Volume 0 (1981), Number 0 pp. 1–9 COMPUTER GRAPHICS *forum*

DSAIT 4120 Final Project Report: Style Transfer

P. Johari, O. Milchi, M. Smink
TU Delft

Implementation found at:
https://gitlab.ewi.tudelft.nl/cgv/dsait4120/student-repositories/2024-2025/dsait4120-24-group16.git

1. Personal Components

1.1. Style Mixing

The implementation of Style Mixing follows Assignment 3, inspired by [GEB16], in which a neural model aims to optimize four losses pertaining to the content image, style images, total value and edges. The following modifications were made to Assignment 3:

- Implemented (Squeeze-and-Excitation) Channel Attention and applied it to the content features when computing the content loss. The channel attention pipeline is shown in Figure 1. Figure 5 highlights the difference made by this addition.
- Merged the two input style images when computing the style loss. Before computing the style Gram matrix, the features of the two images are merged following Equation 1, where α is a configurable weight in the UI. Results are shown in Figure 4

$$final\_style\_feat = (1-\alpha) * style\_1 + \alpha * style\_2 \quad (1)$$

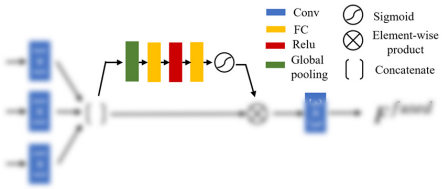

Figure 1: *Channel Attention pipeline applied to the content features. Image taken from [HZL19].*

1.2. Guidance Via Semantic Segmentation

Please see Section 1.3.2.

1.3. Text-Prompt-Based Control

1.3.1. Text Style Transfer

We apply a user-given text style to an input image by using a CLIP text-prompt encoder, a pre-trained FFN to transform text embeddings into style embeddings, and a pre-trained U-shaped CNN to apply style embeddings to the image. See Figure 6.

1.3.2. Text-Based Localization

With this method, we are able to localize style changes created using a style transfer method according to user input. We create a segmentation mask for an input image from a user-given text prompt using a pre-trained GroundingDINO open-set object detector and pre-trained SegmentAnything visual segmentation model. Using this mask, we merge the original and style images using optional edge smoothing. See Figures 7 and 8.

1.3.3. Text-Based Style Texture

We augment a style transfer by adding style-specific texture according to user text input. Steps to create the style texture mask include:

1. From the text prompt, generate an emoji using a T5-Base generative language model, pre-trained on a Text2Emoji dataset.
2. Transform the emoji into a silhouette mask using the Noto-Color-Emoji Font.
3. Apply the silhouette to an empty mask or an existing segmentation mask as a sliding kernel according to several resolutions and a user-defined relative step size.

This texture mask can then be used to merge the original image and any style-transferred image to add stylized texture to the transfer. This is particularly effective when using a segmentation mask with styles that are often translucent, such as fire. See Figure 7.

1.4. Color Palette

This feature takes as input a content image and a style image, then transfers the colour palette of the style image to the content image. The resulting image will have the same composition as the initial input, with its colours modified to match the style image.

The implementation is based on [RAGS01]. Both images are transformed from the RGB colour space to the CIELAB ($l\alpha\beta$) colour space. Then, the content image is normalized and scaled channel-wise. Finally, the channel-wise mean of the style image is transferred to the content image, and the result is transformed back into the RGBcolourr space. Example results are shown in Figure 9.

1.5. Guidance via Depth Map

This component enhances style transfer by incorporating depth awareness through two complementary approaches:

- **Depth-based Style Transfer**: Modifies the style transfer loss function to include a depth preservation term (Equation 2), ensuring stylization respects the content's spatial structure. Depth maps are generated using Depth-Anything-V2-Small-hf [YKH*24].

$$\mathcal{L}_{depth} = \|D(\vec{x}) - D(I_{content})\|_2 \tag{2}$$

- **Multi-Plane Stylization**: Implements depth-aware style mixing by:

  1. Splitting content into n depth layers using adaptive binning (Figure 2)
  2. Applying style transfer with exponentially decreasing strength:

$$h = 1 - k/n$$

$$w_{style}^{(k)} = w_{style} \cdot (e^{h-1/h}) \tag{3}$$

  Where k is the index value of k^{th} image.
  3. Reconstructing the final image by running style transfer on each and combining.

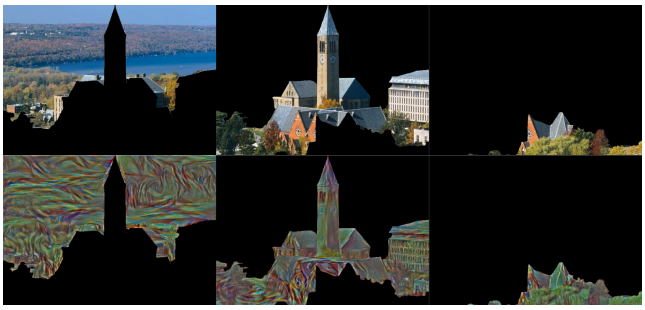

Figure 2: *Depth layer decomposition (n=3) and corresponding stylized layers.*

These approaches try different ways to preserve some scene geometry while enabling artistic control over depth-based stylization intensity with relatively mild results as seen in Figure 11.

1.6. Pixel Art

The pixel art converter transforms images through:

- **Pixelation**: Nearest-neighbor scaling
- **Color Quantization**:
  - 69 colour palettes or image-derived colours (K-means clustering)
  - KD-Tree acceleration for real-time lookup
- **Edge Enhancement**: Adjustable Canny detection

Features include palette interpolation (Fig. 3). Fig. 10 shows the transformation stages.

Figure 3: *Color palette options (Top: palette 34 and gradient, Bottom: Image Derived and Gradient, right: Image Derived From)*

1.7. Video

We perform video style transfer by running individual frames through the image-based style transfer pipeline, with optional added interpolation frames to smooth jittering. Users can also adjust video speed. See Figures 12 and 13.

2. Integration of Components

Our components are fully integrated into a pipeline (Figure 14). Text-based masking and texturing can be used within any component (Figure 15).

References

[Fro21] FROEHLING P.: Mixed neural style transfer with two style images, Dec 2021. URL: https://towardsdatascience.com/mixed-neural-style-transfer-with-two-style-images-9469b26 8

[GEB16] GATYS L. A., ECKER A. S., BETHGE M.: Image style transfer using convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2016). 1, 8

[GLK*17] GHIASI G., LEE H., KUDLUR M., DUMOULIN V., SHLENS J.: Exploring the structure of a real-time, arbitrary neural artistic stylization network, 2017. URL: https://arxiv.org/abs/1705.06830, arXiv:1705.06830. 9

[HZL19] HUANG Z., ZHANG J., LIAO J.: Style mixer: Semantic-aware multi-style transfer network. *Computer Graphics Forum 38*, 7 (Oct. 2019), 469–480. URL: http://dx.doi.org/10.1111/cgf.13853, doi:10.1111/cgf.13853. 1, 8

[IM22] IOANNOU E., MADDOCK S.: Depth-aware neural style transfer using instance normalization. URL: https://diglib.eg.org/handle/10.2312/cgvc20221165, doi:10.2312/CGVC.20221165. 8

[KKM20] KITOV V., KOZLOVTSEV K., MISHUSTINA M.: Depth-aware arbitrary style transfer using instance normalization, 2020. URL: https://arxiv.org/abs/1906.01123, arXiv:1906.01123. 8

[KMR*23] KIRILLOV A., MINTUN E., RAVI N., MAO H., ROLLAND C., GUSTAFSON L., XIAO T., WHITEHEAD S., BERG A. C., LO W.-Y., DOLLÁR P., GIRSHICK R.: Segment anything. *arXiv:2304.02643* (2023). 9

[KY22] KWON G., YE J. C.: Clipstyler: Image style transfer with a single text condition, 2022. URL: https://arxiv.org/abs/2112.00374, arXiv:2112.00374. 9

[LZR*23] LIU S., ZENG Z., REN T., LI F., ZHANG H., YANG J., LI C., YANG J., SU H., ZHU J., ET AL.: Grounding dino: Marrying dino with grounded pre-training for open-set object detection. *arXiv preprint arXiv:2303.05499* (2023). 9

[PWL*23] PENG L., WANG Z., LIU H., WANG Z., SHANG J.: Emojilm: Modeling the new emoji language. *CoRR abs/2311.01751* (2023). URL: https://doi.org/10.48550/arXiv.2311.01751, arXiv:2311.01751, doi:10.48550/ARXIV.2311.01751. 9

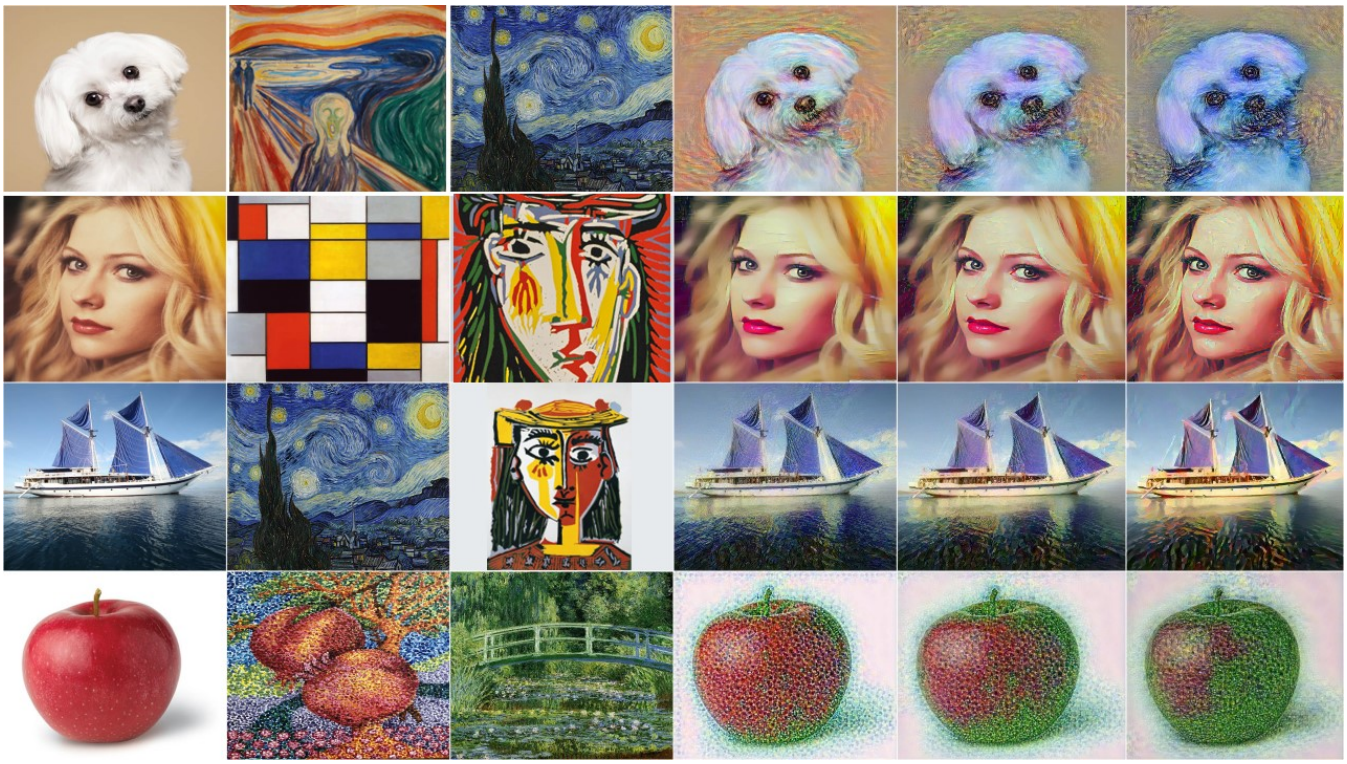

Figure 4: *Style Mixing results for four images. From left to right: content image, two style images, and three results. Proportions of the style images in the final styles are, in order: 80%/20%, 50%/50%, 20%/80%. Note the difference in brushstrokes where the Van Gogh style was used. Limited results were observed in the second example.*

Figure 5: *Results of applying channel attention for style mixing. From left to right: content image, style images, result without channel attention and result with channel attention. Differences (highlighted in red): a more evenly distributed style and a sharper subject*

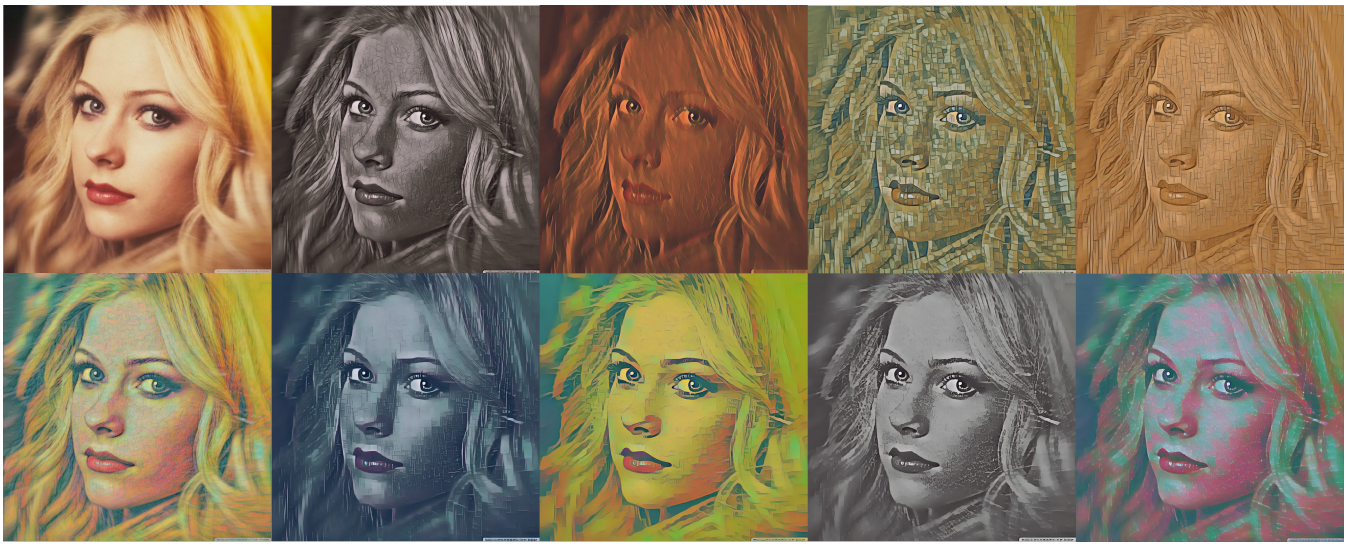

Figure 6: *Text Style Transfer results. In order prompts were: 1) [original image], 2) charcoal drawing, 3) fire, 4) style of cubism, 5) wooden cartoon, 6) sketch with crayons, 7) futuristic cyberpunk, 8) pop art style with bold, contrasting colours, 9) vintage black-and-white film, 10) rgb color crystal.*

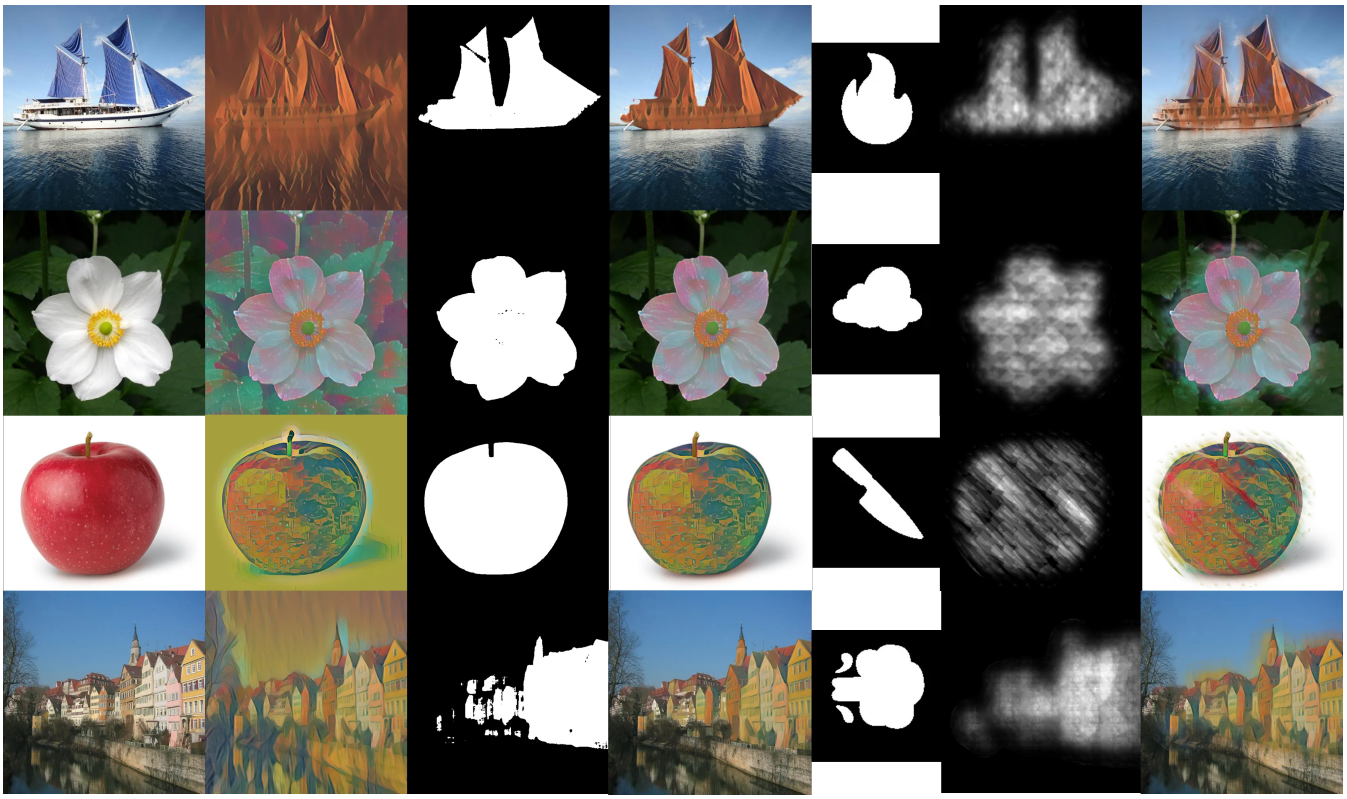

Figure 7: *Text style transfer results with localization and style texture. Prompts and parameters in the format 'Prompts(style, localization, style texture), Style Texture Parameters(flexibility, step size, strength)' per row: 1. ('fire', 'boat', 'fire'), (95, 0.5, 1.5); 2. ('rgb color crystal', 'flower', 'cloud'), (130, 0.5, 5.5); 3. ('pop art style with bold, contrasting colors', 'apple', 'knife'), (110, 1.1, 3.5); 4. ('scream painting', 'house', 'wind'), (130, 0.6, 4).*

Figure 8: *Localized text style transfer with and without edge smoothing. Amount of smoothing is, in order, 0, 5, and 15.*

Figure 9: *Examples of Color Palette Transfer results. From left to right: the content image, the style image, and the final output.*

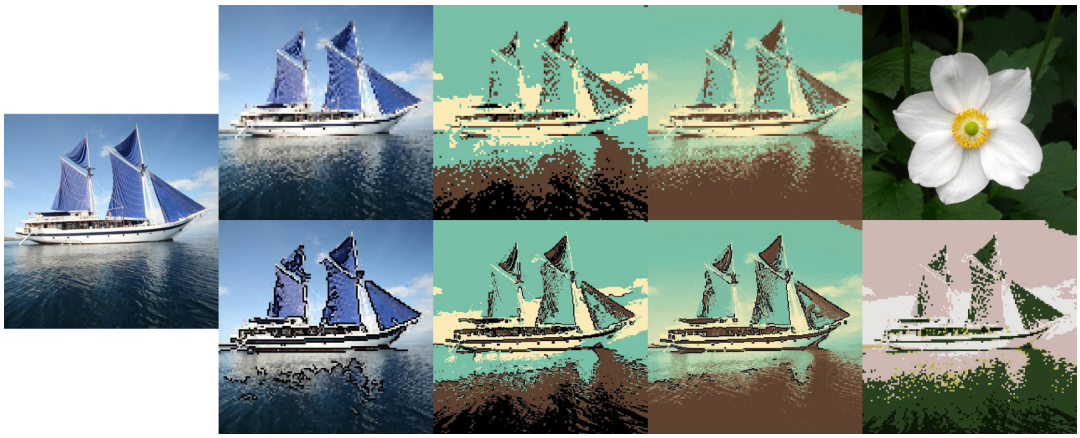

Figure 10: *Pixel art conversion stages(Below with Edge Enhancement): (a) Original, (b) Pixelated, (c) Color quantized, (d) Gradient-based quantized, (d) Image as Colour Pallet*

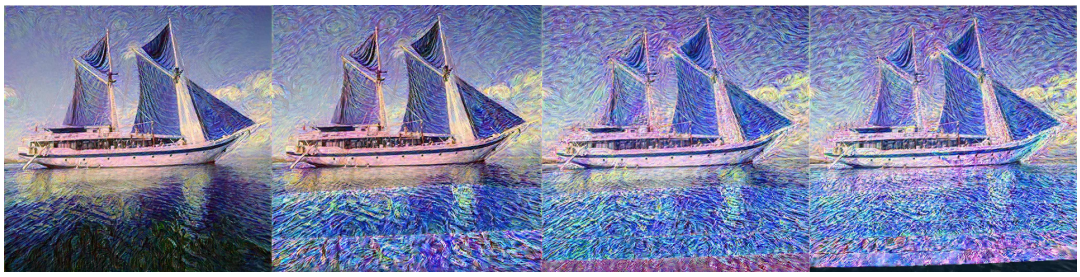

Figure 11: *Depth Based Style transfer; (a) Depth-based Style Transfer, (B) Multi-Plane Stylization for* $n = 3, 6, 9$

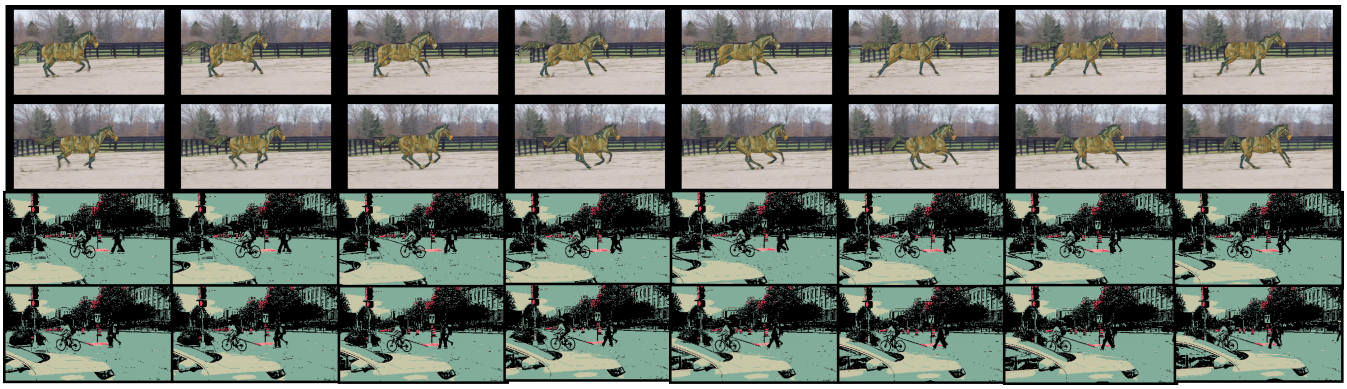

Figure 12: *Results of our different personal components on videos. 16 frames are shown without interpolation.*

Figure 13: *Demonstration of the video interpolation algorithm for 1 and 3 interpolations between adjacent frames.*

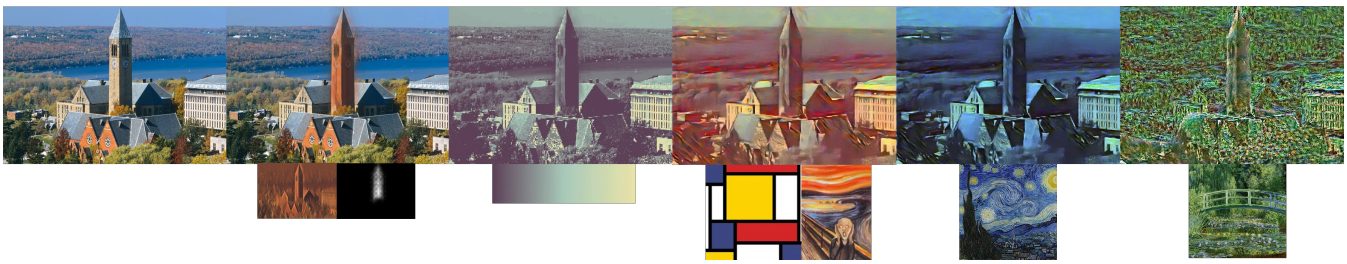

Figure 14: *Example of an image running through the whole pipeline, reporting intermediate steps. From left to right: 1) original image, 2) text-based 'fire' transfer on segmentation mask 'tower' with emoji texture 'fire', 3) pixel art on given continuous palette, 4) style mixing with given input styles, 5) colour palette transfer with given input image as palette, 6) depth-guided style transfer with given style image.*

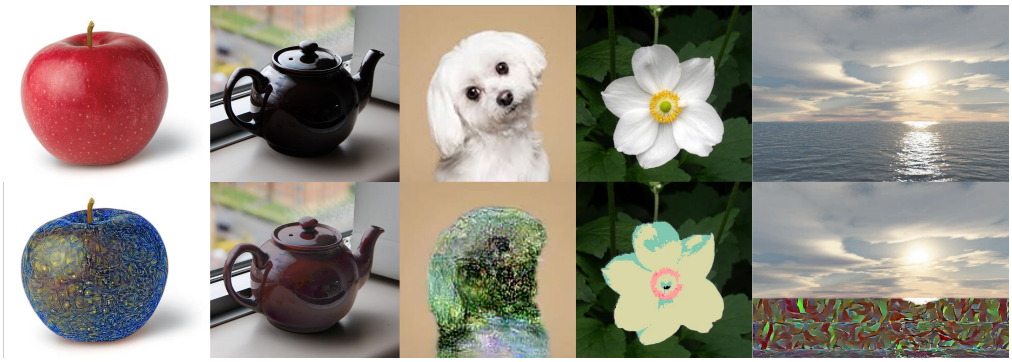

Figure 15: *Examples of text-based segmentation masking being used in (left to right) 1. classic style transfer, 2. colour palette transfer, 3. style mixing, 4. pixel art, and 5. depth-based style transfer.*

[RAGS01] REINHARD E., ADHIKHMIN M., GOOCH B., SHIRLEY P.: Color transfer between images. *IEEE Computer Graphics and Applications 21*, 5 (2001), 34–41. doi:10.1109/38.946629. 1, 8

[RKH*21] RADFORD A., KIM J. W., HALLACY C., RAMESH A., GOH G., AGARWAL S., SASTRY G., ASKELL A., MISHKIN P., CLARK J., KRUEGER G., SUTSKEVER I.: Learning transferable visual models from natural language supervision, 2021. URL: https://arxiv.org/abs/2103.00020, arXiv:2103.00020. 9

[RSR*23] RAFFEL C., SHAZEER N., ROBERTS A., LEE K., NARANG S., MATENA M., ZHOU Y., LI W., LIU P. J.: Exploring the limits of transfer learning with a unified text-to-text transformer, 2023. URL: https://arxiv.org/abs/1910.10683, arXiv:1910.10683. 9

[SJN*24] SURESH A. P., JAIN S., NOINONGYAO P., GANGULY A., WATCHAREERUETAI U., SAMACOITS A.: Fastclipstyler: Optimisation-free text-based image style transfer using style representations. In *2024 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)* (Jan. 2024), IEEE, p. 7301–7310. URL: http://dx.doi.org/10.1109/WACV57701.2024.00715, doi:10.1109/wacv57701.2024.00715. 9

[Smi16] SMITH C.: neural-style-tf. https://github.com/cysmith/neural-style-tf, 2016. 9

[WZBW23] WANG Q., ZHANG B., BIRSAK M., WONKA P.: Instructedit: Improving automatic masks for diffusion-based image editing with user instructions, 2023. arXiv:2305.18047. 9

[YKH*24] YANG L., KANG B., HUANG Z., ZHAO Z., XU X., FENG J., ZHAO H.: Depth anything v2, 2024. URL: https://arxiv.org/abs/2406.09414, arXiv:2406.09414. 2, 8

Table 1: The work distribution table for our project.

Student	Implemented feature and its explanation	What I implemented myself	Where and how I used external resources
P. Johari	**Pixel Art**: Transform images into pixel art through adjustable pixelation (downscale via nearest-neighbour), colour quantization, and edge detection. Colours can be mapped to pre-built palettes or extracted from the image using KNN clustering. A KDTree accelerates palette matching for performance. Canny edge detection adds dark outlines between colour regions using a customizable threshold, enhancing definition.	Pixelation (nearest-neighbour scaling), KDTree-based palette mapping, KNN colour smoothing, edge overlay logic, JSON palette loader, palette interpolation, and image-derived palettes.	Knn clustering code from: www.geeksforgeeks.org/image-segmentation-using-k-means-clustering/ The palette is the list of colour pallets combined from www.github.com/Experience-Monks/nice-color-palettes , www.github.com/thiagodnf/color-palettes/blob/master/data/palettes.json **OpenCV:** for resizing images, Canny edge detection **ChatGPT:** to improve reality(generate docstring, grammar correction)
P. Johari	**Modified Depth Based Style Transfer**: This feature integrates depth information into the style transfer process, ensuring that the stylization respects the depth structure of the content image. This is achieved by using a depth estimation model (Depth-Anything-V2-Small-hf) to generate depth maps and incorporate depth loss into the optimization process.	Implemented the integration of depth maps into the style transfer process, the depth loss function.	I used the Depth-Anything-V2-Small-hf model from the Hugging-Face library to generate depth maps [YKH*24]. Base code of Style transfer implementation from Assignment 3 [GEB16]. Inspired by [IM22] for the depth consistency loss function. **ChatGPT:** to improve readability (generate docstring, grammar correction)
P. Johari	**Mulit-Plane Image Based Depth Style Transfer**: The image is divided into n depth layers using depth maps. Each layer is stylized independently with decreasing style strength as depth increases. This creates a more realistic and depth-aware stylization effect.	Developed the idea of splitting the image into depth layers and applying style transfer with varying strengths. This includes the functions generate_mip_layers and reconstruct_mip_image and the respective logic behind the idea.	Got the idea for varying strength [KKM20]. I used the Depth-Anything-V2-Small-hf model from Hugging-Face library to generate depth maps [YKH*24]. Base code of Style transfer implementation from Assignment 3 [GEB16]. **ChatGPT:** to improve readability(generate docstring, grammar correction)
O. Milchi	**Style Mixing**: Style transfer based on two separate images with the possibility to configure the weight of each image in the final transferred style. Added option for a subtle focus on the content image.	Modified style loss to take into account two merged style images. Implemented channel attention and applied it to the content features in the content loss. Integrated Style Mixing in UI.	Style transfer implementation taken from Assignment 3 [GEB16]. Mixed Style Transfer inspired from [Fro21]. Channel Attention inspired from [HZL19]. Content and style images sourced from https://images.google.com/.
O. Milchi	**Color Palette Transfer**: Transfer the colour palette of one image to another image. The result will have the same content as the input, but its colours will correspond to the style image.	Implementation of an algorithm that transfers the colours of one image to another. Integration of Color Palette Transfer in UI.	Implementation based on [RAGS01]. Content and style images sourced from [RAGS01].

Continued on next page

Table 1: The work distribution table for our project. (Continued)

M. Smink	**Text-Prompt Style Transfer**: Style transfer of whole image based on user text prompt using three pre-trained models: a CLIP text encoder, a pre-trained FFN transforming text embeddings into style embeddings, and a pre-trained U-shaped CNN applying style embeddings to the image.	Integration of pre-trained models into the pipeline that applies style to an input image given a text prompt. Integration of components with others.	Inspired by [SJN*24]. Pre-trained CLIP encoder from [RKH*21]. Pre-trained FFN model from [SJN*24]. Model architecture for CNN from [GLK*17]. Pre-trained weights for CNN from [SJN*24]. Test images from [KY22].
M. Smink	**Guidance Through Semantic Segmentation: Localized Text-Prompt Style Transfer**: Masked style transfer based on user text prompt for location. Creates segmentation mask from text prompt using pre-trained open-set object detector and visual segmentation model. Merges the original and style images using edge smoothing.	Integration of pre-trained models into the pipeline to create segmentation masks based on text prompts. Merging of original and styled images. Implementation of edge smoothing during merge. Integration of segmentation masking in other components.	Drew inspiration from [WZBW23]. Pre-trained GroundingDINO open-set object detector from [LZR*23]. Pre-trained SegmentAnything visual segmentation model from paper [KMR*23]. Used package OpenCV for Gaussian Filter used in edge smoothing.
M. Smink	**Emoji Texture Text-Prompt Style Transfer**: Augments text style transfer by adding a style-specific texture mask. Creates style texture mask by first getting an emoji silhouette for text prompt using a font and a generative language model, pre-trained on a Text2Emoji dataset. Then, the silhouette to an empty mask as a sliding kernel according to user-defined step size. A stylized texture mask is used to merge the original image and a (masked) style transferred image to add stylized texture to the transfer.	Integration of pre-trained T5-Base model into the pipeline to generate an emoji based on text prompts. Creation of emoji silhouette from emoji using Noto-Color-Emoji font. Creation of style texture mask from emoji silhouette using a sliding kernel anchored at the different corners. Merging of original and styled image using style texture mask. Merging of original and locally styled image using style texture mask. Integration into other components.	Pre-trained weights & model architecture for T5-Base [RSR*23] emoji generation model from paper [PWL*23]. Accessed emoji model using HuggingFace's transformers library. Emoji font file found in github googlefonts/noto-emoji repository. Used package OpenCV for Gaussian Filter used in merging. Used package PIL to transform text emoji into an image.
M. Smink	**Video Style Transfer**: Done by running individual frames through an image-based style transfer pipeline. Option to add cross-dissolved frames to improve coherency. Option to slow down or speed up video for closer examination.	Entire video pipeline. Cross-dissolve of frames with OpenCV function. Integration of user choices in frame blending and video speed. Integration with other components.	Test data was obtained from github repository attached to [Smi16] and from personal cameras. Used package OpenCV for video writing and the addWeighted function used in cross-dissolving.