% less than the actual number of UI icons. This is because the UI components are sub-divided into skill units. Therefore, our system shows that UI segmentation is more effectively achieved in mobile games.

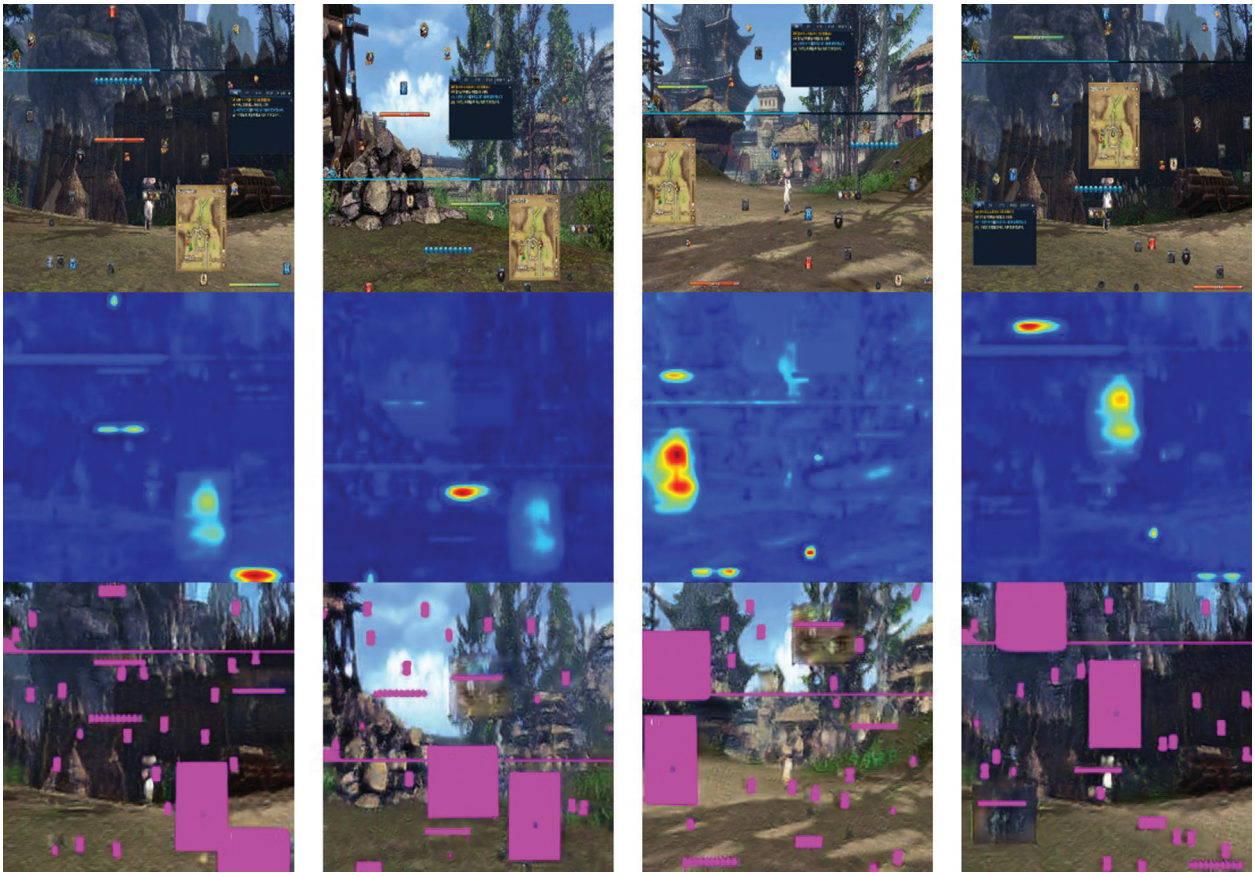

Figure 9: Result of Blade and Soul UI segmentation

Table 1: Evaluation metric for images produced by the generator for each game

	Kingdom Rush	Iron Marine	Blade and Soul
Number of Seg. UI Components	7	8	30
Size of a Single Button	Large (25×25)	Large (25×25)	Small (5×5)
Spacing between UI Components	Long (5 pixel)	Long (5 pixel)	Short (1-2 pixel)
Overall Complexity	Low	Low	High
Pixel Accuracy	0.816	0.735	0.645
Mean Accuracy	0.840	0.712	0.631
Mean IU	0.706	0.689	0.547
Frequency Weighted IU	0.691	0.612	0.514

Fig. 10 shows the result when the proposed method is applied to different games with a network trained with a dataset created by combining Kingdom Rush and Iron Marine data and Blade & Soul. It was applied to a total of four games (Starcraft1, Starcraft1 cartoon version, Fishdom, MMORPG V4). This experiment aims to determine how accurately the network can detect UI for other games of the same genre by training it with different game datasets. As a result of the experiment, in the casual game genre where the number of UIs is

small and the image complexity is low, the UI area could be detected with relatively high accuracy. However, The same was not possible in the case of MMORPG, which has a complex UI composition based on realistic images. Tab. 2 shows the overall segmentation performance of our network when our trained model are applied to different games in similar game genres.

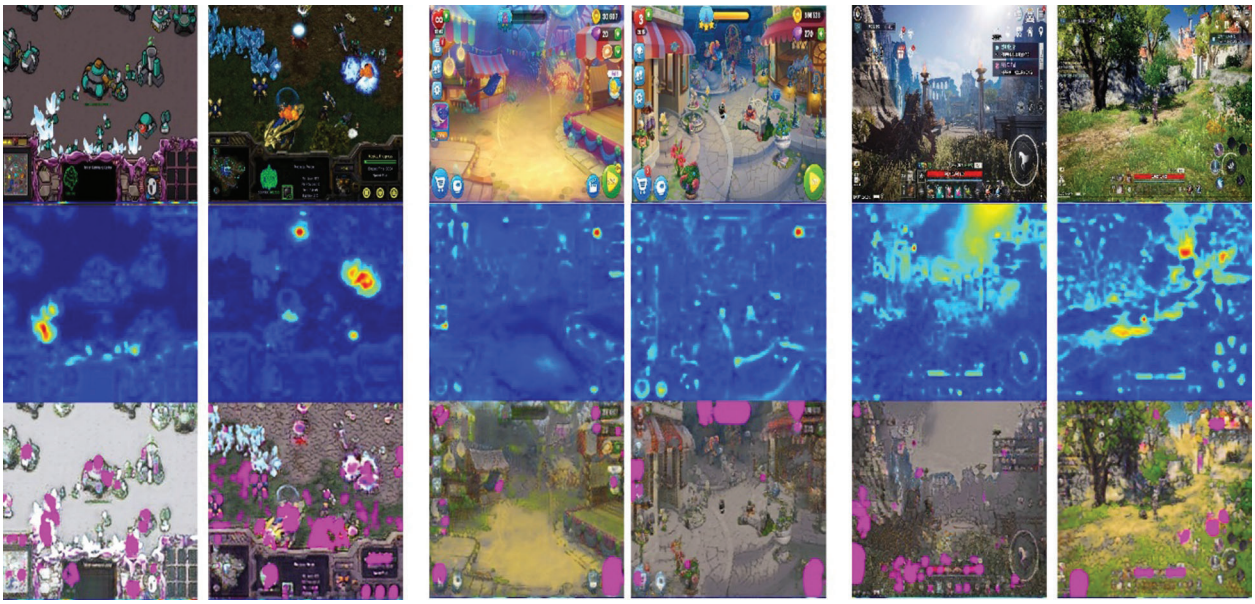

Figure 10: Result of various game segmentation with network from Iron Marine and Kingdom Rush dataset (left: Starcraft cartoon version & Original Starcraft (PC Game), Middle: Fishdom (Multi-Platform), Right: Mobile MMORPG V4 (Mobile)

Table 2: Pixel accuracy when applied to other games of similar genre

	Starcraft cartoon	Fishdom	Starcraft original	MMORPG V4
Network trained with Tower defense	0.63	0.67	0.48	0.45
Network trained with MMORPGs	0.31	0.37	0.39	0.44

5 Conclusions

In this paper, we introduced a method of segmenting only the UI area on the game image using the unsupervised learning technique. We have developed a semi-automatic labeling tool to identify the regions of interest and then assign labels to them, which is performed effectively on an arbitrary game screen. Moreover, an image-to-image translation network was presented that uses the feature information of the figure as the loss value. The image-to-image translation network, trained with data processed by our tool, showed stable segmentation results in casual games and MMORPG games. Additionally, the proposed method shows that there is a possibility that the UI area can be approximately segmented on similar game images of the same genre. Our research technique shows that UI segmentation is possible even though it is hard to create a paired dataset. This feature can be useful when a network needs to be trained based on a large amount of unpaired synthetic image data owing to labeling costs.

Funding Statement: This research was supported by the National Research Foundation of Korea (NRF) grant funded by the Korean government (MSIT) [Nos. NRF-2019R1A2C1002525, NRF-2020R1G1A1100125].

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou *et al.,* "Playing Atari with deep reinforcement learning," NIPS Deep Learning Workshop, Lake Tahoe, USA, 2013.

[2] E. Giacomello, P. L. Lanzi and D. Loiacono, "DOOM level generation using generative adversarial networks," in *Proc. IEEE Games, Entertainment, Media Conf. (GEM)*, Galway, Ireland, pp. 316–323, 2018.

[3] V. Volz, J. Schrum, J. Liu, S. M. Lucas, A. Smith *et al.,* "Evolving Mario levels in the latent space of a deep convolutional generative adversarial network," in *Proc. Genetic and Evolutionary Computation Conf.*, Kyoto, Japan, pp. 221–228, 2018.

[4] M. Awiszus, F. Schubert and B. Rosenhahnm, "TOAD-GAN: Coherent style level generation from a single example," in *Proc. AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, online*, vol. 16. no. 1. pp. 10–16, 2020, 2020.

[5] S. R. Richter, V. Vineet, S. Roth and V. Koltun, "Playing for data: Ground truth from computer games," in *Proc. European Conf. on Computer Vision*, Amsterdam, Netherlands, pp. 102–118, 2016.

[6] S. Kang and J. Choi, "Instance segmentation method of user interface component of games," *Applied Sciences*, vol. 10, no. 18, pp. 6502, 2020.

[7] S. R. Richter, V. Vineet, S. Roth and V. Koltun, "Playing for data: Ground truth from computer games," in *Proc. European Conf. on Computer Vision*, Amsterdam, Netherlands, Springer, pp. 102–118, 2016.

[8] C. Chen, A. Seff, A. Kornhauser and J. Xiao, "Deep driving: Learning affordance for direct perception in autonomous driving," in *Proc. the IEEE Int. Conf. on Computer Vision*, Santiago, Chile, pp. 2722–2730, 2015.

[9] G. R. Taylor, A. J. Chosak and P. C. Brewer, "Ovvv: Using virtual worlds to design and evaluate surveillance systems," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, Minneapolis, MN, USA, IEEE, pp. 1–8, 2007.

[10] J. Marin, D. V´azquez, D. Ger´onimo and A. M. L´opez, "Learning appearance in virtual scenarios for pedestrian detection," in *Proc. IEEE Computer Society Conf. on Computer Vision and Pattern Recognition*, San Francisco, CA, USA, pp. 137–144, 2010.

[11] D. Vazquez, A. M. Lopez, J. Marin, D. Ponsa and D. Geronimo, "Virtual and real world adaptation for pedestrian detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 4, pp. 797–809, 2013.

[12] N. Liao, M. Guzdial and M. Riedl, "Deep convolutional player modeling on log and level data," in *Proc. the 12th Int. Conf. on the Foundations of Digital Games*, Hyannis, MA, USA, pp. 1–4, 2017.

[13] Z. Wu, C. Shen and A. V. D. Hengel, "Bridging category-level and instance-level semantic image segmentation," arXiv preprint arXiv: 1605. 06885, 2016.

[14] B. Hariharan, P. Arbelaez, R. Girshick and J. Malik, "Simultaneous detection and segmentation," in *Proc. European Conf. on Computer Vision*, Zurich, Switzerland, Springer, pp. 297–312, 2014.

[15] R. Girshick, J. Donahue, T. Darrell and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proc. the IEEE Conf. on Computer Vision and Pattern Recognition*, Columbus, OH, USA, pp. 580–587, 2014.

[16] B. Hariharan, P. Ar'belaez, R. Girshick and J. Malik, "Hypercolumns for object segmentation and fine-grained localization," in *Proc. the IEEE Conf. on Computer Vision and Pattern Recognition*, Boston, MA, USA, pp. 447–456, 2015.

[17] K. He, X. Zhang, S. Ren and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 9, pp. 1904–1916, 2015.

[18] R. Girshick, "Fast R-CNN," in *Proc. the IEEE Int. Conf. on Computer Vision*, Santiago, Chile, pp. 1440–1448, 2015.

[19] J. R. Uijlings, K. E. Van De Sande, T. Gevers and A. W. Smeulders, "Selective search for object recognition," *International Journal of Computer Vision*, vol. 104, no. 2, pp. 154–171, 2013.

[20] D. Erhan, C. Szegedy, A. Toshev and D. Anguelov, "Scalable object detection using deep neural networks," in *Proc. the IEEE Conf. on Computer Vision and Pattern Recognition*, Columbus, OH, USA, pp. 2147–2154, 2014.

[21] C. Szegedy, S. Reed, D. Erhan, D. Anguelov, S. Ioffe *et al.,* "Scalable, high-quality object detection," arXiv preprint arXiv:1412.1441, 2014.

[22] W. Kuo, B. Hariharan and J. Malik, "Deepbox: Learning objectness with convolutional networks," in *Proc. the IEEE Int. Conf. on Computer Vision*, Santiago, Chile, pp. 2479–2487, 2015.

[23] C. L. Zitnick and P. Doll'ar, "Edge boxes: Locating object proposals from edges," in *Proc. European Conf. on Computer Vision*, Zurich, Switzerland, Springer, pp. 391–405, 2014.

[24] S. Ren, K. He, R. Girshick and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 6, pp. 1137–1149, 2016.

[25] P. Arbelaez, J. Pont-Tuset, J. T. Barron, F. Marques and J. Malik, "Multiscale combinatorial grouping," in *Proc. the IEEE Conf. on Computer Vision and Pattern Recognition*, Columbus, OH, USA, pp. 328–335, 2014.

[26] P. O. Pinheiro, R. Collobert and P. Dollar, "Learning to segment object candidates," in *Proc. 28th International Conference on Neural Information Processing Systems*, vol. 2, pp. 1990–1998, 2015.

[27] J. Carreira and C. Sminchisescu, "CPMC: Automatic object segmentation using constrained parametric min-cuts," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 7, pp. 1312–1328, 2011.

[28] J. Dai, K. He and J. Sun, "Instance-aware semantic segmentation via multi-task network cascades," in *Proc. the IEEE Conf. on Computer Vision and Pattern Recognition*, Las Vegas, NV, USA, pp. 3150–3158, 2016.

[29] J. Long, E. Shelhamer and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proc. the IEEE Conf. on Computer Vision and Pattern Recognition*, Boston, MA, USA, pp. 3431–3440, 2015.

[30] S. Xie and Z. Tu, "Holistically-nested edge detection," in *Proc. the IEEE Int. Conf. on Computer Vision*, Santiago, Chile, pp. 1395–1403, 2015.

[31] P. Sermanet, K. Kavukcuoglu, S. Chintala and Y. LeCun, "Pedestrian detection with unsupervised multi-stage feature learning," in *Proc. the IEEE Conf. on Computer Vision and Pattern Recognition*, Portland, Oregon, USA, pp. 3626–3633, 2013.

[32] S. Zagoruyko, A. Lerer, T. Y. Lin, P. O. Pinheiro, S. Gross *et al.,* "A multipath network for object detection," in *Proc. the British Machine Vision Conference (BMVC)*, York, UK, BMVA Press, pp. 15.1–15.12, 2016.

[33] P. O. Pinheiro, T. Y. Lin, R. Collobert and P. Dollar, "Learning to refine object segments," in *Proc. European Conf. on Computer Vision*, Amsterdam, Netherlands, Springer, pp. 75–91, 2016.

[34] A. Torralba, "Contextual priming for object detection," *International Journal of Computer Vision*, vol. 53, no. 2, pp. 169–191, 2003.

[35] P. Sermanet, K. Kavukcuoglu, S. Chintala and Y. LeCun, "Pedestrian detection with unsupervised multi-stage feature learning," in *Proc. the IEEE Conf. on Computer Vision and Pattern Recognition*, Portland, Oregon, USA, pp. 3626–3633, 2013.

[36] S. Bell, C. L. Zitnick, K. Bala and R. Girshick, "Inside-outside net: Detecting objects in context with skip pooling and recurrent neural networks," in *Proc. the IEEE Conf. on Computer Vision and Pattern Recognition*, Las Vegas, NV, USA, pp. 2874–2883, 2016.

[37] D. Hoiem, Y. Chodpathumwan and Q. Dai, "Diagnosing error in object detectors," in *Proc. European Conf. on Computer Vision*, Firenze, Italy, Springer, pp. 340–353, 2012.

[38] J. Dai, Y. Li, K. He and J. Sun, "R-FCN: Object detection via region-based fully convolutional networks," in *Proc. the 30th International Conference on Neural Information Processing Systems*, Barcelona Spain, pp. 379–387, 2016.

[39] Y. Li, H. Qi, J. Dai, X. Ji and Y. Wei, "Fully convolutional instance-aware semantic segmentation," in *Proc. the IEEE Conf. on Computer Vision and Pattern Recognition*, Honolulu, HI, USA, pp. 2359–2367, 2017.

[40] J. Dai, K. He and J. Sun, "Convolutional feature masking for joint object and stuff segmentation," in *Proc. the IEEE Conf. on Computer Vision and Pattern Recognition*, Boston, MA, USA, pp. 3992–4000, 2015.

[41] X. Liang, L. Lin, Y. Wei, X. Shen, J. Yang *et al.,* "Proposal-free network for instance-level object segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 12, pp. 2978–2991, 2017.

[42] S. Liu, X. Qi, J. Shi, H. Zhang and J. Jia, "Multi-scale patch aggregation (MPA) for simultaneous detection and segmentation," in *Proc. the IEEE Conf. on Computer Vision and Pattern Recognition*, Las Vegas, NV, USA, pp. 3141–3149, 2016.

[43] A. Arnab and P. H. Torr, "Pixelwise instance segmentation with a dynamically instantiated network," in *Proc. the IEEE Conf. on Computer Vision and Pattern Recognition*, Honolulu, HI, USA, pp. 441–450, 2017.

[44] M. Bai and R. Urtasun, "Deep watershed transform for instance segmentation," in *Proc. the IEEE Conf. on Computer Vision and Pattern Recognition*, Honolulu, HI, USA, pp. 5221–5229, 2017.

[45] A. Kirillov, E. Levinkov, B. Andres, B. Savchynskyy and C. Rother, "InstanceCut: From edges to instances with MultiCut," in *Proc. the IEEE Conf. on Computer Vision and Pattern Recognition*, Honolulu, HI, USA, pp. 5008–5017, 2017.

[46] K. He, G. Gkioxari, P. Dollar and R. Girshick, "Mask R-CNN," in *Proc. the IEEE Int. Conf. on Computer Vision*, Venice, Italy, pp. 2961–2969, 2017.

[47] S. Liu, J. Jia, S. Fidler and R. Urtasun, "SGN: Sequential grouping networks for instance segmentation," in *Proc. the IEEE Int. Conf. on Computer Vision*, Venice, Italy, pp. 3496–3504, 2017.

[48] Z. Huang, L. Huang, Y. Gong, C. Huang and X. Wang, "Mask scoring R-CNN," in *Proc. the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Long Beach, CA, USA, pp. 6409–6418, 2019.

[49] A. Kirillov, Y. Wu, K. He and R. Girshick, "PointRend: Image segmentation as rendering," in *Proc. the IEEE/CVF Conf. on Computer Vision and Pattern Recognition*, Seattle, WA, USA, pp. 9799–9808, 2020.

[50] J. Uhrig, M. Cordts, U. Franke and T. Brox, "Pixel-level encoding and depth layering for instance-level semantic labeling," in *Proc. German Conf. on Pattern Recognition*, Hannover, Germany, Springer, pp. 14–25, 2016.

[51] L. C. Chen, A. Hermans, G. Papandreou, F. Schroff, P. Wang *et al.,* "Masklab: Instance segmentation by refining object detection with semantic and direction features," in *Proc. the IEEE Conf. on Computer Vision and Pattern Recognition*, Salt Lake City, UT, USA, pp. 4013–4022, 2018.

[52] T. Kong, A. Yao, Y. Chen and F. Sun, "Hypernet: Towards accurate region proposal generation and joint object detection," in *Proc. the IEEE Conf. on Computer Vision and Pattern Recognition*, Las Vegas, NV, USA, pp. 845–853, 2016.

[53] S. Liu, L. Qi, H. Qin, J. Shi and J. Jia, "Path aggregation network for instance segmentation," in *Proc. the IEEE Conf. on Computer Vision and Pattern Recognition*, Salt Lake City, UT, USA, pp. 8759–8768, 2018.

[54] A. Newell, Z. Huang and J. Deng, "Associative embedding: End-to-end learning for joint detection and grouping," in *Proc. the 31st International Conference on Neural Information Processing Systems*, Long Beach California USA, pp. 2274–2284, 2016.

[55] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed *et al.,* "SSD: Single shot multibox detector," in *Proc. European Conf. on Computer Vision*, Amsterdam, Netherlands, Springer, pp. 21–37, 2016.

[56] T. Y. Lin, P. Goyal, R. Girshick, K. He and P. Dollar, "Focal loss for dense object detection," in *Proc. the IEEE Int. Conf. on Computer Vision*, Venice, Italy, pp. 2980–2988, 2017.

[57] X. Chen, R. Girshick, K. He and P. Dollar, "Tensormask: A foundation for dense object segmentation," in *Proc. the IEEE/CVF Int. Conf. on Computer Vision*, Seoul, Korea, pp. 2061–2069, 2019.

[58] J. Kim, M. Kim, H. Kang and K. Lee, "U-GAT-IT: Unsupervised generative attentional networks with adaptive layer-instance normalization for image-to-image translation," in *Proc. International Conference on Learning Representations, virtual conference*, pp. 1–10, 2020.

[59] J. Y. Zhu, T. Park, P. Isola and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Proc. the IEEE Int. Conf. on Computer Vision*, Venice, Italy, pp. 2223–2232, 2017.

[60] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh *et al.,* "Grad-CAM: Visual explanations from deep networks via gradient-based localization," in *Proc. the IEEE Int. Conf. on Computer Vision*, Venice, Italy, pp. 618–626, 2017.