
Article:
Ioannou, E. and Maddock, S. (2026) PQDAST: Depth-aware arbitrary style transfer for games via perceptual quality-guided distillation. IEEE Transactions on Games. ISSN: 2475-1502

https://doi.org/10.1109/TG.2026.3660906

PQDAST: Depth-Aware Arbitrary Style Transfer for Games via Perceptual Quality-Guided Distillation

***Abstract*—Artistic style transfer is concerned with the generation of imagery that combines the content of an image with the style of an artwork. In the realm of computer games, most work has focused on post-processing video frames. Some recent work has integrated style transfer into the game pipeline, but it is limited to single styles. Integrating an arbitrary style transfer method into the game pipeline is challenging due to the memory and speed requirements of games. We present *PQDAST*, the first solution to address this. We use a perceptual quality-guided knowledge distillation framework and train a compressed model using the ꟻLIP evaluator, which substantially reduces both memory usage and processing time with limited impact on stylisation quality. For better preservation of depth and fine details, we utilise a synthetic dataset with depth and temporal considerations during training. The developed model is injected into the rendering pipeline to further enforce temporal stability and avoid diminishing post-process effects. Quantitative and qualitative experiments demonstrate that our approach achieves superior performance in temporal consistency, with comparable style transfer quality, to state-of-the-art image, video and in-game methods.**

***Index Terms*—Neural style transfer, computer games, G-buffer, neural network compression, graphics pipeline.**

I. Introduction

Arbitrary Neural Style Transfer (NST) uses the style of any artwork to alter content data such as images [1]–[4], videos [5]–[7] and radiance fields [8]–[12]. In the realm of computer games, image and video NST methods can be utilised as post-processing effects at the end of a game's rendering pipeline. Nevertheless, this treats artistic stylisation as a final filter, ignoring the 3D nature of a computer game's scene, which can result in temporal instabilities and undesired flickering effects. Recently, work has focused on artistic style transfer specifically tailored for games [13], [14], but it is constrained to a single style (a separate network must be trained for each desired style).

While single-style-per-model methods can yield consistent results, arbitrary style transfer offers greater flexibility, particularly in scenarios where a user may wish to experiment with multiple styles or switch styles in real time. This capability is valuable for both designers during development and end-users in interactive contexts. However, a challenge is to maintain high stylisation quality while addressing inherent speed and memory issues. Using intermediate (G-buffer) information that becomes available during the rendering process shows promise for addressing this. Recent methods have demonstrated remarkable improvements in the quality of generated stylised game scenes using G-buffer data [15], [16], while other approaches have integrated NST as part of the 3D computer graphics pipeline to alleviate the issue of temporal incoherence across subsequent frames [13], [14]. These methods avoid applying a trained image or video style transfer approach at the end of the rendering process, however, they utilise a conventional convolutional-based transformation network that is only capable of reproducing one artistic style.

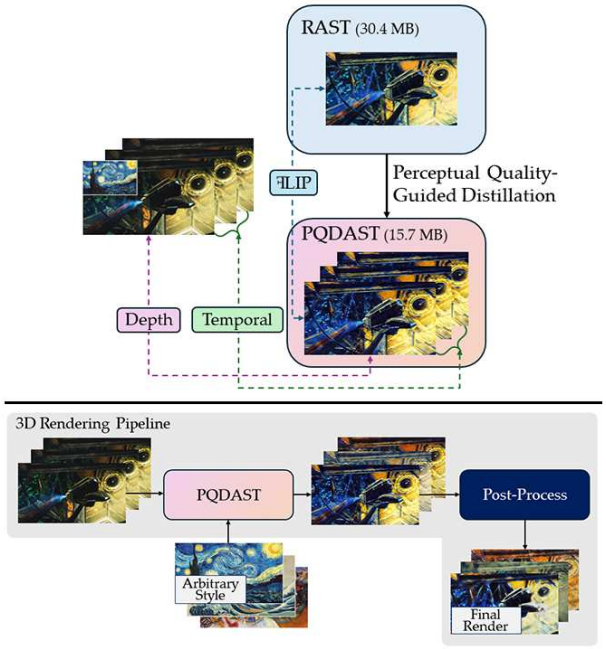


Fig. 1. Top: Our proposed perceptual quality-guided knowledge distillation framework utilises the ꟻLIP evaluator where RAST is the teacher model and PQDAST our proposed method. Depth and temporal loss functions are also defined. Bottom: The trained model is injected into the 3D rendering pipeline.

In this paper, as illustrated in Figure 1, we introduce *PQDAST*, which, to our knowledge, is the first arbitrary style transfer approach that is injected in the game's rendering pipeline [13], [14]. In most cases, algorithms that are capable of reproducing an arbitrary style per trained network [2]–[4] first extract image features of content and style and then do a forward pass through a trained transformer network before a decoder generates the stylised result. We propose a solution that is based on the approach of Ma et al. [4], which involves training a compressed transformer and decoder network using knowledge distillation. We devise a new loss function that is inspired by work on image quality assessment [17] to force the compressed models to retain the quality of the original model. Our novel perceptual quality-guided distillation loss illustrates how image quality assessment research can contribute to model compression for effectively minimising the speed and memory of algorithms dedicated to image generation. Additionally, we utilise an advanced depth estimation network [18] for a depth reconstruction loss that was previously shown to improve the quality of style transfer

results [19], [20]. Instead of training on an image dataset, we train on a synthetic video dataset and employ a temporal loss function for improved temporal stability. The contributions of our work can be summarised as follows:

- We propose a solution for arbitrary style transfer in computer games, enabling users to apply any painting to artistically alter the visuals of the game.
- We present a technique that distils the model of a previous approach [4] into a smaller network, approximately half its size. Our algorithm utilises the ƎLIP evaluator in a novel perceptual quality-guided knowledge distillation technique that achieves comparable stylisation quality and improved temporal stability compared to state-of-the-art methods.
- Our developed network is integrated into the computer game's rendering pipeline (similar to [13], [14]), resulting in enhanced temporal consistency.
- Extensive qualitative and quantitative experiments demonstrate that our proposed algorithm achieves high-quality arbitrary style transfer for computer games.

II. Related Work

A. Neural Style Transfer

NST research has progressed from online image-optimization techniques [21], to offline model-optimization methods capable of reproducing one style per trained network [22]–[25], and arbitrary-style-per-model approaches [1] that can reproduce any given referenced style image on an input photograph [26]–[31]. Early work on arbitrary style transfer, *AdaIN* [1], used an adaptive instance normalization layer that aligns the mean and variance of the content features with the respective mean and variance of style features. Other work suggested patch-based techniques [32], [33], while neural flows [34] and vector quantization [35] have also been exploited for arbitrary stylisation. Recently, the success of attention mechanisms [36], [37] in computer vision has resulted in multiple attention-based methods [2]–[4], [38]–[41], as well as diffusion model-based methods [42], [43] for artistic style transfer. Among these attention-based approaches, *RAST* [4], a system inspired by image restoration shows enhanced structure preservation, a desirable quality in a game setting. Our approach utilises *RAST* (which uses *SANet* [2] as a backbone) in a distillation framework that is also based on style-attentional networks (*SANet*).

Temporal incoherence is the main challenge that arises when stylising videos compared to images. Methods have resorted to optic flow data to improve temporal stability [44], [45]. Multiple-style-per-network models [46] and arbitrary-style-per-network models [6], [7] have been proposed, while depth-aware and structure-preserving video style transfer [47]–[49] attempts to retain depth and global structure of the stylised video frames. Image style transfer approaches have been extended to work for videos with additional temporal loss training [3], [5], and unified frameworks for joint image and video style transfer techniques have been developed [50], [51]. Diffusion-based methods for stylised video generation have also emerged [52].

B. Style Transfer for 3D Computer Games

Whilst image and video NST methods can be applied at the end of the rendering pipeline to achieve real-time computer game stylisation, this is essentially a post-processing effect that interprets the rendered frames as single images and does not prevent undesired artifacts and flickering issues. Multi-style artistic style transfer for games has been shown in work by Unity [53] – this utilises the method of Ghiasi et al. [26] to stylise each intercepted final rendered image. Any G-buffer or 3D data is ignored while the produced stylisations are inconsistent and the post-process effects are diminished, as the stylisation network is used as a final 'filter'. Other approaches have demonstrated improved stylisation quality when G-buffer data is taken into account during training [15], [16]. Style transfer specifically tailored for computer games has only been recently proposed [13], [14], [54]. Here, NST is injected into the rendering pipeline before the post-process stage but is only capable of reproducing one style image per trained network. Yet, arbitrary style transfer could offer a significant advantage to developers and artists, as well as enable users to upload any artwork of their choice to stylise the game scenes.

C. Knowledge Distillation

Pioneered by Hinton et al. [55], knowledge distillation has been a widely adopted technique for training compressed models. This aims to create smaller and faster models that retain quality and performance. Recently, methods have leveraged this technique for the task of style transfer, demonstrating improved performance [56]–[58]. Wang et al. [59] show that training a smaller encoder to replace the large *VGG-19* [60] that is typically utilised in encoder-decoder-based neural style transfer results in ultra-resolution outputs that were hard to achieve before due to memory constraints. High-quality arbitrary style transfer for images is also achieved by designing a network composed of a content encoder, a style encoder and a decoder based on CNNs, and employing symmetric knowledge distillation [58]. The method by Chen et al. [57] – also based on a simple CNN architecture – achieves fast video style transfer without relying on optic flow information during inference, but is only capable of reproducing one style per trained network.

III. Our Approach

Figure 2 provides an overview of the proposed system architecture. We adopt a widely used encoder-decoder design; we train a transformer that encompasses compressed versions of the SANet module, and a small decoder to produce comparable results to *RAST* that uses *SANet* as a backbone. In addition to distillation losses, we employ a temporal loss [3] and a depth reconstruction loss [20] for stable and high-quality results.

A. Preliminaries: SANet, RAST

The recent technique by Ma et al. [4], [61], based on *SANet* [2], has shown remarkable performance in terms of alleviating the Content Leak phenomenon [34]. *RAST* [4], due to its image restoration capabilities, achieves a high perceptual similarity

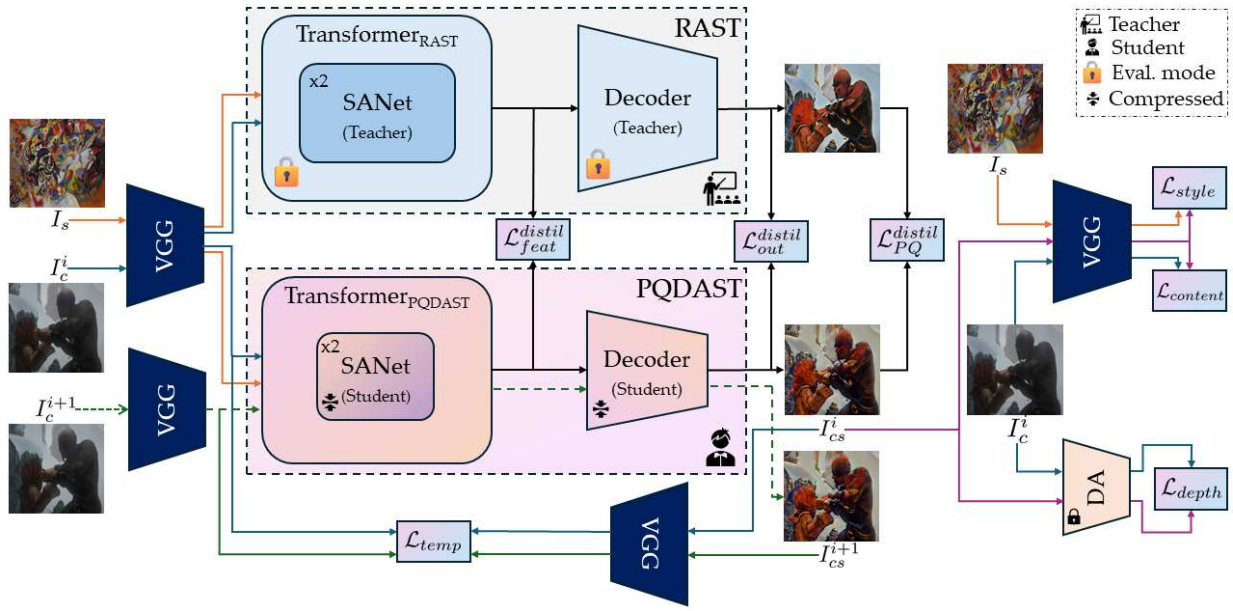


Fig. 2. Overview of PQDAST Architecture. PQDAST distils knowledge from RAST (comprising two SANet modules and a decoder), which serves as the teacher model. Features extracted from VGG are used to compute content ($\mathcal{L}_{content}$), style ($\mathcal{L}_{style}$), and temporal ($\mathcal{L}_{temp}$) losses, while depth predictions from DepthAnything (DA) provide the depth reconstruction loss ($\mathcal{L}_{depth}$). PQDAST trains compressed versions of RAST's decoder and SANet modules using perceptual quality-guided distillation losses ($\mathcal{L}_{*}^{distil}$), alongside these additional objectives.

score which means that fine details are efficiently preserved. This is desirable in the context of computer games. In addition, *RAST* utilises two external discriminators to handle realistic-to-artistic and artistic-to-realistic processes. This ability of the model to stylise images in both directions, combined with its capability to handle photorealistic as well as artistic style transfer, makes it suitable for games that may feature a non-photorealistic style or strive for photorealism. We therefore design a technique that distils the knowledge of *RAST* to a compressed model.

RAST uses *SANet* as a backbone. Assuming content image I_c and style image I_s, and given encoded content and style feature maps F_c and F_s, from a pre-trained *VGG* [60], the SANet module transforms them into two feature spaces f and g and calculates the attention between $\overline{F_c^i}$ and $\overline{F_s^i}$, where $\overline{F}$ denotes mean-variance channel-wise normalised version of F:

$$F_{cs}^i = \frac{1}{C(F)} \sum_{\forall j} exp(f(\overline{F_c^i})^T g(\overline{F_s^j})) h(F_s^j), \quad (1)$$

where $f(\overline{F_c}) = W_f \overline{F_c}$, $g(\overline{F_s}) = W_g \overline{F_s}$ and $h(F_s) = W_h F_s$ are learned weight matrices implemented as 1×1 convolutions.

This output feature map is then used to obtain F_{csc}:

$$F_{csc} = F_c + W_{cs} F_{cs} \quad (2)$$

Two SANet modules are used for features extracted from layers $relu4\_1$, and $relu5\_1$ of *VGG*, respectively. The outputs of the two SANets are then combined:

$$F_{csc}^m = conv_{3\times3}(F_{csc}^{relu4\_1} + upsampling(F_{csc}^{relu5\_1})), \quad (3)$$

before the decoder synthesises the final output:

$$I_{cs} = D(F_{csc}^m). \quad (4)$$

TABLE I
THE NETWORK ARCHITECTURE OF THE ORIGINAL STYLE-ATTENTIONAL NETWORK (SANET) COMPARED TO OUR PROPOSED COMPRESSED SANET USED IN PQDAST.

	SANet	**PQDAST**
Layer	Features In → Features Out	
Conv (f)	512 → 512	512 → 256
Conv (g)	512 → 512	512 → 256
Conv (h)	512 → 512	512 → 256
Conv (out)	512 → 512	256 → 512

Our neural network architecture resembles the architecture of *SANet* but it has significantly reduced complexity. As the SANet module is used twice, we define a student transformer network that utilises a student SANet module with reduced feature maps of each convolution layer, as shown in Table I. This reduces the floating point operations performed (FLOPs) from 15.05G to 12.35G and the number of parameters from 4.46M to 3.41M for the transformer block that combines the outputs of the two SANet modules. In addition, we compress the decoder network from 9 convolutional layers (63.36G FLOPs, 3.51M parameters) to 4 convolutional layers (6.51G FLOPs, 702.40K parameters), as illustrated in Figure 3.

B. Perceptual Quality-Guided Knowledge Distillation

Our proposed framework is trained using a combination of three distillation losses. As our transformer is less complex than the transformer used in *RAST*, combining the outputs of two SANet modules, we initially define a loss that minimises the error between the outputs of the *RAST*'s transformer

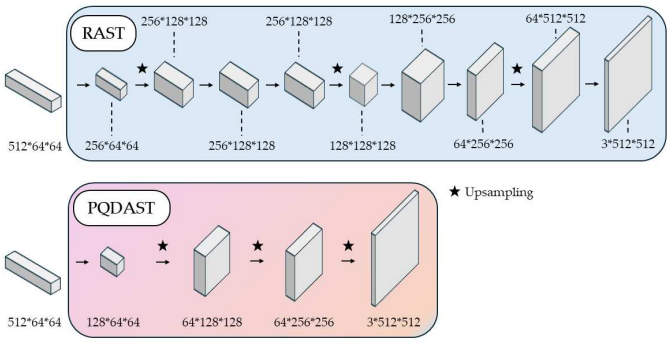


Fig. 3. Decoder Architecture: *RAST* vs *PQDAST*.

($F^m_{csc_{teacher}}$) and *PQDAST*'s transformer ($F^m_{csc_{student}}$):

$$\mathcal{L}^{distil}_{feat} = \|F^m_{csc_{student}} - F^m_{csc_{teacher}}\|^2_2 \tag{5}$$

We repeat the same for the outputs of the decoders:

$$\mathcal{L}^{distil}_{out} = \|I_{cs_{student}} - I_{cs_{teacher}}\|^2_2 \tag{6}$$

The plethora of image and video style transfer methods do not optimise for applicability in computer games. In addition to temporal considerations that are useful for games, we also utilise a synthetic video dataset to better capture the synthetic nature of the visual media our algorithm intends to stylise. Nevertheless, it is necessary for our approach to remain perceptually consistent to *RAST*, which achieves good results in terms of sustaining fine details. Training on a different dataset (and with a less complex model) would result in dissimilarities between the outputs. To better match the output of *RAST*, we define a distillation loss based on perceptual quality. For this, we treat the output of our model as a 'rendered image' that is an attempt to reproduce the output of *RAST*, which can be considered as the 'ground truth'.

For this task, we use ꟻLIP, as SSIM has been shown to lack the necessary qualities for use with colour images [62]. ꟻLIP's architecture is based on a colour pipeline and a feature pipeline, resulting in an image quality metric that performs competently against state-of-the-art methods and coincides with human judgement. The specific focus of ꟻLIP on rendering quality makes it particularly well-suited for our method. While many image quality metrics are designed for general image comparison, ꟻLIP is tailored to assess the types of artefacts and differences commonly encountered in rendered images. This specialization is essential for our distillation loss, as it allows us to penalize precisely those visual discrepancies that are most likely to be noticed by viewers. This targeted approach ensures that our distilled model learns to prioritize the aspects of image quality most relevant to rendering, leading to more visually compelling results. We, thus, utilise ꟻLIP to define an additional perceptual quality-guided distillation loss:

$$\mathcal{L}^{distil}_{PQ} = ꟻLIP(I_{cs_{student}}, I_{cs_{teacher}}), \tag{7}$$

with the resulting total distillation loss defined as:

$$\mathcal{L}^{distil}_{total} = \mathcal{L}^{distil}_{feat} + \mathcal{L}^{distil}_{out} + \mathcal{L}^{distil}_{PQ}. \tag{8}$$

C. Depth and Temporal Considerations

Similarly to previous techniques optimised for computer games [13], [14], we employ a depth reconstruction loss to reinforce the retainment of depth in the synthesised results – depth data has been consistently shown to enhance the quality of artistically stylised imagery [19], [20]. Unlike previous methods [13], [14], we adopt the recent method of Yang et al. [18] ("*Depth Anything*", here, denoted as *DA*) which demonstrates improved performance compared to *MiDaS* [63]. The depth reconstruction loss is thus formulated as:

$$\mathcal{L}^{DA}_{depth}(I_{cs}, I_c) = \|DA(I_{cs}) - DA(I_c)\|^2_2. \tag{9}$$

Additionally, the proposed system, trained on synthetic video data, allows for temporal considerations. Our temporal loss ($\mathcal{L}_{temp}$) is adopted from Liu et al. [3].

D. Full Training Objective

The overall loss function our system optimises is a weighted summation of the knowledge distillation loss $\mathcal{L}^{distil}_{total}$, depth loss $\mathcal{L}^{DA}_{depth}$, temporal loss $\mathcal{L}_{temp}$ and perceptual (content $\mathcal{L}_{content}$ and style $\mathcal{L}_{style}$) losses:

$$\mathcal{L}_{total} = \lambda_c\mathcal{L}_{content} + \lambda_s\mathcal{L}_{style} + \lambda_k\mathcal{L}^{distil}_{total} + \lambda_d\mathcal{L}^{DA}_{depth} + \lambda_t\mathcal{L}_{temp} \tag{10}$$

where content losses are adopted from *SANet* [2]. Content loss is defined as:

$$\mathcal{L}_c = \|\overline{E(I_{cs})^u} - \overline{F^u_c}\|_2 + \|\overline{E(I_{cs})^v} - \overline{F^v_c}\|_2. \tag{11}$$

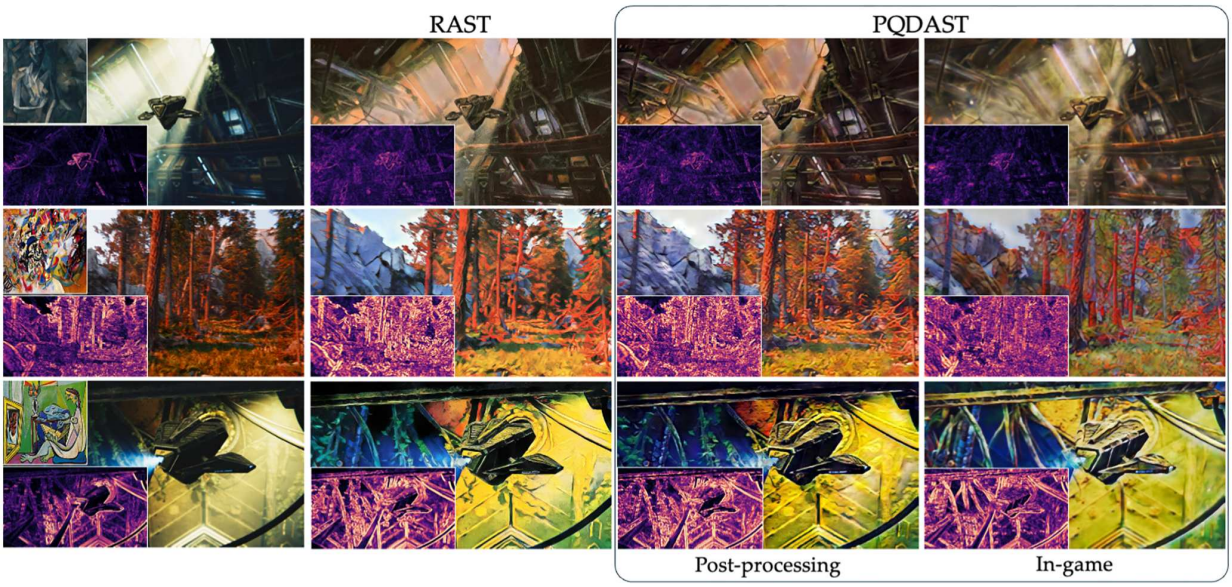
with $u = relu4\_1$ and $v = relu5\_1$ layers of a pre-trained *VGG-19* [60], $\overline{F^*_c}$ denotes mean-variance channel-wise normalised content features, and $\overline{E(I_{cs})^*}$ denotes the corresponding mean–variance channel-wise normalised features of the stylised image. Style loss is defined as:

$$\mathcal{L}_{style} = \sum_{i=1}^{L} \|\mu(\phi_i(I_{cs})) - \mu(\phi_i(I_s))\|_2 + \|\sigma(\phi_i(I_{cs})) - \sigma(\phi_i(I_s))\|_2 . \tag{12}$$

where $L = \{relu1\_1, relu2\_1, relu3\_1, relu4\_1, relu5\_1\}$, and where ϕ_i denotes a feature map of the i-th layer of the *VGG* encoder.

E. PQDAST in the Game's Pipeline

Inspired by previous work for in-game artistic stylisation [13], [14], we implement a *Custom Pass* in the Unity HDRP [64]. The trained network is injected before the Post-Process stage. The user can select any artwork to be used as the reference style image. This leads to generated results of improved temporal coherence for any selected style image, while the post-process effects (e.g., Depth-of-Field) are retained. It is important to note that our framework is trained with gamma-encoded images, whereas Unity HDRP uses a Linear colour space. Therefore, the reference style image and each colour buffer mipmap frame are converted to sRGB space before being processed.



Input	RAST	PQDAST (Post-processing)	PQDAST (in-game)
0.0742	0.1546	0.1354	0.1021
0.3517	0.4969	0.4769	0.4110
0.2445	0.4299	0.4248	0.4262

Fig. 4. Results comparing *PQDAST* to *RAST*. *RAST* is used as a post-processing effect. The input frame is shown on the left. The difference between the shown and previous frames is visualised using the ꟻLIP evaluator. In-game *PQDAST* generates temporally consistent results. The difference between the shown and previous frames is visualised using the ꟻLIP evaluator. The table below the images provides the numerical value of this difference (calculated using ꟻLIP), for each method and for each row of the figure.

IV. Experiments

A. Training Details

Considering the synthetic nature of computer games' imagery, we use the *MPI Sintel* [65] training set to train our network. As the trained *PQDAST* is injected before the post-process stage, and intercepts frames that are not post-processed, we train using frames from both the Clean pass and the Final pass. We chose a synthetic video dataset to benefit from data retrieved at different stages of a game's pipeline, and to match the synthetic nature of game graphics. Exploring realistic datasets, particularly from driving simulators or photorealistic games, would be an interesting direction for future work. *Wikiart* [66] is used as the style images dataset. Adam optimizer [67] is employed with a learning rate of 0.0001 and a batch size of 6 content–style image pairs. During training, both images are rescaled to 256×256 pixels. The hyperparameters λ_c, λ_s, λ_k, λ_d, and λ_t are set to 1.0, 3.0, 1.0, 1.0, and 10.0 respectively. These values were selected empirically based on validation results and were found to generalise reasonably across different scenes and styles. Training requires 160000 steps and lasts $\sim$30 hours on a single NVIDIA Tesla V100 GPU.

B. PQDAST for Computer Games

Our proposed framework trains compressed transformer and decoder models to generate results with comparable stylisation quality to *RAST* [4]. Similarly to [13], [14], the trained model is injected into the game's pipeline. Example results are shown in Figure 4. At the bottom of each image, temporal error maps are provided (ꟻLIP is used to compute the difference between the current and previous frame). Our system used as a post-processing effect synthesises similar results to *RAST*, with improved temporal consistency. When *PQDAST* is used in-game, stylised frames are temporally more stable (the temporal error map is the closest in similarity with the original frame's temporal error map), while the stylisation quality is slightly altered, as the post-process effects in the game are enabled. In addition to the visual fidelity improvements introduced in *PQDAST* and the temporal considerations we make during training, our system gains a boost in performance when injected into the game's pipeline. Intercepting each G-buffer colour frame and producing a stylised version that is then passed through the Post-process stage prevents undesired artefacts and flickering effects, as shown in [13], [14].

C. Comparisons with State-of-the-Art Methods

We compare the performance of *PQDAST* against 7 state-of-the-art methods. As temporal stability is crucial for the stylisation of computer games, we compare against approaches that consider temporal data (*AdaAttN* [3], *CSBNet* [7], *MCCNet* [6], *FVMST* [46]) or they are optimised for games (*NSTFCG* [13], *GBGST* [14]). We also compare against *RAST* [4], the method which our model distils knowledge from. For clarification, *post-processing* refers to applying the trained stylisation

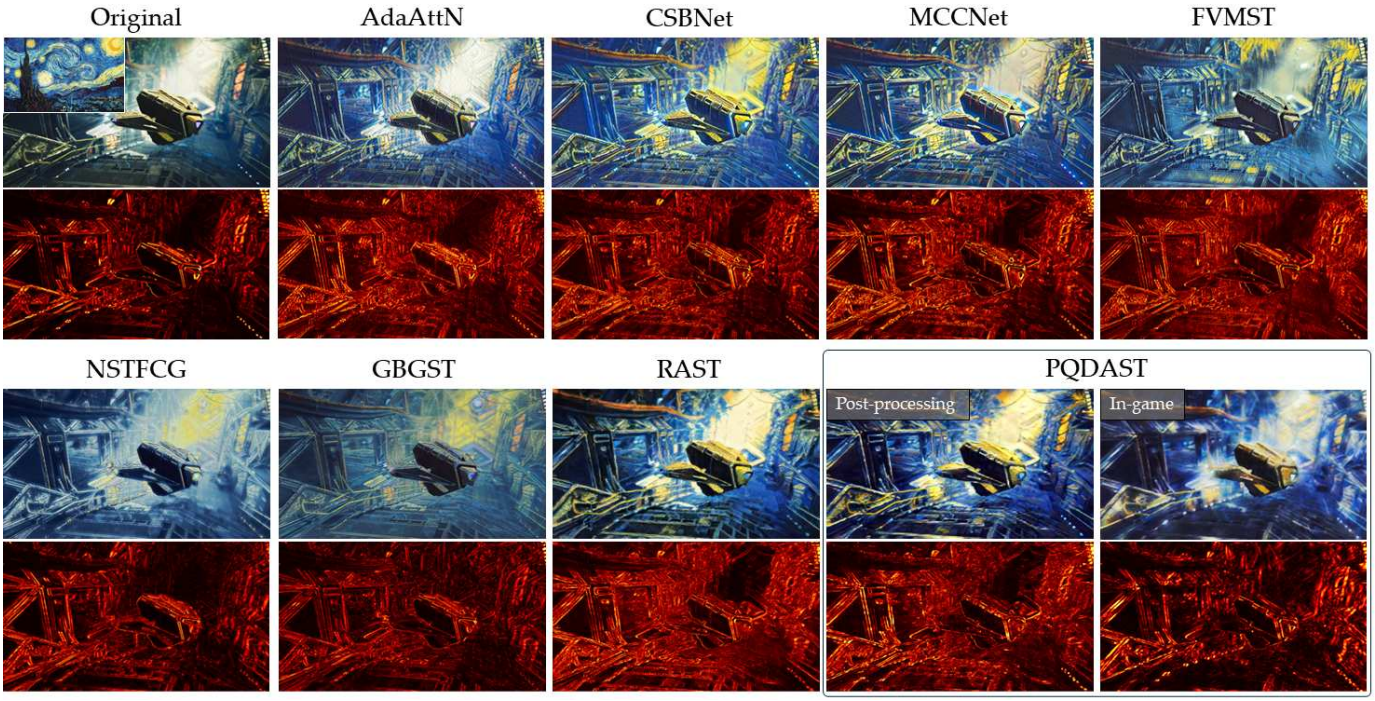


Fig. 5. Qualitative results comparing *PQDAST* to state-of-the-art methods. A heatmap of the temporal error between the current and previous frame is included in the bottom row. Our proposed approach produces high-quality stylisations. The temporal error heatmap of *PQDAST* in-game is closest to the original frame's heatmap along with *NSTFCG* and *GBGST* that are used in-game. Additional results are provided in Figure 6.

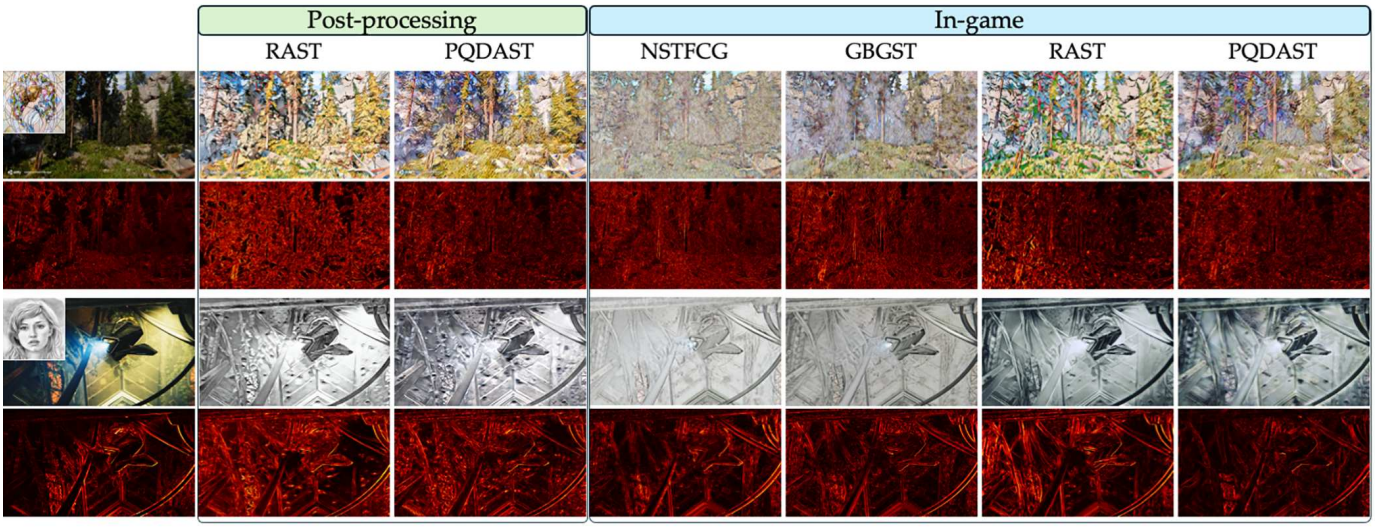


Fig. 6. Results for additional game scenes/style images. The bottom rows provide the temporal error heatmap between the current and previous frame.

network to the final rendered frames outside the real-time pipeline, whereas *in-game* stylisation denotes integration of the network as a Custom Pass within Unity's HDRP before the post-process stage. This configuration enables frame-by-frame stylisation during gameplay while retaining all post-processing effects.

1) Qualitative Results: Qualitative results are shown in Figure 5. The original input frame and style image are shown on the top left, whilst the bottom rows provide a temporal error heatmap showing the difference between the current and previous frame. It is important to note that specific representative frames and their corresponding heatmaps are provided in Figure 5 (and also in Figure 6). The error heatmaps would vary if other frames were chosen. A comprehensive quantitative evaluation for multiple frames and from different scenes is given in Section IV-C2. As shown in Figure 5, *CSBNet* and *MCCNet* demonstrate good stylisation quality but with noticeable artefacts, mainly around the central object's edges. *FVMST* does not adequately transfer the artistic style of the reference painting producing white blobs and inconsistent stylisations. *AdaAttN* manages to retain important content information, however, the stylisation effect is not sufficiently

achieved – the stylised frame does not contain yellow colours that are eminent in the reference style. *RAST* produces high-quality stylisations, justifying our selection for a 'teacher' model to train our distillation framework. Nevertheless, the temporal error heatmap shows temporal incoherence. This is also noticeable for the other image and video approaches used as post-processing effects (*AdaAttN*, *CSBNet*, *MCCNet*, *FVMST*). The in-game approaches *NSTFCG* and *GBGST* demonstrate improved temporal stability performance but sacrifice some stylisation quality. Our proposed system successfully compresses *RAST*, maintaining a high degree of similarity to *RAST* when used post-process while the corresponding temporal error heatmap is improved. When *PQDAST* is used in-game, similar to *NSTFCG* and *GBGST*, the temporal error heatmap closely matches that of the input frame. Additionally, as our model distils knowledge from *RAST*, the stylisation quality is improved.

Additional qualitative comparisons are provided in Figure 6. The temporal error heatmap of *PQDAST* (in-game) is similar to those of *NSTFCG* and *GBGST* (in-game), but the stylisation quality is noticeably better. *PQDAST*'s heatmap more closely resembles the original frame's heatmap than *RAST*'s temporal error heatmap does post-processing and in-game). Yet, there is a slight but noticeable difference in stylisations between our method and *RAST*. We hypothesise that this arises from variations in model size, training objectives, and the different types of data used for training. Our method is trained on synthetic frames, whereas *RAST* was trained exclusively on real-world images from the MS COCO [68] dataset. This highlights the impact of dataset characteristics on the resulting stylistic outcomes, demonstrating the unique advantages and challenges presented by both synthetic and real-world datasets. In this work, we choose to utilise the conventional, widely used MPI Sintel [65] dataset to develop a system with broad generalisability across various games. This decision, instead of using the training set suggested in [14], allows our model to adapt to arbitrary styles and different games. However, this approach comes at the cost of not being able to use G-buffer information. Thus, we do not train our model solely using frames from the games that we test on, similarly to [14]. This dataset mismatch between *RAST* and *PQDAST* also likely contributes to differences in sharpness and structural fidelity. In addition, some fine style pattern detail is diminished when in-game post-process effects are applied, further reducing structural fidelity. Finally, *PQDAST* is a distilled version of RAST, which introduces additional trade-offs between efficiency and style detail preservation.

2) Quantitative Results: Evaluation in the field of style transfer remains an open problem. A range of computational metrics exist to quantify the performance of stylisation approaches, yet there is no standardised evaluation procedure, and the computational metrics utilised are reliable only to a certain degree [69]. Here, we show quantitative evaluation using a few metrics deemed to be the most relevant in the context of style transfer for computer games (Table II). For consistency with previous in-game stylisation methods [13], [14], we use the same test dataset of 2100 frames from 4 different game scenes, and the same 10 style images. Note that our trained model has not seen any frames resembling the test dataset during training.

To gauge the effectiveness of our approach in producing temporally stable stylisations, we measure Warping Error using optic flow information. Similarly to [13], [14], we also calculate LPIPS Error [70], as the average perceptual distances between consecutive frames. The results are gathered in Table II. Our method outperforms state-of-the-art approaches in Warping Error and performs competently in LPIPS Error. This shows that *PQDAST* can generate artistically stylised results given any reference style image while sustaining temporal stability effectively. The slight drop in performance when applied in-game is expected, as post-process effects further modify the stylised frame, reducing similarity to the original input and influencing stylisation quality.

In Section III-B, we justify the use of ꟻLIP, which is utilised as an alternative to SSIM. As advised in [62], we avoid the use of SSIM for colour images. To measure how our approach performs in terms of perceptual and stylisation quality, we use LPIPS [70], SIFID [71], and ArtFID [72]. LPIPS gives a calculation of how well the perceptual information in the original frames is retained. SIFID is a measure of style fidelity, basically measuring FID for single images. ArtFID computes both the performance in capturing content information and reproducing the style image in a single metric. As depicted in Table II, while the performance of our proposed framework drops for the SIFID metric, our system performs competently in terms of retaining important content information (LPIPS), better than the previous in-game stylisation approaches (*NSTFCG* [13] and *GBGST* [14]). Our method also outperforms the in-game methods in overall style transfer quality (ArtFID).

To measure the quality of the proposed knowledge distillation scheme, results are also included for *RAST* – we adapted *RAST* and injected it into the game's pipeline in a similar way to *PQDAST* to provide an additional comparison for our work. Our method trained on a synthetic video dataset with temporal considerations outperforms *RAST* in terms of temporal consistency both when applying stylisations in the rendering pipeline and as a post-process effect. *RAST* performs very well in perceptual similarity score (second best), but its performance drops when embedding it in the game. *PQDAST*'s smaller size does not have a considerable impact when measuring LPIPS, and it outperforms *RAST* when injected into the graphics pipeline. Similarly, while *RAST* performs the best in SIFID and ArtFID metrics, our algorithm's effectiveness is competent, showing that compressing the transformer and the decoder does not result in substantial degradation of style transfer quality.

Table III provides efficiency analysis. A major advantage of *PQDAST* compared to previous in-game stylisation methods is that it is capable of reproducing arbitrary styles. Although our method is faster when inference time is calculated outside Unity, it achieves approximately 5 fps in Unity. This can be justified by the number of inference runs required – to compute a stylised image, 3 forward passes are needed: through the image encoder (VGG), through the transformer and through the decoder; *NSTFCG* and *GBGST* only require one forward pass. Our proposed framework, though, is significantly reduced in

TABLE II
QUANTITATIVE RESULTS. WARPING ERROR AND LPIPS ERROR ARE BOTH IN THE FORM $\times 10$. LPIPS MEASURES PERCEPTUAL SIMILARITY BETWEEN ORIGINAL RENDERED FRAMES AND STYLISED FRAMES. SIFID AND ARTFID QUANTIFY THE STYLE PERFORMANCE. WE PROVIDE RESULTS FOR OUR SYSTEM, *PQDAST*, INJECTED IN THE PIPELINE AND FOR THE TRAINED STYLISATION NETWORK APPLIED AS A POST-PROCESS EFFECT. WE DO THE SAME FOR *RAST*. THE BEST RESULTS ARE INDICATED IN **BOLD**, AND THE SECOND BEST ARE UNDERLINED.

	AdaAttN	CSBNet	MCCNet	FVMST	RAST	PQDAST	NSTFCG	GBGST	RAST	PQDAST
Warping Error ↓	1.6477	1.7458	1.6519	1.8524	1.7119	1.6080	1.5798	1.2984	1.4636	**1.2695**
LPIPS Error ↓	0.3217	0.3908	0.3547	0.3215	0.5285	0.4730	0.2930	**0.2515**	0.4131	0.3371
LPIPS ↓	**0.2692**	0.3378	0.3468	0.3806	0.3176	0.3294	0.3879	0.3494	0.3384	0.3327
SIFID ↓	1.6115	2.2468	1.5555	2.2529	**1.2913**	1.6185	1.8679	1.9401	3.4755	3.7163
ArtFID ↓	49.4115	52.4232	47.6695	53.8949	**46.8992**	51.3266	57.1858	54.1722	51.5226	52.8609
Processing Stage	Post-process						In-game			

TABLE III
EFFICIENCY. DUE TO THE COMPLEXITY OF THE OPERATIONS OF THE FIRST FOUR MODELS IN THE TABLE, THEY COULD NOT BE EXPORTED TO THE APPROPRIATE FORMAT [73] FOR USAGE INSIDE UNITY GAME ENGINE. INFERENCE TIMES (POST-PROCESS) INCLUDE INFERENCE THROUGH THE *VGG* NETWORK IF NECESSARY FOR EXTRACTING FEATURES USED IN STYLISATION. THESE ARE MEASURED ON A SINGLE NVIDIA GEFORCE RTX 3090 GPU, WITH IMAGE RESOLUTION 1920×1080.

Method	No. Styles	Memory (MB)	Speed (ms)	fps
AdaAttN	∞	50.2	86.04	-
CSBNet	∞	16.0	80.66	-
MCCNet	∞	18.3	34.69	-
FVMST	120	18.0	17.39	-
NSTFCG	1	3.03	50.21	10
GBGST	1	4.19	54.01	10
RAST	∞	30.4	31.73	2
PQDAST	∞	15.7	26.06	5

size compared to *RAST* and is therefore faster.

D. Ablation Study

1) Perceptual Quality-Guided Distillation Loss: Our proposed system synthesises results with comparable stylisation quality to *RAST* (Figures 4, 5), justifying the effectiveness of using ꟻLIP in addition to matching the intermediate and output-level feature maps. To further gauge the effectiveness of ꟻLIP, we also train *PQDAST* without $\mathcal{L}_{PQ}^{distil}$. Results are provided in Figure 7. The difference between the generated result and *RAST*'s generated result is also provided. Using ꟻLIP has a noticeable impact on the performance of our compressed model. Not only is the resulting image closer in similarity to *RAST*, but it also avoids incongruities and uneven brushstrokes in parts of the image.

2) Depth Loss: Employing a depth reconstruction loss has been established as a good practice for the style transfer task [19], [20], [49]. Compared to previous approaches, here, we have utilised an advanced depth prediction network that surpasses the performance of previously used methods. In Figure 8, we show that incorporating the proposed depth loss has a noticeable impact not only in preserving depth and fine details (bottom row) but also in temporal consistency (top row – temporal error maps are provided and the error computed is closer to that of the original frame). Our proposed framework, as injected in the game's pipeline, is also compared with other in-game approaches in Figure 9. Although not trained using *MiDaS*, our system's produced depth map is very similar to the original frame's depth map, demonstrating that *PQDAST* preserves depth details and allows the main object in the centre of the frame to stand out. The calculated MSE and PSNR values illustrate that *PQDAST* performs better than *NSTFCG* and competently with *GBGST* which also uses depth during inference. Employing an advanced depth prediction method, as discussed in [20], results in better depth preservation.

E. Limitations

Despite our efforts to minimise temporal inconsistencies across sequential frames, preventing flickering and achieving temporal stability remains a challenge in the realm of games due to the complicated and unpredictable environments. Complex lighting and shadows often introduce flickering in the game scenes, even without post-processing effects taking place. Our approach aims to show that efficient artistic stylisation in games is possible without a large compromise in speed and memory. To further improve upon alleviating temporal or flickering issues, G-buffer information can be used at the inference stage, similarly to [14]. In this work, we have not addressed that to avoid further inference delays.

An alternative direction to further reduce flickering would be to perform stylisation directly in 3D space, operating on meshes or materials instead of rendered frames. However, such methods typically demand full access to the game's geometry and rendering assets, which limits their practicality in commercial settings. Our work instead focuses on a general solution that can be seamlessly integrated into existing pipelines without requiring modifications to in-game assets.

As shown in Table III, although outperforming *RAST*, the frame rate performance of our system drops to $\sim$ 5 fps, whereas *NSTFCG* and *GBGST* achieve 10 fps. However, these are capable of only one style per trained network. In the realm of computer games, speed remains an important issue for style transfer. Temporalisation schemes and manually scheduling the in-game network inference [53] could also help in improving speed.

Another step towards better performance would be to compress the *VGG* encoder used to generate encoded features. Notably, model compression can significantly impact the performance of artistically stylised games, offering advantages in both speed and GPU resource requirements. Unavoidably, speed is interconnected to the memory size of the model –

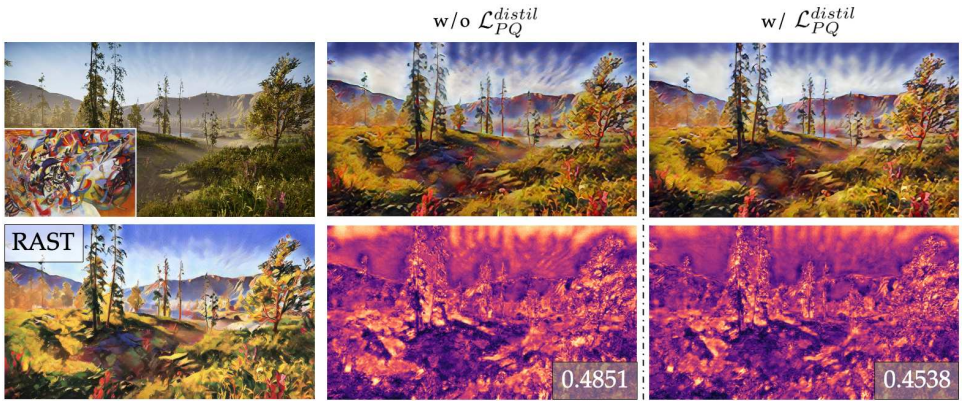


Fig. 7. Ablation study on the effect of ꟻLIP for knowledge distillation. Using $\mathcal{L}_{PQ}^{distil}$ produces results that retain the detail of the content image (right) as in *RAST*. Artefacts and inconsistencies are avoided (middle). The values from the ꟻLIP operation are shown at the bottom right of each difference image.

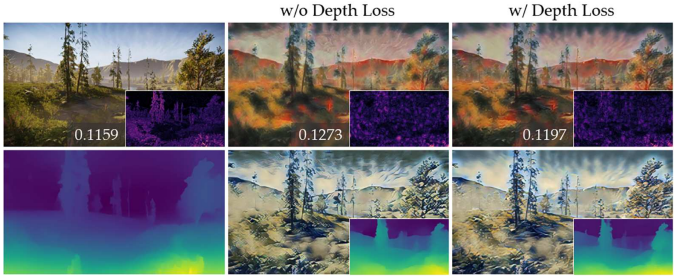


Fig. 8. Ablation study on the effect of depth loss. The depth maps (bottom row) are generated using *MiDaS* [63]. The values from the ꟻLIP operation are shown on the left of each difference image in the top row.

typically, a larger model would require more time to execute a forward pass. Additionally, although our stylisation model occupies a small amount of memory, it is important to note that memory constraints for games are significantly challenging. Modern games require more and more GPU memory, especially when targeting higher frame resolutions [74]. Our experiments follow prior work in using a fixed resolution for comparability. Investigating resolution scaling is an interesting avenue for future experiments. Nevertheless, our framework, the first to address arbitrary stylisation in games, avoids the dependence on multiple single-style-per-model NST models to reproduce multiple artistic styles, while our distillation algorithm promises a new way for compressing image generation models for use in games.It is worth noting that while our ablation study focuses on the perceptual quality-guided distillation term, further analysis of the individual loss components and architectural variations would provide deeper insight into their respective contributions. We consider this an important direction for future work. Finally, although we did not conduct a user study, we acknowledge that user evaluations would provide valuable complementary insights into stylisation quality and temporal perception.

V. Conclusion

We have presented an arbitrary style transfer solution for computer games that makes use of a perceptual quality-guided knowledge distillation scheme inspired by image quality assessment of 3D renderings. Our trained model, *PQDAST*, is smaller and faster than the compared transformer-based arbitrary style transfer approaches, and it is integrated into the game's pipeline. Extensive qualitative and quantitative experiments have demonstrated that our system surpasses state-of-the-art methods in temporal coherence while achieving comparable perceptual and stylisation performance. Our work thus demonstrates an effective new way to perform knowledge distillation for image generation tasks, also showing that arbitrary style transfer for games can be achieved using a conventional GPU. Future work will focus on further improving the speed of the in-game stylisation models for real-time arbitrary style transfer in games.

References

[1] X. Huang and S. Belongie, "Arbitrary style transfer in real-time with adaptive instance normalization," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 1501–1510.

[2] D. Y. Park and K. H. Lee, "Arbitrary style transfer with style-attentional networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 5880–5888.

[3] S. Liu, T. Lin, D. He, F. Li, M. Wang, X. Li, Z. Sun, Q. Li, and E. Ding, "Adaattn: Revisit attention mechanism in arbitrary neural style transfer," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 6649–6658.

[4] Y. Ma, C. Zhao, X. Li, and A. Basu, "RAST: Restorable arbitrary style transfer via multi-restoration," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2023, pp. 331–340.

[5] X. Li, S. Liu, J. Kautz, and M.-H. Yang, "Learning linear transformations for fast image and video style transfer," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 3809–3817.

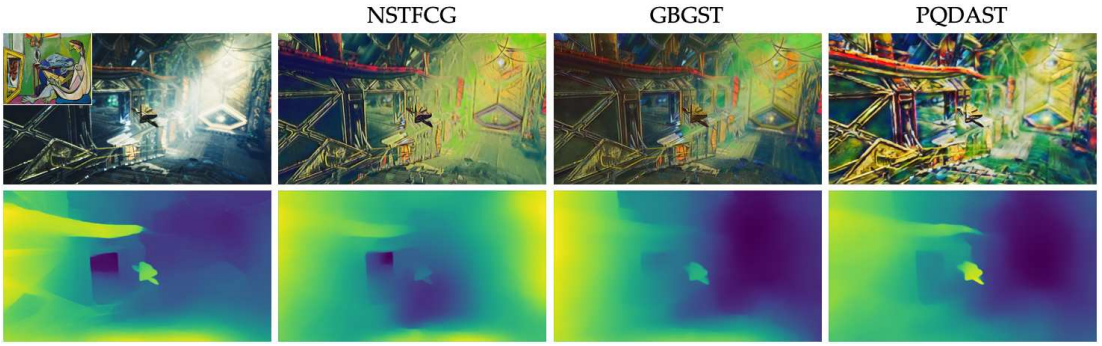
[6] Y. Deng, F. Tang, W. Dong, H. Huang, C. Ma, and C. Xu, "Arbitrary video style transfer via multi-channel correlation," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 2, pp. 1210–1217, 5 2021.

[7] H. Lu and Z. Wang, "Universal video style transfer via crystallization, separation, and blending," in *Proc. Int. Joint Conf. on Artif. Intell.(IJCAI)*, vol. 36, 2022, pp. 4957–4965.

[8] K. Zhang, N. Kolkin, S. Bi, F. Luan, Z. Xu, E. Shechtman, and N. Snavely, "Arf: Artistic radiance fields," in *European Conference on Computer Vision*. Springer, 2022, pp. 717–733.

[9] K. Liu, F. Zhan, Y. Chen, J. Zhang, Y. Yu, A. El Saddik, S. Lu, and E. P. Xing, "Stylerf: Zero-shot 3d style transfer of neural radiance fields," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 8338–8348.

[10] H.-W. Pang, B.-S. Hua, and S.-K. Yeung, "Locally stylized neural radiance fields," in *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE Computer Society, 2023, pp. 307–316.



NSTFCG	GBGST	PQDAST
2092.80 / 14.9235	840.8185 / 18.8838	882.6262 / 18.6730

Fig. 9. Depth preservation performance comparison between our approach *PQDAST (in-game)*, and the in-game methods *NSTFCG* [13], *GBGST* [14]. *NSTFCG* and *GBGST* use *MiDaS* [63] to define depth reconstruction loss. The depth maps in the bottom row are generated using *MiDaS* for fairer comparisons. The table shows the error differences between the original frame's depth map and the depth map generated from the stylisation of each method. Mean square error (MSE) and peak-signal-to-noise ratio (PSNR) are provided. Both MSE and PSNR are used as a numerical measure of similarity between depth maps.

[11] X. Li, Z. Cao, Y. Wu, K. Wang, K. Xian, Z. Wang, and G. Lin, "S-dyrf: Reference-based stylized radiance fields for dynamic scenes," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 20 102–20 112.

[12] M. Fischer, Z. Li, T. Nguyen-Phuoc, A. Bozic, Z. Dong, C. Marshall, and T. Ritschel, "Nerf analogies: Example-based visual attribute transfer for nerfs," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 4640–4650.

[13] E. Ioannou and S. Maddock, "Neural style transfer for computer games," in *British Machine Vision Conference, CVG Workshop*, 2023.

[14] ——, "Towards real-time g-buffer-guided style transfer in computer games," *IEEE Transactions on Games*, pp. 1–9, 2024.

[15] S. R. Richter, H. A. AlHaija, and V. Koltun, "Enhancing photorealism enhancement," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 2, pp. 1700–1715, 2022.

[16] M. Mittermueller, Z. Ye, and H. Hlavacs, "EST-GAN: Enhancing style transfer gans with intermediate game render passes," in *2022 IEEE Conference on Games (CoG)*, 2022, pp. 25–32.

[17] P. Andersson, J. Nilsson, T. Akenine-Möller, M. Oskarsson, K. Åström, and M. D. Fairchild, "ꟻLIP: A Difference Evaluator for Alternating Images," *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, vol. 3, no. 2, pp. 15:1–15:23, 2020.

[18] L. Yang, B. Kang, Z. Huang, X. Xu, J. Feng, and H. Zhao, "Depth anything: Unleashing the power of large-scale unlabeled data," *arXiv preprint arXiv:2401.10891*, 2024.

[19] X.-C. Liu, M.-M. Cheng, Y.-K. Lai, and P. L. Rosin, "Depth-aware neural style transfer," in *Proceedings of the Symposium on Non-Photorealistic Animation and Rendering*, 2017, pp. 1–10.

[20] E. Ioannou and S. Maddock, "Depth-aware neural style transfer using instance normalization," in *Computer Graphics & Visual Computing (CGVC) 2022*. Eurographics Digital Library, 2022.

[21] L. A. Gatys, A. S. Ecker, and M. Bethge, "Image style transfer using convolutional neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2414–2423.

[22] J. Johnson, A. Alahi, and L. Fei-Fei, "Perceptual losses for real-time style transfer and super-resolution," in *European Conference on Computer Vision*. Springer, 2016, pp. 694–711.

[23] C. Li and M. Wand, "Precomputed real-time texture synthesis with markovian generative adversarial networks," in *ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part III 14*. Springer, 2016, pp. 702–716.

[24] D. Ulyanov, V. Lebedev, A. Vedaldi, and V. S. Lempitsky, "Texture networks: Feed-forward synthesis of textures and stylized images." in *ICML*, vol. 1, 2016, p. 4.

[25] D. Ulyanov, A. Vedaldi, and V. Lempitsky, "Improved texture networks: Maximizing quality and diversity in feed-forward stylization and texture synthesis," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 6924–6932.

[26] G. Ghiasi, H. Lee, M. Kudlur, V. Dumoulin, and J. Shlens, "Exploring the structure of a real-time, arbitrary neural artistic stylization network," *arXiv preprint arXiv:1705.06830*, 2017.

[27] S. Gu, C. Chen, J. Liao, and L. Yuan, "Arbitrary style transfer with deep feature reshuffle," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 8222–8231.

[28] J. Huo, S. Jin, W. Li, J. Wu, Y.-K. Lai, Y. Shi, and Y. Gao, "Manifold alignment for semantically aligned style transfer," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 14 861–14 869.

[29] F. Shen, S. Yan, and G. Zeng, "Neural style transfer via meta networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 8061–8069.

[30] J. Svoboda, A. Anoosheh, C. Osendorfer, and J. Masci, "Two-stage peer-regularized feature recombination for arbitrary image style transfer," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 13 816–13 825.

[31] W. Xu, C. Long, and Y. Nie, "Learning dynamic style kernels for artistic style transfer," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 10 083–10 092.

[32] T. Q. Chen and M. Schmidt, "Fast patch-based style transfer of arbitrary style," *arXiv preprint arXiv:1612.04337*, 2016.

[33] L. Sheng, Z. Lin, J. Shao, and X. Wang, "Avatar-net: Multi-scale zero-shot style transfer by feature decoration," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 8242–8250.

[34] J. An, S. Huang, Y. Song, D. Dou, W. Liu, and J. Luo, "Artflow: Unbiased image style transfer via reversible neural flows," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 862–871.

[35] S. Huang, J. An, D. Wei, J. Luo, and H. Pfister, "Quantart: Quantizing image style transfer towards high visual fidelity," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 5947–5956.

[36] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[37] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[38] Y. Deng, F. Tang, W. Dong, C. Ma, X. Pan, L. Wang, and C. Xu, "Stytr2: Image style transfer with transformers," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 11 326–11 336.

[39] X. Luo, Z. Han, L. Yang, and L. Zhang, "Consistent style transfer," *arXiv preprint arXiv:2201.02233*, 2022.

[40] K. Hong, S. Jeon, J. Lee, N. Ahn, K. Kim, P. Lee, D. Kim, Y. Uh, and H. Byun, "AesPA-Net: Aesthetic pattern-aware style transfer networks," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 22 758–22 767.

[41] M. Zhu, X. He, N. Wang, X. Wang, and X. Gao, "All-to-key attention for arbitrary style transfer," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 23 109–23 119.

[42] J. Chung, S. Hyun, and J.-P. Heo, "Style injection in diffusion: A training-free approach for adapting large-scale diffusion models for style transfer," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 8795–8805.

[43] A. Hertz, A. Voynov, S. Fruchter, and D. Cohen-Or, "Style aligned image generation via shared attention," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 4775–4785.

[44] M. Ruder, A. Dosovitskiy, and T. Brox, "Artistic style transfer for videos," in *Pattern Recognition*, B. Rosenhahn and B. Andres, Eds. Cham: Springer International Publishing, 2016, pp. 26–36.

[45] C. Gao, D. Gu, F. Zhang, and Y. Yu, "ReCoNet: Real-time coherent video style transfer network," in *ACCV 2018: 14th Asian Conference on Computer Vision*. Springer, 2019, pp. 637–653.

[46] W. Gao, Y. Li, Y. Yin, and M.-H. Yang, "Fast video multi-style transfer," in *Proceedings of the IEEE/CVF winter Conference on Applications of Computer Vision*, 2020, pp. 3222–3230.

[47] M.-M. Cheng, X.-C. Liu, J. Wang, S.-P. Lu, Y.-K. Lai, and P. L. Rosin, "Structure-preserving neural style transfer," *IEEE Transactions on Image Processing*, vol. 29, pp. 909–920, 2019.

[48] S. Liu and T. Zhu, "Structure-guided arbitrary style transfer for artistic image and video," *IEEE Transactions on Multimedia*, 2021.

[49] E. Ioannou and S. Maddock, "Depth-aware neural style transfer for videos," *Computers*, vol. 12, no. 4, p. 69, 2023.

[50] B. Gu, H. Fan, and L. Zhang, "Two birds, one stone: A unified framework for joint learning of image and video style transfers," *arXiv preprint arXiv:2304.11335*, 2023.

[51] Y. Zhang, F. Tang, W. Dong, H. Huang, C. Ma, T.-Y. Lee, and C. Xu, "A unified arbitrary style transfer framework via adaptive contrastive learning," *ACM Transactions on Graphics*, 2023.

[52] M. Ku, C. Wei, W. Ren, H. Yang, and W. Chen, "Anyv2v: A tuning-free framework for any video-to-video editing tasks," *Transactions on Machine Learning Research*, 2024.

[53] T. Deliot, F. Guinier, and K. Vanhoey, "Real-time style transfer in unity using deep neural networks," 2020. [Online]. Available: https://blog.unity.com/engine-platform/real-time-style-transfer-in-unity-using-deep-neural-networks

[54] H. Park and N. Baek, "A design of real-time style-transfer operations in a game engine," in *International Symposium on Visual Computing*. Springer, 2024, pp. 424–435.

[55] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.

[56] H. Wang, Y. Li, Y. Wang, H. Hu, and M.-H. Yang, "Collaborative distillation for ultra-resolution universal style transfer," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 1860–1869.

[57] X. Chen, Y. Zhang, Y. Wang, H. Shu, C. Xu, and C. Xu, "Optical flow distillation: Towards efficient and stable video style transfer," in *ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part VI 16*. Springer, 2020, pp. 614–630.

[58] W. Chen, Y. Huang, M. Wang, X. Wu, and X. Zeng, "Kbstyle: Fast style transfer using a 200 kb network with symmetric knowledge distillation," *IEEE Transactions on Image Processing*, vol. 33, pp. 82–94, 2023.

[59] H. Chen, F. Shao, X. Chai, Q. Jiang, X. Meng, and Y.-S. Ho, "Collaborative learning and style-adaptive pooling network for perceptual evaluation of arbitrary style transfer," *IEEE Transactions on Neural Networks and Learning Systems*, 2023.

[60] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[61] Y. Ma, C. Zhao, B. Huang, X. Li, and A. Basu, "Rast: Restorable arbitrary style transfer," *ACM Transactions on Multimedia Computing, Communications and Applications*, vol. 20, no. 5, pp. 1–21, 2024.

[62] J. Nilsson and T. Akenine-Möller, "Understanding ssim," *arXiv preprint arXiv:2006.13846*, 2020.

[63] R. Ranftl, K. Lasinger, D. Hafner, K. Schindler, and V. Koltun, "Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2020.

[64] T. Unity, "High definition Render Pipeline: 12.1.12," 2021. [Online]. Available: https://docs.unity.cn/Packages/com.unity.render-pipelines.high-definition@12.1/manual/index.html

[65] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black, "A naturalistic open source movie for optical flow evaluation," in *European Conf. on Computer Vision*, ser. Part IV, LNCS 7577, A. Fitzgibbon et al. (Eds.), Ed. Springer-Verlag, Oct. 2012, pp. 611–625.

[66] F. Phillips and B. Mackintosh, "Wiki art gallery, inc.: A case for critical thinking," *Issues in Accounting Education*, vol. 26, no. 3, pp. 593–608, 2011.

[67] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[68] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: Common objects in context," in *European conference on Computer Vision*. Springer, 2014, pp. 740–755.

[69] E. Ioannou and S. Maddock, "Evaluation in neural style transfer: A review," *Computer Graphics Forum*, vol. 43, no. 6, p. e15165, 2024.

[70] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, "The unreasonable effectiveness of deep features as a perceptual metric," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 586–595.

[71] T. R. Shaham, T. Dekel, and T. Michaeli, "SinGAN: Learning a generative model from a single natural image," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 4570–4580.

[72] M. Wright and B. Ommer, "Artfid: Quantitative evaluation of neural style transfer," in *DAGM German Conference on Pattern Recognition*. Springer, 2022, pp. 560–576.

[73] ONNX, "Open neural network exchange," 2019, https://onnx.ai/. [Online]. Available: https://onnx.ai/

[74] M. Connatser and J. Martindale, "How much gpu memory do i need?" 7 2023. [Online]. Available: https://www.digitaltrends.com/computing/how-much-gpu-memory-do-i-need/