Artistic Style Transfer

Team: 10**, Email:** *mohamedshawky911@gmail.com*

Mohamed Shawky Zaky `Section:2,BN:16` | Remonda Talaat Eskarous `Section:1,BN:20` | Yosry Mohamed Yosry `Section:2,BN:35` | Amr Ahmed Abdel-Baqy `Section:2,BN:3`

***Abstract*— In this work, we will discuss our implementation of artistic style transfer using texture synthesis. We will also show some results and limitations of our implementation, as well as some performance analysis.**

I. INTRODUCTION

Briefly, the problem of style transfer is applying the style (color, texture, ...etc) of a certain image to another one. Generally,the style image is a painting of some sort and the content image is any other image. In our implementation, we used the method proposed in Style-Transfer via Texture-Synthesis by Google Research.

II. OUR IMPLEMENTATION

Edge segmentation techniques are used to get the segmentation mask of the input content.

1) Blurring Edges: This technique is very simple. It's just getting the content edges by Sobel edge detector and blur those edges. Although, this technique is very simple, but it results in a great content fusion effect with our current setting.

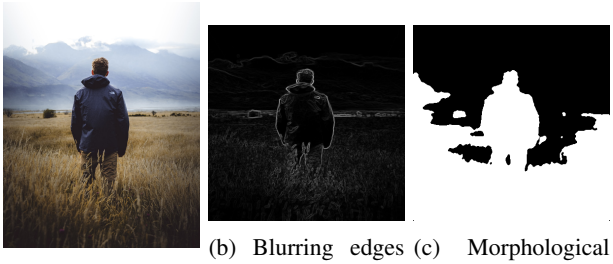
2) Morph Chane Vese: This technique uses histogram equalization with morphological Chane Vese method. The results are very good, however it's very sensitive to lighting conditions which can totally ruins output.

3) Convex Hull: This technique is better with different lighting conditions. However, it suffers from serious localization problems. It detects the important content, but doesn't fit it exactly.

A. COLOR TRANSFER

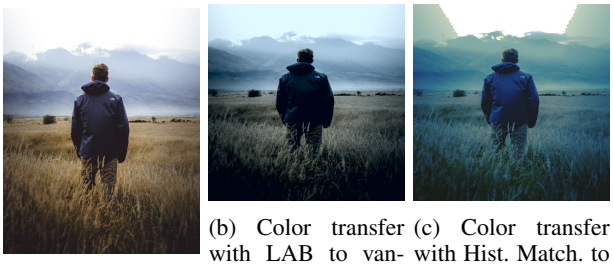
1) LAB Color Space: This technique performs color transfer source image to target image using some stats extracted from both images in the LAB color space.

2) Histogram Matching: This technique performs color transfer from source image to target image using cumulative distribution of both images.



(a) Content (b) Blurring edges with small sigma (c) Morphological Cane-Vese

Fig. 1: Some segmentation outputs.



(a) Content (b) Color transfer with LAB to van-goch style (c) Color transfer with Hist. Match. to van-goch style

Fig. 2: Color transfer results.

B. Image Denoise

1) Denoising using Edge-aware Domain Transform: This technique performs edge-aware denoising (smoothing) on an image using domain transformation and performs 1D smoothing along all rows then along all columns.

C. LEARNING ALGORITHM INPUTS

The inputs to our algorithm is simply a content and a style image, however to enforce content in the output image, we have to use some kind of segmentation masks to help

(a) Noisy image (b) Our denoise (c) Skimage denoise

Fig. 3: Denoising results.

us fuse the important components of the content image, in order to preserve the original content.

D. LEARNING PROCESS

We followed the original paper in the majority of the implementation.

1) PROCESS INITIALIZATION: First, color transfer is performed from the style to the content image. After so, the Gaussian pyramids are built to account for multi level style transfer.
Then, the output X is initialized to be the content image with some strong noise avoid learning collapse.

2) LEARNING ITERATIONS: The algorithm iterates over all resolution scales and all patch sizes within them. At each step the following procedures are done:

- Patch matching: Each patch from the output X is matched with a single style patch using nearest neighbours, after performing Principal Component Analysis (PCA) on each patch to improve performance and RAM usage.

- Robust aggregation: The algorithm, then, tries to apply the style patches on the corresponding output patches iteratively using Iteratively Re-weighted Least Squares (IRLS). This step is also named style fusion, as it adds the style texture to the output.

- Content fusion: The important components of the content image are added to the output, using the color transferred content and the segmentation masks.

- Color transfer: The output X colors are transferred once more to the style image to improve output color quality.

- Denoise: The output is then denoised to remove any artifacts or sharp edges from the output image.

Fig. 4: Some good stylization results, the left column is the content image, the middle one is the style image and the right one is the output.

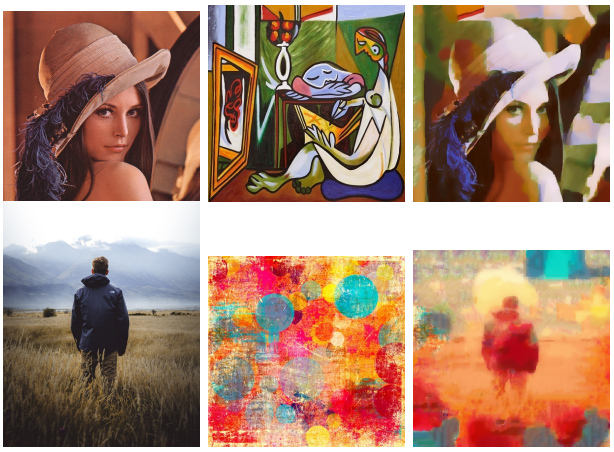


Fig. 5: Some failure cases, the left column is the content image, the middle one is the style image and the right one is the output.

III. EXPERIMENTAL RESULTS

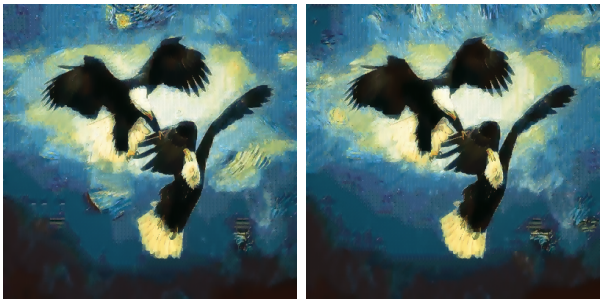
In this section, we will discuss the results of the implemented algorithm and show some failure cases.

Figure 4 represents a set of samples that yields a good stylization results. Here, we see that the content is well preserved and the style is correctly captured by the algorithm. This is mainly due to the clarity of the style pattern, which makes the algorithm captures it correctly.

Meanwhile, Figure 5 represents the main causes of failure of our algorithm. In the first sample, there are two problems. One of which is the face in the content image. The face details are distorted, as we use edge segmentation techniques, which sometimes doesn't work very well with faces, which require face segmentation. The second problem is that the style image doesn't have a consistent pattern, so we notice some randomness in the generated image.
In the second sample, the content fusion fails, so we see that the style is dominating, however this problem can be solved somehow by increasing the weight of the content in content fusion.

Having discussed some success and failure cases, we already have a lot of parameters to play with, most of them have a great impact on the output image.
Let's discuss the effect of number of patches on the image, as it's the most important parameter on which the results vary. In Figure 6, we notice that as the number of patches decrease the output suffers from serious artifacts and random style applying due to large patch size. However, as the number of used patch sizes increase, the output become more coherent.

Fig. 6: The effect of patches size reduction. The used patch sizes are 5, 4, 3, 2 and 1 from top to bottom.



(a) 3 algorithm iterations (b) 10 algorithm iterations

Fig. 7: Comparison with number of iterations.

IV. PERFORMANCE ANALYSIS

The choice of our parameters also has an impact of stylization time, the stylization time can vary from $20sec$ to $170sec$ or more in some cases depending on the image size, the number of iterations and also the specifications of the machine.
A set of default parameters are chosen to yield good stylization results in most cases, the running time of stylizing one image with this set of parameters is $46sec$ on $Intel\ i5\ 6600K$, which is great, as the implementation from original paper running on nearly the same settings takes $25:50Sec$ on $Intel\ Xeon\ E5$-1650. The implementation also runs comfortably on $8GB$ of RAM on a $400X400$ image depending on the settings.

Our proposed parameter set:

1) Image Size: $400X400$

2) Depth of Pyramid: 3

3) Patch Sizes: $[33, 21, 13, 9]$

4) Sub-sampling Gaps: $[28, 18, 8, 5]$

5) IRLS Iterations Number: 3

6) Algorithm Iterations Number: 3

7) Robust Statistic Value: 0.8

8) Content Weight: 5.0

9) Segmentation Method Used: $BlurringEdges$

V. WORKLOAD DIVISION

1) Mohamed Shawky Zaky: Implemented main code structure, training loop and $IRLS$ solver for robust aggregation. Created final report with analysis and hyperparameter tuning.

2) Remonda Talaat Eskarous: Implemented the segmentation algorithms and color transfer with histogram matching. Created the application user interface.

3) Yosry Mohamed Yosry: Implemented the Edge-aware denoising algorithm. Improved the denoising quality and performance to surpass the built-in functions.

4) Amr Ahmed Abdel-Baqy: Implemented color transfer using LAB space color and PCA. Experimented with different feature representations like: DCT and SIFT.

VI. CONCLUSION

In our project, we experimented with many paths to implement artistic style transfer with texture synthesis without the aid of neural networks. We managed to get the thing up and running,however there are some cases that the algorithm totally fails in, especially in case of faces or unstructured style. So, in this work, we discussed our approach to the algorithm, the results that come out and some failure cases, along with our interpretation to the behaviour of the algorithm and different parameters.

REFERENCES

[1] Elad, Michael, and Peyman Milanfar. "Style Transfer Via Texture Synthesis." IEEE Transactions on Image Processing 26.5 (2017): 2338–2351. Crossref. Web.

[2] Eduardo S. L. Gastal and Manuel M. Oliveira. 2011. Domain transform for edge-aware image and video processing. ACM Trans. Graph. 30, 4, Article 69 (July 2011), 12 pages. DOI: https://doi.org/10.1145/2010324.1964964

[3] Saha, Sanjoy Das, Amit Chanda, Bhabatosh. (2006). An Automatic Image Segmentation Technique Based on Pseudo-convex Hull. 4338. 70-81. 10.1007/11949619

[4] Kwatra, Vivek Essa, Irfan Bobick, Aaron Kwatra, Nipun. (2005). Texture optimization for example-based synthesis. ACM Trans. Graph.. 24. 795-802. 10.1145/1186822.1073263